



Máster Universitario en Big Data

Trabajo de Fin de Máster

Analysis of the Application of Microsoft Bonsai with  
Digital Twins for Warehouse Process Optimization

Author

Agustín Rodríguez Agudo

Supervised by

Patricio Ivan Pipp

Madrid

January 2025

## Resumen

El presente trabajo de final de máster se llevó a cabo en la empresa donde realicé mis prácticas y tuvo como objetivo principal el desarrollo de un proyecto basado en las herramientas AnyLogic y Bonsai. El proyecto consistió en la modelización y simulación de procesos en un almacén, con el fin de optimizar la selección del servicio que requeriría menos tiempo para procesar los paquetes, utilizando la herramienta Bonsai.

En una primera fase, se procedió a familiarizarse con las funcionalidades y capacidades de AnyLogic y Bonsai. Se creó un modelo simple de simulación que representaba el procesamiento de paquetes en un almacén, permitiendo evaluar el rendimiento y la eficiencia del sistema en diferentes escenarios. Posteriormente, se utilizó Bonsai para optimizar el proceso de selección del servicio más rápido, con el objetivo de reducir los tiempos de procesamiento.

Una vez adquiridas las habilidades necesarias, se desarrolló otro modelo en AnyLogic, el cual simulaba un almacén con zonas específicas para descarga, carga, almacenaje y empaquetado de paquetes. El objetivo de este modelo era optimizar las olas de pedidos, es decir, la agrupación de pedidos para su procesamiento eficiente. A través de la integración de Bonsai, se buscaba minimizar el tiempo transcurrido desde la recepción de un pedido hasta su despacho.

En la etapa final del proyecto, se utilizó el último modelo desarrollado en AnyLogic para analizar la capacidad de Bonsai en proyectos más complejos de optimización de procesos logísticos en almacenes. El objetivo era determinar si Bonsai podía ser presentado como una solución viable y eficiente para los clientes en el ámbito de la logística de paquetería.

**Palabras clave:** Simulación, Aprendizaje por Refuerzo, AnyLogic, Bonsai, Logística, Almacenamiento, Ola de pedidos.

## Abstract

The present master's thesis project was carried out at the company where I did my internship and had the main objective of developing a project based on the tools AnyLogic and Bonsai. The project consisted of modeling and simulating processes in a warehouse to optimize the selection of the service that would require the least amount of time to process packages, using the Bonsai tool.

In the initial phase, I familiarized myself with the functionalities and capabilities of AnyLogic and Bonsai. A simple simulation model was created to represent the package processing in a warehouse, allowing for the evaluation of system performance and efficiency under different scenarios. Subsequently, Bonsai was used to optimize the selection process for the fastest service, with the aim of reducing processing times.

Once the necessary skills were acquired, another model was developed in AnyLogic, simulating a warehouse with specific areas for unloading, loading, storage, and packaging of packages. The objective of this model was to optimize the wave picking process, which involves grouping orders for efficient processing. By integrating Bonsai, the goal was to minimize the time elapsed from order reception to dispatch.

In the final stage of the project, the last model developed in AnyLogic was used to analyze the capacity of Bonsai in more complex projects for optimizing logistics processes in warehouses. The objective was to determine if Bonsai could be presented as a viable and efficient solution for clients in the realm of package logistics.

**Keywords:** Simulation, Reinforcement Learning, AnyLogic, Bonsai, Logistic, Warehousing, Wave orders

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	1
1.2. Objetivos del TFM . . . . .	2
1.3. Estructura de la memoria . . . . .	3
<b>2. Estado del Arte</b>	<b>4</b>
2.1. Simulación . . . . .	4
2.1.1. Conceptos básicos . . . . .	4
2.1.2. Ventajas y Desventajas . . . . .	5
2.1.3. Discrete-Event Simulation . . . . .	7
2.2. Logística y Warehousing . . . . .	8
2.2.1. Introducción a la logística . . . . .	8
2.2.2. Unidades básicas de un almacén (warehouse) . . . . .	8
<b>3. Herramienta de simulación: AnyLogic</b>	<b>11</b>
3.1. ¿Qué es? . . . . .	11
3.2. ¿Por qué AnyLogic? . . . . .	12
3.2.1. Enfoque multi-metodología . . . . .	12
3.2.2. Interfaz gráfica de usuario . . . . .	13
3.2.3. Integración con otras herramientas . . . . .	15
<b>4. Modelos de Simulación</b>	<b>16</b>
4.1. Modelo Simple . . . . .	16
4.1.1. Objetivo del modelo . . . . .	16
4.1.2. Componentes Principales . . . . .	17
4.1.3. Interacciones entre los componentes . . . . .	18
4.1.4. Variables y parámetros . . . . .	20
4.2. Modelo Warehousing . . . . .	21
4.2.1. Objetivo del modelo . . . . .	21
4.2.2. Componentes principales . . . . .	22
4.2.3. Interacciones entre las componentes . . . . .	24

4.2.4.	Variables y parámetros . . . . .	26
<b>5.</b>	<b>Conceptos Básicos Reinforcement Learning</b>	<b>27</b>
5.1.	Introducción al Reinforcement Learning . . . . .	27
5.1.1.	Definición . . . . .	27
5.2.	Componentes clave Reinforcement Learning . . . . .	28
5.3.	Algunos Algoritmos de Reinforcement Learning . . . . .	30
5.3.1.	APEX-DQN . . . . .	30
5.3.2.	Proximal Policy Optimization . . . . .	31
<b>6.</b>	<b>Bonsai</b>	<b>33</b>
6.1.	¿Qué es Bonsai? . . . . .	33
6.1.1.	¿En que se diferencia? . . . . .	33
6.2.	Componentes principales . . . . .	34
6.2.1.	Entorno y Simulación . . . . .	35
6.2.2.	Proceso de aprendizaje: . . . . .	36
6.2.3.	Curriculum . . . . .	37
6.2.4.	Bonsai Brains . . . . .	37
6.3.	Integración y aplicación en el proyecto . . . . .	38
6.3.1.	Preparación y carga del simulador . . . . .	38
6.3.2.	Configuración y Entrenamiento con Bonsai . . . . .	40
6.3.3.	Validación y Exportación del modelo . . . . .	40
<b>7.</b>	<b>Resultados y Conclusiones</b>	<b>42</b>
7.1.	Resultados Modelo Simple . . . . .	42
7.1.1.	Configuración de entrenamiento . . . . .	42
7.1.2.	Resultados . . . . .	44
7.2.	Modelo Warehousing . . . . .	47
7.2.1.	Configuración de entrenamiento . . . . .	47
7.2.2.	Resultados . . . . .	49
7.3.	Conclusiones . . . . .	55
	<b>Bibliografía</b>	<b>58</b>

# Índice de figuras

3.1. Ejemplo de librería especial para almacenamiento . . . . .	14
3.2. Ejemplo de distintas gráficas en AnyLogic . . . . .	14
4.1. Icono de Agente . . . . .	17
4.2. Icono de Select Output 5 . . . . .	17
4.3. Icono de la Queue . . . . .	18
4.4. Icono del Hold . . . . .	18
4.5. Icono del Servicio . . . . .	18
4.6. Icono de la maquina . . . . .	18
4.7. Icono de la averia . . . . .	18
4.8. Modelo Básico . . . . .	20
4.9. Icono de Move To . . . . .	22
4.10. Icono de Delay . . . . .	22
4.11. Icono de Split . . . . .	23
4.12. Icono de Combine . . . . .	23
4.13. Icono de Wait . . . . .	23
4.14. Icono de Store . . . . .	23
4.15. Icono de Retrieve . . . . .	24
4.16. Icono de Convey . . . . .	24
4.17. Icono de Convey to Exit . . . . .	24
6.1. Esquema propio de la plataforma Bonsai . . . . .	35
6.2. Ejemplo de la observation del modelo de warehousing . . . . .	39
6.3. Ejemplo de la action en el modelo de warehousing . . . . .	39
6.4. Ejemplo del Inkling code del modelo de Warehousing . . . . .	41
6.5. Ejemplo de la imagen del cerebro exportado en el Docker . . . . .	41
7.1. Función Reward a lo largo de los episodios . . . . .	44
7.2. Función Reward en las ultimas iteraciones . . . . .	45
7.4. Imágenes de los test realizados . . . . .	47
7.5. Desarrollo del Reward a lo largo de las iteraciones . . . . .	50
7.6. Acciones . . . . .	52

7.7. Imágenes de los test realizados . . . . . 53

# Índice de cuadros

2.1. Ejemplo de uso de los elementos de una simulación . . . . .	6
2.2. Algunas estrategias de almacenamiento . . . . .	9
2.3. Algunas estrategias de recogida . . . . .	10
7.1. Resultados del test del modelo básico . . . . .	46
7.2. Ejemplo de uso de los elementos de una simulación . . . . .	51

# Capítulo 1

## Introducción

### 1.1. Motivación

La motivación principal para realizar este trabajo de fin de máster radica en la importancia de adquirir conocimientos y habilidades en dos áreas fundamentales en el campo de la optimización de procesos: la simulación de sistemas utilizando AnyLogic y el aprendizaje por refuerzo con Microsoft Bonsai. Estas herramientas son utilizadas ampliamente en la industria para mejorar la eficiencia y la toma de decisiones en diversos escenarios, incluido el ámbito del warehousing.

El proyecto desarrollado en Accenture proporcionó una oportunidad única para aplicar estos conocimientos teóricos y adquirir experiencia práctica en la creación de modelos de simulación y en el desarrollo de cerebros de Bonsai para optimizar problemas reales en el contexto del warehousing. Esta combinación de tecnologías emergentes representa un enfoque innovador y prometedor para abordar los desafíos complejos y dinámicos que enfrentan las empresas en la gestión de almacenes y la optimización de sus operaciones.

Además, al realizar el trabajo en el entorno de una empresa reconocida como Accenture, se tuvo la oportunidad de aplicar estos conocimientos en un contexto empresarial real y comprender las necesidades y desafíos específicos que enfrentan las organizaciones en la gestión de sus operaciones logísticas. Esto permitió no solo evaluar la viabilidad y utilidad de la integración de AnyLogic y Bonsai en proyectos reales, sino también explorar el potencial de estas tecnologías para mejorar la eficiencia y la toma de decisiones en entornos de warehousing.

## 1.2. Objetivos del TFM

El trabajo de fin de máster se planteó con dos objetivos principales. El primero de ellos consistía en adquirir habilidades en la creación y utilización de modelos de simulación en AnyLogic, centrándose específicamente en la gestión de almacenamiento. Este objetivo implicaba comprender los conceptos fundamentales de la simulación, aprender a modelar y simular diferentes escenarios de almacenamiento, así como analizar los resultados obtenidos para la toma de decisiones efectivas. Además, se buscaba adquirir experiencia práctica en el uso de AnyLogic como herramienta de simulación.

La segunda parte de este objetivo del trabajo era familiarizarse con la herramienta de Microsoft llamada Bonsai, la cual se utiliza para desarrollar modelos de Aprendizaje por Refuerzo (RL, por sus siglas en inglés). Mediante el aprendizaje de Bonsai, se pretendía comprender los conceptos clave del RL, como las políticas, las recompensas y los algoritmos de entrenamiento. También se buscaba adquirir habilidades en la creación de cerebros de Bonsai, que son modelos que aprenden a través de la interacción con su entorno.

Una vez alcanzados estos objetivos de aprendizaje, se planteó la tarea de plasmar todos los conocimientos adquiridos en una guía completa y comprensible para el lector. Esta guía debía explicar tanto el porqué como el cómo utilizar las herramientas de AnyLogic y Bonsai para abordar problemas relacionados con la gestión de almacenamiento. Se buscaba que el lector pudiera seguir los pasos detallados y entender la lógica detrás de cada decisión tomada durante el modelado y la implementación de las soluciones.

El objetivo final del trabajo consistió en poner a prueba las capacidades de las herramientas AnyLogic y Bonsai mediante el desarrollo de un modelo más complejo. Este modelo simulaba la llegada de pedidos y su procesamiento en un entorno de almacenamiento, con el propósito de abordar el desafío de controlar las olas de pedidos, una problemática común en los sistemas de comercio electrónico.

El objetivo específico era evaluar si Bonsai podía ser una solución efectiva para optimizar la gestión de almacenamiento en entornos similares, brindando beneficios significativos en términos de eficiencia y rendimiento. Para lograrlo, se planteó un escenario realista que reflejara las complejidades y desafíos inherentes a la gestión de las olas de pedidos.

Mediante la implementación de Bonsai en este contexto, se buscaba demostrar

la aplicabilidad y utilidad de esta herramienta como una solución innovadora y eficaz para la optimización de la gestión de almacenamiento en entornos comerciales.

### 1.3. Estructura de la memoria

La estructura de la memoria sigue una secuencia lógica que busca guiar al lector a través de los conceptos fundamentales y los desarrollos realizados en el trabajo. Comienza con un primer capítulo introductorio que proporciona una visión general de los conceptos básicos de la logística y la simulación. Este capítulo tiene como objetivo establecer una base de conocimientos para el lector, que será fundamental para comprender los temas abordados a lo largo del trabajo.

A continuación, se dedica un capítulo a explicar de manera concisa pero clara qué es AnyLogic y por qué es una herramienta relevante en el contexto de este trabajo. Se presentarán los fundamentos y las funcionalidades básicas de AnyLogic, resaltando su utilidad para la creación de modelos de simulación en el ámbito de la logística y el almacenamiento.

