UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DISEÑO DE SISTEMAS DE CONTROL DE CLIMATIZACIÓN PARA EDIFICIOS, UTILIZANDO APRENDIZAJE REFORZADO

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

RICARDO EXEQUIEL LÓPEZ DÍAZ

PROFESOR GUÍA:
DIEGO MUÑOZ CARPINTERO

MIEMBROS DE LA COMISIÓN:
FRANCISCO RIVERA SERRANO
OSCAR CARTAGENA VILLALOBOS

SANTIAGO DE CHILE
2022

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: RICARDO EXEQUIEL LÓPEZ DÍAZ
FECHA: 2022
PROF. GUÍA: DIEGO MUÑOZ CARPINTERO

DISEÑO DE SISTEMAS DE CONTROL DE CLIMATIZACIÓN PARA EDIFICIOS, UTILIZANDO APRENDIZAJE REFORZADO

Los sistemas de climatización son aquellos que permiten mantener factores de clima deseados al interior de cualquier inmueble, ya sea calentando o enfriando el aire ingresado para mantener la temperatura en los rangos deseados, o generando flujos de aire para mantener la humedad y concentración de CO_2 en niveles cómodos para la presencia humana. Para lograr su cometido estos sistemas utilizan una cantidad considerable de energía y este gasto aumenta considerablemente cuando el inmueble es un edificio. Es por ello que es necesaria una técnica de control que permita optimizar el uso energético de estos sistemas.

En el presente trabajo se diseñan controladores de estos sistemas utilizando distintos algoritmos de aprendizaje reforzado. Además, se analiza el rendimiento de estos algoritmos en distintas variantes del problema.

El rol de los controladores es escoger acciones para llevar el sistema a la zona de operación, es decir, el de máxima recompensa y el resultado del entrenamiento es un controlador. Los resultados del sistema se muestran de forma cualitativa como cuantitativa, es decir, se muestra gráficamente el avance de los estados y las acciones en el tiempo con el objetivo de analizar la forma de estos. Además, se hace una comparación cuantitativa que incluye la cantidad de energía utilizada, recompensa obtenida y penalizaciones de estados durante una simulación de prueba. Es gracias a estos resultados que podemos comparar el desempeño de los algoritmos para, finalmente, escoger alguno como el mejor y más adecuado para este caso de estudio.

Para realizar el diseño de los distintos controladores se utiliza el *software* Matlab, el cual permite utilizar una amplia gama de algoritmos de aprendizaje reforzado, además, de poder simular el sistema de climatización en la herramienta de *Simulink*. La metodología en este trabajo consiste en diseñar e implementar para luego evaluar si el desempeño, en cuanto a cumplimiento de restricciones, es suficientemente bueno para considerarlo una solución o hay que realizar cambios e implementar nuevamente.

Además de los resultados de los controladores, se construye un principio de *toolbox* que es de utilidad para realizar los experimentos de este trabajo. Este permite cambiar de sistema de una manera relativamente simple, lo que le entrega generalidad para trabajar con cualquier sistema. Además, es fácil de configurar y permite entrenar un controlador con cualquiera de los 3 algoritmos tratados en este trabajo.

Dedicado a mis padres Silvia y Ricardo, y al resto de mi familia.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

En la actualidad, en un mundo globalizado donde existe una densidad poblacional creciente y donde en muchos casos dicha población crece de manera exponencial, se vuelve una necesidad la creación de grandes edificios, como oficinas, centros comerciales, hospitales, entre otros. Por otro lado, tenemos que dichos edificios al ser tan grandes y al contar con una gran cantidad de personas en su interior, poseen un gasto energético considerable. Es por ello, que es de suma importancia el disminuir el gasto energético de los edificios, sobre todo en las áreas comunes, pues es una variable más simple de controlar que el consumo individual.

Bajo la misma linea anterior, los edificios son responsables de un gran porcentaje del gasto energético de una ciudad y, con ello de las emisiones de carbono [1]. Por otro lado, los sistemas de climatización en los edificios pueden llegar a utilizar hasta el 40 % de la energía total del edificio [2], lo cual hace relevante el optimizar lo más posible el gasto energético de estos sistemas.

A medida que han pasado los años, los sistemas de climatización han experimentado evoluciones en sus formas de trabajar, sin embargo, a pesar de estas evoluciones, estos sistemas siguen funcionando al calentar y/o enfriar el aire de distintas maneras. En este caso en particular, se considerará el caso de aumentar y disminuir la temperatura mediante *coils* o espiras por las cuales pasa el aire y cambia su temperatura.

Para el correcto funcionamiento de estos sistemas, es necesario que exista algún mecanismo de control. Este mecanismo debe mantener ciertas temperaturas deseadas a lo largo del día sin importar que tanto uso le estemos dando al edificio, es decir, este mecanismo calienta o enfría el aire en una cierta cantidad para cumplir con los requerimientos que se le solicitan. Además, este debe velar por la vida útil de los dispositivos que permiten estas acciones de calentar o enfriar el aire.

Actualmente, ya existen formas de control implementadas para estos sistemas, sin embargo, estas presentan problemáticas, como ineficiencias debido a la poca variedad de acciones a tomar dada la simpleza en su estructura, como es el caso de los controladores PID. Mientras que otros controladores más avanzados, que utilizan métodos de optimización, necesitan simplificar el problema, además de todo un cálculo matemático que implica una optimiza-

ción explicita para un único sistema en específico, que radica en un alto costo analítico por parte de la persona que lo diseña, y un alto costo computacional para el dispositivo que lo implementa.

Por otro lado, tenemos que actualmente la inteligencia artificial y las redes neuronales se encuentran en el foco de atención y, por ello, también se encuentran en constante desarrollo. Donde además la rama del aprendizaje de máquinas, llamada aprendizaje reforzado, tiene cualidades bastante deseables para abordar un problema como este, cualidades como el llegar a soluciones óptimas, pero con la posibilidad de considerar no linealidades.

El hacer un controlador para un sistema de climatización, utilizando aprendizaje reforzado puede, llegar a ser bastante relevante para estos, pues, dependiendo de los resultados se puede llegar a un controlador que cumpla con mantener las restricciones deseadas, con un gasto energético considerablemente bajo. Esto, además, sin realizar algún cálculo muy exhaustivo con respecto a un modelo en particular, o mejor aún, podemos llegar a un método que permita controlar el sistema de climatización de manera *online* sin la necesidad de tener un modelo en sí.

Luego, dado lo anterior, se propone en este trabajo diseñar e implementar (a nivel de simulación), un sistema de climatización controlado mediante aprendizaje reforzado, con el objetivo de minimizar el gasto energético mientras se cumplen ciertas restricciones que pueden ser: el mantener la temperatura y humedad en un rango agradable para las personas, tener un máximo de concentración de CO_2, etc. Se escoge aprendizaje reforzado pues este nos permite la creación de un controlador óptimo sin simplificaciones del problema y, además, permite evitar una optimización explicita del modelo.

Como se puede notar, las restricciones tienen el objetivo de proporcionar un ambiente agradable para las personas que se encuentren en el interior del edificio.

En este trabajo se prueban distintas implementaciones de aprendizaje reforzado para llegar a obtener el sistema de control deseado. También se prueban distintas configuraciones del sistema, con el fin de estudiar cual de los algoritmos utilizados se desempeña mejor en las distintas situaciones. Se considera la realización de distintas configuraciones del sistema con el fin de abarcar distintas posibilidades de un sistema real, pues, estos pueden variar en sus requerimientos de acciones y restricciones a controlar.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo general del trabajo de titulo es diseñar controladores mediante aprendizaje reforzado para sistemas de climatización de un edificio, los cuales permitan mejorar la eficiencia energética del sistema mientras mantienen ciertas restricciones para no perjudicar la comodidad de personas en su interior, restricciones como Temperatura, humedad relativa y concentración de CO_2.

1.2.2. Objetivos específicos

Para lograr el objetivo general, se consideran los siguientes objetivos específicos:

- Resolver distintas variantes del problema mediante aprendizaje reforzado con un algoritmo del tipo *model-free*.
- Analizar el rendimiento de diversos algoritmos de aprendizaje reforzado en términos de cumplimiento de restricciones y consumo energético.
- Utilizar el software *Matlab* para el diseño de los sistema de control generando un prototipo de *software/toolbox* con los códigos utilizados.

Capítulo 2

Marco teórico

2.1. Sistema HVAC

El sistema de climatización tiene la capacidad de aumentar la temperatura de ambiente (calentar), disminuir la temperatura del ambiente (enfriar) y permitir un correcto flujo de aire, tanto dentro del edificio, al recircular el aire, como en un intercambio del interior al exterior y viceversa, donde la ventilación se da por medio de ventiladores, conductos de aire y válvulas.

El problema a formular es regular la temperatura y otros factores de clima, como humedad y concentración de CO2, en una o más habitaciones de un edificio utilizando un sistema de climatización.

Este problema tiene diversos factores que influyen en él, como la cantidad de personas al interior, el número de puertas/ventanas abiertas o cerradas, la cantidad de artefactos electrónicos, la temperatura al exterior, entre otros.

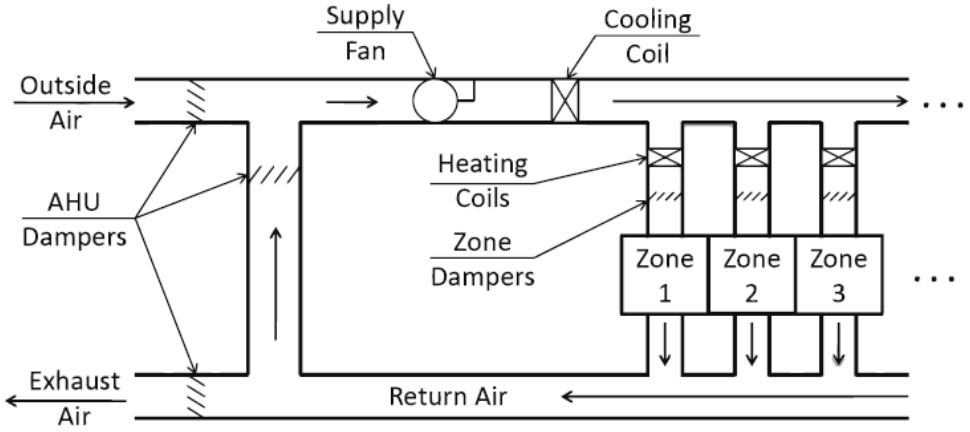


Figura 2.1: Esquema HVAC [3].

Este sistema mostrado en la figura 2.1, genera una corriente de aire utilizando el *Supply Fan* (ventilador). Esta corriente es la que luego puede ingresar o no a cada zona dependiendo de los *Zone Dampers* (válvulas). Estos últimos nos serán útiles únicamente cuando se modele

más de una zona. El *Cooling Coil* (enfriamiento) es la componente del sistema que permite la acción de enfriar al momento que un flujo de aire pasa a través de ella, mientras que los *heating coils* (calefacción) cumplen la función de calentar el aire antes de que ingrese a cada zona. Finalmente, los *AHU Dampers*, controlan el ratio de re-circulación del aire e ingreso de aire fresco.

Luego, el intercambio de la cantidad de CO_2 en una habitación con lo que ingresa a un sistema, se basa en el balance de masas, es decir, si ingresamos aire exterior con una cantidad inferior de CO_2 con respecto a la habitación, entonces se irá haciendo un balance de masas de tal forma que disminuirá la cantidad de CO_2 en la habitación. Por otro lado, tendremos que el mínimo posible será la cantidad que se está ingresando. Esto indicaría que se reemplazó todo el aire existente en la habitación y ocurre de manera similar para el caso en que el aire ingresado contenga una cantidad mayor de CO_2 que el de la habitación, solo que el ingresado ahora sería el máximo.

Dado que la dinámica humedad también se basa en el balance de masas, su comportamiento es similar al de CO_2 anteriormente explicado. Por otro lado, este caso varía debido a que la operación del sistema puede variar con respecto a su localización geográfica. Por ejemplo, si el sistema de climatización se encuentra en una zona muy seca, convendrá utilizar humificadores para regular de mejor forma la humedad relativa dentro del edificio, o bien, en el caso contrario, se utilizarán dehumificadores con el fin de poder cumplir con la restricción de humedad impuesta.

A modo de entender mejor cómo vamos a controlar este sistema con las componentes ya expuestas, y a su vez para estudiar el comportamiento de las características que vamos a restringir y cuáles son los parámetros que las afectan, se presentan las ecuaciones que rigen sus dinámicas. En las ecuaciones siguientes, tenemos que los súper índices 'j' indican la zona a la que se está refiriendo.

En primer lugar, está la dinámica de la temperatura, la cual está basada en el principio de conservación de energía. Se rige por las ecuaciones (2.1) (2.2) (2.3) [3]:

$$C_1^j \dot{T}_1^j = c_p \dot{m}_s^j (T_s^j - T_1^j) + \frac{T_2^j - T_1^j}{R^j} + \frac{T_a - T_1^j}{R_a^j} + P_d^j + \sum_{i \in N^j} \frac{T_1^i - T_1^j}{R_{ij}}, \tag{2.1}$$

$$C_2^j T_2^j = \frac{T_1^j - T_2^j}{R_{12}^j}, \tag{2.2}$$

$$T_s^j = \delta \frac{\sum_{i \in R} \dot{m}_s^i T_1^i}{\sum_{i \in R} \dot{m}_s^i} + (1 - \delta) T_a - \Delta T_c + \Delta T_h^j, \tag{2.3}$$

donde se tiene que los estados T_1^j son temperaturas con dinámicas rápidas, es decir, es la temperatura del aire, mientras que T_2^j es una dinámica lenta, pues pertenece a la temperatura de objetos sólidos, como muebles, paredes, piso, etc.

Además, se tiene que $\dot{m}_s^j$ es el flujo de masa de aire, δ es el ratio de recirculación, es decir, cuánto aire al interior se reutilizará y cuánto se dejará entrar desde el exterior. ΔT_c es la temperatura en el enfriador (*cooling coil* en la figura 2.1), mientras que ΔT_h^j es la temperatu-

ra en el calentador (*heating coils* en la figura 2.1). Donde estas 4 variables sirven para definir la entrada de aire al sistema.

Por otro lado, T_a y P_d^j son la temperatura ambiental y temperaturas internas (como temperatura de personas, aparatos electrónicos, etc), respectivamente. Donde ambas variables se pueden tratar como perturbaciones. Además, tenemos que T_s^j es la temperatura del aire que entra a la habitación j.

Finalmente, la variable C_i^j es la variable de concentración para cada tipo, tanto para sólidos como aire. Mientras que c_p es el calor específico del aire que podemos considerar como $c_p = 1012[J/kg \cdot K]$.

Donde se tiene que la ecuación (2.1) muestra la dinámica de la concentración de temperatura del aire, donde el primer sumando corresponde al intercambio entre la temperatura del aire que entra a la habitación, con el que ya se encuentra en ella; esto multiplicado por el calor específico del aire y el flujo de masa. El segundo sumando corresponde a el intercambio de temperaturas entre el aire de la habitación con los solidos. El tercer sumando corresponde al intercambio de temperaturas entre el ambiente y el aire de la habitación dividido en la resistencia térmica del ambiente. el cuarto sumando son factores internos que influyen en la temperatura (como uso de computadores y gente al interior) y el quinto sumando son los intercambios de temperatura del aire entre habitaciones.

La ecuación (2.2) nos indica la temperatura en los objetos, donde su única componente corresponde al intercambio de temperaturas entre el aire y los objetos.

La ecuación (2.3) indica la temperatura del aire que ingresa a la habitación j, donde el primer sumando indica la cantidad de aire recirculado y su temperatura, el segundo sumando es la influencia de la temperatura del aire exterior ingresado, mientras que T_c es la temperatura en el enfriador (cuánto efectivamente enfría el aire) y T_h^j la temperatura del calentador que está en el nodo de la habitación j.

Por otro lado la concentración de CO2 está dada por [4]:

$$\rho V^j \dot{C}_{CO_2}^j = \dot{m}_s^j (C_{CO_2,s} - C_{CO_2}^j) + g_{CO_2} N_{people}, \tag{2.4}$$

$$C_{CO_2,s} = \delta \frac{\sum_{i \in R} \dot{m}_s^i C_{CO_2}^i}{\sum_{i \in R} \dot{m}_s^i} + (1 - \delta) C_{CO_2,a}, \tag{2.5}$$

donde $C_{CO_2}^j$ es la concentración de CO_2 en la habitación j, mientras que $\dot{m}_s^j$ y δ serán las entradas; estas variables son las mismas que para el caso de la temperatura. Además, $C_{CO_2,a}$ es la concentración de CO_2 en el ambiente y la consideraremos una perturbación, mientras que ρ es la densidad del aire, V^j es el volumen de aire en la habitación j, g_{CO_2} es la generación de CO_2 por persona y $C_{CO_2,s}$ es el ingreso de CO_2 a una habitación.

Luego se tiene que la ecuación (2.4) se refiere a la concentración de CO_2 en la habitación j, donde el primer sumando es el cambio de concentración de CO_2, al interactuar el aire de entrada con el existente en la habitación, y el segundo sumando es el CO_2 generado por las personas en su interior.

Para la ecuación (2.5) se tiene que indica la concentración de CO_2 del aire de entrada, donde el primer sumando es del aire recirculado, mientras que el segundo pertenece al la concentración de CO_2 del aire ingresado desde el exterior del edificio.

Finalmente, se tiene que la dinámica de la humedad se define por [5][6]:

$$\rho V^j \dot{H}^j = \dot{m}_s^j (H_s - H^j) + g_H N_{people}, \tag{2.6}$$

$$H_m = \delta \frac{\sum_{i \in R} \dot{m}_s^j H^i}{\sum_{i \in R} \dot{m}_s^j} + (1 - \delta) H_a, \tag{2.7}$$

$$H_s = min(H_m, H_{sat,T_s}). \tag{2.8}$$

Donde el estado de cada habitación está dado por H^j, la perturbación es el ratio de humedad ambiental H_a, además, g_H es la cantidad de vapor de agua generado por persona, H_m es la humedad mezclada entre la recirculación y el ambiente, H_s es la humedad de entrada, mientras que H_{sat,T_s} es la humedad de saturación, dada por la temperatura actual.

Para las ecuaciones de la dinámica de la humedad, se tiene que actúan de forma similar a las de concentración de CO_2 y dinámica de temperatura. Por lo que solo explicaremos la ecuación (2.8), donde la humedad que ingresa a una sala está dada por el mínimo entre lo recirculado más lo que entra del exterior, y la humedad de saturación. Esto es dado que existe un máximo de humedad absoluta en el aire para cada temperatura, la cual es directamente proporcional a ella, luego si se llega a dicha humedad de saturación, aquella que se añada se condensará.

Luego, esta humedad de saturación depende de la temperatura y la presión, sin embargo, dado que la presión atmosférica no varía en grandes magnitudes, esta se considerará constante (P_a). Luego, se puede modelar la humedad de saturación en función de la temperatura del aire de la siguiente manera[7]:

Tomando la presión atmosférica como $P_a = 1013.25$ y la temperatura del aire donde queremos saber la humedad de saturación como T_s, procedemos a definir como variable auxiliar e_w y a calcular la humedad de saturación(H_s):

$$e_w = 6.1121 \cdot (1.0007 + 3.46 \cdot 10^{-6} \cdot P_a) \cdot \exp\left\{ \frac{17.502 \cdot T_s)}{240.97 + T_s} \right\}, \tag{2.9}$$

$$H_s = 0.62197 \cdot \Big(\frac{e_w}{P_a - 0.378 \cdot e_w} \Big). \tag{2.10}$$

Se observa en la ecuación (2.10) que e_w depende de T_s la cual es la temperatura del aire en ese instante, por lo tanto, podríamos graficar la humedad de saturación en función de la temperatura, como se muestra en la figura 2.2.

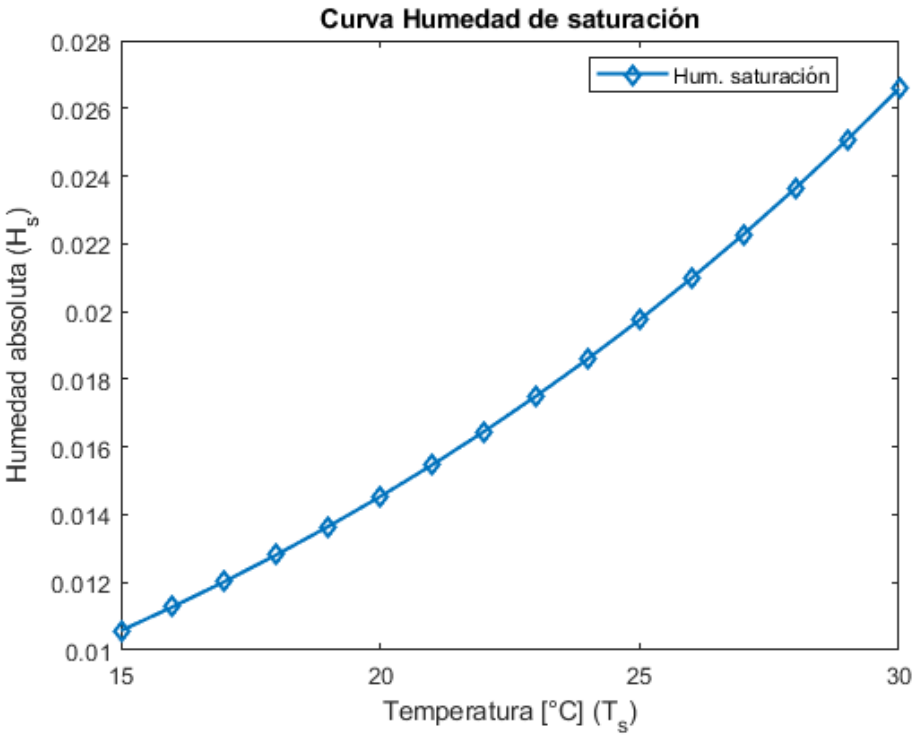


Figura 2.2: Humedad de saturación (H_s) vs temperatura (T_s).

2.2. Aprendizaje reforzado

Es uno de los 3 paradigmas básicos del aprendizaje de máquinas, junto al *aprendizaje supervisado* y el *aprendizaje no supervisado*.

El aprendizaje supervisado es un paradigma de aprendizaje que posee un "supervisor"que guía al agente, con el agente el algoritmo utilizado y el supervisor las salidas a un cierto set de datos como entrada. El objetivo del agente es predecir cómo las entradas llevan a la salida deseada, donde inicialmente predice de manera aleatoria y lógicamente obtiene muchos errores. Luego, el supervisor le indica los errores, permitiendo así, que pueda ir minimizando los errores en la predicción [8].

El aprendizaje no supervisado es cuando al agente le entregamos un set de datos sin ningún tipo de información, es decir, no hay respuestas correctas, sino que el mismo agente explora las estructuras del set de datos[9].

Luego, tenemos que el aprendizaje reforzado se diferencia del supervisado en que el primero no recibe información previa, ni una guía explicita, si no que va interactuando con la planta/el ambiente y recibiendo penalizaciones o recompensas por la interacción y, con ello, logra hacer un balance entre exploración y explotación, obteniendo su propia visión de cómo es la planta. Mientras que la diferencia entre aprendizaje reforzado y no supervisado, es que, la función de recompensa sirve como una pequeña guía para que el agente aprenda de manera correcta. Luego, tenemos que en los problemas de aprendizaje reforzado el agente tiene que 'aprender' qué hacer y cómo actuar para maximizar una señal de recompensa, tomando decisiones mediante *prueba y error*[10].

Para entender cómo funciona lo anteriormente descrito, es que ahondaremos en qué son las

redes neuronales, comenzando por las neuronas. Una neurona toma ese nombre en aprendizaje de máquinas pues trata de simular lo que hace una neurona biológica, que es crear una red capaz de aprender y tomar decisiones. Cada neurona recibe un estimulo y, dependiendo de que tipo de estimulo fue el recibido, entrega una respuesta. Luego, se tiene que una red neuronal es un conjunto de capas de neuronas, tal como podemos ver en el diagrama de la figura 2.3.

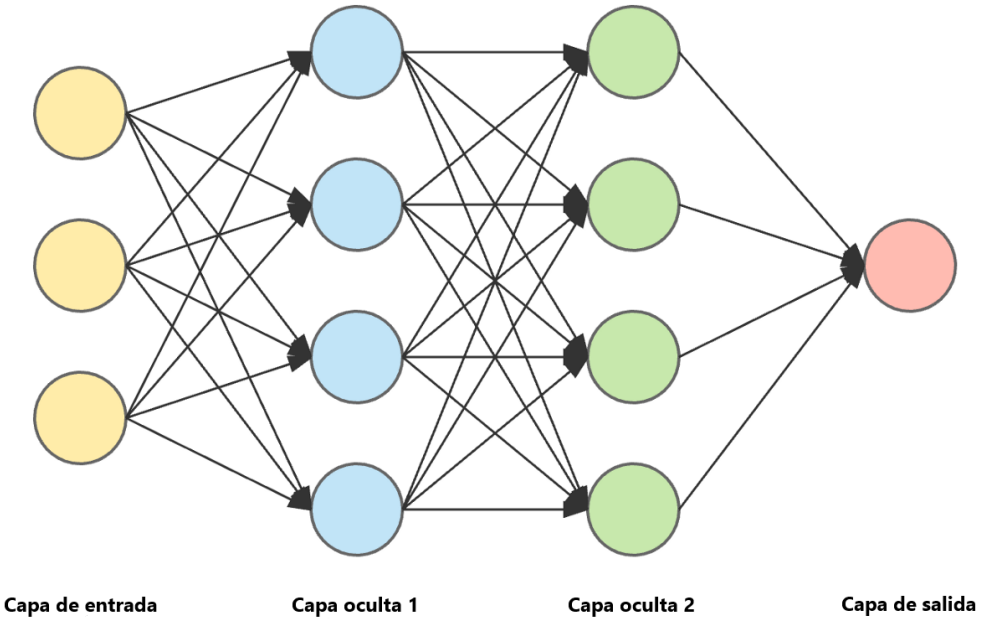


Figura 2.3: Esquema de red neuronal.

Se identifican 4 capas: la capa de entrada, 2 capas ocultas y una capa de salida, donde cada capa oculta posee 4 neuronas o perceptrones. Luego, una red neuronal es una función matemática:

$$salida = f_{NN}(x)$$

. En el caso del diagrama será $salida = f_{NN}(x) = f_{co2}(f_{co1}(entrada))$. Donde, las capas ocultas son funciones vectoriales de la siguiente forma [11].

$$f_l(z) =^{def} g_l(W_l z + b_l). \tag{2.11}$$

Con l el índice de la capa, g_l la **función de activación** que es definida al comienzo y es fija a lo largo del entrenamiento; además, suele ser una función no lineal. Y los parámetros W_l y b_l son entrenados utilizando el gradiente descendente al optimizar alguna función de costo (por ejemplo, la función de error cuadrático medio).

En aprendizaje reforzado se puede resumir su forma de acción en la figura 2.4. esta forma de acción implica que el agente, que es nuestro algoritmo de aprendizaje reforzado, toma una acción A_t (con t indicando el instante en que fue tomado) dada una observación inicial O_t del ambiente (*enviroment*). Esta acción repercute en el ambiente de alguna forma y el agente vuelve a observarlo. Junto con esta nueva observación (O_{t+1}) obtiene también una recompensa (*reward*) R_t y este conjunto de (O_t, A_t, O_{t+1}, R_t) le permiten al algoritmo tener datos suficientes para actualizar la política (*Policy*), la cual es por la que se guía el agente para

tomar la decisión de qué acción tomar frente a alguna observación con el fin de maximizar la recompensa.

Como la finalidad del algoritmo es maximizar una recompensa, se desea que este converja al máximo posible de la función de recompensa y de ahí que se mantendrá estable.

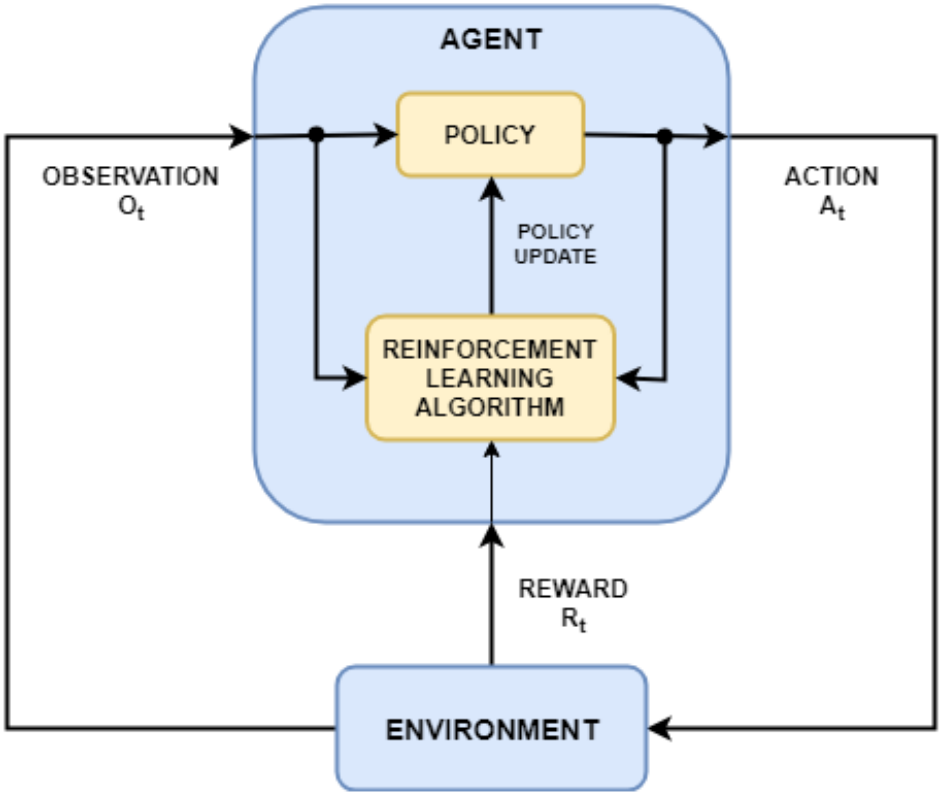


Figura 2.4: Diagrama de funcionamiento de aprendizaje reforzado[12].

