UNIVERSIDAD DE GRANADA

TRABAJO FIN DE MASTER

Máster universitario oficial en Ciencia de Datos e Ingeniería de Computadores

Análisis y desarrollo de funciones de recompensa en control energético de edificios

---

Estudio de influencia de funciones de recompensa en control HVAC de edificios con aprendizaje por refuerzo profundo

Autor
Ahmed Brek Prieto

Directores
Miguel Molina Solana
Juan Gómez Romero

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, septiembre de 2023

Análisis y desarrollo de funciones de recompensa en control energético de edificios

Ahmed Brek Prieto

Palabras clave: control energético de edificios, sistemas de climatización, aprendizaje por refuerzo, función de recompensa

Resumen

Durante las últimas décadas, la preocupación por el cambio climático ha aumentado significativamente, pues el impacto medioambiental es cada vez más notorio. Al tratarse de un problema de carácter global, la búsqueda de soluciones resulta urgente. El consumo energético en edificios es crucial, ya que supone más de un tercio del consumo a nivel mundial, y alrededor de un 40 % de la emisión de gases contaminantes a la atmósfera. Dentro de los edificios, la climatización constituye uno de los mayores gastos de energía. Así, optimizar estos sistemas es decisivo para mitigar este problema.

Tradicionalmente, el control de los sistemas de climatización se ha tratado con controladores basados en reglas, o técnicas de *model predictive control*. Durante los últimos años, el aprendizaje por refuerzo profundo ha atraído bastante la atención de la comunidad científica, lo que ha llevado a numerosos investigadores a tratar este problema mediante dicho paradigma, obteniendo resultados bastante prometedores. Sin embargo, la función de recompensa es un aspecto que no recibe suficiente atención, ya que en algunos estudios se utilizan propuestas bastante mejorables.

En este trabajo de fin de máster se aborda el problema del control energético en edificios utilizando métodos de aprendizaje por refuerzo profundo. Se centra la atención en los sistemas de climatización, con el objetivo de minimizar el consumo energético y mantener el confort térmico. Además, se proponen distintas funciones de recompensa y se estudia su impacto en el aprendizaje del agente. Utilizando un entorno de simulación, se lleva a cabo una completa experimentación y su posterior análisis, con el objetivo de dar respuesta a esta cuestión.

Design and analysis of reward functions for building energy control.

Ahmed Brek Prieto

Keywords: building energy control, HVAC, reinforcement learning, reward function

Abstract

Over the past few decades, concern about climate change has significantly increased as the environmental impact becomes more noticeable. Considering that this is a global issue, the search for solutions has become urgent. Energy consumption in buildings accounts for more than one-third of global energy consumption and is responsible for $\sim 40\,\%$ of greenhouse gas emissions. HVAC (heating, ventilation, and air conditioning) systems represent one of the largest energy expenses in buildings. Therefore, optimizing these systems is decisive to mitigate this problem.

Traditionally, HVAC system control has been handled with rule-based controllers or model predictive control techniques. In recent years, deep reinforcement learning has drawn the attention of the scientific community, leading many researchers to address this problem using this paradigm and achieving promising results. However, the reward function does not receive enough attention, as some studies use rewards that are far from optimal.

This master's thesis addresses the issue of building energy control using deep reinforcement learning methods, focusing on HVAC systems. The goal is minimizing energy consumption while maintaining thermal comfort. Furthermore, various reward functions are proposed, and their impact on the agent's learning is studied. Simulation environments are used to run the necessary experiments. Results are analyzed in order to research this subject.

Yo, **Ahmed Brek Prieto**, alumno del Máster Universitario Oficial en Ciencia de Datos e Ingeniería de Computadores de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77037522B, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Master en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Ahmed Brek Prieto

Granada a 6 de septiembre de 2023.

D. **Miguel José Molina Solana** y D. **Juan Gómez Romero**, profesores del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada

INFORMAN:

Que el presente trabajo, titulado ***Análisis y desarrollo de funciones de recompensa en control energético de edificios. Estudio de influencia de funciones de recompensa en control HVAC de edificios con aprendizaje por refuerzo profundo***, ha sido realizado bajo su supervisión por **Ahmed Brek Prieto**, 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 2023.

Los directores:

Miguel José Molina Solana **Juan Gómez Romero**

Agradecimientos

Me gustaría expresar mi agradecimiento a todas las personas que, de una forma u otra, me han mostrado su apoyo durante esta etapa académica que, sin duda, ha sido de las más enriquecedoras en varios sentidos.

En primer lugar, quiero agradecer a mis tutores Miguel y Juan, por toda la atención que me han dedicado desde el principio, dispuestos a guiarme con su experiencia y sabiduría en todo momento. Estoy muy agradecido con la confianza que han depositado en mí y la oportunidad que me han brindado para seguir formándome académicamente.

Al resto de compañeros del grupo de investigación del que formo parte, por su impecable trato y su cercanía desde el primer día. Poder trabajar con un equipo tan excelente ha sido una experiencia muy gratificante.

A mi familia y amigos, que han estado presentes durante toda esta intensa etapa. Gracias a su apoyo incondicional han conseguido que estos meses de arduo trabajo hayan sido más agradables y llevaderos.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

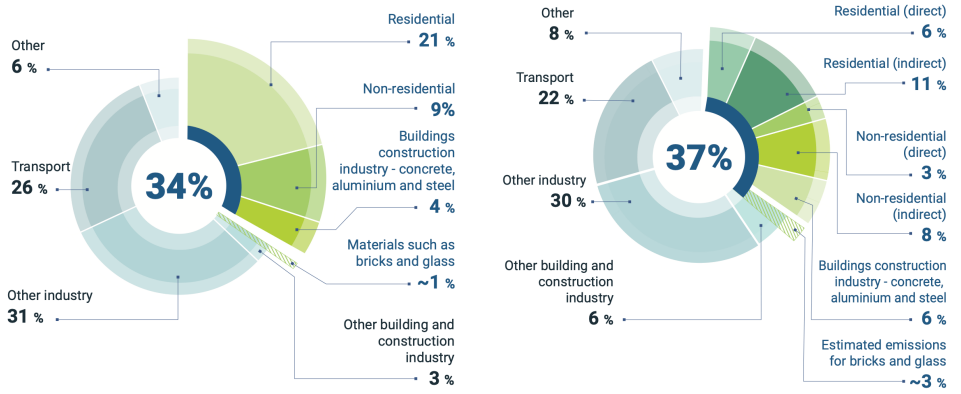
1.1. Motivación

En las últimas décadas, el cambio climático se ha convertido en una de las mayores preocupaciones de toda la humanidad. El constante agravamiento de la situación no se ha conseguido solventar o detener a pesar de las medidas que se han adoptado. De esta manera, la búsqueda de soluciones, de cualquier tipo y en cualquier ámbito, para atenuar o frenar las consecuencias de este fenómeno permanece incesante. El cambio climático es consecuencia directa del desmesurado consumo energético por parte de la sociedad contemporánea en prácticamente todos los países del mundo. Por tanto, resulta conveniente estudiar a qué fines se dedica la energía que se genera.

Según el informe anual de 2022 de *Global Alliance for Buildings and Construction* [1], la energía se destina principalmente a transporte, industria y, en su mayoría a edificios. Más concretamente, alrededor del 30 % de la energía se utiliza en edificios tanto residenciales como no residenciales, aunque fundamentalmente a los primeros (véase Figura 1.1a). Por otro lado, se encuentra también que los edificios son responsables del 28 % de las emisiones de CO_2 de forma directa e indirecta (véase Figura 1.1b). Además, se prevé que el consumo energético en edificios aumentará hasta un 50 % en las próximas tres décadas [2, 3]. De esta manera, reducir el consumo de energía en edificios resulta ser una buena alternativa a la hora de contrarrestar los efectos del cambio climático.

Durante un análisis más profundo de los usos de la energía en edificios, resulta que la mayor parte se destina a su climatización [1]. Esto hace que sea fundamental optimizar el uso de dispositivos de climatización (de aquí en adelante, HVAC, por sus siglas en inglés[1]), minimizando así su consumo.

[1] HVAC: *Heating, Ventilation and Air Conditioning.*



(a) Consumo energético. (b) Emisiones de CO_2.

Figura 1.1: Reporte de *Global Alliance for Buildings and Construction* 2022.

Este problema puede atajarse desde distintas perspectivas, ya sea modificando el diseño de estos aparatos para mejorar su eficiencia, o bien optimizando la política de control que se aplique. Este último enfoque es el desarrollado durante este trabajo.

El control energético en edificios no es en sí un problema reciente, sino que ha sido enfrentado durante las últimas décadas. Tradicionalmente se han utilizado modelos clásicos como sistemas basados en reglas (RBCs). Una ventaja que ofrecen es que, para conseguir un modelo sencillo que funcione de forma razonable, basta con definir un conjunto de reglas que cubran los posibles estados y determinar la acción que debe realizarse en cada caso. El problema surge cuando se intenta afinar y optimizar su funcionamiento, ya que aparece la necesidad de considerar más variables de entrada y diseñar reglas más específicas, a veces sin apenas disponer de conocimiento experto o información adicional. Cuando se utilizan más variables de entrada, el número de posibles estados crece muy rápidamente (explosión combinatoria), por lo que el diseño del RBC se convierte en una tarea tediosa y compleja. También se pierde la interpretabilidad del modelo cuando el número de reglas aumenta. Otra dificultad adicional es la necesidad de adaptar las reglas a cada edificio, ya que difícilmente un RBC de calidad pueda ser genérico. Además, cabe destacar que estos modelos no son capaces de realizar ningún tipo de aprendizaje ni optimización tras su implementación. Por todas estas razones, los RBCs presentan un rendimiento bastante limitado.

Una alternativa bastante extendida a la hora de afrontar este problema son los MPCs (*model predictive control*). Es un método de control avanzado utilizado en ingeniería y sistemas de control de procesos. Consiste en utilizar un modelo matemático del sistema para realizar predicciones futuras y cal-

cular secuencias de acciones de control óptimas para maximizar una función de costo. Por tanto, resulta ser un *framework* adecuado para resolver este problema. Una de sus ventajas es que permite adaptarse a los cambios que pueda haber en el entorno, tratándose así de un modelo dinámico en lugar de uno estático como sería un RBC. También es útil para tratar restricciones, como podría ser mantener el confort térmico de los ocupantes. Sin embargo, como se ha mencionado, se necesita un modelo matemático que describa la dinámica del sistema a controlar, para poder realizar las predicciones y estimar el beneficio de elegir cada acción. Esto es una limitación considerable, ya que no siempre se conoce o se puede formular matemáticamente la dinámica del entorno.

Dicho esto, se propone utilizar aprendizaje por refuerzo (*reinforcement learning*, RL de aquí en adelante), pues permite usar técnicas sofisticadas de *machine learning* aplicables a problemas de control, evitando los principales inconvenientes de los paradigmas anteriormente mencionados. Existe una fase de entrenamiento, en la que el agente se optimiza de forma automática a lo largo de la simulación, por lo que no es necesario que el programador diseñe reglas como en el caso del RBC. Además, permite entrenar agentes sin necesidad de un modelo matemático de la dinámica del sistema (como ocurre con MPC), lo que facilita bastante la creación de modelos. Por su parte, en RL se realiza un entrenamiento del agente mediante una estrategia de ensayo y error en un entorno simulado. En la sección 2.2 se describen con más detalle las bases del aprendizaje por refuerzo.

Gracias al avance tecnológico en el diseño de GPUs y el consecuente desarrollo de potentes y prometedoras técnicas de *deep learning*, el aprendizaje por refuerzo ha retomado importancia y protagonismo durante los últimos años. De esta manera, surge una prometedora vertiente que combina ambos campos: el aprendizaje por refuerzo profundo, o *deep reinforcement learning* (DRL, de aquí en adelante). Se propone utilizar técnicas de este paradigma para resolver el problema de control energético en edificios, dado el buen rendimiento que han demostrado durante los últimos años en problemas de una alta complejidad.

Al utilizar aprendizaje por refuerzo para enfrentar un problema, la manera en que se formule y se definan sus componentes toma una importancia crucial en los resultados que se podrán alcanzar. Para formular un problema, deben definirse elementos como el entorno y sus posibles estados, el agente y su espacio de acciones, y la función de recompensa, entre otros[2]. Es este último aspecto el que quizás recibe menos atención de la que debería, al menos cuando se trata de control energético en edificios. Este problema de

[2]Todos estos aspectos se definen y explican en la sección 2.2.1.

control se ha abordado desde distintas perspectivas, sin embargo, no se ha estudiado con profundidad la influencia de utilizar distintas recompensas, lo que podría afectar de forma significativa a los resultados obtenidos. Normalmente, se elige una única función de recompensa, sin contemplar otras posibilidades, y en muchas ocasiones, se observa que esta recompensa no es óptima.

En definitiva, a lo largo de este proyecto se pretende resolver el problema del control energético en edificios, abordado desde la perspectiva del aprendizaje por refuerzo. Además, se propone diseñar varias funciones de recompensa, experimentar con ellas y estudiar de qué manera influyen en los resultados obtenidos. De esta manera, se pretende aportar conocimiento útil a la comunidad científica, arrojando luz en un aspecto apenas estudiado hasta el momento. Las conclusiones que resulten de este proyecto pueden aplicarse en cualquier estudio sobre este ámbito, ya que no contradicen la dirección en la que avanza la línea de investigación.

1.2. Problema planteado y propuesta

El problema a resolver en este proyecto es el **control energético inteligente en edificios**, o SBEM por sus siglas en inglés (*smart building energy management)*. Consiste en controlar de forma automática y eficiente los dispositivos eléctricos de un edificio, con el objetivo de minimizar el consumo mientras se cumple con las necesidades de los usuarios. En este caso, se centra la atención en los **sistemas de climatización** del edificio, por lo que se trata de maximizar el ahorro energético manteniendo la temperatura del interior dentro de un intervalo de confort deseado.

Para abordar este problema, se propone entrenar modelos utilizando varios algoritmos de **aprendizaje por refuerzo profundo** en un entorno virtual. De forma paralela, se pretende **estudiar la influencia de la función de recompensa**, para lo que se diseñan y comparan distintas propuestas, analizando el impacto en los agentes resultantes. Para ello se desarrolla una amplia experimentación, descrita en el capítulo 5. Adicionalmente, se propone estudiar otros aspectos relevantes en el paradigma del aprendizaje por refuerzo (capítulo 6).

1.3. Objetivos

Con el propósito de ilustrar la finalidad de este proyecto, se muestra a continuación una lista de los objetivos concretos que conlleva la resolución del problema descrito.

Objetivo 1

Estudio del paradigma de aprendizaje por refuerzo, aprendizaje por refuerzo profundo, y su aplicación al control energético.

Objetivo 2

Revisión de la literatura y el estado del arte. Detección de carencias y propuesta de soluciones.

Objetivo 3

Estudio de las herramientas *software* para la resolución del problema: Sinergym y AutoCloud.

Objetivo 4

Diseño y ejecución de la experimentación necesaria para el estudio planteado. Entrenamiento y evaluación de varios modelos candidatos.

Objetivo 5

Análisis de resultados y obtención de conclusiones. Proposición de posibles trabajos futuros.

1.4. Presupuesto

En esta sección se detalla un presupuesto adecuado a este proyecto. Con el propósito de realizarlo de la forma más objetiva y correcta posible, se han tomado como referencia fuentes oficiales y fiables. En particular, se ha consultado la página web oficial de la Seguridad Social Española para obtener los porcentajes referentes a la cotización de la seguridad social. Para estimar un salario anual bruto medio en el caso de un ingeniero de aprendizaje automático en España, se ha tomado la referencia que propone esta web.

El **presupuesto final** obtenido es de **23.224'68€** en total. El detalle se muestra en la Tabla 1.1.

Concepto	**Coste**
Personal contratado	**21.846'30€**
Sueldo anual bruto	42.420'00€
Sueldo mensual bruto	3.535'00€
Duración del proyecto (meses)	5
Seguridad Social	23'60 %
Coste mensual total	4.369'26€
Hardware	**1.178'38€**
Ordenador ASUS ROG Strix G15	1.150'00€
Ratón Logitech M185	15'38€
Teclado Logitech K120	13'00€
Costes de ejecución	**200'00€**
Google Cloud Platform Services	200'00€

Tabla 1.1: Desglose del presupuesto total del proyecto.

1.5. Planificación y cronograma

La duración total de la realización del proyecto ha sido de 122 días, con una media aproximada de 4 horas diarias. Esto se traduce en un total aproximado de 488 horas. En un diagrama de Gantt (Figura 1.2, en la siguiente página) se ilustra con detalle las distintas fases del desarrollo del trabajo.

1.6. Estructura de la memoria

La estructura de este documento es la siguiente. El primer capítulo introduce la motivación y objetivos del proyecto, describiendo también el problema enfrentado y detallando la planificación temporal y el presupuesto. En el segundo capítulo, se explican los conceptos fundamentales necesarios para la comprensión del estudio. A continuación, se dedica un capítulo para analizar el estado del arte y explicar en qué punto se encuentra esta línea de investigación, seguido de un capítulo donde se expone la metodología seguida y algunos detalles referentes al *software* y *hardware* utilizados. El quinto y el sexto capítulos refieren a la experimentación planteada y su posterior análisis, respectivamente. Como cierre, se dedica un último capítulo al análisis de conclusiones que se derivan de este estudio, donde también se proponen posibles líneas de investigación futuras.

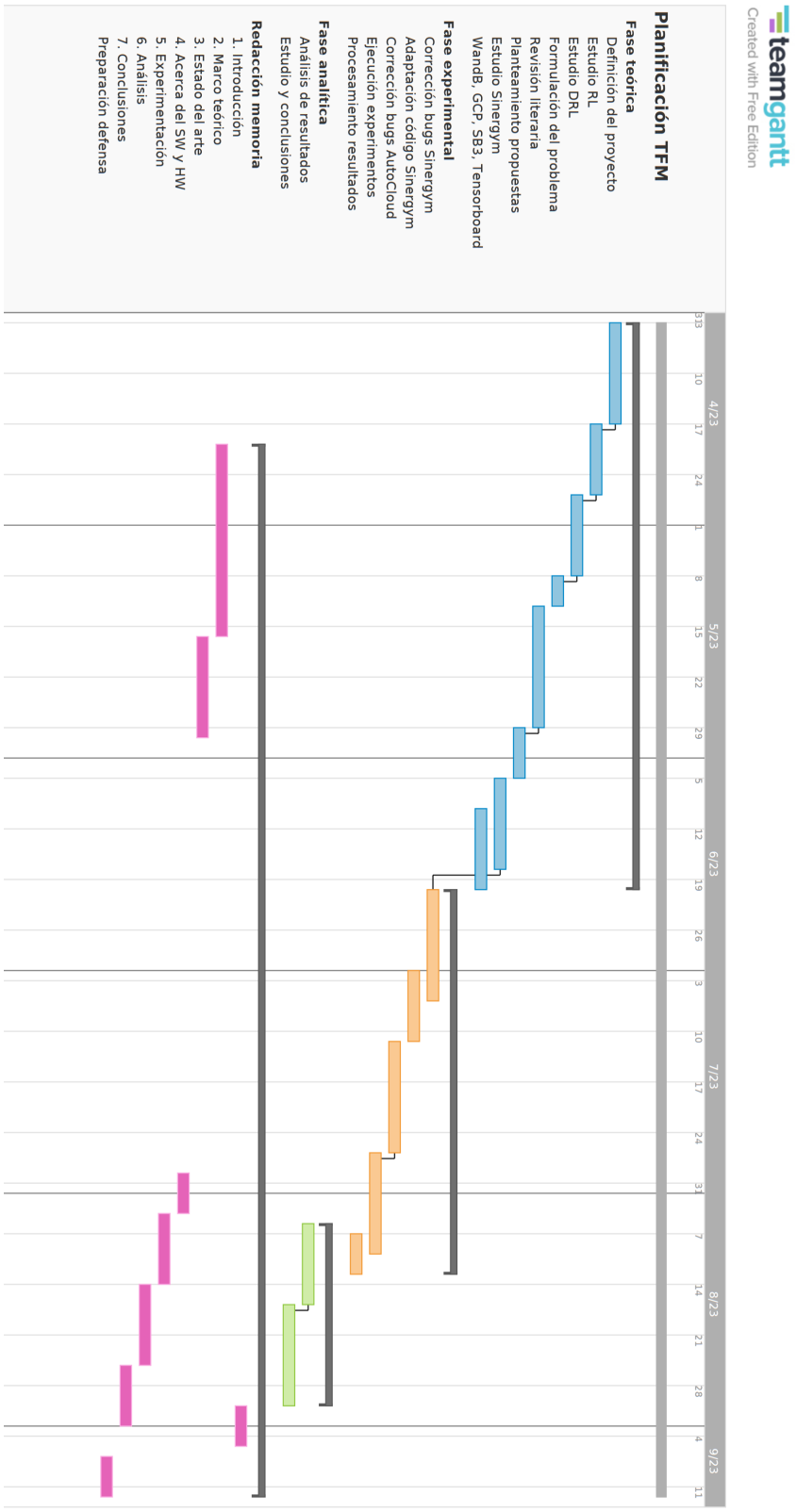


Figura 1.2: Cronograma del proyecto.