Posteriormente, se desarrollan en detalle tanto el modelo básico como el modelo de almacenamiento. Estos capítulos brindan una descripción exhaustiva de la metodología utilizada, los componentes del modelo y las lógicas implementadas. El objetivo es que el lector adquiera una comprensión completa de la construcción y el funcionamiento de ambos modelos, con el fin de poder replicarlos o adaptarlos a situaciones similares.

A continuación, se introduce un capítulo dedicado a los conceptos básicos del aprendizaje por refuerzo. Se explica de manera accesible qué es el aprendizaje por refuerzo y cómo se utiliza para resolver problemas en entornos de simulación. Este capítulo busca proporcionar al lector los conocimientos necesarios para comprender la implementación de Bonsai en los modelos desarrollados.

El siguiente capítulo está dedicado a Bonsai. Se presenta en detalle qué es Bonsai, cómo se utiliza y qué beneficios puede aportar en la optimización de la gestión de almacenamiento. Se ofrece una guía práctica sobre cómo implementar Bonsai en los modelos de simulación, así como ejemplos de su aplicación en el contexto de los problemas abordados en el trabajo.

Finalmente, se presenta un capítulo de resultados y conclusiones. En este capítulo se exponen los resultados obtenidos a partir de los modelos y se analizan en función de los objetivos planteados al inicio de la memoria.

# Capítulo 2

## Estado del Arte

En este capítulo, exploraremos de forma concisa los conceptos básicos de la simulación y la logística. La simulación nos permite modelar y analizar el comportamiento de sistemas complejos, mientras que la logística se enfoca en la planificación y gestión eficiente del flujo de bienes, servicios e información. Descubriremos cómo la simulación puede ser aplicada en la logística para optimizar la cadena de suministro, mejorar la eficiencia y tomar decisiones informadas.

### 2.1. Simulación

En un mundo ideal se espera que el modelo o entorno con el que se trabaja sea lo más simple posible. Con ello se conseguiría obtener la respuesta exacta a las preguntas mediante métodos matemáticos, esto se denomina solución analítica. Esta claro, que no se vive en ese mundo, hoy en día los entornos son demasiado complejos para evaluarse analíticamente. Por ello, se utilizan ordenadores que simulan dicho modelo con el objetivo de aproximar su características concretas.

Un ejemplo claro para entender el concepto de simulación es el siguiente: *Una empresa de logística A llega nueva a una ciudad y quiere optimizar su ruta de transporte en su cadena de suministro. La simulación implica simular diferentes escenarios, como cambios en la ubicación de los centros de distribución, capacidades de carga o restricciones de tiempo, para minimizar los costos de transporte, reducir los tiempos de entrega y mejorar la eficiencia general.*

#### 2.1.1. Conceptos básicos

Las siguientes definiciones pertenecen a [LK91]

**Definición 1. (Simulación.)** Una simulación es la imitación de una operación del mundo real o de un sistema a lo largo del tiempo.

**Definición 2. (Entidad.)** Elemento individual del sistema que se está simulando y cuyo comportamiento se está analizando explícitamente.

**Definición 3. (Sistema.)** Un sistema se define como un conjunto de entidades.

**Definición 4. (Estado.)** Un estado es una colección de variables que son necesarias para describir el sistema en un determinado momento de tiempo.

**Definición 5. (Sistema discreto.)** Un sistema discreto es aquel para el cual las variables de estado cambian instantáneamente en puntos separados en el tiempo.

**Definición 6. (Evento.)** Un evento se define como una ocurrencia que hace que cambie el estado del sistema.

En la tabla 2.1 se muestran una serie de ejemplos para entender mejor estos conceptos.

### 2.1.2. Ventajas y Desventajas

Una pregunta que se puede hacer es ¿Por qué y por qué no utilizar una simulación? La respuesta a esta pregunta la dieron *Pedgen, Shannon y Sadowski* [PSR95]. Algunas de las ventajas son:

- Se pueden explorar nuevas políticas, procedimientos, flujos, entre otros, sin interrumpir las operaciones en curso del sistema real.
- Las hipótesis sobre cómo o por qué ocurren ciertos fenómenos pueden probarse para su viabilidad.
- El tiempo se puede ralentizar o adelantar para permitir una aceleración o desaceleración de los hechos bajo investigación.
- Se obtiene información sobre la iteración de las variables.

Por otro lado, se encuentran algunas desventajas como:

- El diseño y construcción de modelos requiere una gran formación. Además, es muy poco probable que dos modelos construidos por dos personas diferentes sean el mismo.
- Los resultados de la simulación son difíciles de interpretar. La mayoría de los resultados son esencialmente variables aleatorias por lo que puede ser difícil distinguir si una observación es resultado de las iteraciones o de la aleatoriedad.

Cuadro 2.1: Ejemplo de uso de los elementos de una simulación

Sistema	Entidades	Atributos	Actividades	Eventos	Estados
Reservas de un hotel	Cientes, habitaciones, reservas	Nombre del cliente, número de habitación, fecha de reserva	Realizar y cancelar una reserva	Llegada y salida de un cliente, cancelación de una reserva	Estado de la reserva (confirmada, cancelada, en espera)
Seguimiento de paquetes	Paquetes, transportistas, destinos	Nº de envío, estado, fecha estimada	Registrar y actualizar paquete	Paquete entregado o en tránsito	Ubicación, estado, historial
Gestión de inventario	Productos, proveedores, clientes	Cantidad en stock, precio, fecha	Registrar o vender productos, llegada de nuevos productos, agotamiento de stock	Nivel de inventario, estado del pedido	

- El modelado y análisis de simulación puede llevar mucho tiempo y coste computacional. Escatimar en estos recursos puede dar lugar a que la simulación no sea suficiente.

### 2.1.3. Discrete-Event Simulation

La simulación *Discrete-Event* se refiere al modelado de un sistema a medida que evoluciona el tiempo por una representación, en donde los estados cambian instantáneamente en puntos separados del tiempo i.e. el sistema cambia únicamente en un número finito de puntos en el tiempo. [LK91]

Este concepto se entiende mejor mediante un ejemplo:

*Se supone que se desea simular un proceso de cajas de un supermercado, mediante eventos discretos se puede simular las interacciones cliente y cajero. En este caso, las entidades serían los clientes individuales, con sus características como número de productos o tiempo de pago, que llegan al supermercado. Algunos ejemplos de eventos discretos podrían ser la llegada de un cliente, inicio-fin de un pago o apertura de una nueva caja. Durante la simulación, se recopilan datos que ayudan a evaluar y optimizar el rendimiento del sistema.*

Imagínense que toman una captura de su simulación en un determinado instante de tiempo  $t$ , esta captura no solo incluirá el estado del sistema en  $t$  sino que también incluirá una lista de futuros eventos ordenados por tiempo (*FEL*).

El mecanismo para avanzar la simulación y garantizar que todos los eventos ocurren en el momento cronológico correcto se basa en la lista de futuros eventos (*FEL*). Programar futuros eventos significa que, en el instante que una actividad comienza, su duración se computa como una muestra de una función de distribución. Esto se almacena en la *FEL* junto con el tipo de evento que describe. Por tanto, en un instante de tiempo  $t$ , la lista *FEL* está ordenada por tiempo de evento, donde este tiempo satisface

$$t \leq t_1 \leq t_2 \leq \dots \leq t_n$$

Después de que un evento en un tiempo  $t$  sea ejecutado, el tiempo de la simulación avanza a  $t_1$  y se crea una nueva captura, con su *FEL* y su evento a ejecutar. Este proceso se repite hasta que la simulación acabe. La secuencia de acciones que se realizan para avanzar el tiempo de simulación y crear nuevas capturas del sistema se llama *event-scheduling*. Esta última parte se basa en el primer capítulo de [Ric11].

## 2.2. Logística y Warehousing

### 2.2.1. Introducción a la logística

La logística se puede definir como el enfoque sistemático hacia la optimización integral de los sistemas de flujo, tanto de materiales como de información. Su principal objetivo es encontrar el rendimiento óptimo en el servicio logístico mediante la adecuada coordinación entre flujo físico de materiales (productos y recursos físicos) y el flujo de información, es decir, la comunicación e intercambio de datos relevantes.

El flujo de materiales abarca diversas actividades, desde la gestión de inventarios hasta el transporte y distribución, y se extiende a través de múltiples niveles, lo que se conoce como cadena de suministro. Históricamente, la cadena de suministro se centraba únicamente en la relación entre clientes y proveedores, lo que podía generar problemas como el *efecto látigo (bullwhip effect)*. Este fenómeno ocurre cuando pequeñas variaciones en la demanda de los consumidores se amplifican a lo largo de la cadena, causando ineficiencias y distorsiones. Hoy en día, este problema está más que resuelto gracias a los avances de la tecnología y estrategias como *ECR*<sup>1</sup> y *CPFR*<sup>2</sup> que permiten una mejor sincronización entre los elementos de la cadena.

Por último, cabe destacar el papel de la *E-Logistics*, que se refiere a la planificación, gestión y control de la cadena de suministro a través de internet. Este enfoque permite a las empresas interactuar con los clientes de manera online, facilitando procesos como el seguimiento de pedidos y la gestión de inventarios en tiempo real. La eficiencia en este sector es un factor crucial para el éxito de los proyectos y empresas en la actualidad, ya que errores graves en su implementación pueden llevar a pérdidas significativas o incluso a la quiebra.

### 2.2.2. Unidades básicas de un almacén (warehouse)

En un sistema de warehousing, las unidades básicas están presentes en diversas formas y combinaciones, según el sistema de logística implementado. A continuación, se describen los procesos fundamentales en la gestión de un almacén.

---

<sup>1</sup>Efficient Consumer Response es lo que se conoce como la respuesta directa al sistema de logística

<sup>2</sup>Collaborative, Planning, Forecasting and Replenishment propaga la vinculación de todos los elementos de la cadena para coordinar los pedidos y cuellos de botella

## Almacenamiento

El primer paso en el proceso de almacenamiento es determinar el espacio donde se va a ubicar el ítem. Esta decisión se puede tomar siguiendo muchos criterios como por ejemplo: optimización del volumen, minimizar el transporte del ítem, alta disponibilidad...

Para optimizar el proceso de almacenamiento, se aplican diversas estrategias, que pueden variar según el tipo de almacén y los requisitos específicos del producto (ver Tabla 2.2). Es importante que todas las estrategias cumplan con las regulaciones de seguridad y normativas legales.

Cuadro 2.2: Algunas estrategias de almacenamiento

Descripción	Estrategia	Objetivos
Fijar el hueco de almacenamiento	Fijar el hueco para un artículo concreto	Acceder con seguridad y acceso rápido para sistemas manuales
Almacenamiento aleatorio	El artículo se almacena en un hueco aleatorio libre	Maximizar la capacidad libre del almacén
Clustering de familias	Utilizar huecos adyacentes para artículos que se compran frecuentemente juntos	Incrementar el rendimiento de la recogida disminuyendo las rutas

## Recogida (picking)

La fase de recogida implica la extracción de los ítems necesarios para completar un pedido. La cantidad de ítems a recoger suele estar limitada para evitar congestiones en el proceso. El rendimiento en esta fase depende de los objetivos del almacén y de la estrategia de picking empleada (véase Tabla 2.3).

El gestor del almacén debe supervisar este proceso y proporcionar retroalimentación continua para mejorar la eficiencia. Una vez recogido el ítem, se actualiza el inventario y se libera el espacio ocupado en el almacén.

Cuadro 2.3: Algunas estrategias de recogida

Descripción	Estrategia	Objetivos
First In First Out (FI-FO)	Se recoge la primera unidad almacenada de ese ítem	Evitar el vencimiento de las unidades de ese ítem
Last In First Out (LI-FO)	Recogida de la última unidad almacenada de ese ítem	Reflejar los costos del inventario de una forma más realista
Rutas más cortas	Recoger las unidades del ítem que ofrezcan una ruta más corta	Mejorar el rendimiento y minimizar las rutas de recogida
Adaptación a la cantidad	Recogida acorde al volumen de pedidos	Aprovechar al máximo las unidades de carga

### **Empaquetado**

En almacenes grandes, los pedidos suelen estar compuestos por múltiples artículos almacenados en diferentes áreas. Para consolidar estos artículos, se utiliza una zona de empaquetado, donde se preparan los envíos finales.

El empaquetado es fundamental para reducir costes mediante la optimización del volumen, utilizando cálculos para seleccionar el embalaje más adecuado; sin embargo, es común que sea necesario realizar un reempaquetado, lo que puede disminuir la eficiencia obtenida previamente. La última tarea en esta área es verificar que el pedido esté completo y correcto, lo que destaca la importancia de esta zona dentro del almacén.

### **Zona de carga**

A primera vista, la función principal de la zona de carga es trasladar las mercancías al transporte asignado; sin embargo, además de las tareas físicas, este departamento incluye funciones de organización y control, como la planificación del espacio y la coordinación con el personal. Aunque el tiempo de carga puede ser corto, es frecuente que esta área se convierta en un cuello de botella, debido al espacio limitado y al número reducido de operarios. Por esta razón, es esencial una gestión eficiente de recursos para evitar demoras.

# Capítulo 3

## Herramienta de simulación: AnyLogic

En esta sección, se abordará el uso de AnyLogic como herramienta de simulación en nuestro proyecto de *warehousing*. Como punto de partida, es importante comprender qué es AnyLogic y por qué se seleccionó para abordar los desafíos de modelado y simulación en este contexto. Es relevante mencionar que, al inicio de este proyecto, no tenía conocimientos sobre AnyLogic y tuve que embarcarme en un proceso de aprendizaje para familiarizarme con esta potente herramienta. Ahora, veamos qué es AnyLogic y las razones que respaldan su elección.

