ugr | Universidad de Granada

TRABAJO FIN DE MÁSTER

MÁSTER UNIVERSITARIO OFICIAL EN CIENCIA DE DATOS E INGENIERÍA DE COMPUTADORES

Ampliación del espacio de estados en aprendizaje por refuerzo: aplicación al control de sistemas de climatización.

Autor
Francisco Pertíñez Perea

Directores
Juan Gómez Romero
Miguel Molina Solana

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Granada, 6 de septiembre de 2024

Ampliación del espacio de estados en aprendizaje por refuerzo: aplicación al control de sistemas de climatización.

Autor
Francisco Pertíñez Perea

Directores
Juan Gómez Romero
Miguel Molina Solana

Ampliación del espacio de estados en aprendizaje por refuerzo: aplicación al control de sistemas de climatización

Francisco Pertíñez Perea

Palabras clave: Aprendizaje por refuerzo profundo; Espacio de estados; Control energético de edificios

Resumen

El aprendizaje por refuerzo (Reinforcement Learning, RL) es un área del aprendizaje automático en la que un agente interactúa con su entorno para resolver tareas mediante la experimentación continua y la obtención de recompensas. La evolución hacia el aprendizaje por refuerzo profundo (Deep Reinforcement Learning, DRL) introduce un enfoque innovador, utilizando redes neuronales para identificar y seleccionar las acciones más prometedoras.

Un campo emergente para el DRL es el control de sistemas de climatización en edificios, donde ha demostrado ser más eficiente que los métodos reactivos tradicionales. En cualquier problema de RL, un componente esencial es el espacio de estados (S), que representa las percepciones del entorno a las que el agente tiene acceso. En escenarios reales, como la gestión energética de edificios, el agente no dispone de toda la información del entorno, sino solo de las observaciones captadas por ciertos sensores.

Este trabajo se centrará en ampliar el espacio de estados de un agente de DRL incorporando información contextual adicional. Específicamente, se explorará la integración tanto de datos históricos como de predicciones meteorológicas al estado actual del agente. Se evaluará cómo diferentes configuraciones respecto al horizonte temporal y la distancia temporal entre la información añadida influyen en el rendimiento del agente en diversos casos de uso.

Para esta investigación se empleará y mejorará el software Sinergym, una herramienta que facilita la interacción entre agentes de DRL y modelos de simulación de edificios mediante EnergyPlus. Esta plataforma permitirá probar y validar nuevas configuraciones, avanzando hacia un control energético de edificios más inteligente y eficiente.

State space extension in reinforcement learning: application to air conditioning control

Francisco Pertíñez Perea

Keywords: Deep Reinforcement Learning; State space; Building energy control

Abstract

Reinforcement Learning (RL) is an area of machine learning in which an agent interacts with its environment to solve tasks through continuous experimentation and reward. The evolution towards Deep Reinforcement Learning (DRL) introduces an innovative approach, using neural networks to identify and select the most promising actions.

An emerging field for DRL is the control of air conditioning systems in buildings, where it has proven to be more efficient than traditional reactive methods. In any RL problem, an essential component is the state space (S), which represents the perceptions of the environment to which the agent has access. In real scenarios, such as building energy management, the agent does not have access to all the information of the environment, but only to the observations captured by certain sensors.

This work will focus on extending the state space of a DRL agent by incorporating additional contextual information. Specifically, the integration of both historical data and weather forecasts into the current state of the agent will be explored. It will evaluate how different configurations with respect to the time horizon and the temporal distance between the added information influence the performance of the agent in various use cases.

For this research, Sinergym software, a tool that facilitates the interaction between DRL agents and building simulation models through EnergyPlus, will be used and improved. This platform will allow testing and validation of new configurations, moving towards smarter and more efficient building energy control.

Yo, **Francisco Pertíñez Perea**, alumno del Máster Universitario Oficial en Ciencia de Datos e Ingeniería de Computadores de la **Universidad de Granada**, con DNI **77765690E**, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Máster en la biblioteca, para que pueda ser consultada por las personas que lo deseen.

Francisco Pertíñez Perea

Granada, a 6 de septiembre de 2024.

D. **Juan Gómez Romero**, catedrático de universidad del Departamento Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

D. **Miguel Molina Solana**, profesor titular del Departamento Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Ampliación del espacio de estados en aprendizaje por refuerzo: aplicación al control de sistemas de climatización***, ha sido realizado bajo su supervisión por **Francisco Pertíñez Perea**, y autorizan la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 6 de septiembre de 2024.

Los directores:

Juan Gómez Romero **Miguel Molina Solana**

Agradecimientos

Quiero expresar mi agradecimiento a todas las personas que han estado a mi lado y me han brindado su apoyo durante la realización de este trabajo.

En primer lugar, a mi familia, cuyo amor y apoyo constante han sido la piedra angular en este viaje. Su comprensión y aliento han sido fundamentales para que pudiera avanzar y superar cada desafío que se presentó en mi camino.

Agradezco también a mis tutores, Juan y Miguel, cuya guía experta, paciencia y sabios consejos fueron vitales para la elaboración de este trabajo. Las reuniones regulares que mantuvimos no solo moldearon mis ideas, sino que también me ayudaron a descubrir el apasionante mundo del aprendizaje por refuerzo, una disciplina que desconocía y que ahora me apasiona profundamente. Gracias a la confianza que depositaron en mí, he podido explorar y profundizar en este campo, que espero seguir investigando en el futuro.

Asimismo, quiero agradecer a mis compañeros de proyecto, Alejandro y Antonio, por su valiosa colaboración. Sus aportaciones, tanto en persona como a través de nuestras discusiones en Slack, fueron cruciales para el desarrollo y la finalización de este proyecto.

Al comenzar este trabajo, el objetivo final parecía lejano y lleno de retos, pero gracias al apoyo y la colaboración de todas estas personas, he logrado alcanzar esta meta.

A todos ustedes, ¡muchas gracias!

Francisco Pertíñez Perea
Granada, a 6 de septiembre de 2024

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

Capítulo 1

Introducción

En este capítulo, se expondrá la motivación detrás de la realización de este proyecto, se definirán los objetivos específicos del mismo y finalmente se dará una visión general de la estructura del documento.

1.1. Motivación

El creciente consumo energético en edificios residenciales y comerciales está intensificando de manera alarmante el calentamiento global y el cambio climático. Según la Agencia Internacional de Energía (*International Energy Agency*, IEA), este sector es vital en la ecuación del consumo global de energía y las emisiones de gases de efecto invernadero, siendo responsable de más de un tercio de ellas[1]. La rápida expansión de la construcción, especialmente en los países en desarrollo, junto con el incremento en la adquisición de electrodomésticos, subraya la urgente necesidad de mejorar la eficiencia energética para avanzar hacia fuentes de energía más sostenibles.

Los edificios, con su larga vida útil, tienen un impacto duradero en el consumo energético. Por tanto, las decisiones de diseño y construcción de los mismos no solo afectan el consumo inmediato, sino que influenciarán la eficiencia energética durante décadas. Para cumplir con los objetivos de emisiones netas cero para 2050 propuestos por las Naciones Unidas (*United Nations*, UN)[2], es crucial implementar políticas robustas, como estándares mínimos de rendimiento y códigos de energía. Estas iniciativas no solo alinearán al sector con los objetivos de reducción de emisiones, sino que también aprovecharán tecnologías existentes que prometen ahorros significativos de energía y beneficios económicos.

[1] https://www.iea.org/energy-system/buildings
[2] https://www.un.org/es/climatechange/net-zero-coalition

Dentro del complejo universo de la demanda energética en edificios, el manejo eficiente de los sistemas de climatización (*heating*, *ventilating*, *air conditioning*: HVAC) se postula como un pilar fundamental para mejorar la eficiencia energética y mitigar el impacto ambiental [10]. Tradicionalmente, los métodos de control han sido reactivos, respondiendo a las condiciones ya establecidas. Sin embargo, para alcanzar una mayor optimización en el uso de la energía, y así adaptarse de manera eficaz a las cambiantes condiciones ambientales, es esencial avanzar hacia estrategias de control más dinámicas y proactivas.

Para mostrar esta necesidad, la IEA hace hincapié en la importancia de adoptar enfoques avanzados para mejorar la eficiencia energética en la edificación, enfatizando la urgencia en integrar tecnologías y estrategias que permitan una gestión más inteligente y eficiente de la energía, destacando el papel crucial de sistemas avanzados de control HVAC[3].

Ante este panorama, el Aprendizaje Profundo por Refuerzo (*Deep Reinforcement Learning*, DRL) se presenta como una tecnología prometedora en cuanto a la gestión de los sistemas HVAC [36]. A diferencia de los métodos tradicionales, el DRL permite a un agente aprender a tomar decisiones óptimas mediante la interacción continua con su entorno, aprovechando recompensas y acciones repetitivas. Esta metodología ofrece una adaptación dinámica a las fluctuaciones del entorno, alejándose de los enfoques meramente reactivos y abriendo la puerta a una gestión más ágil y eficiente.

En el ámbito del Aprendizaje por Refuerzo (*Reinforcement Learning*, RL), el espacio de estados (S) desempeña un papel crucial, ya que representa la gama de percepciones disponibles para el agente. En el contexto del control energético de edificios, esta característica es crucial, puesto que el agente controlador no siempre tiene acceso a toda la información relevante, limitándose a observaciones específicas de los sensores instalados.

Este proyecto se centrará en la ampliación del espacio de estados de un agente de DRL mediante la incorporación de información contextual adicional. Se investigará como tanto la integración de información de estados pasados como la inclusión de predicciones meteorológicas al estado actual del agente puede afectar (e idealmente, mejorar) al rendimiento de agente DRL. En particular, se evaluará cómo diferentes configuraciones respecto al horizonte temporal y la distancia temporal entre la información añadida, influyen en el rendimiento del agente en diversos casos de uso.

Cabe destacar que el desarrollo de este proyecto están enmarcado en el contexto del proyecto "IA4TES: *Inteligencia Artificial para la Transición Energética Sostenible*", financiado por el Ministerio de Asuntos Económicos

[3] https://www.iea.org/energy-system/buildings#tracking

y Transformación Digital[4].

1.2. Objetivos

El **objetivo principal** que se persigue en este trabajo es el siguiente:

Objetivo principal

Investigar cómo ampliar el espacio de estados de un agente de DRL mediante el uso de información contextual disponible puede mejorar su desempeño en el contexto del control HVAC.

El objetivo principal se puede descomponer en varios **subobjetivos**, cada uno vinculado a las actividades fundamentales que se realizarán a lo largo de este trabajo.

Subobjetivo 1

Investigación en fundamentos de DRL y procesos de Markov parcialmente observables, y su aplicación en control HVAC.

Subobjetivo 2

Revisión bibliográfica y del estado del arte. Detección de carencias en la literatura.

Subobjetivo 3

Extensión de la librería de Python Sinergym con funcionalidades para la incorporación de previsiones meteorológicas al estado de los agentes.

Subobjetivo 4

Definición de una metodología para la evaluación sistemática de diferentes configuraciones de entrenamiento de agentes DRL.

Subobjetivo 5

Experimentación de varios casos de uso con diferentes configuraciones respecto al algoritmo, entorno y espacio de estados utilizados.

[4] https://www.ia4tes.org/

1.3. Estructura del documento

Una vez definidos los objetivos del proyecto, en el Capítulo 2, **Organización y recursos**, se aborda la planificación del proyecto, el cronograma de trabajo, y los recursos utilizados para llevar a cabo este estudio.

En el **Capítulo 3: Marco Teórico**, se exponen los conceptos fundamentales y las teorías relevantes que sustentan el desarrollo del proyecto, proporcionando las bases necesarias para comprender las metodologías y enfoques adoptados. Además, ofrece una revisión de la literatura existente sobre el tema del proyecto, examinando estudios previos y las metodologías empleadas en investigaciones similares, lo que permite situar el trabajo en el contexto del estado del arte.

El **Capítulo 4: Desarrollo de Sinergym** se centra en la herramienta Sinergym, utilizada en este proyecto. Se explican los detalles de dicha librería y las contribuciones aportadas a la misma.

El **Capítulo 5: Experimentación** presenta los experimentos realizados con Sinergym. Se describe la metodología experimental, los escenarios analizados y se discuten los resultados obtenidos.

Finalmente, el **Capítulo 6: Conclusiones y trabajo futuro**, resume los principales hallazgos del proyecto, evaluando el nivel de consecución de los objetivos planteados. Además, se discuten las limitaciones encontradas y se sugieren posibles líneas de investigación futura.

1.4. Codigo implementado

Todo el código implementado a lo largo de este trabajo trabajo se puede encontrar en el siguiente repositorio de GitHub: https://github.com/fpertinezp/TFM-DRL-HVAC

Capítulo 2

Organización y recursos

En este capítulo, se describe la estructura organizativa adoptada para la realización del proyecto y se detallan los recursos disponibles que han sido fundamentales para su desarrollo. Se abordarán los roles y responsabilidades de los miembros del equipo, las herramientas y tecnologías empleadas y la distribución del tiempo y las fases del proyecto.

2.1. Metodología de trabajo

Para llevar a cabo este proyecto, se optó por aplicar la metodología **Agile Data Science** [15]. Esta decisión se basó en su capacidad para mejorar la eficiencia, promover la colaboración y mantener la flexibilidad, características esenciales para garantizar el éxito en proyectos de ciencia de datos.

Agile Data Science se basa en varios principios fundamentales. El primero es la **Iteración y Retroalimentación Continua**, que permite desarrollar el proyecto en ciclos cortos y dinámicos, donde los resultados son ajustados y mejorados constantemente en función de la retroalimentación obtenida en cada fase.

La **Colaboración entre Equipos** es otro pilar esencial, promoviendo una comunicación fluida y constante entre científicos de datos, ingenieros y partes interesadas, lo que garantiza que el proyecto siga alineado con las necesidades y objetivos del negocio en todo momento.

El principio de **Entrega Continua de Valor** se enfoca en proporcionar resultados incrementales y tangibles desde las primeras etapas, permitiendo que los stakeholders comiencen a beneficiarse desde el inicio y ajusten el rumbo del proyecto según sea necesario.

La **Adaptabilidad y Flexibilidad** inherentes a Agile Data Science son cruciales en un campo tan dinámico como la ciencia de datos, donde los

descubrimientos pueden reorientar significativamente el proyecto.

Finalmente, el **Enfoque en el Usuario Final** asegura que las soluciones desarrolladas sean claras, útiles y aplicables en contextos reales, maximizando así tanto el impacto como la utilidad de los resultados obtenidos.

2.2. Fases del proyecto

El desarrollo de este proyecto se estructuró en una serie de fases bien definidas, cada una de ellas diseñada para avanzar progresivamente desde la adquisición de conocimientos iniciales hasta la preparación de la defensa final. A continuación, se detallan las etapas que guiaron el proyecto:

- **Fase 1: Estudio inicial para la adquisición de los conocimientos necesarios para llevar a cabo el proyecto**. En esta fase se llevó a cabo la presentación del proyecto y se definieron las líneas generales de trabajo. Se revisó el material proporcionado para adquirir una comprensión profunda de los temas a tratar, y se dedicó tiempo al aprendizaje de la herramienta `Sinergym`, fundamental para el desarrollo posterior del proyecto.
- **Fase 2: Experimentación con wrapper *MultiObsWrapper***. Durante esta fase, se realizaron experimentos utilizando el wrapper *MultiObsWrapper*, ya disponible en `Sinergym`, en diferentes entornos. Se experimentó primero con el entorno de 5 zonas en su versión discreta, seguido de la versión continua del mismo entorno, y finalmente, se realizaron pruebas en el entorno Radiant, también en su versión continua.
- **Fase 3: Desarrollo de wrapper *WeatherForecastingWrapper***. Esta fase se centró en el desarrollo del wrapper *WeatherForecastingWrapper*. Una vez desarrollado, se comprobó su correcto funcionamiento para asegurar que cumplía con los requisitos del proyecto.
- **Fase 4: Experimentación con wrapper *WeatherForecastingWrapper***. Similar a la fase 2, en esta etapa se experimentó con el wrapper *WeatherForecastingWrapper* en distintos entornos. Se realizaron pruebas en el entorno de 5 zonas en su versión continua y en el entorno Radiant, también en versión continua.
- **Fase 5: Experimentación de uso combinado de wrappers**. En esta fase, se experimentó con el uso combinado de ambos wrappers, *MultiObsWrapper* y *WeatherForecastingWrapper*, en los mismos entornos utilizados en fases anteriores: 5 zonas (versión continua) y Radiant (versión continua).

- **Fase 6: Documentación de memoria**. Durante esta fase, se elaboró la documentación completa del proyecto, incluyendo la confección de la memoria final y la posterior revisión y corrección para asegurar la calidad y coherencia del documento.
- **Fase 7: Preparación de defensa del trabajo**. Finalmente, se destinó tiempo a la preparación de la defensa del trabajo, asegurando que todos los aspectos relevantes del proyecto estuvieran listos para ser expuestos.

2.3. Planificación temporal

La Figura 2.1 muestra la distribución de los días de trabajo y la asignación de las diferentes tareas realizadas. Estas tareas están alineadas con los objetivos y subobjetivos (sección 1.2) previamente establecidos. La duración total del proyecto ha sido de **159 días**.

En cuanto al **tiempo efectivo** dedicado a cada tarea, se ha utilizado un total de **330 horas** de trabajo, ajustándose así a las 300 horas requeridas para un Trabajo de Fin de Máster de 12 créditos. La Tabla 2.1 muestra una comparación entre el tiempo estimado para cada fase y el tiempo real empleado. Como podemos observar, existen diferencias entre las estimaciones y la realidad: algunas tareas tomaron más tiempo del previsto, mientras que otras se completaron en menos tiempo.

Durante el desarrollo del proyecto, es habitual que surjan imprevistos y errores en las distintas fases, lo que puede requerir tiempo adicional para su resolución y aumentar la duración total del proyecto en comparación con las estimaciones iniciales. En el caso de la evaluación de los diferentes agentes de RL, se llevaron a cabo pruebas exhaustivas para asegurar la calidad y confiabilidad de los resultados. Es importante resaltar que cada

Fases	**Tiempo previsto (horas)**	**Tiempo real (horas)**
Fase 1	70	80
Fase 2	50	40
Fase 3	40	20
Fase 4	50	60
Fase 5	10	15
Fase 6	80	95
Fase 7	15	15
TOTAL	310	330

Tabla 2.1: Tabla comparativa entre el tiempo estimado por tarea y tiempo real dedicado.

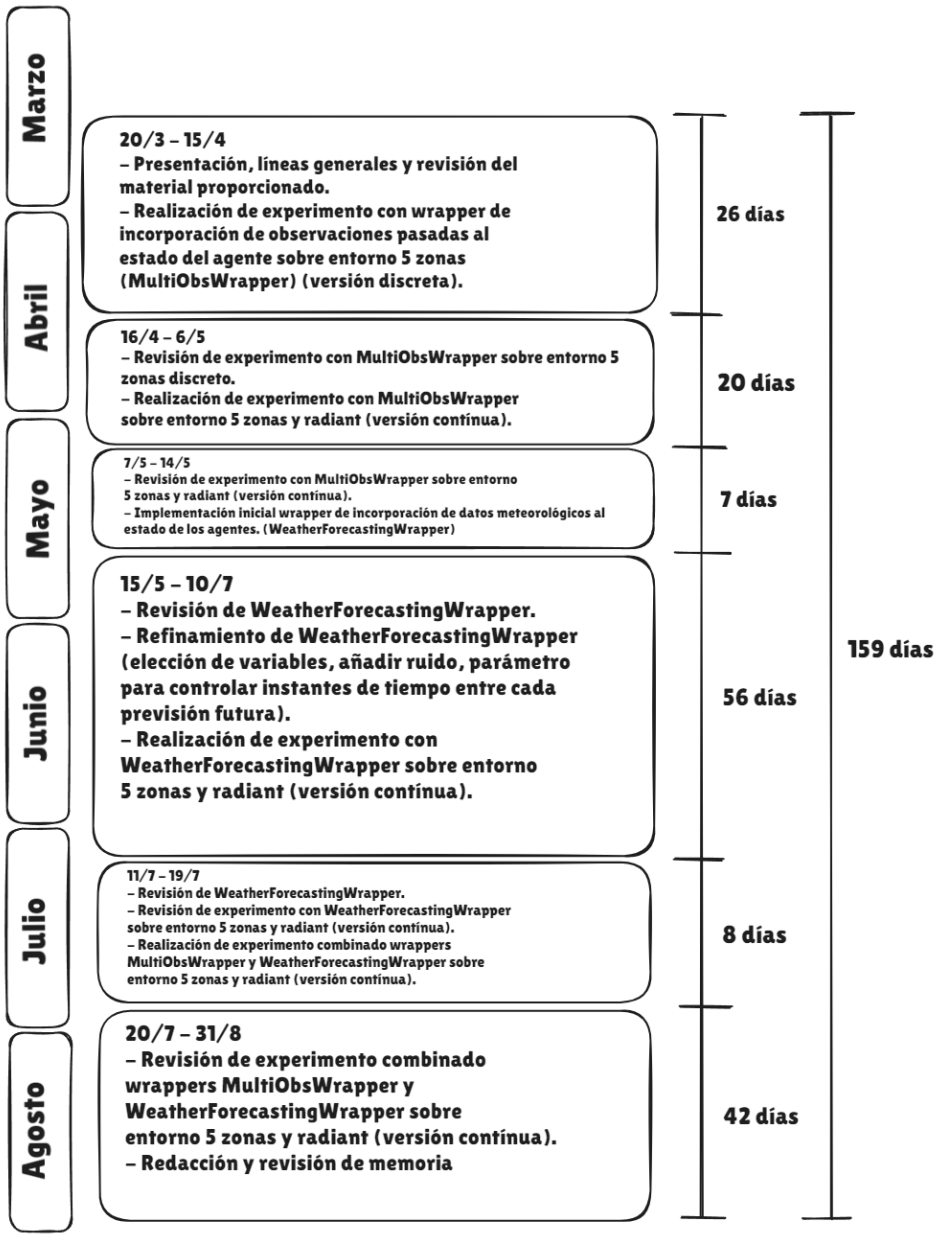


Figura 2.1: Distribución de los días de trabajo y la asignación de las diferentes tareas realizadas

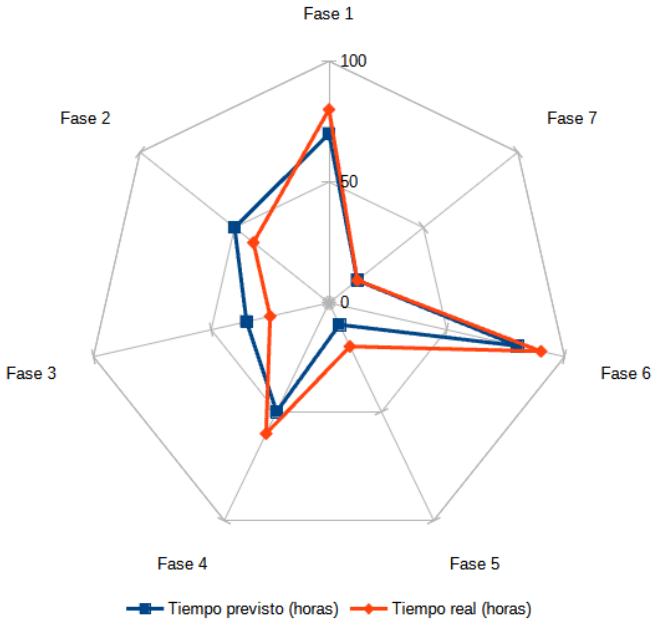


Figura 2.2: Gráfico radial que compara el tiempo estimado por tarea y el tiempo real dedicado.

agente ya entrenado fue expuesto a 10 episodios de evaluación replicando las mismas condiciones en cada ejecución, con el fin de obtener resultados más consistentes y representativos. Dado el considerable tiempo de ejecución que pueden consumir los algoritmos de DRL, esta aproximación requirió una planificación cuidadosa y anticipada.