Capítulo 2

Marco teórico

2.1. Aprendizaje automático

[3]El **aprendizaje automático** (o *machine learning*) es probablemente la rama más popular de la inteligencia artificial en la actualidad. A pesar de no ser un área de estudio nueva, sus posibilidades se han explotado durante las últimas décadas, dando lugar a soluciones inconcebibles hasta hace no muchos años.

Una de sus definiciones más aceptadas es la siguiente: un programa informático aprende de una experiencia E con respecto a alguna tarea T y una medición del rendimiento P, si su rendimiento en la tarea T, medido como P, mejora con la experiencia E [5]. Una aproximación más informal define el aprendizaje automático como el campo de estudio que otorga a los ordenadores la habilidad de aprender sin haber sido explícitamente programados para resolver una tarea [6].

En el aprendizaje automático, se diferencian tres clases, dependiendo del tipo de datos con los que se entrenen:

Aprendizaje supervisado. Se trata de aproximar una función a partir de los **datos** de entrenamiento, que estarán **etiquetados**. Cada dato se caracteriza por una serie de propiedades, acompañado por una etiqueta, que se encarga de indicar el resultado real de la función para esa entrada. El modelo (función) aproximado trata de devolver una respuesta correcta ante nuevos datos nunca antes vistos por el agente. Existen **problemas de regresión y clasificación**, dependiendo del dominio de las etiquetas (continuo o discreto, respectivamente).

Aprendizaje no supervisado. Cuando el conjunto de datos usado para

[3]La redacción de este capítulo se ha basado en el capítulo homónimo de un proyecto anterior del mismo autor, ya que el marco teórico es común [4].

el entrenamiento no tiene etiquetas. Asimismo, no existe conocimiento a priori, por lo que el agente intenta **descubrir patrones** en los datos para poder asociarles una etiqueta. Se distinguen problemas de agrupamiento (*clustering*) y reglas de asociación.

Aprendizaje semisupervisado. Propuesto por S. Abney y Mitchell, entre otros. Es el caso en que solo una parte de los datos, generalmente una minoría, están etiquetados, y el resto no [7].

Aprendizaje por refuerzo. En este caso, el objetivo es determinar qué acciones debe ejecutar un agente en un entorno para maximizar una señal de recompensa. Para ello, se utiliza una estrategia de ensayo y error mediante la que **el propio agente genera los datos** para el aprendizaje. Se explica con mayor detenimiento en la sección 2.2.

En este trabajo fin de máster, centraremos la atención en el aprendizaje por refuerzo, ya que es el que mejor se adapta a los problemas de control, como el que vamos a abordar.

2.2. Aprendizaje por refuerzo

Como ocurre con la IA, el aprendizaje por refuerzo, o *reinforcement learning* (RL de aquí en adelante), es más antiguo de lo que se cree. Surge alrededor de los años 50, donde existían se trataba el mismo problema desde distintos puntos. En uno de ellos, se abordaba el aprendizaje desde el paradigma de ensayo y error. El otro enfoque era más teórico, y se centraba en el problema de control óptimo, tratando de resolverlo mediante programación dinámica y funciones de valor, sin incluir un aprendizaje como tal. Más adelante, las dos vertientes se fusionaron, haciendo resurgir el aprendizaje por refuerzo con un enfoque más parecido al actual [8]. Durante los últimos años, gracias a los avances en el desarrollo *hardware* y su integración con aprendizaje profundo (*deep learning*), el RL ha retomado una gran relevancia en la comunidad científica, dado su éxito en juegos de Atari o el clásico Go [9, 10].

El término "aprendizaje por refuerzo" abarca tanto una categoría de problemas como una clase de métodos que ofrecen soluciones a estos, así como la disciplina científica que los estudia. Se puede definir como un tipo de aprendizaje computacional en el que el objetivo del agente es **maximizar una señal de recompensa**: después de cada acción realizada (o paso de tiempo transcurrido), el agente recibe una señal de recompensa del entorno, la cual indica el beneficio o desventaja resultante de la acción en consideración, teniendo en cuenta el estado actual del entorno. El agente no recibe instrucciones explícitas sobre qué acción tomar en cada momento, sino que debe descubrir por sí mismo cuáles acciones le permiten maximizar la recompensa a largo plazo [8]. Para lograrlo, debe interactuar con su entorno

y desarrollar una estrategia de prueba y error para aproximar una política óptima. Además, este enfoque es particularmente valioso en el contexto de **problemas de toma de decisiones**, donde subyace el concepto de procesos de decisión de Markov, el cual se analiza con más detalle posteriormente.

A pesar de que el aprendizaje supervisado y no supervisado han eclipsado durante muchos años al RL, algunos autores entre los que destacan Barto y Sutton, sostienen la existencia de diferencias sustanciales entre estos paradigmas [8]. Se describen a continuación.

El **aprendizaje supervisado** no es aplicable a problemas interactivos (como problemas de control o juegos), ya que necesita un conjunto de datos etiquetados para su entrenamiento y, en este tipo de problemas, es prácticamente imposible generar una muestra correcta y representativa de los estados posibles del entorno.

Por otro lado, podría considerarse que en RL se trabaja con datos no etiquetados, característica común al **aprendizaje no supervisado**. Sin embargo, en este último, el objetivo consiste en detectar estructuras subyacentes en los datos, no se trata de maximizar una función de recompensa, como sí ocurre en RL.

Un aspecto a destacar sobre el aprendizaje por refuerzo es el compromiso entre exploración y explotación que debe alcanzarse. La explotación consiste en tomar las acciones que el agente sabe por su experiencia que ofrecen una buena recompensa inmediata. Por su parte, durante la exploración se eligen acciones aparentemente "subóptimas" con el objetivo de encontrar aquellas que, a largo plazo, permitan alcanzar una recompensa mayor. Sin embargo, estos procesos no pueden realizarse simultáneamente, por lo que se comete cierto error durante la búsqueda de un equilibrio entre ambas estrategias.

2.2.1. Elementos del aprendizaje por refuerzo

En todo problema de aprendizaje por refuerzo, se distinguen principalmente dos elementos: un agente y el entorno en el que este existe:

- El **agente**, encargado del aprendizaje. A partir de su interacción con el entorno, trata de aprender un comportamiento óptimo con el que maximice la recompensa obtenida a largo plazo. En cada instante, observa el estado del entorno y determina la acción a realizar, tras lo que recibe una señal de recompensa que le premia o penaliza según lo adecuada que haya sido dicha acción.

- Muchos autores describen el **entorno** como «aquello que el agente no puede modificar arbitrariamente»[8]. Así, la función de recompensa forma parte del entorno, pues aunque el agente pueda conocerla, no puede modificarla. Se define como dinámica del entorno aquella que define su comportamiento. Cabe notar que normalmente, el agente no conoce esta dinámica y suele existir cierta aleatoriedad o incertidumbre. Un entorno es **estacionario** si y solo si su dinámica no cambia con el tiempo.

A continuación, se describen algunos elementos adicionales típicos de un problema de RL:

- Una **política** del agente, que define su comportamiento. Indica qué acción ejecutar según el estado del entorno. Destacan las políticas estocásticas, que en lugar de devolver una acción por cada estado, indican una distribución de probabilidad sobre las posibles acciones a realizar, dando normalmente más probabilidad a aquellas que considere mejores. Esto es útil para lidiar con el compromiso explotación-exploración anteriormente mencionado.

- **Función de recompensa**. Se trata de un elemento clave, pues permite definir el objetivo del problema. El agente recibe esta señal tras ejecutar cada acción, que tratará de maximizar a largo plazo. Por sí sola, la señal de recompensa tiene un carácter cortoplacista.

- La **función de valor** es calculada por el agente en base a la función de recompensa. Permite aproximar la calidad de cada acción o estado en base a la experiencia. Así, evita el sesgo "cortoplacista" mencionado, y permite al agente estimar la calidad de cada acción a largo plazo, de tal manera que este pueda elegir acciones con peor recompensa, pero más prometedores, es decir, con mayor recompensa acumulada o esperable en el futuro.

- En algunas ocasiones puede conocerse un **modelo del entorno**. Define su comportamiento y permite realizar predicciones al agente sobre los cambios en el entorno que conllevarían ciertas acciones. Con este conocimiento a priori, se puede realizar cierta planificación antes de ejecutar cada acción. Esto permite la existencia de métodos basados en modelo (*model-based methods*) y métodos libres de modelo (*model-free methods*). En cualquier caso, cabe destacar que en la mayoría de problemas reales no se conoce este modelo del entorno, como ocurre en el problema aquí enfrentado.

Todos estos conceptos serán formalizados y concretados al problema aquí tratado en la sección 2.5.

2.2.2. Tipos de problemas en RL

En el paradigma de aprendizaje por refuerzo, se pueden distinguir varios tipos de problemas según algunas características.

Por un lado, dependiendo de la tarea que se trate de resolver, se distinguen **problemas de predicción y de control**. En los primeros, el objetivo es, dada una política, predecir la recompensa total esperable si se parte de un estado concreto y se sigue la política dada. Por su parte, en un problema de control, se desea aproximar una política óptima capaz de maximizar la recompensa total obtenida a largo plazo. Cabe añadir que el problema tratado en este trabajo es del primer tipo.

Por otro lado, en RL, el proceso de interacción del agente con su entorno puede dividirse en intervalos de tiempo regulares, denominados *timesteps*. En cada paso, el agente podrá realizar una única acción. Dicho esto, en función de si el agente alcanza o no un estado terminal, dando fin a lo que se conoce como episodio, se pueden distinguir dos tipos de problemas: **episódicos o continuados**. En estos últimos, como se puede intuir, no existen estados finales, por lo que no hay fin del episodio. En consecuencia, resulta necesario incluir alguna limitación temporal para detener el aprendizaje o evaluación del agente.

2.2.3. Procesos de decisión de Markov

A la hora de formular un problema en el marco de RL, es necesario definirlo dentro del marco de los **procesos de decisión de Markov** (MDP, por sus siglas en inglés) [11]. Estos permiten formalizar matemáticamente un problema de RL, de tal manera que se pueda garantizar la optimización bajo ciertas circunstancias. Este tipo de problemas se define según los siguientes conceptos:

- **Conjunto de estados** posibles del entorno, es decir, las distintas posibles combinaciones que pueden tomar las variables del entorno.
- **Distribución de probabilidades de transición**. Dado un estado inicial y una acción completa, permite conocer qué probabilidades hay de pasar a cada uno de los posibles estados en el siguiente paso, tras realizar dicha acción. Se corresponde con la dinámica del entorno antes mencionada.
- **Espacio de acciones**, compuesto por las diferentes posibles acciones que un agente puede realizar en cada estado.

- La **función de recompensa**, explicada anteriormente, indica el beneficio o perjuicio inmediato de realizar una acción en cada momento.
- **Factor de descuento**, que permite ajustar la importancia que el algoritmo otorga a la recompensa inmediata respecto a la recompensa futura.

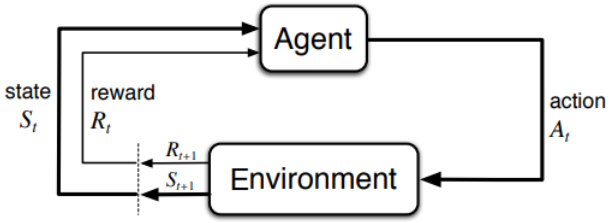


Figura 2.1: Esquema de un proceso de decisión de Markov.

Otros aspecto a destacar de los MDP es la **propiedad de Markov**. Esta sostiene que un estado cumple la propiedad de Markov si la probabilidad de transición a un estado S_t y de obtener una recompensa R_t depende únicamente del estado S_{t-1} y acción A_{t-1} inmediatamente anteriores. Matemáticamente, se define como sigue:

$$p(s' \mid s_t, a_t) \doteq p(s' \mid s_t,\ a_t,\ s_{t-1},\ a_{t-1},\ s_{t-2},\ a_{t-2},\ ...,\ s_0,\ a_0) \tag{2.1}$$

Teniendo en cuenta que las transiciones son eventos excluyentes, la suma de sus probabilidades debe ser unitaria. De forma matemática, queda:

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, for\ all\ s\ in\ \mathcal{S},\ a\ in\ \mathcal{A}(s) \tag{2.2}$$

Dependiendo de la naturaleza de la distribución de probabilidades de transición, se pueden diferenciar dos tipos de entornos: **determinista o estocástico**. El primer caso se da cuando para cada par estado-acción sólo existe una posible transición (no hay ningún tipo de aleatoriedad o incertidumbre). Por su parte, el caso estocástico es aquel que, como su nombre indica, incluye una componente azarosa. Esto es, dado un par estado-acción, existe un conjunto de posibles estados a los que se puede pasar con una probabilidad menor que uno, de forma aleatoria. De esta forma, no se pueden predecir con total seguridad las transiciones futuras, sino que hay que contemplar esta componente estocástica durante cualquier tipo de planificación

o decisión, resultando así en un problema más complejo.

Como se ha comentado anteriormente, en aprendizaje por refuerzo, el agente trata de **maximizar la función de recompensa** que recibe del entorno, siempre a largo plazo. De esta manera, en el campo más sencillo, la recompensa acumulada o *return* (G) es la suma de las recompensas individuales obtenidas en cada *timestep*:

$$G_t \doteq R_{t+1} \ + \ R_{t+2} \ + \ ... \ + R_T, \text{ siendo T el último } \textit{timestep} \tag{2.3}$$

Para ello, se trata de calcular la recompensa estimada o esperable R tras realizar una acción a en un estado s. Formalmente, se expresa como:

$$r(s,a) \doteq \mathbb{E}[R_t \mid S_{t-1} = s, \ A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a) \tag{2.4}$$

Dicho esto, se puede introducir cómo afecta el **factor de descuento** anteriormente mencionado, notado como λ. Como se ha explicado, sirve para ajustar la importancia de las recompensas inmediatas frente a las futuras, permitiendo de esta manera modelar la incertidumbre que se tiene durante las predicciones. Se introduce en el cálculo de la recompensa de la siguiente manera:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + ... \tag{2.5}$$

Definido de forma recursiva quedaría así:

$$G_t \doteq R_{t+1} + \gamma G_{t+1} \tag{2.6}$$

2.2.4. Políticas y funciones de valor

Otro elemento crucial en aprendizaje por refuerzo son las **funciones de valor**. Surgen para tratar de paliar los inconvenientes que presenta la función de recompensa por su enfoque "cortoplacista". Estas funciones de valor se utilizan en todos los algoritmos mínimamente sofisticados en este ámbito. Consisten en funciones de estados o de pares estado-acción, que permiten aproximar el beneficio o perjuicio que supone al agente alcanzar un estado o elegir una acción, en el largo plazo. Así, un estado que permita acumular más recompensa en el futuro, tendrá un mayor valor para la función de

estado-valor, y viceversa. Cabe añadir que estas funciones de valor deben definirse en base a una política concreta, que indica el comportamiento que seguiría el agente en los siguientes pasos.

Se puede definir una **política** de manera formal como una función que, por cada posible estado del entorno, devuelve una distribución de probabilidades de seleccionar cada una de las posibles acciones disponibles en ese momento. Esto es, dada una política π y un instante de tiempo t, se tiene que $\pi(a|s)$ es la probabilidad de $A_t = a$ si $S_t = t$, siendo S_t el estado del entorno en el instante t y A_t la acción elegida en ese mismo *timestep*.

Una vez definido el concepto de política, se pueden formalizar tanto la función de estado-valor como la función acción-valor. Cabe notar que el valor de ambas funciones se puede estimar con la experiencia del agente, promediando la recompensa acumulada que se obtiene probando diferentes "caminos", mediante ensayo y error.

Así, una **función estado-valor** es, dado un estado s y una política π, la recompensa acumulada esperable al seguir la política *pi* partiendo del estado s. Se nota $v_\pi(s)$, y matemáticamente se describe como sigue:

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \;\middle|\; S_t = s\right], \; \forall s \in \mathcal{S} \end{aligned} \tag{2.7}$$

En caso de que exista un estado terminal, la función de valor de este debe ser 0.

Análogamente se define la **función acción-valor**. La única diferencia es que en este caso, además del estado de partida, también se determina la primera acción a realizar. Su notación es $q_\pi(s, a)$:

$$\begin{aligned} q_\pi(s,a) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s, \; A_t = a] \\ &= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \;\middle|\; S_t = s, \; A_t = a\right] \end{aligned} \tag{2.8}$$

Políticas óptimas

En aprendizaje por refuerzo, las funciones de valor anteriormente mencionadas presentan algunas características de recursividad, de donde surge la ecuación de Bellman [12]:

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi\Big[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \mid S_t = s\Big] \\
&= \sum_a \pi(a|s) \sum_{s'}\sum_r p(s', r|s, a)\Big[r + \gamma\mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s']\Big] \qquad (2.9)\\
&= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_\pi(s')\Big],\ \forall s \in \mathcal{S}
\end{aligned}
$$

Adaptando esta ecuación a la función de acción-valor, se obtiene:

$$
\begin{aligned}
q_\pi(s, a) &\doteq \mathbb{E}_\pi\Big[G_t \mid S_t = s, A_t = a\Big] \\
&= \sum_{s'}\sum_r p(s', r|s, a)\Big[r + \gamma\mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s']\Big] \qquad (2.10)\\
&= \sum_{s'}\sum_r p(s', r|s, a)\Big[r + \gamma\sum_{a'}\pi(a'|s')q_\pi(s', a')\Big]
\end{aligned}
$$

Como se ha explicado, durante el tratamiento de un problema de aprendizaje por refuerzo se trata de buscar una política óptima (o al menos la mejor posible). Para ello, resulta necesario poder comparar varias políticas. Así, en un proceso de decisión de Markov, se dice que una política es mejor que otra si la primera presenta una recompensa acumulada esperable mayor que la segunda en todos los estados. Formalmente sería:

$$\pi \geq \pi' \iff v_\pi(s) \geq v_{\pi'}(s)\ \forall s \in \mathcal{S} \qquad (2.11)$$

Esto implica que siempre existirá al menos una o varias políticas óptimas. Estas serán aquellas que sean mejor o igual que el resto, y se notan como π_*. Otra implicación es que todas las políticas óptimas tendrán una misma función estado-valor, por lo que son equivalentes en este sentido. Esta función estado-valor óptima se nota como $v_*(s)$:

$$v_*(s) \doteq \max_\pi v_\pi(s),\ \forall s \in \mathcal{S} \qquad (2.12)$$

De nuevo, existe una analogía para la función acción-valor q_*:

$$q_*(s, a) \doteq \max_\pi q_\pi(s, a),\ \forall s \in \mathcal{S},\ a \in \mathcal{A} \qquad (2.13)$$

La función v_* debe satisfacer la autoconsistencia, dada la ecuación de Bellman para las funciones estado-valor. Además, por ser una función óptima, no hace falta expresar esta condición de autoconsistencia en función de una política específica, ya que la función en cuestión es única y común para todas las políticas óptimas. Esta expresión toma el nombre de **ecuación de optimización de Bellman**:

$$\begin{aligned} v_*(s) &= \max_a \mathbb{E}_{\pi_*}[G_t | S_t = s,\ A_t = a] \qquad \text{(por 2.6)} \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s,\ A_t = a] \\ &= \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')] \end{aligned} \tag{2.14}$$

Una vez más, existe una expresión análoga para q_*:

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\middle|\; S_t = s, A_t = a\right] \\ &= \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma \max_{a'} q_*(s', a')\right] \end{aligned} \tag{2.15}$$

Por último, es importante notar que estas políticas óptimas tienen cierto carácter teórico. Dadas las importantes limitaciones de memoria y cálculo computacional existentes, suele ser extremadamente complicado o inviable encontrar una política verdaderamente óptima. Así, suele ser suficiente encontrar una **política subóptima** que resuelva de forma aceptable el problema enfrentado. Por esta misma razón, nace el éxito de los **métodos no tabulares**, que a pesar de no lograr rápidamente la mejor solución, permiten desarrollar un modelo aceptable en muchos problemas reales.

