Optimización de la Gestión de Inventarios en cadenas de suministros de alimentos mediante Aprendizaje por Refuerzo

Camilo Aguilar León

Universidad de los Andes, c.aguilarl@uniandes.edu.co

Este trabajo se centra en mejorar las decisiones de las empresas de alimentos que compran productos agrícolas para satisfacer la demanda de los consumidores finales, un proceso conocido como la 'primera milla' de la cadena de suministro. Dado que estas decisiones se ven afectadas por factores dinámicos como la demanda, los precios, y la disponibilidad de productos, es crucial una toma de decisiones óptima para maximizar las ventas y minimizar los costos relacionados con el transporte, la pérdida de inventario y otros gastos. En este contexto volátil, propongo una serie de modelos basados en aprendizaje por refuerzo, especialmente utilizando Q Learning, para captar la naturaleza estocástica de la cadena de suministros. Estos modelos están diseñados para ayudar a las empresas a tomar decisiones informadas y eficientes. Se exploran diferentes enfoques, variando en complejidad, que integran variables clave como la demanda, los precios de compra, las cantidades disponibles, los costos de transporte y la gestión de inventarios en cada período. El objetivo es maximizar la recompensa a corto y largo plazo. La implementación de estos modelos requiere de estrategias creativas para adaptar la información disponible a los algoritmos de aprendizaje por refuerzo. Tras un entrenamiento adecuado, los modelos demuestran su capacidad para aprender patrones que maximizan la eficiencia operativa. La comparación con políticas de manejo de inventarios tradicionales muestra una mejora significativa, evidenciando la eficacia de estos modelos en la toma de decisiones estratégicas, considerando tanto el corto como el largo plazo.

Contenido

1 Introducción

El sector de alimentos, crucial para la economía y el bienestar social, se enfrenta a desafíos únicos debido a su complejidad inherente y a la incertidumbre constante en la demanda. Esta incertidumbre afecta significativamente la vida útil de los productos, dada su alta perecibilidad (Gutiérrez, 2021) y en general a la toma decisiones de las partes de la cadena. A esto se suma la falta de un intercambio eficiente de información entre los diferentes agentes de la cadena de suministro, lo que impide un control óptimo y dinámico de las operaciones (Gutiérrez, 2021).

En este contexto, las cadenas de suministro de alimentos cortas están ganando atención por su capacidad para generar beneficios sociales, económicos y ambientales, marcando un contraste con los enfoques más tradicionales (EU Food Information Council, 2021). Sin embargo, las expectativas sobre el valor aportado por las cadenas de suministro son cada vez mayores, especialmente en un entorno marcado por la volatilidad del mercado, el aumento de los costos y la aceleración de la digitalización (Kaizen Institute, 2021).

La agilidad se ha convertido en un aspecto fundamental para las cadenas de suministro modernas. La capacidad de responder rápidamente a cambios en la demanda del cliente, la competencia o interrupciones en el suministro es esencial para mantener la competitividad en el mercado (IBM, 2021). No obstante, los métodos tradicionales de toma de decisiones, particularmente en la optimización y ubicación del inventario, han demostrado ser ineficientes frente a cómo funciona actualmente el comercio (Cozowicz, 2021).

Este trabajo propone un modelo innovador diseñado para manejar eficazmente la incertidumbre inherente en las cadenas de suministros de alimentos. El objetivo es facilitar la toma de decisiones óptimas respecto a la adquisición de productos, considerando una serie de variables críticas como la perecibilidad de los productos, la demanda fluctuante, precios y cantidades inciertas, costos de transporte, y la gestión eficiente del inventario.

Para simplificar el análisis y reducir la complejidad inherente a este tipo de cadena de suministro, el modelo se centra en la 'primera milla', es decir, desde los proveedores hasta el depósito. En cuanto a las decisiones de transporte, se asumen viajes directos de ida y vuelta sin escalas intermedias. Respecto a la demanda y las ventas, se consideran como realizadas directamente en el depósito. Así, el desafío se centra en la toma de decisiones de compra basada en la información recogida de la cadena de suministros, analizada periodo a periodo dentro de un horizonte temporal específico.

La complejidad del problema radica en la necesidad de integrar y balancear múltiples factores. Decisiones basadas exclusivamente en la demanda sin considerar los costos de transporte, o en los precios sin tener en cuenta la demanda, pueden resultar en utilidades subóptimas. Además, la incertidumbre constante en la cadena de suministros agrega un nivel adicional de dificultad,

obligando a los agentes a tomar decisiones sin una comprensión completa de lo que ocurrirá en periodos futuros. Esta incertidumbre subraya la necesidad de un enfoque que no solo sea ‘greedy’ sino que también anticipe y planifique para futuras eventualidades.

Para abordar los desafíos presentados en la cadena de suministros de alimentos, este trabajo propone la creación de un entorno virtual que simule todos los elementos de una cadena de suministro. Este entorno facilitará la interacción con distintos agentes basados en Aprendizaje por Refuerzo (RL, por sus siglas en inglés). Estos agentes, diseñados para procesar los datos de la cadena de suministro de una manera especifica, tomarán decisiones diversas, permitiendo así el análisis de varias alternativas.

Los agentes se fundamentan en dos técnicas de RL: Q Learning y DQN (Deep Q Network). DQN es una evolución de Q Learning que integra el aprendizaje profundo para superar algunas limitaciones de su predecesor. Mediante un proceso iterativo, cada agente aprende interactuando con el ambiente diseñado, capturando los patrones inherentes a este.

La efectividad de estos agentes se analizó comparándolos con políticas de manejo de inventarios implementadas de manera 'greedy'. Los resultados demostraron una mejora significativa en la toma de decisiones por parte de los agentes de RL frente a estas políticas convencionales, aprendiendo a optimizar sus decisiones a lo largo de los episodios. Los hallazgos clave de esta investigación se resumen de la siguiente manera:

- **Capacidad de Captura de Dinámicas Complejas:** Las técnicas de Q Learning y DQN demuestran su eficacia para manejar la complejidad de sistemas como las cadenas de suministros. Sin embargo, hay que considerar que a mayor complejidad del sistema implica un aumento en el costo computacional para entrenar estos agentes. Es importante resaltar igualmente que después de entrenar, la toma de decisiones sucede casi de manera instantánea para ambas técnicas.

- **Mejora sobre Políticas Tradicionales de Manejo de Inventarios:** Al comparar con políticas tradicionales de manejo de inventarios como R,Q y S,S, los agentes de RL más complejos muestran una mejora sustancial en los resultados.

- **Toma de Decisiones Multifactorial:** Los agentes no solo reaccionan a situaciones de bajo inventario o alta demanda, sino que también aprovechan oportunidades como precios bajos y altas disponibilidades de producto.

- **Visión a Largo Plazo en la Toma de Decisiones:** Los agentes consideran cómo sus decisiones afectan tanto el periodo actual como los futuros. Las decisiones están

interconectadas, de modo que el estado al que llega un agente y la decisión que debe tomar en esta situación afecta a la toma de decisión inicial que llevo a ese estado.

- **Eficacia de la Discretización de Datos:** La discretización de datos continuos no afecta negativamente el rendimiento de manera significativa, siempre y cuando se mantenga una granularidad adecuada que represente distintos estados de manera efectiva.

En conclusión, este trabajo aspira a contribuir a la tecnificación y a la implementación de la inteligencia artificial en las cadenas de suministros de alimentos, a través de la construcción de agentes de toma de decisiones. Si bien estas técnicas requieren grandes volúmenes de datos para el aprendizaje, su capacidad para facilitar decisiones rápidas y precisas las hace ideales en contextos donde la velocidad es prioritaria. A pesar de que el entrenamiento inicial puede ser intensivo en términos de recursos y datos, una vez implementados, estos modelos tienen la capacidad de seguir aprendiendo y adaptándose en operación real con un costo marginal mínimo. Este proceso, conocido como 'aprendizaje en línea' ('online learning'), es particularmente valioso en entornos cambiantes que requieren la integración continua de nuevos datos (Hugging Face, n.d.).

2 DESCRIPCIÓN GENERAL

2.1 Objetivos

El propósito principal de esta investigación es desarrollar un modelo de aprendizaje por refuerzo que integre con eficacia los componentes estocásticos de las cadenas de suministros, con el fin de facilitar la toma de decisiones eficientes e informadas. Para lograr este objetivo general, el trabajo se enfoca en varios objetivos específicos. Primero, se identificarán y analizarán los elementos clave y la dinámica temporal de una cadena de suministro para comprender su evolución y complejidad. A continuación, se diseñarán y desarrollarán modelos de aprendizaje por refuerzo capaces de procesar de manera efectiva los datos de la cadena de suministros, proporcionando un soporte crucial para la toma de decisiones estratégicas. Además, se implementarán los modelos propuestos de manera que el trabajo sea replicable y verificable en diferentes contextos. Otro objetivo clave es validar el aprendizaje y el rendimiento de estos modelos en diversos escenarios, comparándolos con un modelo de optimización determinístico, que dispone de todos los datos de todos los periodos, y con las políticas tradicionales de manejo de inventarios. Finalmente, se identificarán las áreas en las que estas técnicas de aprendizaje por refuerzo sobresalen al ser aplicadas en las cadenas de suministros, destacando sus ventajas y áreas de mejora.

2.2 Antecedentes

En el ámbito de las cadenas de suministro, la literatura abarca una amplia gama de modelos decisionales que abordan problemáticas de diversa complejidad. Un ejemplo notable es la investigación de Gómez et al. (2023), que ha sido una fuente de inspiración significativa para este trabajo. Este estudio destaca por su enfoque innovador, que integra la gestión dinámica de inventarios de múltiples productos perecederos a lo largo de varios períodos, con el enrutamiento de adquisiciones caracterizado por demandas fluctuantes, precios de compra variables y suministro incierto. Este trabajo y sus formulaciones son bastantes útiles para el diseño e implementación de las técnicas de RL, utilizando definiciones y notaciones alineadas con este enfoque. Se realizaron ajustes específicos en este entorno para facilitar la implementación efectiva de los agentes de RL.

Ahora bien, la aplicación de la inteligencia artificial, especialmente el aprendizaje por refuerzo, en la optimización de cadenas de suministros es un campo relativamente nuevo con escasos estudios formales. Sin embargo, los avances recientes en hardware y tecnología han propiciado un incremento en la implementación de modelos complejos de RL.