Finalmente, al analizar la Figura 2.2, que muestra el gráfico radial de la planificación prevista en comparación con la real, se puede apreciar que la planificación fue adecuada, ya que no se observan diferencias significativas entre ambas.

2.4. Recursos utilizados

En esta sección se detallan los recursos clave para el proyecto, incluyendo los recursos humanos que comprenden el equipo y sus roles específicos, los recursos hardware como los equipos y tecnología utilizados, los recursos software que abarcan las herramientas y aplicaciones empleadas, y los recursos económicos correspondientes al coste del proyecto.

2.4.1. Recursos humanos

El equipo de recursos humanos del proyecto está conformado por Francisco Pertíñez Perea, quien tiene como tarea principal el desarrollo del proyecto, y los profesores Juan Gómez Romero y Miguel Molina Solana, que se encargan de la supervisión del mismo, y a la vez cumplen el papel de cliente.

2.4.2. Recursos hardware

Para el desarrollo del proyecto, se ha utilizado el ordenador personal del alumno. Las características del equipo son las siguientes:

- **Procesador**: Intel Core i7-12700H (frecuencia de hasta 4.7 GHz con Intel Turbo Boost Technology, 24 MB de caché L3, 14 núcleos, 20 hilos)
- **Memoria RAM**: 16 GB DDR5-4800 MHz (configuración de 2 x 8 GB)
- **Almacenamiento**: 512 GB SSD PCIe NVMe TLC M.2 y 1 TB HDD
- **Tarjeta gráfica**: Discreta, GPU NVIDIA GeForce RTX 3060 (6 GB de memoria GDDR6 dedicada)

Aunque es esencial contar con un equipo con buenas especificaciones de CPU y memoria para tareas de aprendizaje profundo, la GPU es el componente más destacado. En el aprendizaje profundo, especialmente durante el entrenamiento y evaluación de modelos de redes neuronales profundas, la GPU (Unidad de Procesamiento Gráfico) es fundamental debido a su capacidad para realizar cálculos intensivos de manera eficiente gracias a su arquitectura paralela.

2.4.3. Recursos software

Para el presente proyecto, se ha elegido el **lenguaje de programación Python 3**. Python es muy popular y ampliamente utilizado en el ámbito del aprendizaje automático. Las razones por las que Python se ha convertido en la opción preferida para desarrollar aplicaciones de aprendizaje automático incluyen:

- **Extensa disponibilidad de bibliotecas y frameworks**: Python cuenta con una amplia variedad de herramientas especializadas en aprendizaje automático, como TensorFlow, Keras, PyTorch, entre otras. Dentro de esta multitud de herramientas debemos destacar Sinergym,

una biblioteca para la simulación y optimización energética, que ha sido fundamental para el desarrollo de este proyecto.

- **Facilidad de uso y flexibilidad**: Python es conocido por su simplicidad y legibilidad, además de su versatilidad para una variedad de aplicaciones. Esto permite a los desarrolladores de aprendizaje automático utilizar Python para diversas fases del proceso, desde la limpieza y exploración de datos hasta la creación de modelos complejos.
- **Comunidad activa y recursos abundantes**: Python cuenta con una comunidad de desarrolladores muy dinámica y una amplia oferta de recursos en línea, incluyendo documentación detallada, tutoriales, ejemplos de código y foros de discusión. Esta comunidad activa apoya el continuo desarrollo y mejora de las bibliotecas y frameworks de aprendizaje automático en Python, facilitando el acceso a soluciones y la resolución de problemas comunes.

A continuación, se detallan las bibliotecas más relevantes empleadas durante el desarrollo de este trabajo:

- **Pandas**: Esta biblioteca facilita la manipulación y el análisis de datos. Ofrece estructuras de datos flexibles como DataFrames y Series que permiten una gestión sencilla de datos tabulares, incluyendo su limpieza, transformación y agregación.
- **Matplotlib**: Es una biblioteca de visualización en Python que permite la creación de gráficos estáticos, animados e interactivos. Matplotlib se utiliza extensamente para representar gráficamente datos y crear visualizaciones personalizadas de los resultados de análisis.
- **Gymnasium**: Desarrollada por OpenAI, esta biblioteca facilita el desarrollo y la evaluación de algoritmos de aprendizaje por refuerzo. Gymnasium ofrece una variedad de entornos de simulación donde los agentes pueden ser entrenados y evaluados.
- **Stable Baselines 3**: Esta biblioteca de Python proporciona implementaciones de alta calidad para algoritmos de aprendizaje por refuerzo. Basada en OpenAI Baselines, Stable Baselines 3 ofrece implementaciones eficientes y accesibles de algoritmos como PPO, A2C y DQN.
- **EnergyPlus**: Es un software para la simulación energética de edificios que modela el consumo de energía, el confort térmico y la calidad del aire interior. EnergyPlus permite realizar simulaciones detalladas de sistemas energéticos en edificios y es ampliamente usado en la investigación sobre eficiencia energética.

- **Sinergym**: Esta biblioteca combina el simulador EnergyPlus con entornos de aprendizaje por refuerzo. Sinergym proporciona una interfaz para integrar EnergyPlus con algoritmos de aprendizaje por refuerzo, permitiendo la optimización de sistemas de edificios mediante técnicas de aprendizaje automático.

Además de las bibliotecas mencionadas, el proyecto ha hecho uso de tecnologías de **contenerización** para facilitar el desarrollo y la gestión de entornos de ejecución. En particular, se ha utilizado **Docker**, una plataforma que permite crear, desplegar y ejecutar aplicaciones en contenedores. En el contexto de este proyecto, Docker ha sido crucial para trabajar con la biblioteca Sinergym. Dicho proyecto ofrece un contenedor que incluye todas las dependencias necesarias para su funcionamiento. Con esto lo que conseguimos es una mayor flexibilidad y reproducibilidad en el desarrollo del proyecto, garantizando que todos los miembros del equipo y las instancias de ejecución operen en un entorno uniforme y controlado.

Es importante destacar que todo el software utilizado en este proyecto es de código abierto. Este tipo de software se caracteriza por su enfoque colaborativo, permitiendo que una comunidad activa de desarrolladores lo mejore continuamente. Ofrece amplias opciones y soporte técnico, brindando a los desarrolladores la libertad de elegir las herramientas que mejor se adapten a sus necesidades.

2.4.4. Recursos económicos

Desde una perspectiva económica, los **gastos del proyecto** están vinculados principalmente al sueldo del alumno, Francisco Pertíñez Perea, quien tiene un contrato a tiempo parcial con una duración de 6 meses. Este gasto se detalla en la Tabla 2.2. A continuación, se describen los aspectos clave para su cálculo:

- El salario mensual tanto bruto como neto, y el IRPF, se han obtenido directamente de la nómina del trabajador.

- Los porcentajes de cotización a la Seguridad Social se han consultado en la web del Ministerio de Inclusión, Seguridad Social y Migraciones con fecha de agosto de 2024.

- El coste mensual para la empresa se ha consultado en el documento de bases de cotización del Vicerrectorado de Investigación y Transferencia, para un contrato a tiempo parcial para una persona con un grado universitario.

Concepto	**Coste**
Sueldo bruto anual (€)	15.264,00
Número de pagas (€)	6
Sueldo bruto mensual (€)	1.272,00
Cotización a la Seguridad Social (% empresa)	23,60
Cotización a la Seguridad Social (% trabajador)	4,70
IRPF (%)	3,21
Sueldo neto mensual del empleado (€)	**1.163,63**
Coste mensual para la empresa (€)	**1.748,48**

Tabla 2.2: Desglose de gastos del proyecto

Considerando una duración del contrato de 6 meses, el coste total del proyecto se estima en 1.748,48 €× 6 = 10.490,88 €. Además, si tenemos en cuenta el valor del portátil utilizado para el desarrollo del proyecto, que es de 1.400 €, el **coste total** del proyecto asciende a 10.490,88 €+ 1.400 €= 11.890,88 €.

Capítulo 3

Marco teórico

En este capítulo se dará una visión general del campo del aprendizaje por refuerzo y el aprendizaje profundo aplicado a este. Se explicarán los conceptos fundamentales relacionados con estas técnicas y su aplicación en el control eficiente de sistemas HVAC. Finalmente se realizará un revisión de la bibliografía más relevante, para así situar el trabajo en el contexto del estado del arte.

3.1. Aprendizaje por refuerzo

El **aprendizaje por refuerzo** (Reinforcement learning, RL) es una técnica de aprendizaje automático en la que un agente aprende a tomar decisiones interactuando con un entorno dinámico y recibiendo señales de recompensa. El objetivo es maximizar la recompensa acumulada a largo plazo mediante un proceso de prueba y error, ajustando sus estrategias basadas en las recompensas obtenidas [33].

A diferencia del **aprendizaje supervisado**, que se basa en entrenar un modelo utilizando un conjunto de datos etiquetados proporcionados por un supervisor externo, el RL no depende de ejemplos específicos de comportamiento correcto. En el aprendizaje supervisado, cada ejemplo en el conjunto de entrenamiento incluye una descripción de una situación junto con una especificación de la acción correcta que el sistema debe tomar en esa situación. El objetivo es que el sistema pueda extrapolar o generalizar sus respuestas para actuar correctamente en situaciones no presentes en el conjunto de entrenamiento. Sin embargo, en problemas interactivos, a menudo es impráctico obtener ejemplos de comportamiento deseado que sean correctos y representativos de todas las situaciones en las que el agente debe actuar. En ambientes o problemas donde no se tiene información previa sobre cuál es la acción correcta, es crucial que el agente sea capaz de adaptarse

y mejorar su desempeño a través de la experiencia directa. Es aquí donde el RL sobresale sobre el aprendizaje supervisado.

El **aprendizaje no supervisado**, por otro lado, se centra en encontrar estructuras ocultas en colecciones de datos no etiquetados. Aunque podría parecer que el RL es una forma de aprendizaje no supervisado debido a que no se basa en ejemplos de comportamiento correcto, en realidad, el RL se enfoca en maximizar una señal de recompensa en lugar de encontrar estructuras ocultas. Descubrir estructuras en la experiencia de un agente puede ser útil en el RL, pero por sí solo no aborda el problema de maximizar una señal de recompensa.

Por lo tanto, se considera que el RL es un **tercer paradigma** dentro del aprendizaje automático, junto al aprendizaje supervisado y no supervisado.

3.1.1. Marco de trabajo en RL

Entre los muchos aportes que Richard Bellman hizo al campo del RL, uno de los más destacados fue la introducción de un marco estándar fundamental para formular y abordar estos problemas [47]. Este marco proporciona un conjunto de términos y conceptos esenciales que permiten describir cualquier problema de RL de manera coherente y sistemática.

Los elementos que conforman este marco son los siguientes:

- **Objetivo**: Define el propósito general que el sistema de RL debe alcanzar. Puede ser maximizar una recompensa a largo plazo, minimizar costos, o cualquier otra métrica que represente el éxito en el problema específico abordado.

- **Entorno**: Es el contexto en el que opera el agente de RL. Este entorno puede ser físico (como un robot en un espacio físico) o virtual (simulado por software), y proporciona datos relevantes para la toma de decisiones del agente.

- **Estado**: Representa la situación actual del entorno en un momento dado. Es una descripción de los datos relevantes que el agente utiliza para tomar decisiones. El espacio de estados puede ser finito o infinito, y cada estado se define por un conjunto finito de variables.

- **Agente**: Es el componente de RL que interactúa con el entorno y aprende de él. El agente toma decisiones secuenciales para alcanzar el objetivo perseguido utilizando algoritmos y estrategias que seleccionen acciones en función de los estados observados y la retroalimentación recibida.

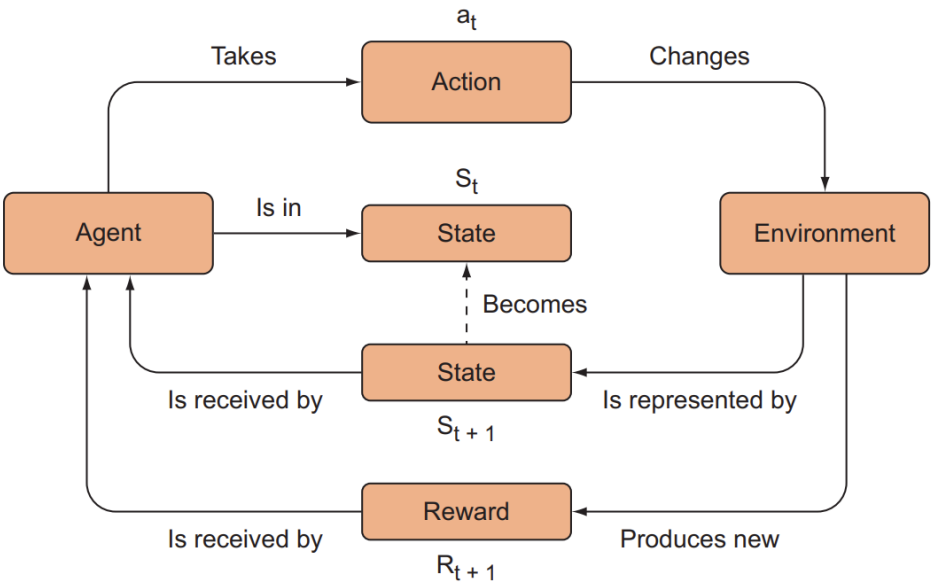


Figura 3.1: Marco de trabajo estándar en RL formulado por Bellman [47]

- **Acción**: Se refiere a las decisiones que el agente toma en respuesta a cada estado del entorno, permitiendo que el sistema transite a otro estado. Estas acciones pueden ser simples (como moverse en una dirección específica) o complejas (como ajustar parámetros en un sistema).
- **Recompensa**: Es la retroalimentación inmediata que el agente recibe del entorno después de tomar una acción en un estado particular. La recompensa puede ser positiva (indicando que la acción fue beneficiosa) o negativa (indicando lo contrario). Es fundamental para que el agente aprenda y ajuste su comportamiento con el fin de alcanzar el objetivo a largo plazo.

Tal y como se muestra en la Figura 3.1, estos elementos se combinan en un ciclo continuo de interacción entre el agente y el entorno: El agente observa el estado actual, selecciona una acción, ejecuta esa acción en el entorno, recibe una recompensa y actualiza su modelo o estrategia en función de la retroalimentación recibida. Este proceso iterativo permite que el agente mejore gradualmente su rendimiento y aprenda a tomar decisiones óptimas en el entorno dado.

3.1.2. Procesos de decisión de Markov

Hemos examinado el marco de trabajo para abordar problemas de RL. No obstante, para alcanzar una mayor formalidad, es necesario un modelo

matemático que nos permita definir estos problemas de manera precisa. Ante esta necesidad, el Proceso de Decisión de Markov (Markov Decision Process, MDP) se presenta como una herramienta muy conveniente. Un MDP es un modelo matemático diseñado para resolver problemas de toma de decisiones secuenciales en entornos con incertidumbre.

Un MDP se define como la tupla que contiene los siguientes elementos:

- $\mathcal{S}$: **Conjunto de estados** $\mathcal{S} = \{s_1, s_2, ..., s_n\}$, que representa todas las posibles situaciones o configuraciones en las que puede encontrarse el sistema.
- $\mathcal{A}$: **Conjunto de acciones** $\mathcal{A} = \{a_1, a_2, ..., a_m\}$, que son las decisiones disponibles para el agente en cualquier estado dado.
- $\mathcal{P}$: **Función de transición** $\mathcal{P}(s' \mid s, a)$, que especifica la probabilidad de transición de un estado s a un estado s' al ejecutar la acción a. Esta función modela cómo evoluciona el sistema en respuesta a las acciones del agente.
- $\mathcal{R}$: **Función de recompensa** $\mathcal{R}(s, a, s')$, que determina la recompensa inmediata recibida por el agente después de realizar la acción a y alcanzar el estado s'. Es crucial para guiar al agente hacia acciones que maximicen la recompensa total esperada a largo plazo.
- γ: **Factor de descuento** γ, que determina la importancia de las recompensas futuras en comparación con las recompensas inmediatas.

Propiedad de Markov

Una de las principales razones por las cuales los MDP se emplean ampliamente para formular problemas de RL radica en la propiedad fundamental que estos modelos exhiben. Esta propiedad establece que el **estado futuro de un sistema depende exclusivamente del estado presente**, sin ser afectado por los estados anteriores que llevaron al sistema a su situación actual. Formalmente, un proceso estocástico tiene la propiedad de Markov si la distribución condicional del estado futuro depende solo del estado presente, es decir:

$$P(s'|s, a) \doteq P(s'|s, a, S_{t-1}, A_{t-1}, S_{t-2}, A_{t-2}, ...) \tag{3.1}$$

En el contexto del RL, esta característica es esencial para la **eficiencia** computacional y la **escalabilidad** de los algoritmos de RL, ya que reduce la complejidad del problema al enfocarse en la relevancia inmediata de las acciones tomadas. Además, facilita la implementación de estrategias de

aprendizaje que maximizan la recompensa a largo plazo al considerar adecuadamente el valor esperado de las futuras interacciones con el entorno.

Es importante señalar que, en ciertos casos, **un problema puede no cumplir de forma natural con la propiedad de Markov, pero podemos inducirla**. Por ejemplo, como ya se ha comentado anteriormente, en este trabajo de fin de máster pretendemos expandir el espacio de estados de un agente de DRL incorporando información contextual adicional (información de estados pasados, previsiones meteorológicas), con el fin de mejorar el rendimiento de este. Al integrar estos elementos contextuales, buscamos tanto inducir la propiedad de Markov como al mismo tiempo ofrecer al agente una representación más completa y detallada del entorno. Este enfoque permite que problemas que no aparentan cumplir con la propiedad de Markov se ajusten a los principios de los MDP, facilitando así la aplicación de algoritmos de RL de manera más efectiva.

Función de transición

La propiedad de Markov es fundamental porque simplifica la modelización de la incertidumbre y la toma de decisiones secuenciales en entornos dinámicos y estocásticos. Gracias a esta propiedad, podemos definir la **función de transición** τ de la siguiente forma:

$$\tau(s, a, s') \doteq p(s'|s, a) = P(S_t = s'|S_{t-1} = s, A_{t-1} = a) \tag{3.2}$$

Esta función describe las probabilidades de transición asociadas a un estado s y una acción a dados. En otras palabras, τ indica la probabilidad de que el sistema se desplace al estado s' después de que el agente ejecute la acción a mientras se encuentra en el estado s.

En problemas **determinísticos**, la función de transición τ asigna una probabilidad de 1 a un único estado s' cuando se realiza una acción a desde un estado específico s. Por el contrario, en problemas **estocásticos**, existen diferentes probabilidades de transición hacia diversos estados futuros s', reflejando la aleatoriedad inherente en las dinámicas del entorno. En cualquier casos, la suma de las probabilidades de transición hacia todos los posibles estados futuros s' debe ser igual a 1:

$$\sum p(s'|s, a) = 1 \tag{3.3}$$

Función de recompensa

Comúnmente denotada como $R(s, a, s')$, determina la recompensa inmediata que el agente recibe al ejecutar la acción a desde el estado s y alcanzar

el estado s'. En otras palabras, R asigna un valor numérico que refleja el beneficio o costo asociado con cada transición de estado bajo una acción específica.

La función de recompensa puede formularse de la siguiente manera:

$$r(s, a, s') \doteq \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a, S_t = s'] \tag{3.4}$$

Esta fórmula expresa que la recompensa inmediata esperada depende de la acción tomada y el cambio de estado asociado, considerando el valor esperado de esa recompensa en función del estado y acción previos.

Factor de descuento

El propósito del agente es maximizar la recompensa acumulada que obtiene a lo largo de un episodio. Siendo T la cantidad total de *pasos* en un episodio, la recompensa acumulada se calcula como la suma de todas las recompensas recibidas a lo largo del episodio, es decir:

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \ldots + R_T \tag{3.5}$$

En el proceso de calcular la recompensa total, es habitual utilizar un factor de descuento, γ, que modula la relevancia de las recompensas en el tiempo. Este factor pondera más las recompensas cercanas en comparación con las futuras, representando la incertidumbre y el costo del tiempo al momento de tomar decisiones.

Para expresar la recompensa acumulada G_t con el factor de descuento γ, podemos escribir:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-1} R_T \tag{3.6}$$

Esta suma representa la suma ponderada de todas las recompensas futuras a partir del tiempo $t+1$, donde cada recompensa R_{t+k+1} se multiplica por γ^k para reflejar su valor actualizado en el tiempo.

El factor de descuento γ generalmente satisface $0 \leq \gamma \leq 1$. Un valor cercano a 1 indica que las recompensas futuras tienen un peso significativo en el cálculo de G_t, mientras que un valor cercano a 0 significa que solo las recompensas cercanas en el tiempo tienen un impacto considerable.

Extensiones del MDP

El marco de MDP se puede extender de varias maneras para abordar diferentes tipos de problemas en el aprendizaje por refuerzo [23]. A conti-

nuación se presentan algunas de las principales extensiones del marco MDP:

- **Proceso de Decisión de Markov Parcialmente Observable (POMDP):** En situaciones donde el agente no puede observar completamente el estado del entorno, se utiliza el POMDP. Este marco maneja la incertidumbre sobre el estado actual del entorno y permite tomar decisiones basadas en observaciones parciales.
- **Proceso de Decisión de Markov Factorizado (FMDP):** El FMDP permite representar de manera más compacta las funciones de transición y recompensa, facilitando la representación de MDPs grandes y complejos.
- **Proceso de Decisión de Markov con Tiempo/Acción/Estado Continuo:** Esta extensión aborda casos en los que el tiempo, las acciones o los estados son continuos en lugar de discretos. Permite una modelización más detallada de entornos en los que estos componentes varían continuamente.
- **Proceso de Decisión de Markov Relacional (RMDP):** El RMDP combina el conocimiento probabilístico con el conocimiento relacional, permitiendo representar y razonar sobre los problemas en términos de relaciones y propiedades de los objetos en el entorno.
- **Proceso de Decisión de Markov Semi-Markoviano (SMDP):** En el SMDP, se permiten acciones abstractas que pueden durar varios pasos de tiempo en completarse. Esto es útil para modelar situaciones en las que las acciones tienen una duración variable y no instantánea.
- **Proceso de Decisión de Markov Multiagente (MMDP):** Esta extensión incluye múltiples agentes interactuando dentro del mismo entorno. El MMDP permite estudiar cómo los agentes cooperan o compiten entre sí para maximizar sus recompensas.
- **Proceso de Decisión de Markov Descentralizado (Dec-MDP):** El Dec-MDP es una extensión del MMDP que se centra en la colaboración entre múltiples agentes para maximizar una recompensa común. Los agentes en un Dec-MDP trabajan conjuntamente, a menudo con información descentralizada, para alcanzar objetivos comunes.