2.3. Iteración de la política. Iteración de valor

Llegados a este punto, se puede explicar el proceso que se sigue para resolver un problema en RL que, como se ha indicado, consiste en encontrar una política óptima (o subóptima). Para ello, se siguen dos procesos que se explicarán a continuación: **evaluación de política** y **mejora de política**. Ambas consisten en dos fases independientes que se aplican de forma iterativa.

2.3.1. Evaluación de la política

Conviene explicar primero la fase de evaluación de política (*policy evaluation*). Para esta evaluación, es necesario calcular la función estado-valor de la política en cuestión. Este cálculo comienza inicializando el valor de v_0 de forma aleatoria para todos los estados[4]. A continuación, se aplica la siguiente expresión repetitivamente para aproximar el valor real de v:

$$\begin{aligned} v_{k+1}(s) &\doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}),\ |\ S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_k(s')\Big],\ \forall s \in \mathcal{S} \end{aligned} \tag{2.16}$$

Así, queda el algoritmo que se define abajo, denominado **algoritmo de evaluación iterativa de la política** *iterative policy evaluation*:

Algoritmo 1: Evaluación iterativa de la política

Entrada: - una política π a evaluar
- el conjunto de estados $\mathcal{S}$
- el espacio de acciones $\mathcal{A}$
- el factor de descuento γ
- un umbral de precisión θ

---

repetir
  $\Delta \leftarrow 0$
  para cada $s \in \mathcal{S}$ **hacer**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  fin
mientras $\Delta > \theta$

Este algoritmo es en cierto modo intuitivo, una vez se entienden los conceptos hasta aquí explicados. El objetivo aquí es aproximar la función de estado-valor de una política concreta iterando hasta alcanzar una precisión deseada, con el subsiguiente objetivo de comparar varias políticas. Para ello, se va actualizando la función estado-valor de forma iterativa hasta alcanzar este umbral de precisión. Para estimar esta función, para cada estado s, se calcula la recompensa asociada a pasar a un siguiente estado s' ponderada con su probabilidad de transición y por la probabilidad de esa transición bajo una política concreta.

[4]Salvo para el terminal, que toma valor 0.

2.3.2. Mejora de la política

Además de la forma de evaluar una política, es necesario explicar cómo mejorarla. Para ello, se parte de un estado s y se considera como mejor opción a aquella que permita transicionar a un siguiente estado s' con la máxima recompensa acumulada esperable. Para la estimación de la recompensa esperable asociada a realizar cada posible acción, se suma las recompensas asociadas a los posibles siguientes estados s' ponderados por la probabilidad de pasar a estos estados s'. Matemáticamente, esta actualización se describe como sigue:

$$\begin{aligned} \pi'(s) &\doteq \arg\max_a q_\pi(s,a) \\ &= \arg\max_a \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big] \end{aligned} \tag{2.17}$$

2.3.3. Iteración de la política e iteración de valor

Con estas dos herramientas, se puede llevar a cabo un método iterativo mediante el que aproximar una política óptima. Básicamente consiste en alternar ambas fases hasta alcanzar una política suficientemente buena.

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{M} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{M} \pi_2 \xrightarrow{E} \ldots \xrightarrow{M} \pi_* \xrightarrow{E} v_*$$

Este algoritmo se conoce como **iteración de la política** (*policy iteration*):

Algoritmo 2: Iteración de la política

Entrada: - una política π a evaluar
- el conjunto de estados $\mathcal{S}$
- el espacio de acciones $\mathcal{A}$
- el factor de descuento γ

---

[1] Inicializar $V(s) \in \mathbb{R}$ y $\pi(s) \in \mathcal{A}(s)$ arbitrariamente $\forall s \in \mathcal{S}$

[2] Evaluación de la política

repetir
  $\Delta \leftarrow 0$
  para cada $s \in \mathcal{S}$ **hacer**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  fin
mientras $\Delta > \theta$ *(determina la precisión de la estimación)*

[3] Mejora de la política

politica_estable $\leftarrow$ *true*
para cada $s \in \mathcal{S}$ **hacer**
  accion_anterior $\leftarrow \pi(s)$
  $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$ **si**
   accion_anterior $\neq \pi(s)$ **entonces**
    politica_estable $\leftarrow$ *false*
fin
si *politica_estable* **entonces**
  devolver $V \approx v_*$ y $\pi \approx \pi_*$
en otro caso
  ir a [2]
fin

Una vez más, existe una alternativa análoga referente a la función estado-valor. Se conoce como **iteración de valor** (*value iteration*, y parte de una función $v(s)$ arbitraria, a la que aplica la estrategia de mejora de Ecuación 2.17 de forma iterativa con el objetivo de aproximar la función de estado-valor óptima $v_*(s)$. El algoritmo se define así:

Algoritmo 3: Iteración de valor

Entrada: - el conjunto de estados $\mathcal{S}$
- el espacio de acciones $\mathcal{A}$
- el factor de descuento γ
- un umbral de precisión θ

Salida: - La política $\pi \approx \pi_*$ aproximada

---

repetir
  $\Delta \leftarrow 0$
  para cada $s \in \mathcal{S}$ **hacer**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  fin
mientras $\Delta > \theta$

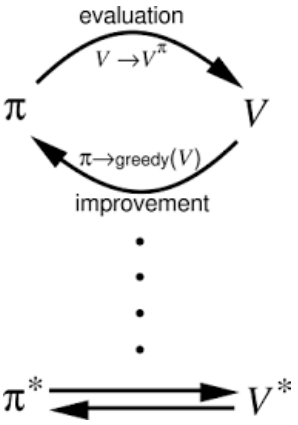
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$

devolver $\pi(s)$

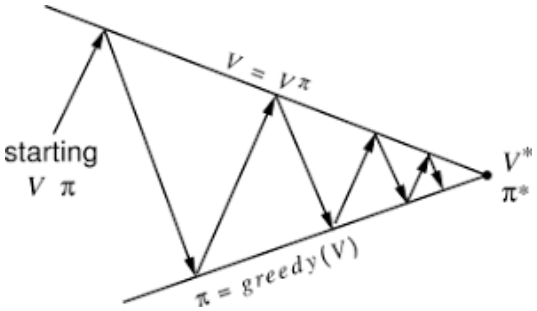
Ante estas dos opciones, surge la pregunta de cuál es el método más adecuado. Empíricamente, el método de iteración de política tiene una convergencia más rápida [8], aunque es más costoso computacionalmente, por lo que la capacidad de cómputo a la hora de enfrentar el problema resulta crucial para decantarse por alguna de estas posibilidades.

Iteración de Política Generalizada

Por último, cabe mencionar la idea de **Iteración de Política Generalizada**, o GPI, por sus siglas en inglés. En términos generales, se conoce como GPI a la estrategia que alterna una fase de evaluación de la política con otra fase de mejora. Es un concepto en el que se basan la gran mayoría de algoritmos de aprendizaje por refuerzo. Las siguientes figuras ilustran esta idea de forma esquemática. Nótese que la definición de GPI es independiente de la granularidad o frecuencia de intercalado de estos procesos.



(a) Esquema GPI.



(b) Representación bidimensional de evolución de GPI.

Figura 2.2: Imágenes ilustrativas del modelo GPI.

2.4. Algoritmos

En esta sección se explican los algoritmos utilizados para la resolución del problema de este proyecto. Se trata de métodos de aprendizaje por refuerzo profundo (DRL). La razones por las que se utilizan estos métodos más complejos y, en cierto modo, sofisticados son varias. Por un lado, es temporal y computacionalmente inviable utilizar métodos más básicos como son los **métodos tabulares**, ya que estos deben almacenar en una tabla todas las combinaciones de estados y posibles acciones junto a la estimación de la función valor de cada uno. Estos métodos solo se pueden utilizar en problemas con espacios de estados y de acciones extremadamente reducidos, ya que cuando estos crecen un poco, el número de posibles combinaciones crece de forma exponencial (explosión combinatoria). Por esta razón, resulta necesario el uso de métodos basados en aproximación de funciones. Otra razón por la que se utiliza aprendizaje por refuerzo profundo es porque estos métodos han demostrado en numerosos estudios un gran potencial. De hecho, durante más de la última década, todas las publicaciones que afrontan este problema, lo hacen utilizando métodos de DRL, por lo que se tiene certeza empírica de que estos modelos ofrecen buenas soluciones. Por último, el *software* utilizado únicamente soporta algoritmos de DRL.

En cualquier caso, los métodos "clásicos" en los que se basan los algoritmos aquí explicados se pueden consultar en [8, 4]. A continuación, se explican los algoritmos A2C, DQN, PPO, SAC y TD3.

2.4.1. A2C

Advantage Actor Critic, abreviado A2C, consiste en un método *on-policy* de la familia *actor-critic*. Estos métodos utilizan dos redes neuronales que trabajan de manera cooperativa. La red *actor* se encarga de elegir las acciones a ejecutar, mientras que la red *critic* valora los distintos estados, informando a la primera red sobre qué estados conviene explotar o explorar. De esta manera se va evaluando la política y mejorando de forma iterativa, siguiendo el esquema GPI anteriormente explicado.

La arquitectura ***actor-critic*** se ilustra de forma esquemática en la Figura 2.3. La red *actor* devuelve una política π_θ para un estado s, y la red *critic* devuelve el valor de un estado s ($V_\varphi(s)$).

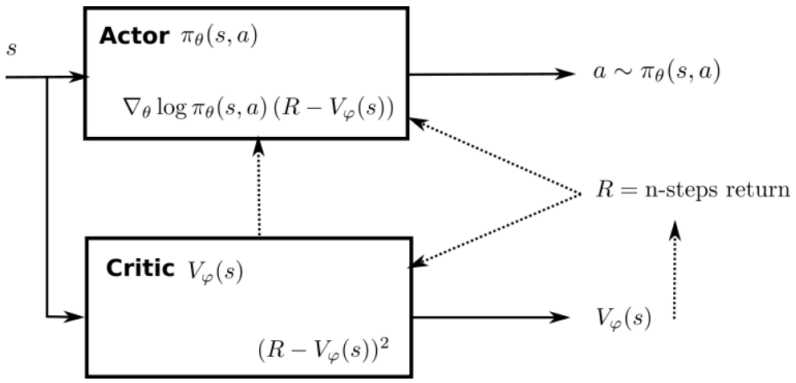


Figura 2.3: Esquema de estructura actor-critic de A2C. Imagen extraída de https://julien-vitay.net/deeprl/ActorCritic.html.

Un concepto importante es el de *advantage function*. Esta permite determinar si un estado o acción resulta mejor de lo esperado o no. En caso afirmativo, se "informa" a la red *actor* para que elija esta acción más frecuentemente, y al revés. Esta función se formaliza como sigue:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim p^\pi, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) A_\varphi(s,a)] \tag{2.18}$$

Aquí, $A_\varphi(s,a)$ refiere a la estimación de la ventaja (*advantage estimate*) y trata de estimar la ventaja real. Para ello, se usan métodos clásicos como Monte Carlo o *temporal difference*, aunque habitualmente se opta por *n-step advantage estimate*, que combina ambos:

$$\begin{aligned} A_\varphi(s,a) &= R(s_t,a_t) - V_\varphi(s_t) \\ &= \sum_{k=0}^{n-1} \gamma^k r_{t+k+1} + \gamma^n V_\varphi(s_{t+n+1}) - V_\varphi(s_t) \end{aligned} \tag{2.19}$$

El algoritmo de A2C es el siguiente:

Algoritmo 4: Advantage Actor-Critic (A2C)

Inicializar *actor* π_θ y *critic* V_φ con pesos aleatorios
Observar el estado inicial s_0
para cada *instante de tiempo* t **hacer**
  Inicializar un minibatch vacío
  para cada $k \in [0, n]$ *siendo* k *un episodio* **hacer**
    Seleccionar una acción a_k usando el *actor* π_θ
    Ejecutar la acción a_k y observar el siguiente estado s_{k+1} y recompensa r_{k+1}
    Almacenar $(s_k, a_k, r_{k+1}$ en el minibatch del episodio)
  fin
  si s_n *no es terminal* **entonces**
    $R = V_\varphi(s_n)$
  en otro caso
    $R = 0$
  fin
  Restablecer el gradiente $d\theta$ y $d\varphi$ a 0
  para cada $k \in [n-1, 0]$ *(Se itera hacia atrás en el episodio)* **hacer**
    Se actualiza la suma con descuento de recompensas: $R = r_k + \gamma R$
    Se acumula el gradiente de la política: $d\theta \leftarrow d\theta + \nabla_\theta \log \pi_\theta(s_k, a_k)(R - V_\varphi(s_k))$ Se acumula el gradiente del *critic*: $d\varphi \leftarrow d\varphi + \nabla_\varphi (R - V_\varphi(s_k))^2$
  fin
  Se actualizan *actor* y *critic* aplicando gradiente descendente sobre los acumulados: $\theta \leftarrow \theta + \eta\, d\theta \quad \varphi \leftarrow \varphi + \eta\, d\varphi$
fin

Cabe mencionar que A2C es una versión síncrona de un algoritmo conocido como *asynchronous advantage actor-critic*, abreviado A3C [13]. Estos métodos comparten una estrategia parecida. Existen varios nodos que se entrenan simultáneamente en entornos iguales, desde un mismo punto de partida. Así, se inicializan igualmente la política y la función de valor. Durante el entrenamiento, se sincronizan todos los nodos con cierta frecuencia. Las principales diferencias entre ambos métodos son las siguientes[5]:

- En el caso de A2C, cada nodo debe calcular un gradiente y, tras finalizar la fase de entrenamiento, se actualiza de manera asíncrona la red global, sumando todos estos gradientes. Tras cada actualización, cada

[5]Nótese que las figuras 2.4 y 2.5 provienen de [14].

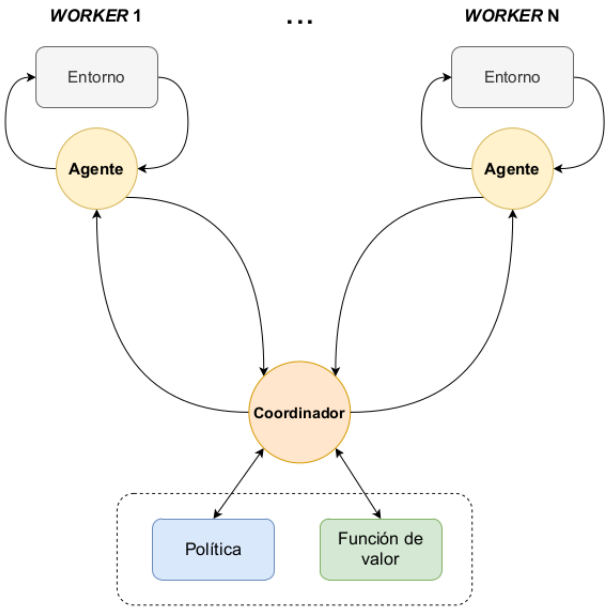


Figura 2.4: Esquema de A2C.

nodo comienza a partir del estado de la red global. De esta forma, un solo nodo aprende la experiencia de todos los actores (véase Figura 2.4).

- En este sentido, A3C es diferente. Cada nodo actualiza la red global de forma parcial, asíncrona e independiente, actualizando. Este entrenamiento es algo menos robusto ya que, tras cada actualización, cada nodo puede partir de una política y función valor diferente. Se ilustra este esquema en la Figura 2.5.

Es por esto que A2C presenta un mejor rendimiento que A3C, además de haber demostrado empíricamente una convergencia más rápida, según OpenAi.

2.4.2. DQN

DQN es la abreviatura de ***Deep Q-Network***, el primer algoritmo relevante de DRL que, a día de hoy, sigue siendo bastante utilizado por su buen desempeño [15]. Integra el uso de **redes neuronales** en el paradigma de RL. Se utilizan para tratar de aproximar funciones no lineales complejas en un tiempo razonable, en concreto, la función óptima de acción-valor:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...|s_t = s, a_t = a, \pi] \tag{2.20}$$

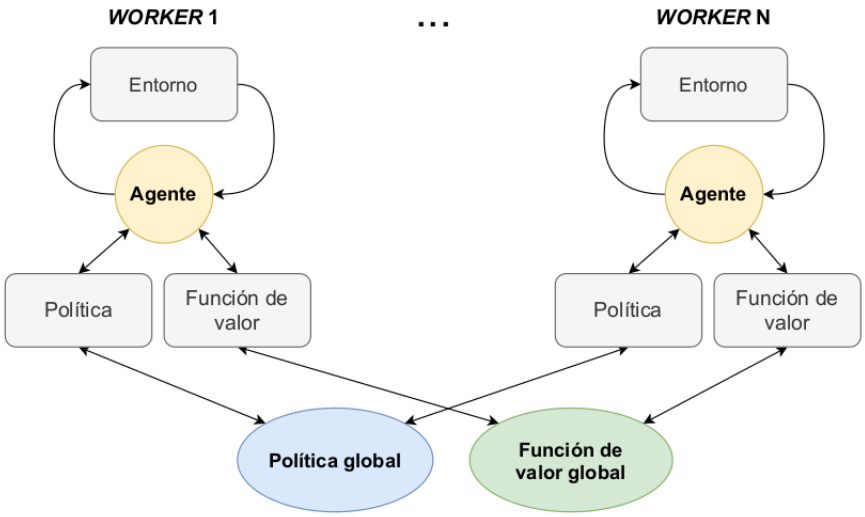


Figura 2.5: Esquema de A3C.

Este algoritmo está fundamentado en el método clásico *Q-learning*. Para aproximar la función $Q^*(s,a)$, utiliza la **ecuación de Bellman**, resultando:

$$Q^*(s,a) = \mathbb{E}\big[r + \gamma \max_{a'} Q^*(s',a') \mid s,a\big] \tag{2.21}$$

Se aplica la función de pérdida de **error cuadrático medio** (*mean squared error*, abreviado MSE). Utlizando esta expresión, se entrena la red minimizando el error de forma iterativa. Adicionalmente, se sustituyen los valores objetivo de $r + \gamma \max_{a'} Q^*(s',a')$ por una estimación, utilizando estados anteriores de la red, es decir, los pesos que tenía en varias iteraciones atrás. Se define así la **función de pérdida** $L_i(\Sigma_i)$ como sigue:

$$\begin{aligned} L_i(\theta_i) &= \mathbb{E}_{s,a,r}\big[(\mathbb{E}_{s'}[y|s,a] - Q(s,a;\theta_i))^2\big] \\ &= \mathbb{E}_{s,a,r,s'}\big[(y - Q(s,a;\theta_i))^2\big] + \mathbb{E}_{s,a,r}[\mathbb{V}_{s'}[y]] \end{aligned} \tag{2.22}$$

Se utiliza el método del **gradiente descendente estocástico** (SGD) para aproximar los pesos de la red neuronal. Si se deriva la función de pérdida en función a los pesos, se obtiene el siguiente gradiente:

$$\nabla_{\theta_i} L(\theta_i) = \mathbb{E}_{s,a,r,s'}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)\nabla_{\theta_i} Q(s,a;\theta_i)\right] \tag{2.23}$$

Cabe mencionar que este método utiliza **dos redes neuronales** a la vez, una para la **predicción**, conocida como red principal o *prediction network*, y otra red objetivo, o *target network*. La red de predicción, se utiliza

para calcular la función de valor del estado y acción actuales. Por su parte, la red objetivo trata de aproximar el valor objetivo. La razón por la que se utilizan dos redes es porque, en caso de utilizar una sola, el valor de Q predicho y el objetivo se desplazan en la misma dirección. Si se usan dos redes independientes, se pueden "congelar" los pesos de la red objetivo y actualizarlos únicamente cada n iteraciones. De esta forma, el valor objetivo de Q se calcula de forma más precisa.