Por otro lado se tiene que podemos identificar que el aprendizaje reforzado tiene los siguientes elementos[10]:

- Agente(Agent): es el elemento 'inteligente' del sistema, el cual posee redes neuronales, y toma decisiones observando el ambiente y maximizando la recompensa, entregada por la señal de recompensa.
- Ambiente(Enviroment): es donde queremos situar el aprendizaje reforzado, con qué queremos que interactúe nuestro agente. Podemos decir que toma el mismo rol que la *planta* en control clásico.
- Política(Policy): es donde se define cómo se debe comportar el agente en ciertas situaciones. Podríamos decir que es donde va quedando el aprendizaje del agente luego de interactuar con el ambiente. La política puede ir desde una simple función o tabla, hasta procesos más complejos como un proceso de búsqueda. En general, las políticas suelen ser estocásticas.

- Señal de recompensa: acá se define la meta del problema, cual es nuestro objetivo. En cada paso, luego de cada acción, el ambiente envía una señal de recompensa hacia el agente, la cual le servirá para entender cómo maximizar dicha recompensa a medida que continúa interactuando con el ambiente. Es importante destacar que el agente no puede modificar la señal de recompensa, pues el valor de esta depende únicamente de las acciones que tome el agente y del estado en que se encuentre el ambiente, es decir, podemos modelarlo como una función dependiente del estado del ambiente y la acción del agente.
- Función valor: tenemos que por un lado la señal de recompensa que entrega si una acción/estado es buena o mala de forma inmediata. Sin embargo, la función valor ve un más a largo plazo, indicando cuánta es la recompensa acumulada que se puede obtener de cierto estado, es decir, es un indicador de qué tan deseable es quedarse cercano a cierto estado.

Es, específicamente, esta característica de interactuar con el ambiente, lo que nos permite abordar la gran cantidad de factores que influyen en la temperatura, en un sistema de HVAC(*Heat, Ventilation and Air Conditioning*).

Sin embargo, los métodos comunes de aprendizaje reforzado no tienen un desempeño óptimo si el espacio estado-acción es demasiado grande debido a que trabajan de una forma similar a un *look-up table*, por lo que su uso se restringe a espacios de estado-acción discretos. Es por lo anterior que se utilizan algoritmos de aprendizaje reforzado profundo, los cuales pueden abordar un espacio de estado-acción grande[13].

Según [1], DRL(*Deep reinforcement learning*, que significa aprendizaje reforzado profundo) puede operar de manera online sin información estadística del edificio sobrellevando incertezas y restricciones temporales relacionadas con sistemas HVAC. *Deep reinforcement learning* se diferencia de *reinforcement learning*, en que el primero tiene una estructura de red neuronal con mayor cantidad de capas ocultas y neuronas, además, estima el valor Q con las experiencias, versus *reinforcement learning* que funciona como un *look-up table*, teniendo así un valor Q específico para cada tupla estado-acción. Luego, si existe una tupla con características similares, pero que aún no se experimenta *Deep reinforcement learning* estima cuánto será el valor Q, pero *reinforcement learning* no.

Además, los métodos *model-free* no necesitan conocer el modelo explícitamente (en términos de cómo está formulado el ambiente), pero, necesitan recolectar suficientes experiencias del ambiente para lograr un buen entrenamiento[1]. Sin embargo, operar de esta manera (*online*) puede requerir un alto trabajo computacional. Es aquí donde entra la posibilidad de pre-entrenar el agente de manera *offline*, donde mientras más cercano sea nuestro modelo a la realidad, mejor actuará el agente al ser implementado en la planta real.

2.3. Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) es un algoritmo del tipo actor-crítico, es decir, es un algoritmo que tiene un actor (μ), que es una red neuronal cuya salida será la acción de control, es decir, es una política, y un crítico (Q) que se entrena mediante la ecuación de

Bellman y permite aprender la política a tomar.

Esto está ligado directamente con *Q-learning*, que dice que si es que tenemos la función óptima $Q^*(s,a)$, entonces para cualquier estado, la acción óptima $a^*(s)$, se puede encontrar resolviendo:

$$a^*(s) = \arg\max Q^*(s,a).$$

DDPG se adapta específicamente para ambientes con espacios de estado-acción continuos debido a que dado a lo anterior, se puede asumir que $Q^*(s,a)$ es diferenciable con respecto a la acción. Esto permite utilizar una eficiente regla de aprendizaje basada en el gradiente para la política. Luego para calcular $max_a Q(s,a)$, se puede aproximar simplemente como $maxQ^*(s,a) \approx Q(s,\mu(s))$.

Por otro lado, la función actor nos entregará cual es la acción de control a tomar al evaluarla como: $\mu(s|\theta^\mu) = A_t$, con s el estado, θ^μ el peso de la red actor y A_t la acción tomada. Los pesos de la red especifican cuál es la política actual al mapear los estados hacia una acción de control determinista en especifico. El crítico o función acción-valor representada por $Q(s, A_t|\theta^Q)$, es la encargada de describir la recompensa estimada al tomar la acción A_t sobre el estado s, en otras palabras, el desempeño de la acción.

Luego, el pseudo código del algoritmo DDPG se presenta en el algoritmo 1 [13] [14].

Algorithm 1 Pseudo código para DDPG[13]

1: Se inicia la red que dará origen al crítico $Q(s_t, A_t|\theta^Q)$ y la red que origina al actor $\mu(s_t|\theta^\mu)$ con pesos aleatorios θ^Q y θ^μ.
2: Luego se inicia la red objetivo $Q'(s_t, A_t|\theta^{Q'})$ y la red que origina al actor $\mu'(s_t|\theta^{\mu'})$ con $\theta^{Q'} \leftarrow \theta^Q$ y $\theta^{\mu'} \leftarrow \theta^\mu$.
3: Se inicia el replay buffer $B \rightarrow B = [\,]$.
4: **for** $episodio = 0, 1, ..., M$ **do**
5: Iniciar un proceso aleatorio $N(t)$, que será agregado a μ'
6: Obtener el estado inicial s_0
7: **for** $t = 0, 1, ...T$: **do**
8: Se calcula la acción de control A_t
9: Al aplicar la acción A_t se obtiene el nuevo estado s_{t+1} y se calcula la recompensa R_t.
10: Almacenar los datos (s_t, A_t, R_t, s_{t+1}) en el replay buffer.
11: Seleccionar N transiciones aleatoriamente desde el replay buffer.
12: Calcular la recompensa estimada para cada transición seleccionada.
13: Actualizar la red del crítico minimizando el error sobre las transiciones escogidas:
14: $min\ L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
15: Actualizar el actor usando el 'policy gradient':
16: $\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$
17: Actualizar las redes objetivo Q' y μ'
18: **end for**
19: **end for**

Al comienzo se crean las redes del crítico, el actor y sus redes objetivos. Esto se hace con pesos aleatorios, pues aún no poseemos datos para iniciarlo de otra forma. Luego, se inicializa

el *replay buffer* B, el cual, inicialmente, se encontrará vacío.

Las redes objetivo, tanto para el actor como para el crítico, inicialmente son iguales a la red original, es decir, al actor y al crítico. Estas cumplen la función de mejorar la estabilidad de aprendizaje al ser actualizadas de la forma: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$ con $\tau << 1$, que trae como consecuencia un entrenamiento lento. Luego, al mejorar la estabilidad, se asegura la existencia de una solución robusta, dado que el entrenamiento se acerca a lo que sería aprendizaje supervisado[14].

El paso siguiente es iniciar el loop que indicará la cantidad de episodios a realizar. Se consideran M episodios con M una cantidad que se considere razonable para que el entrenamiento sea efectivo. Al inicio de cada episodio se inicia un proceso aleatorio $N(t)$, el cual servirá para realizar exploración al agregarlo a la acción a tomar. Luego, se obtiene el primer estado y se inicia el loop temporal, donde definimos el largo de cada episodio: en este caso está definido por T.

Siguiendo en el loop temporal, tenemos que en cada instante se calcula la acción de control A_t (que está dado por $\mu(s|\theta^\mu) + N(t) = A_t$). Luego, se obtiene el nuevo estado S_{t+1} con su recompensa R_t, que se calcula mediante la expresión 2.12 y se almacenan los datos anteriores en el buffer. Se continúa escogiendo aleatoriamente N transiciones (datos (S_i, A_i, R_i, S_{i+1})) del buffer y se calcula la recompensa estimada, mediante la ecuación $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}$. Finalmente, se actualiza la red del crítico Q, minimizando el error cuadrático medio sobre las transiciones escogidas, mientras que el actor se actualiza utilizando el 'policy gradient'(1, línea 16), finalizando con la actualización de las redes objetivo mediante la ecuación expuesta anteriormente.

$$R_t = E \cdot \beta + \begin{cases} 0, & \text{Si Cumple las restricciones.} \\ -\Delta t, & \text{Si No cumple las restricciones} \end{cases} \tag{2.12}$$

Tenemos que la expresión 2.12 es sólo cualitativa, la variable E se va a referir a la utilización de energía (su cantidad física en kW), mientras que x e y son solo números arbitrarios para indicar que en esos casos se obtendrá algún tipo de recompensa, contrastando con el caso de no cumplir ninguna restricción.

2.4. Twin Delayed Deep Deterministic Policy Gradient

El segundo algoritmo escogido es *Twin Delayed Deep Deterministic Policy Gradient* (TD3), esto dado que, en ejemplos comparativos en la documentación de Matlab[15], este parecía tener un aprendizaje más estable que DDPG. Si bien su aprendizaje es más lento, la estabilidad que entrega es lo que nos motiva a realizar pruebas con dicho algoritmo. TD3 viene del inglés *Twin Delayed DDPG*, el cual, como su nombre lo dice, está basado en DDPG con ciertas modificaciones.

Según [16] TD3 nace con el objetivo de abordar la sobre estimación del valor Q, la cual es una propiedad en *Q-learning* y que también se encuentra presente en algoritmos que utilizan *deterministic policy gradients* [17] en control continuo, como es el caso de *DDPG*, lo que motiva el buscar la solución a esta sobre estimación.

Para abordar este problema de sobre estimación, en [16] proponen y aplican dos soluciones. La primera es en base a los algoritmos *Double DQN* y *Double Q-learning*. Consiste en utilizar una variante de *Double Q-learning* que utiliza dos redes de crítico y dos de actor [18]. Estos dos actores serán actualizados de la misma forma propuesta en *Double DQN* [19].

Sin embargo, este método no logra eliminar la sobre estimación del todo, por lo que también ocurrirá en la implementación a un actor-crítico. Para evitar esto, se propone limitar superiormente el crítico que entrega un valor mayor (Q_{θ_2}) por el crítico sobre estimado(Q_{θ_1}), como se ve en la ecuación (2.13).

$$y_1 = r + \gamma min_{i=1,2} Q_{\theta_i'}(s', \pi_{\phi_1(s')}). \tag{2.13}$$

Al hacer la actualización limitando Q_θ al valor mínimo de los dos, permite que no exista una sobre estimación exagerada en ciertas áreas del espacio estado-acción (valor Q muy alto). Se evita la sobre estimación pues el crítico se actualiza utilizando el mínimo como se observa en la línea 13 en 2, por lo que Q_{θ_1} iba a tener un mayor peso.

Mientras que la segunda solución es retrasar las actualizaciones en la política utilizada, hasta que el error entregado por el crítico sea lo más pequeño posible. La modificación principal consiste en actualizar la política y las redes objetivos solo cada d actualizaciones del crítico, tal como se puede observar en la línea 14 en el pseudo código 2.

2.5. Proximal Policy Optimization

Se escoge el método *Proximal Policy Optimization algorithms* (PPO) dado que en las comparaciones realizadas en [16], se puede observar un buen rendimiento en las pruebas a las cuales fue expuesto.

PPO está motivado por la pregunta, ¿Cómo podemos mejorar lo más posible una política utilizando los datos que se tienen actualmente, sin utilizar un paso tan grande que pueda accidentalmente generar un colapso en el rendimiento?

Luego, para responder dicha pregunta el algoritmo utiliza un estilo de '*policy gradient*', donde se corre la política en T pasos, con T « M (M es el largo del episodio), y utiliza los datos recolectados para realizar un entrenamiento [20]. Esto requiere un *advantage estimator* (o estimador de ventaja) que no vea más allá del paso T. El estimador usado se puede observar en la ecuación (2.14).

$$\hat{\mathrm{A}}_t = -V(s_t) + R_t + \gamma R_{t+1} + ... + \gamma^{T-t+1} R_{T-1} + \gamma^{T-t} V(s_T), \tag{2.14}$$

donde t especifica el tiempo en el rango $[0, T]$ y V es el crítico.

Luego, lo que queremos maximizar en cada iteración es el substituto objetivo propuesto, que está dado por la ecuación (2.15) [20] [21].

$$L^{CLIP}(\theta) = \hat{\mathrm{E}}_t[min(r_t(\theta)\hat{\mathrm{A}}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{\mathrm{A}}_t)]. \tag{2.15}$$

Tenemos que el pseudo código para expresar el algoritmo de PPO es el mostado en 3.

La política entrenada es del tipo estocástica, esto significa que la exploración se va reali-

Algorithm 2 Pseudo código para el algoritmo TD3 [16].

Se inician las redes que darán origen a los críticos $Q(\theta_1)$, $Q(\theta_2)$ y la red que origina al actor π_ϕ con parámetros aleatorios θ_1 , θ_2 y ϕ.
Luego se inician las redes objetivo, partiendo por los críticos $\theta'_1 \leftarrow \theta_1$ y $\theta'_2 \leftarrow \theta_2$ y luego la red objetivo del actor $\phi' \leftarrow \phi$
Se inicia el replay buffer B $\rightarrow B = [\,]$.
for $episodio = 0, 1, ..., M$ **do**
    for $t = 0, 1, ...T$: **do**
        Observar el estado inicial.
        Seleccionar una acción A_t con un ruido de exploración $A_t \sim \pi_\phi(s) + \epsilon, \epsilon \sim N(0, \sigma)$ y observar la recompensa obtenida R_t y el nuevo estado s_{t+1}.
        Almacenar los datos (s, A_t, R_t, s_{t+1}) en el replay buffer B.
        Seleccionar N transiciones (s, A_t, R_t, s_{t+1}) aleatoriamente desde el replay buffer.
        Se agrega ruido de exploración a la política $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon$, $\epsilon \sim clip(N(0, \tilde{\sigma}), -c, c)$
        Se escoge el mínimo valor Q entre las dos redes objetivos $y \leftarrow r + \gamma\, min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
        Actualizar la red del crítico minimizando el error sobre las transiciones escogidas:
        $\theta_i \leftarrow argmin_{\theta_i} \frac{1}{N} \sum_i (y - Q_{\theta_i}(s, a))^2$
        if t mod d **then**
            Actualizar el actor ϕ usando el 'policy gradient' deterministico:
            $\nabla_{\theta^\mu} J = \frac{1}{N} \sum_i \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
            Actualizar las redes objetivo:
            $\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
            $\phi' \leftarrow \phi + (1 - \tau)\phi'$
        end if
    end for
end for

Algorithm 3 Pseudo código para el algoritmo PPO estilo actor-crítico [22][23]

Se inicializa el actor $\pi(s)$ con pesos aleatorios θ
Se inicializa el critico $V(s)$ con pesos aleatorios ϕ
for episodio $= 1, 2, ..., M$ **do**
    for iteración $= 1, 2, ...T$ **do**
        for $actor = 1, 2, ...N$: **do**
            Utilizar la política $\pi_{\theta_{old}}$ en el ambiente para los T pasos.
            Calcular los estimadores $\hat{A}_1, ..., \hat{A}_T$
        end for
        Optimizar el substituto objetivo minimizando L wrt θ, en K entrenamientos y un minibatch de tamaño $M \leq NT$
    end for
end for

zando de acuerdo a la última versión de la política. La aleatoriedad de las acciones depende de las condiciones iniciales y el proceso de entrenamiento. A medida que se avanza en el entrenamiento, se espera que la política sea progresivamente menos aleatoria, dado que la regla de actualización ánima al agente a explotar las recompensas conocidas.

Por otro lado, se observa en 3 que el algoritmo para PPO, es considerablemente más rápido que TD3 y DDPG. Se asume esto, pues el algoritmo se salta T pasos y, luego de eso, realiza K entrenamientos. No así DDPG que entrena el actor en cada paso y TD3 cada d pasos. Es por ello que necesita un tiempo de ejecución considerablemente menor por episodio al compararlo con los otros algoritmos.

Capítulo 3

Estado del arte

3.1. Sistemas de control de HVAC

El problema de climatización no es nuevo, sino que es un problema que viene de hace bastante tiempo atrás y el cual ya ha sido abordado de distintas formas, donde una forma de abordarlo es utilizando métodos *model-based*, intentando modelar las dinámicas termales de una forma simplificada. Esto es aplicado tanto en métodos basados en reglas que se basan en una lógica del tipo "si ocurre - luego - sino", como controladores On/Off que son encontrados en edificios más antiguos. Mientras que en edificios más modernos se comenzó a utilizar otros métodos como PID (*Proportional Integrate Derivative*) y se aplica también a métodos basados en optimización como MPC (*Model Predictive Control*) y LQR (*Linear Quadratic Regulator*) [13].

Sin embargo, es complejo modelar de una forma correcta las dinámicas termales tomando en cuenta los variados factores que existen para esto. Por ejemplo, PID puede llevar a un punto de operación, sin embargo, ¿Qué ocurre cuando dicho punto de operación cambia? La respuesta es que el controlador deja de funcionar, por ello es que PID no es la mejor solución disponible para estos tipos de problemas tan dinámicos. En los casos de MPC y LQR se tiene que dado que, son controladores óptimos, no ocurrirá la misma situación que con el método PID, sin embargo, estos no podrán manejar de buena manera a una gran cantidad de factores, sobre todo, los parámetros inciertos[3][24].

En [25] se habla del posible control de un *Smart Home*, donde un control adecuado puede reducir significativamente el uso energético del hogar. Este hogar es un sistema integrado de una gran cantidad de objetos como los sensores, artefactos (como aire acondicionado, refrigeradores, lavadoras, etc), interfaz de usuario, etc.

Aún más relevante es que en [25] se habla de controlar el sistema con una gran variedad de algoritmos: uno de ellos es MPC. Se indica que este tiene muy buenos resultados, desempeñandose mejor que otros algoritmos, pero que depende mucho de la exactitud del modelo.

En este mismo trabajo, también se controla una parte del sistema con aprendizaje reforzado, específicamente, se aplica a agendar: el encendido/apagado de los aparatos, predecir la demanda y generación de las energías renovables del hogar.

Por otro lado, en los últimos años las implementaciones e investigaciones de control en sistemas HVAC han sido mayoritariamente en MPC y en algoritmos basados en aprendizaje, como lo son los algoritmos inteligentes donde los más comunes son aquellos que se basan en redes neuronales artificiales.

Siguiendo esta misma linea, para los controladores basados en MPC existen 3 aspectos críticos en su implementación, el primero sería qué tan preciso es el modelo que se utilizará en conjunto con la precisión al predecir perturbaciones, donde si no se tiene un modelo con un alto grado de exactitud, no se obtendrá un resultado óptimo. En segunda instancia, si se utiliza MPC como controlador en la capa supervisora este necesitará una gran cantidad de capacidad computacional. Y la tercera, es que cada edificio tiene arquitecturas distintas, por lo que habría que adaptar un MPC para cada edificio distinto para que este funcione de manera correcta[26].

En [27] se utiliza aprendizaje reforzado para controlar un sistema de un inmueble con el fin de entregar un ambiente agradable a los usuarios, ya sea en temperatura, calidad del aire o cantidad de luz. Además, este control también incluye el consumo de energía eléctrica. Se obtienen buenos resultados, tanto para el consumo de energía como para el *comfort* de las personas.

Smart grid es a grandes rasgos controlar la energía eléctrica de un hogar, tanto la generada por energías renovables (como paneles solares) como la que proviene de la empresa distribuidora de electricidad. En [28] se utilizan distintos algoritmos de aprendizaje reforzado para un *Smart grid*, estos algoritmos se centran en predecir la carga en el sistema (uso), respuesta a la demanda y ciber seguridad.

En [29] (*Deep Q Network*) se propuso un control de un sistema HVAC basado en aprendizaje reforzado, específicamente, el algoritmo DQN. El objetivo es ahorrar energía eléctrica mientras se mantiene la temperatura deseada en las habitaciones. Al usar 5 zonas, el uso de energía se puede reducir hasta en 35.1 %.

Es por ello que se propone utilizar aprendizaje reforzado, método el cual podría manejar la complejidad de las dinámicas termales.

3.2. Beneficios de aprendizaje reforzado

Tal como fue expuesto en la sección anterior, los métodos de control como PID, MPC y LQR presentan distintos problemas para controlar de buena manera una sistema HVAC. En el caso de PID, se expuso que es un tipo de controlador que puede llevar a un punto de operación, pero si dicho punto de operación cambia el controlador comenzará a funcionar de mala manera. Este punto de operación puede cambiar por influencias externas (en nuestro caso uso de la habitación o condiciones climáticas), o bien por cambios en el sistema (que inclusive pueden ser pequeños). Aprendizaje reforzado no tendría problemas para manejar estos pequeños cambios, debido a que al ir interactuando con el sistema constantemente, este sabe cual es el funcionamiento actual del sistema, también el agente inteligente puede ir observando las influencias externas con el objetivo de saber con mayor precisión que acción

tomar para mantener el estado deseado.

En el caso de MPC y LQR, su rendimiento depende de la exactitud del modelo utilizado, sin embargo, la complejidad matemática y computacional aumenta considerablemente a medida que mejoramos la exactitud del modelo. En el caso de la complejidad matemática, aprendizaje reforzado puede manejarla de mejor manera debido a que nuestro objetivo estará dado por una función de recompensa, cuyo diseño está orientado a cuales son los estados deseados. En el caso de la complejidad computacional, para aprendizaje reforzado estará dada por el tamaño de las redes neuronales, cantidad de observaciones y la cantidad de acciones, luego, en cuánto a gasto computacional, los distintos métodos aumentan el gasto a medida que crece la complejidad del sistema.

Además, si se implementa una solución que se encuentre en un entrenamiento constante, es decir, *online*, esta solución puede ir mejorando constantemente, además de ir adaptándose a las distintas condiciones de clima a las que se puede exponer el sistema.

Capítulo 4

Formulación del problema y diseño del método de control

En la siguiente sección se describirá la formulación del problema y, la formulación y diseño del método para solucionarlo, incluyendo la metodología utilizada. En relación con la formulación y diseño del método , también se verán las estructuras de las redes neuronales utilizadas. Dada la alta complejidad del problema, se toma la decisión de dividirlo en casos más simples e ir aumentando la dificultad poco a poco. Así se evitan errores al correr largas simulaciones del sistema sin la certeza de que funcionará correctamente.

4.1. Metodología

A continuación se expone la metodología que se utilizará para la realización del trabajo, identificando posibles riesgos y restricciones al utilizar dicha metodología.

Se propone una metodología basada en diseñar, implementar y evaluar. El primer paso a tomar es verificar el correcto funcionamiento de la simulación del sistema en *Matlab*, para luego estructurar el algoritmo de aprendizaje reforzado que trabajará entorno a él. Esto implica el diseñar la función de recompensa del sistema, para luego implementar el algoritmo en el software *Matlab* a modo de simulación, utilizando el *toolbox* de *reinforcement learning*.

Luego, se evalúan los resultados de forma cualitativa. La evaluación está ligada al consumo de energía eléctrica, a la forma que toman los estados y acciones del sistema y al cumplimiento de las restricciones de temperatura, CO_2 y humedad . Este proceso se repite varias veces para cada una de las instancias del problema que se proponen en este trabajo, hasta obtener agentes que en efecto controlen el sistema.

Finalmente, se comparan los distintos agentes entrenados mediante funciones de desempeño. Estas funciones de desempeño califican cuantitativamente el cumplimiento de restricciones, la recompensa obtenida y la energía utilizada.

A continuación se identifican los riesgos y restricciones de la metodología planteada. En cuanto a restricciones, se pueden identificar el tiempo y capacidad de cálculo computacional, dado que la metodología es similar a una 'prueba y error'. El proceso de entrenamiento en aprendizaje reforzado puede llegar a tomar varias horas por cada ejecución, e inclusive más si no se cuenta con una capacidad computacional adecuada. Se propone realizar este trabajo

con al menos una computadora que posea un procesador de 4 núcleos como mínimo. También se puede identificar como restricción la disponibilidad y exactitud de predicciones de clima, que son necesarias para llegar a un buen controlador.

Por otro lado, tenemos que los posibles riegos son, en primera instancia, que ninguno de los controladores diseñados no lleguen a una solución que sea de real utilidad para ninguna de las instancias propuestas, es decir, que las soluciones obtenidas por los controladores no sean óptimas. Para evaluar una posible solución óptima, nos basamos en como lucen cualitativamente las acciones y estados del sistema. Bajo este posible riesgo, se propone que, en el caso de no lograr la convergencia de los controladores mediante aprendizaje reforzado, se utilizará otro método para realizar el control del sistema, por ejemplo, controlar el sistema usando MPC o LQR.

4.2. Formulación del problema

En este trabajo se controla un sistema de climatización similar al mostrado anteriormente en la figura 2.1, sin embargo, se controlan diversos casos o instancias de este sistema.

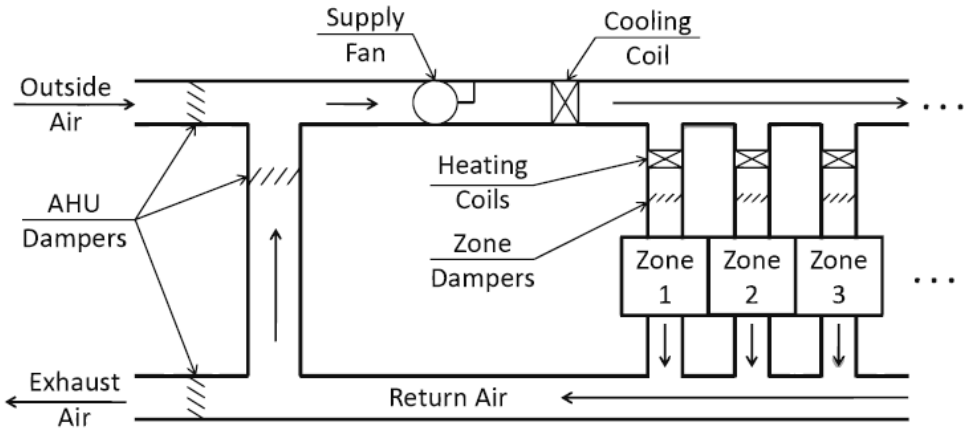


Figura 4.1: Esquema HVAC [3].

Tal como fue expuesto anteriormente, el sistema tiene 5 componentes de control: ventilador, enfriamiento, calefacción, válvulas de zona y válvulas AHU, donde tenemos que las válvulas de zona no son relevantes cuando estamos utilizando una única zona.

Cada una de estas componentes de control está limitada físicamente, donde los rangos posibles son[30]:

- Válvulas AHU(δ)= $[0, 1]$, que indica completamente abierto al tomar el valor 0 o completamente cerrado al tomar el valor 1. Donde el límite superior podría disminuir para garantizar que siempre esté entrando aire fresco al sistema.
- Ventilador(m_s)= = $[0, 5]kg/s$, donde el máximo está limitado por el tamaño de las cajas VAV. Mientras que el mínimo es impuesto, cero indica que el ventilador se encuentra

apagado y no hay circulación de aire. Este límite inferior podría cambiar para garantizar que siempre exista circulación de aire en el interior.

- Cooling coil(T_c)= $[-30, 0]$ ºC, es la temperatura que puede disminuir el aire de entrada.
- Heating coil(T_{h_i}) = $[0, 8]$ ºC, el incremento de temperatura al aire de entrada de la i habitación.
- Válvulas de zona = $[0, 1]$, al contrario de las válvulas AHU, 0 indica completamente cerrado y 1 indica completamente abierto. Además como su nombre lo dice, existe una válvula para cada habitación.

Utilizando estas 5 componentes de control se propone controlar 3 restricciones distintas que son: Temperatura, Concentración de CO_2 y humedad relativa, al interior de cada zona.

Los rangos que se utilizarán como adecuados para las restricciones son aquellos que se consideran empíricamente como cómodos, en el caso de temperatura, y en el caso de CO_2 sobre ese rango comienzan a existir efectos sobre la salud de las personas. Los rangos escogidos son:

- Temperatura: $[22, 25]$ºC.
- Concentración CO2: $[0, 1200]$ ppm.
- Humedad relativa: $[35, 65]$ %.

En vez de controlar el sistema con todas sus componentes de control y las 3 restricciones, se realizan distintas instancias del problema con distintos niveles de complejidad. Las instancias del problema y sus características se muestran en la tabla 4.1. Estas distintas instancias se escogen basándose en posibles combinaciones de estos sistemas, que pueden requerir el controlar solo temperatura, o bien incluir humedad y/o CO_2.

Tabla 4.1: Tabla que indica las componentes y restricciones de las distintas instancias.

Instancia	nºzonas	Componentes de control	Restricciones
1	1	Enfriamiento	Temperatura
2	1	m_s	Temperatura
3	1	T_c y m_s	Temperatura
4	1	T_c, m_s y válvulas AHU	Temperatura
5	1	T_c, m_s, δ y calefacción	Temperatura y CO_2
6	1	T_c, m_s, δ y T_h	Temperatura, CO_2, y humedad
7	4	T_c, m_s, δ, T_{h_i} y válvulas de zona	Temperatura
8	4	T_c, m_s, δ, T_{h_i} y válvulas de zona	T., CO_2 y Hum.

4.3. Formulación y diseño del método de control

Para formular y diseñar el método de control nos guiaremos por el esquema mostrado en la figura 4.2, pues este define el método sin incurrir en la configuración de los algoritmos en

particular.