### 3.1. ¿Qué es?

AnyLogic es una plataforma líder para el modelado y la simulación de sistemas complejos, ampliamente utilizada en sectores como la logística y la gestión de almacenes. Su enfoque multi-metodología permite combinar modelos basados en agentes, simulación de eventos discretos y sistemas continuos dentro de un entorno integrado. Esta versatilidad facilita el análisis de procesos complejos, como los sistemas de *warehousing*.

Una de sus principales ventajas es su interfaz gráfica intuitiva, que incluye bibliotecas específicas para *warehousing*. Estas bibliotecas proporcionan elementos predefinidos, como estanterías, pallets, estaciones de trabajo, forklifts y operadores, que permiten construir modelos visualmente detallados y representativos del almacén.

Además, AnyLogic ofrece herramientas avanzadas de análisis y visualización que permiten identificar cuellos de botella y evaluar el impacto de diferentes polí-

ticas y estrategias. Estas capacidades son esenciales para tomar decisiones y optimizar procesos logísticos.

Un aspecto destacado de AnyLogic es su capacidad de integrarse con otras herramientas avanzadas, como *Nvidia* o *Bonsai*, que utiliza el Aprendizaje por Refuerzo. En este proyecto, esta integración fue clave para optimizar estrategias en tiempo real, mejorando la eficiencia operativa del almacén.

En resumen, AnyLogic es una plataforma robusta y versátil que, gracias a su enfoque multi-metodología y capacidades avanzadas, fue fundamental para el desarrollo de los modelos en este trabajo. Su aplicación permitió representar fielmente los procesos del almacén y evaluar mejoras operativas.

### 3.2. ¿Por qué AnyLogic?

En este apartado, se aborda una cuestión clave: ¿Por qué elegir AnyLogic como herramienta de simulación en este proyecto? En esta sección, se detallan las ventajas y beneficios que hacen de AnyLogic una opción ideal para modelar procesos en el ámbito del warehousing.

Este trabajo se llevó a cabo durante mis prácticas en *Accenture*, donde AnyLogic ha sido ampliamente utilizado en proyectos de supply chain. Además, el director de este trabajo cuenta con más de 10 años de experiencia en el uso de esta herramienta, lo que permitió aportar al proyecto una perspectiva sólida y basada en la experiencia práctica.

La combinación de una herramienta avanzada y el respaldo de un entorno profesional con experiencia brindaron al proyecto una base técnica sólida y acceso a enfoques probados en problemas complejos de simulación logística.

#### 3.2.1. Enfoque multi-metodología

En simulación, se dice que un método es la infraestructura general para pasar de un sistema real al modelo. Este enfoque se refiere a la capacidad que tiene la plataforma para combinar y utilizar diferentes métodos en un mismo modelo de simulación. AnyLogic permite tres enfoques: basado en agentes, eventos discretos y sistemas dinámicos continuos.

### **Enfoque basado en agentes**

Este método modela el comportamiento individual de entidades autónomas dentro del sistema, como operarios, vehículos o paquetes. En el contexto del almacén, permite simular la interacción entre agentes y cómo estas afectan el rendimiento general, por ejemplo, optimizando rutas o evaluando la eficiencia de los operadores.

### **Enfoque basado en eventos discretos**

En la sección 2.1.3, se detalla más profundamente este enfoque. Este método se utiliza para representar procesos que ocurren en momentos específicos, como la llegada de camiones, la carga de mercancías o el procesamiento de pedidos. Es fundamental para modelar flujos logísticos, ya que permite analizar cuellos de botella y tiempos de espera en cada etapa del proceso.

### **Enfoque basado en sistemas dinámicos continuos**

Este método modela cambios graduales y continuos en el tiempo, como el flujo de materiales dentro del almacén, el nivel de inventario o la utilización de recursos. En este trabajo, fue clave para analizar variables como la capacidad de almacenamiento y las restricciones operativas.

La capacidad de combinar estos tres enfoques en un solo modelo permitió analizar el almacén desde diferentes perspectivas: la interacción entre agentes, la eficiencia de procesos y los flujos continuos de recursos. Esto fue esencial para identificar oportunidades de mejora y optimizar tanto el rendimiento operativo como la utilización de recursos en el contexto del warehousing.

## **3.2.2. Interfaz gráfica de usuario**

La interfaz gráfica de usuario de AnyLogic es uno de los elementos clave que facilita la creación y personalización de modelos de simulación. Sus características destacan por su simplicidad y capacidad de adaptarse a las necesidades específicas de proyectos logísticos complejos, como el warehousing.

### **Construcción intuitiva mediante arrastrar y soltar**

La funcionalidad de arrastrar y soltar permite a los usuarios seleccionar elementos de una biblioteca predefinida y ubicarlos directamente en el diseño. Esta característica no solo simplifica la creación del modelo, sino que también acelera



Gracias a su interfaz gráfica intuitiva y sus herramientas especializadas, AnyLogic permite a los usuarios diseñar modelos precisos y detallados de manera eficiente. En este trabajo, estas características fueron esenciales para representar con fidelidad las operaciones del almacén y analizar diferentes escenarios operativos, contribuyendo significativamente al éxito del proyecto.

### **3.2.3. Integración con otras herramientas**

AnyLogic se integra fácilmente con otras plataformas, lo que amplía sus capacidades de modelado y simulación. En este proyecto, se utilizó AnyLogic como entorno de simulación para un problema de Aprendizaje por Refuerzo mediante Bonsai. Gracias a la integración nativa entre ambas herramientas, fue posible optimizar procesos logísticos en el almacén de manera eficiente.

Además, AnyLogic permite integrarse con sistemas de gestión de almacenamiento (WMS) y con fuentes de datos en tiempo real, lo que facilita la actualización dinámica de los modelos basados en datos operativos actuales. Esta capacidad de integrar diferentes sistemas aumenta la precisión de los modelos y optimiza las decisiones logísticas.

# Capítulo 4

## Modelos de Simulación

En este capítulo se presentará en detalle la creación de los dos modelos de simulación desarrollados en AnyLogic: el Modelo *Simple* y el modelo de *Warehousing*. El objetivo de esta sección es brindar al lector una comprensión completa de los componentes, lógicas y configuraciones implementadas en cada modelo.

### 4.1. Modelo Simple

En esta primera parte del capítulo se presenta el modelo denominado Simple, desarrollado como un primer paso para explorar las capacidades de *AnyLogic* y su integración con *Bonsai*. Dado que *AnyLogic* era una herramienta nueva para el autor, este modelo permitió familiarizarse con su entorno y funcionalidades. Además, sirvió como una prueba inicial para evaluar el comportamiento de *Bonsai* en entornos de simulación construidos con *AnyLogic*.

#### 4.1.1. Objetivo del modelo

El modelo *Simple* tiene como propósito principal simular un entorno básico del almacén. Este modelo representa fallos en cuatro servicios, cada uno de los cuales está sujeto a distintos tipos de mantenimiento con tiempos de reparación específicos. Además, los tiempos de procesamiento de los servicios varían, lo que introduce una complejidad y refleja un entorno real de almacenamiento.

El enfoque RL del modelo se centra en predecir y gestionar los mantenimientos de los servicios, aprendiendo a asignar agentes de manera dinámica a los servicios disponibles. Al inicio de la simulación, cada agente se asigna a un servicio específico; sin embargo, cuando un servicio entra en reparación, se genera una cola de espera, ya que los agentes no pueden procesarse hasta que dicho servicio esté

operativo nuevamente.

El objetivo es que Bonsai aprenda a distribuir los agentes entre los servicios activos para minimizar las acumulaciones de espera y maximizar el rendimiento global del sistema. Esto incluye decisiones basadas tanto en los tiempos de procesamiento como en la disponibilidad de los servicios.

De este modo, el modelo no solo sirve como primer acercamiento para familiarizarse con las herramientas AnyLogic y Bonsai, sino que también explora su capacidad para mejorar la gestión de recursos en un escenario simplificado pero representativo de problemas logísticos reales.

### 4.1.2. Componentes Principales



Figura 4.1: Icono de Agente

El elemento principal que recorrerá el modelo de simulación será un agente denominado "paquete", el cual representa una unidad de almacenamiento perteneciente al almacén simulado. Este paquete será sometido a procesamiento por uno de los servicios disponibles en el sistema.

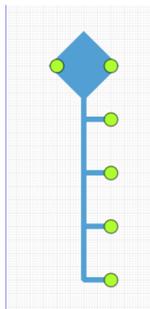


Figura 4.2: Icono de Select Output 5

El bloque *Select Output 5* se emplea para asignar a cada agente un servicio específico en el sistema de almacenamiento. La probabilidad asociada a cada salida determina la probabilidad de que el agente sea asignado a ese servicio en particular. Esto permite una asignación equitativa y dinámica de los paquetes, reflejando la realidad del proceso de asignación en el sistema de almacenamiento.

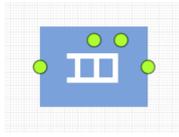


Figura 4.3: Icono de la Queue

La Queue en el modelo de simulación representa una cola ficticia en la que los agentes esperan su turno para ser procesados por el servicio correspondiente.

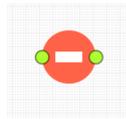


Figura 4.4: Icono del Hold

El bloque *Hold* se utiliza para bloquear el flujo de agentes a través de su salida correspondiente cuando el servicio asociado está en estado de mantenimiento. Esta simulación permite capturar el impacto de las reparaciones en el sistema de almacenamiento, evaluando el tiempo de espera de los agentes y la capacidad de recuperación del servicio.



Figura 4.5: Icono del Servicio

El bloque *Service* se encarga de simular el proceso de servicio en el modelo de simulación. A través de este bloque, se establece la duración o tiempo de procesamiento requerido para atender a los agentes de paquete, permitiendo evaluar el rendimiento y la eficiencia de los servicios dentro del sistema de almacenamiento.



Figura 4.6: Icono de la maquina

El bloque *Resource Pool* se utiliza para simular la máquina o unidad física que realiza el servicio dentro del modelo de simulación. Estas unidades son esenciales para llevar a cabo el procesamiento de los agentes de paquete. La presencia de estas unidades físicas permite aplicar un mantenimiento al servicio, lo que permite evaluar el impacto y la eficiencia del sistema de almacenamiento en situaciones de fallo.



Figura 4.7: Icono de la averia

El bloque *Downtime* se utiliza para simular los mantenimientos y sus propiedades en el modelo de simulación. Permite configurar el tipo de mantenimiento y su duración, y aplica el mantenimiento a una unidad específica del *Resource Pool*.

### 4.1.3. Interacciones entre los componentes

La simulación comienza con la generación de agentes que representan los elementos a procesar. Estos agentes son creados por el componente *Source*, que está

conectado al bloque *SelectOutput5*. Aunque el bloque dispone de cinco posibles salidas, en este modelo se utilizan únicamente cuatro, cada una representando un camino hacia uno de los servicios disponibles. La asignación de agentes a las salidas se realiza mediante probabilidades predefinidas. Una vez asignados, los agentes siguen exclusivamente el flujo hacia el servicio correspondiente.

Cada salida del *SelectOutput5* conduce a un bloque *Queue*, que gestiona una cola de espera bajo una política FIFO ("Primero en entrar, primero en salir"). Posteriormente, el flujo pasa al bloque *Hold* (4.4), cuyo objetivo es determinar si el agente puede avanzar o si debe permanecer bloqueado en caso de que el servicio correspondiente se encuentre en mantenimiento.

Cuando el agente es liberado del bloque *Hold*, llega al bloque *Service*, donde se lleva a cabo el procesamiento. Cada servicio está asociado a un recurso específico y cuenta con un tiempo de procesamiento definido. Los agentes pasan un tiempo en este bloque según el recurso asignado y su tiempo de procesamiento. Una vez procesados, los agentes finalizan su flujo y son eliminados del sistema.

Además de los componentes del flujo principal, el modelo incluye dos bloques auxiliares:

1. **Resource Pool:** Representa la máquina que realiza el servicio en cada uno de los bloques de procesamiento. En este caso, no tiene funcionalidad adicional más allá de simbolizar dicho recurso.
2. **DownTime:** Simula el mantenimiento de los servicios. Configurado como un tipo de fallo, este bloque introduce averías en el sistema. Las fallas ocurren tras un número específico de agentes procesados (100, 150, 200 y 250 respectivamente), imitando el comportamiento real de máquinas sujetas a ciclos de averías.

Estas interacciones combinan elementos básicos y dinámicos, proporcionando un entorno simplificado pero representativo de escenarios reales en el contexto de warehousing.



Figura 4.8: Modelo Básico

#### 4.1.4. Variables y parámetros

El modelo utiliza variables y parámetros clave para definir y controlar el comportamiento del sistema simulado. Estos son:

- **Número de agentes en las colas:** Variable que representa la cantidad de agentes en espera en las colas en un instante dado. Sirve como indicador de la carga actual del sistema y varía dinámicamente durante la simulación conforme los agentes son procesados.
- **Probabilidades de las salidas del Select Output:** Parámetros que determinan la probabilidad de que un agente tome una ruta específica al salir del bloque *SelectOutput*. En este modelo, cada salida tiene inicialmente una probabilidad predefinida, por ejemplo, 0.25 para las cuatro rutas activas, lo que implica una distribución equitativa del 25 % entre ellas.
- **Frecuencia de creación de agentes en el Source:** Parámetro que define la tasa de generación de nuevos agentes. En este caso, se establece en 300 agentes por hora, lo que refleja una frecuencia sintética basada en datos referenciales de un almacén real.