Otro aspecto relevante de DQN es que utiliza ***experience replay*** [16]. Esta técnica guarda la información recogida a lo largo del entrenamiento, y la utiliza durante la fase de predicción.

Por último, DQN se utiliza casi exclusivamente en espacios de acciones discretos, ya que en el caso continuo no se puede elegir una acción óptima, ya que hay infinitas opciones, durante el cálculo del error. Aún así, existen algunas adaptaciones que aplican discretización y permiten trabajar en espacios de acciones continuos.

El algoritmo de DQN tiene la siguiente estructura:

Algoritmo 5: Deep Q-Network (DQN)

Entrada: - C: frecuencia de sincronización de la red objetivo

---

Inicializar memoria de repetición D
Inicializar la función acción-valor Q con pesos aleatorios θ
Inicializar la función acción-valor objetivo $\hat{Q}$ con pesos aleatorios $\theta^- = \theta$
para cada *episodio* **hacer**
  Inicializar la secuencia $s_1 = x_1$ y la secuencia preprocesada $\phi_1 = \phi(s_1)$
  para cada *timestep* **hacer**
    Seleccionar a_t con criterio ε-greedy bajo $Q(\phi(s_t), a; \theta)$
    Ejecutar acción a_t y observar recompensa r_t
    Establecer $s_{t+1} = s_t, a_t$ y preprocesar $\phi_{t+1} = \phi(s_{t+1})$
    Almacenar transición $(\phi_t, a_t, r_t, \phi_{t+1})$ en D
    Seleccionar un minibatch aleatorio de transiciones de D
    Establecer $y_j = r_j$ si el episodio termina en j+1 o $y_j = r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-)$ en otro caso
    Aplicar el paso de descenso de gradiente en $(y_j - Q(\phi_j, a_j; \theta))^2$
    Actualizar $\hat{Q} = Q$ cada C pasos
  fin
fin

Cabe destacar algunas de las variantes de este algoritmo, entre las que destacan: *Double Q-learning* [17], *Dueling DQN* [18], *Multi-step learning DQN* [19] o *Prioritized replay* [20].

2.4.3. PPO

El tercer algoritmo que se presenta se denomina ***Proximal Policy Optimization***, más conocido como PPO. Este algoritmo trata de imitar en eficiencia y calidad al método **TRPO**, aunque utilizando únicamente optimización de primer orden [21]. PPO es un método de aprendizaje *online* y *on-policy*, basado en el gradiente de la política. Al ser *on-policy*, se utiliza una misma política, que se irá evaluando y mejorando iterativamente. Así, se trata de un algoritmo con una estructura más sencilla y permite entrenar la red en tiempos mucho menos que algoritmos *off-policy*. Aún así, esto tiene un coste, ya que esta reducción en el tiempo de entrenamiento se puede traducir en un ligero empeoramiento de los resultados.

La estructura del algoritmo es la siguiente. De forma iterativa, primero acumula información en lotes en base a la experiencia y después, con esta información, optimiza la política utilizada. Esto provoca divergencia entre la política original y la política actualizada, para lo que se propone aplicar TRPO (*Trust Region Policy Optimization*) y el coeficiente de divergencia KL (*Kullback-Leiber*). Esto limita las actualizaciones de la política, manteniendo dichas modificaciones en una región de confianza[22].

La función de pérdida a minimizar en Proximal Policy Optimization se expresa así:

$$L_{PG}(\theta) = \hat{\mathbb{E}}_t[\log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{2.24}$$

Aquí, $\hat{A}_t$ hace referencia a la función acción-ventaja. Esta estima la calidad de cada acción en función de la recompensa esperable asociada a ella y considerando que se sigue una política π dada. Dicha función se expresa así:

$$a_\pi(s,a) = q_\pi(s,a) - v_\pi(s) \tag{2.25}$$

La función objetivo de TRPO es la siguiente:

$$J_{TRPO}(\theta) = \mathbb{E}[r(\theta)\hat{A}_{\theta_{old}}(s,a)] \tag{2.26}$$

donde $r(\theta)$ es la relación:

$$r(\theta) = \frac{\pi_{\theta_{old}}(a|s)}{\pi_\theta(a|s)} \tag{2.27}$$

Para limitar las actualizaciones de la política y mantenerla dentro de la mencionada región de confianza, se utiliza un umbral de desviación máxima (ϵ) referente al ratio $r(\theta)$. De esta forma, resulta la función objetivo que utiliza PPO, formulada a continuación. Nótese que *clip* es una función encargada de truncar el valor del primer parámetro dentro del rango que indican el segundo y el tercer parámetro. El algoritmo PPO elige el valor mínimo entre ambos, logrando paliar así el problema de la divergencia que se explicaba anteriormente.

$$J_{CLIP}(\theta) = \mathbb{E}\Big[\min(r(\theta)\hat{A}_{\theta_{old}}(s,a), clip(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{\theta_{old}}(s,a)\Big] \tag{2.28}$$

2.4.4. SAC

Soft Actor Critic, también conocido como SAC, pertenece a los algoritmos de tipo *actor-critic*, como se puede intuir por su nombre. Además, es un método *off policy* que, al igual que DQN, se basa en el método clásico *Q-learning* [23]. Hace uso de políticas estocásticas, que trata de optimizar.

En este caso, se hace uso de una función objetivo modificada, que incluye el concepto de **entropía**. Se puede definir esta entropía como la impredictibilidad o aleatoriedad de una variable. Cuanta mayor entropía, más impredecible sera dicha variable. De esta forma, permite medir la aleatoriedad en el comportamiento del agente. El método SAC trata de maximizar esta entropía para resolver el conflicto entre exploración y explotación. De esta forma, utiliza la siguiente función objetivo:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))] \tag{2.29}$$

En el caso de SAC, intervienen tres redes neuronales. La primera aproxima la función estado-valor V en función de ψ. La segunda aproxima la función *soft Q* (Q(θ)), y la última estima $\pi(\phi)$ para la política. Haarnoja et al. recomiendan utilizar aproximadores distintos para V y Q, ya que empíricamente presenta una mejora en los resultados [23], aunque no es un aspecto obligatorio. El entrenamiento procede así:

- Para estimar V, se debe minimizar función de pérdida $J_V(\psi)$:

$$J_V(\psi) = \mathbb{E}_{s_t\sim\mathcal{D}}\left[\frac{1}{2}(V_\psi(s_t) - \mathbb{E}_{a_t\sim\pi_\phi}[Q_\theta(s_t,a_t) - \log\pi_\phi(a_t|s_t)])^2\right]$$

Así, queda la siguiente actualización:

$$\hat{\nabla}_\phi J_V(\psi) = \nabla_\psi V_\psi(s_t)(V_\psi(s_t) - Q_\theta(s_t, a_t) + \log \pi_\phi(a_t|s_t))$$

- La función Q se aproxima minimizando el error $J_Q(\theta)$:

$$J_Q(\theta) = \mathbb{E}_{s_t \sim \mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(s_t, a_t) - (r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}[V_\psi(s_{t+1})])\right)^2\right]$$

En este caso, la actualización es:

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(s_t, a_t)(Q_\theta(s_t, a_t) - r(s_t, a_t) - \gamma V_\psi(s_{t+1}))$$

- Para entrenar la red referente a la política, se usa la función de error[6]:

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}}\left[D_{KL}\left(\pi_\phi(\cdot\,|s_t)\middle\|\frac{\exp(Q_\theta(s_t, \cdot))}{Z_\theta(s_t)}\right)\right]$$

Con la actualización:

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \log \pi_\phi(a_t|s_t) + (\nabla_{a_t} \log \pi_\phi(a_t|s_t) - \nabla_{a_t} Q(s_t, a_t))\nabla_\phi f_\phi(\epsilon_t; s_t)$$

Los mismos creadores del algoritmo demostraron que, en la práctica, SAC superaba al resto de algoritmos del estado del arte en ese momento, realizando pruebas con distintos *benchmarks*, por lo que se trata de un método ciertamente potente.

El algoritmo se estructura de la siguiente manera:

[6] Aquí, D_{KL} refiere a la divergencia de KL (*Kullback-Leibler*) antes mencionada [24].

Algoritmo 6: Soft Actor Critic (SAC)

Entrada: - θ: parámetros iniciales de la política
- ϕ_1, ϕ_2: parámetros de la Q-función
- $\mathcal{D}$: *buffer* de repetición

Asignar a los parámetros objetivo el valor de los parámetros principales: $\theta_{obj} \leftarrow \theta, \phi_{obj,1} \leftarrow \phi_1, \phi_{obj,2} \leftarrow \phi_2$
repetir
  Observar el estado s y seleccionar la acción $a \sim \pi_\theta(\cdot|s)$
  Ejecutar a
  Observar el siguiente estado s', recompensa r, y si es estado terminal (flag d)
  Almacenar (s, a, r, s', d) en el búfer de repetición ⌈
  Si s' es terminal, se resetea el estado del entorno
  si *se debe actualizar* **entonces**
    para cada $j \in [0, \textit{número actualizaciones})$ **hacer**
      Extraer un batch de transiciones B aleatorio
      Calcular los objetivos para las Q-funciones: $y(r, s', d) = r + \gamma(1-d)\Big(min_{i=1,2} Q_{\phi_{obj,i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')\Big)$
      Actualizar las Q-funciones con un paso del gradiente descendente: $\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} (Q_{\phi_i}(s,a) - y(r,s',d))^2$ para $i = 1, 2$
      Actualizar la política con un paso del gradiente ascendente: $\nabla_\phi \frac{1}{|B|} \sum_{s \in B} (\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s)|s)$
      donde $\tilde{a}_\theta(s)$ es una muestra de $\pi_\theta(\cdot|s)$, que es diferenciable con respecto a θ usando el truco de la reparametrización.
      Actualizar las redes objetivo con: $\phi_{obj,i} \leftarrow \rho\phi_{obj,i} + (1-\rho)\phi_i \quad$ para $i = 1, 2$
    fin
mientras *no converja*

2.4.5. TD3

El último algoritmo a presentar es ***Twin Delayed DDPG***, o TD3. Se trata de una versión más sofisticada del método ***Deep Deterministic Policy Gradient*** [25]. Este segundo método se basa, a su vez, en *Q-learning*. Asimismo, se utilizan redes neuronales para aproximar una política y una Q-función de forma simultánea, haciendo uso de la ecuación de Bellman

anteriormente explicada. Además, implementa dos trucos para una mayor optimización: *replay buffer* y *target networks*.

Sin embargo, DDPG es demasiado sensible a los hiperparámetros, cuyo ajuste es muy costoso computacionalmente. Es ante esta debilidad donde surge TD3, un algoritmo *off-policy* diseñado para espacios de acciones continuos. Para una mayor optimización, se utilizan tres mejoras:

- **Doble Q-learning recortado** (*clipped double Q-learning*). Este "truco" consiste en aprender dos Q-funciones en lugar de una, como suele hacerse, y utilizar el mínimo Q-valor para calcular la función de pérdida.
- **Actualizaciones retrasadas de la política** (*delayed policy updates*). Consiste en reducir la frecuencia de actualización de la política y las redes objetivo. Concretamente, se debe actualizar menos frecuentemente que la Q-función: se recomienda actualizar la política cada dos actualizaciones de la Q-función [25].
- **Suavizado de la política objetivo** (*target policy smoothing*). Este trata de añadir ruido a la acción objetivo, de tal forma que se "suavizan" los Q-valores, para intentar que la política explote menos los errores de la Q-función.

Dicho esto, el algoritmo de TD3 queda así:

Algoritmo 7: Twin Delayed DDPG (TD3)

Entrada: - θ: parámetros iniciales de la política
- ϕ_1, ϕ_2: parámetros de la Q-función
- $\mathcal{D}$: *buffer* de repetición

Asignar a los parámetros objetivo el valor de los parámetros principales: $\theta_{obj} \leftarrow \theta, \phi_{obj,1} \leftarrow \phi_1, \phi_{obj,2} \leftarrow \phi_2$
repetir
  Observar el estado s y seleccionar la acción $a = clip(\mu_\theta(s) + \epsilon, a_{Low}, a_{High})$
  Ejecutar a
  Observar el siguiente estado s', recompensa r, y si es estado terminal (flag d)
  Almacenar (s, a, r, s', d) en el búfer de repetición ⌈
  Si s' es terminal, se resetea el estado del entorno
  si *se debe actualizar* **entonces**
    para cada $j \in [0,$ *número repeticiones*$)$ **hacer**
      Extraer un batch de transiciones B aleatorio
      Calcular las acciones objetivo $a'(s') = clip(\mu_{\theta_{obj}}(s') + clip(\epsilon, -c, c), a_{Low}, a_{High})$
      Calcular los objetivos $y(r, s', d) = r + \gamma(1-d)\min_{i=1,2} Q_{\phi_{obj,i}}(s', a'(s'))$
      Actualizar las Q-funciones con un paso del gradiente descendente: $\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d)\in B}(Q_{\phi_i}(s,a) - y(r,s',d))^2$ para $i = 1, 2$

      si *j mod policy_delay == 0* **entonces**
        Actualizar l política con un paso del gradiente ascendente usando: $\nabla_\theta \frac{1}{|B|} \sum_{s\in B} Q_{\phi_1}(s, \mu_\theta(s))$
        Actualizar las redes objetivo con:
        $\phi_{obj,i} \leftarrow \rho\phi_{obj,i} + (1-\rho)\phi_i$ para $i = 1, 2$
        $\theta_{obj} \leftarrow \rho\theta_{obj} + (1-\rho)\theta$
    fin
mientras *no converja*

2.5. Formulación del problema

Una vez se han explicado los conceptos fundamentales del aprendizaje por refuerzo y los algoritmos a utilizar, se puede proceder a formular el problema que se trata de resolver en este proyecto, consistente en crear un controlador automático capaz de controlar de forma eficiente la energía en un edificio, particularmente, los sistemas de climatización.

Primero, conviene definir el **conjunto de estados** que define el entorno. Para ello, deben indicarse cuáles son las variables medidas, pues cada combinación de sus posibles valores determinará un estado distinto. En este problema, las variables son la temperatura del aire, la humedad relativa, velocidad del viento y su dirección, radiación solar, ocupación del edificio, entre muchas otras. Todas ellas se listan y definen en el apartado titulado "Variables observadas" de la sección 5.2.3. Nótese que la mayoría de las variables tienen un dominio continuo, por lo que el espacio de acciones es también continuo.

En segundo lugar, también referente al entorno, se debe definir la **función de recompensa**. Esta se encarga de ayudar al agente con cierta realimentación tras cada acción que este ejecute. En este caso, el objetivo es minimizar el consumo energético y maximizar el confort térmico dentro del edificio. Ambos son objetivos contrarios, por lo que busca alcanzar un equilibrio óptimo. Esto se debe a que maximizar el confort térmico implica un mayor consumo energético y, por el contrario, para minimizar el consumo energético, se debería renunciar al confort térmico. Antes de definir la función de recompensa como tal, se debe explicar la forma en que se cuantifican el consumo energético y el confort térmico.

El **consumo energético** es una magnitud física sencilla de cuantificar. Basta con medir la energía que consumen los sistemas de climatización durante su funcionamiento.

No ocurre lo mismo con el confort, ya que no es una magnitud objetiva, sino que debe especificarse de qué forma se va a medir. Concretamente, se tendrá en cuenta el **confort térmico**, ya que los sistemas de climatización considerados sólo pueden influir en la temperatura, y no en otras magnitudes como la humedad o la concentración de CO_2, por ejemplo. Dicho esto, es importante definir un rango de temperaturas adecuado. Según [26, 27, 28], se considera que [23°C - 26°C] y [20°C - 23.5°C] son intervalos idóneos para verano e invierno, respectivamente. Se puede medir el confort de distintas formas. Por ejemplo, la distancia entre la temperatura actual y el punto medio del intervalo, o el cuadrado de esta distancia. Aquí, se utilizará la **distancia al intervalo de confort objetivo** [29], pudiendo formularse la

función de recompensa de la siguiente manera:

$$r(S_t, A_t) = w_t \cdot \lambda_c \cdot f(S_t, S_{target}) + (1 - w_t) \cdot \lambda_e \cdot coste(A_t) \tag{2.30}$$

La función para cuantificar el confort sería:

$$r(S_t, A_t) = w_t \cdot \lambda_c \cdot \sum_{i=1}^{z} ([T_t^i - \overline{T_t^i}]_+ + [T_t^i - \underline{T_t^i}]_+) + (1 - w_t) \cdot \lambda_e \cdot coste(A_t) \tag{2.31}$$

Tanto λ_c como λ_e, son factores de escala, utilizados para normalizar los términos de confort y consumo de energía, respectivamente.

Donde w_t es una variable que pondera la relevancia del confort térmico frente al consumo energético, y toma valores en el rango [0, 1]. Es importante destacar que este parámetro es dinámico, es decir, varía a lo largo del tiempo. Por ejemplo, podría variar según la ocupación del edificio, dando más importancia al ahorro energético cuando no hay nadie, y viceversa.

Además de esta función de recompensa "básica", se utilizarán algunas otras propuestas para comparar su rendimiento. Estas se definirán y explicarán en el apartado 5.3.4.

Por otro lado, se debe definir las posibles acciones que puede ejecutar el agente, es decir, su **conjunto de acciones**. En este caso, el agente puede intervenir en el ajuste de los denominados ***setpoints*** de los dispositivos de climatización del edificio. Estos *setpoints* son los umbrales de temperatura a partir de los cuales deben activarse o desactivarse los sistemas de calefacción y refrigeración. Por ejemplo, si la temperatura disminuye por debajo del *setpoint* de frío, es decir, la cota inferior, se activará el sistema de calefacción en cuestión. Así, el agente se encarga de definir estos umbrales de temperatura de forma óptima. Por tratarse de una variable numérica, se debe indicar si el dominio es discreto o continuo. Habitualmente, los sistemas de HVAC permiten ajustar estos umbrales con una cifra decimal, aunque existen otros con mayor precisión (dos cifras decimales). Se trata así de un dominio discreto, aunque en este problema se considerarán ambos casos para comprobar si existen diferencias sustanciales entre sí.

Existen otros elementos adicionales que se deben definir:

- **Agente**. En este caso, es el controlador de los sistemas de climatización del edificio en cuestión.

- **Entorno**. Formado por todo aquello que el agente no puede modificar arbitrariamente. Abarca elementos desde el edificio y sus materiales, hasta las condiciones meteorológicas, entre muchos otros.
- **Política**. La política óptima a aproximar definida en base a la función de recompensa anterior se formula:

$$\pi^* = \arg\min_{\pi_\theta} \sum_{t=1}^{T} w_t \cdot f(S_t, S_{target}) + (1 - w_t \cdot coste(A_t)) \tag{2.32}$$

- **Modelo del entorno**. En este problema no se dispone de él, ya que no se puede predecir sin ninguna incertidumbre el estado resultante tras aplicar cualquier acción. Esto se debe a la inmensidad de factores que intervienen en el problema, además de la posible presencia de ruido. De aquí surge la necesidad de usar métodos libres de modelo.

Cabe mencionar que se trata de un **problema continuado**, donde cada simulación se limitará a un año de duración.

Capítulo 3

Revisión literaria

Antes de comenzar a analizar el estado del arte, cabe destacar cómo ha aumentado el interés en este tema durante los últimos años. Y es que desde la última década, en especial desde 2019, el número de artículos publicados ha aumentado considerablemente y, hasta 2022, sigue sufriendo un crecimiento considerable (véase Figura 3.1). Esto se debe a que el desarrollo de algoritmos de aprendizaje por refuerzo profundo durante el último lustro ha permitido aplicarlos al problema del control energético en edificios y obtener resultados prometedores.