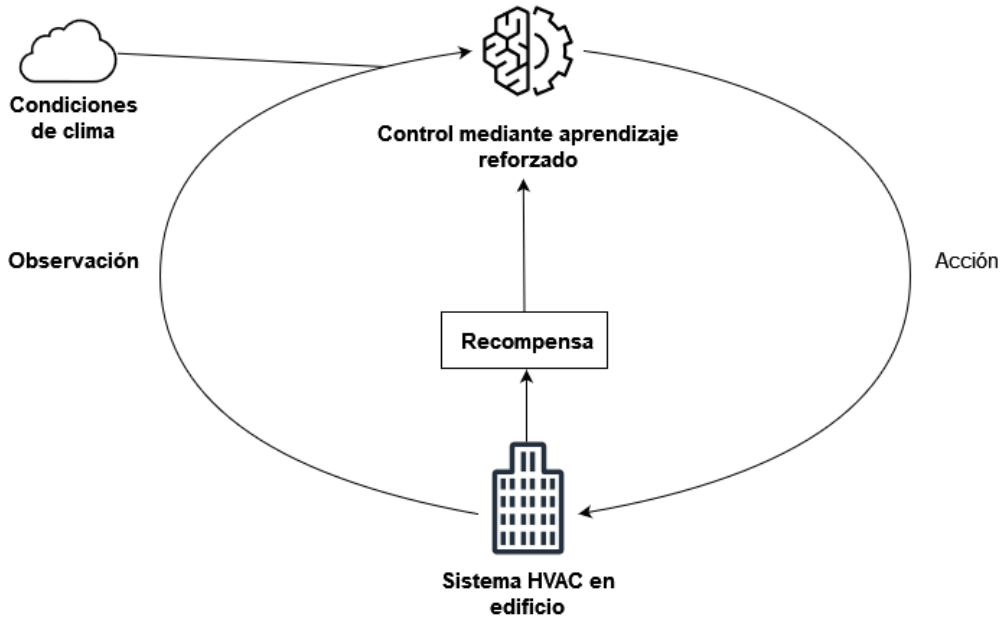


Figura 4.2: Esquema del funcionamiento de aprendizaje reforzado.

4.3.1. Observaciones

Las observaciones cumplen un rol fundamental en el entrenamiento de las redes neuronales del sistema de control, dado que son las entradas para el actor y además es una de las entradas del crítico. Luego, dado el sistema descrito anteriormente en el capítulo 3, las variables que nos interesan observar son:

- Temperaturas en las zonas (del aire y sólidos).
- Porcentaje de humedad en cada zona.
- Concentración de CO_2 en cada zona.
- Cantidad de calor generado por uso en cada zona.
- Humedad y CO_2 generado en cada zona.
- Temperatura exterior.
- Humedad y concentración de CO_2 exterior.

Pues estas son o bien las variables a controlar, o bien factores que influyen en ellas, como es el caso de las predicciones de las variables ambientales y ocupación. Cabe destacar que estas mediciones son todas en el mismo instante, que se puede denominar como k.

En la tabla 4.2 se muestran las observaciones utilizadas para cada una de las instancias propuestas. Donde T son las temperaturas de las zonas, CO_2 es la concentración de CO_2 por zona, hum es el porcentaje de humedad por zona, P_d el calor generado, $CO_{2_{gen}}$ el CO_2 generado, H_{gen} la humedad generada, T_a la temperatura ambiente, $CO_{2_{amb}}$ el CO_2 ambiental

y H_a la humedad ambiental. En las instancias que solo tenemos restricciones de temperatura, se observan la temperatura en la habitación, la temperatura ambiental y la generada. Por otro lado, en las instancias en que se agrega CO_2, se agrega a lo observado para temperatura, la concentración de CO_2 en la habitación, el CO_2 generado y la concentración ambiental. Finalmente, ocurre esto mismo cuando se agrega la humedad, es decir, además de lo observado en la instancia de CO_2, se agrega la observación de la humedad en la habitación, la humedad generada y la humedad ambiental.

Tabla 4.2: Observaciones para cada instancia, se marca con una x las que son utilizadas por la instancia.

Instancia	T	CO_2	hum	P_d	$CO_{2_{gen}}$	H_{gen}	T_a	$CO_{2_{amb}}$	H_a
1	x			x			x		
2	x			x			x		
3	x			x			x		
4	x			x			x		
5	x	x		x	x		x	x	
6	x	x	x	x	x	x	x	x	x
7	x			x			x		
8	x	x	x	x	x	x	x	x	x

4.3.2. Acciones

Tal como se observa en la tabla 4.1 cada una de las instancias tiene acciones de control distintas.

A pesar de que se tengan claras las componentes de control, es necesario identificar también la cantidad de acciones que controlará el sistema y cuántas se dejarán constantes. Esto se ve en la tabla 4.3.

Tabla 4.3: Cantidad de acciones controladas y no controladas (constantes) por instancia.

Instancia	Componentes de control	nº acc.	nº cons.
1	Enfriamiento (T_c)	1	4
2	flujo (m_s)	1	4
3	T_c y m_s	2	3
4	T_c, m_s y válvulas AHU (δ)	3	2
5	T_c, m_s, δ y calefacción (T_h)	4	1
6	T_c, m_s, δ y T_h	4	1
7	T_c, m_s, δ, T_h y válvulas de zona	11	0
8	T_c, m_s, δ, T_h y válvulas de zona	11	0

En la tabla 4.3, se muestra la cantidad de acciones controladas por el agente y la cantidad de acciones constantes. En particular, para una zona tenemos un máximo de 5 acciones,

mientras que para n zonas, las variables T_h y válvulas de zona se vuelven un vector de tamaño n, es por ello que las instancias 7 y 8 tienen 11 acciones controladas.

4.3.3. Ambiente

El ambiente en este caso es el sistema HVAC. El modelo del sistema que fue presentado anteriormente no es exactamente el que se utiliza, debido a que utiliza un sistema discretizado con el fin de disminuir la carga computacional y con ello el tiempo que demora la simulación del sistema y el entrenamiento de los algoritmos.

Además, a pesar de solo utilizar métodos de aprendizaje *model-free*, se usa un modelo del sistema pues estamos trabajando netamente con simulaciones (no poseemos un sistema real en el momento) y, además, porque uno de los objetivos de este trabajo es realizar una comparación entre los distintos algoritmos, donde cada entrenamiento toma un tiempo de 6 a 8 horas. Si fuese con un sistema real, los entrenamientos podrían llevar semanas por cada uno. Por otro lado, siempre podemos utilizar un modelo, por ejemplo, en el caso de querer implementarlo en un sistema real el modelo puede ser de utilidad para pre-entrenar el agente, con el objetivo de que este ya tenga un buen funcionamiento al momento de ser implementado, y que su entrenamiento *online* sea de utilidad solo para pulir este buen funcionamiento.

Luego, la dinámica de la temperatura que realmente se utiliza está dada por la discretización de la dinámica continua, utilizando el método de Euler [31], la que se ve representada en las ecuaciones (4.1), (4.2) y (4.3)

$$T_k^{i+1,j} = T_k^{i,j} + \frac{\Delta t \cdot c_p}{C_1}\dot{m}_s^j(T_s^j - T_1^j) + \frac{\Delta t(T_2^j - T_1^j)}{C_1 R^j} + \frac{(T_a - T_1^j)}{C_1 R_a^j} + \frac{\Delta t P_d^j}{C_1} + \sum_{i \in N^j} \frac{\Delta t(T_1^i - T_1^j)}{C_1 R_{ij}}, \tag{4.1}$$

$$T_2^{i+1,j} = T_2^{i,j} + \frac{\Delta t(T_1^{i,j} - T_2^{i,j})}{C_2^i R_{12}^j}, \tag{4.2}$$

$$T_s^j = \delta \frac{\sum_{j \in R} \dot{m}_s^j T_1^j}{\sum_{j \in R} \dot{m}_s^j} + (1-\delta)T_a - \Delta T_c + \Delta T_h^j, \tag{4.3}$$

donde Δt es el tiempo de muestreo del sistema, el cual toma el valor $\Delta t = 60[s]$. Además, en el modelo discreto, se añade un súper índice 'i' el cual indica el tiempo. Cabe destacar que para el modelo se utiliza un tiempo de muestreo distinto al del controlador, en el modelo se utiliza un tiempo de muestreo menor que en el controlador, con el fin de ser más precisos en la simulación del sistema.

Esta dinámica de la temperatura es utilizada en todas las instancias, pues, al ver la tabla 4.1, todas tienen a la temperatura como restricción.

Por otro lado, la dinámica de la concentración de CO_2 es sólo utilizada en las instancias 5, 6 y 8. Esta dinámica está representada en las ecuaciones (4.4), (4.5).

$$C_{CO_2}^{i+1,j} = C_{CO_2,k}^{i,j} + \frac{\Delta t}{\rho V^j}(\dot{m}_s^{i,j}(C_{CO_2,s} - C_{CO_2}^j) + g_{CO_2} N_{people}), \tag{4.4}$$

$$C_{CO_2,s} = \delta \frac{\sum_{j \in R} \dot{m}_s^j C_{CO_2}^j}{\sum_{j \in R} \dot{m}_s^j} + (1 - \delta) C_{CO_2,a}. \tag{4.5}$$

Hay que tener mucho cuidado con este modelo pues para un tiempo de muestreo considerablemente grande, tenemos que si la concentración de CO_2 del aire de entrada es menor que la de habitación en la ecuación 4.4, entonces podríamos obtener una concentración negativa.

Para evitar que esto último ocurra, se hace la modificación 4.6

$$C_{CO_2}^{i+1,j} = \frac{\Delta t}{\rho V^j} g_{CO_2} N_{people}^j + \begin{cases} C_{CO_2,k}^{i,j} + \frac{\Delta t}{\rho V^j}(m_s^{i,j}(CO_2, s - C_{CO_2}^j)), & \text{si la suma anterior} \geq CO_2, s \\ CO_2, s, & \text{sino} \end{cases} \tag{4.6}$$

En palabras, si la suma del estado anterior con la cantidad que disminuyó por la cantidad de aire fresco ingresado en una muestra de tiempo, es menor a la concentración de CO_2 que ingresa, entonces tendremos que el estado será la concentración de CO_2 del aire que ingresa(podríamos decir que se reemplazó todo el aire que había en la habitación), dado que la concentración no puede disminuir más que el mínimo entre el estado anterior y la concentración de CO_2 de entrada.

Finalmente, la dinámica de la humedad absoluta se utiliza en las instancias 6 y 8, y queda representada en la ecuación 4.7.

$$\dot{H}^{i+1,j} = H^{i,j} + \frac{\Delta t}{\rho V^j} \cdot \dot{m}_s^j (H_s - H^{i,j}) + \frac{\Delta t}{\rho V^j} \cdot g_H N_{people} \tag{4.7}$$

A esta dinámica es necesario hacer la misma modificación que para el caso de CO_2, por las mismas razones, quedando la modificación como se muestra en la ecuación (4.8)

$$\dot{H}^{i+1,j} = \frac{\Delta t}{\rho V^j} \cdot g_H N_{people} + \begin{cases} H^{i,j} + \frac{\Delta t}{\rho V^j} \cdot \dot{m}_s^j (H_s - H^{i,j}), & \text{si la suma anterior} \geq H_s \\ H_s, & \text{sino} \end{cases} \tag{4.8}$$

4.3.4. Recompensa

Se tiene que la recompensa está basada en la expresión (2.12). Donde tal y como se dijo anteriormente, β corresponde a una constante negativa, la cual es útil para sintonizar el peso que tendrá el gasto energético versus los estados, mientras que E es la cantidad física de energía consumida, además, Δx pertenece a la penalización por romper la restricción del estado.

Luego, dado que la función de recompensa depende de las restricciones impuestas, tendremos 5 variantes en total.

La primera variante es aplicable sólo al primer caso, pues penalizamos solo el estar fuera de los estados, pero sin considerar energía. Luego, la función cuando se rompen las restricciones queda expresada en (4.9), donde se considera $\beta = 0$.

$$r(t) = min(|T - T_{min}|, |T - T_{max}|), \tag{4.9}$$

donde T es la temperatura del aire en la habitación, $T_{min} = 22$ y $T_{max} = 25$, como fue definido en los rangos. Cuando no se rompe la restricción, la función entrega el valor 0.

La segunda variante es aplicable a la instancia 2, donde se agrega la energía. Tomando el cambio de variable $w_t = min(|T - T_{min}|, |T - T_{max}|)$, la expresión para la segunda variante queda representada en la ecuación (4.10).

$$r(t, E) = -E \cdot \beta - \Delta T, \tag{4.10}$$

donde w_t es distinto de cero sólo si se rompe la restricción de temperatura, $\beta = 0.1$ y E está dada por la expresión (4.11)[32] [33].

$$E = \frac{c_p}{\eta_c} \cdot m_{s,k} \Delta T_k + \kappa_f (\sum_i \cdot m^i_{s,k})^2. \tag{4.11}$$

La tercera variante es aplicable a las instancias 3 y 4. Acá se penaliza aún más el romper la restricción de temperatura con el fin de mejorar la estabilidad en el entrenamiento. La expresión de la recompensa se muestra en la ecuación (4.12).

$$r(t, E) = -0.1 \cdot E + \gamma \cdot (w_t + w_t^2), \tag{4.12}$$

con $\gamma = 0.5$. Se aplica esta variación en la penalización de la temperatura con el fin de ser menos estrictos si $w_t \in [0, 1]$ y penalizar aún más si $w_t \in (1, \infty)$.

Para la cuarta y quinta, se utiliza la expresión de energía representada en las ecuaciones (4.13) [32].

$$E = \frac{c_p}{\eta_c} \cdot (\sum_j \cdot m^j_{s,k}) \Delta T_k + \frac{c_p}{\sum_j} (\eta^j_h \cdot m^j_{s,k}) \Delta T_h + \kappa_f (\sum_i \cdot m^j_{s,k})^2, \tag{4.13}$$

con j indicando a que habitación pertenece.

Luego, la cuarta expresión añade la restricción por CO_2 y se aplica a la instancia 5, quedando expresada en la ecuación (4.14).

$$r(t, E, CO_2) = -0.1 \cdot E + \gamma \cdot (w_t + w_t^2) + 10^4 \cdot (1200 \cdot 10^{-6} - CO_2). \tag{4.14}$$

Para la quinta expresión, se añade la restricción de humedad relativa, donde la expresión (4.14) queda expresada en la ecuación (4.15).

Dado lo anterior, se propuso añadir una penalización cuando se rompe la restricción de humedad a la expresión (4.14) la cual es:

$$r(t, E, CO_2, hum_1) = r(t, E, CO_2) - min(|hum_1 - 0.35|, |hum_1 - 0.65|). \tag{4.15}$$

Para la instancia 7, tenemos que es una combinación de la tercera variante (4.12) con la energía expresada en (4.13). Mientras que la instancia 8, es similar a la 6, solo que se penalizan los estados por zona.

Además, se plantea la posibilidad de implementar una penalización por tomar acciones de control con cambios muy repentinos en el caso de que el controlador no se desempeñe de manera correcta; en casos como, por ejemplo, apagar y prender el ventilador al máximo en cada instante de control. Esto con el fin de garantizar el tener una solución óptima y que, además, no perjudique la vida útil del sistema de climatización.

Cabe destacar que el diseño de la función de recompensa es la característica más desafiante de aprendizaje reforzado, dado que es con esta función que el agente aprenderá a tomar decisiones correctas, por lo que si la función no está bien formulada, puede llevar a un mal entrenamiento y, por lo tanto, a acciones de control no deseadas[34].

4.4. Configuración de los algoritmos

Esta sección se dividirá en 3 partes, donde cada una de ellas explicará la conformación de la redes neuronales e hiper parámetros a utilizar para cada algoritmo.

A modo de configuración general, para todos los algoritmos e instancias se consideró un tiempo de muestreo $T_s = 600[s]$, es decir, de $10[min]$. Además en la tabla 4.4, se observan los largos de los episodios para cada instancia en tiempo y cantidad de pasos. Se utiliza $T_s = 600[s]$ en vez de $T_s = 60[s]$ que era el tiempo de muestreo para el ambiente, pues un minuto es un tiempo muy corto para que la acción pueda afectar el ambiente en una dinámica como la temperatura, en ese sentido, también se utiliza $T_s = 60[s]$ para el ambiente con el fin de que este sea más preciso en el estado actual. El largo de los episodios fue escogido

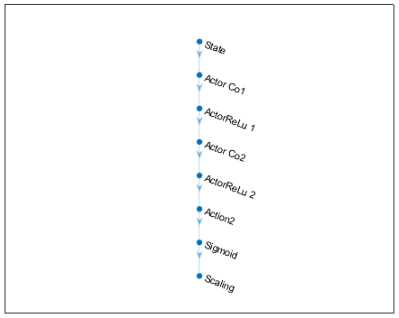
Tabla 4.4: Largo de los episodios por instancia.

Instancias	Tiempo [h]	Pasos
1 a 4	6	36
5 a 7	48	288

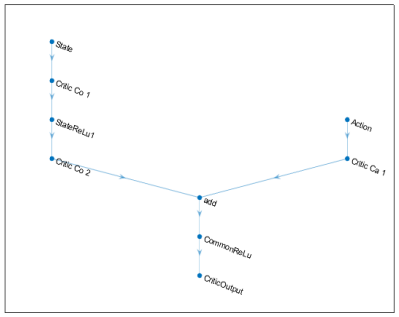
basándose en que en [13] se utiliza un día (24 horas) como tiempo de simulación y 30 minutos como tiempo de muestreo, entregando así 48 pasos por cada episodio. En [35] se utiliza un tiempo de muestreo de 15 minutos, mientras que cada episodio dura dos meses, es decir, existen 5760 pasos por episodio.

4.4.1. DDPG

Tal como se menciona en la sección 3.3, tenemos que el agente de DDPG está conformado por dos partes fundamentales, el Crítico y el Actor, donde ambas están conformadas por redes neuronales. En este trabajo se definen una cantidad distinta de neuronas dependiendo de la instancia que se está entrenando. Además se define un distinto tamaño para el *minibatch* en los casos más complejos.
Por parte del crítico tenemos que este recibe los estados y las acciones (adquiridos del *minibatch*) en caminos distintos, donde se utiliza una capa oculta para las acciones (C_a) y dos para las observaciones (C_o). En el caso del Actor es un camino único con dos capas ocultas, como se observa en la figura 4.3.b. La configuración utilizada para el algoritmo se observa en la tabla 4.5.



(a) Esquema de la red neuronal para el actor en DDPG y TD3.



(b) Esquema de la red neuronal para el critico en DDPG y TD3.

Figura 4.3: Esquemas de las redes neuronales.

Tabla 4.5: Configuración de la red según instancias para DDPG.

Instancias	C_o	C_a	Actor	Activación	Salida	minibatch
1	[16, 8]	8	[6, 3]	*ReLu*	*Sigmoid*	128
2	[16, 8]	8	[6, 3]	*ReLu*	*Sigmoid*	128
3	[32, 16]	16	[16, 8]	*ReLu*	*Sigmoid*	128
4	[64, 32]	32	[16, 8]	*ReLu*	*Sigmoid*	128
5	[128, 128]	128	[128, 128]	*ReLu*	*Sigmoid*	256
6	[128, 128]	128	[128, 128]	*ReLu*	*Sigmoid*	256

Se tomó en consideración lo dicho en [36], donde nos dicen que las funciones *sigmoid* y *tanh* pueden ocasionar problemas al ser utilizadas como función de activación de capas ocultas, quedando así las funciones *relu* en dicha posición.

La función de activación *sigmoid* se utiliza ya que entrega una salida en el rango $[0, 1]$, por lo que nos es bastante útil dadas las cotas de nuestras acciones, ya que para todas al menos una de sus cotas es 0. Luego, dado que la salida estará entre 0 y 1, hay que multiplicar la salida por $max_{action} + min_{action}$ para así escalarla al rango deseado. Es exactamente esto ultimo lo que hace la capa para escalar o *scaling layer* que se agrega después de la función *sigmoid*, esto se ve representado en la figura 4.3.a.

4.4.2. TD3

En términos de estructura de red, para TD3 se utiliza la misma estructura que para DDPG, tanto en la estructura del actor como del crítico. La diferencia principal es que TD3 utiliza dos redes de crítico, sin embargo, la estructura de la red es igual, no así los pesos de las neuronas.

A parte de la diferencia anteriormente mencionada, el resto de la estructura de la red es exactamente la misma: misma cantidad de capas, neuronas y funciones de activación y de salida. Por lo que la tabla 4.5 también aplica para TD3, con la salvedad de que TD3 controla 2 instancias más que se ven en la tabla 4.6.

Tabla 4.6: Configuración de la red para instancias controladas solo por TD3.

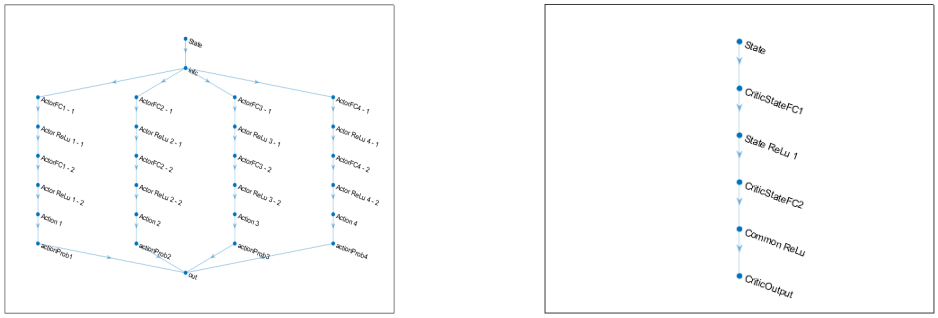
Instancias	C_a	C_o	Actor	Activación	Salida	minibatch
7	[400, 300]	300	[400, 300]	*ReLu*	*Sigmoid*	256
8	[400, 300]	300	[400, 300]	*ReLu*	*Sigmoid*	256

4.4.3. PPO

Para PPO fue necesario cambiar la estructura de la red neuronal anterior. Esto se debe en mayor medida a que PPO utiliza métodos estocásticos para realizar exploración, mientras que DDPG y TD3 utilizan métodos deterministas. Bajo ese argumento, se mantuvo la estructura del crítico de manera similar con la salvedad de que en PPO el crítico recibe las observaciones mas no las acciones. Por otro lado, la estructura del Actor cambió considerablemente.

Para poder construir la red neuronal del algoritmo de PPO, nos basamos en los ejemplos dispuestos por matlab en su página [37], preguntas en el foro de matlab[38](con el fin de tener más de una acción) y un trabajo previo[35] el cual nos fue útil para definir de manera correcta los hiper parámetros.

La configuración de la red se presenta en la tabla 4.7, donde las neuronas del actor son para cada acción, pues tenemos un 'camino' por cada acción, como se puede observar en la figura 4.4.a.



(a) Esquema de la red neuronal para el actor en PPO.

(b) Esquema de la red neuronal para el critico en PPO.

Figura 4.4: Esquemas de las redes neuronales para PPO.

Tabla 4.7: Configuración de la red según instancias para PPO.

Instancias	C_o	Actor	Activación	Salida	T	minibatch
1	[16, 8]	[16, 16]	*ReLu*	*Softmax*	50	36
2	[16, 8]	[16, 16]	*ReLu*	*Softmax*	50	36
3	[32, 32]	[16, 16]	*ReLu*	*Softmax*	50	36
4	[64, 64]	[16, 16]	*ReLu*	*Softmax*	50	36
5	[64, 64]	[32, 32]	*ReLu*	*Softmax*	300	256
6	[128, 128]	[32, 32]	*ReLu*	*Softmax*	300	256

Capítulo 5

Resultados

En este capítulo se exponen los resultados de las simulaciones realizadas en distintas configuraciones del sistema. Se parte abordando el caso más simple pasa pasar a los casos más elaborados. Se realiza de esta manera con el fin de explorar distintas configuraciones de un sistema de climatización y para evitar encontrarnos con un problema de diseño muy grande al avanzar considerablemente en complejidad. Además, se muestran los resultados obtenidos para cada algoritmo en términos de las acciones realizadas y estados conseguidos. Con ello se realiza una comparación cualitativa entre los algoritmos para, finalmente, analizar cuantitativamente en términos de las recompensas obtenidas y el gasto energético asociado.

5.1. Resultados para cada instancia

5.1.1. Primera instancia, control de temperatura de enfriamiento

El caso más simple a abordar consta de solo una de las variables disponibles, la cual es la temperatura de enfríamiento, mientras que la valvula AHU se mantiene en $\delta = 0.8$, el flujo de aire en $ms = 2.5$ y la calefacción se puede considerar en 0. Además no se considera el consumo energético, pues esta instancia es tomada como la más simple, con el fin de tener una prueba de que el algoritmo funciona en el caso más simplificado, es por ello que en expresión (2.12) tendrá un $\beta = 0$.

Para abordar este problema, dado que es una situación bastante simple, se utiliza una red neuronal pequeña, con el actor utilizando 2 capas, una de 6 y otra de 3 neuronas, mientras que el critico tiene 2 caminos, uno con 2 capas de 16 y 8 neuronas y otro con 8 neuronas.

Luego de un corto periodo de tiempo el resultado de la curva de aprendizaje es el que se muestra en la figura 5.1.

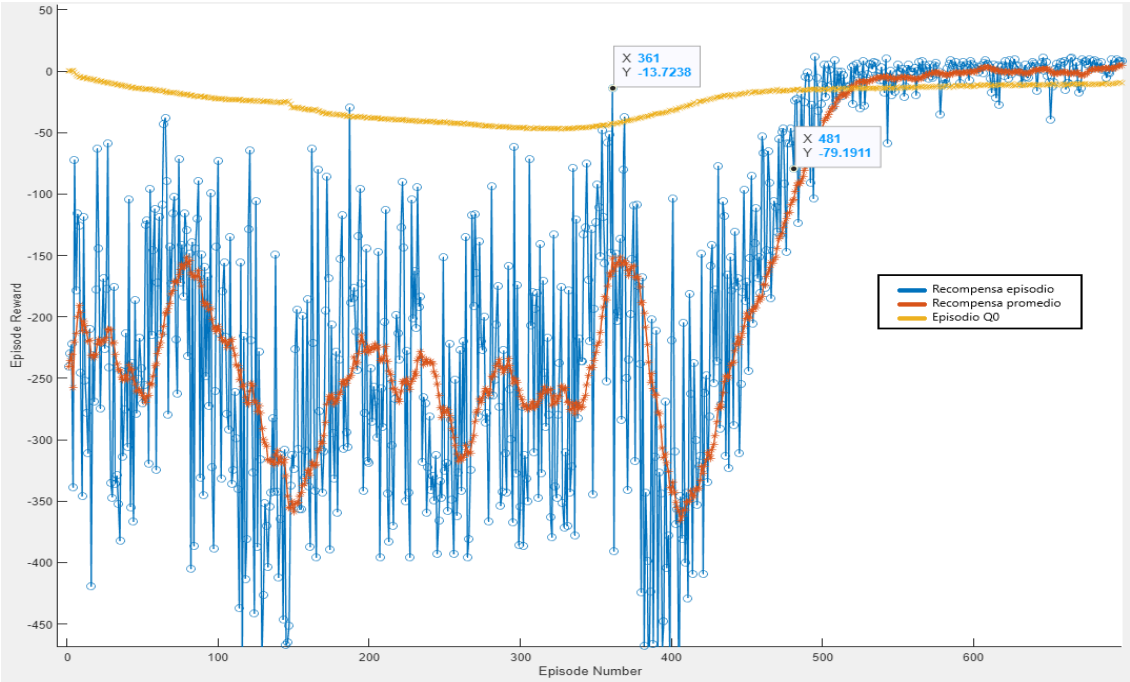
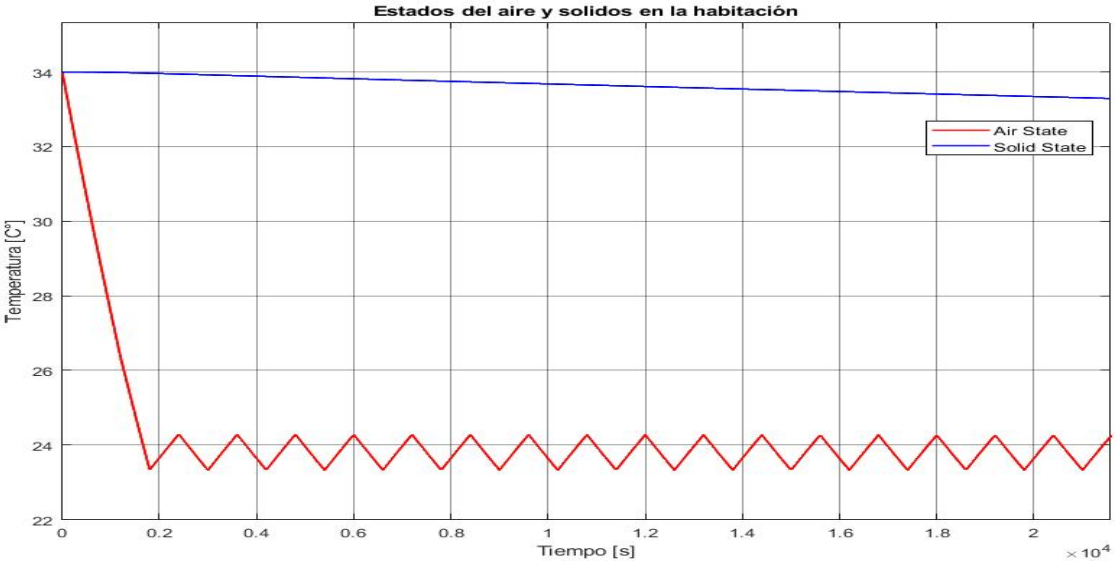
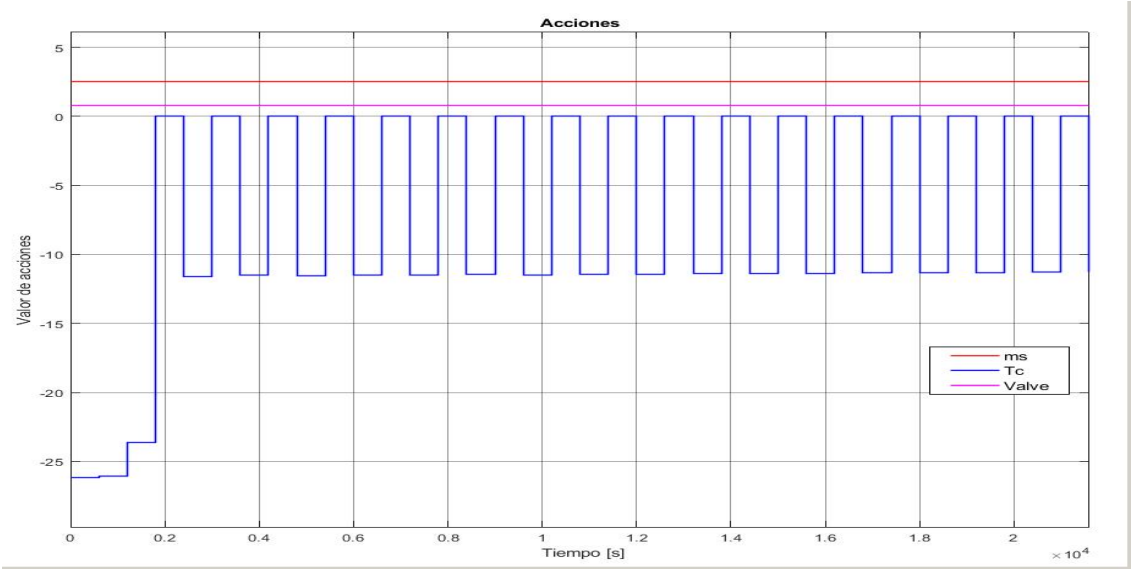


Figura 5.1: Curva de aprendizaje del caso más simple. Líneas azules representan la recompensa en el episodio, la naranja representa un promedio entre las últimas 25 recompensas obtenidas y la línea amarilla representa a una estimación de la política actual.