Estas variables y parámetros son esenciales para configurar el modelo y simular diferentes escenarios operativos. Ajustar estos valores permite explorar variaciones en la carga, el flujo y la distribución de recursos, proporcionando información valiosa sobre el comportamiento del sistema bajo distintas condiciones.

## 4.2. Modelo Warehousing

### 4.2.1. Objetivo del modelo

El objetivo de este modelo es simular las operaciones de un almacén que realiza el proceso completo de recepción, almacenamiento, recogida, empaquetado y despacho, según su destino, de paquetes. El almacén se organiza en diferentes zonas, comenzando por la zona de carga, donde los paquetes son recibidos desde cuatro áreas de llegada: Norte, Sur, Este y Oeste, simulando la distribución de los camiones provenientes de distintas ubicaciones geográficas.

Una vez descargados, los paquetes se almacenan aleatoriamente en los 20 racks del almacén, cada uno con 8 bays y 1 estante(shelf). El almacenamiento aleatorio mejora la utilización del espacio y reduce los tiempos de búsqueda, optimizando la eficiencia operativa del almacén. Según la bibliografía consultada, el almacenamiento aleatorio ha demostrado mejorar la utilización del espacio y reducir los tiempos de búsqueda de productos, lo que se traduce en una mayor eficiencia en la gestión del almacén [PA04][dLR07]

El siguiente paso en el proceso es la recogida de productos para formar los pedidos, los cuales se transportan a la zona de empaquetado. En esta zona, tres puestos de empaquetado, cada uno ocupado por dos trabajadores, se encargan de empaquetar los pedidos y enviarlos por medio de una cinta transportadora hacia la zona de salida, que también está dividida en cuatro partes, según el destino de cada paquete.

Durante este ciclo de almacenamiento y despacho, se simulan la llegada de pedidos, los cuales determinan los productos específicos que deben ser recogidos del almacén para satisfacer cada pedido.

El propósito de esta simulación es crear un entorno adecuado para la implementación de un modelo de Aprendizaje por Refuerzo (RL), cuyo objetivo principal es optimizar la gestión de las olas de pedidos en el almacén. El modelo de RL se enfoca en reducir el tiempo total transcurrido desde que un pedido es recibido hasta que es despachado, un factor clave que impacta directamente en la programación de los horarios de llegada de los pedidos y los camiones destinados a diferentes ubicaciones dentro del almacén.

El modelo de RL se diseñará para aprender y mejorar de manera continua las estrategias de gestión de los pedidos, basándose en las interacciones con el entorno

simulado. De esta forma, el modelo será capaz de ajustar sus decisiones para mejorar la eficiencia operativa del almacén. El objetivo final será encontrar una política óptima que permita tomar decisiones precisas y eficientes, acelerando el proceso global desde la recepción de un pedido hasta su despacho.

Cada pedido en el sistema estará asociado a un destino específico, y el modelo de RL deberá aprender a tomar decisiones sobre qué productos recoger del almacén, así como la asignación eficiente de estos productos a los camiones adecuados, con el fin de minimizar el tiempo total del proceso.

### 4.2.2. Componentes principales

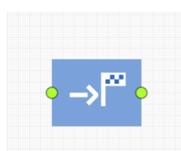


Figura 4.9: Icono de Move To

El bloque *MoveTo* en AnyLogic se utiliza para controlar y simular el movimiento de un agente o entidad a una ubicación específica en el modelo. Este bloque permite especificar las coordenadas o la entidad destino a la que se desea mover el agente. En este caso se utilizara para simular la llegada y partida de los camiones.

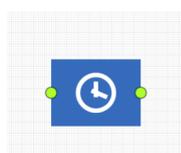


Figura 4.10: Icono de Delay

El bloque *Delay* se utiliza para introducir un retraso o pausa en el flujo del modelo durante un período de tiempo determinado. Este bloque permite simular el tiempo que tarda una entidad o proceso en completarse antes de continuar con la siguiente etapa del modelo. En este caso se emplea para simular la carga y descarga de los camiones.

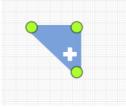


Figura 4.11: Icono de Split

El bloque *Split* se utiliza para dividir el flujo de agentes o entidades en diferentes ramas o caminos en función de ciertos criterios o condiciones. Este bloque permite simular la bifurcación del flujo de trabajo o de procesos en el modelo, creando múltiples rutas para los agentes en función de las reglas establecidas. El criterio en nuestra simulación, es obtener del objeto camión, el objeto paquete y lo mismo con el pedido.

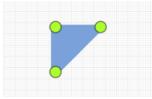


Figura 4.12: Icono de Combine

El bloque *Combine* se utiliza para fusionar o combinar diferentes flujos de entidades en un solo flujo. Este bloque permite reunir entidades que han seguido caminos separados o han sido procesadas de manera independiente, para que puedan continuar juntas en el modelo. La fusión será entre los agentes producto almacenado y producto que proviene del pedido.



Figura 4.13: Icono de Wait

El bloque *Wait* se utiliza para introducir una pausa o retraso en el flujo del modelo durante un período de tiempo determinado. Este bloque permite simular períodos de espera o inactividad en el modelo, donde las entidades no realizan ninguna acción o proceso.



Figura 4.14: Icono de Store

El bloque *Store* se utiliza para representar una ubicación de almacenamiento o depósito en el modelo. Este bloque permite simular el almacenamiento y la retención de entidades en una ubicación específica dentro del sistema.



Figura 4.15: Icono de Retrieve

El bloque *Retrieve* se utiliza para obtener entidades específicas de una fuente o almacenamiento en el modelo. Este bloque permite recuperar entidades del almacén que cumplen ciertos criterios o condiciones, para que puedan ser utilizadas o procesadas en el flujo del modelo.



Figura 4.16: Icono de Convey

El bloque *Convey* se utiliza para simular el transporte o movimiento de entidades de un lugar a otro dentro del modelo. Este bloque permite modelar el flujo de las entidades a través de cintas transportadoras.



Figura 4.17: Icono de Convey to Exit

El bloque *Conveyor Exit* se utiliza para representar la salida o finalización del transporte de entidades a través de una cinta transportadora.

Además de todas estas componentes, también se utilizaron los siguientes elementos: la Queue (Figura 4.3), el Service (Figura 4.5), el SelectOutput5 (Figura 4.2), el Hold (Figura 4.4) y el Resource Pool" (Figura 4.6),

### 4.2.3. Interacciones entre las componentes

El flujo de la simulación se organiza en varios pasos que replican las operaciones dentro del almacén. En primer lugar, se simula la llegada de camiones que siguen un horario establecido. Los camiones se dirigen a un nodo específico, utilizando el bloque *MoveTo*. Al llegar al nodo, se activa el bloque *Delay* para simular el tiempo de descarga del camión.

Una vez descargados, los productos traídos por los camiones se separan utilizando el bloque *Split*. Por un lado, el agente camión se mueve fuera del almacén mediante otro bloque *MoveTo*. Por otro lado, los productos se envían al bloque *Wait*, donde permanecen hasta ser almacenados por los trabajadores simulados a través de un *Resource Pool*.

Una vez que los productos han sido almacenados, los agentes producto pasan al bloque *Wait* para esperar a ser recogidos y asignados a un pedido. Este proceso sincroniza la llegada de los camiones con la disponibilidad de los productos en el almacén.

En paralelo, se genera un flujo de agentes de tipo Pedido, que se crean siguiendo un calendario que simula las olas de pedidos. Estos pedidos esperan en el bloque *Wait* hasta que haya unidades disponibles de los productos solicitados. Cuando los productos están disponibles, se utiliza el bloque *Combine* para mezclar los agentes de producto del almacén con los de los pedidos.

Una vez combinados, los agentes producto esperan en una cola hasta ser recogidos por otro grupo de trabajadores, quienes los trasladan a la zona de empaquetado utilizando el bloque *Retrieve*. En la zona de empaquetado, el proceso de empaquetado de los pedidos se simula mediante el bloque *Service*.

Una vez empaquetados, los pedidos se colocan en una cinta transportadora. Los bloques *Convey* y *Conveyor Exit* simulan el transporte de los pedidos hacia la zona de carga. En esta zona, los pedidos se dividen según su destino utilizando el bloque *SelectOutput5*.

En este punto, los pedidos quedan bloqueados por el bloque *Hold* hasta que el camión asignado a cada destino llegue a la zona de carga. Este bloque *Hold* sincroniza la disponibilidad de los pedidos con la llegada de los camiones correspondientes.

El flujo de los camiones se repite de manera similar al flujo inicial. Una vez que un camión llega, se carga con los pedidos asignados y se simula su salida hacia su destino. Este ciclo de llegada, descarga, almacenamiento, empaquetado y carga de pedidos se repite de acuerdo con el horario establecido para los camiones. Los bloques descritos representan las diferentes etapas y procesos dentro del sistema simulado, desde la llegada de los camiones hasta la carga de los pedidos para su despacho.

#### 4.2.4. Variables y parámetros

Los parámetros del modelo están relacionados principalmente con la configuración de los trabajadores y las capacidades de transporte. Existen parámetros que permiten ajustar el número de trabajadores disponibles tanto en la zona de carga como en la de empaquetado. El número de trabajadores en la zona de empaquetado puede variar entre 2 y 4 por puesto, mientras que en la zona de carga, el número de trabajadores es fijo, pero estos parámetros determinan la distribución de trabajadores en cada destino, dependiendo de la cantidad de pedidos asignados.

Además, se establece un parámetro que define el número máximo de pedidos que un camión puede transportar. Este parámetro limita la capacidad de carga de los camiones y es esencial para la planificación de la distribución de los pedidos.

En cuanto a las variables, estas se dividen en dos grupos principales: variables de control y variables de configuración.

Las variables de control se utilizan para monitorizar y evaluar el estado del modelo durante la simulación. Entre estas variables se encuentran el conteo de pedidos pendientes de cargar en cada destino, la cantidad de paquetes que quedan por empaquetar y el número de pedidos restantes por procesar. Estas variables permiten obtener una visión general sobre el progreso y la eficiencia del sistema.

Por otro lado, las variables de configuración se encargan de gestionar las olas de pedidos y su flujo a través del sistema. Entre estas se incluyen la duración del pedido, que representa el tiempo transcurrido desde la llegada del pedido hasta que se carga, y la configuración de los pedidos a procesar, la cual se basa en la programación de los camiones entrantes. Por ejemplo, si se espera la llegada de camiones del Sur y del Oeste, los pedidos de esas regiones recibirán prioridad en el procesamiento. Además, existe una variable que permite liberar todos los pedidos en caso de que la carga de pedidos sea baja y no haya problemas para su procesamiento.

Tanto los parámetros como las variables son elementos clave que el modelo de aprendizaje por refuerzo utiliza para comprender el comportamiento del sistema, tomar decisiones informadas y aprender a controlar y optimizar las olas de pedidos y sus tiempos. Estos componentes son fundamentales para lograr el objetivo principal del proyecto: optimizar las operaciones de gestión de olas de pedidos y mejorar los tiempos de entrega.

# Capítulo 5

## Conceptos Básicos Reinforcement Learning

### 5.1. Introducción al Reinforcement Learning

En esta sección se introduce el Reinforcement Learning (RL), una rama del aprendizaje automático utilizada para resolver problemas mediante la interacción dinámica entre un agente y su entorno. Su aplicación en la optimización de procesos logísticos, como la agrupación de pedidos y la reducción de tiempos de procesamiento en almacenes simulados, demuestra su relevancia en el ámbito industrial.

#### 5.1.1. Definición

Reinforcement Learning consiste en aprender qué hacer en un entorno para maximizar una recompensa i.e. a partir de una situación qué acciones debo tomar para conseguir la máxima recompensa. Al agente no se le dice qué debe de hacer sino que debe de descubrir qué acciones son las que le llevarán a alcanzar esa máxima recompensa. La idea básica es simplemente capturar los aspectos más importantes del problema real al que se enfrenta el agente e interactuar para lograr el objetivo. [SB18]

El agente realiza acciones en diferentes estados, observa las respuestas del entorno (en forma de recompensas y cambios de estado), y ajusta su comportamiento para mejorar el rendimiento. Con el tiempo, el objetivo es desarrollar una política óptima que maximice las recompensas esperadas a largo plazo.

Un aspecto clave en RL es el equilibrio entre la *exploración* y la *explotación*. El agente debe explotar las estrategias conocidas que producen buenos resultados,

pero también debe explorar nuevas posibilidades para descubrir acciones potencialmente mejores. Este balance es fundamental para alcanzar soluciones óptimas en entornos complejos y dinámicos.

## **5.2. Componentes clave Reinforcement Learning**

En esta sección, se presentarán y explicarán los componentes fundamentales del Reinforcement Learning, que son esenciales para comprender cómo se aborda el problema de optimización de las olas de pedidos y tiempos de carga en el almacén simulado.

### **Agente, entorno y acciones**

En el contexto del Reinforcement Learning aplicado al almacén simulado, es fundamental comprender los conceptos de agente, entorno y acciones.

El agente se refiere a la entidad que interactúa con el entorno y toma decisiones en base a la información que recibe. El agente realiza acciones en el entorno y recibe observaciones y recompensas como retroalimentación. Su objetivo principal es maximizar la recompensa acumulada a lo largo del tiempo aprendiendo de su interacción con el entorno.

El entorno representa el sistema completo en el que opera el agente. El entorno proporciona información al agente a través de observaciones (por ejemplo, estado actual del inventario o tiempos de espera) y recompensas (por ejemplo, reducción del tiempo total de procesamiento). Además, el entorno evoluciona dinámicamente en respuesta a las acciones tomadas por el agente.