Es importante señalar que en la práctica, a menudo se encuentran diferentes definiciones y enfoques para los MDPs y sus extensiones, lo que puede causar confusión. Por ejemplo, los POMDPs a veces se refieren simplemente como MDPs en algunos textos, sin especificar que se trata de una variante con observaciones parciales.

En un POMDP, se añaden los siguientes componentes a los de un MDP:

- **Espacio de Observaciones O:** Este componente define el conjunto de todas las posibles observaciones que el agente puede recibir sobre el entorno. Dado que el agente no puede observar el estado real del entorno de manera completa, el espacio de observaciones proporciona información parcial que el agente utiliza para inferir el estado del entorno.

- **Probabilidad de Emisión ε:** La probabilidad de emisión especifica la probabilidad de que el agente reciba una observación particular o_t dado el estado real s_t. Formalmente, $\varepsilon(o_t|s_t)$ representa la probabilidad de observar o_t cuando el sistema está en el estado s_t. Esta función es esencial en un POMDP porque permite al agente actualizar su creencia sobre el estado del entorno en función de las observaciones parciales que recibe.

Esta extensión es fundamental para abordar problemas en los que la información disponible es incompleta y el agente debe tomar decisiones basadas en observaciones parciales, como es el caso de problemas de control HVAC.

3.1.3. Políticas y funciones de valor

Con el modelo matemático del problema de RL ya establecido, el próximo paso es diseñar una estrategia efectiva para resolverlo. Ante este problema, la política desempeña un papel crucial al determinar cómo el agente interactúa con su entorno.

Políticas

Una política, denotada como π, es la estrategia que un agente sigue en un entorno dado. En términos formales, una política es una función que mapea un estado a una distribución de probabilidad sobre el conjunto de acciones posibles en ese estado:

$$\pi : s \to \Pr(A \mid s), \quad \text{donde } s \in S \tag{3.7}$$

Aquí, s es un estado y $\Pr(A \mid s)$ es una distribución de probabilidad sobre el conjunto de acciones A, dado el estado s. La probabilidad de cada acción en la distribución define cómo es de probable que dicha acción produzca la mayor recompensa esperada.

Una política óptima, denotada como π^*, es aquella que maximiza las recompensas esperadas:

$$\pi^* = \arg\max \mathbb{E}(R \mid \pi) \tag{3.8}$$

La política óptima es la estrategia que, cuando se sigue, produce las máximas recompensas posibles, basadas en las recompensas esperadas bajo cualquier política posible.

Funciones de Valor

El objetivo central de un algoritmo de RL es seleccionar acciones que maximicen las recompensas esperadas. Para lograr esto, podemos entrenar a nuestro agente de dos formas distintas [47]:

- **Directamente**: Enseñamos al agente a aprender cuáles acciones son mejores según el estado actual.
- **Indirectamente**: Enseñamos al agente a valorar qué estados son más beneficiosos y luego tomar acciones que conduzcan a esos estados de mayor valor.

Este enfoque indirecto introduce al concepto de funciones de valor.

Función de Estado-Valor

La función de estado-valor mapea un estado al valor esperado (la recompensa esperada) de estar en ese estado y seguir una política específica. Formalmente, una función de estado-valor se define como:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] \tag{3.9}$$

Para aclarar mejor esta sección, presentamos un ejemplo tomado del libro de Alexander Zai y Brandon Brown, *Deep Reinforcement Learning in Action*, que expone lo siguiente:

> "Si nuestra política fuera elegir acciones de manera completamente aleatoria (es decir, muestrear acciones de una distribución uniforme), el valor de un estado sería probablemente bajo, ya que no estaríamos seleccionando las mejores acciones posibles. En cambio, buscamos utilizar una política cuya distribución de probabilidad maximice las recompensas esperadas al ser muestreada. La política determina las recompensas observadas, y la función de valor refleja estas recompensas observadas."

La función de estado-valor óptima, denotada como $v_*(s)$, es aquella que proporciona el máximo valor esperado entre todas las políticas para cada estado s:

$$v_*(s) = \max_{\pi} v_\pi(s), \quad \forall s \in S \tag{3.10}$$

Esta función de estado-valor óptima representa el valor máximo alcanzable desde cualquier estado bajo la mejor política posible.

Función de Acción-Valor

Las funciones de acción-valor, también conocidas como funciones q, se definen como:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \tag{3.11}$$

La función q_π asigna un valor esperado a un par (s, a), donde s es un estado y a es una acción, representando la recompensa esperada al tomar la acción a en el estado s, bajo la política π.

La función de acción-valor óptima, denotada como $q_*(s, a)$, es aquella que ofrece el máximo valor esperado entre todas las políticas para cada par estado-acción (s, a):

$$q_*(s, a) = \max_{\pi} q_\pi(s, a), \quad \forall s \in S, \forall a \in \mathcal{A}(s) \tag{3.12}$$

Esta función de acción-valor óptima representa el valor máximo alcanzable desde cualquier estado y al tomar cualquier acción bajo la mejor política posible.

3.1.4. Evaluación y mejora de políticas

Si realizamos un desarrollo recursivo de la función estado-valor, obtenemos la conocida **ecuación de Bellman** [3].

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t | S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']] \\ &= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a)[r + \gamma v_\pi(s')], \forall s \in \mathcal{S} \end{aligned} \tag{3.13}$$

Esta ecuación establece una relación entre el valor de un estado y el valor esperado de sus estados sucesores, de manera que el valor de un estado s se calcula como la suma ponderada de todas las posibles transiciones desde ese estado más la recompensa esperada descontada del siguiente estado.

Análogamente, para la función acción-valor tambien podemos realizar un desarrollo recursivo, resultando en:

$$\begin{aligned} q_\pi(s,a) &\doteq \mathbb{E}_\pi[G_t|S_t=s,A_t=a] \\ &= \mathbb{E}_\pi[R_{t+1}+\gamma G_{t+1}|S_t=s,A_t=a] \\ &= \sum_{s',r} p(s',r|s,a)[r+\gamma v_\pi(s')], \forall s\in\mathcal{S}, \forall a\in\mathcal{A}(s) \end{aligned} \tag{3.14}$$

Las ecuaciones de Bellman pueden abordarse utilizando **programación dinámica** (Dynamic Programming en inglés, DP) [4], algoritmos que utilizan funciones de valor para buscar y perfeccionar estrategias óptimas. En esencia, los algoritmos de DP se derivan al convertir ecuaciones de Bellman en reglas de actualización para mejorar las aproximaciones de las funciones de valor deseadas y están muy relacionadas con el RL.

A continuación se va a realizar una revisión de dichos algoritmos [33].

Algoritmo de evaluación iterativa de políticas

El algoritmo de **evaluación iterativa de la política** nos permite determinar la efectividad de una política específica, basándonos en la recompensa acumulada esperada que se obtiene al seguir dicha política. Este algoritmo (ver Algoritmo 1) aproxima iterativamente la función de *estado-valor* de la política en evaluación, utilizando la siguiente fórmula:

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r+\gamma v_k(s')] \tag{3.15}$$

En cada iteración $k+1$, el valor del estado s se actualiza considerando todas las acciones a posibles desde ese estado, ponderadas por la probabilidad de tomar cada acción bajo la política $\pi(a|s)$. Para cada acción, se suma sobre todos los posibles estados siguientes s' y recompensas r, ponderados por la probabilidad conjunta $p(s',r|s,a)$ de llegar al estado s' y recibir la recompensa r al tomar la acción a desde el estado s. El término $[r+\gamma v_k(s')]$ representa la recompensa inmediata r más el valor descontado del estado siguiente $v_k(s')$, donde γ es el factor de descuento.

Algoritmo 1: Evaluación Iterativa de la Política

Entrada: π: política a evaluar
$\mathcal{S}$: espacio de estados
γ: factor de descuento
Salida: V: valores finales asociados a cada estado

$V(s) = 0, \forall s \in \mathcal{S}+$
$V(\text{terminal}) = 0$
repetir
  $\Delta \leftarrow 0$
  para cada $s \in \mathcal{S}$ **hacer**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
    $\Delta \leftarrow \text{máx}(\Delta, |v - V(s)|)$
mientras que $\Delta < \theta$
devolver V

Algoritmo de mejora iterativa de políticas

Una vez que sabemos cómo *evaluar* políticas, podemos realizar comparaciones entre ellas. Esto nos permite comenzar con cualquier política inicial e intentar *optimizarla* de manera iterativa utilizando la función de *valor de acción*. Para lograrlo, aplicaremos el algoritmo de **mejora iterativa de políticas** (ver Algoritmo 2), que se describe con la siguiente fórmula:

$$\pi'(s) = \underset{a}{\text{argmax}} \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_\pi(s')] \tag{3.16}$$

Dado un estado s, la nueva política $\pi'(s)$ selecciona la acción a que maximiza el valor esperado de la suma de la recompensa inmediata r y el valor descontado del estado siguiente $v_\pi(s')$. La probabilidad conjunta $p(s', r \mid s, a)$ de llegar al estado s' y recibir la recompensa r al tomar la acción a desde el estado s se usa para ponderar estas recompensas y valores. La función $\underset{a}{\text{argmax}}$ indica que se elige la acción que resulta en el valor máximo de esta suma.

Iteración de política

Al aplicar repetidamente los procesos de evaluación y mejora de la política en un ciclo, logramos refinar π de manera iterativa. Este método se denomina **iteración de políticas** (*policy iteration*) y se basa en alternar entre

Algoritmo 2: Mejora iterativa de la política

Entrada: π: la política a evaluar
$\mathcal{S}$: el espacio de estados del problema
γ: el factor de descuento
Salida: V_*: los valores finales asociados a cada estado
π_*: la política óptima obtenida

```
1  politica_estable ← verdadero
2  para cada s ∈ S hacer
3  |  a_0 ← π(s)
4  |  π(s) ← argmax_a ∑_{s',r} p(s', r|s, a)[r + γV(s')]
5  |  si a_0 ≠ π(s) entonces
6  |  |  politica_estable ← falso
7  si politica_estable entonces
8  |  devolver V ≈ v_* y π ≈ π_*
9  si no
10 |  aplicar evaluación de la política y repetir
```

las fases de *evaluación* y *mejora* hasta que se converge en la política óptima ($\pi \approx \pi_*$).

Iteración de valor

Una alternativa a la iteración de políticas es el algoritmo de **iteración de valor** (*value iteration*), que se ejecuta de la siguiente manera (ver Algoritmo 3):

1. Determinar la función *estado-valor* óptima, v_*. El algoritmo comienza con una función de valor inicial arbitraria, la cual se ajusta de manera iterativa hasta alcanzar el valor óptimo.

2. Obtener la política óptima, π_*, asociada a v_*. Tras obtener la función de valor óptima, la política óptima π_* se define como aquella que actúa de manera voraz con respecto a v_*.

3.1.5. Exploración vs explotación

En problemas de RL, y en general en cualquier problema de optimización, uno de los dilemas fundamentales que enfrenta un agente es el de exploración

Algoritmo 3: Iteración de valor

Entrada: π: la política a evaluar
$\mathcal{S}$: el espacio de estados del problema
γ: el factor de descuento
θ: un umbral de convergencia

Salida: V_*: los valores finales asociados a cada estado
π_*: la política óptima obtenida

$V(s) = 0, \forall s \in \mathcal{S}+$
$V(terminal) = 0$
repetir
  $\Delta \leftarrow 0$
  para cada $s \in \mathcal{S}$ **hacer**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
    $\Delta \leftarrow max(\Delta, |v - V(s)|)$
mientras que $\Delta < \theta$
$\pi_*(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
devolver V_*, π_*

frente a explotación. Esta cuestión surge debido a la necesidad de equilibrar dos objetivos contrapuestos:

- **Exploración**: Implica que el agente debe intentar nuevas acciones y visitar diferentes estados para adquirir más información sobre el entorno. Esta información puede ayudar a descubrir políticas mejores o más óptimas a largo plazo. La exploración es crucial en las primeras etapas del aprendizaje, ya que el agente tiene un conocimiento limitado del entorno y necesita recopilar datos sobre las recompensas asociadas con diferentes acciones y estados.

- **Explotación**: Consiste en que el agente utiliza su conocimiento actual para tomar decisiones que maximicen su recompensa inmediata. Esto significa elegir las acciones que, según la política o el modelo actual, parecen ser las más prometedoras en términos de recompensa acumulada. La explotación es esencial cuando el agente ya ha adquirido suficiente conocimiento del entorno y busca optimizar su desempeño.

Un equilibrio adecuado entre exploración y explotación es esencial para el éxito en el RL. Si el agente se enfoca demasiado en la exploración, puede perder oportunidades de obtener recompensas altas en el corto plazo. Por

otro lado, si se centra excesivamente en la explotación, puede quedarse atrapado en una política subóptima sin descubrir alternativas potencialmente mejores.

Existen varios enfoques y estrategias para gestionar el equilibrio entre exploración y explotación:

- **Estrategia ϵ-greedy**: Es uno de los métodos más simples y comunes. Consiste en elegir una acción aleatoria con probabilidad ϵ (exploración) y la mejor acción conocida con probabilidad $1-\epsilon$ (explotación). El valor de ϵ puede ser fijo o decrecer con el tiempo a medida que el agente adquiere más experiencia.
- **Métodos de Políticas de Exploración**: Estrategias como el *Softmax* utilizan una función de probabilidad que selecciona acciones basadas en sus valores relativos, permitiendo una mezcla más suave entre exploración y explotación dependiendo de las diferencias en los valores de acción.

3.1.6. Algoritmos RL

En el campo del RL, los algoritmos se pueden clasificar según dos criterios principales: el uso de políticas y el uso de modelos. Estos conceptos son fundamentales para comprender cómo los algoritmos aprenden y toman decisiones. A continuación, se explica cada uno de estos tipos.

Tipología

Algoritmos On-Policy vs. Off-Policy Los algoritmos on-policy aprenden una política específica mientras recolectan experiencias del entorno utilizando esa misma política. Durante el entrenamiento, la política que se utiliza para interactuar con el entorno es la misma que se está tratando de mejorar. El utilizar la misma política tanto para actual como para aprender puede conducir a una mayor estabilidad en el aprendizaje. Sin embargo, este enfoque puede requerir más tiempo y datos para aprender una política óptima debido a la dependencia directa de la política actual.

Por otro lado, los algoritmos off-policy pueden aprender de experiencias recolectadas por cualquier política. Estos algoritmos no están restringidos a usar la misma política para explorar y aprender. Esto permite una mayor flexibilidad en la recolección de datos y la posibilidad de reutilizar experiencias previas para mejorar el aprendizaje. No obstante, los algoritmos off-policy también presentan desventajas, como una mayor complejidad en la implementación y ajuste de hiperparámetros.

Algoritmos Model-Based vs. Model-Free Los algoritmos model-based construyen un modelo explícito del entorno que representa cómo se comporta el entorno dado un estado y una acción. Este modelo puede ser utilizado para simular futuros estados y recompensas, permitiendo una planificación y toma de decisiones informada. Una de las principales ventajas de este enfoque es su eficiencia, ya que un modelo preciso permite la planificación a largo plazo y reduce la necesidad de interacción directa con el entorno. Sin embargo, construir un modelo preciso puede ser difícil y costoso en términos de datos y computación.

En contraste, los algoritmos model-free no intentan construir un modelo del entorno. En lugar de eso, aprenden directamente a partir de la experiencia y las recompensas observadas. Este enfoque simplifica el proceso de aprendizaje al no requerir un modelo del entorno, lo cual es particularmente útil en entornos complejos o parcialmente observables. No obstante, los algoritmos model-free también tienen sus desventajas, como la necesidad de más datos y tiempo para aprender una política óptima debido a la falta de planificación explícita.

Aprendizaje por refuerzo profundo

Los algoritmos que se han empleado en este proyecto utilizan redes neuronales, ubicándolos dentro del ámbito del DRL. Optamos por este tipo de algoritmos frente a otros por los siguientes motivos.

Primero, los métodos "clásicos", como los tabulares, resultan inviables en términos de tiempo y recursos computacionales. Estos métodos requieren almacenar en una tabla todas las combinaciones posibles de estados y acciones junto con sus valores estimados, lo cual es factible solo en problemas con espacios de estados y acciones muy reducidos debido a la explosión combinatoria. Por ello, es esencial utilizar métodos que se basan en la aproximación de funciones.

Además, el aprendizaje por refuerzo profundo ha demostrado ser extremadamente eficaz. Numerosos estudios han destacado su potencial, y en la última década, la mayoría de las investigaciones que abordan problemas similares al de este trabajo (revisaremos algunas de estas en la sección 3.3) utilizan métodos de DRL. Esto proporciona una sólida evidencia empírica de que estos modelos son capaces de ofrecer soluciones de alta calidad.

Para una referencia más detallada sobre los métodos "clásicos" en los que se basan los algoritmos aquí explicados, se puede consultar en [33, 47].

DQN Los algoritmos fundamentados en ***Deep Q-Networks*** (**DQN**) marcaron un avance crucial en el ámbito del aprendizaje por refuerzo, siendo

la primera aproximación al enfoque conocido hoy como DRL [21, 20]. Estos métodos emplean redes neuronales para aproximar la función de valor Q (ver ecuación 3.22) de manera no lineal. En lugar de utilizar una tabla, como en el Q-learning convencional, se hace uso de una red neuronal, como se muestra en la Figura 3.2.

En cuanto a la arquitectura de la red utilizada, la capa de entrada de la red neuronal tendrá un número de neuronas igual a la cantidad de variables presentes en un estado o observación, mientras que la capa de salida estará compuesta por tantas neuronas como acciones posibles puede tomar el agente. Alternativamente, algunas formulaciones permiten utilizar pares estado-acción como entradas, generando como salida el valor asociado a ese par.

Respecto al proceso de aprendizaje, la función de pérdida utilizada por DQN es el error cuadrático medio (MSE) de la diferencia entre el valor Q predicho y el valor objetivo, que se define como:

$$MSE = \mathbb{E}_\pi[(Q(S,A) - Q(S,A;\theta))^2] \tag{3.17}$$

Por tanto, el objetivo es medir la diferencia entre el valor real de Q y el valor proporcionado por la red, y luego aplicar el descenso del gradiente para optimizar la función de error. La actualización de los pesos de la red se realiza según:

$$\Delta\Theta = \Delta w = -\eta\frac{\partial E}{\partial w} \tag{3.18}$$

$$\frac{\partial E}{\partial w} = 2(Q(S,A) - Q(S,A;\Theta))\frac{\partial Q(S,A;\Theta)}{\partial w} \tag{3.19}$$

$$\frac{\partial Q(S,A;\Theta)}{\partial w} = \nabla_\Theta Q(S,A;\Theta) \tag{3.20}$$

$$\Delta\theta = -2\eta(Q(S,A) - Q(S,A;\Theta))\nabla_\Theta Q(S,A;\Theta) \tag{3.21}$$

Al no conocer $Q(S,A)$, se aplica la expresión utilizada en Q-learning:

$$Q(S_0,A_0) \leftarrow Q(S_0,A_0) + \alpha[R_1 + \gamma \max_{a\in A} Q(S_1,a) - Q(S_0,A_0)] \tag{3.22}$$

$$\Delta\Theta = \alpha[R + \gamma \max_{a\in A} Q(S',a;\Theta) - Q(S,A;\Theta)]\nabla_\Theta\, Q(S,A;\Theta) \tag{3.23}$$

Destacar que este metodo utiliza dos redes neuronales durante el proceso de entrenamiento:

- **Red de predicción**: es la encargada de calcular el valor $Q(S,A;\Theta)$.

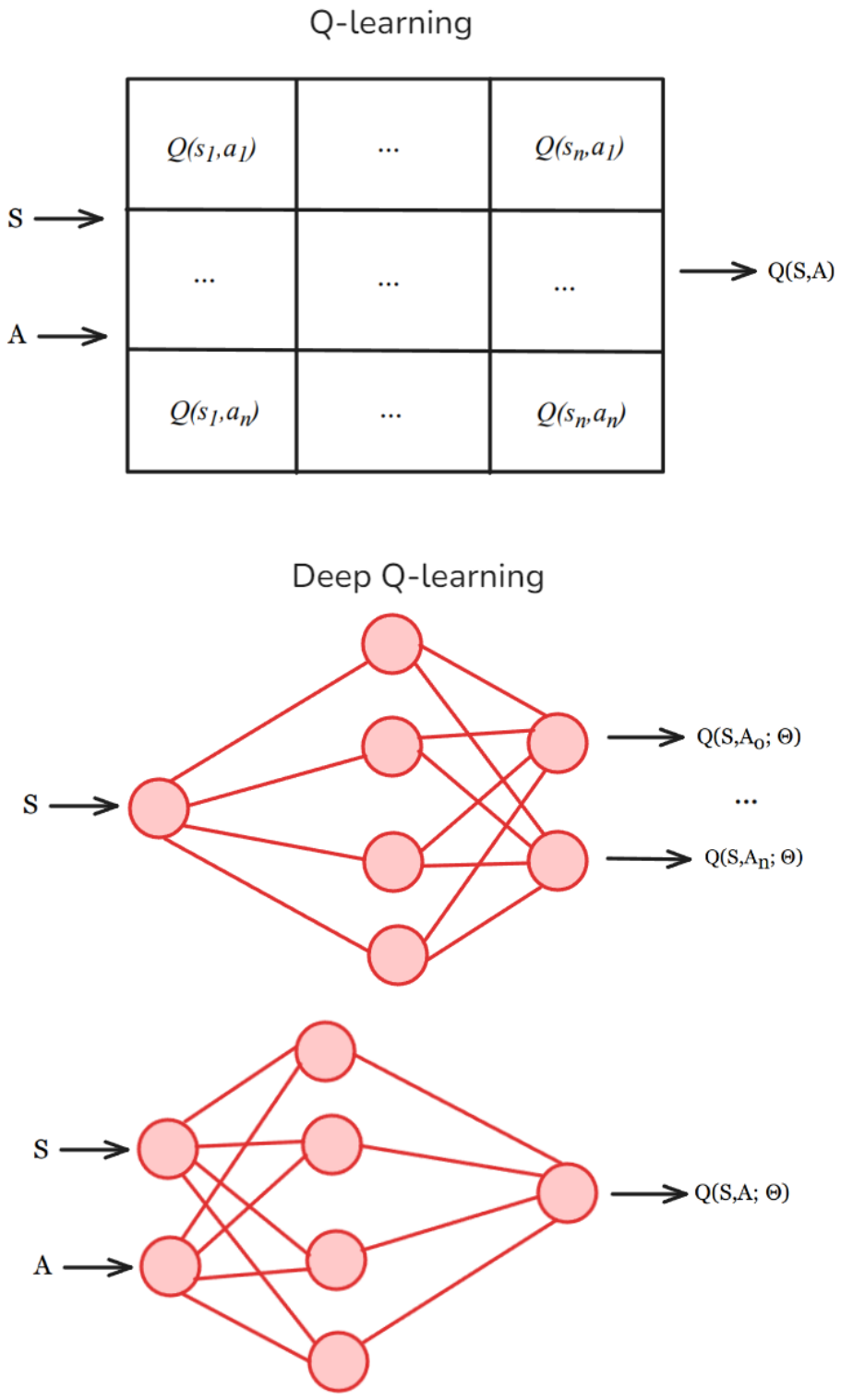


Figura 3.2: Comparación entre Q-learning y DQN en sus distintas formulaciones

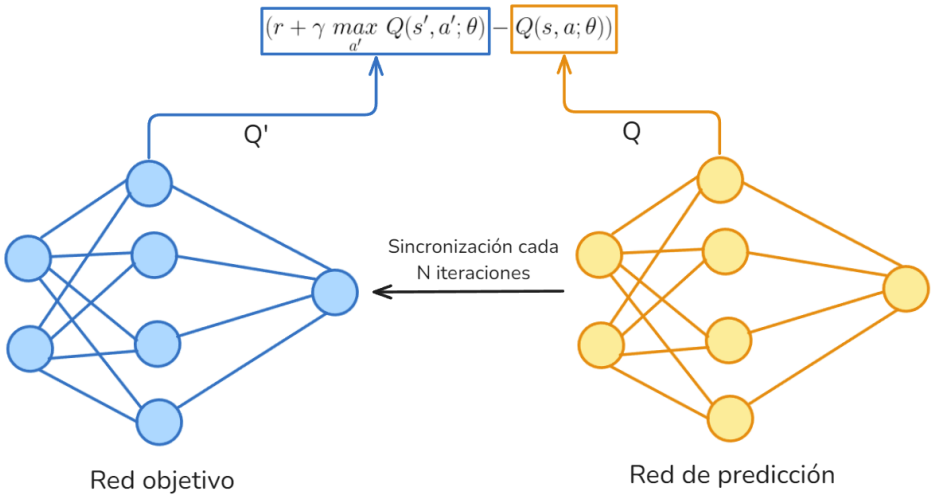


Figura 3.3: Uso de las redes de objetivo y predicción por DQN

- **Red objetivo**: su función es calcular el valor objetivo, lo que posibilita una evaluación imparcial del error. Se alinea con la red de predicción tras un número determinado de iteraciones, lo que implica que sus pesos se modifican con menor frecuencia.