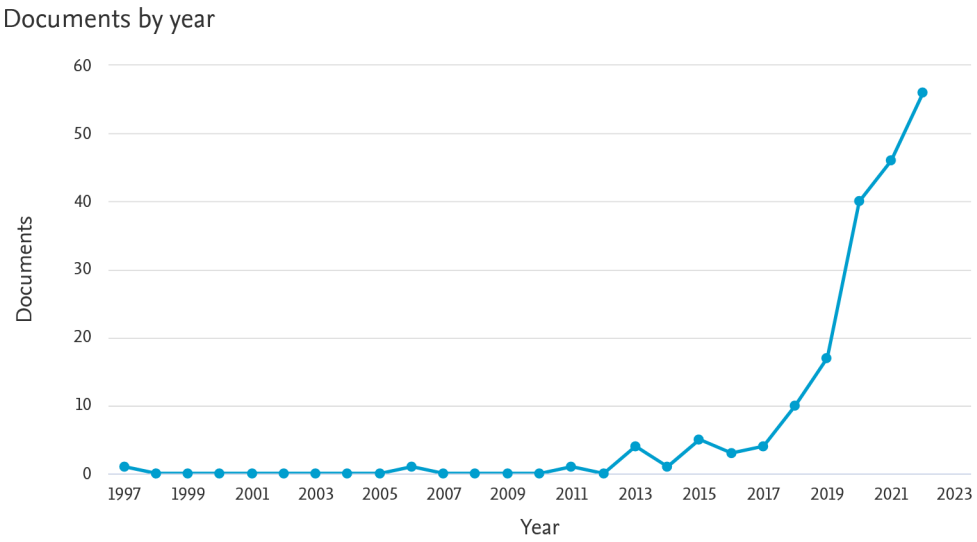


Figura 3.1: Número de artículos sobre RL (ó DRL) y HVAC en Scopus.

Sin embargo, y a pesar de lo que pueda sugerir la figura anterior, la aplicación del aprendizaje por refuerzo en control energético no es una solución reciente. De hecho, la primera vez que se utiliza aprendizaje por refuerzo y

aprendizaje profundo para abarcar este problema fue en 1997, cuando W. Anderson demuestra las posibilidades que ofrecía combinar una red neuronal con un controlador PI [30]. En 1998, Mozer publica su estudio *Neural Network House*, donde diseña un sistema para controlar los dispositivos HVAC de un edificio, la iluminación y la caldera [31].

Especialmente a partir de 2013, aparecen **numerosas publicaciones** que abarcan este problema. Muchas de ellas son en edificios residenciales o comerciales, donde mantener el confort térmico es un punto clave [32, 33, 34, 35, 36, 37, 38, 39, 40]. También se consideran edificios con **múltiples habitaciones** [41, 42] o de mayores dimensiones como almacenes [43]. Incluso existen algunos estudios orientados a lo que se conoce como **ciudades inteligentes**.

Por su parte, abordar el problema del control energético tratando de diseñar controladores automáticos para los **sistemas de climatización** es una línea de investigación que ya han abierto autores como Ruelens [36], Urieli [38].

Para la resolución de este problema, se han utilizado numerosas técnicas distintas. Desde la incorporación de lógica difusa [44, 45], hasta el uso de técnicas basadas en ***set-back*** [41, 38, 36], que permiten priorizar el ahorro energético cuando el edificio o estancia están desocupados. También se han diseñado **sistemas multiagente** [46] para controlar los dispositivos de forma descentralizada considerando la tarifa variable de la electricidad.

Algunos de los **resultados** numéricos obtenidos en estos y otros estudios son los siguientes:

- En [47] se alcanza un ahorro semanal de aproximadamente el 22 %, con su *DR-aware RL controller* en entornos con grandes fluctuaciones en la demanda energética (una variabilidad de hasta 50 %).
- Por su parte, Brandi *et al.* consiguen un ahorro energético que oscila entre el 5 % y el 12 %, con un modelo de gran robustez [34].
- En [41], logra una reducción del 14 % de la energía destinada a la climatización, y una mejora del 11 % en confort térmico.
- Deng trata de resolver el problema de control HVAC en un entorno no estacionario, consiguiendo un ahorro de hasta el 13 % y una mejora en el confort del 9 % [35].
- En comparación con un RBC y un sistema PID, Yuan consigue una mejora del 4.7 % al 7.7 % [48].

- En [32], Li et. logra un ahorro del 9.17 % en comparación a un RBC que mantiene un ajuste de *setpoints* constante, consiguiendo reducir el pico de demanda.
- Un estudio a mayor escala es el de Yu, quien con su algoritmo obtiene un ahorro de entre el 56 % y el 75 % en un edificio donde se distinguen 30 zonas diferentes [49].
- En [50], se utiliza un sistema multiagente para controlar varios edificios, demostrando su superioridad frente a controladores centralizados, logrando un ahorro energético de entre el 5 % y el 12 %, una mejora en el confort térmico de entre el 15 % y el 30 %.
- Por último, en [51] se utiliza un modelo de DRL que alcanza un ahorro del 15 % respecto al otro mejor modelo del estado del arte cuando este fue publicado, logrando mejorar bastante también en términos de confort.

Cabe mencionar que la mayoría de los modelos de estos estudios solo se prueban en un único entorno. Resultan muy convenientes las pruebas de robustez, como es el caso de [34], donde se aplican los modelos resultantes en cuatro escenarios con características diferentes.

Ante estos resultados, es habitual preguntarse si son comparables unos estudios con otros. La respuesta ante esta pregunta es negativa, ya que los experimentos no se suelen realizar en igualdad de condiciones, ni siquiera en los mismos simuladores. Tampoco tienen por qué coincidir los modelos de referencia (*baselines*), por lo que los resultados son aún menos comparables. Además, varían los algoritmos, edificios, funciones de pérdida (formas en las que se mide el consumo energético o el confort térmico), *software* utilizado, etcétera. Autores como Vázquez [46] y Azuatalam [47] denuncian esta **falta de comparabilidad**, y proponen un *framework* en el que desarrollar y comparar distintos modelos de aprendizaje por refuerzo profundo para el problema tratado. Otro de los *frameworks* a destacar referentes a este campo es ***Sinergym***, un *software* diseñado por miembros de la Universidad de Granada cuyo objetivo es permitir entrenar, evaluar y comparar modelos en un mismo simulador, proponiendo unos estándares para permitir comparar distintos controladores. Esta herramienta es la utilizada para el desarrollo de este proyecto (se explica con mayor detenimiento en el apartado 4.2).

Otro aspecto a destacar es la **ausencia de estudios acerca de la función de recompensa**. No existe ninguna publicación en la que se analice su impacto en el agente, o se sugieran propuestas innovadoras. Esto se debe en parte a la mencionada falta de estándares en este campo. Aún así, un estudio de este tipo resulta necesario, pues se encuentran algunos artículos

cuyos agentes han sido entrenados con una función de recompensa poco adecuada y, por tanto, bastante mejorable. Uno de los propósitos principales de este proyecto es analizar la influencia que tienen distintas recompensas propuestas, con el objetivo de aportar cierta información a la comunidad científica a la hora de su diseño.

Capítulo 4

Metodología

Este capítulo se dedica a explicar la metodología que se ha seguido a la hora de abordar el problema. Adicionalmente, se exponen algunos aspectos referentes al *software* y *hardware* utilizados.

4.1. Enfoque del problema

Primero, resulta conveniente recordar el problema afrontado en este proyecto. Consiste en la **optimización del control energético en edificios**, con el objetivo de minimizar el consumo energético mientras se mantienen las restricciones de confort. Se centra la atención en los sistemas de **climatización**, por lo que la tarea del controlador consiste en ajustar los *setpoints* de forma óptima, para reducir el consumo y mantener una temperatura adecuada en el edificio.

Para ello, se entrenan modelos automáticos utilizando algoritmos de aprendizaje por refuerzo profundo. Un componente clave de estos modelos es la función de recompensa, encargada de indicar al agente el beneficio o perjuicio inmediato de realizar cada acción. El principal objetivo de este proyecto, además de resolver el problema de control en sí, radica en **estudiar la influencia de la función de recompensa** sobre el entrenamiento del agente.

Para ello, se diseñan **cuatro funciones de recompensa** de distinta naturaleza: una lineal, otra que considera los niveles de ocupación, y otras dos con una penalización más dura cuando la temperatura en el interior no está dentro de los intervalos deseados. Con cada una de ellas, se entrenan distintos agentes y posteriormente se analiza su desempeño, con el propósito de estudiar si algunas resultan más convenientes o preferibles que otras. Además, durante este análisis se utilizan **varias métricas** de evaluación,

que permiten discernir si una función de recompensa es más útil para primar el ahorro energético y/o el confort térmico.

Con el fin de evitar posibles sesgos y realizar un estudio completo y objetivo, se propone realizar una experimentación relativamente extensa. Se plantea utilizar **cinco algoritmos** de diferentes tipos, además de un **controlador basado en reglas** que se tomará como *baseline*. Por otro lado, se entrenará cada algoritmo en **tres escenarios climatológicos** distintos. Por último, por cada combinación algoritmo-clima se entrena un agente con cada una de las funciones de recompensa propuestas. Resultan así **60 modelos** en total (3 climas $\times$ 5 algoritmos $\times$ 4 funciones de recompensa).

De esta manera, se tiene una visión completa que permite estudiar de qué forma influye cada función de recompensa en el comportamiento de los agentes entrenados. Todos los detalles de la experimentación propuesta se desarrollan en el capítulo 5. A continuación se muestra un diagrama que ilustra la estructura de estos experimentos:

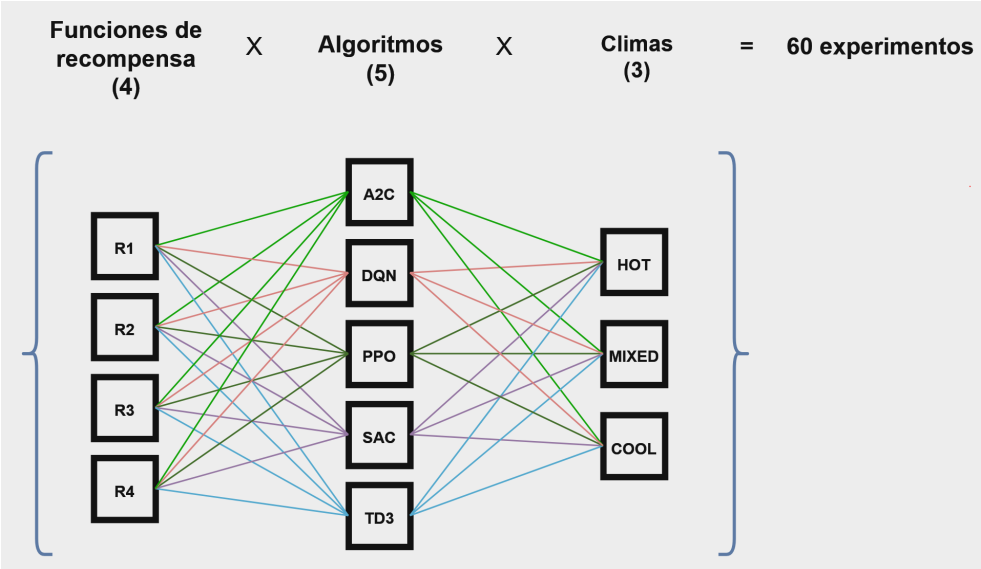


Figura 4.1: Esquema sobre la experimentación propuesta.

4.2. *Software*

En esta sección, se mencionan algunos de los aspectos referentes a la componente *software* de este proyecto. Sin embargo, antes de comenzar, conviene mencionar que este es un trabajo de investigación, donde la parte de desarrollo *software* apenas es relevante, pues el verdadero interés radica en la experimentación realizada y el análisis de resultados. En cualquier ca-

so, el código de Sinergym utilizado y sus pertinentes modificaciones se alojan en este repositorio de Github para que pueda consultarse si el lector lo desea.

La herramienta fundamental en esta investigación es **Sinergym** [52]. Se trata de un *software open source* en continuo desarrollo, por parte del grupo SAIL, formado por investigadores de la Universidad de Granada. Esta herramienta, basada en EnergyPlus, permite **simular edificios** y su dinámica con un alto grado de realismo en un entorno virtual, en los que se pueden crear, entrenar y evaluar **modelos de DRL**. Este proyecto busca crear unos **estándares**, en pos de la futura comparabilidad de modelos en igualdad de condiciones. Para ello, cumple con las directrices indicadas en la guía de Wöfle et al. [53], en la que se determina cómo deben ser diseñados los entornos de *benchmarking* para los problemas de control energético en edificios. En esta publicación se critica, por un lado, la baja o nula generalización de la mayoría de los modelos del estado del arte o la literatura en general, ya que son únicamente aplicables a un entorno o edificio específico. Por otro lado, se denuncia la consecuente dificultad de adaptarlos a otros casos, por lo que son soluciones bastante limitadas. Sinergym trata de dar solución a estos problemas, ya que incluye numerosos edificios en los que probar y comparar los agentes que se creen. Por ejemplo, un agente puede entrenarse con un clima y evaluarse con otro, o incluso ser adaptado para funcionar en otro edificio.

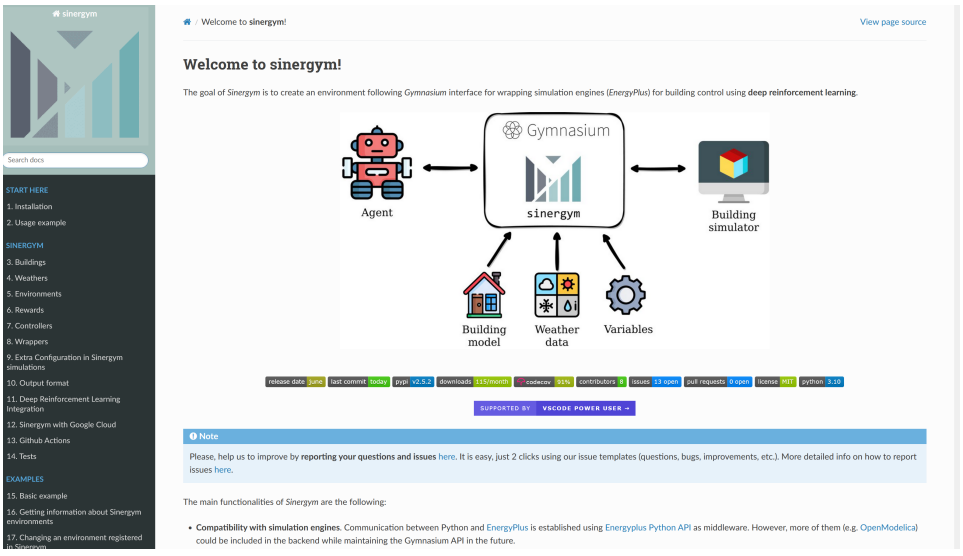


Figura 4.2: Documentación de Sinergym.

Como se decía, se trata de un proyecto en actual desarrollo. En este caso, no se ha contribuido apenas de forma directa, sino más bien probando la

herramienta y detectando algunos errores o *bugs*. Al encontrarlos, en lugar de corregirlos, se ha informado al principal desarrollador de este *software* para su corrección, sugiriendo en varias ocasiones alguna posible solución. Una vez se han corregido los fallos encontrados, se han realizado algunas ejecuciones de prueba en el ordenador del usuario, durante las que se ha observado que, considerando la batería de experimentos a realizar, era necesario acudir a un **entorno remoto de cómputo masivo**, dados los altos periodos de ejecución que se requieren.

Así, para entrenar y evaluar los modelos planteados, se ha recurrido a **AutoCloud**, una herramienta privativa diseñada por el mismo equipo, que facilita la interacción entre el usuario y la plataforma de ***Google Cloud***, cuando la tarea consiste en entrenar modelos de aprendizaje por refuerzo. Subiendo un contenedor con los cambios necesarios y la configuración deseada de Sinergym, se ha podido usar esta herramienta para la automatización en la ejecución de experimentos ejecutados en máquinas remotas. De nuevo, se ha contribuido de forma directa e indirecta en la detección y corrección de errores (véase Figura 4.3).

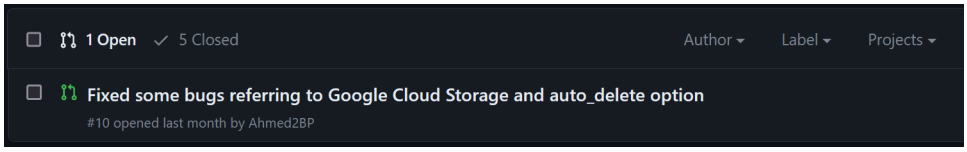


Figura 4.3: Ejemplo de *pull request* al repositorio de AutoCloud.

Adicionalmente, se han utilizado otras herramientas *software* bastante útiles para el almacenamiento y la monitorización de los entrenamientos y evaluaciones, entre las que figuran ***Weights and Biases*** (también conocido como WandB), Tensorboard y Goodle Cloud Storage. Su utilización ha resultado fundamental a la hora de mantener una buena organización en los experimentos. De hecho, todos los resultados obtenidos se pueden encontrar en este repositorio de WandB: `https://wandb.ai/ahmed2bp/projects`.

4.3. *Hardware*

Respecto al *hardware* utilizado para este proyecto, cabe diferenciar la componente referente a la parte "local" y la ejecución en remoto. Para las pruebas individuales en local, se ha utilizado un ordenador ASUS ROG Strix G15 G513RM. Este portátil tiene un procesador AMD Ryzen 7 6800H, 16 GB de memoria RAM y una GPU Nvidia GeForce RTX 3060.

Por su parte, para la realización de la batería de experimentos se han utilizado servicios de ***cloud computing***. Concretamente, se ha optado por

usar la herramienta de ***Google Cloud Platform***. En concreto, las máquinas utilizadas son *n2-highcpu-8*, equipadas con un procesador *Intel Cascade Lake* de 2.2 GHz, 8 vCPU y 8192 MiB de RAM[7]. Esto ha permitido lanzar los experimentos de diez en diez, permitiendo paralelismo y agilizando su ejecución.

[7]Si se desea más información sobre estas máquinas, consúltese esta web.

Capítulo 5

Experimentación

En este capítulo, se detallan los aspectos más relevantes de la experimentación realizada. Se comienza hablando de la herramienta y las posibilidades que ofrece en su configuración, siguiendo con los posibles escenarios y variables a considerar. Por último, se concretan los experimentos realizados para llevar a cabo la investigación propuesta.

5.1. Configuración

Como se ha mencionado en el capítulo anterior, la herramienta utilizada para las simulaciones es Sinergym. Este *software* permite crear entornos en los que simular distintos edificios y entrenar agentes utilizando algoritmos de DRL. Sinergym ofrece un amplio abanico de posibilidades en su configuración. Los aspectos más relevantes que se pueden modificar se listan a continuación.

El **espacio de estados** viene determinado por las variables observadas. Esta herramienta permite decidir qué variables serán consideradas por el agente durante su entrenamiento, pudiendo descartar aquellas que no se crean relevantes, o con el mero objetivo de hacer un modelo más sencillo o "liviano".

La **función de recompensa** es otro aspecto que puede ser modificado. Aunque se incluyen algunos ejemplos, no hay ninguna restricción a la hora de diseñar una función a partir de las variables observadas del entorno. Esta es una de las razones por la que se ha optado por este *software*, ya que para este proyecto es necesario diseñar distintas funciones para su posterior comparación.

A la hora de elegir un **agente**, también existen diversas opciones, ya que se puede elegir entre los distintos algoritmos implementados en *Stable Ba-*

selines 3, si se desea entrenar un controlador basado en DRL. Por su parte, también existe un controlador basado en reglas (RBC), cuyo comportamiento también puede ser modificado.