En esta línea, Hubbs et al. (2019) introdujeron 'or-gym', un entorno de simulación especializado para abordar problemas de investigación de operaciones para casos como el de cadena de suministro y manejo de inventarios. Este entorno simula escenarios realistas, lo que permite el entrenamiento y validación de modelos de RL en contextos que reflejan los retos actuales. Además, este trabajo propone formulaciones detalladas para varios problemas de esta área, como el de la cadena de suministros.

Por su parte, Hutse (2019) contribuye al ámbito incrementando la complejidad de los problemas al incorporar variables como tiempos de entrega y espacios de acción continuos. Este estudio explora técnicas avanzadas de RL, como DQN y DDPG, ofreciendo un enfoque más preciso en la actualización de recompensas y en la estimación de estados y acciones. La utilización de la librería GYM de Python en este trabajo facilita notablemente su replicabilidad.

En un enfoque similar, van Helsdingen (2022) desarrolla y evalúa agentes de Q Learning y DQN para las cadenas de suministros, enfocándose en aspectos críticos como el ajuste de hiperparámetros. Este estudio también incluye una comparación de estos agentes con métodos de simulación comerciales, utilizando datos reales para validar sus hallazgos.

Finalmente, van Hasselt (2010) aportó significativamente al campo con la introducción de Double Q Learning, una variante de Q Learning diseñada para entornos de RL en espacios continuos. Esta metodología, que utiliza dos redes de valor para minimizar la sobreestimación en las acciones, ha demostrado ser más robusta y eficiente en comparación con los métodos tradicionales de Q Learning, especialmente en contextos estocásticos como las cadenas de suministro.

A pesar de estos avances, la mayoría de los estudios existentes se han centrado en la formulación inicial y básica de agentes de RL para problemas de cadenas de suministros, dejando un amplio margen para explorar y maximizar el potencial de estas técnicas avanzadas en investigaciones futuras.

3 DISEÑO Y ESPECIFICACIONES

3.1 Definición del problema

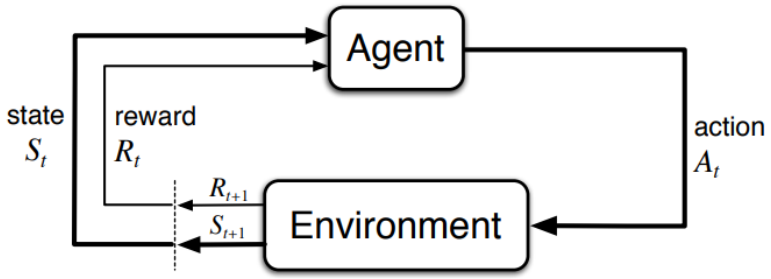
El trabajo que se desarrolla en esta tesis aborda una problemática intrínsecamente vinculada con la operatividad de las cadenas de suministro alimenticias, enfocándose en la etapa inicial o "primera milla". Específicamente, se examina el rol de un intermediario situado en un punto medio de la cadena, cuya función es adquirir alimentos de agricultores y otros productores para luego venderlos y obtener beneficios. En este contexto, se plantea un escenario donde múltiples proveedores ofrecen variados productos alimenticios, y el objetivo es maximizar la rentabilidad derivada de la compra y venta de estos productos.

Cada proveedor posee un catálogo específico de productos, y se recopila periódicamente información referente al precio y disponibilidad de cada ítem. Por otro lado, el intermediario, quien es el foco principal de este estudio, administra un almacén donde guarda el inventario adquirido. Este inventario debe satisfacer la demanda de productos, que se conoce al inicio de cada ciclo y que varía a lo largo del tiempo, considerando además la tasa de perecimiento fija de los alimentos. El precio de venta se mantiene constante durante todos los periodos y siempre es superior al de compra, que puede variar. Por otro lado, existen costos asociados a no satisfacer la demanda o tener inventarios sobrantes. De esta manera, el objetivo general es maximizar las ventas y minimizar el inventario sin afectar decisiones futuras.

Adicionalmente, la ubicación de los proveedores respecto al almacén es fija, pero se incurre en costos de transporte directamente relacionados con la distancia y la cantidad de vehículos necesarios, limitados por su capacidad de carga. Este problema se enfoca en un horizonte temporal discreto y fijo, buscando maximizar las utilidades considerando todos los factores mencionados. El intermediario, enfrentando la incertidumbre de la información futura, debe tomar decisiones de compra cada periodo, determinando cuánto y qué productos adquirir de cada proveedor.

3.2 Formulación del problema

En el contexto de este trabajo, la formulación del problema de la cadena de suministro alimentaria como un Proceso de Decisión de Markov (MDP) ofrece un marco robusto para aplicar técnicas de aprendizaje por refuerzo, como se explica en "Reinforcement Learning: An Introduction" de Sutton and Barto [http://www.incompleteideas.net/book/RLbook2020.pdf]. Este enfoque es particularmente pertinente dada la naturaleza secuencial y la interdependencia entre decisiones y recompensas en la gestión de la cadena de suministro.



(cita aquí http://www.incompleteideas.net/book/RLbook2020.pdf

En este MDP, el agente (el intermediario en la cadena de suministro) interactúa con un ambiente que cambia en respuesta a sus decisiones. Cada acción tomada por el agente se basa en el estado actual del ambiente, y como resultado de estas acciones, el ambiente proporciona una recompensa que refleja la efectividad de la decisión. El MDP se define por una tupla (S, A, P, R), donde S representa el conjunto de estados posibles, A el conjunto de acciones disponibles, P la función de transición de estados, y R la función de recompensa.

Antes de detallar los elementos específicos de la tupla que constituye el MDP, resulta esencial contextualizar y describir los atributos adicionales del ambiente en el que opera el sistema de cadena de suministro. Primordialmente, este entorno se caracteriza por funcionar en un horizonte de tiempo finito y discreto, representado por un conjunto de periodos, denotado como T, que comprende una serie de periodos temporales fijos y secuenciales.

Además, el sistema incluye un conjunto de proveedores, señalado como S, y una diversidad de productos, indicados como P. Cada proveedor s en el conjunto S está localizado a una distancia específica del depósito, identificada como l_s. Esta distancia es crítica, ya que influye directamente en los costos y la logística de transporte. También, se denota al conjunto de productos de un proveedor como P_s y al conjunto de proveedores que ofrecen un producto como S_p. Por otra parte, cada producto p en el conjunto P se caracteriza por una tasa de perecimiento, denotada como ϕ, y un precio de venta, x_p. De este precio de venta surgen los costos relacionados con no cumplir la demanda k_p o por mantener el inventario h_p. Estos valores son un porcentaje del precio de venta y pueden tener varios significados en el contexto de las cadenas de suministros por lo que fueron

incluidos para darle más flexibilidad al modelo y también para apoyar al diseño de los modelos. Estos factores son vitales para la gestión eficiente del inventario y la estrategia de precios.

En lo que respecta a la logística de transporte, cada vehículo utilizado para el traslado de alimentos posee una capacidad máxima, Q, y opera a una velocidad v. También se considera un tiempo de carga fijo, τ, y un costo de transporte, c, que se calcula por unidad de tiempo. Estos parámetros del vehículo son fundamentales para planificar las rutas de entrega y optimizar los costos operativos. Por lo que es necesario modelar también la cantidad de vehículos que se usan para recoger los productos de cada proveedor como n_{st} Todas estas características –distancias, tasas de perecimiento, precios de venta, y especificaciones de los vehículos– son constantes a lo largo de los periodos y establecen las condiciones iniciales del ambiente.

Ahora bien, con estos elementos intrínsecos al ambiente definidos, se puede detallar cada uno de los elementos del MDP en el contexto específico de nuestro problema:

- **Estados (S):** El estado es una tupla que incluye el inventario actual I_{pt}, la demanda d_{pt}, el precio de compra p_{spt}, y la cantidad disponible q_{spt} para cada proveedor $s \in S$, para cada producto $p \in P$ y para cada periodo $t \in T$. El inventario es directamente influenciado por las decisiones del agente, mientras que los otros componentes son determinados aleatoriamente por el ambiente.
- **Acciones (A):** Las acciones disponibles para el agente son definidas por la cantidad a comprar z_{spt} de cada producto $p \in P$, de cada proveedor $s \in S$ durante cada periodo $t \in T$. Estas acciones son cruciales para la actualización del estado y la gestión del inventario.
- **Función de Transición (P):** La función de transición modela la probabilidad de pasar de un estado a otro dada una acción específica. En este caso, la demanda, el precio de compra y las cantidades disponibles aportan elementos aleatorios a esta función de transición que no dependen ni del estado actual ni de la acción. Por otro lado, el inventario depende únicamente del estado actual y de la decisión por lo que esta parte es determinista.
- **Función de Recompensa (R):** La función de recompensa en este MDP es un poco distinta a la generalización que se suele hacer en el sentido de que depende únicamente del estado actual y la acción tomada, y no del estado resultante. Esto refleja la naturaleza inmediata de la recompensa obtenida de la decisión de compra, como las ganancias o pérdidas realizadas en un periodo dado.

Además, es crucial resaltar la naturaleza continua y potencialmente, pero poco probable, infinita de las variables involucradas (inventario, precios, demanda, disponibilidad). Aunque en la práctica, estas variables operarán dentro de un rango específico, su naturaleza continua y no finita agrega una capa de complejidad al problema. Además, el modelo incluye características fijas como la distancia de los proveedores al depósito, la tasa de perecimiento de los productos, los precios de

venta, y las especificaciones de los vehículos de transporte, que son constantes a lo largo del tiempo, pero influyen en las decisiones y recompensas.