Las acciones son las decisiones que el agente toma en cada paso de interacción con el entorno. Para maximizar la recompensa acumulada, el agente debe aprender a tomar estas decisiones de manera óptima, equilibrando entre explorar nuevas estrategias y explotar aquellas que han demostrado ser efectivas.

### **Estado y observación**

El estado representa la información relevante sobre el entorno en un momento dado. Es una descripción completa del contexto actual que el agente utiliza para tomar decisiones óptimas. En el almacén simulado, el estado podría incluir variables como el número de pedidos pendientes, la disponibilidad de recursos en las áreas de almacenamiento, o el tiempo estimado de procesamiento para cada tarea. El objetivo principal es capturar solo los aspectos esenciales del entorno que

afectan directamente las acciones del agente.

Por otro lado, las observaciones son las señales que el agente recibe del entorno y que le permiten inferir o estimar el estado actual. Estas observaciones pueden ser datos directos, como la cantidad de pedidos en cola, o datos indirectos, como el tiempo promedio de espera, que requieren procesamiento adicional para derivar información útil.

### **Política**

La política es una función que define la forma en que un agente debe comportarse en un momento específico. Es una aplicación que mapea los estados del entorno a las acciones que el agente debe tomar en esos estados. La política es el núcleo central de un agente de aprendizaje por refuerzo, ya que por sí sola es suficiente para determinar el comportamiento del agente.[SB18]

La elección de una política adecuada depende de los objetivos y restricciones del problema. En el aprendizaje por refuerzo, el agente puede aprender a mejorar su política mediante la iteración continua de la interacción con el entorno, evaluando el desempeño y actualizando la política en base a las recompensas recibidas.

### **Recompensa y función valor**

La recompensa define el objetivo del agente. En cada paso de interacción con el entorno, el agente recibe un número, denominado recompensa, que refleja qué tan bueno o malo fue el evento o la acción realizada. Esta recompensa proporciona una señal de retroalimentación al agente y sirve como base primaria para ajustar su política de toma de decisiones. Si el agente sigue una política específica y recibe una recompensa baja, esto indica que la acción tomada no fue favorable, y en futuras situaciones similares, la política debería elegir otra acción.[SB18]

La función valor es una medida que indica cuán bueno es un estado a largo plazo. El valor de un estado específico se refiere a la recompensa total que el agente puede esperar acumular si comienza en ese estado y sigue una determinada política. Esta función valor permite al agente evaluar y comparar diferentes estados en términos de su potencial de recompensa acumulada. Al maximizar el valor de los estados, el agente busca encontrar una política óptima que le permita alcanzar las mayores recompensas posibles a lo largo del tiempo

## 5.3. Algunos Algoritmos de Reinforcement Learning

En esta sección, se presentan dos métodos avanzados RL: APEX-DQN y PPO. Estos algoritmos se destacan porque son los que pueden ser utilizados por Bonsai.

### 5.3.1. APEX-DQN

El algoritmo *APEX-DQN* es una mejora del algoritmo original *Deep Q-Network (DQN)*. El objetivo es superar algunas de sus limitaciones, como la falta de eficiencia en la utilización de recursos computacionales y la necesidad de una gran cantidad de datos para el entrenamiento.

DQN es un método de RL que combina redes neuronales convolucionales y técnicas de optimización para entrenar a agentes en entornos complejos. DQN utiliza una red neuronal para aproximar una función de valor Q, que estima la recompensa esperada para cada posible acción en un estado dado. El agente interactúa con el entorno, selecciona acciones basadas en una política de exploración y explotación, y utiliza la experiencia acumulada para actualizar iterativamente la red neuronal y mejorar las estimaciones de Q. DQN también utiliza una técnica llamada *Replay Memory* para almacenar y muestrear experiencias pasadas, lo que ayuda a decorrelacionar los datos de entrenamiento y mejorar la eficiencia del aprendizaje. Para más detalles de este método consultar [Mni+15]

#### APEX

La mejora principal de *APEX-DQN* es su enfoque en la distribución de la experiencia de aprendizaje entre múltiples actores <sup>1</sup>. APEX-DQN utiliza un grupo de actores en paralelo para generar experiencias, lo que permite generar experiencias a partir de una variedad de estrategias, así se consigue priorizar elegir las experiencias más efectivas.

Otra mejora importante es el uso de una compartida, centralizada y priorizada replay memory. En lugar de seleccionar aleatoriamente experiencias pasadas para el entrenamiento, APEX-DQN asigna prioridades a las experiencias según su importancia para el aprendizaje. Esto significa que las experiencias más prometedoras se seleccionan con mayor frecuencia para el entrenamiento, lo que acelera el proceso de aprendizaje.

---

<sup>1</sup>Siguen una política  $\epsilon$ -greedy, con distintos valores de  $\epsilon$

Estas son las dos cualidades más importantes del método *APEX-DQN*, para mas información [Hor+18]

### 5.3.2. Proximal Policy Optimization

A diferencia de los métodos basados en la función valor, como Q-Learning, los métodos *Policy Gradient* se centran directamente en la optimización de la política i.e. en encontrar una estrategia de toma de decisiones óptima para el agente. Estos métodos utilizan el gradiente de una función objetivo que está relacionada con el rendimiento esperado del agente y realizan actualizaciones de los parámetros de la política utilizando SGD<sup>2</sup>. Al actualizar la política mediante el gradiente, los métodos de gradiente de política buscan mejorar el rendimiento del agente en función de las recompensas obtenidas en el entorno de aprendizaje.

#### PPO

Obtener buenos resultados con policy gradient methods es difícil ya que son muy sensibles a la elección del *steepsizes*. Para evitar cambios drásticos en la política que puedan afectar la estabilidad del aprendizaje, PPO utiliza una medida de proximidad entre la política anterior  $\pi_{\theta_{Old}}$  y la política actualizada  $\pi_{\theta}$ . Para limitar las actualizaciones de política, se utiliza una función de pérdida PPO que incluye dos componentes principales: un término de rendimiento esperado y un término de divergencia entre las políticas.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon)\hat{A}_t)] \quad (5.1)$$

Donde:

- $\theta$  es el parámetro de la política
- $\hat{E}_t$  denota la esperanza empírica
- $r_t$  es el ratio de la probabilidad entre la nueva y la antigua política, respectivamente.
- $\hat{A}_t$  es la ventaja estimada en el tiempo  $t$
- $\varepsilon$  hiperparametro que suele estar entre 0.1 y 0.2

---

<sup>2</sup>Stochastic Gradient Descent

La función de clipp compara el ratio con dos límites:  $1 - \epsilon$  y  $1 + \epsilon$ . Si el ratio está dentro de este rango, no se realiza ninguna modificación. Sin embargo, si el ratio está fuera de este rango, se ajusta al límite correspondiente.

El objetivo de utilizar la función de clipp es limitar las actualizaciones de política en un rango acotado alrededor de la política anterior. Esto evita cambios drásticos que podrían afectar la estabilidad del aprendizaje y ayuda a mantener un progreso gradual y estable en la mejora del rendimiento del agente. La función de pérdida PPO se maximiza utilizando SGD para ajustar los parámetros  $\theta$  de la política.

Para profundizar en PPO se recomienda [Sch+17] y para los conceptos de Policy Gradient [SB18]

# Capítulo 6

## Bonsai

### 6.1. ¿Qué es Bonsai?

Bonsai es una plataforma desarrollada por Microsoft que se define como: *una solución de inteligencia artificial (IA) de bajo código (low-code) que se especializa en el uso de simulaciones y técnicas de enseñanza automática (machine teaching)*. Este proyecto permite a los usuarios desarrollar soluciones eficientes y de alta calidad sin requerir un conocimiento avanzado en programación o en algoritmos de aprendizaje automático.

La característica distintiva de Bonsai radica en su capacidad para democratizar el desarrollo de soluciones de IA. A través de una interfaz gráfica intuitiva y herramientas visuales, los usuarios pueden diseñar sistemas de control complejos sin necesidad de escribir grandes volúmenes de código. Este enfoque es especialmente útil en proyectos donde la optimización y la toma de decisiones son clave, como en simulaciones logísticas.

El entrenamiento con Bonsai tiene los siguientes componentes principales, que trabajan conjuntamente: simulación de entrenamiento (training simulation), motor de entrenamiento (training engine), currículum de entrenamiento (training curriculum) y el cerebro.

#### 6.1.1. ¿En que se diferencia?

La plataforma ofrece una serie de ventajas que la distinguen de otras plataformas de aprendizaje por refuerzo y simulación. Estas características la convierten en una herramienta altamente efectiva, incluso para usuarios con poca experiencia en este mundo. A continuación, se destacan las principales diferencias:

- **Simplicidad y accesibilidad:** Bonsai se destaca por su enfoque en facilitar el desarrollo de modelos de aprendizaje por refuerzo. Gracias a su interfaz gráfica intuitiva y herramientas de desarrollo low-code. Esto resulta especialmente útil en entornos empresariales, donde los equipos multidisciplinarios necesitan colaborar en la resolución de problemas complejos.
- **Integración con simuladores:** La plataforma permite una integración fluida con simuladores como AnyLogic, creando un entorno virtual en el que los agentes de aprendizaje pueden practicar y optimizar sus estrategias.
- **Infraestructura escalable y robusta:** Bonsai se ejecuta sobre la nube de Microsoft Azure, ofreciendo escalabilidad y alta disponibilidad para proyectos de cualquier tamaño. Esto asegura que los entrenamientos puedan manejar grandes volúmenes de datos de simulación sin comprometer el rendimiento.

## 6.2. Componentes principales

Bonsai tiene como objetivo principal simplificar el entrenamiento de sistemas inteligentes mediante el uso de deep reinforcement learning. La plataforma se puede desglosar en tres fases fundamentales:

1. **Integración:** Durante esta fase, los simuladores y modelos que representan problemas reales son configurados e integrados en Bonsai. Esto crea un entorno donde los agentes pueden interactuar y aprender de manera controlada.
2. **Entrenamiento:** En esta etapa, los agentes son entrenados definiendo metas, objetivos y estados reales. A través de interacciones repetidas con el entorno simulado, Bonsai ajusta los parámetros del modelo para optimizar su rendimiento.
3. **Despliegue:** Una vez entrenado, el cerebro optimizado se exporta para su implementación, ya sea en infraestructura local (on-premise) o en la nube (cloud), permitiendo su uso en escenarios reales.

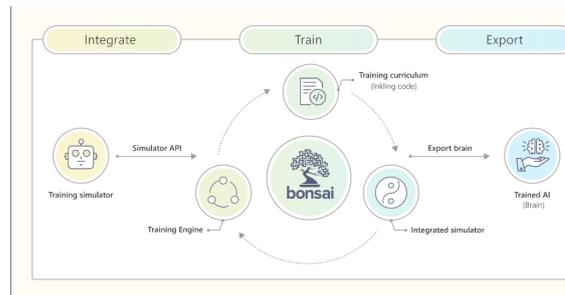


Figura 6.1: Esquema propio de la plataforma Bonsai

### 6.2.1. Entorno y Simulación

Las simulaciones de entrenamiento en Bonsai están diseñadas para replicar sistemas del mundo real y proporcionar un contexto en el que el cerebro entrenado pueda aprender. Estas simulaciones funcionan procesando y modificando su estado en función de las acciones que el cerebro toma. A través de este ciclo de interacciones, el agente (el cerebro) aprende a tomar decisiones y a mejorar sus comportamientos.

Para que una simulación sea efectiva en Bonsai, debe cumplir con las siguientes características:

- **Entorno bien definido:** debe estar claramente definido y ser accesible en cada paso de la simulación, lo que permite que el cerebro interactúe con él de manera coherente.
- **Acciones discretas:** Debe existir un conjunto de acciones que el cerebro pueda tomar para afectar el estado del sistema, permitiendo que el agente aprenda qué decisiones conducen a los mejores resultados.
- **Estado terminal:** La simulación debe ser capaz de determinar cuándo el sistema alcanza un estado en el que ya no es posible continuar (un estado terminal), lo que permite que el entrenamiento se ajuste en consecuencia.
- **Estado de éxito:** Es fundamental poder identificar cuándo el sistema alcanza un estado de éxito, lo que marca el cumplimiento de los objetivos de aprendizaje del agente.

En este proyecto se han utilizado dos modelos, presentados en el capítulo 4, realizados con AnyLogic que es socio de Microsoft y posee una fácil implementación de sus modelos.

### 6.2.2. Proceso de aprendizaje:

El motor de entrenamiento de Bonsai es el núcleo del proceso de aprendizaje, compuesto por varios componentes que trabajan de manera conjunta para entrenar a los agentes. Estos componentes son:

- **Architect:** es responsable de generar y optimizar los modelos de aprendizaje, basándose en el currículum definido para el entrenamiento.
- **Instructor:** coordina el entrenamiento del *Learner* (agente), utilizando los datos de la simulación y los objetivos del currículum.
- **Learner:** es el componente que ejecuta los algoritmos de aprendizaje y, a través de interacciones continuas con el entorno, se ajusta para mejorar su rendimiento. Durante el entrenamiento, trabaja en conjunto con el *Instructor* para establecer los parámetros iniciales y evaluar las respuestas.
- **Predictor:** representa el cerebro entrenado. Una vez que el modelo ha sido optimizado, se puede exportar y utilizar en un modo de predicción.

#### Architect

La función principal del Architect es crear y optimizar las redes neuronales que se basan en el currículum, es decir, se encarga de evaluar la efectividad de las redes neuronales. Basándose en el currículum, el Architect propone una configuración de algoritmos y redes que ofrecen el mejor aprendizaje del entorno. Los algoritmos que soporta, que se introdujeron en la sección (5.3), son: *Distributed Deep Q Network*, *proximal Policy Optimization* y *Soft Actor Critic*.