El uso de dos redes responde a la necesidad de evitar la inestabilidad que surge cuando una sola red realiza tanto la predicción de los valores Q como el cálculo de los valores objetivos. Al actualizar los pesos de la red, los valores Q predichos cambian, y los valores objetivos se recalculan con estos nuevos pesos. Este ajuste mútuo puede dificultar la convergencia del proceso de aprendizaje haciéndolo inestable.

La introducción de la red objetivo ayuda a mantener los valores objetivos constantes durante las actualizaciones, lo que estabiliza el proceso de entrenamiento y mejora la convergencia hacia una solución óptima.

Además, es común combinar DQN con una **memoria de experiencias**, que almacena y utiliza diversas muestras de experiencias pasadas (S_t, A_t, R_t, S_{t+1}) como entradas para predecir nuevos valores. Esto se conoce como **repetición de la experiencia**, la cual se usa para mitigar un problema conocido como *olvido catastrófico*. Este ocurre cuando un modelo, al entrenar continuamente con datos nuevos, tiende a olvidar lo que aprendió anteriormente.

Una de las principales limitaciones de DQN es su manejo de espacios de acciones continuos, ya que, debido a la infinitud de posibilidades, no es posible identificar la acción óptima para calcular el error. Este problema puede abordarse considerando el par (s, a) como entrada o discretizando el espacio de acciones, aunque esto último representa una restricción significativa pa-

ra estos algoritmos. Como resultado, DQN generalmente no se emplea con espacios de acciones continuos.

El procedimiento de aprendizaje a través de DQN se sintetiza en el Algoritmo 4. Mencionar que existen varias versiones modificadas de DQN que buscan optimizar el enfoque original, entre las cuales destacan el *Double Q-Learning* [34], el *Prioritized replay* [31] y el *Dueling DQN* [40].

Algoritmo 4: DQN

Entrada: N: número de iteraciones necesarias para sincronizar las redes de predicción y objetivo.

```
Inicializar la memoria de experiencias.
Inicializar la red de predicción aleatoriamente.
Crear la red objetivo como una copia de la red de predicción.
k ← 0
para cada episodio hacer
    Inicializar el estado inicial
    para cada timestep hacer
        a ← seleccionar una acción (vía exploración o explotación)
          en base al valor de Q predicho.
        Ejecutar la acción a, y observar la recompensa r y el nuevo
          estado s'.
        Almacenar la tupla (s, a, r, s') en la memoria de experiencias.
        Seleccionar un lote (batch) aleatorio de la memoria de
          experiencias.
        Introducir batch en la red de predicción
        Obtener la salida de la red objetivo para s'.
        Calcular el error (loss) entre los valores Q de salida y
          objetivo.
        Aplicar descenso del gradiente para actualizar los pesos de la
          red de predicción y reducir el valor de loss.
        si k = N entonces
            Actualizar pesos de la red objetivo con los valores de los
              pesos de la red de predicción.
        k ← k + 1
```

PPO Los métodos que emplean **Optimización de la Política Próxima** (*Proximal Policy Optimization*, **PPO**) [32, 13] integran el gradiente de políticas dentro del aprendizaje por refuerzo. Este es un algoritmo *online*, lo que significa que no utiliza una memoria de experiencias. El procedimiento de aprendizaje sigue estos pasos:

1. Recopilar experiencia hasta completar un lote o *batch*.
2. Usar esa experiencia para optimizar la política.
3. Desechar el *batch* utilizado y regresar al paso 1.

Dado que es un método *online*, cada *batch* se usa solo una vez para actualizar el gradiente y luego se descarta. Esto hace que los métodos basados en gradiente tiendan a ser menos eficientes en el aprendizaje en comparación con los métodos *offline* como DQN.

A continuación vamos a explicar en qué consiste este proceso de aprendizaje. PPO se basa en la **función de acción-ventaja**, que mide qué tan beneficiosa es una acción a en función de la recompensa esperada al seguir la política π:

$$a_\pi(s,a) = q_\pi(s,a) - v_\pi(s) \tag{3.24}$$

A partir de esta función de acción-ventaja, definimos la función de pérdida como:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t[\log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{3.25}$$

donde π_θ es la política seguida y $\hat{A}_t$ es la función de acción-ventaja. Lo que buscamos con esto es favorecer aquellas acciones que resulten en una ventaja positiva, aumentando así la probabilidad de seleccionar nuevamente la acción a desde el estado s.

Un posible problema de esta actualización de la política es la posible alta divergencia entre la política original y la actualizada. Para evitar esto, PPO utiliza **TRPO** (*Trust Region Policy Optimization*) y la **divergencia KL** (coeficiente de divergencia de Kullback-Leibler). TRPO es un algoritmo de optimización de políticas que mejora la estabilidad y eficiencia en el RL al maximizar la función de recompensa y mantener la política actualizada dentro de una región de confianza mediante restricciones matemáticas. Esta región se controla usando la divergencia KL, que mide la diferencia entre la política nueva y la anterior. TRPO utiliza esta medida para evitar grandes cambios en la política que podrían afectar negativamente el rendimiento, asegurando actualizaciones seguras y controladas.

Esto garantiza que la nueva política no difiera demasiado de la anterior. Así, el siguiente ratio:

$$r(\theta) = \frac{\pi_{\theta_{old}}(a|s)}{\pi_\theta(a|s)} \tag{3.26}$$

se usa en la función objetivo de TRPO:

$$J(\theta)^{TRPO} = E[r(\theta)\hat{A}_{\theta_{old}}(s,a)] \tag{3.27}$$

Si añadimos la restricción de que este ratio debe estar entre $1-\epsilon$ y $1+\epsilon$, obtenemos la función objetivo que PPO utiliza para actualizar la política:

$$J^{CLIP}(\theta) = \mathbb{E}[\text{mín}(r(\theta)\hat{A}_{\theta_{old}}(s,a), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{\theta_{old}}(s,a))] \tag{3.28}$$

De esta manera, el ratio se trunca en el rango $[1-\epsilon, 1+\epsilon]$ (usando la función *clip*), evitando grandes desviaciones en la política. PPO toma el valor mínimo entre el valor original y el truncado, asegurando así actualizaciones más estables y controladas.

3.2. Formulación de problemas de control HVAC

La viabilidad del control HVAC mediante DRL se demostró inicialmente utilizando espacios de acción reducidos y modelos de edificios simplificados [25, 51]. El objetivo principal es desarrollar una política de control óptima que maximice el confort de los ocupantes mientras minimiza el consumo de energía. Este problema se formula como un POMDP, y la meta es encontrar una política que abarque el mayor número posible de estados deseables, evitando acciones que impliquen un alto costo energético. Durante el proceso de optimización, también se pueden considerar parámetros adicionales como el precio de la energía y el uso de fuentes de energía renovables. No obstante, nos centraremos en el confort y el consumo energético como los objetivos principales.

El consumo energético es una magnitud física que se puede cuantificar fácilmente midiendo la energía utilizada por los sistemas de climatización durante su operación. En contraste, el confort no es una magnitud objetiva y requiere una definición clara de cómo será evaluado. En este caso, se considerará el confort térmico, dado que los sistemas de climatización solo pueden afectar la temperatura, y no otros factores como la humedad o la concentración de CO2. Por lo tanto, es crucial definir un rango de temperaturas adecuado para asegurar el confort térmico.

El confort puede medirse de diversas formas, tales como:

1. La distancia entre la temperatura actual y la deseada:

$$C(T, \hat{T}) = |T - \hat{T}| \tag{3.29}$$

2. La distancia al cuadrado:

$$C(T, \hat{T}) = (T - \hat{T})^2 \tag{3.30}$$

3. La distancia a un rango de confort objetivo:

$$C(T, T_{\text{lower}}, T_{\text{upper}}) = |T - T_{\text{lower}}| + |T - T_{\text{upper}}| \tag{3.31}$$

4. La distancia a la temperatura objetivo y al rango de confort según el estándar 55 de ASHRAE [1], que define un rango de temperatura aceptable de 23–26ºC para verano y 20–23.5ºC para invierno:

$$C(T, \hat{T}, T_{\text{lower}}, T_{\text{upper}}) = |T - \hat{T}| + (T - T_{\text{upper}})^2 + (T - T_{\text{lower}})^2 \tag{3.32}$$

En base a un conjunto de variables observadas que definen las condiciones ambientales del entorno (como la temperatura exterior e interior, la concentración de CO2, la humedad y la ocupación), se consideran los siguientes objetivos:

- En cuanto al consumo energético (medido en kWh), buscamos la política que lo minimice, es decir:

$$\pi^* = \underset{\pi_\theta}{\text{argmin}} \sum_{t=1}^{T} P_t \tag{3.33}$$

- Respecto al confort, buscamos minimizar la diferencia entre el estado actual del edificio y el objetivo. Esto se puede definir como:

$$\pi^* = \underset{\pi_\theta}{\text{argmin}} \sum_{t=1}^{T} C(S_t, S_{\text{target}}) \tag{3.34}$$

Un estado es considerado óptimo cuando las variables o condiciones ambientales que lo conforman se ajustan a las preferencias del usuario. Del mismo modo, se define como una *ruptura del confort* cuando la temperatura del estado actual excede los límites establecidos.

En las Formulas 3.34 y 3.33, P representa el consumo energético del equipo térmico del edificio (como bombas de calor y calderas), mientras que $C(S_t, S_{\text{target}})$ se refiere a la diferencia entre la temperatura actual y la deseada en las zonas controladas del edificio. Podemos combinar la minimización de la demanda de energía y la maximización del confort en una única expresión:

$$\pi^* = \underset{\pi_\theta}{\text{argmin}} \sum_{t=1}^{T} \omega \cdot C(S_t, S_{\text{target}}) + (1 - \omega) \cdot P_t \tag{3.35}$$

donde ω y $(1-\omega)$ son los pesos asignados al confort y a la demanda de energía, respectivamente. Con esto, podemos definir nuestra función de recompensa de la siguiente manera:

$$r(S_t, A_t) = (1-\omega) \cdot \lambda_P \cdot P_t + \omega \cdot \lambda_C \cdot C(S_t, S_{\text{target}}) \tag{3.36}$$

donde P_t es la demanda de energía del sistema HVAC en el instante de tiempo t, $C(S_t, S_{\text{target}})$ representa la distancia entre la temperatura actual de la zona y los límites de confort deseados, ω representa el peso asignado a cada parte de la recompensa, y λ_P y λ_C son factores constantes utilizados para escalar las magnitudes de la demanda de energía y la temperatura.

Es común expresar esta recompensa en términos negativos, ya que se busca su maximización, lo que nos lleva a reescribirla de la siguiente manera:

$$r(S_t, A_t) = -(1-\omega) \cdot \lambda_P \cdot P_t - \omega \cdot \lambda_C \cdot C(S_t, S_{\text{target}}) \tag{3.37}$$

Dependiendo las variables que se consideren en el problema, la función de recompensa tendrá una formulación u otra.

Problemas similares pueden abordarse con una configuración parecida, como la regulación del flujo de aire [29] o el control de la iluminación [7].

3.3. Revisión bibliográfica

En los últimos años, el uso de técnicas de RL y DRL para el control de sistemas HVAC ha experimentado un notable crecimiento. Si buscamos en LENS.ORG utilizando la cadena "`reinforcement learning`" `AND` "`HVAC`" revela el creciente interés en este campo de estudio.

La Figura 3.4 refleja claramente esta tendencia. Desde 2018, se ha observado un notable aumento en el número de publicaciones sobre este tema, alcanzando su punto más alto en 2023. Estos datos muestran el creciente interés de la comunidad científica y de ingeniería en la aplicación de estas tecnologías avanzadas en los sistemas HVAC.

Aunque Mozer fue pionero en 1998 al aplicar el aprendizaje por refuerzo en el control de edificios [26], el verdadero avance en este campo ha ganado protagonismo en años recientes, coincidiendo con el auge del DRL. Estudios recientes, como los de [39, 36, 44, 12, 16, 19, 28, 48, 42, 22], presentan un análisis exhaustivo del uso de DRL en la gestión energética de edificios y sistemas HVAC, explorando detalladamente diversas aplicaciones, enfoques y metas en este ámbito.

Al analizar los métodos de DRL empleados, se observan aplicaciones de algoritmos consolidados como Deep Q-Networks (DQN) [11, 43, 30, 17],

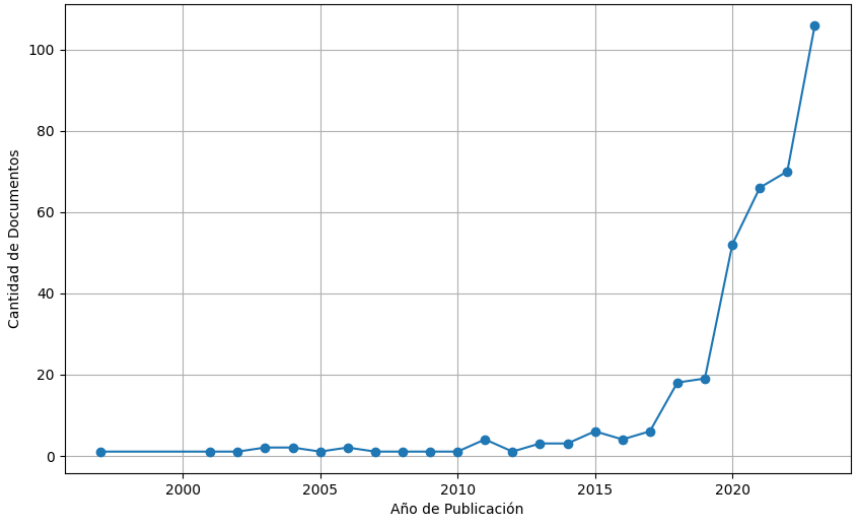


Figura 3.4: Número de publicaciones relacionadas con el aprendizaje por refuerzo y el control HVAC durante el período 1997-2023.

Deep Deterministic Policy Gradient (DDPG) [9, 52], y métodos actor-crítico [24, 38].

A continuación, se detallan algunos resultados cuantitativos obtenidos en estos y otros estudios:

- Un controlador RL adaptado a la demanda energética variable permitió un ahorro semanal del 22 % en [2].
- Vázquez-Canteli et al. alcanzaron un 10 % de ahorro energético aplicando DQN en el control de una bomba de calor [37, 35].
- Brandi et al. lograron un ahorro energético del 5 % al 12 % utilizando un agente basado en Double DQN [6].
- Zhang et al. lograron una reducción del 16.7 % en la demanda energética de un edificio de oficinas mediante el uso del algoritmo A3C [50, 51].
- Deng abordó el control HVAC en un entorno no estacionario, obteniendo hasta un 13 % de ahorro energético y un 9 % de mejora en confort [52].
- Yuan reportó una mejora del 4.7 % al 7.7 % al comparar con un control basado en reglas (RBC) y un sistema PID [46].
- Yu consiguió entre un 56 % y un 75 % de ahorro en un edificio con 30 zonas utilizando su algoritmo [45].

Un estudio reciente de Manjavacas et al. [18] identifica varios desafíos actuales en el campo del aprendizaje por refuerzo profundo. En primer lugar, al igual que lo señalaron Biemann et al. [5], subraya que la predominancia de algoritmos como DQN en problemas de control continuo podría estar frenando avances importantes. Métodos más modernos y efectivos, como Twin Delayed DDPG (TD3), Soft Actor-Critic (SAC) y Proximal Policy Optimization (PPO), aún no se explotan lo suficiente.

Además, muchas de las implementaciones actuales tienden a desarrollarse de manera aislada, enfocándose en comparar un solo algoritmo con una línea base simplista y resolviendo problemas muy específicos en contextos limitados. Esto ha llevado a la falta de estudios que evalúen comparativamente múltiples algoritmos de DRL bajo condiciones estandarizadas, una práctica común en otras áreas del aprendizaje automático.

El estudio de Manjavacas et al. aborda de manera integral estas limitaciones. No solo evalúa algoritmos tradicionales como DQN, sino que también incluye enfoques más recientes como TD3, SAC y PPO, proporcionando una perspectiva más amplia y actualizada sobre el rendimiento de estos métodos en problemas de control continuo.

Además, el estudio realiza comparaciones bajo condiciones estandarizadas, lo que permite una evaluación más rigurosa y coherente entre diferentes algoritmos de DRL.

Es importante destacar el uso de frameworks específicamente diseñados para este propósito como Sinergym [14], para abordar la falta de comparabilidad. Este software, desarrollado por la Universidad de Granada, facilita el entrenamiento, evaluación y comparación de modelos de DRL en un entorno de simulación común. Al establecer estándares claros para la evaluación de diferentes controladores, Sinergym permite comparaciones más consistentes y reproducibles entre algoritmos.

Por último, en el contexto de este proyecto, el **impacto del aumento de la dimensionalidad** de las entradas en el rendimiento del aprendizaje por refuerzo profundo ha sido investigado en estudios previos, como el de Ota et al. [27]. Sin embargo, la **literatura en esta área es todavía escasa cuando se trata de su aplicación específica al control de sistemas de climatización**, ya que no existen estudios centrados explícitamente en este tema. Este trabajo busca llenar ese vacío explorando cómo la dimensionalidad de las entradas afecta el rendimiento en el control de sistemas de climatización, aportando nuevas perspectivas y contribuciones a este campo de estudio.

Capítulo 4

Desarrollo de Sinergym

En este capítulo realizaremos una revisión de Sinergym, la principal herramienta utilizada a lo largo de este proyecto. Abordaremos además la aportación realizada a dicha herramienta.

4.1. Introducción a la herramienta

Sinergym es una plataforma basada en la interfaz de OpenAI Gym que sirve como un marco de pruebas estándar para ejecutar algoritmos de RL/DRL en diversos edificios y climas, facilitando así su evaluación y comparación. Este proyecto se origina a partir del entorno Gym-Eplus de Zhang y Lam [51, 49], cuyo *back-end* se tomó como referencia para crear una nueva versión que es actualizada, escalable y fácilmente reutilizable.

4.2. Características

Revisando la web de Sinergym, esta herramienta ofrece el siguiente conjunto de características:

- **Compatibilidad con Motores de Simulación**: Sinergym utiliza la API de Python de EnergyPlus para la comunicación con este simulador. En el futuro, se prevé la incorporación de otros motores como OpenModelica.
- **Entornos y Componentes Personalizables para Benchmarking**: Sinergym ofrece entornos específicos para evaluar y probar algoritmos de RL u otras estrategias, de manera similar a los entornos de Atari o Mujoco. Además, permite a los usuarios personalizar las configuraciones experimentales, crear sus propios entornos o ajustar los ya

existentes. También facilita la creación de nuevos componentes personalizables, como funciones de recompensas, *wrappers* y controladores, lo que hace que la plataforma sea escalable y flexible.

- **Automatización de Modelos de Edificio y Actuadores**: Sinergym automatiza tanto la actualización del modelo de edificio en función de los cambios realizados por el usuario en la configuración del entorno, como el control de actuadores a través de la interfaz de Gymnasium. El usuario solo necesita proporcionar los nombres de los actuadores, y Sinergym gestiona el resto del proceso.

- **Información Detallada del Entorno**: Proporciona datos completos sobre los componentes internos de Sinergym mediante la interfaz del entorno.

- **Integración con Stable Baselines 3, Google Cloud y Weights & Biases**: Sinergym ofrece funcionalidades personalizadas para probar entornos con algoritmos de SB3, incluyendo *callbacks* y un registro en tiempo real del entrenamiento, aunque es compatible con cualquier algoritmo de DRL. Además, proporciona directrices para su uso en la infraestructura de Google Cloud, permitiendo ejecutar experimentos en la nube. Asimismo, cuenta con compatibilidad con Weights & Biases, lo que simplifica el entrenamiento, la reproducibilidad y la comparación de agentes, facilitando la gestión y el monitoreo eficiente del ciclo de vida del modelo.

- **Documentación Extensa, Pruebas Unitarias y Flujos de Trabajo de GitHub Actions**: Asegura que Sinergym sea un ecosistema eficiente para la comprensión y el desarrollo.

4.3. Funcionamiento

Una vez que hemos definido las principales características de Sinergym, pasemos a analizar su funcionamiento. Tal como se ilustra en la Figura 4.1, Sinergym consta de los siguientes elementos principales:

- **Agente**: Interactúa con el entorno enviando acciones y recibiendo observaciones a través de la interfaz de Gymnasium.