De toda la formulación descrita anteriormente pueden surgir casos en los que los estados son inconsistentes por lo que se establecen ciertas restricciones en el ambiente que permiten modelas estas reglas que se deben cumplir para validar la factibilidad de las acciones tomadas y los estados del ambiente.

$$I_{pt} = I_{pt-1}(1-\phi) + \sum_{s \in S_p} z_{spt} - \min\left(d_{pt}, I_{pt-1}(1-\phi) + \sum_{s \in S_p} z_{spt}\right), \forall p \in P, t \in T | t > 0 \quad (1)$$

$$\sum_{p \in P} z_{spt} \leq Q n_{st}, \qquad \forall s \in S, t \in T \quad (2)$$

$$z_{spt} \leq q_{spt}, \qquad \forall s \in S, p \in P, t \in T \quad (3)$$

$$z_{spt} \geq 0, \qquad \forall s \in S, p \in P, t \in T \quad (4)$$

El modelo incorpora varias restricciones clave para garantizar su operatividad y realismo. La primera restricción se enfoca en la dinámica de actualización del inventario en cada periodo y en cómo se atiende la demanda utilizando las cantidades disponibles de cada producto. La segunda restricción asegura coherencia entre la cantidad de vehículos necesarios para el transporte desde un proveedor y la cantidad total de producto adquirido. La tercera restricción, no menos importante, establece que la cantidad de producto comprada no puede exceder la cantidad disponible en el proveedor. Finalmente, se impone que la cantidad de producto a comprar debe ser siempre un valor positivo. Estas restricciones, combinadas con la formulación previamente descrita, permiten construir adecuadamente tanto el ambiente como el problema en cuestión.

En cuanto a la función de recompensas para el modelo, se define de la siguiente manera:

$$R(s_t, a_t) = \sum_{p \in p} \sum_{s \in S_p} \left(x_p z_{spt} - \max\left(k_p\left(d_{pt} - z_{spt}\right), h_p\left(z_{spt} - d_{pt}\right)\right) - p_{spt} z_{spt}\right) - \frac{c}{v} \sum_{s \in S} l_s n_{st} \quad (5)$$

Esta ecuación tiene como objetivo maximizar las ganancias obtenidas de la venta de productos, descontando los costos asociados a mantener el inventario, no suplir la demanda, la compra de productos y el transporte de estos. La función de recompensas es crucial para modelar la eficacia de las acciones tomadas por el agente dentro del ambiente, evaluando el rendimiento de estas decisiones en términos de rentabilidad. En este sentido se busca un agente que logre maximizar estas recompensas individualmente para cada periodo. Esto no significa que el agente tomara decisiones únicamente pensando en el periodo en el que esté tomando la decisión. Esta búsqueda de las decisiones que maximicen las recompensas durante todos los periodos se representa como la búsqueda de una política π^*, la cual define las acciones tomadas por el agente, que permita

maximizar las recompensas obtenidas durante todos los periodos como se describe en Gomez et al. (2023)

$$\pi^* = argmax_{\pi \in \Pi} \mathbb{E}\left(\sum_{t \in T} R(s_t, a_t^{\pi}) \mid S_0\right) \qquad (6)$$

Por lo que, resumiendo el problema y la formulación en tanto a este, se busca obtener un agente que tenga la política optima que maximice las recompensas obtenidas a través de los episodios. Este agente deberá desarrollar maneras de aproximarse a esta política optima de distintas maneras logrando en muchos casos soluciones poco adecuadas para este ambiente.

4 DESARROLLO DEL DISEÑO

En el proceso de diseño de este sistema, se adoptó un enfoque multifacético para crear agentes con propósitos variados. Inicialmente, se desarrollaron dos agentes inspirados en políticas de manejo de inventarios existentes en la literatura. Estos agentes, fundamentados en decisiones 'greedy' o codiciosas, establecen políticas base que actúan como referencia para evaluar y mejorar las estrategias de los agentes subsecuentes.

Además, se diseñó un modelo de optimización entero mixto. Este modelo, que se alinea con la formulación del MDP previamente descrita, es esencial para determinar una cota superior teórica, representando el ideal o el mejor resultado posible que un agente pudiera alcanzar en un escenario específico. Este enfoque ofrece un punto de comparación valioso para medir la efectividad de las políticas implementadas por los agentes.

En lo que respecta a los agentes diseñados específicamente para abordar y optimizar el problema, se desarrollaron tres modelos distintos: dos utilizando el aprendizaje Q-Learning y uno empleando Deep Q-Networks (DQN). Cada uno de estos agentes se diferencia en términos de la técnica utilizada y su formulación específica, permitiendo un análisis detallado y comparativo de sus respectivas eficacias y particularidades.

4.1 Recolección de Información

Las decisiones de diseño fundamentales para la construcción de los diversos agentes y modelos se centraron en la utilización y el tratamiento de los datos empleados para instanciar el ambiente del problema. La naturaleza de estos datos fue determinante en la aplicación de distintas técnicas de diseño e implementación. Una parte significativa de estos datos proviene del trabajo realizado por Gómez et al. (2023), quienes proporcionaron información detallada y específica sobre el ambiente que utilizaron en su investigación.

Para ciertos elementos, especialmente aquellos relacionados con el ruteo, se emplearon datos distintos o con pequeños cambios. Por ejemplo, se realizaron cálculos adicionales para determinar variables necesarias, como la distancia. La disponibilidad y el procesamiento de estos datos fueron cruciales para tomar decisiones clave en el diseño, tales como la discretización de variables continuas, la formulación de penalizaciones para los agentes y otros aspectos relevantes.

4.2 Políticas de manejos de inventarios

En el desarrollo de las políticas de manejo de inventarios, se optó por diseñar dos agentes basados en las conocidas políticas deterministas R,Q (también llamada s,Q) y S,s. La política R,Q implica realizar un pedido de tamaño Q cuando el nivel de inventario de un producto cae por debajo del umbral R. Por otro lado, la política S,s rellena el inventario hasta alcanzar un nivel S cuando este desciende por debajo de una cantidad crítica s. Estas políticas, a pesar de su simplicidad, son ampliamente reconocidas y utilizadas tanto en la literatura académica como en la práctica (fuente: https://repositorio.uniandes.edu.co/server/api/core/bitstreams/cd4ea271-aac3-4aca-aa5c-0fa131138e87/content).

Dado el contexto del problema definido, que contempla múltiples productos y proveedores, se decidió adaptar estas políticas a una aproximación 'greedy'. Así, para ambas políticas, la decisión de realizar un pedido se basa en el nivel actual del inventario de cada producto. Sin embargo, la selección del proveedor se realiza considerando el precio de venta de cada uno de ellos. En el caso de que la cantidad disponible en el proveedor con el precio más bajo sea insuficiente para cumplir con el pedido requerido, se procede entonces con el siguiente proveedor que ofrezca el menor precio, y así sucesivamente, hasta completar la cantidad deseada. Esta adaptación permite la implementación efectiva de estas políticas en el ambiente diseñado para el estudio.

4.3 Modelo mixto-entero de optimización

El modelo mixto-entero de optimización difiere significativamente de las políticas de manejo de inventarios previamente mencionadas. Mientras que estas últimas fueron diseñadas como agentes que interactúan con el ambiente de manera secuencial, el modelo de optimización se concibe como una herramienta comparativa y no como un agente per se. Este modelo se nutre de información simulada correspondiente a todos los periodos del estudio, incluyendo datos sobre la demanda, precios y cantidades, entre otros. A partir de esta información, el modelo es capaz de generar los resultados óptimos y máximos posibles para el escenario en cuestión.

Aunque podría parecer que este modelo no tiene una aplicación directa en la práctica, resulta ser de gran valor para comprender los puntos débiles de los agentes y para identificar áreas de mejora. Al comparar los resultados de los agentes con los del modelo de optimización, es posible discernir dónde y cómo se pueden ajustar las estrategias de los agentes para aumentar sus recompensas. Así, la formulación de este modelo de optimización se convierte en un componente crucial para el análisis y la mejora continua del sistema.

$$\max \sum_{t\in T}\sum_{p\in p}\sum_{s\in S_p}\left(x_p z_{spt} - p_{spt} z_{spt}\right) - \frac{c}{v}\sum_{s\in S} l_s n_{st} \qquad (7)$$

$$I_{pt} = I_{pt-1}(1-\phi) + \sum_{s\in S_p} z_{spt} - d_{pt}\,, \qquad \forall p \in P, t \in T | t > 0 \qquad (8)$$

$$\sum_{p\in P} z_{spt} \leq Q n_{st}, \qquad \forall s \in S, t \in T \qquad (9)$$

$$z_{spt} \leq q_{spt}, \qquad \forall s \in S, p \in P, t \in T \qquad (10)$$

$$z_{spt} \geq 0, \qquad \forall s \in S, p \in P, t \in T \qquad (11)$$

$$n_{st} \in \mathbb{Z}^+, \qquad \forall, p \in P \qquad (12)$$

$$I_{p0} = 0, \qquad \forall, p \in P \qquad (13)$$

Este Modelo de Programación Entero Mixto (MIP) representa una versión simplificada del problema, incorporando ciertos supuestos basados en las discusiones previas. Un aspecto notable en esta versión es la omisión de penalizaciones. La razón detrás de esta simplificación radica en la naturaleza de los datos utilizados para la implementación del modelo, los cuales permiten hallar una solución factible sin necesidad de aplicar penalizaciones. En esencia, esto implica que siempre existen cantidades adecuadas de productos disponibles para satisfacer completamente la demanda.

Dado este contexto, el modelo proporciona una perspectiva más específica del escenario ideal, esencial para realizar ejercicios de 'backtesting'. Este enfoque permite evaluar cómo se comportarían los agentes en condiciones óptima, ofreciendo así una referencia clara para comprender el potencial máximo de rendimiento de un agente.

4.4 Agentes Q-Learning

Tras presentar los dos agentes iniciales y el modelo de optimización que se emplearán como benchmarks para la comparación y validación de resultados, ahora enfocamos nuestra atención en la formulación de agentes que utilizan técnicas de aprendizaje por refuerzo, con especial énfasis en Q-Learning. Para adentrarnos en los detalles de esta formulación, es crucial primero entender el principio fundamental que subyace a Q-Learning, una técnica prominente dentro del aprendizaje por refuerzo.