Donde se observa que, aproximadamente en el episodio 500, el agente aprende como puede obtener una buena recompensa y explora cerca de ese punto para mejorar aún más. Por otro lado las acciones y estados conseguidos con dicha curva de aprendizaje se ven representadas en la figura 5.2.



(a) Estados para el caso más simple utilizando DDPG.



(b) Acciones para el caso más simple utilizando DDPG.

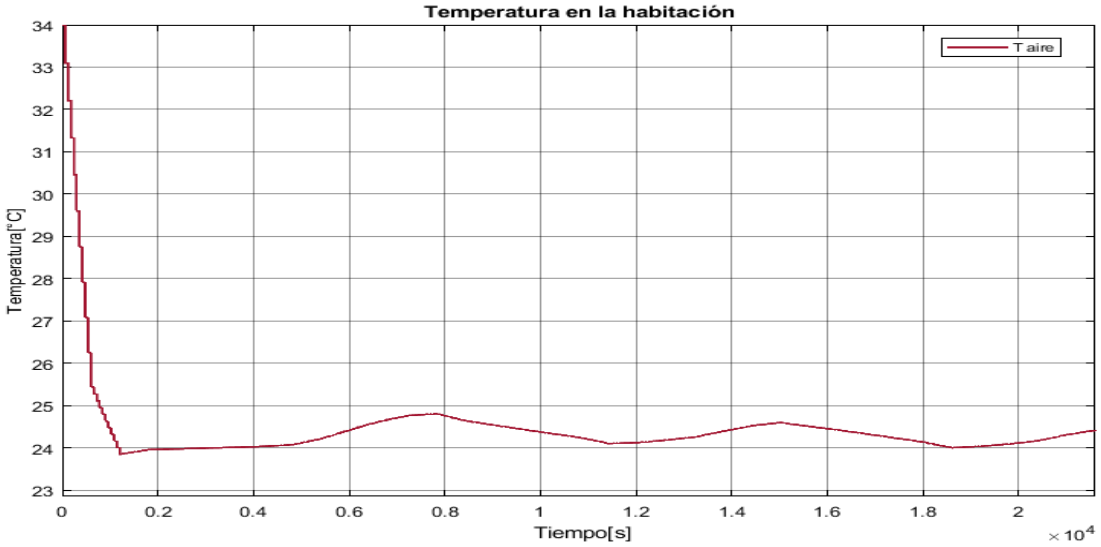
Figura 5.2: Resultados para el caso más simple, al controlar solo el flujo de aire.

Luego, la acción de control lleva al sistema a un estado deseado, pues este se encuentra sobre los 22°C y bajo los 25°C. Sin embargo, las acciones de control podrían no ser las más deseadas, pues implica un cambio abrupto de carga sobre el actuador. Por otro lado, se espera que al incluir el consumo energético estos cambios abruptos puedan disminuir. En caso de que esto no ocurra, se tendrá que implementar una tercera componente en 2.12 que implique una penalización por realizar este tipo de acción. En este caso la temperatura exterior se consideró constante en 40°C, se considera esta temperatura externa para poner el sistema bajo una alta carga, y verificar que efectivamente realiza la acción de enfriar el aire ingresado.

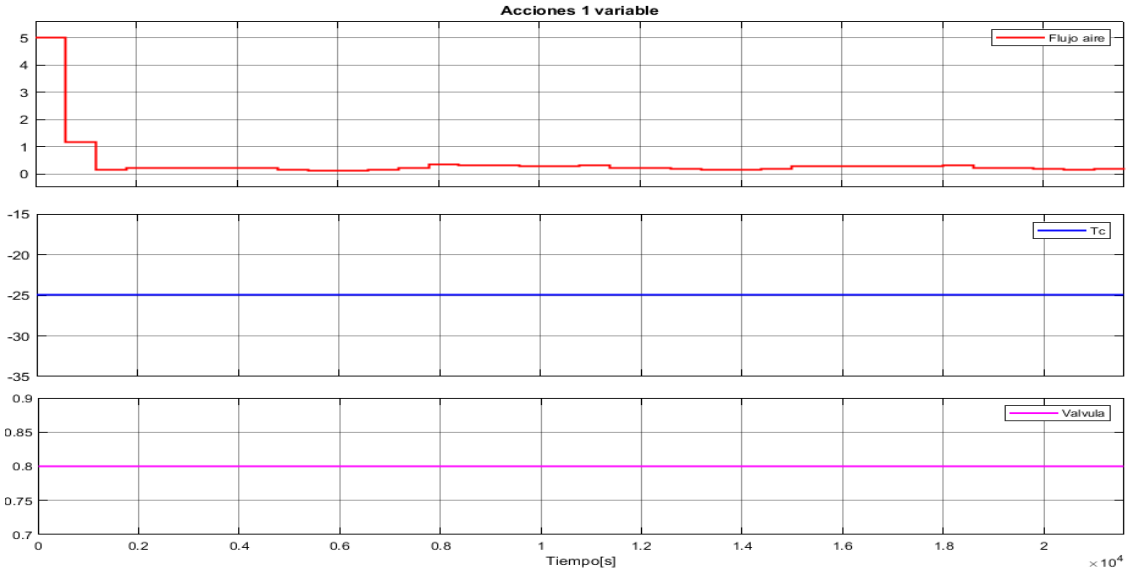
Dado que el agente no posee otra guía más que el permanecer dentro del rango de temperaturas, cualquier tipo de acción que mantuviera la mayor cantidad de tiempo posible el estado deseado, sería en efecto, una acción deseada.

5.1.2. Segunda instancia, control de flujo de aire

Dado los buenos resultados de la instancia anterior, se toma la decisión de realizar este mismo caso sencillo, pero esta vez utilizando solo el flujo másico de aire como variable a controlar. Mientras que se mantiene la variable de enfriamiento(ΔTc) constante en -25. Donde los resultados para dicha configuración se muestran en la figura 5.3.



(a) Estados para el caso más simple, controlando flujo de aire.



(b) Acciones para el caso más simple, controlando flujo de aire.

Figura 5.3: Resultados para el caso más simple, al controlar solo el flujo de aire.

En este caso, además de cambiar la variable de control, también se incluyó la penalización por uso de energía como fue expuesto anteriormente en la expresión (4.10), esto se hace con el objetivo de mejorar las acciones oscilantes que se obtuvieron en las primera instancia.

Para la expresión de energía dada en (4.11) se considera $c_p = 1012[J/kg \cdot K]$ el calor específico del aire, $\eta_c = 4$ la eficiencia al enfriar y $\kappa_f = 65$ la eficiencia del ventilador.

Luego tenemos que esta expresión es válida si consideramos una única zona, como lo estamos haciendo en este caso. Además, se tiene el primer sumando es la energía utilizada al enfriar, mientras que el segundo sumando es la energía utilizada para generar el flujo de aire.

Luego podemos observar en la figura 5.3 que al aplicar energía, el estado se mantiene más cercano al límite superior de la restricción, pues varía en el rango $[24, 25]$. Esto se debe a que la temperatura exterior es mayor a la deseada, por lo que enfría a penas lo suficiente para estar dentro de la restricción, esto se ve reflejado en la acción realizada, donde se tiene que se toman muy pequeñas acciones que son cercanas a cero, esto con el objetivo de no enfriar más de lo necesario.

Finalmente, se entrena esta misma configuración utilizando TD3, pero dada la simpleza del problema se obtuvieron resultados cualitativamente similares, con el agente controlando el flujo de aire entre 0 y 1 manteniendo así la temperatura cercana a los $25°C$, es por ello que no se incluyen.

5.1.3. Tercera instancia, controlando flujo de aire y enfriamiento

A continuación, se muestran los resultados obtenidos al entrenar un agente con 2 variables. Para esta configuración la válvula se queda fija en $\delta = 0.9$, indicando que se encuentra lo más cerrada posible, mientras que se controlan las variables de flujo de aire y enfriamiento. Al igual que en el caso anterior, esta simulación también se encuentra considerando el gasto energético.

Por otro lado, en este caso se utiliza una temperatura externa variable y mayor a los $25°C$ deseados para el entrenamiento, la cual se observa en la figura A.1.

Pese a que los resultados para una variable fueron bastante satisfactorios, para el caso con dos variables se obtienen buenos resultados, sin embargo, inestables a la hora de entrenar. Esto se puede ver en la curva de entrenamiento de la figura 5.4.

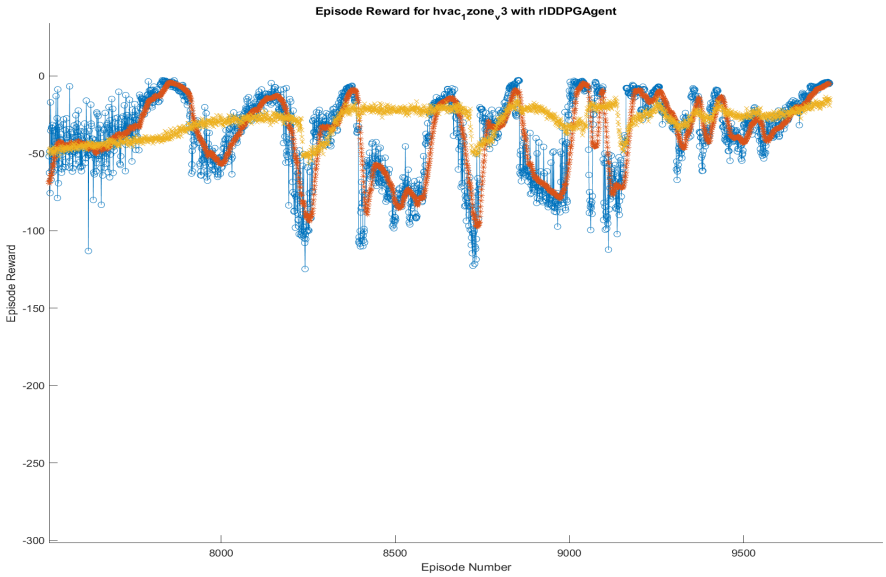


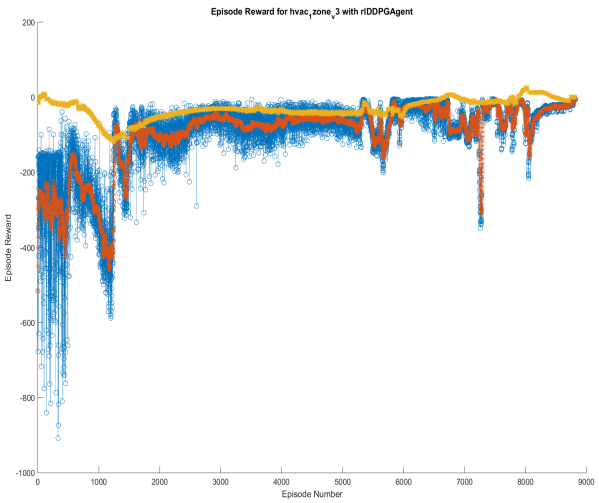
Figura 5.4: Curva de entrenamiento para dos variables.

Donde se observa una clara inestabilidad y oscilación entre casos buenos y otros considerablemente malos (recompensas altas y bajas) , por lo que podríamos considerar que el hecho de

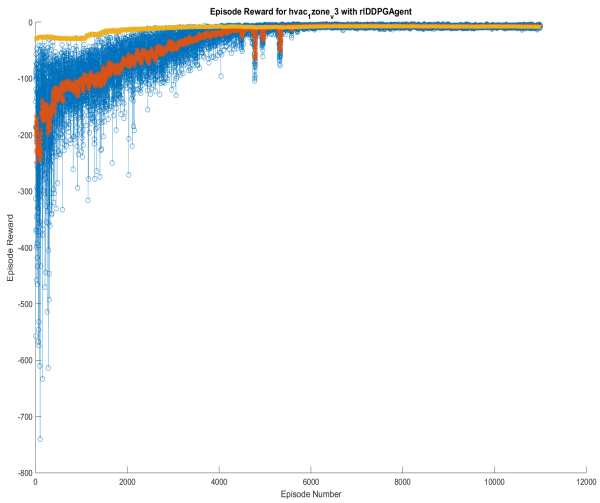
haber terminado la simulación en un caso donde efectivamente se controla de buena manera, fue un hecho fortuito en vez de que la red fuera aprendiendo constantemente de sus errores. Son estas oscilaciones las que nos motivan a modificar la función de recompensa anteriormente expresada 2.12 y, además, de investigar las posibles causas de este comportamiento. Cabe destacar que dado que la función de recompensa está basada en la penalización de estados no deseados, hay 2 posibilidades de recompensas, una recompensa negativa y una recompensa igual a cero, esta última es el caso ideal y realmente no se cumple, pues necesitamos utilizar energía para llevar al estado deseado, luego, las recompensas deseadas son las más cercanas a este punto ideal, es decir, cero.

La primera medida que se tomó para mejorar el comportamiento mostrado en la figura 5.4, fue modificar la función de recompensa, entregando una mayor penalización al estar fuera del estado deseado, representada en 4.12.

Realizar este cambio en la función de recompensa mejoró levemente la estabilidad en el aprendizaje como se ve en el figura 5.5.a, lo que motiva a seguir investigando una posible solución a esta problemática.



(a) Curva de entrenamiento al cambiar la función de recompensa.



(b) Curva de entrenamiento al cambiar la función de recompensa y la estructura de la red neuronal del actor.

Figura 5.5: Curvas de entrenamiento al aplicar los dos cambios descritos.

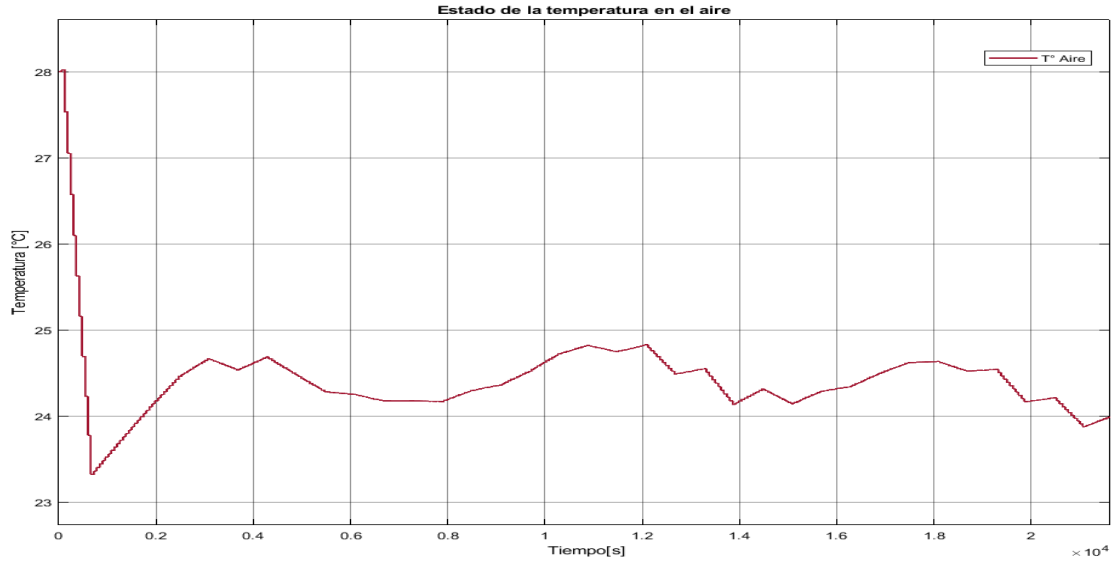
Al realizar una investigación de las posibles causas de este comportamiento, se llegó al concepto de '*Catastrofic forgetting*' [39][40]. Este concepto explica un comportamiento similar al mostrado en la figura 5.4, donde la causa es que luego de pasado un tiempo considerable el agente '*olvida*' cuales son las acciones no deseadas y, por lo tanto, vuelve a explorar dichas acciones, de donde puede ser que pase un tiempo y vuelva a las buenas acciones, o definitivamente no vuelva.

Es al ver esta problemática que se decide realizar cambios en la red neuronal, donde se identificó que dado que las salidas de la red del actor tienen una magnitud tan dispar (comparamos la válvula $[0, 0.9]$, el flujo de aire $[0, 5]$, la temperatura de enfriamiento $[-30, 0]$ y la temperatura de calefacción $[0, 8]$), se complica considerablemente para la red el aprendizaje

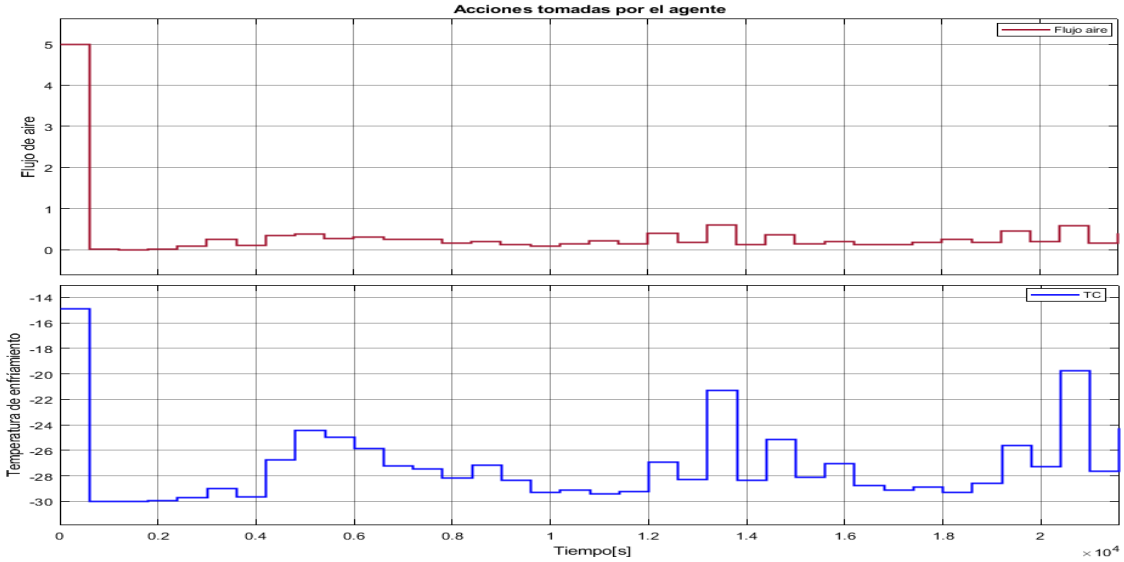
de las acciones que son correctas, dado que para aprender tienen que realizar una actualización de la magnitud de los pesos de las neuronas considerable. Es por ello que se toma la decisión de agregar una capa no lineal al final del actor, específicamente una capa *sigmoid* la cual entrega una salida en el rango $[0, 1]$, seguida de una capa para escalar dichas salidas a las magnitudes deseadas, ya que las salidas tiene que ir en el rango $[0, m]$ que es el rango para todas nuestras acciones, menos para el enfriamiento. Sin embargo, este se encuentra en el rango $[m, 0]$ lo cual es bastante similar y, por lo tanto, no afecta de forma negativa. Esta acción mejora considerablemente la estabilidad del entrenamiento, tal como podemos observar en la figura 5.5.b.

5.1.3.1. Gráficos de los estados y acciones

A continuación se muestran los resultados de las acciones tomadas por los distintos métodos, y los estados obtenidos por dichas acciones.

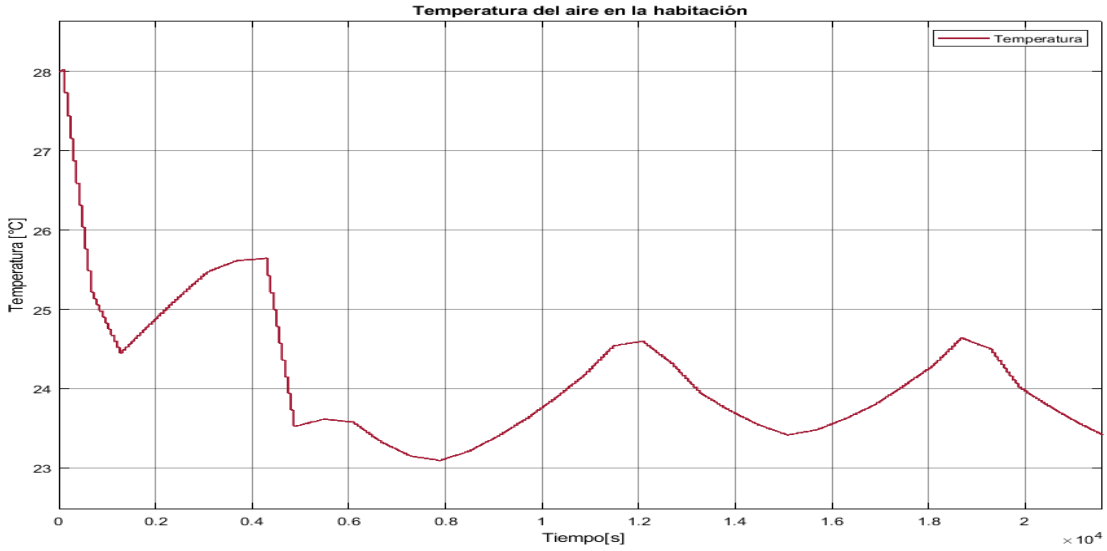


(a) Estados del sistema HVAC.

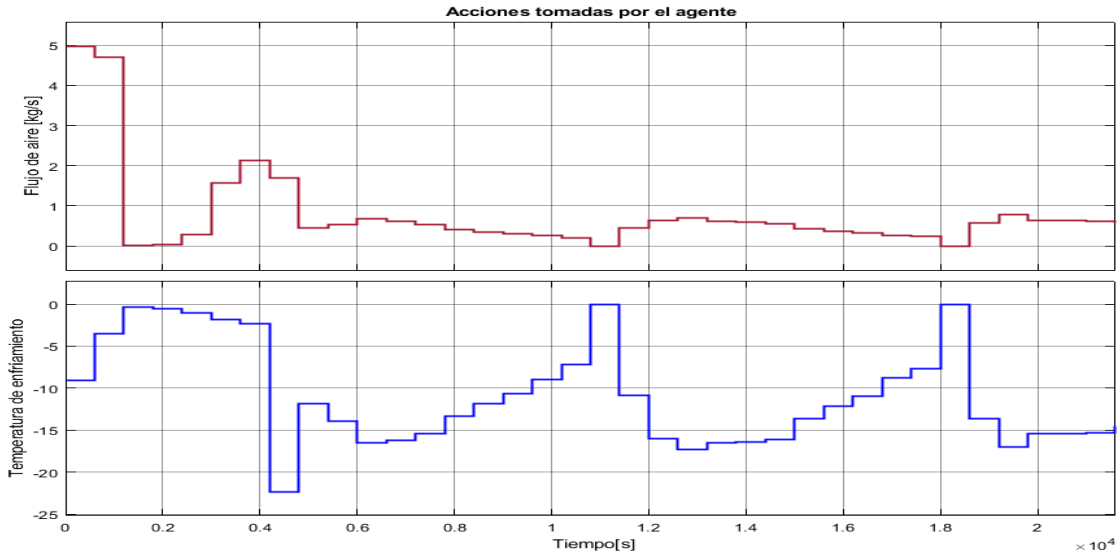


(b) Acciones tomadas por el agente.

Figura 5.6: Resultados para el caso con 2 variables de control utilizando DDPG.

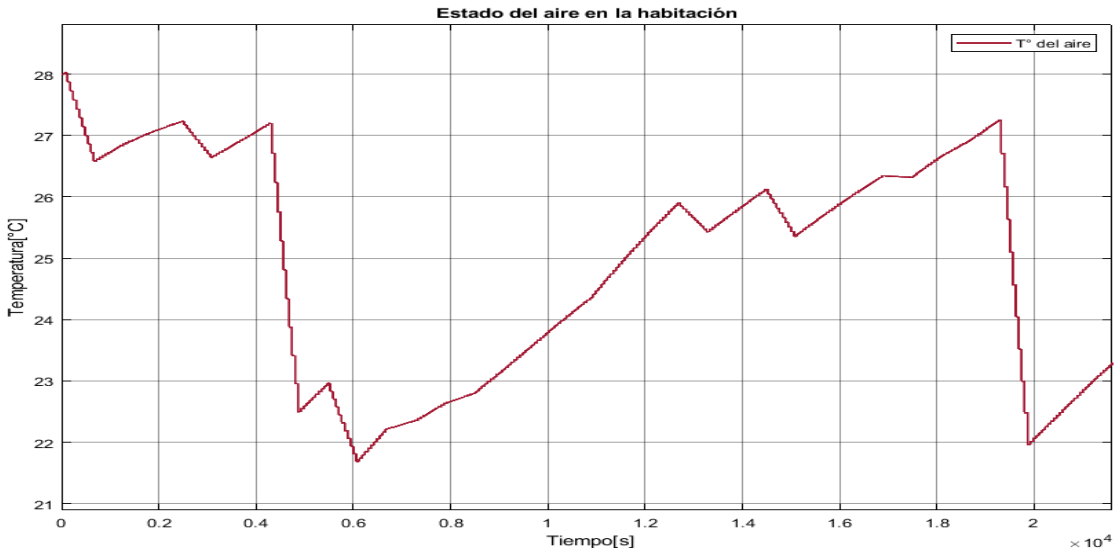


(a) Estados del sistema HVAC.

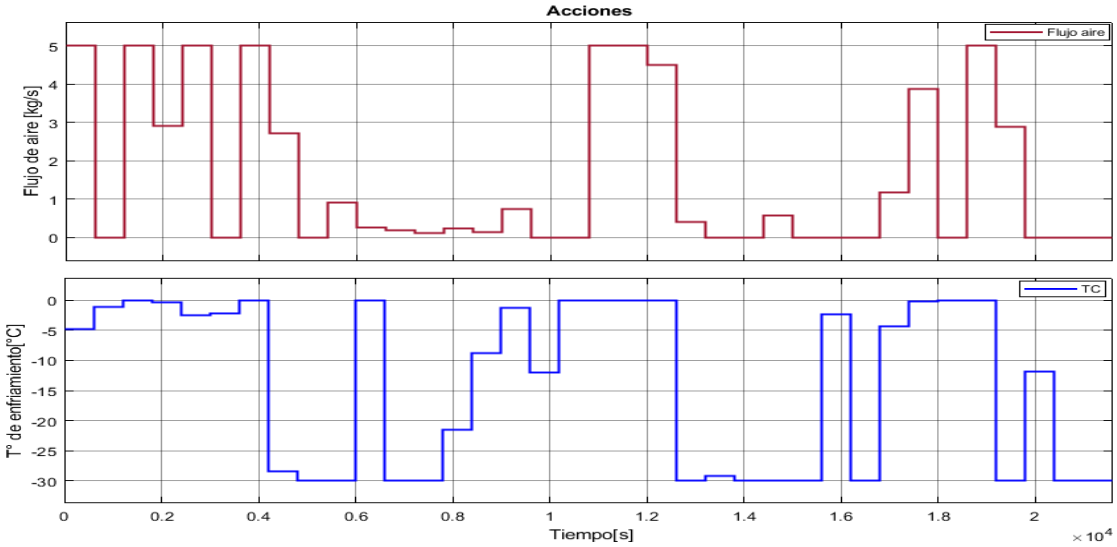


(b) Acciones tomadas por el agente.

Figura 5.7: Resultados para el caso con 2 variables de control utilizando TD3.



(a) Estados del sistema HVAC.



(b) Acciones tomadas por el agente.

Figura 5.8: Resultados para el caso con 2 variables de control utilizando PPO.

5.1.3.2. Análisis y comparación de resultados

En primer lugar se analizan los resultados obtenidos por DDPG. Se observa en la figura 5.6 que la temperatura del aire en la zona se mantiene en los rangos deseados y, es más, varía muy poco en el rango $24 - 25[°C]$, lo cual nos indica que dada la mayor temperatura exterior, el agente minimiza el uso energético al llevar al estado más cerca de la cota superior.

Por otro lado, podemos observar en la figura 5.6.b que las acciones tomadas son bastante razonables, en el sentido de que solo en el caso inicial hay un cambio brusco en el valor de la acción tomada y el cual se debe a que nos encontrábamos considerablemente lejos del estado deseado. Luego de llegar a la zona deseada, las variaciones son bastante pequeñas, donde se observa que el flujo de aire varía muy poco y con magnitudes inferiores a 1. Esto

se debe a que en la expresión de energía, el flujo de aire es un término cuadrático, por lo que queremos que sea lo más pequeño posible. Por otro lado, la variable de enfriamiento también tiene una variación considerablemente pequeña. Dadas estas pequeñas variaciones, tenemos que la solución podría llegar a ser implementada en un sistema real sin mayor riesgo.