- **Interfaz de comunicación**: La interfaz de Gymnasium se comunica con el motor de simulación mediante la API de Python de EnergyPlus.

- **Simulación**: La API de Python de EnergyPlus ofrece la funcionalidad para gestionar controladores como actuadores, medidores y variables,

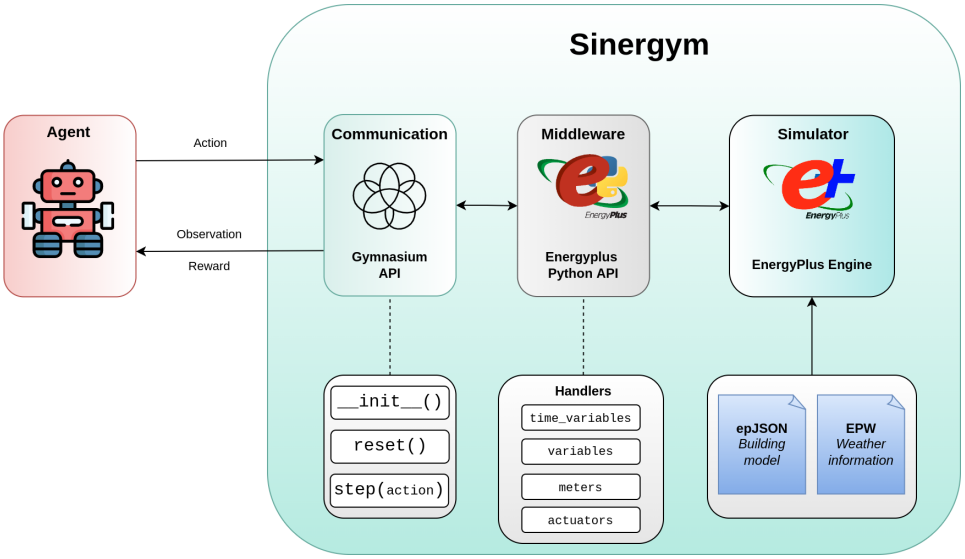


Figura 4.1: Elementos que componen el ecosistema de Sinergym. Imagen obtenida de la web de Sinergym.

cuyos valores actuales influyen directamente en el progreso de la simulación.

- **Ficheros de configuración** que se utilizarán en la simulación:
  - Un fichero `.epJSON` que contiene el modelo del edificio utilizado en la simulación. Este se complementa con un fichero `.json` que proporciona información sobre las acciones, variables de entrada y salida a emplear, sus rangos, etc.
  - Un fichero `.epw` (*EnergyPlus Weather*, EPW) con la información meteorológica empleada en la simulación.

Para un mayor entendimiento de la herramienta la Figura 4.2 proporciona una vista más detallada de el flujo que esta sigue durante su funcionamiento.

4.3.1. *Wrappers*

Antes de explicar la contribución proporcionada a la herramienta, es importante entender el concepto de *wrapper* en el contexto de Sinergym.

En Sinergym, los wrappers son componentes que añaden funcionalidades adicionales al entorno, permitiendo personalizar y extender su comportamiento. A continuación, se describen algunos de los wrappers más utilizados en este proyecto:

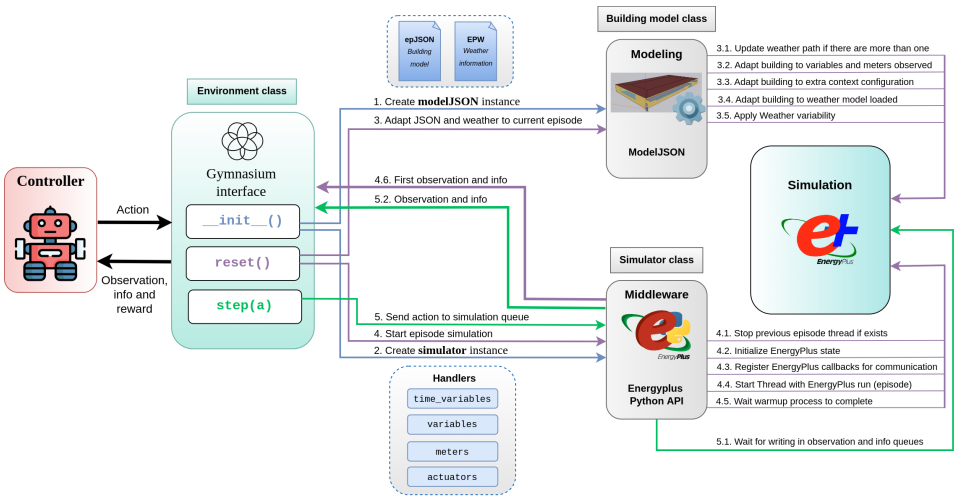


Figura 4.2: Funcionamiento general de Sinergym. Imagen obtenida de la web de Sinergym.

- ***NormalizeAction***: Este wrapper normaliza el espacio de acción, lo que es especialmente útil para algoritmos de DRL que requieren valores normalizados para funcionar correctamente.

- ***NormalizeObservation***: Transforma las observaciones recibidas del simulador a valores comprendidos entre $[-1, 1]$. Este wrapper incluye características adicionales acceso a los valores de media y varianza usados para la calibración, y la posibilidad de desactivar la actualización automática durante la evaluación.

- ***LoggerWrapper***: Este wrapper se encarga de registrar todas las interacciones entre el agente y el entorno. Permite seleccionar una clase de registro diferente en el constructor si se necesita otro tipo de logging.

- ***MultiObsWrapper***: Este wrapper acumula las observaciones en una cola de historial, cuyo tamaño es personalizable.

Estos wrappers permiten una mayor flexibilidad y control sobre el entorno de simulación, facilitando la implementación y evaluación de diversas estrategias de aprendizaje automático.

4.4. Contribucion a la herramienta

La contribución realizada a la herramienta consiste en la implementación de un wrapper llamado *WeatherForecastingWrapper* que amplía las obser-

vaciones del entorno original de Sinergym, añadiendo datos meteorológicos futuros [1].

Para la implementación del wrapper fueron establecidos los siguientes requisitos:

Requisito 1

El wrapper debe permitir la selección de las variables de las previsiones temporales que se deseen agregar a las observaciones.

Requisito 2

El wrapper debe permitir la configuración del horizonte temporal de las previsiones que se añadirán a las observaciones.

Requisito 3

El wrapper debe permitir ajustar el intervalo entre cada previsión temporal que se incluirá en las observaciones.

Requisito 4

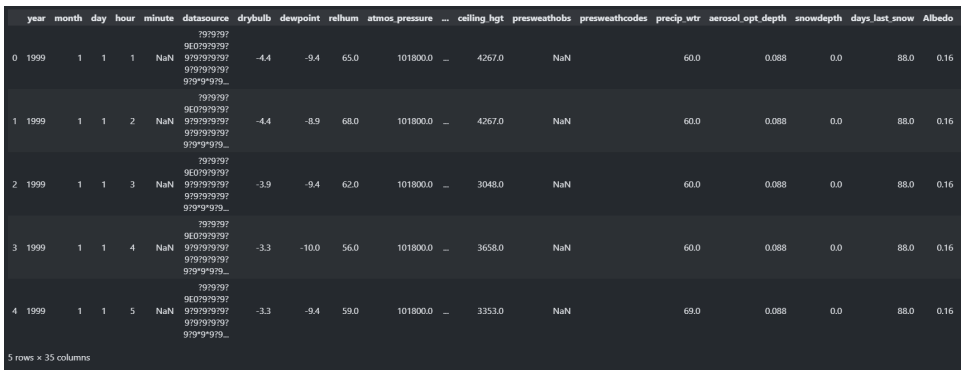
El wrapper debe contar con un mecanismo configurable para agregar ruido a las observaciones.

Para cumplir con el requisito 1, las previsiones temporales se obtienen del archivo `.epw`, que contiene la información meteorológica utilizada en la simulación. Específicamente, dicho archivo proporciona datos meteorológicos por hora durante un año para una ubicación específica (ver Figura 4.3). Usando los campos de mes, día y hora, se busca la información meteorológica correspondiente a las previsiones temporales que se necesitan en cada momento. Finalmente, para seleccionar las variables de las previsiones temporales, el wrapper incluye un parámetro llamado `columns`, que es un array donde cada elemento representa una de las variables que se desea utilizar del archivo meteorológico.

Respecto al requisito 2, el wrapper incluye un parámetro llamado `n`, que define el número de previsiones temporales que se van a agregar a la observación actual.

En cuanto al requisito 3, el wrapper cuenta con un parámetro llamado `delta` (δ), que define el intervalo temporal entre cada previsión que se añade a las observaciones.

[1] La implementación de dicho wrapper se puede revisar en el repositorio de GitHub del proyecto: https://github.com/fpertinezp/TFM-DRL-HVAC



	year	month	day	hour	minute	datasource	drybulb	dewpoint	relhum	atmos_pressure	...	ceiling_hgt	presweathobs	presweathcodes	precip_wtr	aerosol_opt_depth	snowdepth	days_last_snow	Albedo
0	1999	1	1	1	NaN	?9?9?9? 9E0?9?9?9? 9?9?9?9?9? 9?9?9?9?9? 9?9*9*979...	-4.4	-9.4	65.0	101800.0	...	4267.0	NaN		60.0	0.088	0.0	88.0	0.16
1	1999	1	1	2	NaN	?9?9?9? 9E0?9?9?9? 9?9?9?9?9? 9?9?9?9?9? 9?9*9*979...	-4.4	-8.9	68.0	101800.0	...	4267.0	NaN		60.0	0.088	0.0	88.0	0.16
2	1999	1	1	3	NaN	?9?9?9? 9E0?9?9?9? 9?9?9?9?9? 9?9?9?9?9? 9?9*9*979...	-3.9	-9.4	62.0	101800.0	...	3048.0	NaN		60.0	0.088	0.0	88.0	0.16
3	1999	1	1	4	NaN	?9?9?9? 9E0?9?9?9? 9?9?9?9?9? 9?9?9?9?9? 9?9*9*979...	-3.3	-10.0	56.0	101800.0	...	3658.0	NaN		60.0	0.088	0.0	88.0	0.16
4	1999	1	1	5	NaN	?9?9?9? 9E0?9?9?9? 9?9?9?9?9? 9?9?9?9?9? 9?9*9*979...	-3.3	-9.4	59.0	101800.0	...	3353.0	NaN		69.0	0.088	0.0	88.0	0.16

5 rows × 35 columns

Figura 4.3: Ejemplo de archivo `.epw` con información meteorológica.

Por último, para cumplir con el requisito 4, el wrapper dispone de un parámetro denominado `weather_variability`, que es una tupla de tamaño 3 donde se establecen los parámetros σ, μ y τ para introducir ruido en los datos meteorológicos mediante un proceso de Ornstein-Uhlenbeck (ver Figura 4.4).

El proceso de Ornstein-Uhlenbeck es un modelo estocástico que describe la evolución de un proceso de ruido con una tendencia a revertirse hacia un valor promedio a lo largo del tiempo. Se caracteriza por su capacidad para generar fluctuaciones que tienden a regresar a una media, lo que lo hace útil para simular ruido en datos temporales de manera realista. En la forma discreta (la ecuación que se utiliza en la implementación) es la siguiente:

$$x_{t+1} = x_t + \frac{dt}{\tau}(\mu - x_t) + \sigma\sqrt{dt} \cdot \mathcal{N}(0,1)$$

Donde:

- μ es el valor medio hacia el que tiende la variable meteorológica. Este es el valor que el proceso intenta alcanzar a lo largo del tiempo, siendo el punto de equilibrio del sistema.
- τ es la **constante de tiempo**, que controla la rapidez con la que la variable vuelve a su media. Cuanto más pequeño es τ, más rápida es la reversión hacia μ.
- σ es la **desviación estándar** del ruido, que define la magnitud de las fluctuaciones aleatorias. Un valor mayor de σ introduce más variabilidad en el sistema.
- $\mathcal{N}(0,1)$ representa una variable aleatoria normalmente distribuida con media 0 y desviación estándar 1, que introduce ruido gaussiano en el sistema.

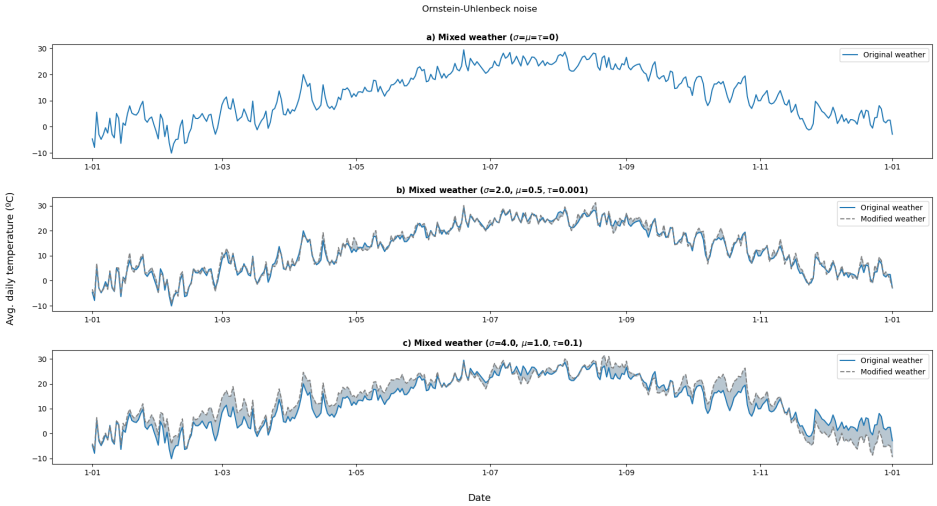


Figura 4.4: Ejemplo de proceso de ruido Ornstein-Uhlenbeck aplicado a datos meteorológicos. Imagen obtenida de la web de Sinergym.

En la figura 4.5 se muestra todo el proceso seguido para agregar las previsiones temporales a la observación.

La implementación de este *wrapper* se justifica en el valor de la información que aporta a la observación actual del entorno. Como se muestra en la Figura 4.6, el *WeatherForecastingWrapper* incorpora predicciones de los próximos pasos temporales en cada observación, asegurando que cada nueva observación contenga información futura que no estaba disponible en observaciones anteriores. De este modo, se busca ofrecer información más valiosa y novedosa en comparación con el *MultiObsWrapper*, con el cual se comparará el rendimiento mostrado por ambos.

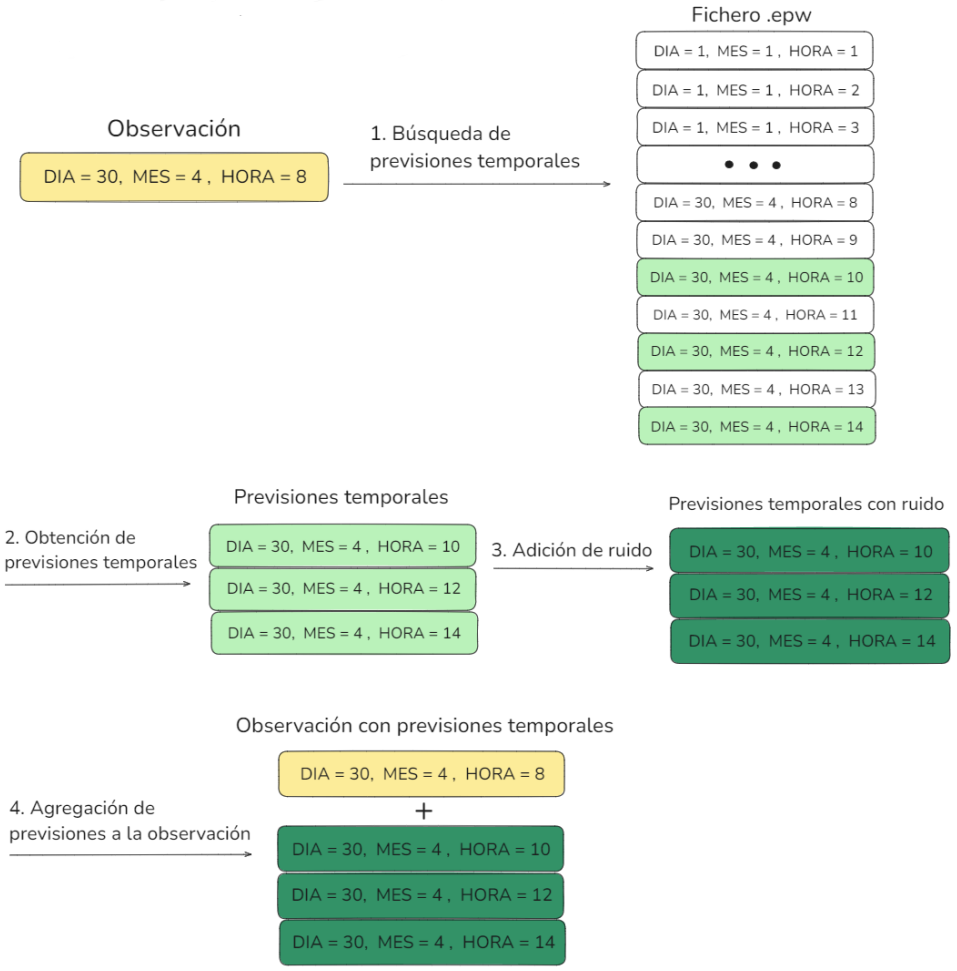


Figura 4.5: Proceso para agregar previsiones temporales a las observaciones (ejemplo con configuración $n = 3$ y $\delta = 2$). Para cada observación del entorno, el *WeatherForecastingWrapper* identifica las previsiones temporales a añadir, les aplica ruido, y finalmente las integra a la observación original.

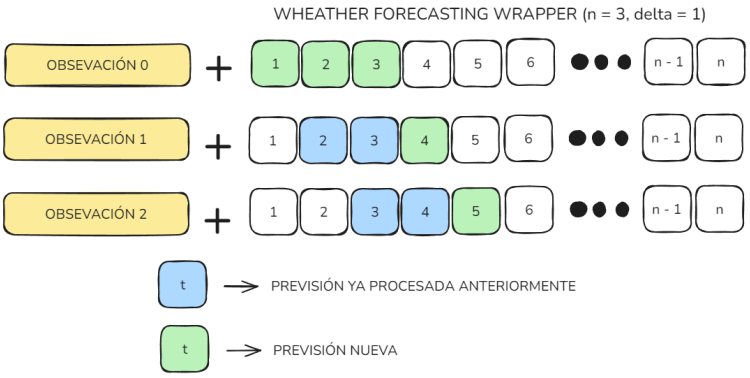


Figura 4.6: Ejemplo de configuración $n = 3$, $\delta = 1$. En cada observación, el *WeatherForecastingWrapper* añade predicciones para los próximos 3 pasos temporales, garantizando que cada observación incluya al menos una predicción nueva que no estaba presente en observaciones anteriores.

Capítulo 5

Experimentación, resultados y discusión

En este capítulo se presenta la experimentación llevada a cabo con Sinergym en diversos entornos. Se detallarán los aspectos técnicos de la experimentación, se expondrán los resultados obtenidos, y finalmente realizará un análisis de los mismos.

5.1. Configuración

En esta sección se describen los aspectos técnicos relacionados con la experimentación, abarcando tanto el entorno de simulación como la configuración utilizada para el entrenamiento, validación y evaluación de los agentes.

5.1.1. Entorno de simulación

La experimentación se llevó a cabo en varios entornos de simulación, que incluyen dos modelos de edificios, un tipo de clima, y un conjunto de variables de entrada y salida proporcionadas por el simulador.

Modelos de edificios

Para los experimentos se seleccionaron los dos siguientes modelos de edificios incluidos en Sinergym como escenarios de prueba:

- `5ZoneAutoDXVAV` [41, 8]. Este modelo representa un edificio de oficinas rectangular de una sola planta (ver Figura 5.1). El edificio está

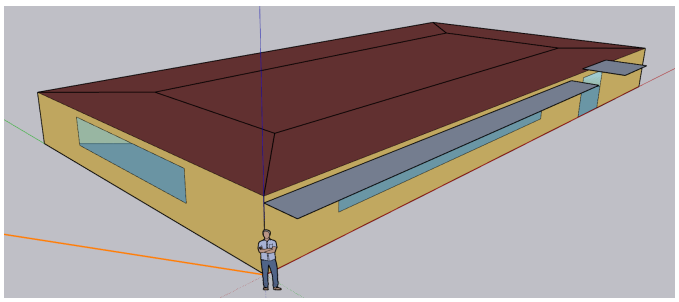


Figura 5.1: Edificio `5ZoneAutoDXVAV`

dividido en cinco zonas: cuatro exteriores y una interior. El sistema de climatización incluye un sistema de Volumen de Aire Variable (VAV) empaquetado, con una bobina de enfriamiento de Expansión Directa (DX) y bobinas de calefacción a gas, con entradas dimensionadas automáticamente para el sistema HVAC a controlar.

Este modelo, que forma parte del repositorio de modelos de ejemplo de EnergyPlus[1], ha sido utilizado previamente en investigaciones relacionadas [49].

- `radiant_residential_building_free_heating`: Este modelo está basado en un edificio residencial real de dos plantas, con cinco zonas climatizadas: sala de estar y cocina en la planta baja, y tres dormitorios en la planta superior. El sistema de calefacción es radiante de baja temperatura, gestionado por el flujo de agua caliente en superficies radiantes en cada zona. Este modelo, recientemente añadido a Sinergym, no cuenta con estudios previos de experimentación sobre este, lo que aporta un valor añadido a este trabajo.

Climas

Para llevar a cabo la simulación energética de un edificio, es esencial considerar las condiciones climáticas que lo impactan. Estas condiciones se recopilan en un conjunto de datos que reflejan un período y ubicación específicos, y luego se emplean para reproducir ese entorno.

El Departamento de Energía de los Estados Unidos (DOE) ofrece una clasificación que distingue 19 tipos de climas[2], que abarcan desde climas extremadamente cálidos y húmedos hasta climas árticos. Actualmente, existen numerosos repositorios que proporcionan modelos climáticos abiertos para

[1] El listado completo puede consultarse en: https://github.com/NREL/EnergyPlus/tree/v8.6.0/testfiles.

[2] Se puede consultar esta clasificación en: https://www.energycodes.gov/development/commercial/prototype_models

la simulación energética de edificios, como *Climate.OneBuilding*. En este estudio, se utilizó el modelo **Clima templado** (`mixed humid, 4A`), correspondiente a las condiciones registradas en el aeropuerto *John F. Kennedy International Airport* de Nueva York entre los años 1973 y 2005.

Es importante señalar que se introdujo ruido en estos conjuntos de datos, lo que mejora el entrenamiento al permitir que los agentes enfrenten diferentes variaciones del mismo clima en cada episodio. Por lo tanto, los archivos climáticos empleados en las simulaciones fueron versiones modificadas de los originales, con un componente de estocasticidad adicional.