El algoritmo de Q-Learning opera sobre la base de una estructura conocida como tabla Q, que es esencialmente una matriz que mapea pares de estados y acciones a sus correspondientes valores de recompensa esperada. En esta tabla, cada entrada refleja la recompensa estimada de ejecutar una acción específica a en un estado dado s (http://www.incompleteideas.net/book/RLbook2020.pdf). El objetivo primordial del Q-Learning es desarrollar una política que maximice estas recompensas esperadas. En un escenario ideal donde se conocieran con precisión los valores de la tabla Q para todas las combinaciones de estados y acciones, la política óptima simplemente consistiría en seleccionar, para cada estado, la acción que ofrezca la mayor recompensa esperada según esta tabla. El desafío central en Q-Learning reside en la construcción eficaz de la tabla Q, de modo que se reflejen de manera precisa las recompensas asociadas a la toma de ciertas acciones en determinados estados. Este proceso se fundamenta en la ecuación de Bellman, un principio clave en la teoría del aprendizaje por refuerzo. La ecuación se formula de la siguiente manera:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha \left[R_t + \gamma \max_{a \in A} Q(S_{t+1}, a) - Q(S_t, A_t)\right]$$

Otra formulación, que facilita su interpretación, es:

$$Q(S_t, A_t) = (1 - \alpha) Q(S_t, A_t) + \alpha \left[R_t + \gamma \max_{a \in A} Q(S_{t+1}, a)\right]$$

En estas ecuaciones, $\alpha \in [0,1]$ representa la tasa de aprendizaje, que modula la velocidad con la que el valor Q se actualiza a lo largo de los episodios. Por otro lado, $\gamma \in [0,1]$ es la tasa de descuento, la cual determina la importancia que se otorga a las recompensas futuras en comparación con las inmediatas. Este último parámetro es especialmente relevante en el contexto de las cadenas de suministro, ya que ajusta el enfoque del modelo entre maximizar recompensas inmediatas y preservar el inventario para beneficios futuros.

En resumen, la ecuación de Bellman en Q-Learning es una función iterativa destinada a actualizar el valor $\boldsymbol{Q(S_t, A_t)}$, mejorando progresivamente la política del agente. Esta actualización se orienta a asignar valores más altos a aquellas acciones que, según las estimaciones del modelo, conducen a mejores resultados en cada estado.

Dentro de este marco, comprendiendo cómo Q-Learning mejora iterativamente sus resultados, un desafío clave para su aplicación en cadenas de suministro es la modelación de la tabla Q. Como se mencionó anteriormente en la formulación del problema, tanto los estados como las acciones en nuestro contexto son continuos y no finitos. Esto implica que una tabla Q que abarque todas las posibles combinaciones de estados y acciones sería impracticablemente grande, volviendo inviable el cálculo iterativo de los valores Q para cada par estado-acción.

Para abordar este desafío, se adoptó una estrategia de discretización tanto para los estados como para las acciones. Este proceso de discretización implica definir un rango máximo y mínimo dentro del dominio de cada variable y dividirlo en k intervalos de tamaño igual. Además, se incluyen dos rangos adicionales para cubrir valores que caen fuera de estos límites: uno para valores menores al mínimo y otro para valores que exceden el máximo. Así, cada variable dentro del estado y las acciones se divide en un total de $k + 2$ intervalos. Aunque se discretizaron el espacio de estados y de acciones, el tamaño de la tabla es un aspecto importante para considerar. Por ejemplo, en un problema en el que se tenga un proveedor con un solo producto y se escoge que $k = 5$ para todas las variables del estado, se tienen un total de $7^4 = 2401$ estados posibles en la Q tabla a los que hay que multiplicarles la cantidad de acciones posibles para determinar la cantidad de Q valores dentro de esta tabla. Este problema se detallará mejor en secciones posteriores del trabajo, pero es algo importante a considerar y que demuestra la complejidad del problema de las cadenas de suministros.

Tras el proceso de discretización, se establece el espacio de estados necesario para la creación de la tabla Q en los agentes de Q Learning. No obstante, dada la cuestión de escalabilidad previamente mencionada, se optó por desarrollar dos enfoques distintos para dos agentes de Q Learning diferentes.

El primer agente se basa en la política de manejo de inventarios s,S, y su objetivo es construir y perfeccionar su tabla Q para mejorar marginalmente el desempeño de esta política. En este diseño, el espacio de acciones se define de modo que, para cada producto, exista una opción binaria que determine si se procede a realizar un pedido de reabastecimiento o no. En caso de reordenar, se aplica el método 'greedy' previamente descrito para calcular las cantidades a comprar de cada producto y de cada proveedor. Así, el espacio de acciones de este agente es de tamaño equivalente al número de productos en el ambiente.

El segundo agente de Q Learning se alinea más estrechamente con la formulación del MDP, pero requiere la discretización de la cantidad a comprar de cada proveedor para cada producto. El enfoque de discretización aquí es diferente: en lugar de dividir en rangos dentro del dominio de la variable, se opta por relacionar la decisión con el porcentaje de la cantidad disponible de un producto de un proveedor a comprar. Este porcentaje también se discretiza. Por ejemplo, si se elige una discretización en 5 valores, las opciones de compra para cada producto de cada proveedor se limitarían a 0%, 25%, 50%, 75% y 100% de la cantidad disponible. Esta metodología, junto con la información discretizada del estado sobre la cantidad disponible, permite al agente tomar decisiones informadas sobre la cantidad a comprar.

De esta manera, se han desarrollado dos agentes distintos basados en Q Learning, cada uno con su propio nivel de complejidad y eficiencia computacional. Mientras uno es más sencillo y por ende

menos demandante en términos de recursos computacionales, el otro ofrece un enfoque más detallado y, por consiguiente, con mayor requerimiento computacional.

La clave para ambos agentes reside en la implementación efectiva del proceso de aprendizaje iterativo. Este proceso emerge de la interacción continua del agente con el ambiente, lo cual facilita la exploración exhaustiva del amplio espacio de estados y acciones. Tal exploración es esencial para el llenado progresivo y la actualización de la tabla Q, lo cual, a su vez, es crítico para la identificación y extracción de la política óptima. Esta política óptima es aquella que, según los datos acumulados en la tabla Q, generaría los mejores resultados en términos de las recompensas obtenidas por el agente.

4.5 Agente DQN

En el contexto de las cadenas de suministro, nos enfrentamos a un problema significativo de escalabilidad, particularmente porque el espacio de estados y el de acciones tienden a crecer exponencialmente con la incorporación de más proveedores, productos y rangos en la discretización, entre otros factores. Rellenar y converger la tabla Q en este escenario se convierte en una tarea extremadamente exigente desde el punto de vista computacional. Como solución a este desafío surge el enfoque de Deep Q-Network (DQN), que integra las redes neuronales y el aprendizaje profundo para modelar de forma no lineal la función $Q(s,a)$, tal como se discute en la literatura especializada (ver "Deep Reinforcement Learning Hands-On").

En DQN, la red neuronal toma como entrada el estado actual y produce como salida los valores Q para cada acción posible, proporcionando una representación no lineal de la función Q. Esta técnica permite manejar directamente variables de estado continuas, eliminando la necesidad de una discretización uno a uno entre estados y acciones, como en el Q-Learning tradicional. Esto permite alcanzar soluciones más precisas y eficientes. Sin embargo, el espacio de acciones aún requiere ser discretizado, ya que las salidas de la red representan los valores Q para cada acción, no la cantidad específica a comprar.

El principal reto en la implementación de DQN es su fundamento en el aprendizaje supervisado, lo que normalmente requeriría conocer los valores Q óptimos para entrenar la red. Sin embargo, en el aprendizaje por refuerzo, estos valores se construyen de manera iterativa. Para adaptar este enfoque iterativo a las redes neuronales, se introducen dos conceptos clave: el 'replay buffer' y la 'target network'.

El 'replay buffer' actúa como una cola FIFO (First In, First Out) que almacena tuplas (S, A, R, S') y permite simular un conjunto de datos de entrenamiento independientes e idénticamente distribuidos (i.i.d), esencial para el entrenamiento de la red neuronal. Aunque la independencia de los datos es

parcial debido a la naturaleza secuencial de las decisiones y el estado, el 'replay buffer' proporciona un mecanismo eficaz para muestrear y utilizar estos datos en el entrenamiento.

Por otro lado, la 'target network' aborda la correlación intrínseca en los pasos secuenciales del agente, que puede generar inestabilidad en la estimación de los valores Q. Recordando la ecuación de Bellman y la naturaleza secuencial del MDP, cada actualización del valor Q se basa en la recompensa obtenida y los valores Q del estado siguiente. La 'target network' sirve para mantener estables los valores Q del estado siguiente ($Q(s', a')$) durante el proceso de actualización. Esto se logra mediante el uso de dos redes neuronales: la red principal, que se actualiza constantemente, y la red objetivo, que es una copia de la red principal, pero se actualiza con menos frecuencia para mantener un equilibrio entre la estabilidad y la relevancia de la información.

Basándonos en los conceptos anteriormente descritos, el proceso de entrenamiento de las redes neuronales en el marco de DQN es relativamente directo, aunque implica varios pasos clave. Después de una fase inicial de preparación, en la cual se acumulan suficientes datos en el 'replay buffer', cada episodio de entrenamiento procede seleccionando aleatoriamente una muestra de estos datos acumulados. A continuación, se calculan los valores Q empleando la ecuación de Bellman, utilizando los valores Q del estado siguiente obtenidos de la 'target network'. Estos valores Q calculados se tratan como si fueran los óptimos para el estado correspondiente y se utilizan para entrenar la red principal.

Este método iterativo asegura que la red neuronal se entrene de manera gradual, permitiéndole desarrollar una representación cada vez más precisa y refinada de los valores $Q(s, a)$. Con cada iteración, la red aprende y ajusta sus parámetros para estimar mejor los valores Q, acercándose así a la política óptima.

En cuanto al espacio de estados y acciones utilizado para esta técnica, se empleó la misma configuración que para el agente de Q Learning más complejo. La principal diferencia radica en el tratamiento de los estados: para DQN, el espacio de estados no requiere discretización, lo que permite que los valores continuos sean procesados directamente por la red neuronal. Sin embargo, el conjunto de acciones sí debe ser discretizado, al igual que en el caso de Q Learning, debido a que la red produce valores Q para acciones discretas. Esta combinación de entradas continuas para los estados y acciones discretas constituye un enfoque híbrido que capitaliza las fortalezas de DQN en el manejo de complejos espacios de estados y acciones en el contexto de las cadenas de suministro.

5 IMPLEMENTACIÓN

En esta sección se van a detallar como fueron implementados los agentes y los modelos descritos anteriormente. Mostrando los datos usados, los parámetros, las tecnologías y todos los detalles para entender el desarrollo y la implementación.

5.1 Instancia del ambiente

Para la implementación del ambiente en el que interactuarán los agentes, se establecieron valores específicos basados en el problema definido anteriormente y en datos obtenidos de la sección de diseño de experimento del trabajo de Gómez et al. (2023).