#### Instructor

El Instructor lleva a cabo el plan de entrenamiento configurando el Learner y las fuentes de datos requeridas en función de las necesidades del currículum. Si bien algunas operaciones se realizan por lotes, el Instructor está diseñado para trabajar de forma interactiva. Responde en tiempo real a medida que el agente recorre el entorno, calcular una respuesta, ser evaluado y aprender del resultado.

#### Learner

El Aprendiz lleva ejecuta los algoritmos seleccionados por el Architect. Durante el entrenamiento, el Learner se coordina con el Instructor para establecer los parámetros iniciales del algoritmo de aprendizaje, luego determina una respuesta y califica su rendimiento.

## Predictor

El Predictor es esencialmente un cerebro entrenado. Una vez entrenado, el algoritmo de IA se aloja en modo de predicción. El modo de predicción tiene un cerebro para usar como un punto final de API para que los usuarios puedan enviar datos de entrada al cerebro para obtener una predicción.

### 6.2.3. Curriculum

Las metas y los objetivos dan forma al curriculum. Le permiten utilizar su experiencia para crear objetivos clave de un problema complejo y que la IA sea capaz de aprender. Las metas son una especificación de alto nivel de lo que se pretende que aprenda el sistema. Por otro lado, los objetivos son directivas como evitar o minimizar, que el motor utiliza para medir como esta aprendiendo la IA.

Bonsai utiliza un lenguaje propio llamado Inkling para codificar metas y objetivos. El código Inkling define qué (y cómo) quieres enseñarle a tu IA. Una expresión en código Inkling es cualquier entidad sintáctica que se puede evaluar para determinar su valor. Estas entidades pueden incluir comentarios, identificadores, palabras clave, literales y operadores. Para más información sobre este lenguaje acudir a [Tea22].

### 6.2.4. Bonsai Brains

Un Bonsai Brain es un modelo de IA entrenado para controlar y optimizar sistemas del mundo real. Utiliza Deep Reinforcement Learning y simulaciones para aprender a tomar decisiones y mejorar su rendimiento a lo largo del tiempo. Durante el entrenamiento, el currículum establece metas y objetivos que guían el aprendizaje del cerebro. A medida que interactúa con el entorno simulado, el cerebro ajusta sus acciones para alcanzar los mejores resultados posibles. Estos pueden ser fácilmente exportados y utilizados en otras plataformas o sistemas, lo que permite integrarlos en entornos reales. Además, se pueden probar y ajustar a través de la API de Bonsai.

## 6.3. Integración y aplicación en el proyecto

### 6.3.1. Preparación y carga del simulador

Para integrar el modelo de AnyLogic con Bonsai RL, es necesario configurar el experimento de reinforcement learning (RL) dentro del modelo. Este proceso involucra definir tres secciones clave: Observation, Action y Configuration.

#### Observation

En esta sección, se definen las variables clave que el agente necesita para tomar decisiones durante el entrenamiento. Estas observaciones pueden incluir datos como el tiempo entre la llegada de un pedido y su carga, el número de pedidos pendientes, o la cantidad de paquetes restantes por empaquetar.

Por ejemplo, en el modelo de Warehousing (presentado en el Capítulo 4.2), en esta sección podrían incluirse variables como la diferencia de tiempo entre la llegada de un pedido y su carga, el número de pedidos pendientes para cada destino, la cantidad de paquetes restantes por empaquetar o cargar, o cualquier otro dato que sea relevante para el problema específico que se está abordando.

Es crucial elegir las observaciones que proporcionen la información necesaria para que el agente tome decisiones óptimas. Además, es posible aplicar transformaciones o preprocesamientos a las observaciones con el fin de mejorar la calidad de la información entregada al agente.

#### Action

Aquí se definen los valores que el agente utilizará para interactuar con el modelo. Cada vez que se da un paso en el experimento, los valores del modelo de AnyLogic, como la capacidad de los camiones o el número de trabajadores, se asignan a las variables de observación. Estas acciones guían la simulación y afectan el estado del sistema.

#### Configuration

Esta sección establece los parámetros iniciales antes de comenzar el entrenamiento. En modelos de AnyLogic sin configuraciones iniciales, la sección de configuración se centra en la preparación del agente para el entrenamiento de RL, sin alterar el modelo de simulación.

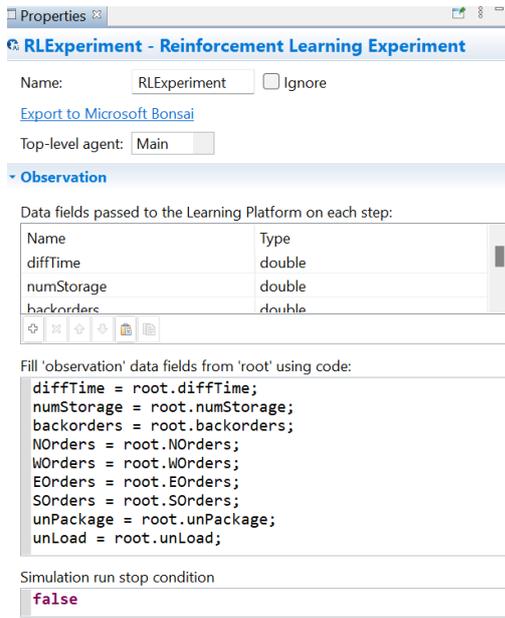


Figura 6.2: Ejemplo de la observation del modelo de warehousing

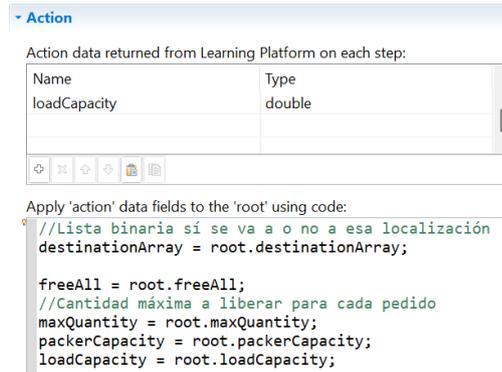


Figura 6.3: Ejemplo de la action en el modelo de warehousing

La carga del simulador en Bonsai es un proceso sencillo que se realiza pulsando el botón *Export as RLEExperiment* en el entorno de AnyLogic. Al hacerlo, se generará automáticamente un archivo zip que contiene todos los archivos necesarios para la simulación. Posteriormente, este archivo zip se puede cargar en Bonsai a través de la sección *Add Sim* en la plataforma. Este proceso permite transferir el modelo de simulación desarrollado en AnyLogic al entorno de Bonsai para llevar a cabo el entrenamiento de aprendizaje por refuerzo de manera integrada y eficiente.

### 6.3.2. Configuración y Entrenamiento con Bonsai

Una vez cargado el simulador en Bonsai, el siguiente paso es configurar el *Architect* para definir el cerebro del agente y establecer el curriculum de entrenamiento. En esta fase, se deben especificar los elementos clave: el *SimState*, que corresponde a la sección de observación, y el *SimAction*, que corresponde a la sección de acción del simulador.

Es importante diseñar tanto la función de recompensa (Reward) como la condición terminal (Terminal) en el código del Architect, ya que son requisitos obligatorios para crear el curriculum de entrenamiento. La función de recompensa asigna una puntuación al agente según su desempeño en el sistema, mientras que la condición terminal define las condiciones que indican el final de un episodio de entrenamiento.

La pregunta que se puede tener en este punto es: *¿entonces, para qué se han definido la observación y acción en el simulador?* Se hace porque aparte de ser obligatorio para la carga, una vez que se carga Bonsai te sugiere un código ya preestablecido con estas funciones creadas con esos valores.

Finalmente, en el bloque principal (graph) de este código, se insertan tanto el SimState como el SimAction<sup>1</sup>, y se define el curriculum de entrenamiento utilizando las funciones de recompensa y condición terminal.

### 6.3.3. Validación y Exportación del modelo

La última etapa del proceso consiste en exportar el cerebro entrenado en la sección anterior; sin embargo, para poder trabajar con el cerebro exportado, es necesario crear un contenedor Docker que contenga la imagen del mismo. En este proceso, se deben indicar dos elementos clave: el nombre del contenedor de Azure (ACR\_NAME) y el nombre e ID del espacio de trabajo correspondiente.

Una vez creado el contenedor Docker, se accede a él y se muestra una pantalla similar a la imagen proporcionada (6.5). A partir de este punto, se puede utilizar la API para realizar pruebas con nuevos datos y observar las predicciones que el cerebro realizaría para ese estado. Además, es posible establecer una conexión entre el cerebro exportado y AnyLogic a través de la conexión con el contenedor Docker.

---

<sup>1</sup>En caso de tener configuration se crea del mismo modo en el SimConfig

```
# Define a type that represents the per-iteration state
# returned by the simulator.
type SiniState {
  diffTime: number,
  numOrders: number,
  backorders: number,
  WOrders: number,
}

using Math
using Goal

# Define a type that represents the per-iteration action
# triggered by the simulator.
type SiniAction {
  maxQuantity: number<90 .. 110 step 1>,
  maxCapacity: number<6..12 step 1>,
  loadCapacity: number<0 .. 20 step 1>,
}

simulator Simulator(action: SiniAction): SiniState {
  # Randomly launch the simulator with this
  # registered package name.
  package "a_20-04"
}

function Terminal(obs: SiniState) {
  return obs.diffTime > 9000
}

function Reward(obs: SiniState) {
  if obs.diffTime < 3000 {
    # Si se cumple un paquete, se otorga un reward positivo
    return 10
  } else {
    # Si no hay cambios en la cantidad de paquetes cargados y el tiempo promedio no excede el límite, se otorga
    return 0
  }
}
```

Figura 6.4: Ejemplo del Inkling code del modelo de Warehousing



Figura 6.5: Ejemplo de la imagen del cerebro exportado en el Docker

# Capítulo 7

## Resultados y Conclusiones

### 7.1. Resultados Modelo Simple

El capítulo de resultados presenta los hallazgos y conclusiones derivados de la implementación de los modelos de simulación en los escenarios de los Modelos *Simple* y *Warehousing*.

#### 7.1.1. Configuración de entrenamiento

Configuración del entorno de entrenamiento, modelo de AnyLogic (4.1).

Se utilizó una configuración del entorno de simulación con las siguientes características:

- Se emplearon 4 servicios que representan máquinas, cada uno con un tiempo de procesamiento de 10 minutos. Estos servicios simulan el procesamiento de elementos en el sistema.
- Cada servicio cuenta con un programa de mantenimiento que se lleva a cabo después de un cierto número de elementos procesados. Los servicios tienen programados mantenimientos después de procesar 100, 200, 300 y 400 elementos, respectivamente.

Configuración del modelo de Reinforcement Learning:

Para el modelo de Reinforcement Learning se establecieron los siguientes parámetros y variables:

- **Observation:** La parte de observación del modelo incluyó las siguientes variables: la duración (tiempo que tarda en procesarse un paquete), la cantidad

de elementos en espera en las cuatro colas de los servicios y los tiempos de duración en cada servicio.

- **Action:** Las acciones posibles para el agente son las cuatro probabilidades correspondientes a la selección de uno de los servicios para procesar el elemento.
- **Configuration:** No se estableció ningún parámetro inicial en cada iteración. El entorno se mantuvo fijo una vez que fue cargado.
- **Reward:** Se definió una función de recompensa que se basó en dar una recompensa positiva si la duración estaba por debajo de un umbral aceptable. La fórmula de la recompensa es la siguiente:

$$\text{Reward} = \begin{cases} 1000 & \text{si Duración} < 10 \text{ min,} \\ -1000 & \text{Otro caso.} \end{cases}$$

- **Terminal:** Se consideró que la ejecución debe finalizar si la duración supera los 1440 minutos, que es el número de minutos en un día. Si ocurre esta condición, se asume que algo está fallando en la simulación y se detiene.

Como se mencionó en el capítulo teórico de Bonsai (Capítulo 6), el modelo de Bonsai lleva a cabo una serie de interacciones en una fase inicial para determinar el algoritmo óptimo. En este caso particular, el modelo seleccionó Proximal Policy Optimization (PPO 5.3.2) como el algoritmo más adecuado.

Algunas de las razones por las que Bonsai elige el algoritmo Proximal Policy Optimization (PPO) para el modelo *simple* pueden ser las siguientes:

1. Eficiencia en la utilización de datos: PPO es un algoritmo que aprovecha eficientemente los datos recopilados durante la simulación. Utiliza técnicas de actualización de políticas basadas en una relación de probabilidad de acciones, lo que permite una utilización óptima de los datos recopilados.
2. Balance entre exploración y explotación: PPO es capaz de encontrar un equilibrio adecuado entre explorar nuevas acciones y explotar las acciones que se consideran óptimas según las experiencias previas. Esto es crucial en un entorno de simulación donde se busca mejorar constantemente el tiempo de procesamiento y adaptarse a los cambios en las condiciones.
3. Capacidad para aprender políticas de alto rendimiento: PPO tiene una alta capacidad de aprendizaje y es capaz de converger hacia políticas de alto rendimiento en entornos complejos y variables. Esto es beneficioso en un

modelo de simulación donde pueden existir múltiples interacciones y dependencias entre las variables que influyen en el tiempo de procesamiento de los paquetes.

### 7.1.2. Resultados

El proceso de entrenamiento del modelo concluyó después de 3 horas y 14 minutos, durante los cuales se llevaron a cabo 750,273 iteraciones y 165 episodios. A continuación, se presentarán una serie de gráficas que ilustran el progreso del entrenamiento del modelo.