A diferencia del caso donde se utilizó DDPG, podemos observar que el estado al entrenar con TD3 es ligeramente más inestable, debido a que oscila entre $23 - 25[°C]$ a diferencia de DDPG que oscila en un rango más pequeño, como se puede observar en la figura 5.6.a. En el caso de las acciones, podemos ver que, a simple vista, si bien el flujo de aire toma acciones con una magnitud relativamente pequeña, este aún podría mejorar, pues disminuyendo la temperatura de enfriamiento podría mejorarse aún más la cantidad de flujo de aire utilizada, como es el caso de la figura 5.6.b.

Comparando con los casos anteriores, los resultados obtenidos con el algoritmo de PPO (fig. 5.8), son considerablemente malos, dado que no se mantiene el estado deseado y, además, se realizan acciones completamente azarosas, donde los resultados son malos tanto para el estado como la optimización energética.

Luego, a modo de comparar de forma cuantitativa a los agentes entrenados y a su vez poner a prueba su robustez, se les prueba utilizando como temperatura ambiental una simulación de 6 horas de 4 días distintos, medidos por la Dirección meteorológica de Chile, donde los días escogidos corresponden a días en la quincena de los meses de Noviembre, Diciembre, Enero y Febrero del 2021, dado que estos son los meses más cálidos del año en Chile.

A continuación, en la tabla 5.1, se muestran los promedios de los resultados obtenidos, en términos de la recompensa obtenida por el agente en dichas 6 horas, la energía utilizada y cuánta penalización recibió por romper la restricción de temperatura.

Tabla 5.1: Comparación cuantitativa de los algoritmos al controlar las variables de flujo y enfriamiento.

Método	Recompensa	Energía [kW]	Penalización T^o
DDPG	-12.7325	58.8725	6.84475
TD3	-17.0075	65.4325	10.46.75
PPO	-61.192	96.1025	51.57

Como se observa en la tabla anterior 5.1, tal como fue analizado cualitativamente, tenemos que PPO es el algoritmo con peor desempeño, pues es el que tiene una menor recompensa y mayor gasto energético y penalización por estar fuera del estado deseado. Mientras que DDPG, tal como fue dicho anteriormente, es el que entrega la mejor solución en este caso, donde supera por un pequeño margen al algoritmo TD3 tanto en consumo de energía, como en cumplimiento de restricciones.

Luego el algoritmo que mejor se desempeña en este caso es DDPG, sin embargo, tanto TD3 y DDPG cumplen de buena manera con lo que se les solicita.

5.1.4. Cuarta instancia, controlando flujo de aire, enfriamiento y válvula de re circulación

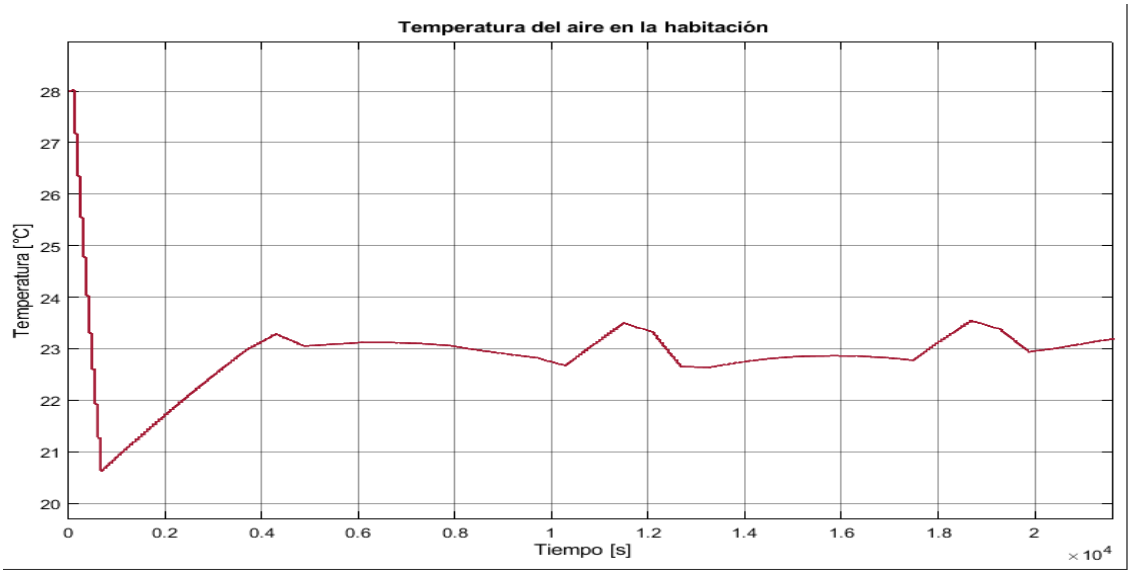
A continuación, tenemos el caso con 3 variables de control, donde aún estamos en pruebas con el enfriamiento solamente, por lo que la variable de calentamiento se puede considerar constante en cero. Por otro lado, se controlarán las variables de flujo de aire, temperatura de enfriamiento y posición de la válvula mientras se tiene una temperatura externa mayor al rango deseado.

Además para este caso, tenemos que se realiza el mismo experimento utilizando MPC[31] (trabajo no publicado), donde este será el único caso donde se comparará con MPC, pues estamos comparando con un trabajo anterior el cual estaba configurado solo para funcionar en este caso. Por otro lado, tenemos que tomar en cuenta tres consideraciones de la solución en MPC. Primero, la variable de enfriamiento tiene límites distintos, pues esta tiene un mínimo de $-10[°C]$ y no los $-30[°C]$ con los que se ha trabajado hasta ahora. Segundo lugar tenemos que los límites deseados son $[21, 24]°C$, es decir, un grado menos que lo instaurado en este trabajo y, finalmente, tenemos que el MPC está hecho para funcionar dentro de dichas restricciones en los horarios de 8 a $20[hrs]$; fuera de estos horarios trabaja a menor potencia.

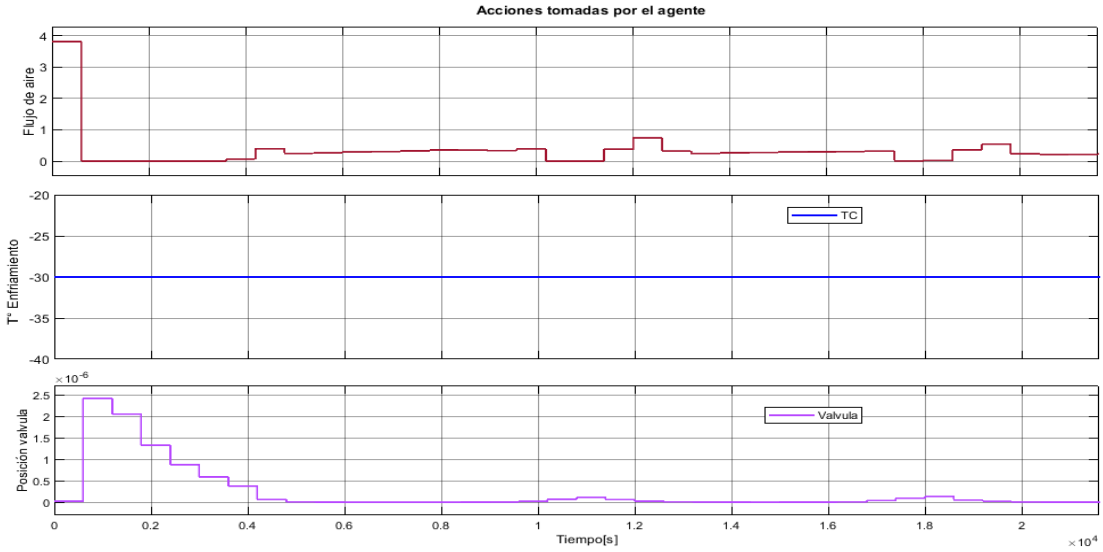
Es por esto último que se simula solo un día y se toman 6 horas desde las $14[hrs]$ a las $20[hrs]$, donde el día completo se puede ver en A.4.

5.1.4.1. Gráficos de los estados y acciones

Los resultados obtenidos con esta configuración se ven a continuación.

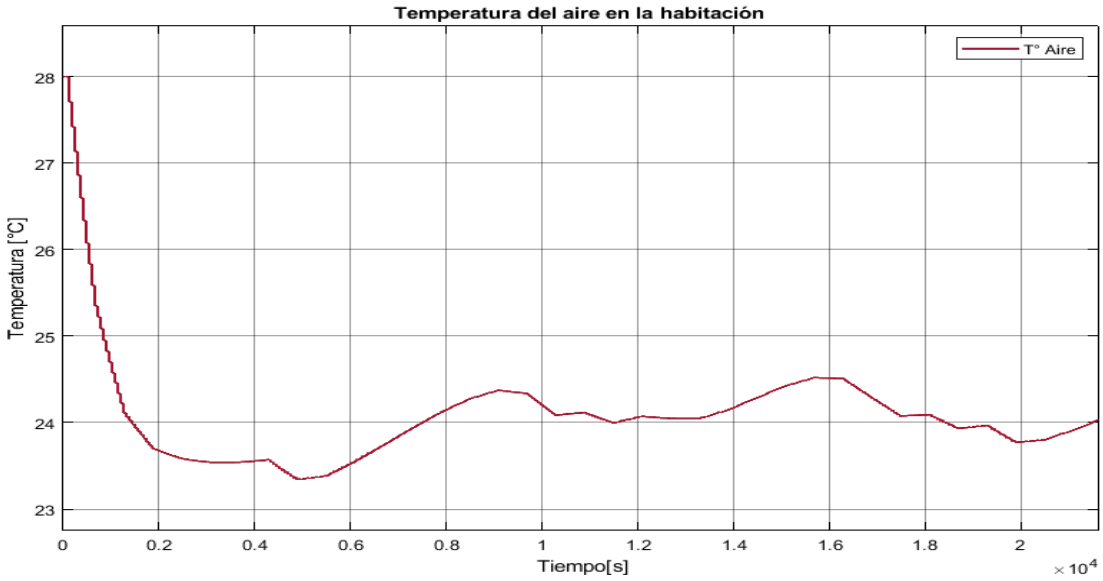


(a) Estados del sistema HVAC.

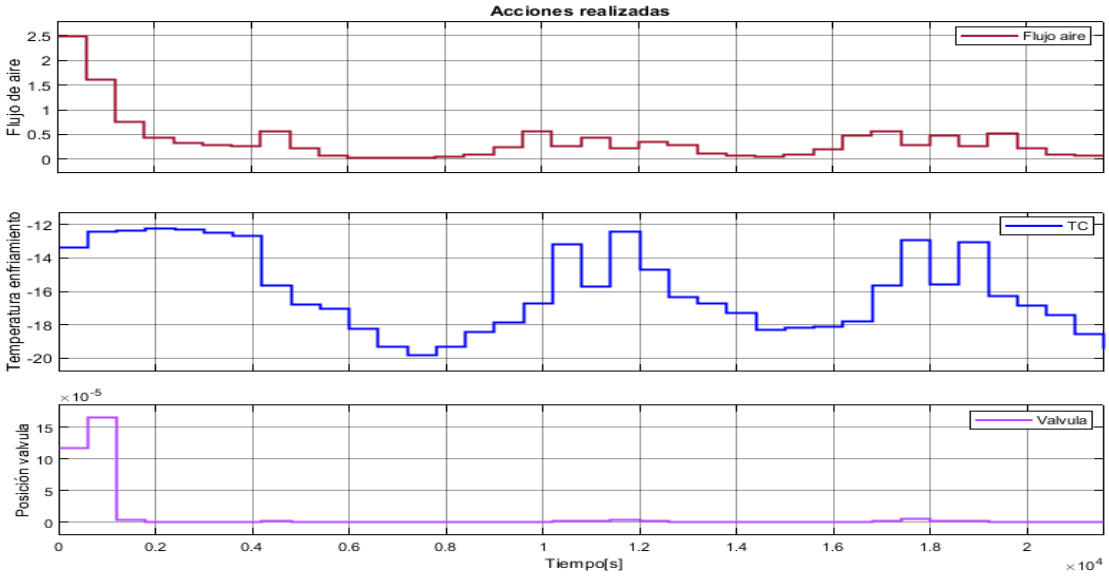


(b) Acciones tomadas por el agente.

Figura 5.9: Resultados para el caso con 3 variables de control utilizando DDPG.

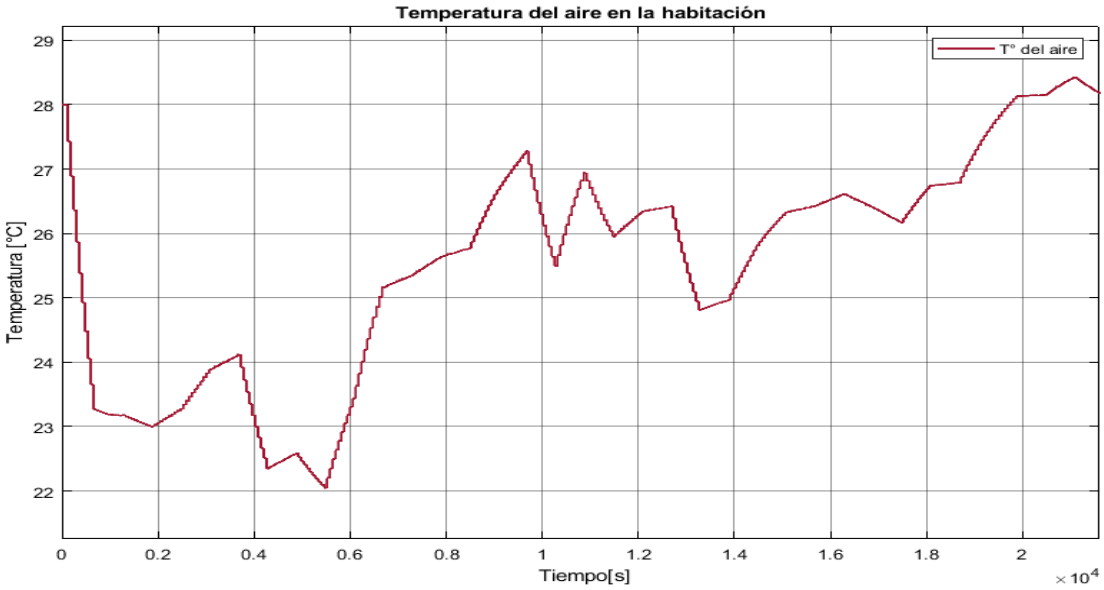


(a) Estados del sistema HVAC.

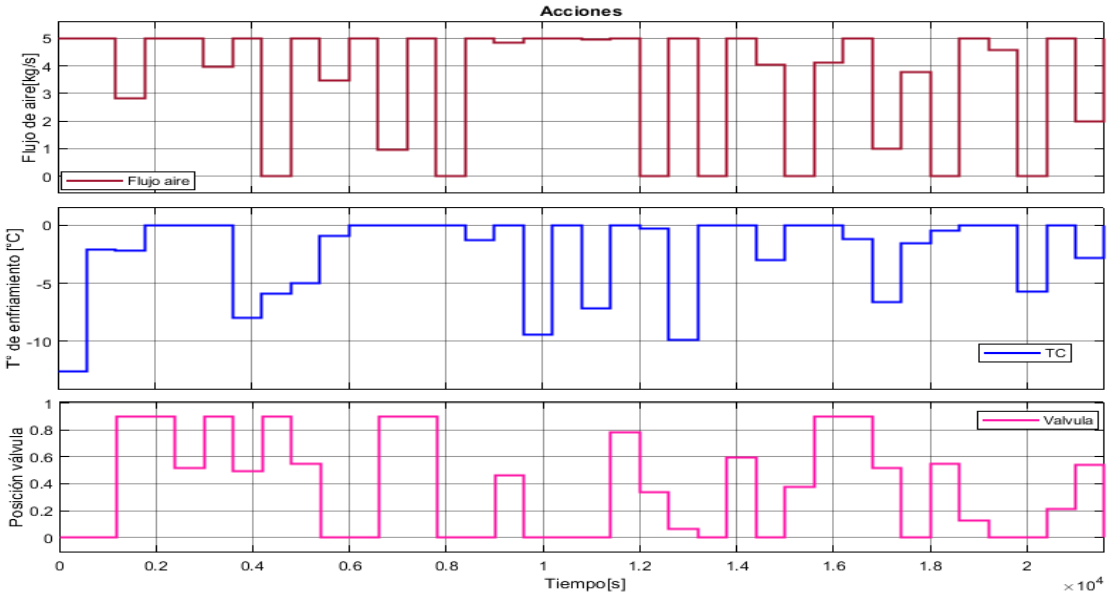


(b) Acciones tomadas por el agente.

Figura 5.10: Resultados para el caso con 3 variables de control utilizando TD3.

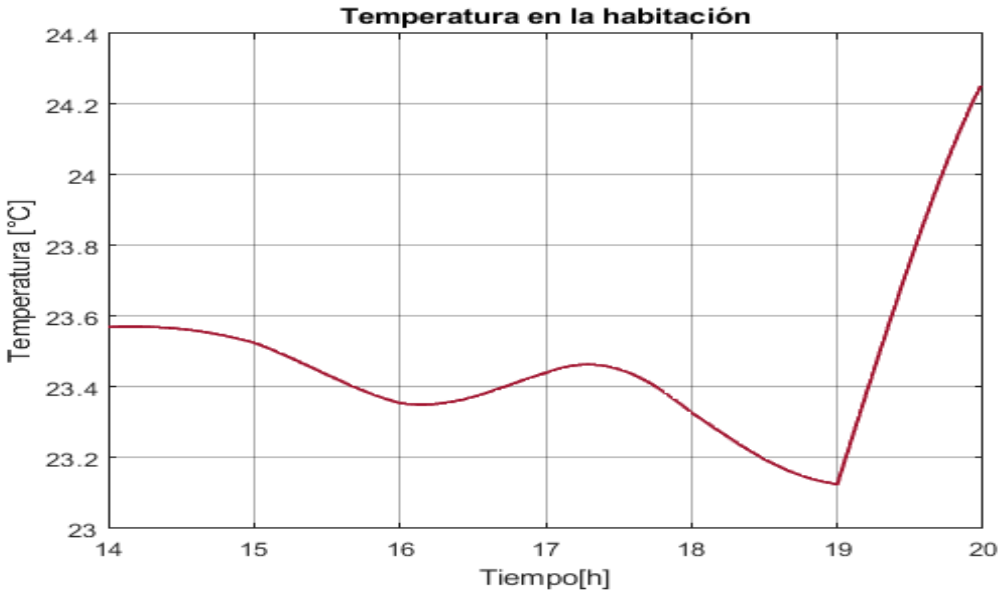


(a) Estados del sistema HVAC.

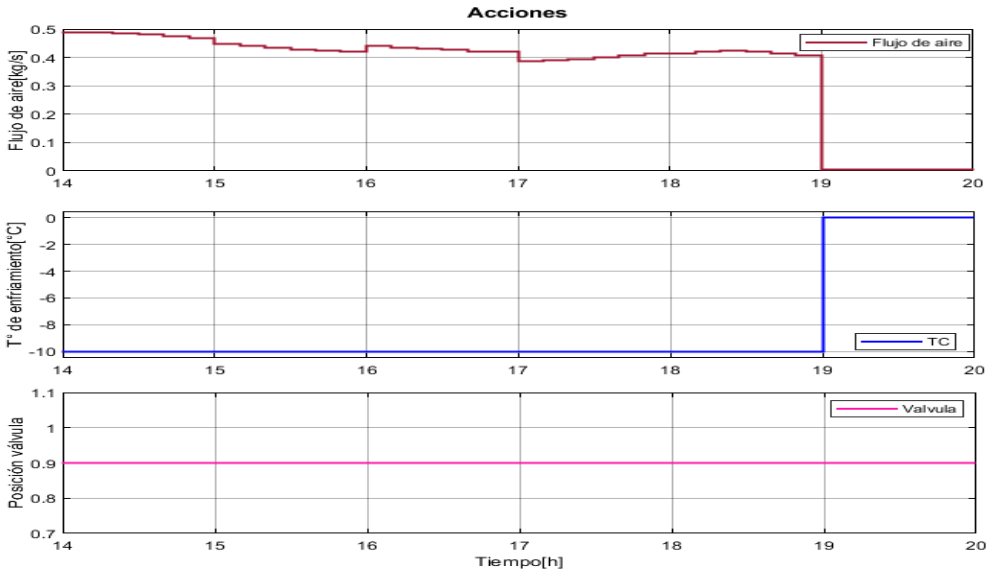


(b) Acciones tomadas por el agente.

Figura 5.11: Resultados para el caso con 3 variables de control utilizando PPO.



(a) Estados del sistema HVAC.



(b) Acciones tomadas por el agente.

Figura 5.12: Resultados para el caso con 3 variables de control utilizando MPC.

5.1.4.2. Análisis y comparación de resultados

Como podemos ver en los resultados anteriores, tenemos que es claro que el peor desempeño está dado por el algoritmo PPO, pues ni se mantiene en los estados ni toma acciones eficientes en el uso de energía y son acciones que realmente bordan lo imposible.

Por otro lado, tenemos que DDPG y TD3 actúan de forma similar, donde si observamos los estados en las figuras 5.9.a y 5.10.a, tenemos que TD3 mantiene un mejor control de los estados, pues DDPG en un principio enfría más de lo necesario, mientras que TD3 enfría solo lo necesario y se mantiene más cercano a a la cota superior que es lo esperado para ser

eficientes en el uso energético. Sin embargo, ambos se mantienen dentro de las restricciones de buena manera.

Por el lado de las acciones, tenemos que al observar las figuras 5.9.b y 5.10.b ambos toman buenas acciones en cuanto a enfriamiento y flujo de aire. Sin embargo, ambos dejan la válvula casi completamente abierta y que, como ya hemos dicho, en este caso es una mala acción a tomar. Además, podríamos decir que DDPG toma mejores acciones dado a que sus acciones varían muy poco y en casos nada, como en la temperatura de enfriamiento.

Finalmente, tenemos que el mejor comportamiento se obtiene con MPC. Sin embargo, tenemos que hacer una salvedad acá, pues para los casos de aprendizaje reforzado, se comienza estando fuera del estado deseando, mientras que en MPC comenzamos ya dentro del estado. A pesar de esto anterior, podemos observar que las acciones en la figura 5.12.b varían muy poco y, además, tenemos que el flujo de aire toma una magnitud menor a 0.5 permitiendo así una alta eficiencia energética. Por otro lado, si observamos los estados (figura 5.12.a), tenemos que el estado se encuentra la mayor parte del tiempo entre los 23 a 24$[°C]$ lo cual está bien si lo pensamos en un sentido de optimización de energía. Y tal como fue mencionado anteriormente el sistema funciona dentro de la restricción $[21, 24]$ solo dentro del horario $[8AM, 8PM]$, por lo que vemos que a las $7PM$ ya comienza a aumentar la temperatura para, finalmente, salirse de la restricción minutos antes de las $8PM$. Esto puede estar bueno o malo dependiendo de que tan estrictos queremos ser con las restricciones pero, de todas formas, el estado solo se sobrepasa con solo 0.2$[°C]$, aproximadamente.

A continuación, tenemos una tabla comparativa (5.2) al igual que en la configuración anterior, donde en este caso añadimos a MPC, donde cabe destacar que para MPC no se realizó un promedio, sino que es una simulación única donde se obtienen las métricas del fragmento de simulación que se ve en la figura 5.12.

Tabla 5.2: Comparación cuantitativa de los algoritmos.

Método	Recompensa	Energía [kW]	Pen. T^o
DDPG	-15.6975	36.0175	12.0978
TD3	-13.595	52.54	8.34
PPO	-142.52	126.675	129.8675
MPC	-3.524	33.1104	0.2114

Como podemos ver en la tabla anterior 5.2, esta avala la comparación de algoritmos anterior, donde se observa un muy mal desempeño por parte de PPO, mientras que TD3 y DDPG tienen un comportamiento bastante similar y MPC queda con los mejores resultados. Sin embargo, al comparar MPC con TD3 y DDPG, tenemos que tomar en consideración que estos dos últimos comienzan fuera de la restricción, lo que implica un gasto mayor de energía y una penalización por estado al comienzo de la simulación, partiendo con una desventaja numérica. Al tomar esto en consideración, podríamos considerar que los 3 algoritmos son bastante similares numéricamente, pues TD3 y DDPG comienzan con una penalización de $Pen_{T^\circ} = 6$ según la expresión 4.12 al comenzar en 28$[°C]$, luego la penalización por temperatura sería bastante similar entre los 3. Por otro lado, tenemos que sin considerar el gasto energético inicial, DDPG y MPC tienen un gasto bastante similar, con solo 3$[kW]$ más en

DDPG por sobre MPC, por lo que los 3 métodos son bastante similares en general, al menos en términos numéricos.

Dicho esto, a pesar de que MPC sea ligeramente mejor numéricamente, tenemos que considerar el costo que tiene llegar a obtener las soluciones, donde llegar a este tipo de solución tiene todo un complejo trasfondo matemático relacionado al modelo específico del sistema, el cual lo hace un método bastante especializado, lento de formular y muy complejo de generalizar para más sistemas. Por parte de aprendizaje reforzado, no fue necesario hacer cálculo alguno con respecto a cómo es el modelo del sistema, sino que el único esfuerzo matemático fue la definición de la función de recompensa, la cual incluso se puede mejorar aún más. Sin embargo, este método tiene un gasto computacional mayor pues para entrenar la red de manera óptima simula el sistema una gran cantidad de veces.

5.1.5. Quinta instancia, controlando 4 variables con restricciones de temperatura y CO2

Luego de los buenos resultados obtenidos anteriormente, se decide dar un paso más grande y realizar la implementación de la cuarta variable a controlar, que es la calefacción de las zonas, en conjunto con la segunda restricción, la cual es la concentración de CO_2.

Por otro lado, dado que la idea es ir avanzando en la dificultad del problema, significa que en este caso estaremos enfrentando el problema con 4 variables de control y dos restricciones de estados, además de las limitaciones físicas de las acciones de control; esta cuarta variable es la variable de calefacción del aire. Y no volveremos atrás a utilizar una menor cantidad de variables de control. Por otro lado, para garantizar que sea necesario el uso de estas 4 variables de control, es que se simulan 2 días completos con temperaturas bastante extremas y variables, pudiendo llegar a temperaturas bajo cero en horas de la madrugada y en horas de la tarde sobre los 30º, abarcando así un amplio espectro de decisiones a tomar.

Las incertezas, es decir, temperatura del ambiente (externo), generación de calor por uso, concentración de CO_2 del ambiente y CO_2 generado por uso, se pueden observar en el siguiente gráfico de la figura 5.13.

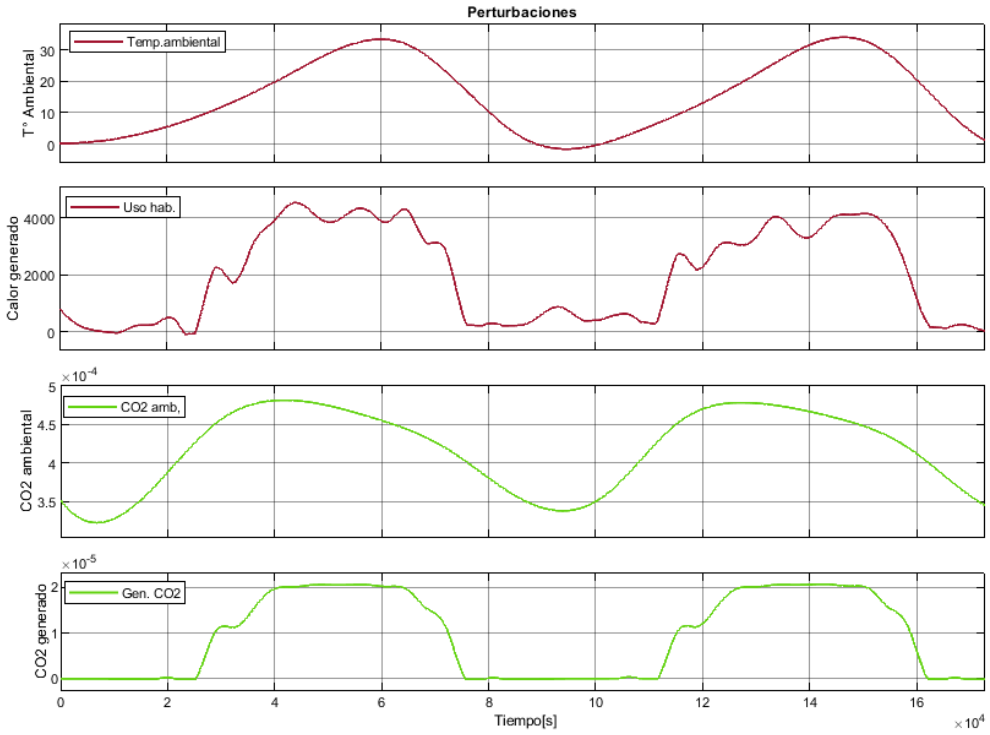
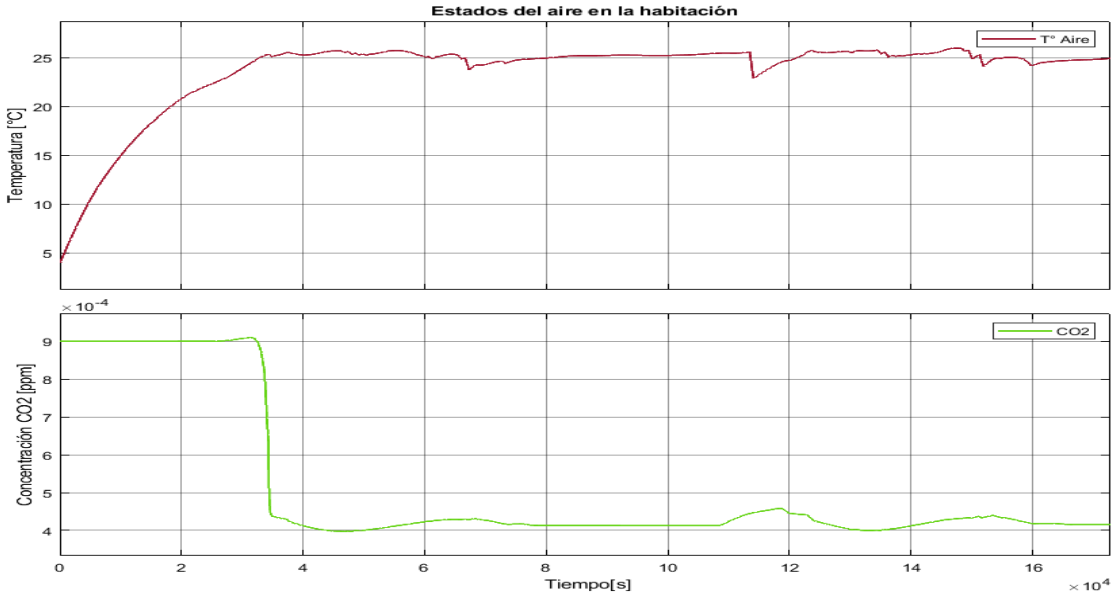


Figura 5.13: Gráfico mostrando las perturbaciones a lo largo de 2 días.