En lo referente a la estructura general de la cadena de suministros, se optó por una configuración simplificada con 10 periodos, 2 proveedores y 1 producto, debido a consideraciones computacionales relacionadas con el entrenamiento de los agentes. Esta simplificación se analizará más adelante. Los proveedores están ubicados aleatoriamente en una región de 250km x 250km, con el depósito centralizado. Cada vehículo de transporte tiene una capacidad de 6 toneladas (6000 kg), se desplaza a una velocidad promedio de 60 km/h y requiere 10 minutos para la carga de productos en cada proveedor. A partir de estos parámetros se calcula el tiempo de ida y vuelta desde cada proveedor l_s, y se establece un costo unitario de transporte de $c = \$1$ por minuto.

Para el único producto en el modelo, todos los proveedores lo ofrecen. Se fijó una tasa de perecimiento de $\phi = 10\%$. La demanda se calcula aleatoriamente siguiendo el método descrito en el trabajo base: la demanda media de cada simulación se obtiene de una distribución uniforme entre 400 y 900 kg, y en cada periodo se calcula d_{pt} usando una distribución normal con esta media y una desviación estándar del 10%. Similarmente, la cantidad disponible q_{spt} se determina mediante una distribución uniforme entre 1000 y 1500 kg. Para asegurar que las cantidades disponibles superen la demanda y reducir la penalización en instancias pequeñas, se permite que los valores negativos de demanda y cantidad disponible se limiten a 0 y al doble de la media.

Los precios p_{spt} se definen usando un proceso similar, con valores medios entre \$50 y \$120 por cada 100 kg de producto. El valor mínimo permitido es de \$1, y para mantener la simetría, el máximo es el doble de la media menos \$1. El precio de venta x_p se calcula como un 25% superior al promedio de los precios medios de todos los proveedores, mientras que los costos de penalización por no cumplir la demanda k_p y mantenimiento del inventario h_p se fijan en el 20% del precio de venta.

Este ambiente fue implementado usando la librería de Gymnasium que es ampliamente usada para trabajar con problemas de aprendizaje por refuerzo. Esta librería permite la creación de una instancia aleatoria que incluye todos los elementos definidos anteriormente y permite la evolución de esta instancia del ambiente a partir de las distintas decisiones que se toman externamente y el calculo aleatorio de algunos valores. Para crear esta instancia se necesita definir un espacio de estados y de acciones, este coincide con el mismo que se definió inicialmente para el MDP.

5.2 Agentes de políticas de manejos de inventarios

La implementación de estos agentes se centró en analizar el nivel de inventario de cada producto en cada periodo y, a partir de ello, determinar la cantidad a ordenar utilizando el método 'greedy' previamente establecido. Los parámetros para la política R,Q se definieron considerando el 50% de la demanda máxima como umbral para el punto de reorden y el 150% de esta misma demanda como la cantidad a solicitar. Por ejemplo, si la demanda máxima es de 900 kg, se realizará un pedido de 1350 kg cuando el inventario de un producto descienda por debajo de 450 kg.

En cuanto a la política s,S, los parámetros se establecen de manera similar: si el inventario es de 400 kg, se efectúa un pedido para alcanzar un nivel de inventario de 950 kg. La selección de estos parámetros específicos se basó en la demanda máxima, una métrica preconocida, y los porcentajes derivados de un proceso de búsqueda de hiperparámetros orientado a maximizar las recompensas obtenidas por los agentes.

Para facilitar la interacción de estos agentes con el ambiente diseñado, se implementaron clases específicas utilizando el marco de trabajo Gymnasium. Este enfoque permite una integración efectiva y eficiente de las políticas de manejo de inventarios dentro del ambiente de simulación, asegurando así una evaluación precisa y consistente del desempeño de los agentes bajo las condiciones establecidas.

5.3 Modelo mixto-entero de optimización

Para implementar y resolver el modelo de optimización, se empleó la librería Gurobipy, una herramienta eficaz para trabajar con modelos algebraicos en problemas de optimización. Esta librería facilita la integración del modelo algebraico diseñado previamente y utiliza los datos generados por el ambiente para encontrar la solución óptima del problema planteado.

Cabe destacar que el funcionamiento de este modelo de optimización se diferencia del ciclo de interacción estándar entre el agente y el ambiente en simulaciones de aprendizaje por refuerzo. En lugar de integrarse en este flujo interactivo, el modelo se ejecuta de manera independiente. Para ello, se generan datos aleatorios utilizando una semilla específica, lo que asegura la consistencia y reproducibilidad de los escenarios simulados. Una vez generados estos datos, el modelo de optimización se ejecuta sobre ellos para calcular la solución óptima. Esta metodología permite evaluar y comparar el desempeño de los agentes con un escenario de referencia óptimo, proporcionando una valiosa perspectiva sobre la eficacia de las estrategias de los agentes en comparación con el ideal teórico.

5.4 Agentes Q Learning

En la implementación de los agentes de Q Learning, se seleccionaron parámetros específicos tanto para el entrenamiento como para la discretización de los espacios de estados y acciones. Un aspecto crucial en el entrenamiento de un agente de Q Learning es el equilibrio entre exploración y explotación. La exploración implica que el agente debe investigar todas las posibles decisiones y estados disponibles, mientras que la explotación se centra en aprovechar las soluciones más prometedoras identificadas hasta el momento.

Para gestionar este balance, se empleó el método Epsilon-greedy, basado en un parámetro $\epsilon \in [0,1]$. Según este enfoque, con una probabilidad ϵ, el agente opta por una acción aleatoria, fomentando la exploración, y con una probabilidad $1 - \epsilon$, selecciona la acción 'greedy', es decir, la que tiene el mayor valor Q, favoreciendo la explotación. En la configuración inicial de los agentes, se estableció un valor de $\epsilon = 0.3$.

Conforme avanza el entrenamiento y se busca afinar aún más las políticas, se introduce el concepto de épsilon-decay, donde el valor de ϵ se reduce progresivamente en cada episodio hasta alcanzar un mínimo. Este mínimo se fijó en 0.01, permitiendo mantener un grado de exploración incluso en etapas avanzadas del entrenamiento. La tasa de reducción de ϵ varía en función del ambiente específico; para este caso, se adoptó una tasa base de 0.00002. Este enfoque de épsilon-decay asegura que el agente no se quede atrapado prematuramente en una política subóptima, al mismo tiempo que incrementa gradualmente la importancia de explotar las estrategias más efectivas identificadas.

En relación con los parámetros fundamentales de la ecuación de Bellman en la implementación de Q Learning, se eligió trabajar con una tasa de aprendizaje (α) de 0.2 y una tasa de descuento (γ) de 0.8. La elección de γ es particularmente crítica en este contexto, ya que un valor alto permite que las decisiones actuales tomen en consideración los posibles estados futuros que resultan de dichas decisiones. Esta combinación de tasas de aprendizaje y descuento, junto con las estrategias de exploración y explotación previamente mencionadas, proporciona los elementos esenciales para el proceso iterativo de entrenamiento en Q Learning.

Sin embargo, para implementar efectivamente estos agentes, fue necesario detallar la discretización de los espacios de estados y acciones. En el caso del espacio de estados, se optó por una discretización con $k = 4$, lo que resulta en un total de 6 posibles valores para cada variable. Las variables que constituyen la tupla de estados en la tabla Q incluyen el nivel de inventario, la demanda, el precio de compra, la cantidad disponible y, adicionalmente, el costo de transporte. Aunque el costo de transporte no varía dentro de una misma simulación, es crucial incluirlo para evaluar la viabilidad de comprar a ciertos proveedores, especialmente cuando enfrentan costos de transporte altos o

tienen precios elevados o cantidades limitadas de producto. Este esquema de estados se aplicó a ambos agentes.

Respecto al espacio de acciones, el agente basado en la política s,S no requiere una discretización adicional, ya que sus acciones son binarias: reordenar o no. En cambio, para el agente de Q Learning más complejo, se decidió por una discretización con $k = 6$, permitiendo que la decisión de compra para cada producto de cada proveedor se divida en incrementos de aproximadamente 16.6% en un rango de 0% a 100%. Esta granularidad ofrece suficiente flexibilidad para realizar compras que se acerquen a lo óptimo.

Para el agente basado en s,S, se implementó también un parámetro de cantidad a reordenar del 150% de la demanda máxima. Con estos detalles definidos, se procedió a la implementación de los agentes y al diseño de sus respectivas tablas Q, que almacenan los valores para cada combinación de estado y acción.

5.5 Agente DQN

En la implementación del agente DQN, se realizaron algunas adaptaciones en comparación con el agente de Q Learning más complejo, especialmente en lo que respecta a la ecuación de Bellman y el manejo del espacio de estados y acciones. Una diferencia fundamental es que, para DQN, no es necesario discretizar el espacio de estados, eliminando así la necesidad del parámetro k para los estados. Además, se optó por una tasa de reducción de ϵ más rápida, específicamente de 0.0005, debido al mayor costo computacional y a una convergencia más rápida del modelo en comparación con el Q Learning tradicional. Para la ecuación de Bellman en DQN, se utilizó una tasa de aprendizaje (α) de 0.3 y una tasa de descuento (γ) de 0.7, parámetros que se demostraron más efectivos en este contexto debido a la manera en que se estiman los valores Q.

En cuanto a la arquitectura de la red neuronal, se implementaron tres capas: las dos primeras con 64 neuronas cada una y la tercera con 32 neuronas. Las capas intermedias utilizan la función de activación ReLU, mientras que la capa de salida, asociada a los valores Q, emplea una activación lineal. El tamaño de la muestra aleatoria tomada del 'replay buffer' es de 64, lo que implica que se deben haber acumulado al menos esa cantidad de datos en la cola antes de comenzar el entrenamiento. Los pesos de la red neuronal principal se actualizan cada dos pasos de la simulación o al finalizar esta, y la red objetivo actualiza sus pesos cada 100 pasos, lo que equivale a aproximadamente 10 episodios de 10 periodos cada uno. Con estos parámetros y estructura, se construyeron las redes neuronales y se diseñó el proceso de aprendizaje del agente en su interacción con el ambiente.