En primer lugar, se muestra la evolución de la recompensa a lo largo de los episodios de entrenamiento. Como se puede observar en la figura (7.1), inicialmente la recompensa se encontraba en -1.60 millones y, a medida que avanzaban los episodios, fue mejorando gradualmente hasta alcanzar una recompensa de 90.000. Este progreso evidencia claramente una mejora y aprendizaje significativos por parte del modelo.

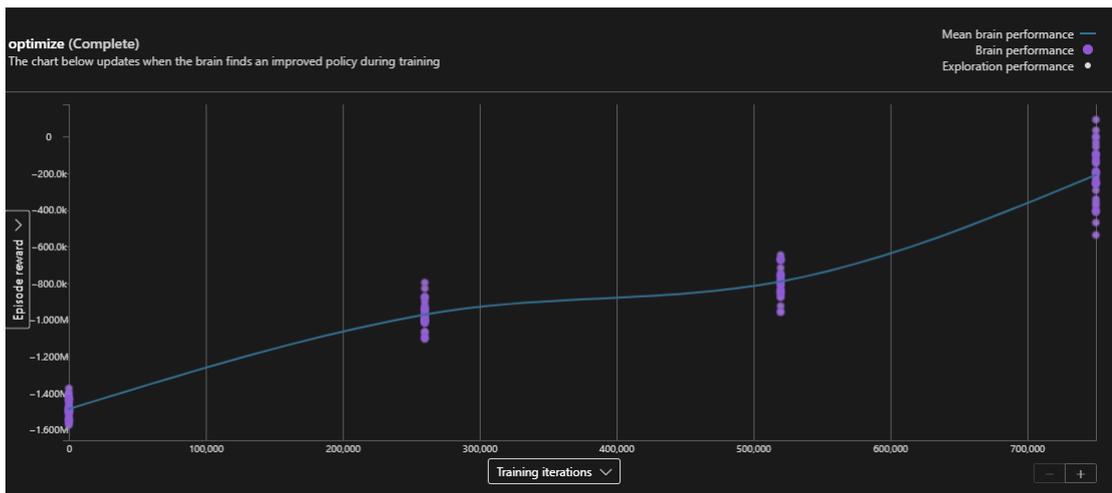


Figura 7.1: Función Reward a lo largo de los episodios

Si nos enfocamos en las últimas iteraciones del entrenamiento, podemos analizar el desarrollo de la recompensa en estas interacciones finales. En la figura 7.2, se observa una tendencia ascendente en la recompensa, lo que indica un progreso continuo en el desempeño del modelo.

Es importante destacar que este progreso en las últimas iteraciones es un indicador positivo del aprendizaje y la capacidad de adaptación del modelo de RL. A medida que el modelo continúa entrenándose, es probable que siga mejorando y

optimizando aún más el procesamiento de los pedidos en el almacén.

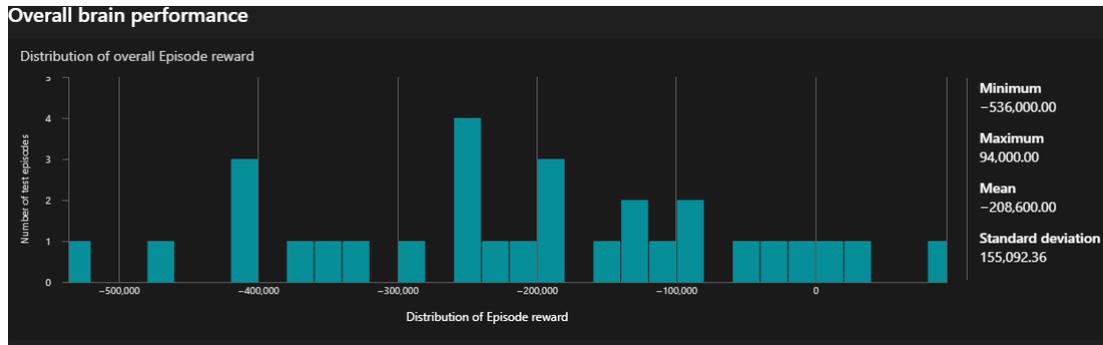


Figura 7.2: Función Reward en las últimas iteraciones

Por último, se presentan algunos resultados de la política que muestran cómo se van ajustando las probabilidades para obtener las mejores acciones en función de los diferentes escenarios.

La política aprendida durante el entrenamiento del modelo de RL ha evolucionado y se ha adaptado a lo largo de las iteraciones. A medida que el modelo adquiere experiencia y aprende de las interacciones con el entorno, ajusta las probabilidades de las diferentes acciones para maximizar la recompensa esperada en cada situación.

Estos resultados muestran cómo el modelo ha logrado capturar los patrones y las características del entorno, permitiéndole tomar decisiones óptimas en diferentes escenarios. Al ajustar las probabilidades de las acciones, el modelo selecciona las mejores opciones disponibles para cada situación, lo que conduce a una mejora en el tiempo de procesamiento de los pedidos y, en última instancia, a una mayor eficiencia en el funcionamiento del almacén.



(a) Probabilidad de elegir el servicio 1



(b) Probabilidad de elegir el servicio 2



(c) Probabilidad de elegir el servicio 3



(d) Probabilidad de elegir el servicio 4

Una vez que el modelo de RL fue entrenado con éxito, se procedió a exportarlo y realizar una serie de pruebas para evaluar su desempeño. En la tabla 7.1 se presentan los resultados detallados de estas pruebas.

Estas pruebas fueron realizadas con el fin de verificar la efectividad y el correcto entrenamiento del modelo de RL, así como para identificar posibles áreas de mejora y optimización. Los resultados obtenidos en estas pruebas son fundamentales para evaluar la idoneidad del modelo entrenado y tomar decisiones informadas sobre su implementación en entornos de producción reales.

En la Figura 7.4, se presentan las imágenes correspondientes a los test realizados.

Escenario	Q1	Q2	Q3	Q4	Prob 1	Prob 2	Prob 3	Prob 4
1	100	100	100	100	0.743	0.344	0.416	0.433
2	1	100	100	100	0.916	0.489	0.338	0.316
3	1000	10	10	10	0.167	0.561	0.616	0.655

Cuadro 7.1: Resultados del test del modelo básico

De los resultados presentados en la Tabla 7.1, se puede inferir que el cerebro responde de manera adecuada a las pruebas realizadas. A continuación se describen los hallazgos obtenidos en cada uno de los escenarios evaluados:

- En el **Escenario 1**, se simuló un entorno donde las cuatro colas tenían el mismo número de agentes en espera. Como era de esperar, la probabilidad más alta correspondió al servicio 1, el cual tenía el menor tiempo de mantenimiento.
- En el **Escenario 2**, el objetivo fue evaluar el funcionamiento general del modelo. En este caso, se forzó al cerebro a dirigirse únicamente al servicio 1, y los resultados reflejaron de manera acertada que las probabilidades indicaban esta preferencia.
- En el **Escenario 3**, se deseaba observar el comportamiento del modelo cuando la cola del primer servicio era considerablemente más larga que las demás. Como resultado, las probabilidades ajustadas evitaron en gran medida dirigirse al servicio 1, demostrando una respuesta coherente ante la situación planteada.

```
"qS1": 100,
"qS2": 100,
"qS3": 100,
"qS4": 100,
```

(a) Escenario 1

```
{
  "prob1": 0.7425366044044495,
  "prob2": 0.34434401988983154,
  "prob3": 0.4166921377182007,
  "prob4": 0.4330193102359772
}
```

(b) Probabilidad escenario 1

```
"qS1": 1,
"qS2": 100,
"qS3": 100,
"qS4": 100,
```

(c) Escenario 2

```
{
  "prob1": 0.9164726734161377,
  "prob2": 0.4896218180656433,
  "prob3": 0.338235080242157,
  "prob4": 0.3157693147659302
}
```

(d) Probabilidades escenario 2

```
"qS1": 1000,
"qS2": 10,
"qS3": 10,
"qS4": 10,
```

(e) Escenario 3

```
{
  "prob1": 0.16772273182868958,
  "prob2": 0.5601528286933899,
  "prob3": 0.6157240867614746,
  "prob4": 0.6552175879478455
}
```

(f) Probabilidades escenario 3

Figura 7.4: Imágenes de los test realizados

## 7.2. Modelo Warehousing

### 7.2.1. Configuración de entrenamiento

La configuración del entorno del modelo de Reinforcement Learning (RL) en el simulador AnyLogic se describe a continuación:

- **Horario de llegada y salida de camiones:** Tanto los camiones en el área de recepción (inbound) como en el área de carga (outbound) siguen un horario establecido. Este horario fue creado para simular la llegada y salida programada de los camiones.
- **Horario de llegada de pedidos:** La cantidad de pedidos que llegan se determina mediante un horario. Se simularon olas de pedidos masivos en horas clave, como la franja de 20:00 a 22:00 o de 10:00 a 12:00, para representar momentos de alta demanda.

- **Trabajadores en la zona de descarga de camiones y almacenamiento:** Se estableció un total de 10 trabajadores en la zona de descarga de camiones y en la zona de almacenamiento. Estos trabajadores son responsables de las tareas relacionadas con la recepción y almacenamiento de los productos.
- **Cantidad de trabajadores en la zona de empaquetado y carga:** La cantidad de trabajadores en la zona de empaquetado y en la zona de carga es un parámetro que utiliza el modelo de RL como acción. El rango establecido para esta cantidad va desde 2 hasta 6 en cada una de las zonas mencionadas. Esta configuración permite que el modelo de RL determine la cantidad óptima de trabajadores necesarios en función de las condiciones y la demanda de la simulación.

A continuación se presenta la configuración final del modelo de Reinforcement Learning:

- **Observation:** La observación en este modelo incluye variables como el tiempo transcurrido desde la recepción hasta el despacho de un pedido, la cantidad de pedidos pendientes de procesar, empaquetar o cargar, así como la cantidad de pedidos restantes para cada destino.
- **Action:** Las acciones definidas para el agente en cada estado incluyen la determinación de la capacidad máxima de los camiones en un rango de 90 a 110 paquetes, y la asignación de la cantidad de trabajadores en las áreas de empaquetado o carga. Además, se dispone de una acción para decidir qué pedidos deben procesarse en función de su destino. Por ejemplo, si el siguiente camión se dirige al *Norte*, se dará prioridad al procesamiento de los pedidos con ese destino. Esta acción se representa como un array de 4 posiciones binarias, donde se asigna el valor 1 si se procesan los pedidos para ese destino y 0 en caso contrario.
- **Configuration:** No se establecieron parámetros iniciales en cada iteración del modelo. El entorno se mantuvo constante una vez que fue cargado.
- **Reward:** La función de recompensa utilizada sigue la idea de asignar una recompensa inversamente proporcional a la duración del procesamiento de un pedido. A mayor duración, menor será la recompensa obtenida.

$$\text{Reward} = \frac{1}{\text{Tiempo procesamiento} + \varepsilon} \quad \text{donde } \varepsilon = 1e^{-4}$$

- **Terminal:** La condición terminal para este problema se establece cuando la duración del procesamiento de un pedido excede los 2880 minutos (2 días). Esto se debe a que para cada destino existe más de un camión diario, por lo que no tiene sentido que un pedido tarde más de dos días en ser procesado.

En el proceso de selección de algoritmo realizado por Bonsai para este entorno de simulación de warehouse, se determinó que Proximal Policy Optimization (PPO) es el algoritmo más adecuado. Durante las primeras interacciones del modelo, Bonsai evaluó diferentes opciones y concluyó que PPO es la elección óptima para este caso específico.

Algunas de las razones por las que Bonsai elige el algoritmo Proximal Policy Optimization (PPO) para el modelo *Warehousing* pueden ser las siguientes:

1. Efectividad en optimización continua: PPO es conocido por su capacidad para realizar optimizaciones continuas en políticas de toma de decisiones. Dado que el objetivo principal del modelo de RL es optimizar el procesamiento de pedidos para reducir el tiempo de espera desde la recepción hasta la carga, PPO podría ser una opción adecuada.
2. Aprovechamiento de políticas anteriores: PPO permite el aprovechamiento de políticas anteriores durante el entrenamiento, lo que significa que el modelo puede aprender de experiencias pasadas y mejorar progresivamente su desempeño. En un entorno de warehouse donde hay repeticiones de patrones y condiciones similares, este enfoque puede ser beneficioso para acelerar la convergencia y obtener políticas más robustas.
3. Estabilidad y convergencia rápida: PPO está diseñado para ser estable y converger rápidamente a políticas óptimas. Al tener un entorno de simulación complejo y dinámico, es importante contar con un algoritmo de RL que sea capaz de manejar estos desafíos y encontrar soluciones óptimas en un tiempo razonable.

### 7.2.2. Resultados

El proceso de entrenamiento del modelo concluyó después de 17 horas y 23 minutos, durante los cuales se llevaron a cabo un total de 753.093 iteraciones y 193 episodios. Se generaron una serie de gráficas que representan el rendimiento del modelo durante el entrenamiento. Sin embargo, el rendimiento obtenido no fue el esperado, ya que no logró aprender de manera efectiva.

En particular, al observar la gráfica 7.5, se puede notar que la recompensa no experimenta una variación significativa ni muestra una disminución considerable a lo largo de las iteraciones. Esto indica que el modelo no fue capaz de mejorar su desempeño en términos de recompensa a medida que avanzaba el entrenamiento.