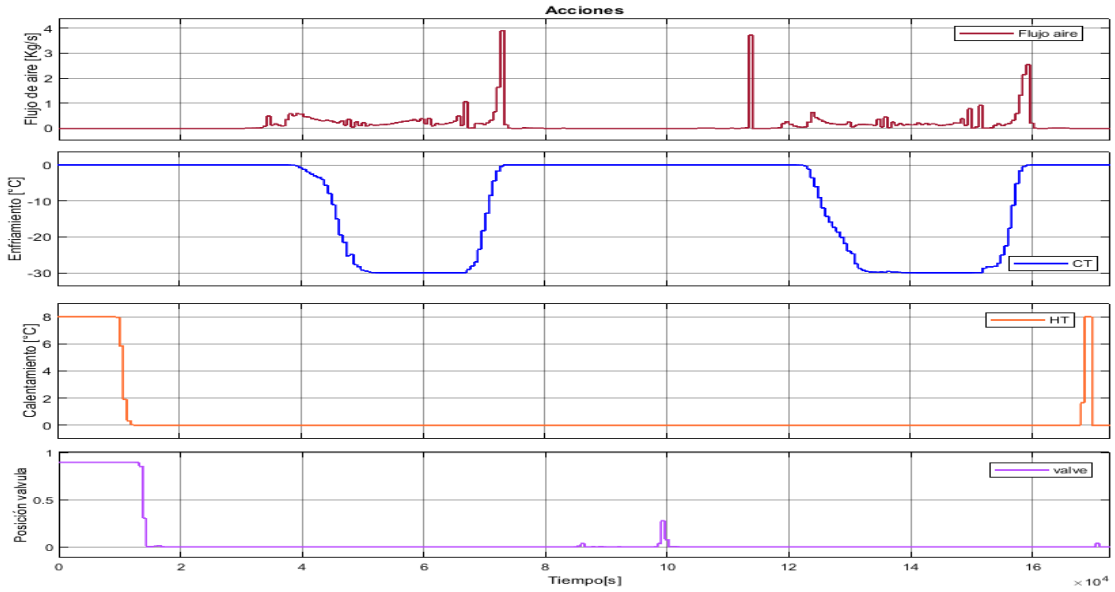
Finalmente, como estamos añadiendo una nueva restricción, es necesario modificar la función de recompensa para que de esta forma el agente pueda reconocer dicha restricción. Esto se ve expresado en la ecuación 4.14.

5.1.5.1. Gráficos de los estados y acciones

A continuación tenemos los resultados obtenidos por los distintos algoritmos bajo dichas condiciones externas.

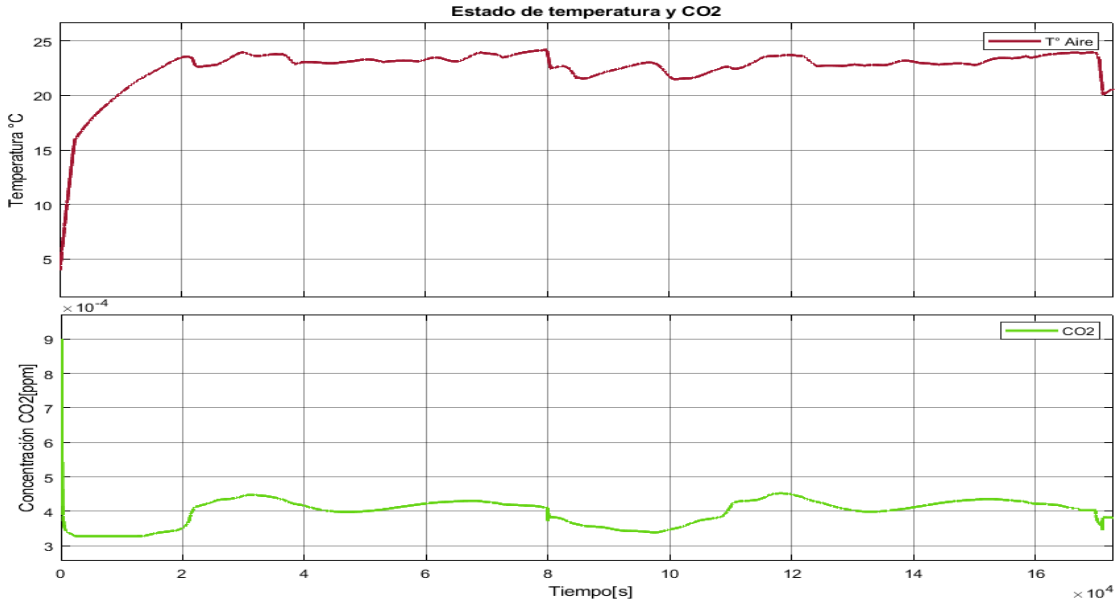


(a) Gráfico mostrando los estados a los que llega el agente con un tiempo de simulación de 2 días.

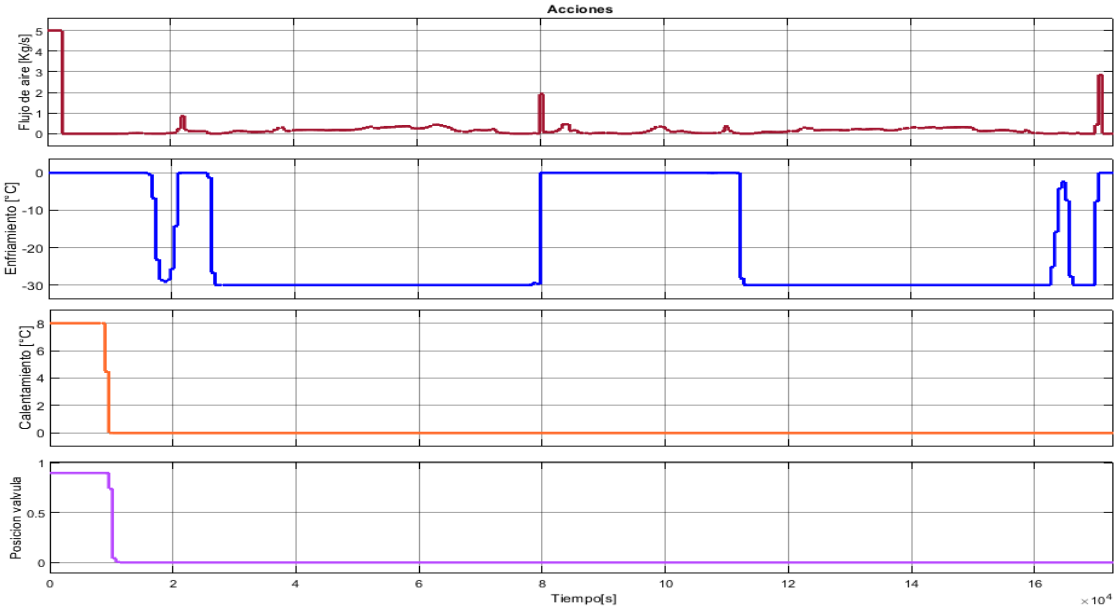


(b) Gráfico mostrando las acciones tomadas por el agente con un tiempo de simulación de 2 días.

Figura 5.14: Resultados utilizando DDPG para el caso con 4 variables de control, restricciones de temperatura y CO2.

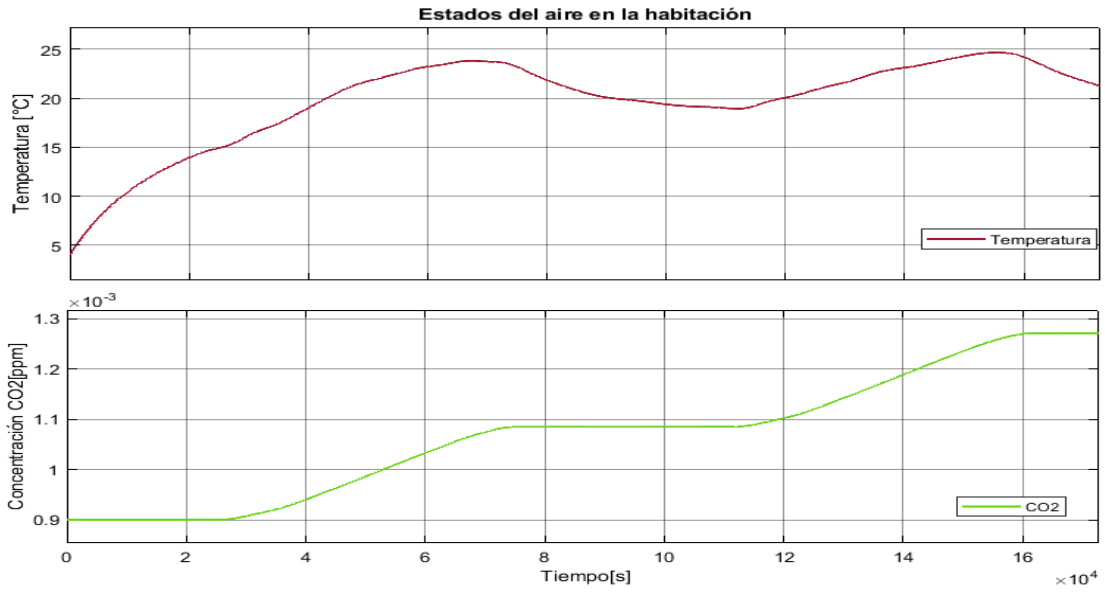


(a) Gráfico mostrando los estados a los que llega el agente con un tiempo de simulación de 2 días.

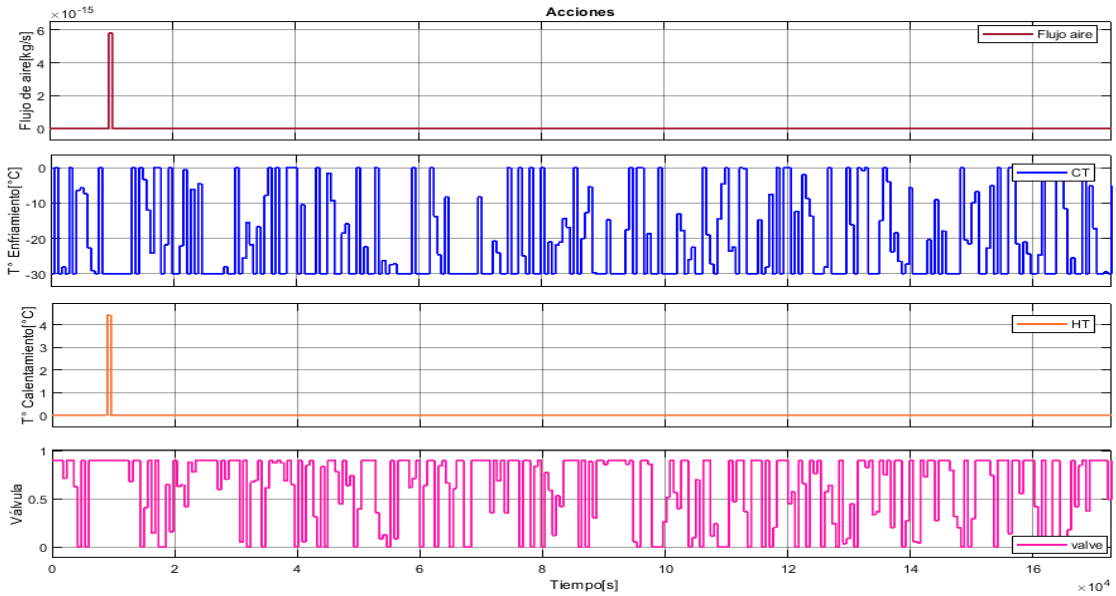


(b) Gráfico mostrando las acciones tomadas por el agente con un tiempo de simulación de 2 días.

Figura 5.15: Resultados utilizando TD3 para el caso con 4 variables de control, restricciones de temperatura y concentración de CO2.



(a) Gráfico mostrando los estados a los que llega el agente con un tiempo de simulación de 2 días.



(b) Gráfico mostrando las acciones tomadas por el agente con un tiempo de simulación de 2 días.

Figura 5.16: Resultados utilizando PPO para el caso con 4 variables de control, restricciones de temperatura y concentración de CO2.

5.1.5.2. Análisis y comparación de resultados

Como podemos observar, los resultados utilizando TD3 son bastante prometedores, no solo dado que llevan el sistema al estado deseado manteniéndose ahí, sino que, además, lo hace siendo eficiente enérgicamente y con soluciones que a simple vista son posibles de realizar.

Se observa en la figura 5.15.a que el estado llega a los rangos deseados, con una variación bastante razonable dadas las perturbaciones que afectan al sistema y que se mantiene dentro del valor deseado por largos períodos de tiempo para el caso de la temperatura. mientras que se mantiene permanentemente en el estado deseado en el caso de concentración de CO_2. Esto último es principalmente dado a que la válvula se mantiene abierta durante la simulación,

permitiendo que la concentración de CO_2 se mantenga a niveles similares a la concentración ambiental.

En el caso de las acciones, se observa en la figura 5.15.b que el flujo de aire toma acciones muy pequeñas y con poca variación. Salvo tres picos específicos, estás acciones de baja magnitud son las esperadas y deseadas dado que queremos optimizar energía y, como vimos anteriormente, el flujo de aire tiene una componente cuadrática en el gasto energético, por lo que queremos que su magnitud sea lo más baja posible. Por otro lado, estos 3 picos ya mencionados, se dan en los momentos donde la temperatura exterior comienza a disminuir. Esto dado el modelo de temperatura ambiental utilizada, donde estos picos cumplen la función de intentar calentar el sistema, pues lleva la acción de enfriar a cero, sin embargo, no aplica una acción de calentar por lo que no logra su cometido.

Luego, tenemos que para DDPG de todas formas se obtienen buenos estados, en el sentido que una vez que se llega al estado deseado se mantiene ahí. Sin embargo, podemos notar en la figura 5.14.a que se demora aproximadamente 20000 segundos, es decir, aproximadamente 5 horas y media, en llegar a los 20$[°C]$, donde el calentamiento obtenido está únicamente dado por la temperatura de los sólidos en la habitación, a los cuales se les dio el valor inicial de 18$[°C]$ y luego viene dada por las perturbaciones 5.13. Es decir, la utilización de la habitación y la temperatura exterior.

En el caso de las acciones notamos en la figura 5.14.b que se realizan acciones de calentar sin generar un flujo de aire con lo cual no genera una acción realmente. Esta podría ser la causa de lo lento que lleva el sistema al estado deseado. Esto se podría evitar modificando la función de recompensa, penalizando la acción de calentar sin un flujo de aire. Además, se notan 3 picos exagerados en el flujo de aire, los cuales son repentinos y tienen una corta duración, lo cual a largo plazo podría provocar fallas en el ventilador. Por lo tanto tenemos que las acciones son buenas debido que llevan a los estados deseados pero también hay que tomar en cuenta las dos consideraciones ya mencionadas que son puntos a mejorar en el entrenamiento.

Finalmente, tenemos PPO, algoritmo que tiene el peor desempeño de los tres, pues es el único en romper la restricción de concentración de CO_2 y la temperatura de la habitación. Esto dado que el algoritmo termina realizando muy poco control, debido al nulo flujo de aire que aplica. Por otro lado, tenemos que las acciones realizadas son imposibles, dado que estas oscilan de manera exagerada, y el desgaste que significa realizar dichas acciones muy probablemente constarían el funcionamiento de los actuadores.

Dado que las comparaciones anteriores fueron solo de una forma cualitativa, al igual que en los experimentos anteriores se pone a prueba a los agentes entrenados con ello, vamos a compararlos cuantitativamente. Para esto, se toman 2 días de cada mes del año y se toman los datos de su temperatura y humedad ambiental real en Santiago, medidas por la *Dirección Meteorológica de Chile* [41].

Luego, se simulan los agentes entrenados con esas 12 configuraciones y se obtiene el promedio de recompensa total obtenida, energía utilizada y penalización por romper las restricciones. Sin embargo, se toma la decisión de dividir los resultados en 3 tablas, dado que hay diferencias considerables al usar temperaturas ambientales cálidas (6 meses más cálidos) y temperaturas externas frías (6 meses más fríos), por lo que se hace una tabla para am-

bas temperaturas y, además, una tercera tabla con el promedio total obtenido, como se ve expresado en las tablas 5.3, 5.4 y 5.5.

Tabla 5.3: Comparación cuantitativa de los algoritmos en climas cálidos en el caso con restricciones de CO2 y temperatura.

Método	Recompensa	Energía [kW]	Penalización T°	Penalización CO_2
DDPG	-151.545	236.467	127.9	0
TD3	-111.928	111.938	100.72	0
PPO	-463.317	$9.39 \cdot 10^{-19}$	409.85	53.46

Tabla 5.4: Comparación cuantitativa de los algoritmos en climas fríos en el caso con restricciones de CO2 y temperatura.

Método	Recompensa	Energía [kW]	Penalización T°	Penalización CO_2
DDPG	-2400.167	120.0983	2388.167	0
TD3	-2468	41.8367	2463.833	0
PPO	-2178	$6.427 \cdot 10^{-15}$	2124.667	53.46

Tabla 5.5: Comparación cuantitativa de los algoritmos en el caso con restricciones de CO2 y temperatura.

Método	Recompensa	Energía [kW]	Penalización T°	Penalización CO_2
DDPG	-1275.8558	178.2825	1258.033	0
TD3	-1290.0475	76.8875	1282.27667	0
PPO	-1320.658	$3.21 \cdot 10^{-15}$	1267.258	53.46

Como podemos diferenciar de las tablas 5.3 y 5.4, tenemos que los tres algoritmos se desempeñan de peor manera en invierno. Esto dado a lo mencionado anteriormente de que se realiza la acción de calentar sin utilizar un flujo de aire, por lo que se está en el estado equivocado una mayor cantidad de tiempo.

Luego, al igual que en casos anteriores, tenemos que PPO es el que se desempeña de peor manera. Sin embargo, en este caso no existió una diferencia tan notoria, pero, tenemos según la figura 5.16.b que realmente PPO no realiza acción alguna. A pesar de esto, logra conseguir una recompensa comparable con los otros algoritmos. Sin embargo, rompe la restricción de CO_2, la cual es un restricción que no se le agregó un gran peso, pues tanto DDPG y TD3 lo controlaron de buena manera y, al incorporar PPO, no se realizó ningún cambio con respecto a esto, pues su control es bastante trivial y basta con generar un pequeño flujo de aire constante para mantener la restricción. A pesar de esto, tal vez es uno de los estados que más debería tener peso, pues las altas concentraciones de CO_2 son perjudiciales para la salud, o al menos se le podría agregar un mayor peso al entrenar con PPO. A pesar de lo anterior, tenemos que PPO se desempeña mejor que DDPG y TD3 en los climas fríos, tendencia que se espera cambie al incluir la penalización por la no existencia de flujo de aire.

Por otro lado, en promedio DDPG y TD3 son los mejores algoritmos, los cuales están bastante cerca entre si en las métricas, donde TD3 no solo es mejor en temas de estado en verano, si no que gasta aproximadamente la mitad de energía que DDPG. Mientras que al ver el invierno, DDPG rinde mejor que TD3 en términos de estado, sin embargo, utiliza el triple de energía para lograr su cometido. Finalmente, tenemos que según la tabla 5.5 DDPG se desempeña de mejor forma en promedio en los estados en comparación con TD3. No obstante, DDPG utiliza aproximadamente 2.3 veces más energía que TD3 para lograr este cometido, lo que dada la pequeña diferencia en término de estados, nos permite concluir que TD3 es el algoritmo que mejor se desempeña para esta configuración en particular.

5.1.6. Sexta instancia, restricciones de temperatura, CO2 y humedad

A continuación se agrega la restricción correspondiente a la humedad relativa al interior de la habitación. Al igual que cuando se agregó la restricción de la concentración de CO_2, para que el agente pueda reconocer la restricción es necesario modificar la función de recompensa. Para este caso en particular, se tiene que los actuadores, es decir, las cuatro variables de control, no son suficientes para llevar un óptimo control, dado que vamos a estar sujetos a realizar un control sujeto a la humedad ambiental, donde si esta rompe las restricciones impuestas, no tendremos forma de controlar la humedad. Es por esta razón que utilizaremos una baja penalización por romper la restricción, pues le daremos mayor importancia a aquellas dos restricciones que podemos controlar todo el tiempo.

La ecuación 4.15 indica la nueva función de recompensa, donde la penalización **máxima** por romper la restricción de humedad será de tan solo 0.35. Permitiendo así poner más énfasis en el resto de los estados deseados y en la eficiencia energética.

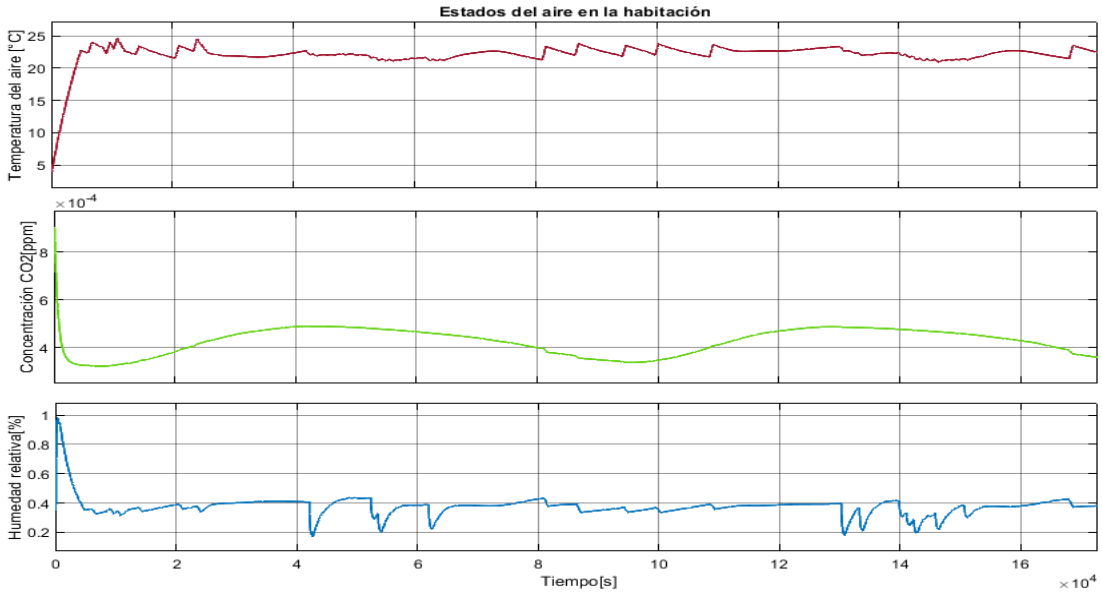
Por otro lado, podemos observar en la figura 5.14.a que el sistema calienta muy lentamente dado que no genera un flujo de aire en conjunto con la acción de calentar. Es por ello que se toma la decisión de agregar una penalización a este tipo de acciones.

Luego, dado que en el caso anterior los agentes realizaban la acción de calentar sin ningún flujo de aire (por lo que la acción no tenía ningún efecto en el sistema), tenemos que se entregará una penalización de -10, si se toma la acción de calentar cuando no hay flujo de aire, y se entregará una penalización de -1 para el caso del enfriamiento. Se penaliza menos el enfriamiento pues este se encontraba ya funcionando de manera óptima.

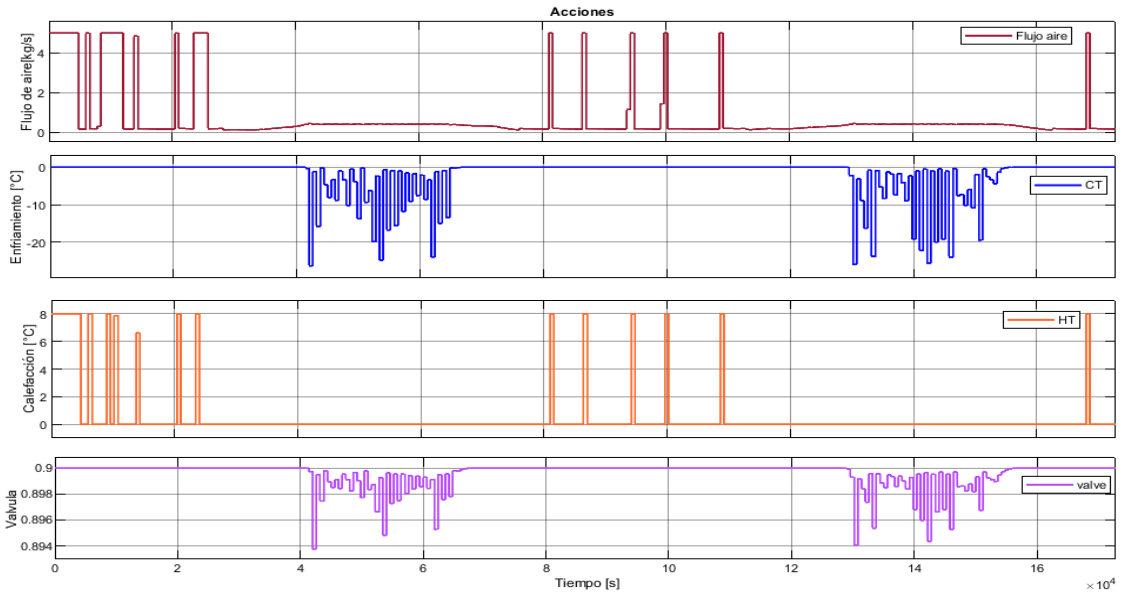
Por otro lado, las perturbaciones utilizadas serán las mismas del caso anterior añadiendo la humedad ambiental y generada, donde además entrenaremos utilizando dos días con temperaturas y humedad medidas por la Dirección General de Aeronáutica de Chile, los cuales corresponden a días de verano e invierno para que el sistema pueda comportarse de mejor manera a lo largo del año.

5.1.6.1. Gráficos de los estados y acciones

A continuación se muestran los distintos resultados obtenidos por cada algoritmo.

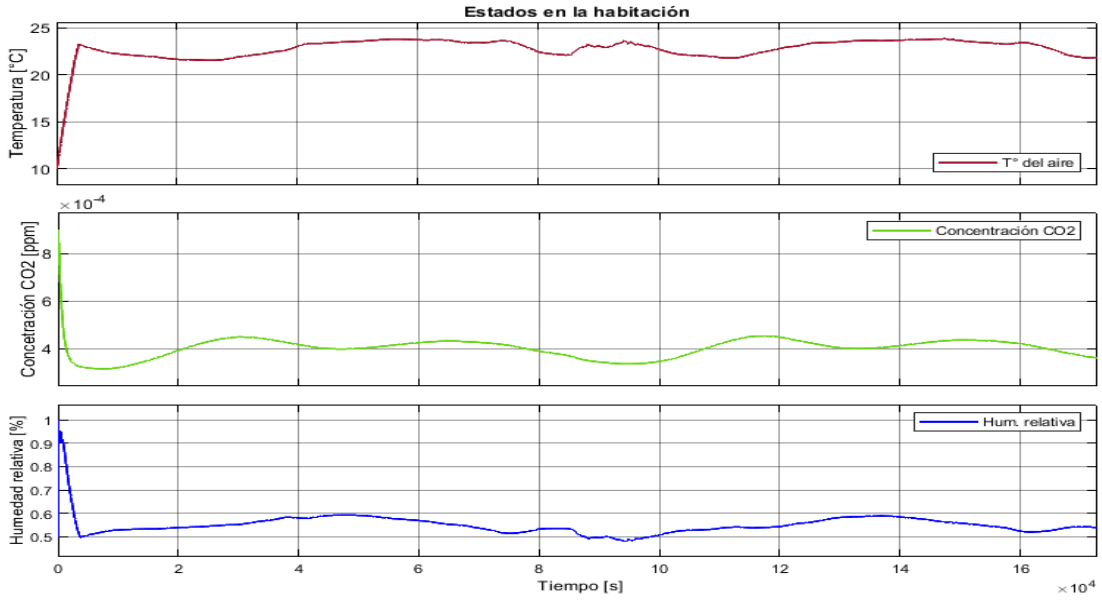


(a) Gráfico mostrando los estados a los que llega el agente con un tiempo de simulación de 2 días.