5.6 Interacción con el ambiente y recursos usados

Después de haber descrito como se diseñaron los agentes y modelos para ser usados junto al ambiente, es importante ver como se hace la integración entre el agente y el ambiente. Este proceso se describe en el siguiente código:

```
for episodio in range(num_episodes):
    # Reiniciar el entorno y obtener el estado inicial
    estado, info = env.reset(seed=seed)
    while True:
        # Elegir una acción basada en el estado actual
        accion = agente.escoger_accion(estado)
        # Ejecutar la acción y obtener el resultado
        estado_siguiente, recompensa, terminado, truncado, info = env.step(accion)
        # Actualizar la tabla Q con los nuevos datos
        agente.actualizar(estado, accion, estado_siguiente, recompensa)
        # Actualizar el estado actual al siguiente estado
        estado = estado_siguiente
        # Verificar si el episodio ha terminado
        if terminado or truncado:
            break
 # Ajustar la probabilidad de exploración
agente.epsilon = max(agente.epsilon * (1- tasa_reduccion), min_epsilon)
```

Este código de Python ilustra el proceso de entrenamiento de agentes en un entorno simulado. Inicialmente, el ambiente se reinicia para cada episodio, generando un conjunto de datos aleatorios que definen las condiciones iniciales. Durante cada episodio, que concluye al alcanzar el décimo periodo, el agente lleva a cabo una serie de simulaciones.

En cada periodo, el agente toma una decisión sobre la acción a ejecutar. Para los agentes que siguen políticas de manejo de inventarios, esta decisión se basa en la evaluación del nivel de inventario

actual. En cambio, para los agentes de Q Learning, se emplea el método Epsilon-greedy, que equilibra entre explorar acciones nuevas y explotar aquellas que ya se conocen como beneficiosas.

Una vez que el agente selecciona una acción, esta se comunica al ambiente. El ambiente, a su vez, actualiza su estado y calcula la recompensa correspondiente, además de verificar si el episodio ha llegado a su fin. Este nuevo estado y la recompensa obtenida son utilizados por el agente para actualizar su conocimiento y, en el caso de los agentes de Q Learning, para entrenar el modelo.

Posteriormente, el estado actual del agente se actualiza con la información más reciente del ambiente y se verifica si el episodio ha concluido. Este proceso se repite a lo largo del número predefinido de episodios. A medida que avanza el entrenamiento, el valor de épsilon se reduce gradualmente, promoviendo así una mayor explotación de las acciones conocidas en contraposición a la exploración.

Al combinar este proceso iterativo con los agentes definidos previamente, se logra entrenar efectivamente a estos agentes en diversos escenarios, tanto específicos como aleatorios, y se obtienen resultados que reflejan su desempeño en el entorno simulado. Para el entrenamiento de los agentes se usó un computador con un procesador Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz 2.50 GHz, 12Gb de RAM, Windows 11, Python 3.9.7 y Gurobipy 10.0.3.

5.7 Resultados esperados

Tras detallar el diseño y la implementación de los agentes y modelos, es evidente que diversas decisiones críticas influyen en los resultados que se esperan obtener. Una de las consideraciones más significativas ha sido el tamaño del ambiente, que se vio restringido por las limitaciones de los recursos computacionales disponibles. Al intentar ejecutar instancias más grandes, se enfrentaron problemas de rendimiento.

A pesar de estas limitaciones, se anticipa que los resultados obtenidos de los agentes implementados en entornos de menor escala superarán notablemente a aquellos derivados de las políticas tradicionales de manejo de inventarios. En particular, se espera que el desempeño del agente de Q Learning basado en la política SS sea ligeramente superior al de las políticas convencionales de manejo de inventarios. Se prevé que tanto el agente de Q Learning complejo como el agente DQN ofrezcan resultados aún mejores que los anteriores. Es probable que el modelo DQN, gracias a su capacidad para aproximar mejor la tabla Q y manejar estados no discretizados, muestre un rendimiento ligeramente superior al de los otros agentes de Q Learning.

Sin embargo, también se prevé que el costo computacional para entrenar estos agentes, especialmente el DQN, sea considerablemente alto. Esta expectativa se basa en la complejidad inherente al entrenamiento de modelos basados en redes neuronales y en la necesidad de procesar

espacios de estados y acciones de mayor tamaño y complejidad. A pesar de estos costos, la precisión y la capacidad de generalización que ofrecen estos modelos avanzados podrían justificar la inversión en recursos computacionales.

6 VALIDACION

6.1 Métodos

En la validación y comparación de resultados, utilizamos una métrica que se alinea directamente con la fórmula de recompensas del Proceso de Decisión Markoviano (MDP, por sus siglas en inglés) descrito en la sección 1. Esta métrica consiste en la suma acumulada de recompensas obtenidas en todos los periodos de un episodio. Este total no solo refleja las ganancias de los agentes, sino que también incorpora todos los ingresos y costes asociados con la operación de la cadena de suministro simulada en el entorno.

Para validar los resultados, adoptamos un enfoque dual. Primero, evaluamos los agentes de aprendizaje por refuerzo de forma individual para analizar la efectividad y evolución de sus modelos. Segundo, realizamos una evaluación comparativa, integrando todos los agentes y modelos para determinar cómo se desempeñan los enfoques propuestos frente a las alternativas.

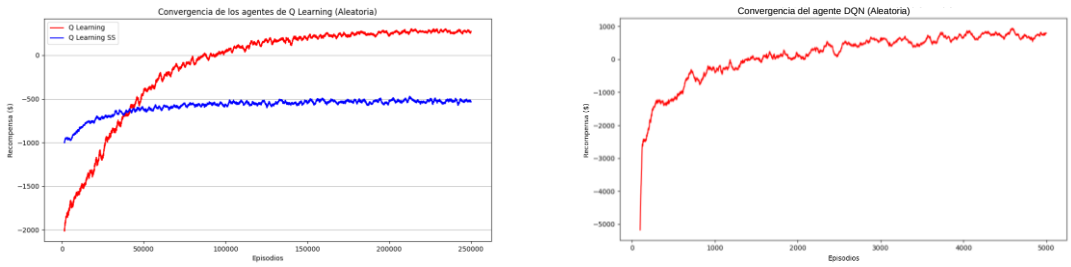
Para cada comparación, ejecutamos 200 episodios por modelo. Este enfoque permite capturar la incertidumbre intrínseca del entorno y de los agentes. Si bien realizar comparaciones en un escenario específico no es suficiente para establecer una jerarquía definitiva entre modelos, sí es útil para validar y entender el proceso de convergencia en el aprendizaje por refuerzo.

6.2 Convergencia de agentes

Uno de los criterios fundamentales para la efectividad de los agentes de aprendizaje por refuerzo es su capacidad para aprender y mejorar a lo largo de las iteraciones y episodios, con el objetivo de acercarse a una política óptima en el marco del Proceso de Decisión Markoviano (MDP). En este contexto, hemos centrado nuestra evaluación en la convergencia de los dos agentes que emplean Q-Learning y el agente que utiliza Deep Q-Network (DQN).

La convergencia de estos agentes se analiza idealmente bajo una variedad de escenarios aleatorios para garantizar la robustez de los modelos. Sin embargo, también hemos considerado útil realizar una evaluación de la convergencia utilizando un único escenario de entrenamiento. Este enfoque nos permite realizar comparaciones más directas entre los agentes y comprender mejor cómo evolucionan y se adaptan a lo largo del tiempo.

Para visualizar esta evolución, se utilizan gráficos de convergencia. Estos gráficos muestran la progresión de las recompensas obtenidas por los modelos a lo largo de los episodios, proporcionando una representación visual clara de cómo los agentes aprenden y mejoran su desempeño en el entorno simulado.



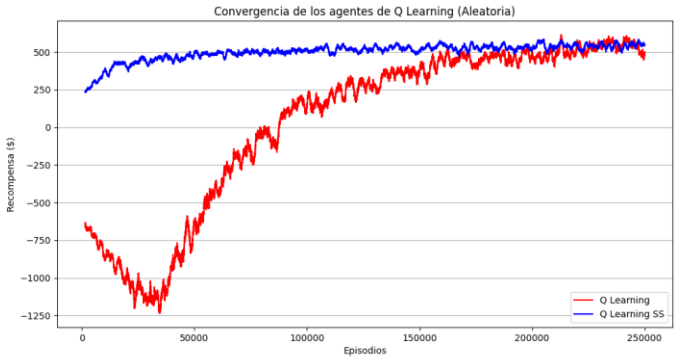
Entrenamiento de los agentes con datos aleatorios (media móvil 1500 y 100 episodios respectivamente)

Estas gráficas ilustran la convergencia de los tres agentes, destacando las diferencias en la velocidad y los valores de convergencia entre ellos. Notablemente, el agente de Q-Learning basado en la política S,s muestra una convergencia más rápida. Esta eficiencia se atribuye a su espacio de acciones más reducido, facilitando la convergencia de los valores.

Por otro lado, los agentes de Q-Learning complejo y DQN presentan un contraste interesante. A pesar de que el tiempo de entrenamiento del agente DQN fue aproximadamente diez veces mayor al del Q-Learning, se ejecutaron 50 veces menos episodios. Ambos modelos parecen alcanzar un punto de convergencia después del número de episodios asignado, con el modelo DQN mostrando una recompensa ligeramente superior.

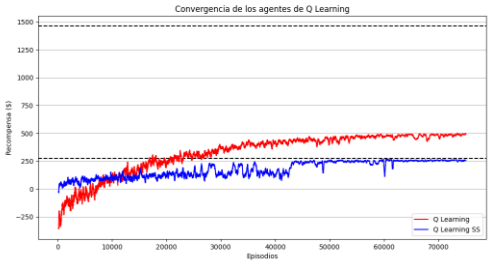
Es relevante observar que, en las etapas iniciales, el modelo basado en S,s obtiene recompensas más altas en comparación con los otros dos agentes. Esto podría explicarse por la menor complejidad en su espacio de acciones, lo que reduce las posibilidades de tomar decisiones altamente erróneas que resulten en menores recompensas. Sin embargo, este agente no logra alcanzar un punto de convergencia positivo en términos de recompensas, lo que sugiere una tendencia a penalizaciones constantes, ya sea por excesos de compra o por incumplir con la demanda.

En este contexto, también es útil considerar la variante del modelo sin penalizaciones. Este enfoque podría ser más apropiado en ciertos contextos específicos de algunas cadenas de suministros, donde las características del producto o las dinámicas del mercado pueden justificar una estrategia de inventario diferente.