Variables de entorno

En el contexto de simulaciones, se distinguen dos tipos principales de variables: las de entrada (*input variables*) y las de salida (*output variables*). Las variables de entrada son aquellas que el agente tiene la capacidad de ajustar, afectando directamente al entorno. Por otro lado, las variables de salida sirven para mostrar cómo se encuentra el entorno en un momento dado, proporcionando datos sobre su estado actual.

- `5ZoneAutoDXVAV`: Este edificio posee un total de 16 variables que componen su observación del entorno[3]. Estas variables son:
  - **Temperatura seca del aire exterior** (*Site Outdoor Air Drybulb Temperature*).
  - **Humedad relativa del aire exterior** (*Site Outdoor Air Relative Humidity*).
  - **Velocidad del viento** (*Site Wind Speed*).
  - **Dirección del viento** (*Site Wind Direction*).
  - **Radiación solar difusa por área** (*Site Diffuse Solar Radiation Rate per Area*).
  - **Radiación solar directa por área** (*Site Direct Solar Radiation Rate per Area*).
  - **Temperatura de consigna actual para calefacción** (*Zone Thermostat Heating Setpoint Temperature*).
  - **Temperatura de consigna actual para refrigeración** (*Zone Thermostat Cooling Setpoint Temperature*).
  - **Temperatura del aire** (*Zone Air Temperature*).
  - **Humedad relativa del aire** (*Zone Air Relative Humidity*).

[3]Para más detalles, consulte: https://ugr-sail.github.io/sinergym/compilation/main/pages/buildings.html#zone

- **Número de ocupantes** (*Zone People Occupant Count*).
- **Emisiones de CO_2** (*Environmental Impact Total CO2 Emissions Carbon Equivalent Mass*).
- **Demanda energética de dispositivos HVAC** (*Facility Total HVAC Electric Demand Power*).
- **Día** (*Current Day*).
- **Mes** (*Current Month*).
- **Hora** (*Current Hour*).

Las variables de entrada que el agente puede modificar son:

- **Temperatura de consigna para calefacción** (*Heating Setpoint*).
- **Temperatura de consigna para refrigeración** (*Cooling Setpoint*).

- `radiant_residential_building_free_heating`: Para este entorno, además de las variables proporcionadas por el entorno `5ZoneAutoDXVAV` (excepto las temperaturas de consigna para calefacción y refrigeración, emisiones de CO_2 y demanda energética de dispositivos HVAC), el agente recibe las siguientes variables adicionales:

  - **Temperatura de salida del HVAC radiante** (*Zone Radiant HVAC Outlet Temperature*).
  - **Temperatura de entrada del HVAC radiante** (*Zone Radiant HVAC Inlet Temperature*).
  - **Temperatura interna especificada por el usuario en la superficie** (*Surface Internal User Specified Location Temperature*).
  - **Caudal másico del nodo del sistema** (*System Node Mass Flow Rate*).
  - **Energía de transferencia de calor del componente de la fuente de temperatura de la planta** (*Plant Temperature Source Component Heat Transfer Energy*).
  - **Temperatura de salida del componente de la fuente de temperatura de la planta** (*Plant Temperature Source Component Outlet Temperature*).
  - **Temperatura de entrada del componente de la fuente de temperatura de la planta** (*Plant Temperature Source Component Inlet Temperature*).
  - **Tasa de transferencia de calor del componente de la fuente de temperatura de la planta** (*Plant Temperature Source Component Heat Transfer Rate*).

- **Caudal másico del componente de la fuente de temperatura de la planta** (*Plant Temperature Source Component Mass Flow Rate*).

En este caso, la variable de entrada que el agente puede modificar es:

- **Temperatura de consigna para calefacción en la zona** (*Zone Thermostat Heating Setpoint Temperature*).

Métricas de evaluación

Para medir el rendimiento de los agentes, nos enfocaremos en una métrica principal derivada de las variables de salida previamente mencionadas: la **recompensa media**. Esta métrica evalúa la eficacia del agente en mantener un equilibrio adecuado entre el confort de los ocupantes y la minimización del consumo energético.

Como se detalló en la sección 3.2, la función de recompensa que utilizan los agentes está definida por la siguiente fórmula:

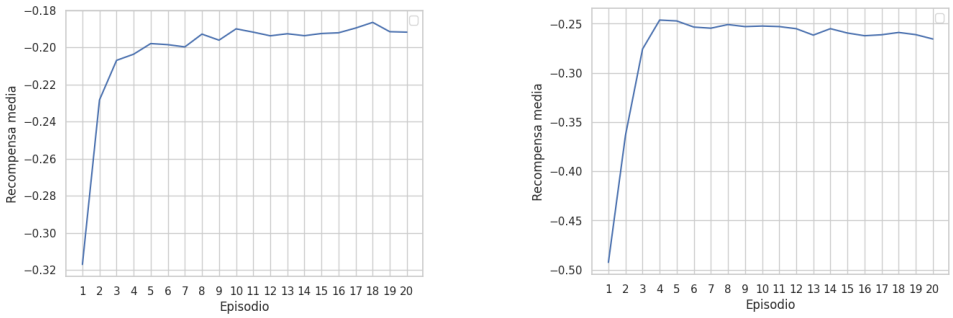
$$r(S_t, A_t) = -(1 - w) \cdot \lambda_P \cdot \text{Consumo} - w \cdot \lambda_C \cdot \text{Confort} \tag{5.1}$$

Dado que la función de recompensa está formulada en términos negativos, el objetivo es maximizar el valor de la recompensa, acercándolo lo más posible a 0. La recompensa media se obtiene calculando el promedio de las recompensas recibidas por los agentes a lo largo de cada episodio.

5.1.2. Entrenamiento, validación y evaluación

Para configurar las fases de entrenamiento, validación y evaluación, se establecieron una serie de parámetros. El entrenamiento de los agentes se llevó a cabo en 20 episodios (se comprobó experimentalmente que este número de episodios era suficiente para que todos los algoritmos alcanzaran la convergencia, como se muestra en la Figura 5.2.), con una sesión de validación programada cada dos episodios de entrenamiento. Tanto el entrenamiento como la evaluación se realizaron en el mismo entorno, configurado con un intervalo de 15 minutos por *timestep*, lo que permitía al agente ejecutar una acción en cada intervalo. Posteriormente, se llevaron a cabo 10 episodios de evaluación para evaluar el desempeño de los agentes entrenados.

Para asegurar la reproducibilidad y consistencia de los experimentos, se utilizó una semilla fija (42) en todas las fases: entrenamiento, validación y evaluación.



PPO DQN

Figura 5.2: Ejemplo de la convergencia de agentes PPO y DQN durante el entrenamiento.

En cuanto a los wrappers empleados en cada entorno, se aplicaron los siguientes:

- En la versión discreta del entorno 5 zonas: `NormalizeObservation`, `MultiObsWrapper` y `LoggerWrapper`.
- En la versión continua del entorno 5 zonas: `NormalizeActions`, `NormalizeObservation`, `MultiObsWrapper`, `WeatherForecastingWrapper` y `LoggerWrapper`.
- En el entorno continuo Radiant: `NormalizeActions`, `NormalizeObservation`, `MultiObsWrapper`, `WeatherForecastingWrapper`, `LoggerWrapper`, `HeatPumpEnergyWrapper` y `ExtremeFlowControlWrapper`.

Cada wrapper cumple con funciones específicas:

- ***NormalizeActions* y *NormalizeObservation***: Normalizan las acciones y observaciones, mejorando el rendimiento de los algoritmos según la evidencia empírica.
- ***MultiObsWrapper* y *WeatherForecastingWrapper***: Expanden el espacio de estados con información contextual, siendo la piedra fundamental para la experimentación realizada.
- ***LoggerWrapper***: Registra información durante el entrenamiento, validación y prueba, permitiendo la obtención de métricas como la recompensa media, la recompensa acumulada y las violaciones del confort.
- ***HeatPumpEnergyWrapper* y *ExtremeFlowControlWrapper***: Esenciales para el funcionamiento efectivo del entorno Radiant.

Es importante destacar que los wrappers se aplicaron de manera consistente en todas las fases del proceso, garantizando uniformidad en las condiciones experimentales.

Respecto a los algoritmos empleados, se eligió DQN para los entornos discretos y PPO para los entornos continuos. Para asegurar que los agentes operaran bajo condiciones uniformes, se optó por no realizar un ajuste de hiperparámetros. En su lugar, se utilizaron los valores predeterminados, enfocando la experimentación en la configuración del espacio de estados.

Toda esta información, junto con detalles adicionales, se encuentra en el Apéndice A.

5.2. Organización de los experimentos

La organización de los experimentos y la aproximación metodológica en cada caso han sido las siguientes:

- **Experimento con *MultiObsWrapper*:** Este experimento investigará cómo la inclusión de observaciones anteriores en la observación actual afecta al rendimiento del modelo. Se variará el número de observaciones previas, denotado por n, para determinar cuál es la configuración óptima para el entorno específico. Los valores de n se evaluarán con un intervalo temporal constante de 15 minutos entre observaciones (correspondientes al *timestep* configurado en la simulación), como se especifica en la Tabla 5.1.
- **Experimento con *WeatherForecastingWrapper*:** En este caso, se analizará el impacto de incorporar predicciones meteorológicas en la observación actual. Se variará tanto el número de predicciones (n) como el intervalo temporal entre ellas (*delta*), para encontrar la configuración más eficaz para cada entorno. Las combinaciones de n y *delta* se presentan en la Tabla 5.2.
- **Experimento de combinación de wrappers:** Se evaluará si la combinación de múltiples wrappers que amplían el espacio de estado proporciona mejoras respecto al uso de cada wrapper por separado. Para esto, se combinarán las mejores configuraciones encontradas en los experimentos individuales con *MultiObsWrapper* y *WeatherForecastingWrapper* para cada entorno.

Este enfoque permitirá una evaluación exhaustiva de cómo la variación en los parámetros afecta el desempeño del modelo, ofreciendo una comprensión detallada de la influencia de las observaciones históricas y las predicciones meteorológicas en la solución de problemas en diversos entornos.

n	Intervalo entre observación	Horizonte temporal (Pasado)
4	15 minutos	1 hora
10		2 horas, 30 minutos
20		5 horas

Tabla 5.1: Intervalo entre observación y horizonte temporal hacia el pasado para diferentes tamaños de historial n en el *MultiObsWrapper*.

n	Intervalo entre previsión (δ)	Horizonte temporal (Futuro)
3	1 hora	3 horas
	2 horas	6 horas
	3 horas	9 horas
5	1 hora	5 horas
	2 horas	10 horas
	3 horas	15 horas
7	1 hora	7 horas
	2 horas	14 horas
	3 horas	21 horas

Tabla 5.2: Intervalo entre predicción y horizonte temporal hacia el futuro para diferentes tamaños de historial n en el *WeatherForecastingWrapper*.

5.3. Resultados

En esta sección, se detallan los resultados de los diferentes experimentos realizados. Para una visión completa de todos los resultados, consulte el Apéndice B.

5.3.1. MultiObsWrapper

A continuación, se detallan los experimentos realizados utilizando el wrapper *MultiObsWrapper*. Los entornos evaluados en estos experimentos son: el entorno 5 zonas (correspondiente al edificio `5ZoneAutoDXVAV`), en sus versiones discreta y continua, y el entorno Radiant (correspondiente al edificio `radiant_residential_building_free_heating`).

Entorno 5 Zonas (Versión Discreta)

Los resultados presentados en la Figura 5.3 y en la Tabla B.1 ilustran el impacto del wrapper *MultiObsWrapper* en la recompensa media de un agente de aprendizaje por refuerzo en el entorno discreto del edificio de 5 zonas.

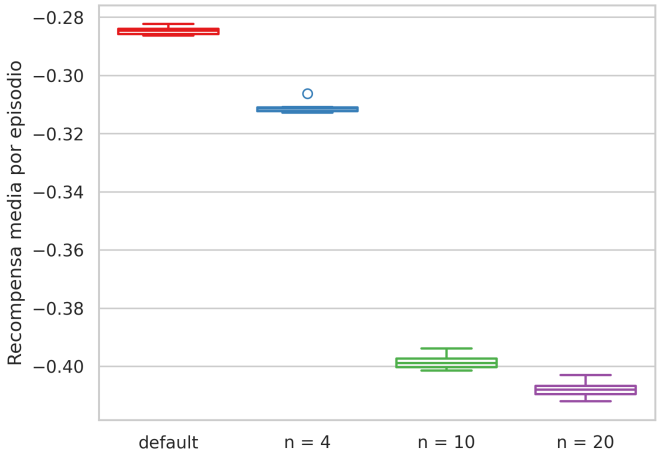


Figura 5.3: Impacto del wrapper *MultiObsWrapper* en el entorno 5 zonas (versión discreta).

Al iniciar sin historial de observaciones ($n = 0$), el agente obtiene una recompensa media de -0.285. Con la inclusión de un historial de 4 observaciones, la recompensa media empeora a -0.311, lo que sugiere que un historial limitado no sólo podría no proporcionar suficiente contexto para mejorar la toma de decisiones del agente, sino incluso empeorarla.

Para determinar si aumentar el historial podría revertir esta tendencia, se probaron configuraciones con un mayor número de observaciones. No obstante, al incrementar el historial a 10 observaciones, la recompensa media empeora aún más (a -0.399), indicando que el agente no solo no está aprovechando la información adicional para mejorar su desempeño, sino que su rendimiento empeora.

Finalmente, con un historial de 20 observaciones, la recompensa media alcanza el valor más bajo observado, -0.408. Este patrón refuerza la idea de que ampliar el historial de observaciones no beneficia al agente, sino que lo perjudica.

Entorno 5 Zonas (Versión Continua)

Los resultados presentados en la Figura 5.4 y en la Tabla B.2 ilustran el efecto del wrapper *MultiObsWrapper* en la recompensa media obtenida por un agente de aprendizaje por refuerzo en la versión continua del entorno de 5 zonas.

Sin historial de observaciones ($n = 0$), el agente obtiene una recompensa media de -0.442. Al incorporar un historial de 4 observaciones, la recompensa media disminuye ligeramente a -0.450, sugiriendo que una pequeña

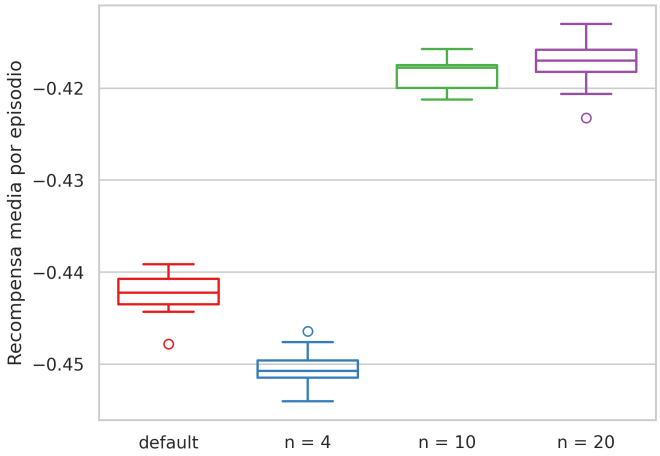


Figura 5.4: Impacto del wrapper *MultiObsWrapper* en el entorno 5 zonas (versión contínua).

ampliación del historial no tiene un impacto considerable en la mejora del desempeño del agente en comparación con la configuración inicial.

Cuando el historial se incrementa a 10 observaciones, la recompensa media mejora a -0.418, lo que indica que un historial de 10 observaciones proporciona un contexto adicional que beneficia al agente.

Finalmente, al aumentar el historial a 20 observaciones, la recompensa media se mantiene casi constante en -0.417, un valor muy cercano al obtenido con 10 observaciones. Este estancamiento sugiere que un historial de 20 observaciones no ofrece beneficios adicionales significativos respecto a un historial de 10 observaciones, indicando que la utilidad de la información adicional podría estar alcanzando su límite en este entorno.

Entorno Radiant

La Figura 5.5 y la Tabla B.3 ilustran el efecto del *MultiObsWrapper* en la recompensa media obtenida por un agente de aprendizaje por refuerzo en la versión continua del entorno Radiant.

En ausencia de historial de observaciones ($n = 0$), el agente obtiene una recompensa media de -4.568. Al añadir un historial de 4 observaciones, la recompensa media empeora considerablemente a -7.477, junto con un ligero incremento en la desviación estándar a 0.377. Esta reducción sugiere que un historial limitado no proporciona suficiente contexto para mejorar la toma de decisiones del agente y, en cambio, puede dificultarla.

El impacto negativo del historial se vuelve más pronunciado a medida que aumenta el número de observaciones. Con un historial de 10 observacio-

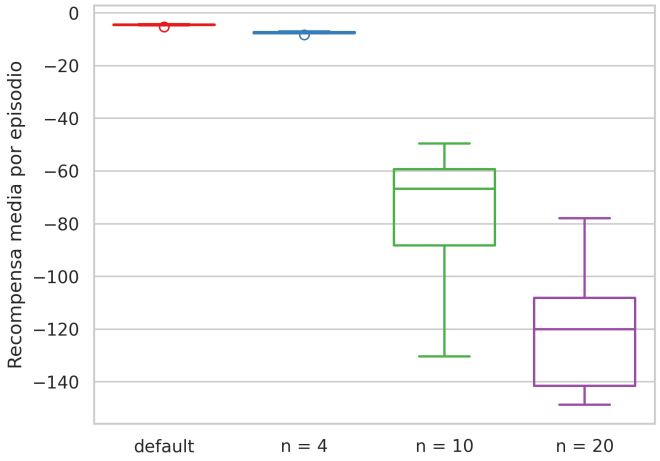


Figura 5.5: Impacto del *MultiObsWrapper* en el entorno Radiant.

nes, la recompensa media cae drásticamente a -76.046, con una desviación estándar significativamente alta de 26.114. Esto indica una fuerte inestabilidad en el comportamiento del agente, lo que sugiere que el agente está siendo abrumado por el exceso de información histórica, resultando en decisiones menos efectivas.

Al extender el historial a 20 observaciones, la recompensa media desciende aún más a -120.930, con una desviación estándar de 22.725. Este deterioro continuo refuerza la conclusión de que un historial extenso en este entorno continuo es perjudicial, ya que el volumen de información adicional parece sobrepasar la capacidad del agente para procesarla de manera efectiva.

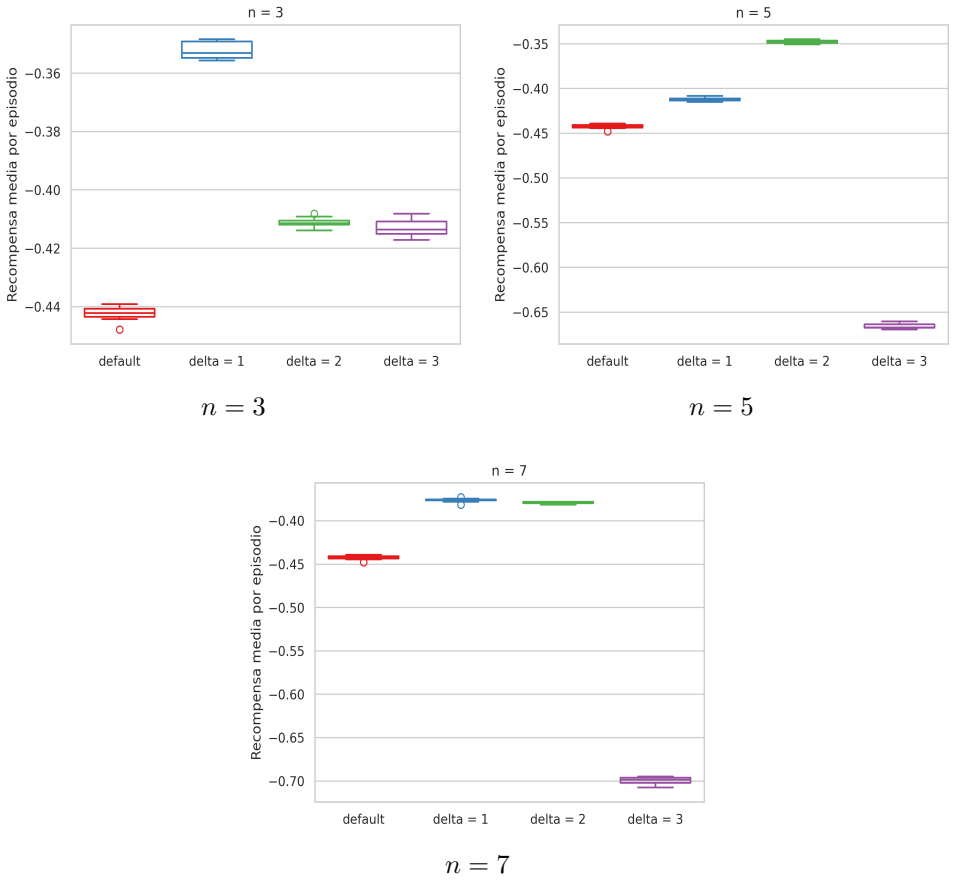
5.3.2. *WeatherForecastingWrapper*

A continuación, se describen los experimentos realizados utilizando el wrapper *WeatherForecastingWrapper*. Los entornos evaluados en estos experimentos son el entorno de 5 zonas (en su versión continua) y el entorno Radiant.

Entorno 5 Zonas (Versión Continua)

Los resultados, que se presentan en la Figura 5.6 y se detallan en la Tabla B.4, ilustran el impacto del *WeatherForecastingWrapper* sobre la recompensa media del agente en la versión continua del entorno de 5 zonas.

Sin historial ($n = 0$), la recompensa media es de -0.442. Al utilizar un historial de $n = 3$, la recompensa media mejora notablemente a -0.352 con



$n = 3$ $n = 5$

$n = 7$

Figura 5.6: Impacto del *WeatherForecastingWrapper* en el entorno 5 zonas (versión continua) para diferentes configuraciones de n y δ.

un intervalo entre previsiones (δ) de 1. Sin embargo, al incrementar δ a 2 y 3, la recompensa media disminuye a -0.411 y -0.413, respectivamente. Esto sugiere que, aunque un historial de $n = 3$ es beneficioso al principio, su efectividad disminuye con intervalos entre previsión más largos.

Con un historial de $n = 5$, se observa una mejora en la recompensa media con $\delta = 2$, alcanzando -0.348. Comparado con el resultado para $n = 3$ y $\delta = 1$, este horizonte temporal de $5 \times 2 = 10$ horas no ofrece beneficios adicionales significativos en comparación con un historial de $3 \times 1 = 3$ horas, sugiriendo que la utilidad de la información adicional puede estar alcanzando su límite en este entorno continuo. Al aumentar δ a 3, la recompensa media cae a -0.666, reforzando esta observación.

Para historiales más extensos ($n = 7$), las recompensas medias con $\delta = 1$ y $\delta = 2$ son de -0.376 y -0.379, respectivamente. Aunque estas recompensas superan el valor por defecto, son inferiores a la obtenida con $n = 3$ y $\delta = 1$. Finalmente, con $\delta = 3$, la recompensa media se desploma a -0.7, reforzando la conclusión de que un horizonte temporal mayor a 3 horas no resulta mucho más beneficioso para el agente, incluso pudiendo ser perjudicial.