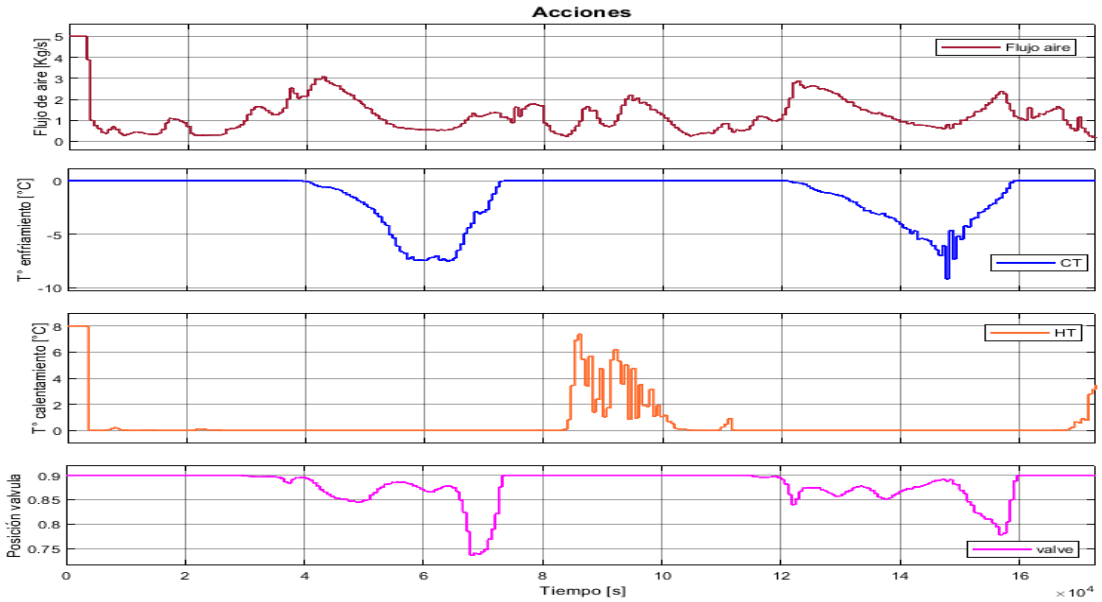


(b) Gráfico mostrando las acciones tomadas por el agente con un tiempo de simulación de 2 días.

Figura 5.17: Resultados utilizando DDPG para el caso con 4 variables de control y restricción de temperatura, CO2 y humedad.

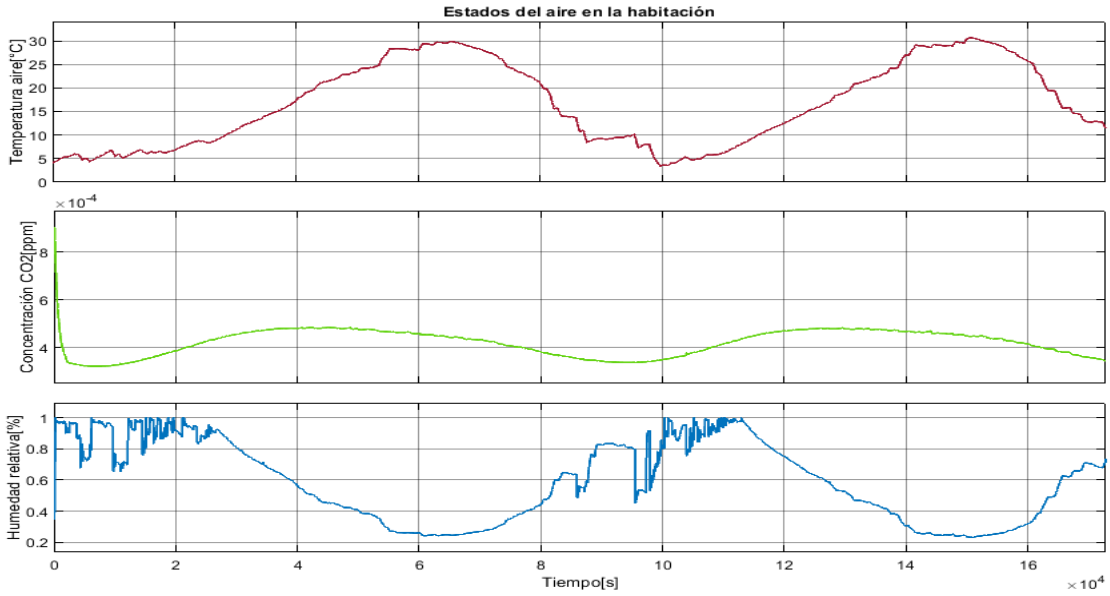


(a) Gráfico mostrando los estados a los que llega el agente con un tiempo de simulación de 2 días.

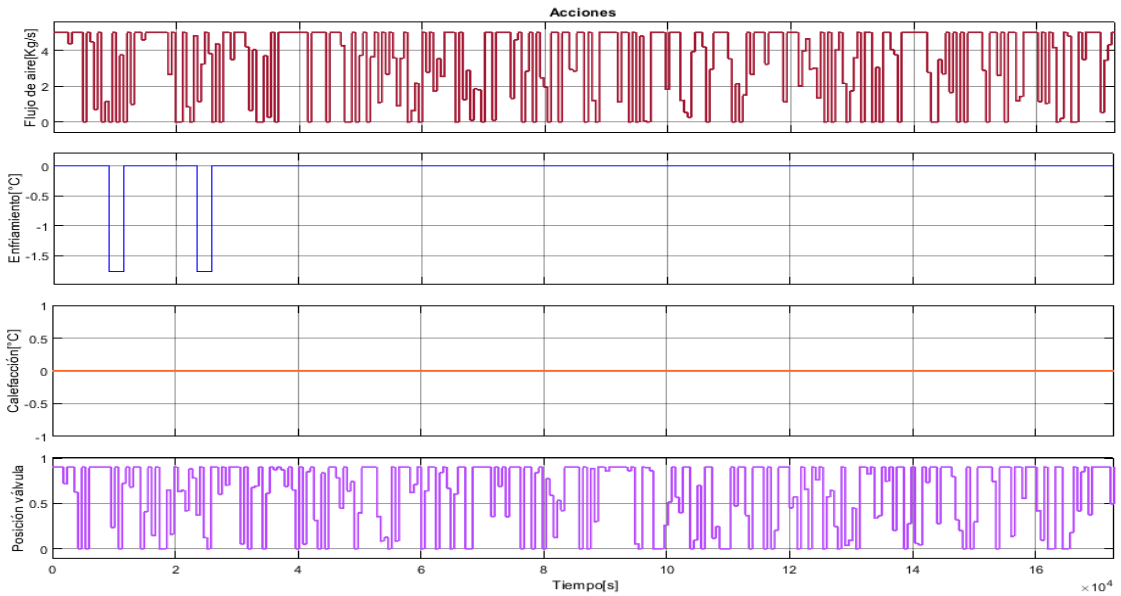


(b) Gráfico mostrando las acciones tomadas por el agente con un tiempo de simulación de 2 días.

Figura 5.18: Resultados utilizando TD3 para el caso con 4 variables de control y restricción de temperatura, CO2 y humedad.



(a) Gráfico mostrando los estados a los que llega el agente con un tiempo de simulación de 2 días.



(b) Gráfico mostrando las acciones tomadas por el agente con un tiempo de simulación de 2 días.

Figura 5.19: Resultados utilizando PPO para el caso con 4 variables de control y restricción de temperatura, CO2 y humedad.

5.1.6.2. Análisis y comparación de resultados

Podemos notar en la figura 5.17 que DDPG se comporta realmente bien, con especial énfasis en la figura 5.17.a, donde el resultado obtenido es un estado donde se mantiene casi permanentemente en los rangos de restricción, inclusive para la restricción de humedad, que como ya fue mencionado anteriormente, es la más compleja de cumplir. Además, se logra apreciar que tanto la temperatura como la concentración de CO_2 se mantienen dentro de los rangos con muy pocas variaciones. Además, notamos que se llega rápidamente al estado deseado debido a que en efecto se toma la acción de calentar. Esto fue gracias a la modificación realizada en la función de recompensa.

Sin embargo, tenemos que al observar las acciones tomadas en 5.17.b nos damos cuenta que estas no tienen un comportamiento tan bueno como el de los estados, puesto que tanto el flujo de aire como la calefacción toman acciones repentinas pasando desde el mínimo al máximo posible y viceversa en repetidas ocasiones y por cortos períodos de tiempo. Lo que no sólo implica un alto gasto energético sino que también genera una gran carga en el actuador, mermando así su vida útil. Por otro lado, la válvula y el enfriamiento, en general, se comportan de buena manera pero sus acciones podrían ser un poco más suaves y más escaladas en vez de ser algo ruidoso y variante.

En el caso de TD3, se observa en la figura 5.18 que el controlador entrenado por el algoritmo realiza un muy buen trabajo. Esto dado que al analizar los estados del aire mostrados en la figura 5.18.a tenemos que las restricciones se cumplen casi en su totalidad, con la excepción de los momentos iniciales, donde inclusive la humedad se mantiene dentro de los rangos deseados y la temperatura oscila muy suavemente entre las dos cotas, dependiendo de la hora del día (clima exterior más cálido o frío).

Luego al observar las acciones en la figura 5.18.b, lo único que podríamos reprocharle al controlador, son las oscilaciones de la calefacción en la madrugada del segundo día. Sin embargo, al dejar esto de lado tenemos que las acciones realizadas son bastante buenas, pues el flujo de aire se mantiene dentro de magnitudes que son lo más bajo posible para garantizar los estados, permitiendo así un óptimo uso de la energía eléctrica y, además, se observan acciones que son escaladas, es decir, no se utilizan cambios repentinos en el actuador como fue el caso en DDPG. Por lo que se garantiza que la vida útil de los actuadores no se verá mermada por las decisiones del controlador.

Finalmente, tenemos el controlador entrenado mediante PPO en la figura 5.19, donde dado el rendimiento de este algoritmo a lo largo de este trabajo y en especial al de la sección anterior, no se tenían muchas expectativas de que funcionara de forma correcta. Tal como se esperaba, el algoritmo tuvo un rendimiento deficiente donde se observa en los estados (fig 5.19.a) que realmente no se controla nada pues se rompen las restricciones impuestas en casi todo momento a excepción de la concentración de CO_2 que se mantiene dentro del estado deseado. Además, al ver las acciones tomadas en 5.19.b se observan acciones poco eficientes e imposibles de implementar para cualquier actuador.

Al igual que la sección anterior, ahora compararemos numéricamente los algoritmos, donde los resultados de las simulaciones nuevamente fueron divididos en 3 tablas diferenciando entre climas fríos y cálidos. Además, se utilizaron los mismos días que fueron utilizados en la sección anterior, donde se le añadió la humedad correspondiente a cada día la cual fue calculada con los datos entregados por la Dirección General de Aeronáutica (humedad relativa, temperatura y presión atmosférica) y la ecuación de humedad de saturación 2.10.

Tabla 5.6: Comparación cuantitativa de los algoritmos en climas cálidos en el caso con restricciones de temperatura, CO2 y humedad.

Método	Recompensa	Energía [kW]	Pen. T^o	Pen. CO_2	Pen. hum.
DDPG	-119.31	788.55	33.154	0	7.301
TD3	-86.04	618.68	20.95	0	3.216
PPO	-2953.33	293.1	2911.83	0	11.3265

Tabla 5.7: Comparación cuantitativa de los algoritmos en climas fríos en el caso con restricciones de temperatura, CO2 y humedad.

Método	Recompensa	Energía [kW]	Pen. T^o	Pen. CO_2	Pen. hum.
DDPG	-296.4	991.5	183.63	0	13.625
TD3	-289.93	797.78	198.783	0	11.367
PPO	-14878.83	293.1	14840.5	0	7.418

Tabla 5.8: Comparación cuantitativa de los algoritmos en promedio en el caso con restricciones de temperatura, CO2 y humedad.

Método	Recompensa	Energía [kW]	Pen. T^o	Pen. CO_2	Pen. hum.
DDPG	-207.855	890.025	108.394	0	10.463
TD3	-187.99	708.23	109.87	0	7.291
PPO	-8915.08	293.1	8876.17	0	9.376

Observamos de las tablas 5.6, 5.7 y 5.8 que, tal como en el caso con solo dos restricciones, se obtiene un mejor rendimiento en los climas cálidos que en los climas fríos, esto tanto en términos de los estados como de la energía utilizada. Sin embargo, podemos notar una gran diferencia en los resultados de la tabla 5.4 y 5.7, donde en esta última tabla, se obtiene una recompensa mucho mayor aunque con un mayor gasto energético. Esto se debe a la modificación hecha en la función de recompensa para evitar acciones sin flujo de aire.

Luego, tenemos que en la tabla 5.6 existe un claro ganador que es TD3, pues supera a los otros dos algoritmos en casi todas las métricas, obtiene una mejor recompensa con un gasto energético más eficiente y recibe una menor o igual penalización en todos los estados. Luego de TD3, tenemos a DDPG que a pesar de no hacerlo tan bien como TD3, obtiene unos resultados considerablemente buenos, estando bien cercano a los resultados de TD3. Finalmente, PPO obtiene pésimos resultados en términos de estados pero, utiliza una cantidad de energía menor que los otros dos métodos. Esto realmente no es relevante para nosotros, pues de nada nos sirve una baja utilización de energía si los estados ni siquiera llegan a estar cerca de los deseado.

Al observar la tabla 5.7 tenemos que TD3 nuevamente es mejor que DDPG en cuánto a la energía utilizada y a la recompensa obtenida. Sin embargo, esta ultima tiene una diferencia tan pequeña de tan solo 6.47, lo que podría llevarnos a decir que están técnicamente empatados, pues DDPG maneja de mejor forma la temperatura y por lo tanto recibe menos

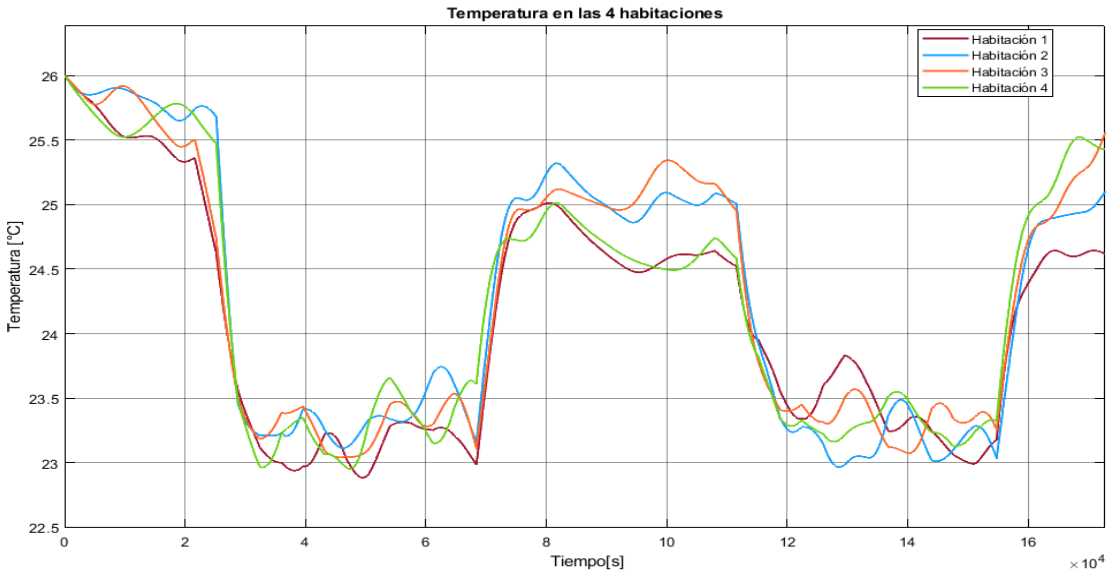
penalización por aquello, pero al costo de un mayor gasto energético, mientras que TD3 tiene una mayor penalización por temperatura, pero un gasto energético mucho menor, luego, tenemos que la diferencia en la recompensa es pequeña debido a que la penalización por estado tiene un peso considerablemente mayor a la utilización de energía. Dado lo anterior es que en el análisis de la tabla 5.7 no podemos tomar una decisión sobre cual se desempeña de mejor manera, ya que va a depender de que tan estrictos queremos ser con la utilización de energía y la violación de restricción de temperatura. En el caso de PPO, tenemos que nuevamente utiliza menos energía que los otros métodos, pero generando una penalización por temperatura enorme, mientras que obtiene menos penalización por humedad. Esto se debe a que PPO, en general, se encuentra en temperaturas más bajas que la cota inferior de la restricción, por lo que la humedad relativa es mayor que cuando se encuentra en o sobre dicha cota.

Finalmente, tenemos que al observar el promedio anual en la tabla 5.8 tenemos un claro ganador que es TD3, pues obtiene una mayor recompensa que los otros dos métodos a un gasto energético considerablemente menor, mientras que la penalización por temperatura es solo mayor en 1.476 que la penalización obtenida por DDPG, por lo que podríamos considerarlo un empate. Mientras que tiene una penalización por humedad aún menor que DDPG y PPO la cual si esta penalización tuviera más peso, sería una gran diferencia.

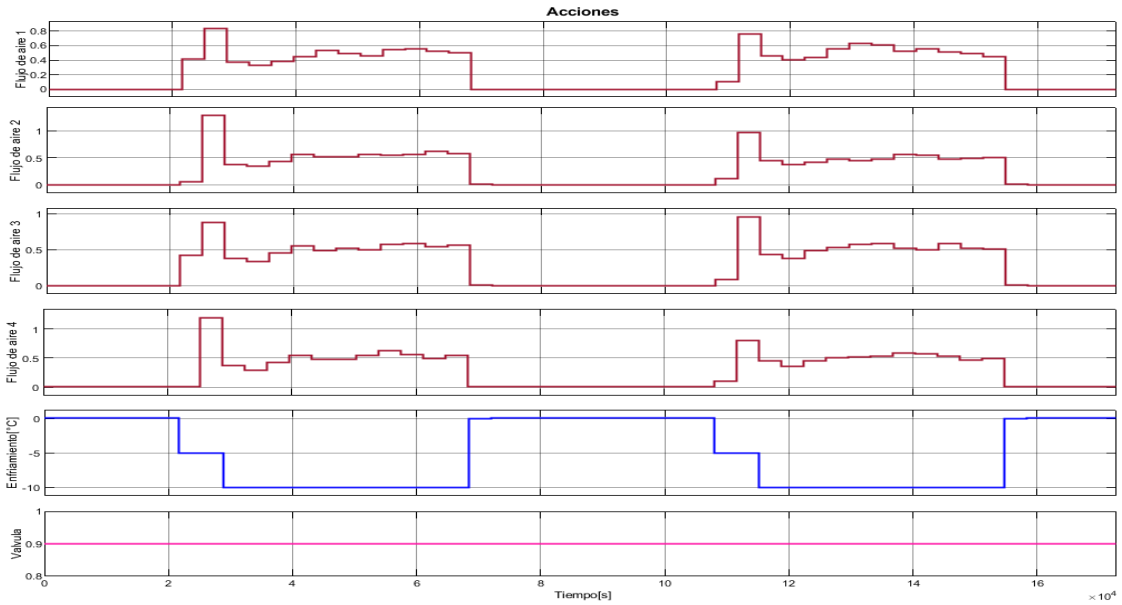
Es por estos dos últimos casos, donde tenemos que TD3 se desempeñó mejor que los otros algoritmos, que lo utilizaremos para expandir el caso de 4 variables y 3 restricciones a utilizar un mayor número de zonas, donde en específico utilizaremos 4 zonas, dado que el caso con 2 zonas es bastante similar al ya estudiado.

5.1.7. Séptima instancia, restricciones de temperatura en 4 zonas

5.1.7.1. Gráficos de los estados y acciones

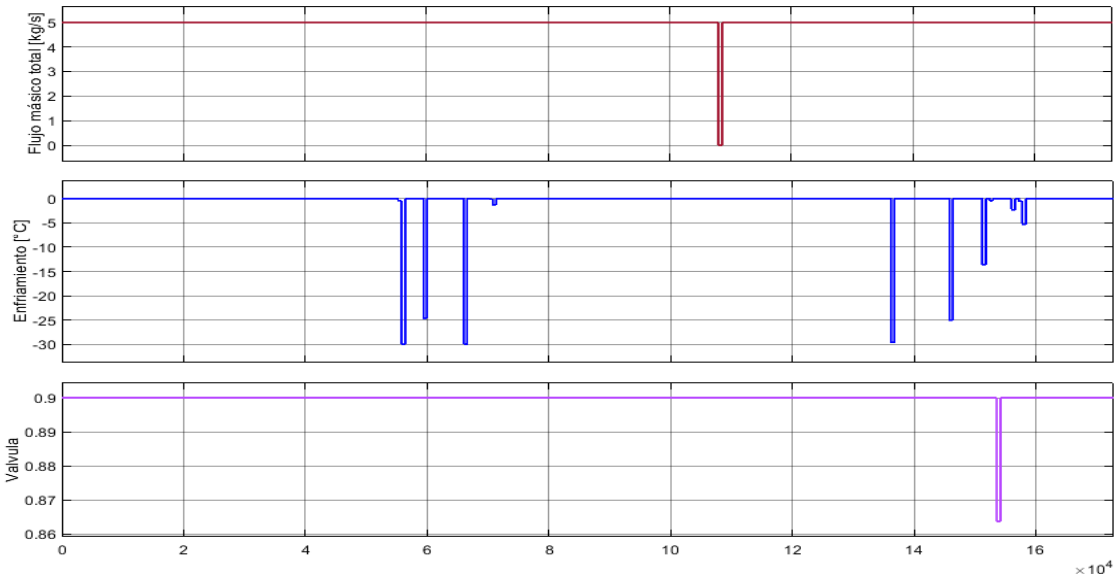


(a) Gráfico mostrando los estados a los que llega el agente con un tiempo de simulación de 2 días.

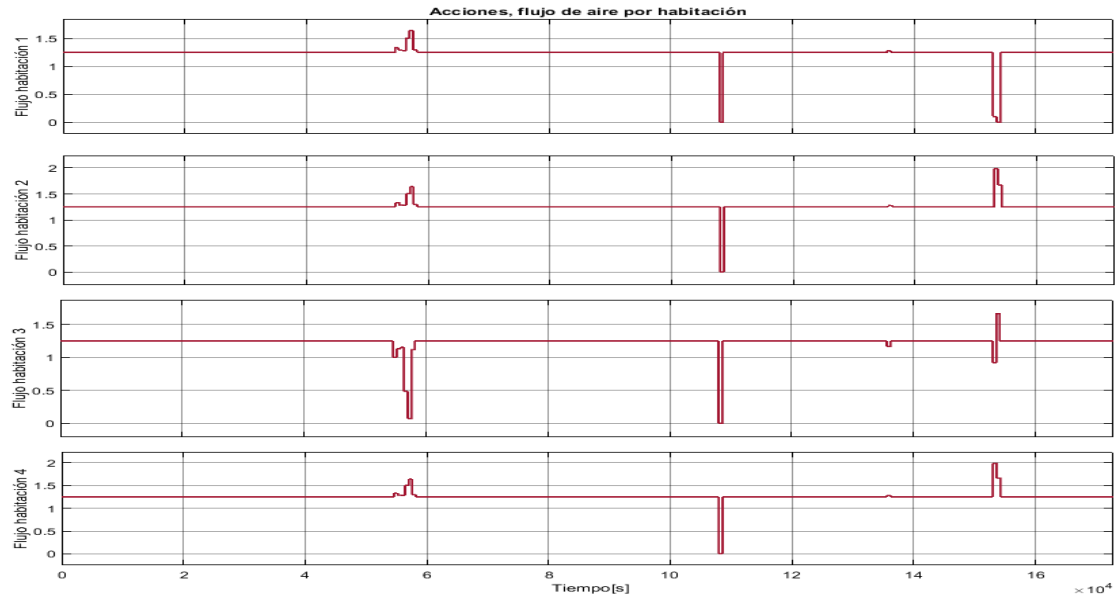


(b) Gráfico mostrando las acciones tomadas por el agente con un tiempo de simulación de 2 días.

Figura 5.20: Resultados utilizando MPC para el caso con 4 zonas, 6 variables de control y restricciones de temperatura, CO2 y humedad.



(a) Gráfico mostrando las acciones realizadas por el agente en un tiempo de simulación de 2 días.



(b) Gráfico mostrando el flujo másico de aire por zona en un tiempo de simulación de 2 días.

Figura 5.21: Resultados acciones utilizando TD3 para el caso con 4 zonas, 11 variables de control y restricciones de temperatura.

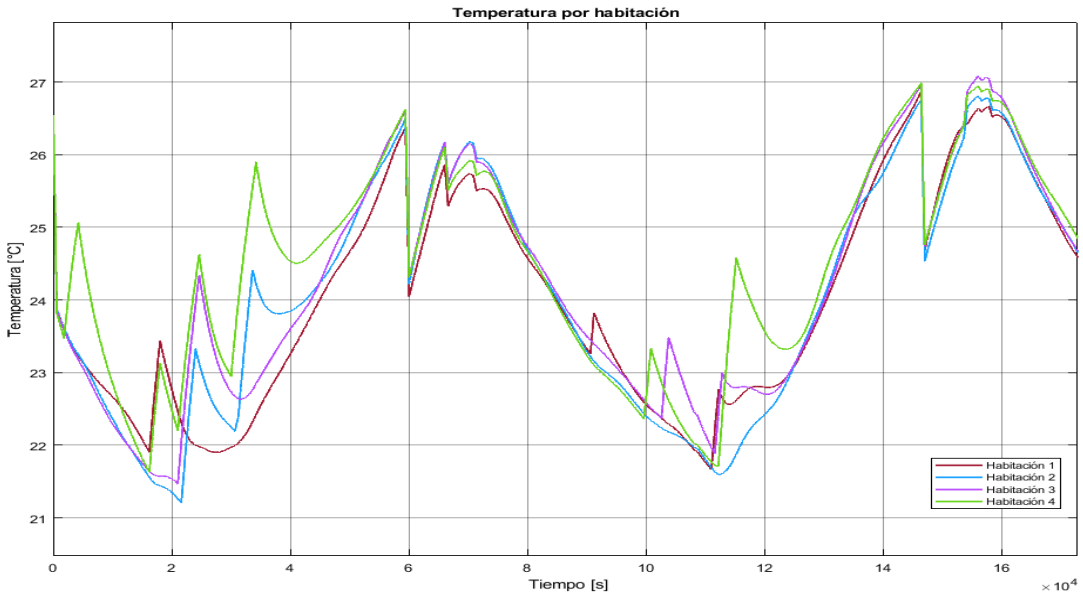


Figura 5.22: Gráfico mostrando los estados a los que llega el agente con un tiempo de simulación de 2 días.

5.1.7.2. Análisis y comparación

Lo primero que tenemos que mencionar, es que si bien son problemas con algunas similitudes, estos tienen diferencias bastante notorias.

Una de ellas es la cantidad de variables a controlar. Para MPC no se está considerando variables de calefacción, lo que ya son 4 variables menos. Además, se está formulando cada flujo másico de las habitaciones como independiente (es decir, como si hubiera un ventilador para cada una de las habitaciones). Esto no está del todo mal formulado, dado que en MPC podemos ser bastante estrictos con las restricciones, que en este caso es la suma de esos flujos. Sin embargo, para aprendizaje reforzado se utilizó el modelo con un solo ventilador y válvulas de zona. Se realizó así pues como TD3 es un método exploratorio, en algún momento la suma de los flujos podría haber llegado a $20[kg/s]$ (los cuatro ventiladores independientes al máximo). Esto podría ser catastrófico para un sistema real (pensando que es un método que puede aprender *online*).

Otra diferencia es que MPC está agendado para mantener restricciones en ciertos horarios, mientras aprendizaje reforzado no.

Como podemos observar en las figuras 5.20.b y 5.21.b, MPC genera un flujo por zona de aproximadamente 0.5. Esto es lo esperado, pues es similar a lo estudiado anteriormente, sin embargo, TD3 genera un poco más del doble de aquello, el cual se mantiene casi constante debido a la restricción superior del flujo de aire y a que no tenemos horarios de funcionamiento. Luego, al observar también la figura 5.21.a, notamos que el enfriamiento realizado es muy puntual y oscilante entre máximo y cero. Mientras que MPC enfría de forma constante.

Luego, en las figuras 5.20.a y 5.22, se observan estados fluídos y estables cercanos a un punto en MPC. No así en TD3 que son estados que oscilan mucho y muy repentinamente. Esto último se debe a las acciones con magnitudes tan grandes tomadas por TD3.

En resumen, en ese caso TD3 no se comportó de buena manera y no logró ser un real punto

de comparación con MPC, como ya lo había hecho antes. Este mal comportamiento podría indicar la necesidad de mejorar la función de recompensa. Tal vez el primer paso a seguir sería no solo penalizar las malas acciones y malos estados, sino que bonificar las buenas.