Entrenamiento de los agentes de Q Learning (gráfico sin penalizaciones)

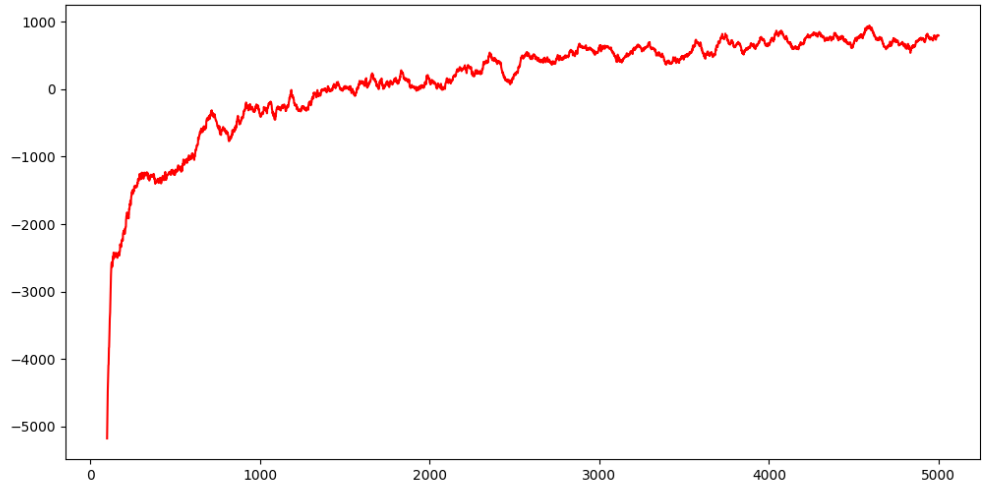
Esta gráfica es generada a partir de las recompensas obtenidas sin penalizaciones. Esto significa que se grafican las recompensas sin tener en cuentas las penalizaciones pero que los agentes fueron igualmente entrenados con estas penalizaciones. Del gráfico se puede ver como las recompensas en el punto de convergencia de ambos modelos son positivas y casi iguales. Esto se debe a que debido a que no hay penalizaciones, la diferencia que puede haber solamente teniendo en cuenta los ingresos por ventas y los costos por compra no son lo suficientes para ver una diferencia entre ambos agentes. Asimismo, es interesante ver como el agente más complejo al inicio de su entrenamiento si tiene recompensas negativas y además tiene un patrón decreciente. Esto se explicaría principalmente a que no necesariamente cuando el modelo tiene menores penalizaciones este tiene unos mayores beneficios. Tienen que pasar aproximadamente 45 mil episodios para que el modelo empiece su comportamiento creciente. Además, al igual que para las anteriores gráficas, el amplio espacio de acciones permite que el modelo tome decisiones muy equivocadas y por lo tanto sus recompensas, aún sin penalizaciones, sean negativas.



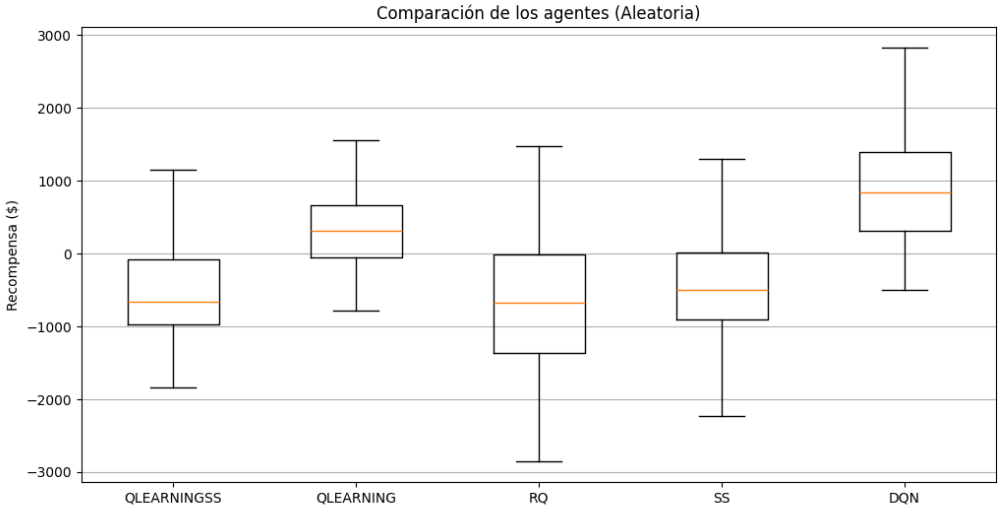
Entrenamiento de los agentes de Q Learning con un escenario especifico (semilla 42)