Otro aspecto a ajustable por el usuario es el **espacio de acciones** de dicho agente. En este caso, como solo se consideran los sistemas de climatización, las posibles acciones son los *setpoints* de estos dispositivos. Se pueden utilizar tanto espacios de acciones discretos como continuos. En caso de que el edificio incluyese otro tipo de dispositivos, como persianas inteligentes o sistemas de ventilación, también podrían ser controlados.

Se pueden modificar también otros aspectos referentes a la simulación como son la duración de cada simulación o la granularidad (número de posibles pasos o *timesteps* por hora) de estos.

5.2. Escenarios y variables

Existen muchos posibles escenarios a simular, los cuales surgen de las numerosas combinaciones entre edificios y climas.

5.2.1. Edificios

En el momento en que se redacta esta memoria, existen hasta seis posibles edificios entre los que elegir. Cada uno tiene sus propias característica: un centro de procesamiento de datos (CPD, o *data center*), un edificio sencillo de una planta con cinco zonas, un almacén, un edificio de oficinas de tres plantas, una pequeña tienda y otro edificio de oficinas de nueve plantas. Se pueden encontrar más detalles en la documentación.

Para el desarrollo de este proyecto, se ha utilizado el segundo, llamado **edificio de 5 zonas**. Esta decisión se debe a que es un entorno suficientemente complejo con un gran flujo de personas, un aspecto muy útil para la investigación que se desea desarrollar. El resto de edificios no son buenas opciones ya que o bien son demasiado grandes, y el entrenamiento de agentes tomaría demasiado tiempo, o no son edificios residenciales o frecuentados por personas, y el uso de dispositivos de climatización está orientado a mantenimiento (como en el caso del CPD), en lugar de al confort de las personas.

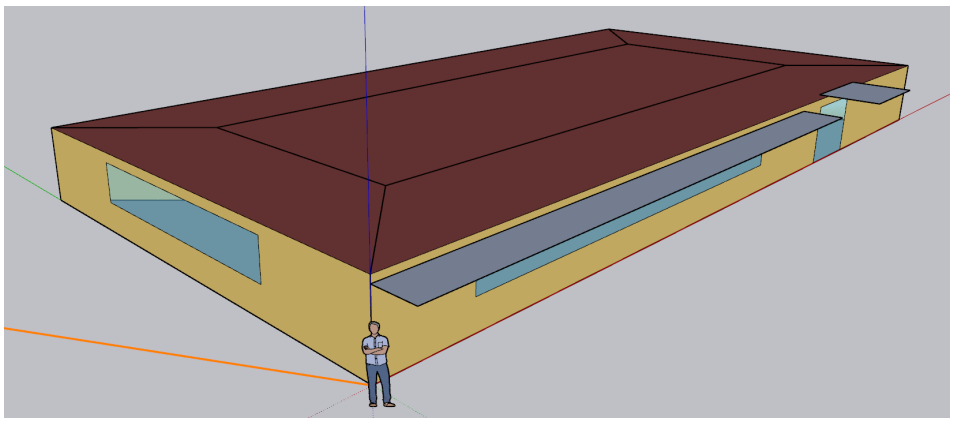


Figura 5.1: Representación gráfica del edificio de 5 zonas.

Cabe destacar que en el edificio elegido, se trata de controlar la temperatura únicamente en la estancia central, haciendo uso de un sistema de volumen de aire variable (VAV).

5.2.2. Climas

Como se ha comentado, la climatología es otro aspecto ajustable en Sinergym. Por defecto, incluye tres tipos de climas para la mayoría de los edificios: cálido, mixto (o templado) y fresco (o frío).

- El primero es un **clima cálido** y seco. Basado en el histórico de un área de Arizona, con una temperatura anual promedio de 21.7°C y una humedad relativa de 34.9 %.
- El **clima fresco** se basa en la meteorología registrada en un aeropuerto de Washington, cercano al mar. Así, mantiene una temperatura de 9.3°C y una alta humedad relativa, concretamente de un 81.1 %.
- El **clima mixto** representa un punto más intermedio. Corresponde a Nueva York, y presenta una temperatura promedio de 12.6°C y un 68.5 % de humedad relativa.

Con el objetivo de realizar un estudio más completo, se han utilizado los tres climas. De esta forma, se podrá analizar si existen diferencias sustanciales entre ellos, o si alguno resulta más difícil de tratar que otro.

Cabe mencionar que cada clima presenta una variante estocástica y otra determinista. Esto permite añadir o no ruido en la dinámica del entorno, respectivamente. Para el desarrollo de los experimentos, se ha optado por la **variante estocástica**, ya que esto suele generar modelos ligeramente más

robustos, pues no todos los episodios son iguales. Nótese que la inclusión de este ruido en las simulaciones no impide su reproducibilidad, ya que se puede incluir una semilla para cada ejecución.

5.2.3. Variables observadas

Como se ha explicado, el espacio de estados viene definido por las variables observadas del entorno. Estas aportan al agente la información que debe considerar a la hora de ejecutar cada acción. Las variables que se miden en los edificios pueden variar. En el caso del edificio de 5 zonas, figuran las siguientes:

- **Temperatura del aire en el interior** (*Zone Air Temperature*).
- **Temperatura seca del aire en el exterior** (*Site Outdoor Air Drybulb Temperature*).
- **Velocidad del viento** (*Site Wind Speed*).
- **Dirección del viento** (*Site Wind Direction*).
- **Humedad relativa del aire en el interior** (*Zone Air Relative Humidity*).
- **Humedad relativa del aire en el exterior** (*Site Outdoor Air Relative Humidity*).
- **Tasa de radiación solar directa por área** (*Site Direct Solar Radiation Rate per Area*).
- **Tasa de radiación solar difusa por área** (*Site Diffuse Solar Radiation Rate per Area*).
- **Número de ocupantes de la zona** (*Zone People Occupant Count*).
- **Valor del confort térmico de acuerdo al ropaje** (*Zone Thermal Comfort Clothing Value*).
- **Confort térmico según la temperatura radiante media** (*Zone Thermal Comfort Mean Radiant Temperature*).
- **Confort térmico de acuerdo al modelo Fanger** (*People Air Temperature*).
- **Confort térmico de acuerdo al modelo Fanger PDD** (*Zone Thermal Comfort Fanger Model PPD*).
- **Demanda energética total del sistema de climatización** (*Facility Total HVAC Electricity Demand Rate*).

- **Temperatura de activación (*setpoint*) para calefacción** (*Zone Thermostat Heating Setpoint Temperature*).
- **Temperatura de activación (*setpoint*) para refrigeración** (*Zone Thermostat Cooling Setpoint Temperature*).
- **Año actual** (*Current year*).
- **Mes actual** (*Current month*).
- **Día actual** (*Current day*).
- **Hora actual** (*Current time*).

Por último, es importante puntualizar que la herramienta permite ignorar algunas de estas variables. Sin embargo, en este proyecto se han considerado todas ellas, para que el modelo tenga la mayor cantidad de información posible, y tratar de evitar eliminar alguna variable útil durante la toma de decisiones.

5.3. Experimentos planteados

En este apartado, se van a detallar las condiciones y los aspectos más relevantes sobre la experimentación realizada. Cabe recordar que toda esta experimentación va orientada principalmente al estudio de la influencia de la función de recompensa en la resolución del problema de control energético en edificios, utilizando aprendizaje por refuerzo. También se ha planteado de tal manera que se puedan estudiar otros aspectos, como el rendimiento de distintos tipos de algoritmos, o la dificultad de distintos climas, entre otros. Las conclusiones a las que se llega tras analizar los resultados se exponen en el último capítulo.

5.3.1. Algunos aspectos básicos

Antes de entrar en más detalle, conviene analizar algunos aspectos básicos referentes al propio entrenamiento y evaluación de los agentes.

Por un lado, el tiempo de simulación de cada **entrenamiento** es de **20 episodios**. Esto se debe a que en [4], se demuestra que esta duración es más que suficiente para la convergencia de los algoritmos utilizados (véase Figura 5.2). Considerando que cada episodio tiene una duración de un año, por cada entrenamiento se simulan 20 años de aprendizaje para cada modelo. Respecto a la granularidad de las simulaciones, cabe indicar que transcurre **un *timestep* cada 15 minutos**, por lo que el agente puede ejecutar una

acción con esta frecuencia. Esto resultaría en 35040 pasos por episodio.

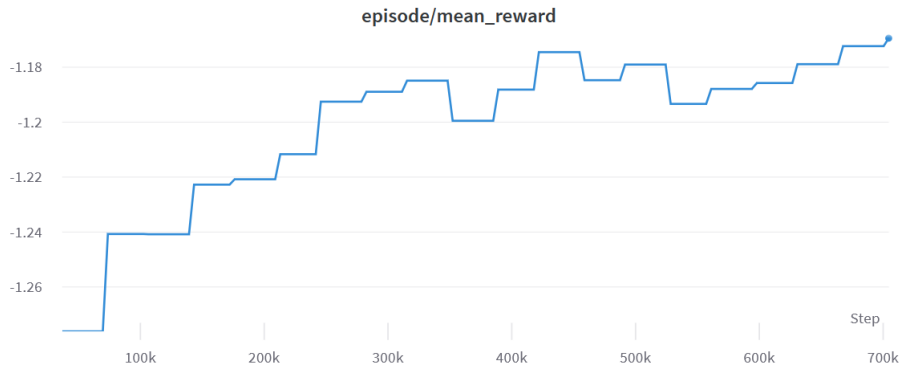


Figura 5.2: Ejemplo de convergencia de un agente durante su entrenamiento.

Por otro lado, en la fase de **validación**, se consideran **5 episodios**. Para medir el rendimiento, se calcula un promedio de las métricas obtenidas en todos ellos, pues se considera un tiempo suficientemente representativo para estimar el desempeño del agente.

Por último, es importante mencionar que no existe una fase de **ajuste de hiperparámetros**. Esto se debe a la elevada carga computacional que esto supondría, pues multiplicaría los tiempos de entrenamiento que ya son considerablemente altos. Así pues, la elección de los valores de dichos hiperparámetros se ha hecho en base a conocimiento experto, y algunas fuentes como [14, 52].

5.3.2. Algoritmos utilizados

Un aspecto fundamental durante la resolución de un problema mediante RL es el algoritmo utilizado. Ya se ha indicado en varias ocasiones, que en este caso se utilizan técnicas de aprendizaje por refuerzo profundo (DRL) para abarcar este problema. Concretamente, Sinergym soporta los algoritmos implementados en Stable Baselines 3, una de las librerías de código abierto más útiles y populares en DRL por las implementaciones que ofrece. Además, se ha optado por utilizar todos los algoritmos que permite la herramienta, para comprobar si existe un mejor rendimiento por parte de alguno de ellos. Otra razón por la que probar todos estos algoritmos es la diferencia en su naturaleza. Algunos de ellos, solo pueden trabajar con espacios de acciones discretos, o continuos. Por otro lado, existen algoritmos *on-policy* y *off-policy* para cada espacio de acciones. De esta manera, se podrá analizar si, de nuevo, algún tipo de algoritmo resulta superior a otro en términos de

optimización.

Dicho esto, los algoritmos elegidos son los descritos en el capítulo 2, en la sección de algoritmos (véase sección 2.4): **A2C, DQN, PPO, SAC y TD3**. Tanto A2C como DQN se emplean en entornos con espacios de acciones discretos, mientras que PPO, SAC y TD3 se usan en el caso continuo. Por otro lado, nótese que A2C y PPO son algoritmos *on-policy*, a diferencia del resto, que son *off-policy*.

Para medir y comparar el rendimiento de estos algoritmos, se propone utilizar un **controlador basado en reglas** (*rule based controller*, RBC) como ***baseline***. Este controlador tiene un comportamiento relativamente sencillo. Se limitará a establecer un par de *setpoints* durante todo el tiempo según la estación en la que se encuentre: 23°C y 26°C durante los tres meses de verano, y 20°C y 23.5°C durante el invierno, tal y como recomienda la asociación ASHRAE [27].

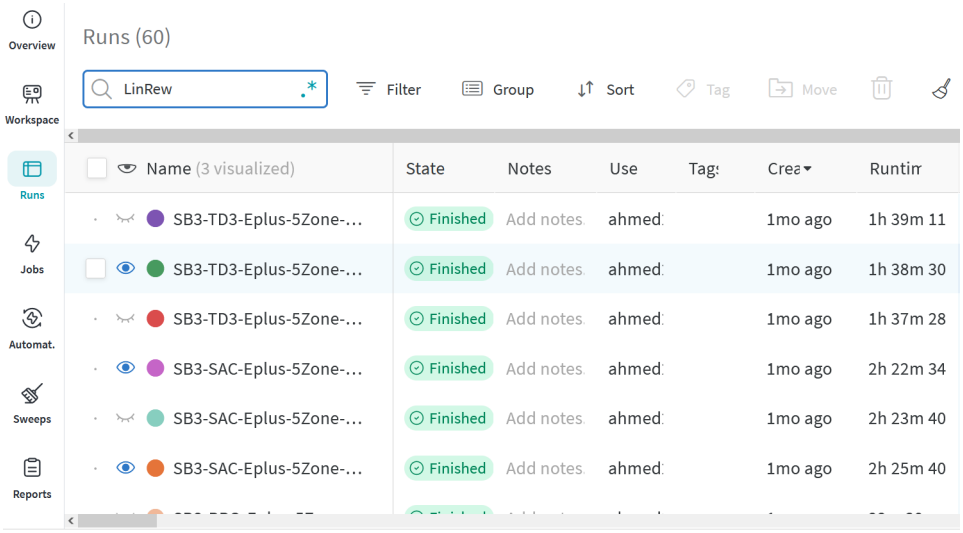


Figura 5.3: Visualización de experimentos con WandB.

5.3.3. Espacio de acciones

Otro punto referente a la experimentación que debe ser detallado es el espacio de acciones. Se utilizan dos conjuntos diferentes: un espacio de acciones discreto y otro continuo.

El **espacio de acciones discreto** se define mediante 12 pares de setpoints, para activar la calefacción o la refrigeración, según el umbral que se supere. Nótese que los umbrales de temperatura se expresan en grados

centígrados. Serían lo siguientes:

Setpoint calef.	15	16	17	18	19	20	21	22	23	23.5	20.5	20.2
Setpoint refrig.	30	29	28	27	26	25	24	25.5	26.5	25.8	23.5	23

Tabla 5.1: Conjunto de pares de *setpoints* del espacio de acciones discreto.

Por otra parte, el **espacio de acciones continuo** se define de distinta manera. Concretamente, se determina un rango en el que los setpoints pueden tomar cualquier valor. Estos intervalos son los siguientes:

Setpoint	**Mínimo**	**Máximo**
Frío	22.5	30.0
Calor	15.0	22.5

Tabla 5.2: Rangos de *setpoints* del espacio de acciones continuo.

5.3.4. Métricas de evaluación

Una vez presentados todos estos detalles sobre el entrenamiento de los agentes, queda por definir un aspecto fundamental: las métricas que se van a utilizar para su evaluación. Sin ellas, es imposible medir el rendimiento de los distintos modelos y compararlos entre sí. Asimismo, no se utilizará una única métrica, sino que se han elegido varias para poder realizar un análisis más completo. Además, al tratarse de un **problema multi-objetivo** (se trata de maximizar el confort térmico y minimizar el consumo energético simultáneamente), conviene estudiar cada aspecto por separado. De esta manera, se han utilizado **tres métricas distintas** (considerando el promedio de cada una en lugar del valor acumulado, ya que así es más intuitivo y fácil de tratar): **recompensa media**, **consumo energético** y **confort térmico**.

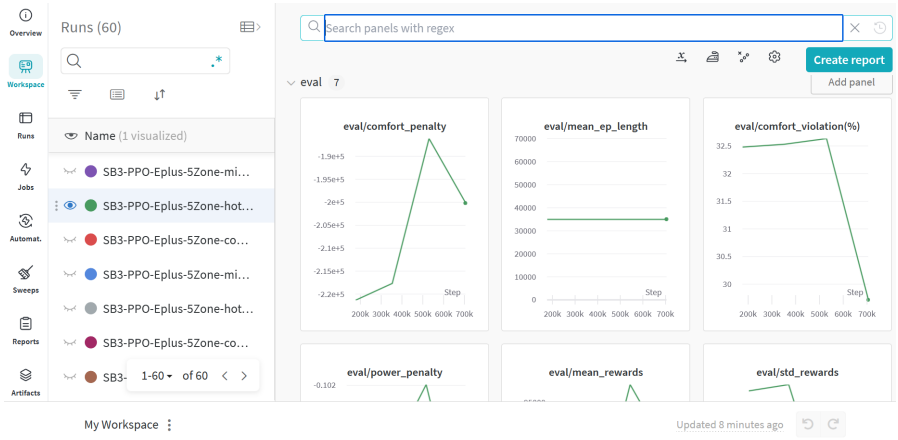


Figura 5.4: Monitorización de métricas de evaluación con WandB.

Como se puede intuir, el valor de la **función de recompensa** que obtiene cada agente es una métrica imprescindible, ya que de cierta manera considera todos los aspectos que se desean optimizar a la hora de resolver el problema. Uno de los principales objetivos de este proyecto consiste en estudiar la influencia de distintas recompensas. Por esta razón, se han diseñado varias funciones de recompensa, para posteriormente compararlas. Se listan más adelante. Antes de detallar cada una, conviene recordar el esquema que comparten todas ellas. Las funciones de recompensa aquí propuestas tienen la siguiente estructura:

$$r(S_t, A_t) = w_t \cdot \lambda_c \cdot f(S_t, S_{target}) + (1 - w_t) \cdot \lambda_e \cdot coste(A_t) \tag{5.1}$$

En esta expresión se diferencian dos términos, uno referente al consumo energético y otro al confort energético, que se suman tras aplicar una ponderación (definida por w_t). Cabe recordar que λ_c y λ_e eran simplemente factores de normalización, para mantener ambas variables en una misma escala. Además, nótese que la función $coste(A_t)$ representa el coste energético, como su nombre indica, mientras que $f(S_t, S_{target})$ hace referencia a la penalización por violación de confort. Las diferencias entre una recompensa y otra residen en la forma de ponderar dicha suma (w_t) y en la forma de penalizar el confort. A continuación, se explica cada función de recompensa con detalle.

La recompensa básica, propuesta por defecto en Sinergym, se denomina ***Linear Reward*** (abreviada LinRew). Como su propio nombre indica, se trata de una recompensa lineal, ya que la penalización por salir del rango de confort es lineal, cuanto más se aleje. Además, siempre pondera con la misma importancia tanto el confort como el consumo, por lo que la variable

w_t siempre toma valor 0.5.

La primera recompensa propuesta se nombra ***Occupation Reward*** (abreviada como OccupRew), ya que se rige en parte por la ocupación del edificio. Concretamente diferencia dos escenarios, uno en el que no hay nadie en el edificio o zona a controlar, y otro en el que hay al menos una persona. La idea subyacente es que cuando no hay nadie en el edificio, no merece la pena considerar el confort, por lo que w_t toma valor 0, otorgando toda la prioridad al ahorro energético. Siendo n_p el número de personas dentro de la zona que se desea climatizar, el valor de w_t se define como sigue:

$$w_t = \begin{cases} 0, & \text{si } n_{occup} = 0 \\ 0.5, & \text{si } n_{occup} > 0 \end{cases}$$

Las recompensas propuestas a continuación también siguen este criterio. Sin embargo, la diferencia reside en la forma de penalizar el confort. En las dos propuestas anteriores, la penalización es lineal, y que se penaliza de forma directamente proporcional a la magnitud de la violación de confort. Es decir, la penalización por exceder el umbral en dos grados centígrados será el doble de si se excede en un solo grado, por ejemplo.

La siguiente recompensa que se plantea toma como nombre ***Exponential Reward***[8], o ExpReward. Como se puede intuir, en este caso la penalización por violación de confort es exponencial en lugar de lineal. Concretamente la penalización toma el siguiente valor:

$$f(T_{current}, T_{target}) = \sum_{i=1}^{z} e^{[T_t^i - \overline{T_t^i}]_+ + [T_t^i - \underline{T_t^i}]_+)} \tag{5.2}$$

El propósito de esta penalización exponencial es tratar que el agente no exceda los umbrales propuestos, al menos en gran medida, y que se mantenga más próximo al intervalo de temperatura deseado.