Figura 7.5: Desarrollo del Reward a lo largo de las iteraciones

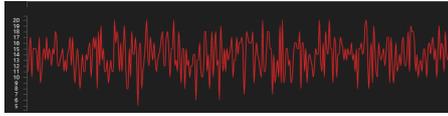
Es importante destacar que el problema del bajo rendimiento del modelo no se debe al cerebro en sí, sino más bien a la configuración de Bonsai y cómo se han establecido tanto las acciones como, sobre todo, las observaciones. Aunque se sabe que el modelo ha funcionado correctamente en términos de computación, como se puede apreciar en las gráficas de acciones (ver Figura 7.6), donde se muestra un funcionamiento normal a lo largo de las interacciones y estados.

Esto implica que el problema radica en cómo se ha diseñado la forma en que el modelo interactúa con el entorno y cómo se han representado y capturado las observaciones relevantes para el aprendizaje. La configuración actual no sea adecuada para el problema en cuestión, lo que ha llevado a que el modelo no aprenda correctamente.

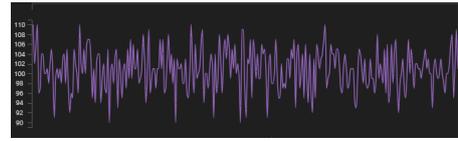
Para confirmar que el modelo no ha aprendido correctamente, se llevaron a cabo pruebas en tres escenarios distintos utilizando la API. Los resultados de estas pruebas se resumen en la tabla 7.2.

Cuadro 7.2: Ejemplo de uso de los elementos de una simulación

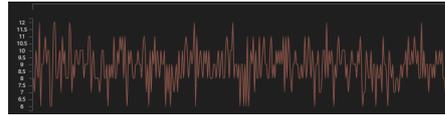
Back Or- ders	NOrders	EOrders	WOrders	SOrders	Packer Capacity	Load Ca- pacity	max Quantity	Destination Array
10	0	0	0	0	9	14	101	[0, 1, 1, 1]
1000	0	0	100	0	9	13	101	[1, 1, 1, 1]
100	100	100	0	0	9	14	101	[0, 0, 1, 1]



(a) Capacidad de la zona de carga



(b) Capacidad máxima de los camiones



(c) N<sup>o</sup> de trabajadores en la zona de empaquetado

Figura 7.6: Acciones

En la Tabla 7.2, se presentan tres escenarios distintos, cada uno con sus resultados correspondientes, en la Figura 7.7 se pueden ver las pruebas de estos resultados:

- **Escenario 1:** En este escenario, se simuló una situación con pocos pedidos en cola y ninguno en carga, lo que indica un período de baja actividad. Los resultados obtenidos muestran un rendimiento satisfactorio, aunque podrían ser mejorados, ya que los pedidos del *Norte* no fueron activados y no se proporciona ninguna explicación para esta omisión.
- **Escenario 2:** En este caso, se simuló una alta cantidad de pedidos para procesar, con uno de los destinos saturado. La respuesta del sistema es apropiada, ya que ante una alta demanda se activan todos los destinos y se alcanza la máxima capacidad de carga de los trabajadores.
- **Escenario 3:** En este escenario, se simuló una carga de pedidos moderada, pero con dos destinos saturados. Nuevamente, la respuesta del sistema es correcta, ya que solo se activan los destinos que no están saturados.

```
{
  "backorders": 10,
  "NOrders": 0,
  "EOrders": 0,
  "WOrders": 0,
  "SOrders": 0
}
```

(a) Escenario 1

```
{
  "maxQuantity": 101,
  "packerCapacity": 9,
  "loadCapacity": 14,
  "destinationArray": [
    0,
    1,
    1,
    1
  ]
}
```

(b) Resultados escenario 1

```
{
  "backorders": 100,
  "NOrders": 100,
  "EOrders": 100,
  "WOrders": 0,
  "SOrders": 0
}
```

(c) Escenario 2

```
{
  "maxQuantity": 101,
  "packerCapacity": 9,
  "loadCapacity": 12,
  "destinationArray": [
    0,
    0,
    1,
    1
  ]
}
```

(d) Resultados escenario 2

```
{
  "backorders": 1000,
  "NOrders": 0,
  "EOrders": 0,
  "WOrders": 100,
  "SOrders": 0
}
```

(e) Escenario 3

```
{
  "maxQuantity": 101,
  "packerCapacity": 9,
  "loadCapacity": 13,
  "destinationArray": [
    1,
    1,
    1,
    1
  ]
}
```

(f) Resultados escenario 3

Figura 7.7: Imágenes de los test realizados

El modelo no ha logrado aprender correctamente debido a las limitaciones de Bonsai. Como se mencionó anteriormente, una de las principales razones es la dificultad para definir las observaciones de manera adecuada. Una parte crucial del aprendizaje en este contexto es que el modelo sea capaz de capturar los patrones en los calendarios de llegada de camiones y las entradas de pedidos, que son indicadores clave de las fluctuaciones en la demanda.

Sin embargo, Bonsai tiene limitaciones en cuanto a la inclusión de variables no numéricas, lo que impide incorporar directamente los calendarios como observaciones. Esto resulta en observaciones de baja calidad, lo que afecta negativamente el aprendizaje del modelo. Además, esta limitación también repercute en las acciones que el modelo puede tomar.

La incapacidad para considerar los calendarios de llegada de camiones y las olas de pedidos como variables de observación representa una limitación significativa en la capacidad del modelo para comprender y adaptarse a los patrones de la demanda. Como resultado, el rendimiento del modelo se ve comprometido y se vuelve bastante limitado en su capacidad para tomar decisiones informadas y óptimas.

Es importante tener en cuenta estas limitaciones al evaluar el desempeño del modelo y considerar alternativas o enfoques adicionales que permitan abordar de manera más efectiva el aprendizaje y la adaptación a los patrones y características clave del problema.

## 7.3. Conclusiones

Al inicio de este proyecto, se establecieron dos objetivos principales: comprender el funcionamiento de AnyLogic y Bonsai, y lograr su integración en entornos para obtener beneficios. Una vez que se alcanzaron estos objetivos y se adquirió una comprensión adecuada de cómo funcionan y cómo se pueden combinar de manera efectiva, se planteó un objetivo más ambicioso. Dado que este trabajo estaba relacionado con las prácticas en Accenture, se propuso explorar el potencial de utilizar Bonsai en proyectos de almacenamiento y logística.

Para abordar este objetivo, se planteó el desafío de optimizar las olas de pedidos en un entorno de almacenamiento, con el objetivo de determinar si Bonsai podía proporcionar beneficios significativos y ofrecer una solución real para problemas complejos y de gran escala en este campo.

El objetivo era evaluar si Bonsai podía contribuir a mejorar la eficiencia y la productividad en los procesos de almacenamiento y logística al utilizar técnicas avanzadas de aprendizaje por refuerzo. Se buscaba determinar si la capacidad de Bonsai para tomar decisiones óptimas basadas en el aprendizaje y la adaptabilidad podría ser aprovechada para optimizar las operaciones de almacenamiento, como la asignación de pedidos, durante olas de pedidos.

Al perseguir este objetivo, se buscaba no solo demostrar la viabilidad y efectividad de Bonsai en proyectos de almacenamiento, sino también abrir la puerta a la posibilidad de ofrecer esta solución a los clientes de Accenture como una opción valiosa para abordar problemas complejos y amplios en el ámbito del warehousing.

El modelo *Simple* implementado y los resultados obtenidos demuestran que Bonsai es capaz de aprender y proporcionar soluciones óptimas para entornos específicos mediante el uso de Reinforcement Learning. En este caso, se logró cumplir el primer objetivo del proyecto, que consistía en comprender y utilizar tanto AnyLogic como Bonsai de manera efectiva, integrando ambas herramientas para obtener resultados significativos.

La conclusión principal es que los cerebros de Bonsai son una solución poderosa y eficiente para optimizar problemas en entornos simples, donde las variables de observación pueden ser fácilmente modeladas como valores numéricos fijos.

Cuando nos referimos a variables fácilmente modeladas, nos referimos a aquellas que se pueden representar de manera precisa y clara mediante un valor numérico.

En el contexto del modelo *Simple* presentado, las variables de observación, como el tiempo transcurrido desde que se recibió un pedido hasta que se despachó o la cantidad de pedidos pendientes de procesar, pueden ser fácilmente representadas como valores numéricos fijos. Estas variables proporcionan información clara sobre el estado del entorno en cada momento.

Además, los conjuntos de acciones utilizados en el modelo también son simples pero efectivos para mejorar la recompensa. Por ejemplo, las distintas probabilidades de seleccionar uno u otro servicio. Estas acciones tienen un impacto directo en el rendimiento y la eficiencia del procesamiento de pedidos.

La conclusión que se puede obtener del modelo de Warehousing es que Bonsai aún no está completamente preparado para abordar este tipo de entornos. En este caso específico, los cerebros de Bonsai no son capaces de aprender una solución óptima para los problemas relacionados con las olas de pedidos en el almacenamiento.

La raíz de este problema reside en las dificultades de implementación de las observaciones y acciones en este contexto. En el caso del modelo *Simple* presentado anteriormente, las observaciones y estados podían ser fácilmente representados mediante valores numéricos, lo cual facilitaba el aprendizaje del cerebro; sin embargo, en el caso del Warehousing y los problemas relacionados con las olas de pedidos, existen desafíos adicionales en la representación de las observaciones y acciones.

En cuanto a las observaciones, la dificultad radica en la representación de los calendarios de llegada de camiones y pedidos, los cuales son elementos clave para el aprendizaje del cerebro. Estos calendarios no pueden ser representados fácilmente mediante valores numéricos, lo que dificulta la obtención de información relevante por parte del cerebro.

En relación a las acciones, también se presentan dificultades. Por ejemplo, la selección del destino se implementa utilizando un array, el cual es una estructura que Bonsai tiene dificultades para trabajar. Esto limita las opciones disponibles para modelar y tomar decisiones óptimas en el proceso de optimización de las olas de pedidos.

Desde el punto de vista del autor, se concluye que Bonsai no aporta grandes beneficios en entornos y problemas complejos como el del modelo Warehousing para la optimización de olas de pedidos. Por lo tanto, en la actualidad, no sería una solución factible para ser ofrecida a clientes.

La evaluación realizada en este trabajo reveló que los desafíos asociados con la representación de observaciones y acciones en entornos complejos limitan la capacidad de aprendizaje y optimización de los cerebros de Bonsai. Aunque se han identificado dificultades específicas, es importante destacar que los problemas planteados en este trabajo no eran de naturaleza avanzada y, sin embargo, Bonsai mostró dificultades para brindar soluciones óptimas.

Considerando que los clientes suelen plantear problemáticas mucho más complejas que requieren un enfoque más sofisticado, se puede concluir que Bonsai no sería una solución viable para ser comercializada en su forma actual. Las limitaciones identificadas en la representación de observaciones y acciones indican que se requerirían avances significativos en el desarrollo de Bonsai para abordar problemas más desafiantes y ofrecer soluciones efectivas a los clientes.

Además, es importante destacar que la disponibilidad de documentación y ejemplos para aprender a utilizar Bonsai en la implementación de modelos ha sido muy limitada. Esta falta de recursos dificulta el proceso de aprendizaje y comprensión de cómo aplicar eficazmente Bonsai para resolver problemas específicos.

En este contexto, el presente trabajo adquiere un valor significativo al proporcionar una guía integral y detallada para la implementación y análisis de modelos de simulación que utilizan Bonsai para optimizar problemas en entornos de Warehousing.

# Bibliografía

- [dLR07] René de Koster, Tho Le-Duc y Kees Jan Roodbergen. “Design and control of warehouse order picking: A literature review”. En: *European Journal of Operational Research* 182.2 (2007), págs. 481-501. ISSN: 0377-2217. DOI: <https://doi.org/10.1016/j.ejor.2006.07.009>. URL: <https://www.sciencedirect.com/science/article/pii/S0377221706006473>.
- [Hor+18] D. Horgan et al. *DISTRIBUTED PRIORITIZED EXPERIENCE REPLAY*. <https://arxiv.org/abs/1803.00933>. Accessed on 2023-04-29. 2018.
- [LK91] Averill M. Law y W. David Kelton. *Simulation Modeling and Analysis*. McGRAW·HILL INTERNATIONAL EDITIONS, 1991. ISBN: 0-07-100803-9.
- [Mni+15] V. Mnih et al. *Human-level control through deep reinforcement learning*. *Nature*, 529-533. <https://www.nature.com/articles/nature14236>. Accessed on 2023-04-29. Feb. de 2015.
- [PA04] Charles G. Petersen y Gerald Aase. “A comparison of picking, storage, and routing policies in manual order picking”. En: *International Journal of Production Economics* 92.1 (2004), págs. 11-19. ISSN: 0925-5273. DOI: <https://doi.org/10.1016/j.ijpe.2003.09.006>. URL: <https://www.sciencedirect.com/science/article/pii/S0925527303002937>.
- [PSR95] Pedgen, R.E. Shannon y R.P.Sadowski. *Introduction to Simulation Using SIMAN 2d ed.* McGRAW-Hill, 1995.
- [Ric11] Gwynne Richards. *Warehouse Management: A complete guide to improving efficiency and minimizing costs in the modern warehouse*. Kogan Page, 2011. ISBN: 978-0-7494-6934-4.
- [SB18] Richard S. Sutton y Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018. ISBN: 9780262039246.

- [Sch+17] John Schulman et al. *Proximal Policy Optimization Algorithms*. <https://arxiv.org/abs/1707.06347>. Accessed on 2023-04-29. 2017.
- [Tea22] Project Bonsai-Microsoft Team. *Inklings programming language reference*. <https://learn.microsoft.com/en-us/bonsai/inkling/>. 2022.