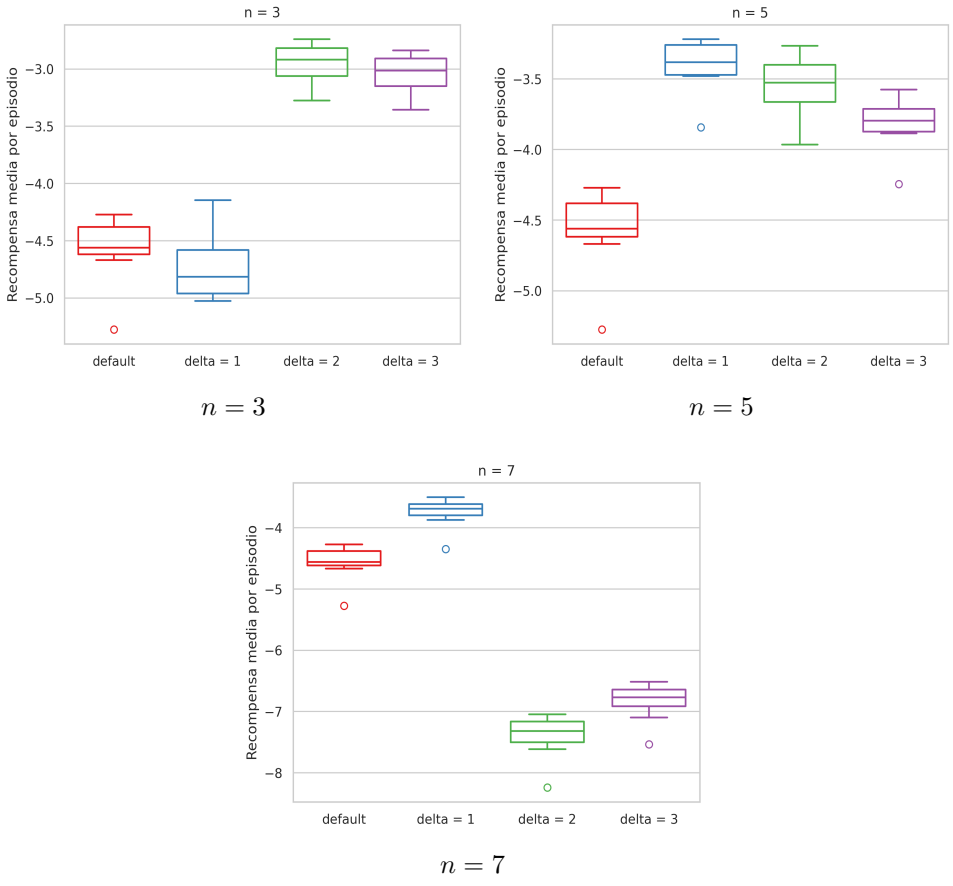
Entorno Radiant (Versión Continua)

Los resultados presentados en la Figura 5.7 y la Tabla B.5 ilustran el efecto del *WeatherForecastingWrapper* sobre la recompensa media del agente en la versión continua del entorno Radiant.

Sin utilizar historial ($n = 0$), la recompensa media es de -4.568. Con un historial de $n = 3$, la recompensa media con un intervalo entre previsión de $\delta = 1$ es de -4.717, ligeramente inferior al valor por defecto. Sin embargo, al aumentar δ a 2, la recompensa media mejora notablemente, alcanzando -2.945. Al aumentar δ a 3, la recompensa media se estabiliza en -3.038, indicando que no se obtienen beneficios adicionales al incrementar el intervalo entre previsión más allá de dos.

Para $n = 5$, la recompensa media con $\delta = 1$ es de -3.398, indicando que un historial de $n = 5$ con un intervalo entre previsión corto puede ser efectivo. No obstante, al aumentar δ a 2, la recompensa media disminuye ligeramente a -3.556, sugiriendo que la utilidad de la información adicional puede estar alcanzando su límite. Al incrementar δ a 3, la recompensa baja aún más a -3.813, confirmando que mayores intervalos de previsión pueden no ser beneficiosos.

Para $n = 7$, la recompensa media con $\delta = 1$ es de -3.748, lo que indica que un historial de $n = 7$ no produce mejoras en el rendimiento del agente en comparación con configuraciones anteriores. Al aumentar δ a 2, la recompensa media disminuye drásticamente a -7.396, sugiriendo que el



$n = 3$ $n = 5$

$n = 7$

Figura 5.7: Impacto del *WeatherForecastingWrapper* en el entorno Radiant para diferentes valores de n y δ.

exceso de información puede causar inestabilidad en el agente. Aunque al incrementar δ a 3 se observa una leve mejora, alcanzando -6.837, este valor sigue siendo inferior al de otras configuraciones. Estos resultados refuerzan la conclusión de que un historial de $n = 7$ combinado con intervalos de previsión prolongados no es óptimo para este entorno.

5.3.3. Combinación de wrappers

Los resultados presentados en la Figura 5.8 y la Tabla B.6 muestran el impacto de combinar los wrappers *MultiObsWrapper* y *WeatherForecastingWrapper* con sus mejores configuraciones encontradas en dos entornos distintos: 5 zonas y Radiant.

Para el entorno 5 zonas, utilizamos la siguiente configuración:

- *MultiObsWrapper*: $n = 10$ [4].
- *WeatherForecastingWrapper*: $n = 3$, $\delta = 1$ [5].

Para el entorno Radiant, la configuración utilizada fue:

- *MultiObsWrapper*: $n = 4$.
- *WeatherForecastingWrapper*: $n = 3$, $\delta = 2$.

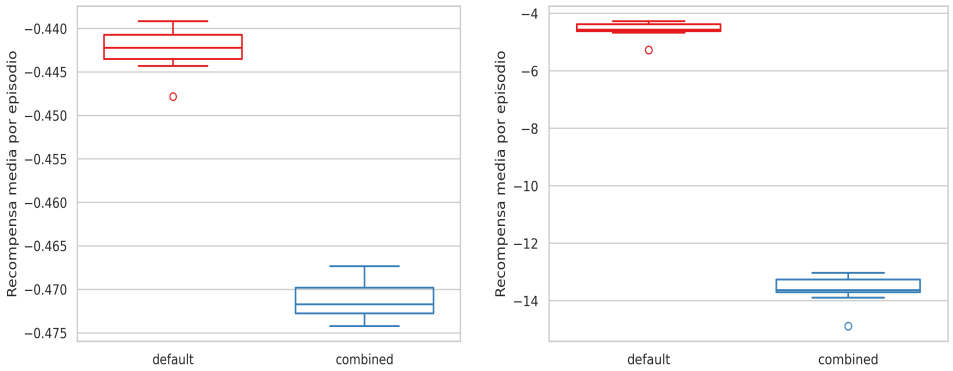
Los resultados indican que la combinación de los wrappers *MultiObsWrapper* y *WeatherForecastingWrapper* provoca una disminución en la recompensa media del agente.

En el entorno 5 zonas (versión continua), la configuración por defecto del agente proporciona una recompensa media de -0.442. Al combinar ambos wrappers, la recompensa media disminuye ligeramente a -0.471. Estos resultados indican que la combinación de wrappers no mejora el rendimiento, sino que empeora la recompensa media en comparación con la configuración original.

En el entorno Radiant, el efecto de la combinación es aún más negativo. La recompensa media con la configuración por defecto es de -4.568 con una desviación estándar de 0.282. Sin embargo, al combinar ambos wrappers, la

[4] Aunque podríamos haber utilizado $n = 20$, el resultado es prácticamente idéntico (-0.418 frente a -0.417), sin embargo, un mayor valor de n duplicaría el tamaño del historial de observaciones, incrementando la carga computacional sin una mejora significativa.

[5] Podríamos haber empleado $n = 5$ y $\delta = 2$, pero dado que los resultados son casi iguales (-0.352 frente a -0.348), incrementar el historial de predicciones de 3 a 10 añadiría una carga computacional innecesaria sin una mejora sustancial.



5 zonas (versión continua) Radiant

Figura 5.8: Impacto de la combinación de wrappers *MultiObsWrapper* y *WeatherForecastingWrapper* en los entornos 5 zonas y Radiant.

recompensa media cae drásticamente a -13.608 con una desviación estándar de 0.531. Esta marcada disminución sugiere que la combinación de wrappers introduce una complejidad adicional que no solo no beneficia el rendimiento, sino que lo deteriora de manera significativa.

5.4. Discusión

En esta sección, se discutirán las observaciones más relevantes obtenidas a partir de los experimentos realizados.

5.4.1. Comparación entre entornos continuos y discretos

Al comparar los resultados entre la versión continua y la versión discreta del entorno de 5 zonas, se observa una diferencia significativa en el impacto del historial de observaciones pasadas. En el entorno discreto, el uso del historial no solo no mejoró el rendimiento del agente, sino que, de hecho, resultó en una disminución de la recompensa media. Como se puede observar en la Tabla B.1 conforme vamos aumentando el número de observaciones pasadas va empeorando la recompensa media, pasando de -0.285 cuando no usamos historial de observaciones, hasta -0.408 cuando utilizamos un historial de observaciones de tamaño 20.

En contraste, en el entorno continuo, el historial condujo a mejoras sustanciales en el rendimiento del agente. Tal y como se ilustra en la Tabla B.2 tenemos los casos de $n = 10$ y $n = 20$ con unas recompensas medias de -0.418 y -0.417, respectivamente. Ambos resultados muestran una mejora clara respecto a no usar un historial de observaciones (-0.442)

Esta discrepancia puede explicarse por la diferencia en la representación de los estados entre entornos discretos y continuos. En los entornos discretos, los estados se representan mediante valores enteros o etiquetas, lo que limita la capacidad del modelo para captar relaciones sutiles entre ellos. Debido a que el espacio de estados es finito, estados que son muy similares en entornos continuos pueden ser considerados idénticos en entornos discretos, de manera que la información adicional del historial sea interpretada de manera confusa por el modelo.

Por otro lado, en un entorno continuo, los estados se representan mediante valores numéricos, lo que permite distinguir incluso diferencias sutiles entre ellos. Esto facilita al modelo la captura de relaciones y patrones útiles. En este contexto, el historial de observaciones puede ser más beneficioso, ya que proporciona un contexto adicional sin introducir el mismo nivel de ruido que en los entornos discretos.

5.4.2. Impacto de *MultiObsWrapper* en entornos continuos

El *MultiObsWrapper* se utiliza para proporcionar al agente de aprendizaje por refuerzo un contexto más amplio mediante un historial de observaciones pasadas. La efectividad de este enfoque varía significativamente según las características dinámicas del entorno.

En la versión contínua del entorno de 5 Zonas, los resultados indican que el uso del *MultiObsWrapper* con un historial de observaciones mejora el rendimiento del agente. Tal y como comentamos en la anterior apartado, la Tabla B.2 muestra que existen configuraciones que presentan una mejora respecto al no uso de un historial de observaciones.

Estos resultados sugiere que este entorno presenta una dinámica relativamente baja, ya que la información proveniente de estados pasados sigue siendo útil para la toma de decisiones. En este caso, el agente se beneficia de un historial de observaciones al permitirle comprender mejor la evolución del entorno y así tomar decisiones más informadas.

Sin embargo, al incrementar el tamaño del historial más allá de un punto óptimo, el rendimiento del agente puede estabilizarse o incluso empeorar, lo que indica que solo se necesita un historial limitado para capturar la dinámica de este entorno. Este hecho se puede apreciar para las configuraciones $n = 10$ y $n = 20$, las cuales presentan unos resultados de -0.418 y -0.417, respectivamente. Teniendo en el segundo caso un historial de observaciones con el doble de tamaño que el primero, la diferencia de rendimiento entre ambos es mínima.

De igual manera que es importante no excedernos con el tamaño del historial de observaciones, también es crucial no quedarnos cortos. Esto se

puede observar en la configuración $n = 4$ donde obtenemos una recompensa media de -0.450, siendo este valor peor al obtenido sin usar un historial de observaciones (-0.442). Usar un historial demasiado corto puede significar que no solo no alcancemos el máximo rendimiento potencial del agente, sino que incluso podemos generar un empeoramiento. Esta reflexión se aplica tanto al *MultiObsWrapper* como al *WeatherForecastingWrapper*.

Por lo tanto, un horizonte temporal de 2 horas y media parece ser una elección adecuada para este tipo de entornos.

En cuanto al entorno Radiant, el uso del *MultiObsWrapper* con un historial de observaciones tiene un impacto negativo en el rendimiento del agente, incluso cuando el historial es corto. La Tabla B.3 muestra como para todas las configuraciones (desde -7.477 para $n = 4$ hasta -120.930 para $n = 20$) se obtiene un empeoramiento respecto al no uso de historial de observaciones (-4.568)

Este resultado sugiere que el entorno tiene una dinámica alta, donde los estados pasados pierden relevancia rápidamente debido a la rápida evolución del entorno. En estos casos, un historial más breve, o incluso prescindir de él, puede ser más efectivo para mantener la relevancia de la información que el agente utiliza para tomar decisiones.

En base a este análisis, podemos inducir que el **entorno 5 Zonas tiene una dinámica menor en comparación con el entorno Radiant**. A través del siguientes análisis terminaremos de ver que es cierta esta afirmación.

5.4.3. Impacto del *WeatherForecastingWrapper* en entornos continuos

El *WeatherForecastingWrapper* permite al agente anticipar estados futuros mediante un historial de previsiones meteorológicas, con la opción de ajustar tanto el tamaño del historial como el intervalo de previsión. Los resultados obtenidos destacan que la efectividad de este enfoque varía considerablemente según la dinámica del entorno.

En entornos con baja dinámica, como en la versión continua del entorno de 5 Zonas, se observa que un historial breve y un intervalo de previsión corto son generalmente suficientes para que el agente tome decisiones efectivas. Los resultados muestran que un horizonte temporal más largo no ofrece beneficios adicionales significativos y, en algunos casos, puede resultar en una sobrecarga de información que perjudica el rendimiento del agente. Este hecho se refleja en la Tabla B.4, donde al comparar la configuración con un horizonte más corto ($n = 3$, $\delta = 1$) con la de un horizonte más largo ($n = 7$, $\delta = 3$), se observa que la primera configuración muestra una mejora signifi-

cativa respecto a no utilizar un historial de previsiones temporales (-0.352 frente a -0.442), mientras que la segunda presenta un deterioro considerable (-0.700 frente a -0.442). Configuraciones con un horizonte temporal de tamaño medio, como por ejemplo $n = 5$ y $\delta = 2$, también pueden proporcionar mejoras, aunque estas no superan (o lo hacen de manera mínima) a las obtenidas con horizontes temporales cortos. Además, estas configuraciones generan una mayor carga computacional.

Por lo tanto, un horizonte temporal de 3 horas, con información por hora, parece ser la elección más adecuada para este tipo de entornos.

En contraste, entornos con alta dinámica, como el entorno Radiant, presentan una mayor capacidad para aprovechar un horizonte temporal más largo. Aquí, los resultados sugieren que un historial más extenso y un intervalo de previsión más amplio permiten al agente anticipar y adaptarse mejor a las rápidas fluctuaciones del entorno. La Tabla B.5 muestra que la configuración $n = 3$ y $\delta = 5$ es la que claramente presenta el mejor rendimiento obtenido (-2.945), siendo una mejora importante respecto al no uso de un historial de previsiones temporales (-4.568). En este caso, un horizonte temporal de 6 horas, con información cada 2 horas, parece ser la mejor elección para entornos con esta naturaleza.

Este análisis confirma la afirmación previa: el entorno de 5 Zonas presenta una dinámica menor en comparación con el entorno Radiant.

Para finalizar, es importante mencionar el impacto de la variabilidad aplicada a las previsiones temporales. Como se mencionó en la sección 4.4, se ha introducido ruido en los datos meteorológicos utilizando un proceso de Ornstein-Uhlenbeck, para el cual hemos definido los siguientes valores de parámetros:

- σ: se ha fijado en 1.0, lo que significa que, aunque el proceso presenta fluctuaciones debido al ruido, estas están limitadas a una magnitud promedio de 1.
- μ: se ha establecido en 0.0, lo que implica que el proceso tiende siempre a regresar hacia este valor central.
- τ: se ha definido en 0.001, lo que indica una alta velocidad de reversión, haciendo que las fluctuaciones se corrijan rápidamente hacia la media.

En definitiva, es un proceso que aunque permite introducir cierta variabilidad en los datos, la limita rápidamente hacia la media, dando lugar a un comportamiento suave y centrado.

5.4.4. Comparación entre *MultiObsWrapper* y *WeatherForecastingWrapper*

Al comparar el *MultiObsWrapper* con el *WeatherForecastingWrapper*, se destacan diferencias notables en su impacto sobre el rendimiento del agente de aprendizaje por refuerzo en diversos entornos.

En todos los entornos continuos evaluados, el *WeatherForecastingWrapper* mostró una mejora en el rendimiento del agente (ver Tablas B.4 y B.5), mientras que el *MultiObsWrapper* resultó en una disminución del rendimiento en el entorno Radiant (ver Tabla B.3). Además, el *WeatherForecastingWrapper* demostró una mayor capacidad para optimizar el rendimiento del agente, proporcionando las mejores configuraciones en términos de recompensa media tanto en el entorno de 5 Zonas (-0.352 frente a -0.418) como en el entorno Radiant (-2.495 frente a -7.477). Esto subraya la superioridad del *WeatherForecastingWrapper* en la optimización del desempeño del agente[6].

Las diferencias en los resultados entre ambos wrappers pueden explicarse por el tipo de información que cada uno proporciona. El *MultiObsWrapper* únicamente ofrece información histórica del pasado a la observación actual del agente. Aunque esta representación puede ayudar a identificar tendencias recientes y prever el futuro inmediato, se basa en datos previamente procesados por la red neuronal, sin introducir nueva información de valor.

En contraste, tal como se comentó en la sección 4.4, el *WeatherForecastingWrapper* proporciona previsiones futuras en cada nuevo estado que no han sido procesadas previamente por la red neuronal, lo que aporta un valor añadido respecto al *MultiObsWrapper*.

Además, de cara a conocer el futuro, anticipar eventos próximos con cierta precisión suele ser más útil que conocer únicamente el pasado. La capacidad de prever lo que es probable que ocurra y ajustar las acciones en función de esas previsiones permite al agente adaptarse proactivamente a las condiciones cambiantes del entorno de mejor forma.

5.4.5. Impacto de la combinación de *MultiObsWrapper* y *WeatherForecastingWrapper* en diferentes entornos

Los resultados indican que, en ambos entornos analizados, la combinación de estos wrappers ha conducido a una disminución en el rendimiento

[6] Tener en cuenta que el *WeatherForecastingWrapper* permite ajustar el intervalo temporal entre cada previsión (aunque solo en horas), mientras que el *MultiObsWrapper* se ajusta al timestep de la simulación, que en este caso es de 15 minutos. Esta flexibilidad adicional en el *WeatherForecastingWrapper* facilita una adaptación más precisa a las necesidades específicas del entorno y del agente.

del agente en comparación con las configuraciones predeterminadas. En el entorno de 5 Zonas, se observa una reducción en la recompensa media del agente al utilizar la configuración combinada (-0.471 frente a -0.442). En el entorno Radiant, esta disminución es aún más pronunciada (-13.608 frente a -4.568).

Este deterioro en el rendimiento puede explicarse por la naturaleza complementaria de la información proporcionada por cada wrapper. El *WeatherForecastingWrapper* ofrece previsiones sobre eventos futuros, lo que resulta extremadamente útil para la toma de decisiones. Sin embargo, al combinarse con el *MultiObsWrapper*, que proporciona datos históricos, la adición de esta información pasada puede resultar irrelevante o incluso contraproducente, ya que puede introducir ruido o confusión en el proceso de toma de decisiones del agente.

Capítulo 6

Conclusiones y trabajo futuro

Este capítulo final resume las principales contribuciones de este trabajo, destacando especialmente el cumplimiento de los objetivos planteados y los resultados alcanzados. Además, se presentarán posibles líneas de trabajo futuro.

6.1. Conclusiones

A continuación revisaremos las conclusiones obtenidas obtenidos, centrándonos en las mejoras de rendimiento alcanzadas por los distintos agentes de DRL y en los horizontes temporales efectivos establecidos para cada entorno en base a la experimentación.

6.1.1. Mejoras de rendimiento

Los resultados obtenidos, presentados en la Tabla 6.1, destacan el impacto significativo que los diferentes *wrappers* tienen en el rendimiento de los entornos evaluados. Entre los dos enfoques probados, el *WeatherForecastingWrapper* se declara como un claro ganador, demostrando tener una mayor capacidad para mejorar la eficiencia del agente respecto al wrapper *MultiObsWrapper*.

Por un lado, en el entorno de 5 zonas, este *wrapper* logró un aumento del 20.4 % en comparación con los resultados estándar. Esto subraya cómo la incorporación de previsiones meteorológicas tiene una gran capacidad para mejorar el rendimiento del agente. En cuanto a desempeño de *MultiObsWrapper*, si bien no mostró mejoras en la versión discreta de este entorno, la

Entorno	Versión	Resultado por defecto	Wrapper	Mejor Resultado con wrapper	Mejora
5 zonas	**Discreta**	−0.285	*MultiObsWrapper*	−0.311	No conseguida
	Contínua	−0.442	*MultiObsWrapper*	−0.418	5.43 %
			WeatherForecastingWrapper	−0.352	20.4 %
			Combinados	−0.471	No conseguida
Radiant		−4.568	*MultiObsWrapper*	−7.477	No conseguida
			WeatherForecastingWrapper	−2.945	35.5 %
			Combinados	−13.608	No conseguida

Tabla 6.1: Resultados de los experimentos con diferentes configuraciones y wrappers.

versión continua, más representativa de situaciones reales y complejas, sí registró una mejora del 5.43 %. Esto indica que, aunque más modesto, el uso de datos históricos aún puede jugar un papel positivo en ciertas circunstancias.

Por otro lado, el entorno Radiant proporcionó un escenario aún más prometedor: el *WeatherForecastingWrapper* sobresalió nuevamente, con una significativa mejora del 35.5 %. En contraste, el *MultiObsWrapper* no solo no logró mejoras, sino que incluso perjudicó al rendimiento.

Finalmente, la combinación de ambos wrappers no muestra una mejora en los resultados para ningún entorno, debido a las razones comentadas en la sección 5.4.5.

6.1.2. Horizonte temporal efectivo

El *horizonte temporal efectivo* se refiere al período de tiempo durante el cual la información pasada o futura resulta relevante para la toma de decisiones de un agente en un entorno de aprendizaje por refuerzo. En otras palabras, define hasta qué punto los datos históricos y las previsiones influyen positivamente en el rendimiento del agente.

La Tabla 6.2 ofrece una visión sobre los horizontes temporales efectivos para los entornos 5 zonas y Radiant.

En el entorno de 5 zonas (versión continua), el horizonte temporal efectivo es de 2 horas y 30 minutos hacia el pasado, y de 3 horas hacia el futuro. Este entorno, caracterizado por una dinámica baja, muestra cómo una cierta cantidad tanto de información histórica como de previsiones futuras es útil para la toma de decisiones.