La última función de recompensa propuesta (***Strict Exponential Reward***, o StrExpRew) surge a partir de esta misma idea. De hecho, la violación del confort también se penaliza de forma exponencial. La única diferencia es que se suma una constante a la penalización, para que tome una mayor magnitud. Dada la escala de la penalización, se propone que esta constante tome el valor 5. Se formula como sigue:

[8]No confundir con homónima ya implementada en Sinergym

$$f(T_{current}, T_{target}) = \sum_{i=1}^{z} C + e^{[T_t^i - \overline{T_t^i}]_+ + [T_t^i - \underline{T_t^i}]_+)} \tag{5.3}$$

Donde C, como se ha indicado, siempre toma el valor 5.

El **consumo energético** es un aspecto de gran importancia, dado el interés que existe en reducirlo lo máximo posible (recuérdese que esta era una parte relevante de la motivación de este proyecto). También es la variable más sencilla de medir, ya que la propia herramienta permite consultarla directamente sin necesidad de realizar cálculos complejos. Asimismo, el consumo promedio referente a cada episodio se puede examinar gracias a la variable *Facility Total HVAC Electric Demand Power*. También existe una variable que mide, según el mismo criterio, la penalización por consumo de energía eléctrica (ponderando el consumo por un factor de normalización). Esta penalización es la que figura en las recompensas propuestas.

Como ya se ha indicado, el otro aspecto que trata de optimizarse en este problema es el **confort térmico** de los ocupantes dentro del edificio. En este caso, Sinergym ofrece dos medidas diferentes para medirlo. Por un lado, existe el **porcentaje de violación de confort**, que hace referencia al porcentaje de tiempo que transcurre cuando la temperatura del edificio o la estancia no está dentro del intervalo deseado, sin importar en qué medida se aleje dicha temperatura del rango objetivo. Este último detalle es poco deseado, ya que no es igual de deseable alejarse en mayor o menor medida del rango deseado, y esta medida no contempla esta diferencia. Es por ello que conviene utilizar la **penalización de confort** (o *confort penalty)*. Esta métrica refleja la diferencia promedio entre la temperatura del edificio o zona y la temperatura deseada (considerando el límite más cercano del intervalo), de tal manera que se puede analizar la magnitud de la "violación de confort". Nótese que en las distintas recompensas, el criterio para considerar que la temperatura está fuera de los rangos deseado varía, pues sólo en la recompensa lineal se considera una violación de confort independientemente de la ocupación, mientras que en el resto de recompensas, sólo se puede penalizar en los tramos horarios donde hay personas en el edificio o habitación. En este estudio, para el análisis se utiliza la penalización de confort.

Capítulo 6

Análisis

Este capítulo se dedica al análisis de los resultados obtenidos tras la experimentación. Durante este estudio, se comparan los numerosos modelos generados: tras todas las combinaciones de algoritmos, climas y funciones de recompensa propuestas, se entrenan 60 modelos. Estos surgen de entrenar agentes con cinco algoritmos distintos, en tres climas y utilizando cuatro funciones de recompensa. Además, como se ha indicado en el capítulo anterior, no se utilizará la función de recompensa como única métrica para comparar los modelos, sino que también se considerarán y estudiarán el consumo energético y la violación de confort por separado. Esto se hace para observar si algunos modelos son superiores en uno o ambos aspectos.

Dicho esto, la estructura de este capítulo es la siguiente. Se comienza analizando el valor de las distintas funciones de recompensa que obtiene cada modelo. Más adelante, se analiza el consumo de estos controladores, según la recompensa con la que se han entrenado para estudiar su impacto. Por último, de forma análoga, se estudia el desempeño desde el punto de vista del confort térmico.

6.1. Recompensa

En primer lugar, se muestran los resultados que obtienen los agentes para la primera recompensa: **Linear Reward** (véase la tabla 6.1). Uno de los aspectos que más llama la atención es cómo el RBC obtiene un rendimiento superior al resto de modelos en términos de recompensa media obtenida, logrando una mejora notable sobre el resto de algoritmos. Se cree que esta superioridad se debe al propio comportamiento del RBC, ya que este mantiene siempre los dispositivos HVAC encendidos y con los *setpoints* a la temperatura deseada (aunque esto requiera más energía). De esta manera, se minimiza la violación de confort, y la recompensa resultante resulta más

alta. Sin embargo, con las siguientes funciones de recompensa se verá cómo esta "política" no es óptima en todas las circunstancias.

Además, según estos mismos resultados, se observa que el entorno más difícil de tratar es, con diferencia, el cálido, mientras que el más sencillo es el mixto. Probablemente, esto se debe que las temperaturas en los climas cálido y frío son más extremas, de ahí que sea más costoso mantener el edificio con una temperatura deseada.

	HOT	**MIXED**	**COOL**
A2C	-1.356	-0.573	-0.630
DQN	-1.365	-0.559	-0.618
PPO	-1.341	-0.545	-0.604
SAC	-1.357	-0.552	-0.613
TD3	-1.495	-0.592	-0.596
RBC	**-1.215**	**-0.436**	**-0.565**

Tabla 6.1: Recompensa media obtenida por agentes entrenados con Linear Reward. En negrita figura el mejor resultado para cada clima.

A continuación se muestra la recompensa obtenida de los agentes entrenados con la segunda función propuesta: **Occupation Reward**. Cabe recordar que esta función de recompensa sólo penaliza el confort cuando hay ocupación en el edificio.

	HOT	**MIXED**	**COOL**
A2C	-1.109	-0.461	-0.431
DQN	-1.088	-0.442	-0.418
PPO	**-1.045**	**-0.431**	-0.411
SAC	-1.087	-0.438	-0.433
TD3	-1.156	-0.458	**-0.407**
RBC	-1.230	-0.518	-0.483

Tabla 6.2: Recompensa media obtenida por agentes entrenados con Occupation Reward. En negrita figura el mejor resultado para cada clima.

En este caso, el RBC no es el mejor modelo, sino todo lo contrario, todos los modelos entrenados con DRL superan a este controlador. La razón de esta mejora en el desempeño de estos modelos radica en su capacidad de aprender. Mientras que el RBC mantiene siempre el mismo comportamiento, con esta nueva recompensa los modelos entrenados con DRL son capaces de

aprender. Concretamente son capaces de, automáticamente, primar el ahorro energético cuando no hay personas en el interior de la infraestructura. Con esta reducción de consumo energético, se consigue superar al RBC, que pasa a ser una alternativa menos deseable.

Adicionalmente, resaltan PPO y TD3 como los mejores algoritmos, aunque cabe apuntar que la diferencia entre los mejores algoritmos y el resto no es especialmente elevada.

Sin embargo, se espera que este rendimiento puede mejorarse aún más "refinando" las funciones de recompensa. Para ello se proponen las dos recompensas adicionales, que se analizarán de forma conjunta dada su gran similitud. Se muestran los resultados obtenidos a continuación.

	HOT	**MIXED**	**COOL**
A2C	-1.152	-0.566	-0.539
DQN	-1.127	-0.482	-0.461
PPO	**-1.083**	**-0.442**	**-0.451**
SAC	-1.111	-0.497	-0.506
TD3	-1.209	-0.470	-0.578
RBC	-1.523	-0.957	-0.833

Tabla 6.3: Recompensa media obtenida por agentes entrenados con Exponential Reward. En negrita figura el mejor resultado para cada clima.

	HOT	**MIXED**	**COOL**
A2C	-1.165	-0.601	-0.585
DQN	-1.135	-0.540	-0.517
PPO	**-1.101**	**-0.457**	-0.499
SAC	-1.162	-0.506	**-0.476**
TD3	-1.301	-0.611	-0.525
RBC	-1.242	-0.529	-0.843

Tabla 6.4: Recompensa media obtenida por agentes entrenados con Strict Exponential Reward. En negrita figura el mejor resultado para cada clima.

En ambos casos se encuentran diferencias entre el RBC y los modelos de DRL aumentan en la mayoría de los casos. Esto se debe principalmente a que el RBC tiene un peor rendimiento cuando se utilizan estas dos funciones de recompensa, ya que salir de los rangos de temperatura deseados supone

una mayor penalización (exponencial en este caso). De nuevo, PPO resulta como uno de los mejores algoritmos, tal y como ocurría en el caso anterior.

Dicho esto, se procede a analizar los modelos según dos criterios (funciones de pérdida): consumo energético y confort térmico.

6.2. Consumo energético

Utilizar este criterio durante el análisis resulta crucial, ya que permite estudiar en qué medida se logra un ahorro energético respecto al modelo de referencia. No debe olvidarse que este es el objetivo final de no solo este proyecto, sino toda una línea de investigación. Se comienza analizando el consumo obtenido cuando se entrena con la **primera recompensa**.

	HOT	**MIXED**	**COOL**
A2C	1738	**1315**	**605**
DQN	**1731**	1326	624
PPO	1793	1332	612
SAC	1750	1341	621
TD3	1810	1360	622
RBC	1984	1453	709

Tabla 6.5: Consumo energético medio obtenido por agentes entrenados con Linear Reward. En negrita figura el mejor resultado para cada clima.

En este caso, el RBC no resulta la mejor opción en ningún caso. De hecho, es el controlador que más energía consume en todos los climas. Esto se debe a que, por su comportamiento, trata de mantener siempre una temperatura deseable, lo que se traduce en un aumento de la demanda energética. A2C y DQN resultan las mejores opciones en este sentido aunque, como ocurría con la recompensa acumulada, las diferencias entre unos y otros modelos de DRL no son demasiado notorias, por lo que todas las opciones resultan similares. En definitiva, se logra un ahorro energético de un 12.75 % en clima cálido, 9.49 % en el templado y 14.67 % en el frío (considerando los mejores modelos). Estas cifras sugieren que aplicar DRL para resolver este problema de control es una solución bastante prometedora, ya que permite alcanzar un ahorro significativo.

En la siguiente tabla, se muestran los resultados de esta métrica para los modelos entrenados con **Occupation Reward**:

	HOT	**MIXED**	**COOL**
A2C	1248	833	436
DQN	1311	906	517
PPO	1205	875	**424**
SAC	**1178**	**778**	469
TD3	1493	1030	553
RBC	1984	1453	709

Tabla 6.6: Consumo energético medio obtenido por agentes entrenados con Occupation Reward. En negrita figura el mejor resultado para cada clima.

En este caso, se observa que las diferencias entre el consumo de los modelos y el *baseline* aumentan bastante. Concretamente, se consigue disminuir el consumo en desde un 40 % en climas frío y cálido, hasta un 46.46 en el templado. Esto demuestra que, efectivamente, los modelos entrenados resultantes son capaces de adaptarse a los horarios de ocupación del edificio, modificando únicamente la función de recompensa, para así dar prioridad al ahorro energético o al bienestar según corresponda. En este caso, PPO presenta un mejor rendimiento en climas fríos, mientras que SAC prevalece como mejor algoritmo en el resto.

De nuevo, dada la similitud de las dos funciones de recompensa restantes, se analizan de forma conjunta. Los resultados se muestran en las dos siguientes tablas:

	HOT	**MIXED**	**COOL**
A2C	1322	**849**	501
DQN	1359	1008	538
PPO	**1288**	913	**484**
SAC	1313	854	498
TD3	1462	1095	619
RBC	1984	1453	709

Tabla 6.7: Consumo energético medio obtenido por agentes entrenados con Exponential Reward. En negrita figura el mejor resultado para cada clima.

	HOT	MIXED	COOL
A2C	1380	**932**	755
DQN	1411	1058	610
PPO	**1332**	1000	562
SAC	1353	939	**537**
TD3	1499	991	693
RBC	1984	1453	709

Tabla 6.8: Consumo energético medio obtenido por agentes entrenados con Strict Exponential Reward. En negrita figura el mejor resultado para cada clima.

Los resultados de estos modelos resultan bastante similares entre sí, en términos de consumo energético. En el clima cálido se consigue ahorrar un 35 % aproximadamente con ambas recompensas. En el caso templado, el ahorro varía entre 41.57 % y 35.85 %, mientras que en el frío va desde un 31.73 % a un 24.26 %. En todos los casos, el ahorro máximo lo alcanzan los modelos entrenados con Exponential Reward. Se observa que estas funciones de recompensa permiten una reducción del consumo ligeramente menor que la anterior. Esto se debe a que se penaliza más la violación de confort, por lo que los agentes son algo más estrictos al tratar de mantener la temperatura, y eso supone un ligero aumento en el consumo, aunque este incremento no resulta considerablemente alto, resultando ser ambas buenas propuestas de función de recompensa.

6.3. Confort térmico

Por último, se analiza en qué medida consigue cada agente mantener la temperatura en un intervalo deseado. Cabe recordar que la medida utilizada en este caso representaba la temperatura promedio que se desviaba de los límites de confort. Para realizar una comparación justa, aunque los modelos se hayan entrenado con distintas penalizaciones en el confort, se han evaluado utilizando un mismo criterio (el que sigue la recompensa Occupation Reward). De esta manera, sólo se penaliza cuando hay gente en el edificio o zona a controlar. La magnitud de esta penalización es lineal, y se corresponde con la temperatura promedio que se aleja del intervalo deseado.

Para empezar, se muestran los resultados de los modelos entrenados con **Linear Reward**:

	HOT	**MIXED**	**COOL**
A2C	1.39	0.53	0.51
DQN	1.35	0.50	0.39
PPO	1.29	0.42	0.48
SAC	1.36	0.46	0.47
TD3	1.29	0.47	0.42
RBC	**0.45**	**0.39**	**0.33**

Tabla 6.9: Penalización por confort obtenida por agentes entrenados con Linear Reward. En negrita figura el mejor resultado para cada clima.

Se encuentra que el RBC es el modelo que mejor respeta el confort térmico, por su propio comportamiento como se ha explicado anteriormente. El resto de modelos muestran un desempeño peor, especialmente en el clima cálido, que resulta ser el más complejo. Asimismo, en el resto de climas, las diferencias no son especialmente elevadas, por lo que se puede asumir que los modelos generados son bastante adecuados.

Dicho esto, se desea estudiar si el resto de funciones de recompensa propuestas consiguen mejorar los modelos en este aspecto. A continuación, se muestra el caso de la función **Occupation Reward**:

	HOT	**MIXED**	**COOL**
A2C	1.43	0.55	0.55
DQN	1.39	0.51	0.53
PPO	1.35	0.49	0.52
SAC	1.41	0.50	0.55
TD3	1.34	0.48	0.51
RBC	**0.45**	**0.39**	**0.33**

Tabla 6.10: Penalización por confort obtenida por agentes entrenados con Occupation Reward. En negrita figura el mejor resultado para cada clima.

De nuevo, el controlador basado en reglas sigue siendo la mejor alternativa según esta métrica. Además, la magnitud de la violación de confort para el resto de modelos no parece variar demasiado, incluso parece empeorar ligeramente. Esto puede deberse a que con esta nueva política, se relajan las restricciones de confort cuando no hay ocupación (de ahí el ahorro energético antes observado). Consecuentemente, cuando las personas vuelven a entrar

al edificio, es posible que este no esté perfectamente acondicionado, lo que causaría este ligero aumento en las medidas. De todos modos, es importante destacar que este aumento es bastante bajo y, considerando el ahorro energético que se alcanza con esta propuesta, no se considera un problema relevante.

Las dos siguientes funciones de recompensa penalizan la violación de confort de forma más estricta, así que se va a estudiar si efectivamente consiguen reducir la métrica en cuestión. Véase primero el caso de la función **Exponential Reward**:

	HOT	**MIXED**	**COOL**
A2C	0.68	0.43	0.66
DQN	0.53	**0.35**	0.56
PPO	0.96	0.49	0.49
SAC	0.85	0.59	0.79
TD3	1.29	0.45	0.49
RBC	**0.45**	0.39	**0.33**

Tabla 6.11: Penalización por confort obtenida por agentes entrenados con Exponential Reward. En negrita figura el mejor resultado para cada clima.

Observando estos resultados, se encuentra una notable mejora en el confort térmico que permiten alcanzar estos algoritmos, especialmente en el clima cálido, que es el más difícil de tratar. Además, DQN resulta ser ligeramente mejor que el RBC en el clima templado. De esta forma, se concluye que la penalización exponencial ha tenido el efecto deseado y, aunque suponga un pequeño aumento en el coste energético, resulta una buena opción cuando mantener una temperatura deseada es una prioridad. Sin embargo, en el clima frío se encuentran algunas oscilaciones, tanto incrementos como decrementos. Dada su pequeña magnitud, se puede considerar que se deben al ruido incluido en las simulaciones.

Por último, cabe recordar que la función de recompensa **Strict Exponential Reward** penaliza de forma más drástica al agente cada vez que la temperatura sale del rango deseado. En consecuencia, se plantea si los modelos resultantes reducen aun más la violación de confort. Se muestran los valores de esta métrica a continuación:

	HOT	**MIXED**	**COOL**
A2C	0.65	0.46	0.43
DQN	0.48	**0.39**	**0.30**
PPO	0.68	0.58	0.45
SAC	0.56	0.68	0.39
TD3	0.96	0.59	0.35
RBC	**0.45**	**0.39**	0.33

Tabla 6.12: Penalización por confort obtenida por agentes entrenados con Strict Exponential Reward. En negrita figura el mejor resultado para cada clima.

Tras estudiar los resultados, se encuentra que efectivamente, esta función de recompensa permite mejorar aún más el confort térmico. En los tres climas, el mejor agente consigue un rendimiento muy próximo o incluso superior al que obtiene el RBC. Además, se observa que, para cada clima, ninguno supera con diferencia al resto, quedándose cerca de un límite. Esto sugiere la posibilidad de que exista una cota en esta métrica. Se plantea por tanto que las fracciones de tiempo en las que se infringen las restricciones de confort se deben al tiempo de reacción de los propios dispositivos de climatización, ya que estos se activan cuando la temperatura excede un umbral y tardan unos minutos en volver a calentar o refrigerar la estancia hasta la temperatura deseada. De todos modos, cabe añadir que los resultados de la mayoría de los agentes son extremadamente bajos, por lo que se puede considerar que respetan los intervalos de confort propuestos, ya que no se trata de una restricción dura.

Capítulo 7

Conclusiones

Para culminar el proyecto, se dedica este último capítulo al análisis de las conclusiones que se derivan del estudio que se ha llevado a cabo. Se revisa todo el trabajo realizado y se comprueba si se ha cumplido con los objetivos inicialmente propuestos. Adicionalmente, se propone una serie de posibles investigaciones futuras que surgen a raíz de este trabajo. Para coronar el episodio, se incluye una breve valoración personal acerca de este proyecto.

7.1. Objetivos realizados

Antes de avanzar a las conclusiones, es conveniente revisar si se han cumplido los objetivos propuestos al principio. Para ello, se recupera la lista inicial, y se analizan uno a uno los distintos objetivos.

Objetivo 1 Estudio del paradigma de aprendizaje por refuerzo, aprendizaje por refuerzo profundo, y su aplicación al control energético.

Antes de abordar el problema en cuestión, ha sido necesario estudiar los fundamentos del aprendizaje por refuerzo. Asimismo, en el capítulo 2 se han explicado los conceptos más importantes necesarios para la comprensión de este paradigma y su aplicación en este problema. Se han detallado además las técnicas de aprendizaje por refuerzo profundo utilizadas en el proyecto.

Se estudia el problema concreto de control energético inteligente en edificios, centrando la atención en la climatización. Se trata de un problema de optimización, que se formula desde el punto de vista de RL, concretando cada uno de sus elementos fundamentales (sección 2.5).

Objetivo 2 Revisión de la literatura y el estado del arte. Detección de carencias y propuesta de soluciones.