Entrenamiento Q Learning Aleatorio Sin Penalizaciones

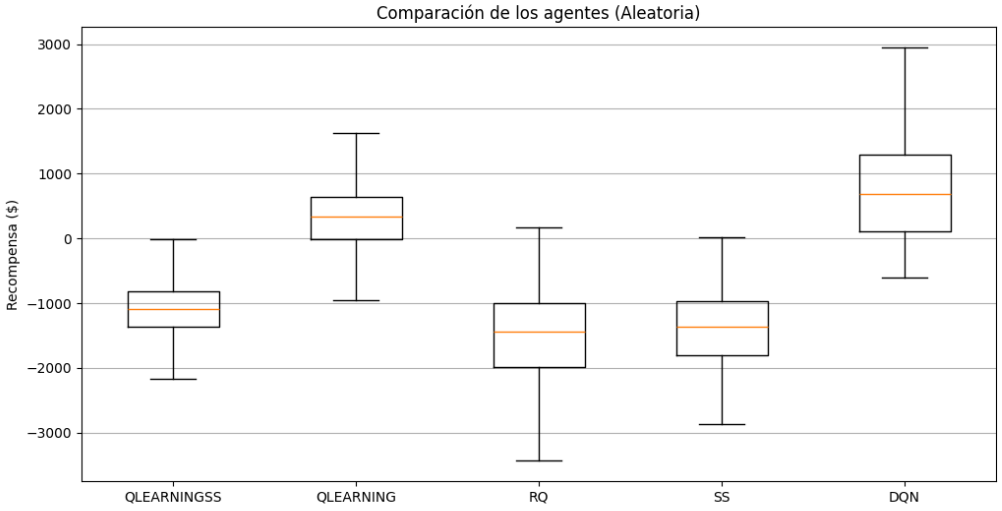
DQN Entrenamiento Aleatorio



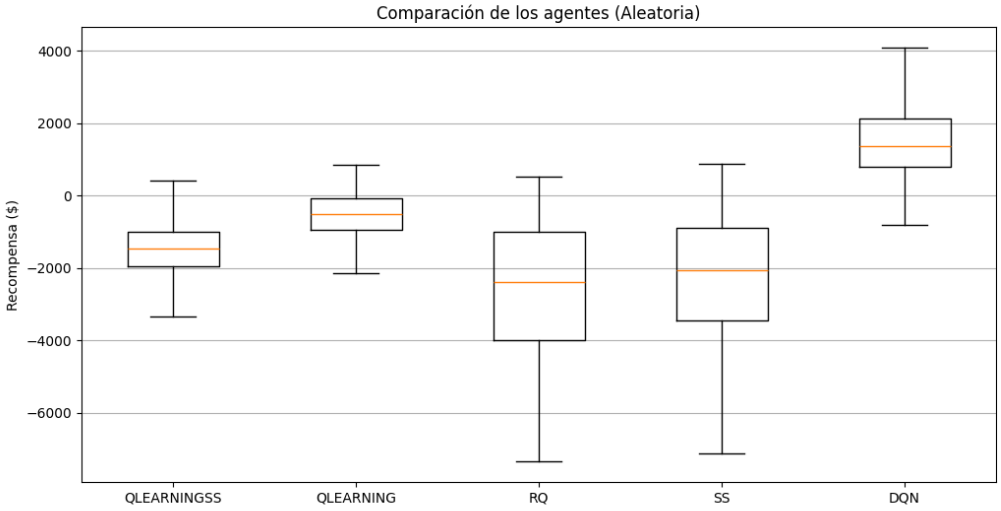
Comparación modelos aleatoria



Comparación modelos perecibilidad 0.4

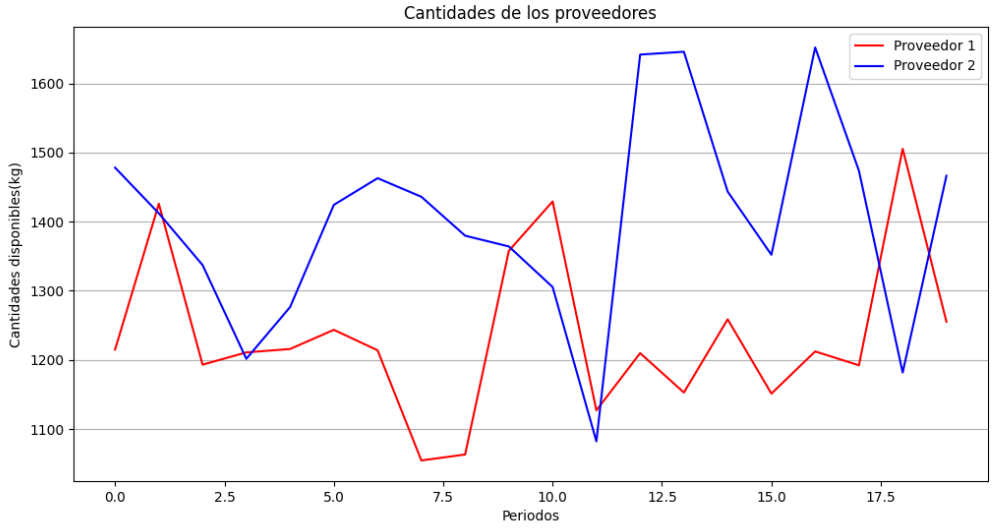


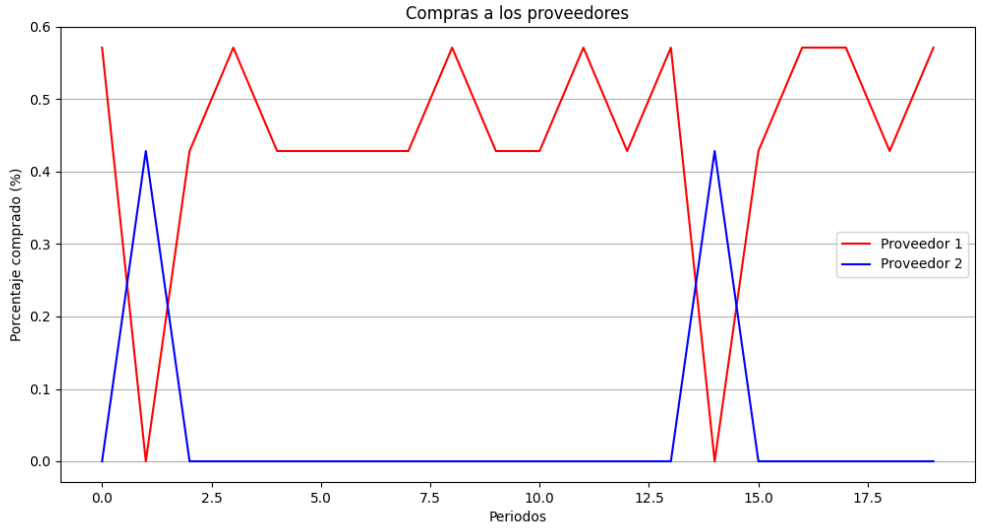
Comparación modelos máxima demanda 1800

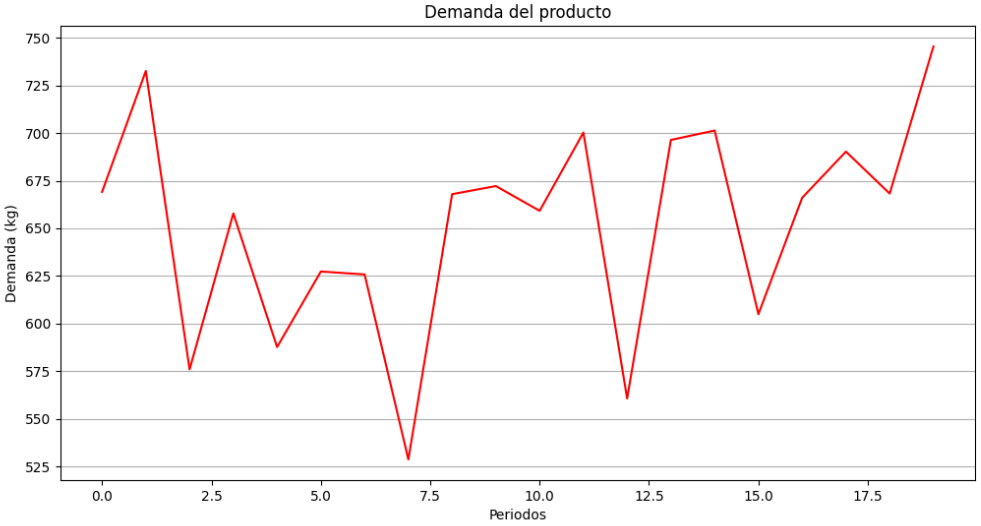


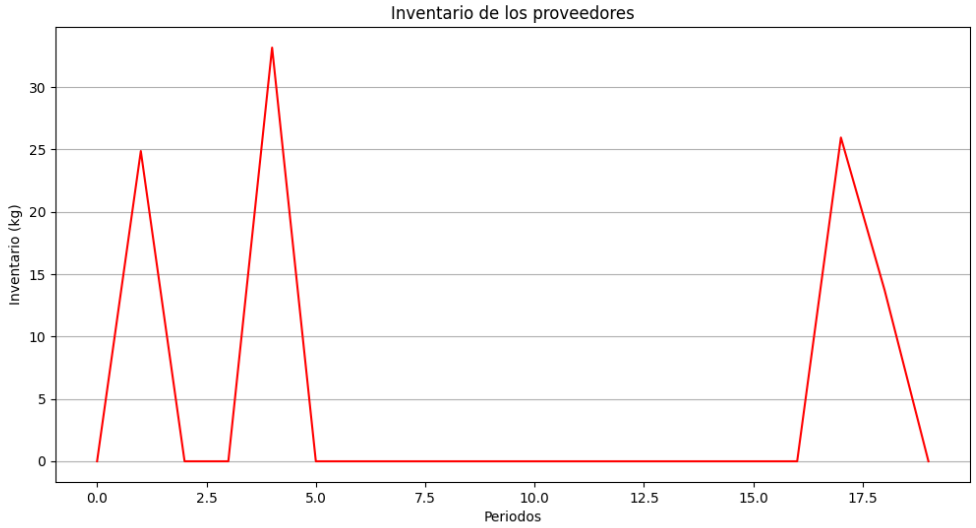
Modelo DQN Semilla 42

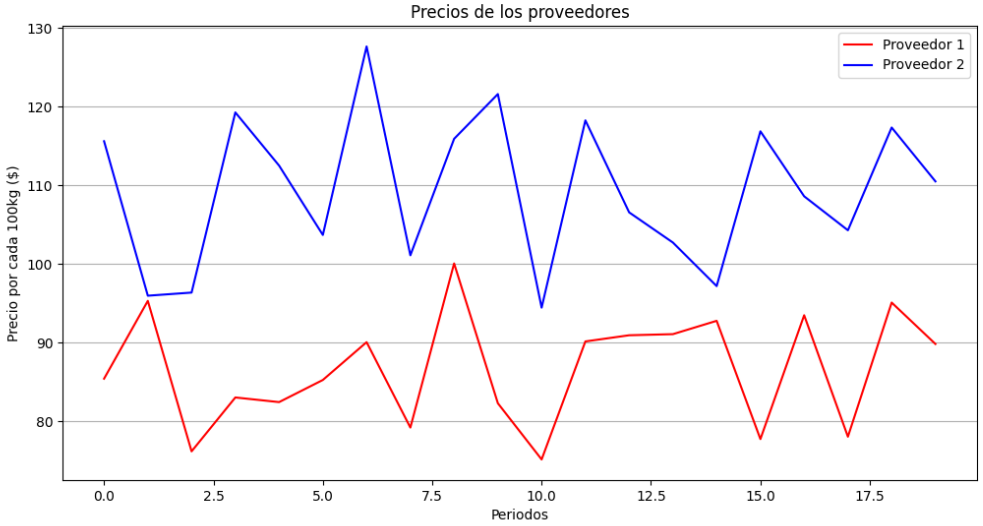
Tiempo de transporte = [159,127]




Compras a los proveedores
Porcentaje comprado (%)
Periodos
Proveedor 1
Proveedor 2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Demanda del producto
Demanda (kg)
Periodos
525
550
575
600
625
650
675
700
725
750
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Inventario de los proveedores
Inventario (kg)
Periodos
0
5
10
15
20
25
30
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Precios de los proveedores
Precio por cada 100kg ($)
Periodos
Proveedor 1
Proveedor 2
80
90
100
110
120
130
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

7 CONCLUSIONES

7.1 Discusión

Resumen del trabajo, discutiendo el desempeño y las limitaciones; problemas encontrados y cómo pudieron o podrían resolverse; lo que falta por hacer.

Validación de los resultados de forma cuantitativa y cualitativa.

7.2 Trabajo futuro

Lista de sugerencias para trabajos futuros. Enfatizar aquellos resultados que merezcan consideración especial (v.gr., casos especiales que deban ser tenidos en cuenta, necesidad de proteger la propiedad intelectual, etc.).

List of suggestion for future work. Emphasize any results that merit special consideration (e.g., special cases to be treated, need to protect intellectual property, etc.).

8 REFERENCIAS

Gutiérrez, A. (2021). Estudio de la cadena de suministro [PDF document]. Recuperado de https://repositorio.ulima.edu.pe/bitstream/handle/20.500.12724/13303/Gutierrez_Estudio-cadena-suministro.pdf?sequence=1

EU Food Information Council. (2021). Los beneficios y la sostenibilidad de las cadenas de suministro de alimentos cortas. Recuperado de https://www.eufic.org/es/produccion-de-alimentos/articulo/Los-beneficios-y-la-sostenibilidad-de-las-cadenas-de-suministro-de-alimentos-cortas

Kaizen Institute. (2021). Importancia de la optimización de la cadena de suministro. Recuperado de https://kaizen.com/es/insights-es/importancia-optimizacion-cadena-suministro/

IBM. (2021). Optimización de la cadena de suministro. Recuperado de https://www.ibm.com/mx-es/topics/supply-chain-optimization

Cozowicz, M. (2021). Reinforcement Learning in Supply Chain [LinkedIn article]. Recuperado de https://www.linkedin.com/pulse/reinforcement-learning-supply-chain-markus-cozowicz/

HUGGING FACE. (N.D.). OFFLINE VS ONLINE LEARNING. RECUPERADO DE HTTPS://HUGGINGFACE.CO/LEARN/DEEP-RL-COURSE/EN/UNITBONUS3/OFFLINE-ONLINE

HUBBS, C., HEITZ, G., & DILKINA, B. (2019). OR-GYM: A REINFORCEMENT LEARNING ENVIRONMENT FOR OPERATIONS RESEARCH PROBLEMS. RECUPERADO DE HTTP://EGON.CHEME.CMU.EDU/PAPERS/HUBBS_OR_GYM_9_11.PDF

van Hasselt HP (2010) Double Q-Learning. In: Advances in Neural Information Processing Systems, The MIT Press, vol 23

HTTPS://LIBSTORE.UGENT.BE/FULLTXT/RUG01/002/790/831/RUG01-002790831_2019_0001_AC.PDF

HTTPS://ARNO.UVT.NL/SHOW.CGI?FID=159101

http://www.incompleteideas.net/book/RLbook2020.pdf

HTTPS://REPOSITORIO.UNIANDES.EDU.CO/SERVER/API/CORE/BITSTREAMS/CD4EA271-AAC3-4ACA-AA5C-0FA131138E87/CONTENT

Lapan, Maxim. *Deep Reinforcement Learning Hands-On: Apply Modern RL Methods to Practical Problems of Chatbots, Robotics, Discrete Optimization, Web Automation, and More, 2nd Edition*. 2nd ed. Birmingham: Packt Publishing, 2020. Print.

APÉNDICES

Datos relevantes que puedan ser consultados para soportar el diseño, la implementación y / o los resultados.