Por otro lado, el entorno Radiant presenta un escenario completamente diferente. Aquí, el horizonte temporal efectivo hacia el pasado es prácticamente inexistente, mientras que para el futuro se extiende a 6 horas, siendo el doble del que presenta el entorno 5 zonas. En un entorno tan dinámico como Radiant, los datos históricos pierden relevancia rápidamente debido a la constante evolución de las condiciones. En este caso, una mayor capacidad de prever y adaptarse al futuro se convierte en un factor decisivo,

Entorno	Horizonte temporal efectivo (Pasado)	Horizonte temporal efectivo (Futuro)
5 zonas (versión continua)	2 horas, 30 minutos	3 horas
Radiant (versión continua)	No hay, o es inferior a 1 hora	6 horas

Tabla 6.2: Horizonte temporal efectivo para los entornos contínuos estudiados.

permitiendo al agente responder eficazmente a los cambios rápidos y tomar decisiones estratégicas con una visión a largo plazo.

6.2. Cumplimiento de los objetivos

A continuación, se aborda el grado de cumplimiento de los objetivos planteados al inicio de este proyecto:

- **Subobjetivo 1: Investigación en fundamentos de DRL y procesos de Markov parcialmente observables, y su aplicación en control HVAC.** En el Capítulo 3, se exploran los principios fundamentales del aprendizaje profundo por refuerzo y se detallan los algoritmos que representan el estado del arte en este campo. Además, se discute su aplicación específica en el control HVAC, presentando el marco formal utilizado para definir este problema.
- **Subobjetivo 2: Revisión bibliográfica y del estado del arte. Detección de carencias en la literatura.** La sección 3.3 del Capítulo 3 ofrece una revisión exhaustiva de la literatura relevante, destacando los estudios más significativos en esta línea de investigación. A partir de este análisis, se identifican carencias notables en la literatura, como el bajo número de trabajos que tratan el impacto del aumento de la dimensionalidad de las entradas en el rendimiento del aprendizaje profundo por refuerzo, especialmente en su aplicación al control de sistemas de climatización.
- **Subobjetivo 3: Extensión de la librería de Python, Sinergym, con funcionalidades para la incorporación de previsiones meteorológicas al estado de los agentes.** En el Capítulo 5, se describe en detalle la herramienta Sinergym, destacando sus principales características y funcionamiento. Tras esto, se explica la contribución realizada mediante el desarrollo del *WeatherForecastingWrapper*, incluyendo los requisitos y detalles de implementación del mismo.

- **Subobjetivo 4: Propuesta de una metodología para la evaluación sistemática de diferentes configuraciones de entrenamiento de agentes DRL.** En la subsección 5.2 del Capítulo 6, se detalla la metodología de experimentación utilizada. Se especifica el conjunto de experimentos planificados y se muestra cómo, al variar los parámetros de los wrappers que afectan el horizonte temporal, se puede crear una batería de experimentos para evaluar una amplia gama de configuraciones y así encontrar la más adecuada para cada entorno.

- **Subobjetivo 5: Experimentación con diversos casos de uso en función de las configuraciones del algoritmo, el entorno y el espacio de estados utilizados.** Las subsecciones 5.3.1, 5.3.2 y 5.3.3 del Capítulo 6 presentan los experimentos realizados durante el proyecto, abordando diferentes configuraciones relacionadas con los algoritmos (DQN y PPO), entornos (5 zonas y Radiant) y espacios de estados (historial de observaciones pasadas y previsiones futuras).

Con el cumplimiento de estos subobjetivos, se ha alcanzado el objetivo principal del proyecto: **Investigar cómo ampliar el espacio de estados de un agente de DRL mediante el uso de información contextual disponible puede mejorar su desempeño en el contexto del control HVAC.** Los resultados presentados en las Tablas 6.1 y 6.2 de este mismo capítulo confirman que, efectivamente, la incorporación de información contextual es capaz de mejora el rendimiento del agente en el control de sistemas de climatización.

6.3. Trabajo futuro

Este trabajo abre numerosas posibilidades para futuras investigaciones. A continuación, se presentan algunas ideas para investigar:

- **Experimentación con nuevos entornos, climas y variables.** Sinergym posee otros entornos a parte de los utlizados en este trabajo, como son el *Datacenter*, *Small Datacenter*, y *Warehouse*, entre otros.[1]. El uso de estos entornos permitirá investigar en más profundidad cómo distintos tipos de edificios y sus características afectan el rendimiento del agente en el control HVAC.

- **Evaluación de algoritmos adicionales de DRL.** Se propone evaluar otros algoritmos de aprendizaje profundo por refuerzo disponibles en Stable Baselines3 (como DDPG, A2C, TD3 y SAC) así como

[1] Listado completo en https://ugr-sail.github.io/sinergym/compilation/main/pages/buildings.html

en otras librerías disponibles. Comparar estos algoritmos puede proporcionar una visión más completa sobre cuál es el más efectivo para diferentes configuraciones y entornos.

- **Extensión de los wrappers MultiObsWrapper y WeatherForecastingWrapper.** Se propone ampliar la funcionalidad de estos wrappers para permitir una configuración más flexible del intervalo entre estados y previsiones. En la actualidad, el intervalo entre estados en MultiObsWrapper está determinado por el timestep de la simulación, y en WeatherForecastingWrapper, el ajuste del intervalo entre previsiones se realiza por horas. Se sugiere extender estas capacidades para que el intervalo pueda configurarse en **minutos**, lo que permitirá una mayor granularidad en la revisión de configuraciones y la optimización de los entornos.

- **Ajuste de hiperparámetros**: Como comentamos en la sección 5.1.2, no se ha llevado a cabo un ajuste de hiperparámetros de los algoritmos utilizados para realizar unas comparaciones equitativas de estos. Un trabajo futuro podría consistir en encontrar los mejores parámetros para cada algoritmo dependiente del entorno, así conseguiríamos las mayores mejoras en el rendimiento de los agentes.

Apéndice A

Configuración de experimentos

En esta sección se detalla la configuración utilizada para los experimentos llevados a cabo. Cada tabla proporciona información para la reproducibilidad y comprensión de los experimentos hechos en Sinergym, desde la configuración del entorno y los parámetros del entrenamiento hasta las especificaciones de los algoritmos utilizados.

La Tabla A.1 proporciona detalles sobre el entorno utilizado para los experimentos en el edificio de 5 zonas (tanto versión contínua como discreta).

La Tabla A.1 proporciona detalles sobre el entorno utilizado para los experimentos en el edificio Radiant.

La Tabla A.3 presenta la configuración para el entrenamiento, validación y evaluación de los agentes de DRL.

La Tabla A.4 proporciona los detalles de los wrappers utilizados durante los experimentos. Se describen los parámetros de configuración para cada tipo de wrapper empleado.

La Tabla A.5 muestra los parámetros específicos utilizados en el algoritmo DQN.

Campo	**Valor**
Rama	main
Versión Actual	v3.3.3
Entorno	Eplus-5zone-mixed-{continuous/discrete}-stochastic-v1
Semilla	42

Tabla A.1: Información para reproducibilidad de experimentos sobre entorno 5 zonas en Sinergym.

Campo	Valor
Rama	uponor-develop
Versión Actual	v3.3.8
Entorno	Eplus-radiant_free_heating-mixed-continuous-stochastic-v1
Semilla	42

Tabla A.2: Información para reproducibilidad de experimentos sobre entorno Radiant en Sinergym.

Parámetro	Valor
Episodios de entrenamiento	20
Episodios validación	1 cada 2 episodios de entrenamiento
Episodios de evaluación	10
timesteps	15 minutos

Tabla A.3: Configuración entrenamiento, validación y evaluación de agentes DRL.

Wrapper	Parámetros
NormalizeActions	rango de normalización: [-1.0, 1.0]
NormalizeActions	epsilon: 0.00000001
MultiObsWrapper	flatten: TRUE
WeatherForecastingWrapper	columns: ['drybulb', 'relhum', 'winddir', 'windspd', 'dirnorrad', 'difhorrad'], weather_variability: [1.0, 0.0, 0.001]

Tabla A.4: Configuración wrappers utilizados

Parámetro	Valor
Tasa de aprendizaje	0.001
Tamaño de búfer	4096
Tamaño de lote	128
Tau	1.0
Gamma	0.99
Intervalo (pasos) actualización red objetivo	10000
Fracción de exploración	0.1
Exploración inicial	1.0
Exploración final	0.05

Tabla A.5: Parámetros de algorítmo DQN

Parámetro	Valor
Tasa de aprendizaje	0.001
Número de pasos	4096
Tamaño de lote	128
Número de épocas	15
Gamma	0.9
GAE lambda	0.9

Tabla A.6: Parámetros de algorítmo PPO

Finalmente, la Tabla A.6 proporciona los parámetros del algoritmo PPO.

Apéndice B

Resultados de evaluación

En esta sección se presentan los resultados obtenidos de los experimentos realizados utilizando los wrappers *MultiObsWrapper* y *WeatherForecastingWrapper*.

La Tabla B.1 muestra los resultados para diferentes tamaños de historial n en la versión discreta del entorno de 5 zonas utilizando *MultiObsWrapper*.

La Tabla B.2 presenta los resultados de la versión continua del entorno 5 zonas con diferentes tamaños de historial n utilizando *MultiObsWrapper*.

La Tabla B.3 muestra los resultados obtenidos en el entorno Radiant con diferentes tamaños de historial n utilizando *MultiObsWrapper* en su versión continua.

La Tabla B.4 muestra la recompensa media y la desviación estándar para distintos valores de n y δ en la versión continua del entorno 5 zonas utilizando *WeatherForecastingWrapper*.

La Tabla B.5 muestra los resultados para diferentes valores de n y δ en la versión continua del entorno Radiant utilizando *WeatherForecastingWrapper*.

Finalmente, la Tabla B.6 presenta la recompensa media y la desviación estándar para las mejores configuraciones combinadas de los wrappers *Mul-*

Configuración	**Recompensa Media**
Por defecto ($n = 0$)	-0.285 ± 0.001
$n = 4$	-0.311 ± 0.002
$n = 10$	-0.399 ± 0.002
$n = 20$	-0.408 ± 0.003

Tabla B.1: Resultado de experimentos con *MultiObsWrapper* en la versión discreta del entorno 5 zonas.

Configuración	Recompensa Media
Por defecto ($n = 0$)	-0.442 ± 0.003
$n = 4$	-0.450 ± 0.002
$n = 10$	-0.418 ± 0.002
$n = 20$	-0.417 ± 0.003

Tabla B.2: Resultado de experimentos con *MultiObsWrapper* en la versión contínua del entorno 5 zonas.

Configuración	Recompensa Media
Por defecto ($n = 0$)	-4.568 ± 0.282
$n = 4$	-7.477 ± 0.377
$n = 10$	-76.046 ± 26.114
$n = 20$	-120.930 ± 22.725

Tabla B.3: Resultado de experimentos con *MultiObsWrapper* en la versión contínua del entorno Radiant.

Recompensa Media	$\delta = 1$	$\delta = 2$	$\delta = 3$
$n = 3$	-0.352 ± 0.003	-0.411 ± 0.002	-0.413 ± 0.003
$n = 5$	-0.412 ± 0.002	-0.348 ± 0.002	-0.666 ± 0.003
$n = 7$	-0.376 ± 0.002	-0.379 ± 0.001	-0.700 ± 0.005
Valor por Defecto: -0.442 ± 0.003			

Tabla B.4: Resultado de experimentos con *WeatherForecastingWrapper* en la versión contínua del entorno 5 zonas.

Recompensa media	$\delta = 1$	$\delta = 2$	$\delta = 3$
$n = 3$	-4.717 ± 0.326	-2.945 ± 0.171	-3.038 ± 0.167
$n = 5$	-3.398 ± 0.187	-3.556 ± 0.225	-3.813 ± 0.181
$n = 7$	-3.748 ± 0.236	-7.396 ± 0.352	-6.837 ± 0.302
Valor por defecto: -4.568 ± 0.282			

Tabla B.5: Resultado de experimentos con *WeatherForecastingWrapper* en la versión contínua del entorno Radiant.

Entorno	Configuración	Recompensa Media
5 zonas	Default	-0.442 ± 0.003
	Combined	-0.471 ± 0.002
Radiant	Default	-4.568 ± 0.282
	Combined	-13.608 ± 0.531

Tabla B.6: Recompensa media y desviación estándar para el uso combinado de las mejores configuraciones de los wrappers *MultiObsWrapper* y *WeatherForecastingWrapper*.

tiObsWrapper y *WeatherForecastingWrapper* en los entornos de 5 zonas y Radiant.

Bibliografía

[1] ASHRAE ANSI y M ASHRAE. «Standard 55—Thermal Environmental Conditions for Human Occupancy». En: *ASHRAE, Atlanta* (2004).

[2] Donald Azuatalam et al. «Reinforcement learning for whole-building HVAC control and demand response». En: *Energy and AI* 2 (2020), pág. 100020.

[3] Richard Bellman. «Dynamic programming». En: *Science* 153.3731 (1966), págs. 34-37.

[4] Richard Bellman. «The theory of dynamic programming». En: *Bulletin of the American Mathematical Society* 60.6 (1954), págs. 503-515.

[5] Marco Biemann et al. «Experimental evaluation of model-free reinforcement learning algorithms for continuous HVAC control». En: *Applied Energy* 298 (2021), pág. 117164. DOI: 10.1016/j.apenergy.2021.117164.

[6] Silvio Brandi et al. «Deep Reinforcement Learning to optimise indoor temperature control and heating energy consumption in buildings». En: *Energy and Buildings* 224 (2020), pág. 110225.

[7] Yujiao Chen et al. «Optimal control of HVAC and window systems for natural ventilation through reinforcement learning». En: *Energy and Buildings* 169 (2018), págs. 195-205.

[8] Xianzhong Ding, Wan Du y Alberto E. Cerpa. «MB2C: Model-Based Deep Reinforcement Learning for Multi-zone Building Control». En: BuildSys '20. New York, NY, USA, 2020, págs. 50-59. DOI: 10.1145/3408308.3427986.

[9] Guanyu Gao, Jie Li y Yonggang Wen. «DeepComfort: Energy-Efficient Thermal Comfort Control in Buildings Via Reinforcement Learning». En: *IEEE Internet of Things Journal* 7.9 (2020), págs. 8472-8484. DOI: 10.1109/JIOT.2020.2992117.

[10] Nicolas Pardo Garcia et al. *Heat and cooling demand and market perspective*. Publications Office of the European Union, 2012.

[11] Surbhi Gupta, Gaurav Singal y Deepak Garg. «Deep Reinforcement Learning Techniques in Diversified Domains: A Survey». En: *Archives of Computational Methods in Engineering* 28.7 (dic. de 2021), págs. 4715-4754. DOI: 10.1007/s11831-021-09552-3.

[12] Mengjie Han et al. «A review of reinforcement learning methodologies for controlling occupant comfort in buildings». En: *Sustainable Cities and Society* 51 (2019), pág. 101748. DOI: 10.1016/j.scs.2019.101748.

[13] Chloe Ching-Yun Hsu, Celestine Mendler-Dünner y Moritz Hardt. «Revisiting Design Choices in Proximal Policy Optimization». En: *arXiv preprint arXiv:2009.10897* (2020).

[14] Javier Jiménez-Raboso et al. «Sinergym: a building simulation and control framework for training reinforcement learning agents». En: BuildSys '21. New York, NY, USA: Association for Computing Machinery, 2021, págs. 319-323. DOI: 10.1145/3486611.3488729.

[15] Russell Jurney. *Agile data science 2.0: Building full-stack data analytics applications with spark*. O'Reilly Media Inc, 2017.

[16] Joaquim Leitão et al. «A Survey on Home Energy Management». En: *IEEE Access* 8 (2020), págs. 5699-5722. DOI: 10.1109/ACCESS.2019.2963502.

[17] Paulo Lissa et al. «Deep reinforcement learning for home energy management system control». En: *Energy and AI* 3 (2021), pág. 100043. DOI: 10.1016/j.egyai.2020.100043.

[18] Antonio Manjavacas et al. «An experimental evaluation of deep reinforcement learning algorithms for HVAC control». En: *Artificial Intelligence Review* 57.7 (jun. de 2024), pág. 173. DOI: 10.1007/s10462-024-10819-x.

[19] Karl Mason y Santiago Grijalva. «A review of reinforcement learning for autonomous building energy management». En: *Computers & Electrical Engineering* 78 (2019), págs. 300-312. DOI: 10.1016/j.compeleceng.2019.07.019.

[20] Volodymyr Mnih et al. «Human-level control through deep reinforcement learning». En: *nature* 518.7540 (2015), págs. 529-533.

[21] Volodymyr Mnih et al. «Playing atari with deep reinforcement learning». En: *arXiv preprint arXiv:1312.5602* (2013).

[22] Elena Mocanu et al. «On-Line Building Energy Optimization Using Deep Reinforcement Learning». En: *IEEE Transactions on Smart Grid* 10.4 (2019), págs. 3698-3708. DOI: 10.1109/TSG.2018.2834219.

[23] M. Morales. *Grokking Deep Reinforcement Learning*. Manning Publications, 2020.

[24] Takeshi Morinibu, Tomohiro Noda y Tanaka Shota. «Application of Deep Reinforcement Learning in Residential Preconditioning for Radiation Temperature». En: *2019 8th International Congress on Advanced Applied Informatics (IIAI-AAI)*. 2019, págs. 561-566. DOI: 10.1109/IIAI-AAI.2019.00120.

[25] Takao Moriyama et al. «Reinforcement learning testbed for power-consumption optimization». En: *Asian Simulation Conference*. Springer. 2018, págs. 45-59.

[26] Michael C Mozer. «The neural network house: An environment hat adapts to its inhabitants». En: *Proc. AAAI Spring Symp. Intelligent Environments*. Vol. 58. 1998.

[27] Kei Ota et al. «Can Increasing Input Dimensionality Improve Deep Reinforcement Learning?» En: *Proceedings of the 37th International Conference on Machine Learning*. Ed. por Hal Daumé III y Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, jul. de 2020, págs. 7424-7433.

[28] Batchu Rajasekhar et al. «A Survey of Computational Intelligence Techniques for Air-Conditioners Energy Management». En: *IEEE Transactions on Emerging Topics in Computational Intelligence* 4.4 (2020), págs. 555-570. DOI: 10.1109/TETCI.2020.2991728.

[29] Naren Srivaths Raman et al. «Reinforcement Learning for Control of Building HVAC Systems». En: jul. de 2020, págs. 2326-2332. DOI: 10.23919/ACC45564.2020.9147629.

[30] Yuiko Sakuma e Hiroaki Nishi. «Airflow Direction Control of Air Conditioners Using Deep Reinforcement Learning». En: *2020 SICE International Symposium on Control Systems (SICE ISCS)*. 2020, págs. 61-68. DOI: 10.23919/SICEISCS48470.2020.9083565.

[31] Tom Schaul et al. «Prioritized experience replay». En: *arXiv preprint arXiv:1511.05952* (2015).

[32] John Schulman et al. «Proximal policy optimization algorithms». En: *arXiv preprint arXiv:1707.06347* (2017).

[33] Richard S Sutton y Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[34] Hado Van Hasselt, Arthur Guez y David Silver. «Deep reinforcement learning with double q-learning». En: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016.

[35] José Vázquez-Canteli, Jérôme Kämpf y Zoltán Nagy. «Balancing comfort and energy consumption of a heat pump using batch reinforcement learning with fitted Q-iteration». En: *Energy Procedia* 122 (2017), págs. 415-420.

[36] José R Vázquez-Canteli y Zoltán Nagy. «Reinforcement learning for demand response: A review of algorithms and modeling techniques». En: *Applied energy* 235 (2019), págs. 1072-1089.

[37] José R Vázquez-Canteli et al. «Fusing TensorFlow with building energy simulation for intelligent energy management in smart cities». En: *Sustainable cities and society* 45 (2019), págs. 243-257.

[38] Yuan Wang, Kirubakaran Velswamy y Biao Huang. «A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems». En: *Processes* 5.3 (2017), pág. 46.

[39] Zhe Wang y Tianzhen Hong. «Reinforcement learning for building controls: The opportunities and challenges». En: *Applied Energy* 269 (2020), pág. 115036.

[40] Ziyu Wang et al. «Dueling network architectures for deep reinforcement learning». En: *International conference on machine learning*. PMLR. 2016, págs. 1995-2003.

[41] Tianshu Wei, Yanzhi Wang y Qi Zhu. «Deep reinforcement learning for building HVAC control». En: *Proceedings of the 54th annual design automation conference 2017*. 2017, págs. 1-6.

[42] Ting Yang et al. «Reinforcement learning in sustainable energy and electric systems: a survey». En: *Annual Reviews in Control* 49 (2020), págs. 145-163. DOI: 10.1016/j.arcontrol.2020.03.001.

[43] Young Ran Yoon y Hyeun Jun Moon. «Performance based thermal comfort control (PTCC) using deep reinforcement learning for space cooling». En: *Energy and Buildings* 203 (2019), pág. 109420. DOI: 10.1016/j.enbuild.2019.109420.

[44] Liang Yu et al. «A Review of Deep Reinforcement Learning for Smart Building Energy Management». En: *IEEE Internet of Things Journal* 8.15 (2021), págs. 12046-12063. DOI: 10.1109/JIOT.2021.3078462.

[45] Liang Yu et al. «Multi-agent deep reinforcement learning for HVAC control in commercial buildings». En: *IEEE Transactions on Smart Grid* 12.1 (2020), págs. 407-419.

[46] Xiaolei Yuan et al. «Study on the application of reinforcement learning in the operation optimization of HVAC system». En: *Building Simulation*. Springer. 2020, págs. 1-13.

[47] Alexander Zai y Brandon Brown. *Deep reinforcement learning in action*. Manning Publications, 2020.

[48] Dongxia Zhang, Xiaoqing Han y Chunyu Deng. «Review on the research and practice of deep learning and reinforcement learning in smart grids». En: *CSEE Journal of Power and Energy Systems* 4.3 (2018), págs. 362-370. DOI: 10.17775/CSEEJPES.2018.00520.

[49] Zhiang Zhang y Khee Poh Lam. «Practical implementation and evaluation of deep reinforcement learning control for a radiant heating system». En: *Proceedings of the 5th Conference on Systems for Built Environments*. 2018, págs. 148-157.

[50] Zhiang Zhang et al. «A deep reinforcement learning approach to using whole building energy model for hvac optimal control». En: *2018 Building Performance Analysis Conference and SimBuild*. Vol. 3. 2018, págs. 22-23.

[51] Zhiang Zhang et al. «Whole building energy model for HVAC optimal control: A practical framework based on deep reinforcement learning». En: *Energy and Buildings* 199 (2019), págs. 472-490.

[52] Zhengbo Zou, Xinran Yu y Semiha Ergan. «Towards optimal control of air handling units using deep reinforcement learning and recurrent neural network». En: *Building and Environment* 168 (2020), pág. 106535. DOI: 10.1016/j.buildenv.2019.106535.