Se ha realizado una revisión literaria a lo largo del capítulo 3. Se resume el recorrido de esta línea de investigación, destacando los estudios más relevantes y exponiendo su principal novedad o interés. Tras este análisis, se han detectado algunas faltas o carencias comunes. Por un lado, se resalta la falta de estandarización en estos estudios, lo que dificulta o imposibilita la comparabilidad entre modelos de distintos estudios. Ante este problema, se propone el uso de la herramienta Sinergym. Por otro lado, se encuentra que la función de recompensa utilizada en muchos de ellos no es demasiado adecuada, pues existen propuestas mejores como las aquí presentadas.

Objetivo 3 Estudio de las herramientas *software* para la resolución del problema: Sinergym y AutoCloud.

Sinergym y AutoCloud son las dos principales herramientas *software* que han permitido el desarrollo de este proyecto. La primera permite la simulación de entornos virtuales en los que entrenar y evaluar agentes de DRL para control energético de edificios. La segunda tiene como objetivo posibilitar y agilizar la ejecución de experimentos en la nube. Por tanto, ha resultado imprescindible realizar un estudio profundo de ambas antes de utilizarlas. Además, ha sido necesario realizar modificaciones en el código fuente para poder llevar a cabo las pruebas deseadas.

Adicionalmente, se ha recurrido a herramientas adicionales: Google Cloud Platform (para ejecución de experimentos), Weights and Biases y Tensorboard (para monitorización del entrenamiento y evaluación), y Google Cloud Storage para el almacenamiento de resultados y modelos. Así, el autor ha tenido que familiarizarse con el uso de estos programas.

Objetivo 4 Diseño y ejecución de la experimentación necesaria para el estudio planteado. Entrenamiento y evaluación de varios modelos candidatos.

Se ha diseñado y ejecutado una batería de experimentos para estudiar distintos aspectos, detallada en el capítulo 5. Acerca de esta experimentación, cabe destacar la proposición de varias funciones de recompensa, con el objetivo de estudiar su impacto en el entrenamiento de los agentes.

Objetivo 5 Análisis de resultados y obtención de conclusiones. Proposición de posibles trabajos futuros.

A lo largo del séptimo capítulo, se analizan los resultados obtenidos de la experimentación desde distintos enfoques, con el fin de comparar los modelos utilizando distintos criterios y métricas de evaluación.

Para terminar, en este último capítulo se concentran todas las conclusiones a las que se ha llegado tras el análisis. Se ha tratado de extraer toda la información útil para la comunidad científica y futuros estudios en esta línea de investigación. Se analizan en la siguiente sección. De forma adicional, se sugieren algunas posibles investigaciones consideradas de interés relacionadas con este proyecto.

7.2. Conclusiones extraídas

Antes de comenzar, conviene recordar que la optimización del control energético en edificios es un problema de gran interés, ya que es una vía interesante para reducir el consumo de energía y, consecuentemente, las emisiones de gases contaminantes a la atmósfera. Al tratarse de un problema de optimización, es extremadamente difícil o incluso imposible determinar si un controlador es verdaderamente óptimo. A pesar de esto, a día de hoy no se sospecha que se haya obtenido una solución inmejorable, ya que se siguen perfeccionando los modelos actuales. Además, esta línea de investigación resulta muy útil, pues aún se siguen utilizando controladores bastante sencillos o poco sofisticados en edificios reales, especialmente para control automático de sistemas de climatización.

A continuación, se procede a condensar las conclusiones que se derivan de este estudio. Cabe destacar que se ha tratado de extraer información útil para la comunidad científica, de tal manera que se pueda utilizar o incorporar en futuras investigaciones de esta línea.

Por un lado, se encuentra que el **espacio de acciones continuo no siempre permite alcanzar un desempeño superior al espacio discreto**, al contrario de lo que podría pensarse, ya que un espacio de acciones continuo otorga más flexibilidad al agente a la hora de ajustar los *setpoints*. Esto sugiere que el espacio de acciones discreto utilizado es suficiente para realizar un ajuste adecuado. Además, cabe destacar que, normalmente, entrenar un agente con un espacio de acciones discreto y, por tanto, más reducido, suele ser menos costoso computacionalmente, por lo que es conveniente considerar esta opción. Además, esto también sugiere que el espacio de acciones discreto diseñado para resolver este problema (diferente al que presenta Sinergym por defecto) es bastante adecuado.

Algo similar ocurre con los algoritmos *on-policy*, como son A2C y PPO.

Al mejorar la misma política que se utiliza para realizar las predicciones en lugar de usar dos distintas, se tiende a pensar que son técnicas menos potentes. Sin embargo, observando los resultados, se encuentra que esto no es cierto, y **los algoritmos *off-policy* no siempre superan a los *on-policy***. Incluso PPO resulta ser uno de los mejores algoritmos en términos de consumo energético. De todos modos, esta afirmación requiere un estudio más completo, ya que no se puede descartar que esta superioridad se deba a la ausencia de un ajuste de hiperparámetros. De hecho, algunos algoritmos como TD3 resultan más inestables (con un desempeño más variable), lo que podría deberse a una mala elección de hiperparámetros. Como se ha indicado anteriormente, realizar una fase de *hyperparameter tuning* ha resultado inviable por la alta carga computacional que esto supondría.

Recuperando uno de los principales objetivos de este proyecto, es imprescindible destacar el **considerable ahorro energético** que se logra obtener con los modelos creados. Además, se ha comprobado que estos modelos son capaces de respetar el confort térmico bastante bien. Para resumir toda esta información, se muestra una tabla a modo de resumen, en la que se incluyen los mejores modelos según distintos criterios, y en qué porcentaje superan al *baseline* en cada uno de ellos.

	HOT		**MIXED**		**COOL**	
	Agente	%	*Agente*	%	*Agente*	%
Consumo	SAC_2	40.63	SAC_2	46.46	PPO_2	40.20
Confort	RBC	-	DQN_3	10.26	DQN_4	9.09
LinRew	RBC	-	RBC	-	RBC	-
OccupRew	PPO_2	15.0	PPO_2	16.8	$TD3_2$	15.7
ExpRew	PPO_3	28.9	PPO_3	53.8	PPO_3	45.9
StrExpRew	PPO_4	11.4	PPO_4	13.6	SAC_4	43.5

Tabla 7.1: Mejores controladores según distintos criterios.

Los subíndices indican la recompensa con la que se han entrenado cada modelo: LinRew (1), OccupRew (2), ExpRew (3) o StrExpRew (4).

Primero, cabe destacar que los modelos basados en DRL permiten reducir considerablemente el consumo energético. Concretamente, los mejores modelos alcanzan un **ahorro del 40.63 % en el clima cálido, 46.46 % en el templado y 40.20 % en el caso frío**. Aunque estas cifras son bastante elevadas de por sí, si estos modelos se distribuyen y aplican a gran escala, se traduciría en un ahorro inmenso a nivel mundial, por lo que aplicar DRL para el control de sistemas HVAC resulta una opción muy interesante para tratar el problema del cambio climático.

En lo que respecta al **confort**, cabe decir que en general, casi todos los modelos obtienen una **métrica aceptable**. En particular, los modelos entrenados con las recompensas con penalización exponencial obtienen mejor desempeño en este sentido, ya que el valor de la violación de confort es tan bajo que podría considerarse nulo.

En tercer lugar, se concluye que **las funciones de recompensa propuestas son correctas**, ya que permiten mejorar bastante las métricas anteriores respecto al modelo de referencia. Además, se demuestra la gran influencia y consecuente importancia en su diseño (este era otro de los aspectos principales a estudiar en este proyecto). A la hora de determinar qué función de recompensa es más recomendable, es necesario tener claros los objetivos y prioridades al afrontar el problema, ya que cada opción conduce a modelos con distinto comportamiento. Por ejemplo, si se quiere **primar el ahorro energético**, resulta más útil la función **Occupation Reward**. Por su parte, si se desea **afinar más en el confort térmico**, aunque esto suponga un ligero aumento en la demanda energética, **Exponential Reward y Strict Exponential Reward** pasan a ser las mejores opciones, especialmente esta última. De todos modos, sí que existe una gran diferencia entre la recompensa lineal básica y el resto. De esta manera, se concluye que **considerar los niveles de ocupación** del edificio para priorizar el ahorro energético cuando no haya personas en el interior **supone una mejora crucial**, por lo que se aconseja implementar esta variable en todos los estudios que no la consideren, que como se ha comentado en varias ocasiones, no son pocos. Dejando este aspecto a un lado, cabe añadir que, una vez se considera la ocupación, tratar de afinar más o menos los pesos de la señal de recompensa no tiene un impacto crítico en el desempeño de los modelos. Por tanto, no merece la pena dedicar mucho "esfuerzo computacional" en tratar de optimizar la función de recompensa a no ser que existan restricciones o requisitos muy concretos a la hora de resolver el problema, ya que una función sencilla permite obtener resultados correctos y aceptables.

Otro aspecto a destacar es la **falta de estandarización** existente entre las publicaciones de este campo, tal y como denuncian numerosos investigadores en [54]. Esto resta objetividad a los estudios que se realizan, ya que resulta extremadamente complicado (o incluso imposible) comparar los modelos desarrollados por distintos autores. Esto se debe a que se utilizan distintos escenarios, modelos de referencia (*baselines*), métricas de evaluación, etc. En consecuencia, es muy difícil realizar una comparación justa para determinar qué modelos resultan mejores que otros. Con este proyecto se trata de arrojar un poco de luz en este aspecto, presentando y proponiendo Sinergym como una herramienta capaz de solucionar este problema, dadas sus numerosas posibilidades a la hora de crear agentes usando técnicas

de DRL.

Por último, se ha tratado de plantear un estudio completo en varios sentidos, para lo que se ha propuesto utilizar distintos tipos de algoritmos y espacios de acciones, condiciones meteorológicas diferentes, etc. El objetivo es permitir que el resto de la comunidad científica pueda estudiar, si lo desea, cuestiones diferentes a las que se han analizado aquí, utilizando los mismos resultados. De esta forma, se trata de **contribuir a posibles investigaciones futuras** relacionadas con este campo.

7.3. Trabajo futuro

A pesar de que el estudio realizado es bastante completo, hay algunos aspectos interesantes que han quedado fuera del alcance de este proyecto. La mayoría de estos aspectos no se han investigado con mayor profundidad por el elevado coste computacional que esto supondría. De todos modos, se lista a continuación una serie de trabajos futuros de interés que están relacionados o derivan de este proyecto:

- **Ajuste de hiperparámetros**. No se ha podido realizar una fase de *hyperparameter tuning* de los algoritmos para tratar de optimizar su rendimiento. Sin embargo, resultaría muy interesante para saber si existe un algoritmo superior al resto para este problema o algunas circunstancias concretas. Una opción sería realizar un ajuste "automático", utilizando AutoRL.
- Experimentar en **otro tipo de entornos o edificios**. En este caso se ha utilizado un edificio pequeño con una zona a climatizar. Se sugiere probar con otro tipo de entornos más grandes como centros comerciales o edificios con varias plantas. Para ello puede resultar muy útil el uso de **técnicas multiagente**.
- **Aprendizaje por refuerzo explicable**. Dado el reciente interés en la inteligencia artificial explicable (XAI), se propone la creación de **modelos subrogados** que traten de imitar al modelo de DRL. Basándose en la política de este último, se puede diseñar un sistema basado en reglas cuyo comportamiento sea más comprensible por el ser humano.
- ***Curriculum learning***. Aplicar currículum learning en el problema de control energético en edificios puede ser interesante para comprobar si, gracias a esta técnica, se pueden obtener modelos con un mejor rendimiento.

Bibliografía

[1] The Global Alliance for Buildings and Construction. 2022 global status report for buildings and construction. Technical report, GABC, 2022.

[2] Sumedha Sharma, Yan Xu, Ashu Verma, and Bijaya Panigrahi. Time-coordinated multi-energy management of smart buildings under uncertainties. *IEEE Transactions on Industrial Informatics*, 2019.

[3] Nan Zhou, Nina Khanna, Wei Feng, Jing Ke, and M. Levine. Scenarios of energy efficiency and co2 emissions reduction potential in the buildings sector in china to year 2050. *Nature Energy*, 2018.

[4] Ahmed Brek Prieto. Estudio y aplicaciones del aprendizaje por refuerzo para control energético. Master's thesis, Universidad de Granada, 2022.

[5] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[6] Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3), 1959.

[7] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[8] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2nd edition, 2018.

[9] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[11] M. L. Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. 2008.

[12] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[13] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Procs. 33rd International Conference on International Conference on Machine Learning*, volume 48 of *ICML'16*, pages 1928—-1937. JMLR.org, 2016.

[14] Antonio Manjavacas Lucas. Deep reinforcement learning para control energético eficiente de edificios. Master's thesis, Universidad de Granada, 2021.

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[16] Long-Ji Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Carnegie Mellon University, 1992.

[17] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning, 2015.

[18] Marc Lanctot Ziyu Wang, Nando de Freitas. Dueling network architectures for deep reinforcement learning, 2015.

[19] J. Fernando Hernandez-Garcia and Richard S. Sutton. Understanding multi-step deep reinforcement learning: A systematic study of the dqn target, 2019.

[20] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2015.

[21] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv*, 2017.

[22] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *arXiv*, 2015.

[23] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning

with a stochastic actor. In *Procs. 35th International Conference on Machine Learning*, Stockholm, Sweden, 2018.

[24] S. Kullback and R.A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[25] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv*, 2015.

[26] AHSRAE. *ASHRAE standard thermal environmental conditions for human occupancy.* 1992.

[27] AHSRAE. *2004 Revision of ASHRAE standard thermal environmental conditions for human occupancy.* 2004.

[28] H.E. Burroughs and Shirley J. Hansen. *Managing Indoor Air Quality.* River Publishers, 5th edition, 2011.

[29] Tianshu Wei, Yanzhi Wang, and Qi Zhu. Deep reinforcement learning for building hvac control. In *Proceedings of the 54th Annual Design Automation Conference 2017.* Association for Computing Machinery, 2017.

[30] Charles W. Anderson, Douglas C. Hittle, Alon D. Katz, and R.Matt Kretchmar. Synthesis of reinforcement learning, neural networks and pi control applied to a simulated heating coil. *Artificial Intelligence in Engineering*, 11(4):421–429, 1997.

[31] Michael C. Mozer. The neural network house: An environment that adapts to its inhabitants. In *Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments*, pages 110–114, 1998.

[32] Zeyang Li, Zhe Sun, Qinglong Meng, Yuxiang Wang, and Yang Li. Reinforcement learning of room temperature set-point of thermal storage air-conditioning system with demand response. *Energy and Buildings*, 259:111903, 2022.

[33] José Vázquez-Canteli, Jérôme Kämpf, and Zoltán Nagy. Balancing comfort and energy consumption of a heat pump using batch reinforcement learning with fitted q-iteration. *Energy Procedia*, 122:415–420, 2017. CISBAT 2017 International ConferenceFuture Buildings & Districts – Energy Efficiency from Nano to Urban Scale.

[34] Silvio Brandi, Marco Savino Piscitelli, Marco Martellacci, and Alfonso Capozzoli. Deep reinforcement learning to optimise indoor temperature control and heating energy consumption in buildings. *Energy and Buildings*, 224, 2020.

[35] Xiangtian Deng, Yi Zhang, Yi Zhang, and He Qi. Towards optimal hvac control in non-stationary building environments combining active change detection and deep reinforcement learning. *Building and Environment*, 211:108680, 2022.

[36] Frederik Ruelens, Sandro Iacovella, Bert J. Claessens, and Ronnie Belmans. Learning agent for a heat-pump thermostat with a set-back strategy using model-free reinforcement learning. *Energies*, 8(8):8300–8318, 2015.

[37] Giuseppe Tommaso Costanzo, Sandro Iacovella, Frederik Ruelens, T. Leurs, and Bert Claessens. Experimental analysis of data-driven control for a building heating system. *Sustainable Energy, Grids and Networks*, 6:81–90, 2016.

[38] D. Urieli and P. Stone. A learning agent for heat-pump thermostat control. In *Procs. 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, volume 2, pages 1093–1100, 2013.

[39] Bocheng Li and Li Xia. A multi-grid reinforcement learning method for energy conservation and comfort of hvac in buildings. In *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 444–449, 2015.

[40] P. Fazenda, K. Veeramachaneni, P. Lima, and Una-May O'Reilly. Using reinforcement learning to optimize occupant comfort and energy usage in hvac systems. *Journal of Ambient Intelligence and Smart Environments*, 6(6):675–690, 2014.

[41] Yue Lei, Sicheng Zhan, Eikichi Ono, Yuzhen Peng, Zhiang Zhang, Takamasa Hasama, and Adrian Chong. A practical deep reinforcement learning framework for multivariate occupant-centric control in buildings. *Applied Energy*, 324:119742, 2022.

[42] Xi Fang, Guangcai Gong, Guannan Li, Liang Chun, Pei Peng, Wenqiang Li, Xing Shi, and Xiang Chen. Deep reinforcement learning optimal control strategy for temperature setpoint real-time reset in multi-zone building hvac system. *Applied Thermal Engineering*, 212:118552, 2022.

[43] Ibrahim Shaer and Abdallah Shami. Data-driven methods for the reduction of energy consumption in warehouses: Use-case driven analysis. *Internet of Things (Netherlands)*, 23, 2023.

[44] Zhen Yu and Arthur Dexter. Online tuning of a supervisory fuzzy controller for low-energy building system using reinforcement learning. *Control Engineering Practice*, 18(5):532–539, 2010.

[45] K. Dalamagkidis, Denia Kolokotsa, Kostas Kalaitzakis, and G. Stavrakakis. Reinforcement learning for energy conservation and comfort in buildings. *Building and Environment*, 42:2686–2698, 07 2006.

[46] José R. Vázquez-Canteli and Zoltán Nagy. Reinforcement learning for demand response: A review of algorithms and modeling techniques. *Applied Energy*, 235:1072–1089, 2019.

[47] Donald Azuatalam, Wee-Lih Lee, Frits de Nijs, and Ariel Liebman. Reinforcement learning for whole-building hvac control and demand response. *Energy and AI*, 2, 2020.

[48] Xiaolei Yuan, Yiqun Pan, Jianrong Yang, Weitong Wang, and Zhizhong Huang. Study on the application of reinforcement learning in the operation optimization of hvac system. *Building Simulation*, 14, 12 2019.

[49] Liang Yu, Yi Sun, Zhanbo Xu, Chao Shen, Dong Yue, Tao Jiang, and Xiaohong Guan. Multi-agent deep reinforcement learning for hvac control in commercial buildings. *IEEE Transactions on Smart Grid*, PP:1–1, 2020.

[50] Anchal Gupta, Youakim Badr, Ashkan Negahban, and Robin G. Qiu. Energy-efficient heating control for smart buildings with deep reinforcement learning. *Journal of Building Engineering*, 34:101739, 2021.

[51] Yan Du, Helia Zandi, Olivera Kotevska, Kuldeep Kurte, Jeffery Munk, Kadir Amasyali, Evan Mckee, and Fangxing Li. Intelligent multi-zone residential hvac control strategy based on deep reinforcement learning. *Applied Energy*, 281:116117, 2021.

[52] Javier Jiménez-Raboso, Alejandro Campoy-Nieves, Antonio Manjavacas-Lucas, Juan Gómez-Romero, and Miguel Molina-Solana. Sinergym: A building simulation and control framework for training reinforcement learning agents. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 319—-323, New York, NY, USA, 2021. Association for Computing Machinery.

[53] David Wölfle, Arun Vishwanath, and Hartmut Schmeck. A guide for the design of benchmark environments for building energy optimization. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, BuildSys '20, pages 220—-229. Association for Computing Machinery, 2020.

[54] Zoltan Nagy, Gregor Henze, Sourav Dey, Javier Arroyo, Lieve Helsen, Xiangyu Zhang, Bingqing Chen, Kadir Amasyali, Kuldeep Kurte, Ahmed Zamzam, Helia Zandi, Ján Drgoňa, Matias Quintana, Steven

McCullogh, June Young Park, Han Li, Tianzhen Hong, Silvio Brandi, Giuseppe Pinto, Alfonso Capozzoli, Draguna Vrabie, Mario Bergés, Kingsley Nweye, Thibault Marzullo, and Andrey Bernstein. Ten questions concerning reinforcement learning for building energy management. *Building and Environment*, 241:110435, 2023.