5.1.8. Octava instancia, restricciones de temperatura, CO2 y humedad relativa en 4 zonas.

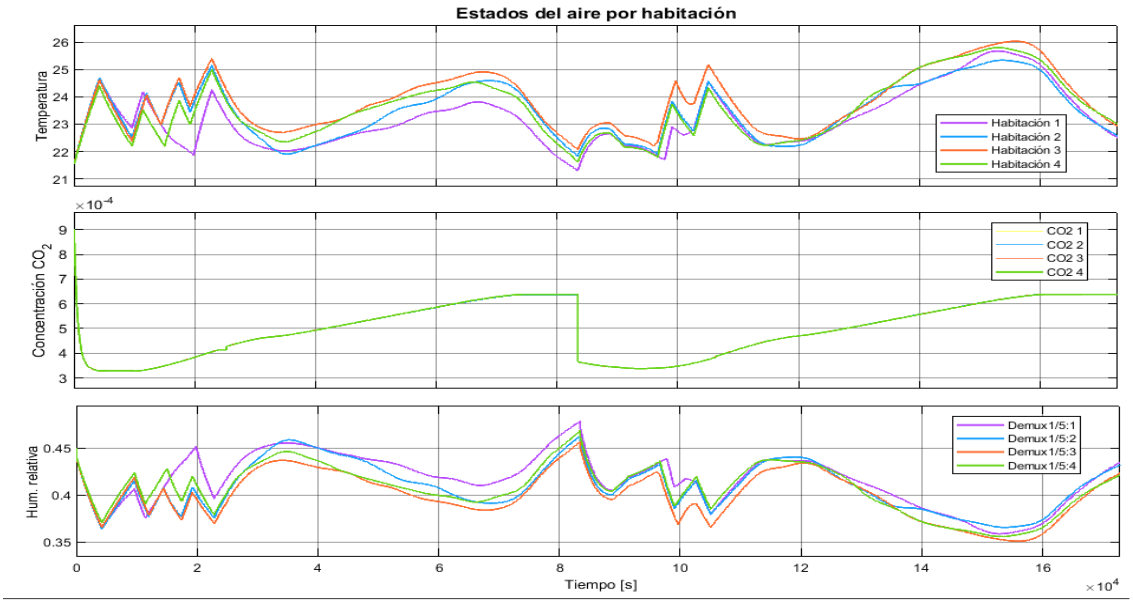
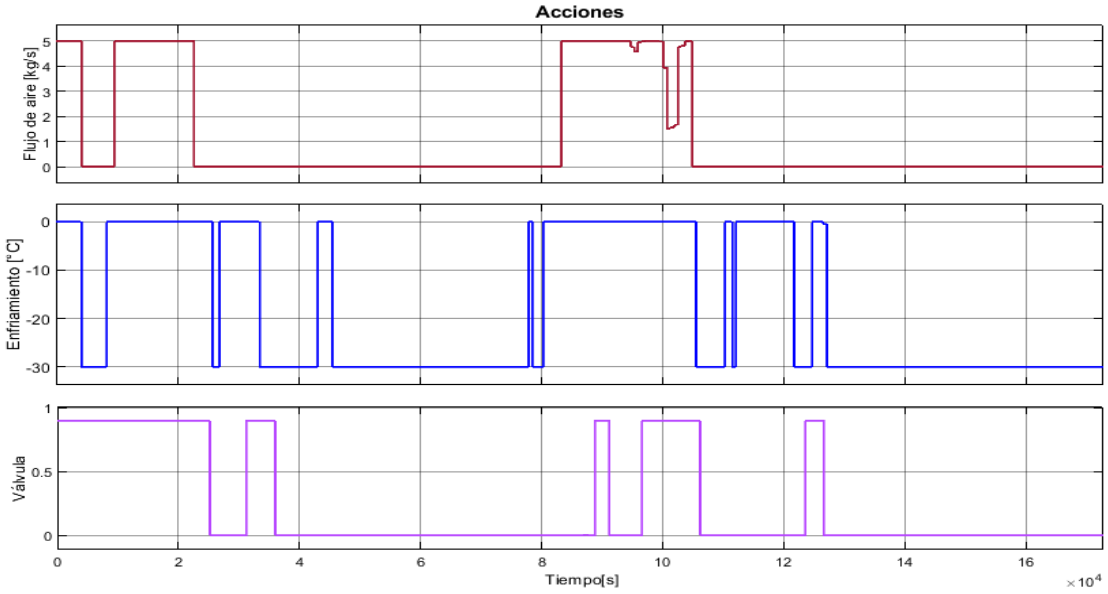


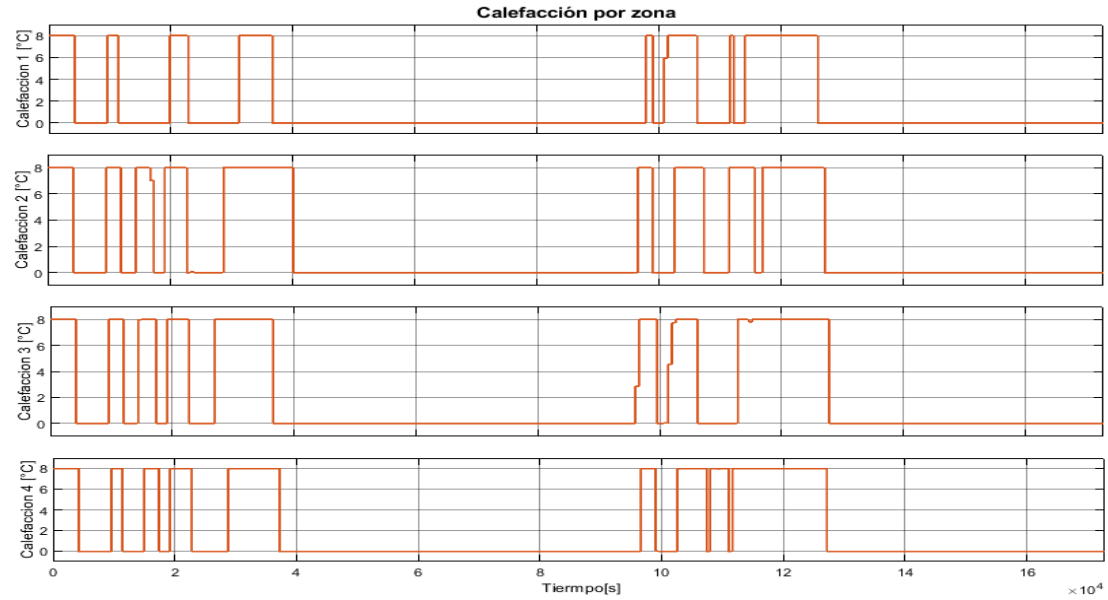
Figura 5.23: Gráfico mostrando los estados a los que llega el agente con un tiempo de simulación de 2 días, en 4 zonas distintas.

Cómo podemos observar en la figura 5.23, los 12 estados se mantienen la mayor parte del tiempo en los rangos deseados, donde la humedad y la concentración de CO_2 no rompen la restricción en ningún momento.

Sin embargo, a pesar de este buen resultado, tenemos que los estados, son bastante ruidosos. Al igual que en el caso anterior, esto se debe a las acciones tan repentinas que toma el agente, acciones que se ven en 5.24. En este sentido, tal vez el siguiente paso es guiar mejor las acciones en la función de recompensa, dado que los estados están dentro de los rangos deseados.



(a) Gráfico mostrando las acciones tomadas por el agente con un tiempo de simulación de 2 días, en 4 zonas distintas.



(b) Gráfico mostrando la calefacción por zona en un tiempo de simulación de 2 días, en 4 zonas distintas.

Figura 5.24: Resultados acciones utilizando TD3 para el caso con 4 zonas, 11 variables de control y restricciones de temperatura, CO2 y humedad.

5.2. Resultados con predicciones

Luego de observar los resultados obtenidos con MPC, se planteó la interrogante de, ¿Qué pasaría si incluimos predicciones del clima en aprendizaje reforzado?. Entonces se repitieron las instancias 3 y 4, solamente con DDPG, dado que es el algoritmo que obtuvo mejores respuestas en dichas instancias, incluyendo predicciones de clima con 6 pasos adelante.

Esta variación significó aumentar la cantidad de observaciones, pues, las predicciones se

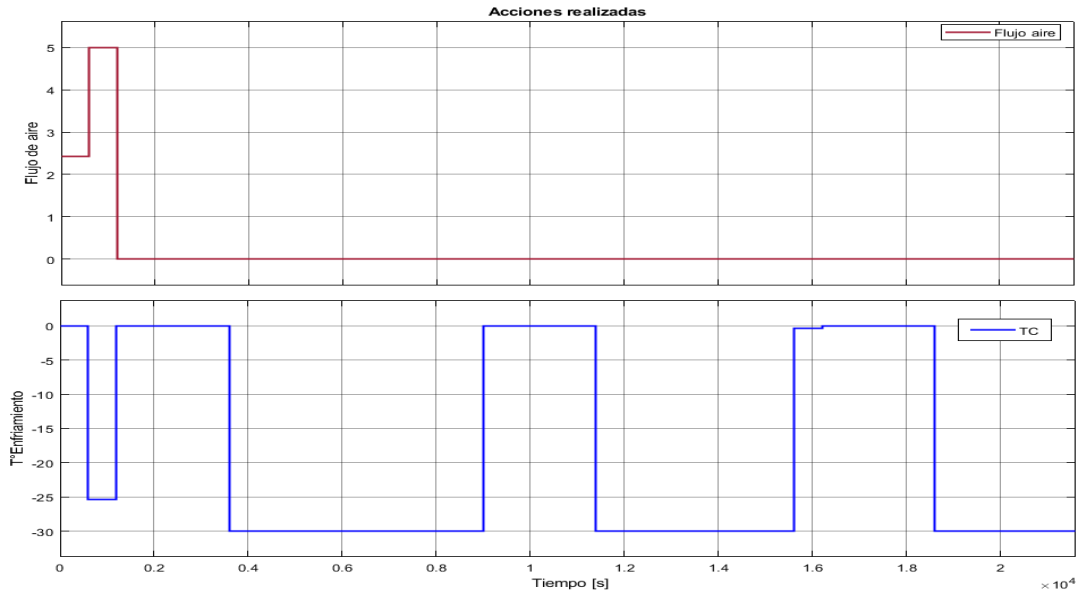
incluyen como observación para el agente. Luego, para ambas instancias pasamos de 4 observaciones a 16 observaciones, dado que en cada instante de clima hay 2 observaciones (temperatura ambiental y generación de calor por uso), entonces a estas 4 observaciones iniciales se le añaden los 6 pasos a futuro, quedando en 16.

Dado que para ambas instancias aumentó considerablemente el número de observaciones, y en dichas instancias se utilizaron redes neuronales relativamente pequeñas, se hace necesario el aumentar la cantidad de neuronas para que el sistema tenga un correcto funcionamiento. Finalmente, en ambas instancias se utiliza un crítico del tamaño de la instancia 4, es decir $[[64, 32], 32]$ y un actor de tamaño $[32, 16]$.

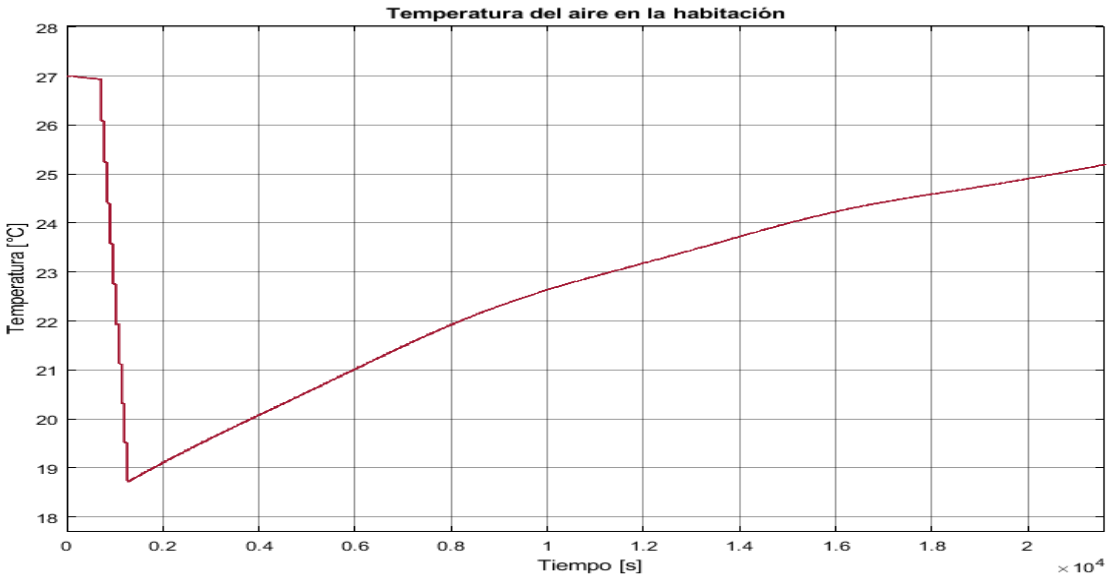
5.2.1. Tercera instancia

Esta instancia se configura exactamente igual que en la sección anterior, con la salvedad que los tamaños de las redes se aumenta a lo ya expuesto anteriormente.

Luego, los resultados se pueden observar en la figura 5.25



(a) Gráfico mostrando las acciones al utilizar predicción con 6 pasos para la instancia 3, utilizando DDPG.



(b) Gráfico mostrando los estados al utilizar predicción con 6 pasos para la instancia 3, utilizando DDPG.

Figura 5.25: Resultados acciones utilizando TD3 para el caso con 4 zonas, 11 variables de control y restricciones de temperatura, CO2 y humedad.

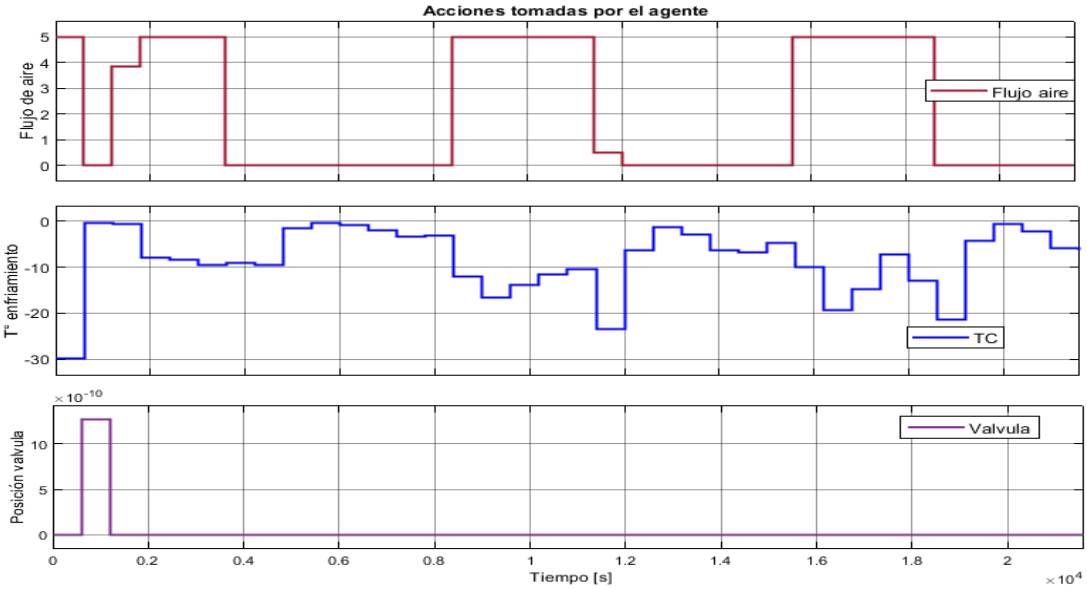
Como se puede observar en la figura 5.25.a, las acciones no son las más eficientes, pues estamos tomando una acción de control efectiva únicamente al comienzo de la simulación, y el resto del tiempo el sistema no está tomando acciones, pues, no hay flujo de aire.

Por otro lado, al ver la figura 5.25.b, la temperatura del aire se mantiene mayor parte del tiempo dentro de los rangos, sin embargo, inicialmente se enfría bastante más que lo debido (aproximadamente $3°C$), y luego de eso aumenta la temperatura por los intercambios de calor (calor generado dentro de la oficina y temperatura externa). Finalmente, se obtiene un peor

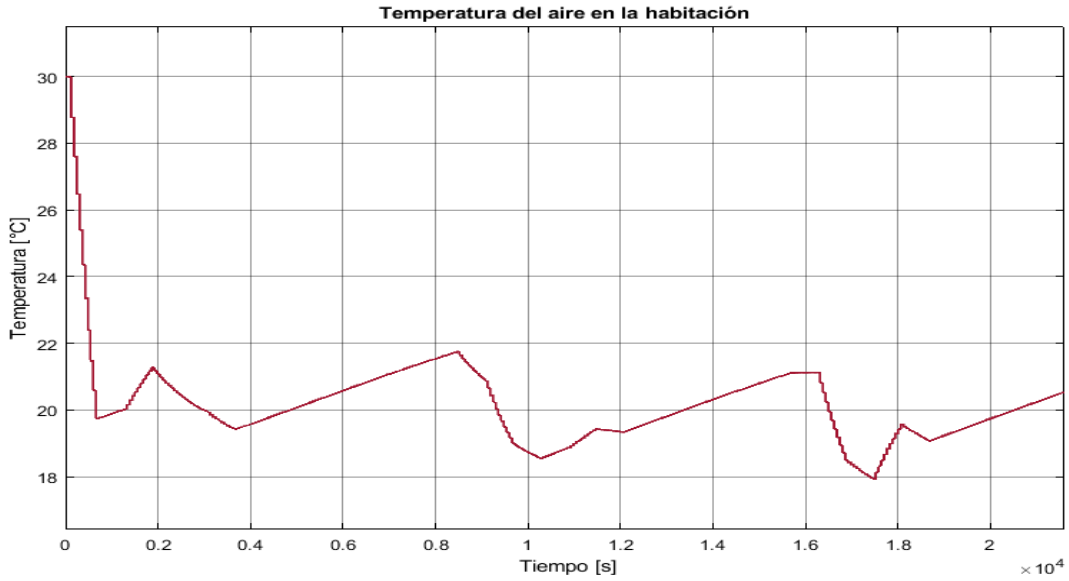
rendimiento que al no incluir las predicciones.

5.2.2. Cuarta instancia

A continuación en la figura 5.26 se muestran los resultados obtenidos para la cuarta instancia al utilizar predicción de clima a 6 pasos.



(a) Gráfico mostrando las acciones al utilizar predicción con 6 pasos para la instancia 4, utilizando DDPG.



(b) Gráfico mostrando los estados al utilizar predicción con 6 pasos para la instancia 4, utilizando DDPG.

Figura 5.26: Resultados acciones utilizando TD3 para el caso con 4 zonas, 11 variables de control y restricciones de temperatura, CO2 y humedad.

Como se observa en la figura 5.26.a, las acciones son considerablemente malas, debido a que el flujo de aire oscila entre el mínimo y el máximo, casi sin existir puntos intermedios.

Además, la válvula se encuentra completamente abierta, mientras que la temperatura de enfriamiento se comporta de buena manera.

Al observar la figura 5.26.b, es claro que el sistema se encuentra fuera del rango de restricciones en casi todo momento, dado que el sistema se encuentra enfriando más de lo necesario, esto es malo tanto en el sentido de los estados como en el consumo energético innecesario.

Finalmente, al compararlo con los resultados sin predicciones, es claro que se obtiene un peor rendimiento al utilizar las predicciones.

5.3. Resultados *toolbox*

En este trabajo se realizan los entrenamientos utilizando el *software* Matlab, y más aún específicamente el *toolbox* de aprendizaje reforzado. A pesar de que el programa ya cuente con un *toolbox*, este es relativamente nuevo y aún se encuentra en una posición poco amigable para el usuario.

Para ello, se estructuran los códigos utilizados de modo que exista un código principal o *main*, que permita definir una serie de parámetros que podría querer modificar el usuario, y otros necesarios que tiene que definir. Además permite indicar una sub-carpeta donde se encuentre el modelo de *simulink* y donde pueda guardar el agente en caso que así lo requiera, como se observa en el código 5.1 en la línea 14.

Código 5.1: Inicio del codigo base

```
1
2  %& Clearing the workspace
3 clear all
4  % Set to true, to resume training from a saved agent
5 resumeTraining = 1;
6 save_agent = false;
7 use_parallel = 1;
8  %device for critic & actor
9 device = "cpu";
10  %Save & Load options
11 save_agent_name = "train_agent_td3_3.mat";
12 load_agent_name = "train_agent_td3_2.mat";
13  %folder where to load/save agents and models.
14 subcarpeta = "";
15  %quantity of observations and actions
16 numObservations = 5*4 + 15 ;%5 states per zone(T_air, T_solid, CO2, hum_rel, hum_abs)
17 numActions = 2 + 4 + 4;
18  %limits for obs and acts. (in continuous)
19 limit_act_low = [0; 0; 0; 0; -30 ; 0; 0; 0; 0; 0];
20 limit_act_h = [5; 5; 5; 5; 0 ; 8; 8; 8; 8 ;0.9];
21 obs_low = -40;
22 obs_high = 1e7;
23  %defining scale variable
24 scale = limit_act_low + limit_act_h;
```

Además en 5.1 hay parámetros modificables que son universales para cualquier sistema, son opcionales y se presentan a continuación:

- resumeTraining = *boolean*, indica si queremos cargar un agente ya entrenado, y comenzar un entrenamiento con la red ingresada.
- save_agent = *boolean*. Permite guardar el agente luego de finalizado el entrenamiento.
- use_parallel = *boolean*. Define si utilizaremos la computación paralela o no.
- device = cpu/gpu. Define que dispositivo utilizar para entrenar las redes.
- save_agent_name = *string*. Indica el nombre con que guardar el agente, en caso que save_agent sea verdadero.
- load_agent_name = *string*. Indica el nombre del agent a cargar , en caso que resumeTraining sea verdadero.

Por otro lado, hay parámetros obligatorios para el funcionamiento del sistema, que se encuentran repartidos en 5.1 5.2 5.3 y se enlistan a continuación:

- Cantidad de observaciones(lineas 16 5.1).
- Cantidad de acciones(linea 17 5.1).
- Limites de las acciones y observaciones(lineas 19 y 20 5.1).
- Tiempo de muestreo y simulación (lineas 2, 3 y 4 5.2).
- Cantidad de episodios (linea 5 5.2).
- Nombre del modelo a utilizar(linea 2 5.3).

Estas opciones, en conjunto con una buena documentación en el código permitirán al usuario una experiencia más fluida a la hora de querer entrenar su propio agente.

Código 5.2: Se definen los tiempos de simulación y la magnitud del ruido a utilizar

```
1  %Sample Time & Simulation Time
2  Ts = 60*1; %2 min - HVAC System sample Time
3  Ts2 = 60*10; %n min - Neural network Sample time
4  Tf = 3600*48; % n horas  - Simulation Time
5  maxepisodes = 3000;% max number of episodes to stop learning
6  StopReward = 2; %Episode reward to stop learning
7  maxsteps = ceil(Tf/Ts2); % Cantidad de pasos en un episodio
8  %Noise options
9  standard_deviation = [0.5 ;0.5 ;0.5 ;0.5 ;0.1*sqrt(Ts2); 0.6;0.6;0.6;0.6; 0.1]/sqrt(Ts2);
10 min_std_dv = [0.015/1.5; 0.015/1.5; 0.015/1.5; 0.015/1.5; 0.5/1.5; 0.02/1.5; 0.02/1.5;
       ↪ 0.02/1.5; 0.02/1.5; 0.03/1.5]/(sqrt(Ts2));
11 %min_std_dv = 0;
12 decay_rate = 2e-5;
13 %RL Layers
14 criticlayers = [2,1];
```

```
criticNeurons = [400, 300, 300];
actorlayers = 2;
actorNeurons = [400, 300];
```

Por otro lado, en 5.2, se observan más parámetros que se pueden definir como el *StopReward* que es el valor de recompensa que detendrá el entrenamiento, por ejemplo, si $StopReward = 2$ y en el último episodio se obtuvo una recompensa $r_t = 3$, entonces se detendrá el entrenamiento.
Además se puede definir la magnitud de la desviación estándar para el ruido de exploración, el mínimo de esta y que tan rápido decae a ese mínimo. Este ruido es aquel que se añade a la acción tomada y permite explorar diversas combinaciones de acciones. Finalmente, se pueden modificar los parámetros de red neuronal, cambiando fácilmente el número de neuronas o capas ocultas a utilizar para cada una de las redes.

Finalmente 5.3 es donde se declara el ambiente con el cual se trabajará.

Código 5.3: Se define cual es el bloque a utilizar y el ambiente del sistema

```
%% Load and open the system in simulink
blk = "hvac_maborrelli_singapur_nzones_TCH";
mdl = subcarpeta + blk;

%% Define model parameters
load('values4z_tch.mat'); % uncomment this for TCH (temp, co2, hum) params
params = params1;

%% Reinforcement learning
%Se define cual es el ambiente en el sistema a utilizar.
[obsInfo,actInfo,env] = makeenv(mdl,numObservations,numActions,limit_act_low,
    ↪ limit_act_h,obs_low,obs_high);

%Función para reiniciar el sistema
%Esta función es útil cuando se quiere usar distintas condiciones iniciales
%en cada inicio de un episodio.
env.ResetFcn = @(in) localResetFcn(in, blk, T_s,H_sup,Hs_struct);
```

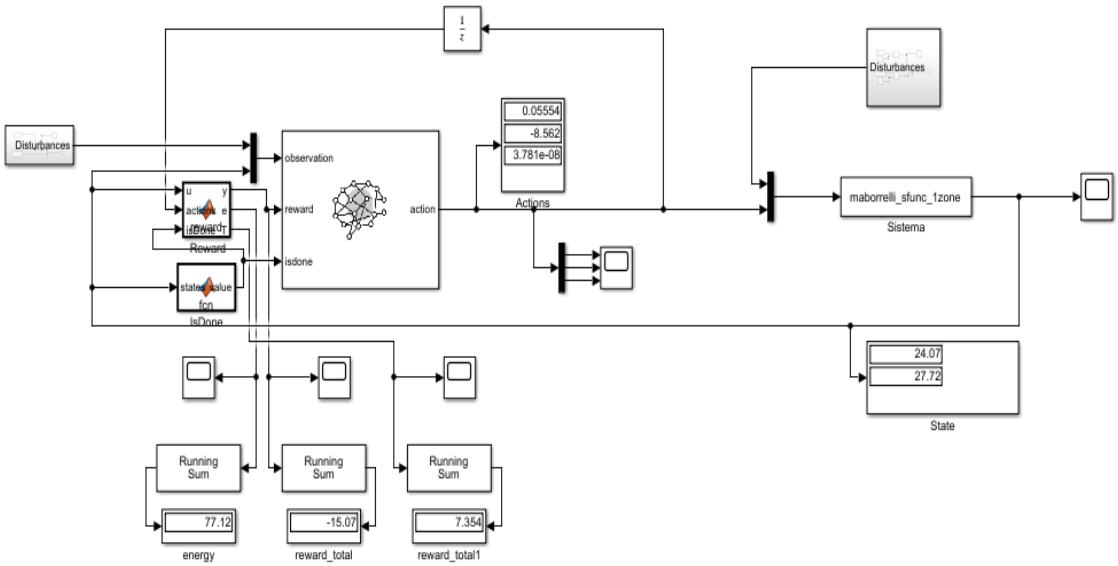


Figura 5.27: Modelo de simulink para el caso de 3 variables.

En la figura 5.27 tenemos el sistema con 3 variables, modelo se puede observar que existe un bloque llamado *Sistema*, este bloque es el modelo de la planta y puede ir cualquier otra planta en su lugar. Por otro lado ya se encuentran las funciones de recompensa e *IsDone*, donde en el caso de usar otra planta, solo habría que modificarlas. La función *IsDone*, es la que genera un reinicio del sistema, este reinicio se da cuando se termina el episodio, o cuando nosotros definimos que el episodio está desempeñandose muy fuera de los rangos deseados, por ejemplo, llegando a temperaturas negativas. La función es importante, pues al alejarnos mucho de los estados y no volver pronto a ellos, se generan muchas muestras con un mal rendimiento, aumentando la probabilidad que el *minibatch* utilizado para entrenar posea muchas de estas muestras.

Además, se observa que la función de recompensa tiene 3 entradas, los estados, las acciones y el resultado de la función *IsDone*. Esto puede mantenerse de esta misma forma independiente del sistema, y en el caso de no utilizar alguna de estas entradas simplemente no se hace la llamada a la variable dentro de la función.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

En el presente trabajo se realizó un estudio previo del estado del arte de formas de controlar un sistema de climatización y de los avances realizados en cuanto al control de dichos sistemas, utilizando aprendizaje reforzado. Además, se realiza un estudio de qué es el aprendizaje reforzado, cómo funciona y en qué se basa para funcionar.

Con esto último, se explica el porqué se toma la decisión de utilizar este método para controlar los sistemas de climatización expuestos en este trabajo y cualquier otro que se quisiera controlar. Por otro lado, se explica el funcionamiento principal del sistema de climatización a utilizar para el estudio, exponiendo y explicando las distintas dinámicas físicas que se implementan, esclareciendo a su vez, cual es el rol que toma cada una de las acciones en el cambio de las dinámicas.

Luego, se escogen tres algoritmos de aprendizaje reforzado distintos los cuales entran en la categoría de *model free* y estos son utilizados para entrenar un controlador, capaz de tomar decisiones con respecto a que acción tomar para llevar al sistema a un estado deseado.

Se realizó el estudio del rendimiento de cada uno de los tres algoritmos escogidos en distintas variantes del problema, las cuales consistían en tomar el problema desde el caso más sencillo posible y, desde aquel punto de partida, comenzar a complicar el problema, agregando variables de control y restricciones paulatinamente hasta llegar al caso más complejo, el cual en este trabajo fue tomar el sistema con 4 zonas o habitaciones, con las tres restricciones propuestas al comienzo de este trabajo, es decir, temperatura, concentración de CO_2 y humedad relativa.

Los resultados de este estudio indican que, en efecto, podemos controlar un sistema de climatización utilizando aprendizaje reforzado, donde el desempeño del controlador va a depender en gran medida de nuestra formulación de la función de recompensa, tal como se experimentó al pasar de la configuración de 4 variables y 2 restricciones, a la de 4 variables y 3 restricciones, donde la primera no realizaba de buena manera la acción de calentar y, al modificar la función de recompensa, se obtuvieron muy buenos resultados para la configuración siguiente, donde el sistema utilizó de buena manera la calefacción en los momentos deseados. Y que no siempre se obtendrán buenos resultados, como es el caso de las últimas

dos instancias.

Además, el control realizado no solo mantiene los estados y, por lo tanto, la comodidad de las personas en su interior, si no que además es eficiente energeticamente, llegando al punto de ser comparable con métodos de optimización como lo sería MPC. Siguiendo la misma temática, al comparar el método utilizado con métodos de optimización como MPC, se obtienen resultados bastante similares, donde MPC sigue siendo, ligeramente, mejor que el método utilizado. Sin embargo, la diferencia es tan pequeña que tenemos que plantearnos otros puntos de comparación, como por ejemplo, qué tanto costó llegar a la solución y el conocimiento que se tiene del sistema. Para MPC hubo que hacer todo un cálculo matemático y metódico complejo con el fin de llegar a la solución (a la cual se pudo llegar gracias al buen conocimiento del sistema), mientras para la solución entregada en este trabajo basta con tener una idea de como funciona el sistema y a que queremos llegar para poder formular la función de recompensa. Por otro lado, si somos aún más exhaustivos con la función de recompensa, es posible obtener mejores resultados comparables al nivel de MPC. Luego, la solución propuesta en este trabajo podría llegar a ser mejor, debido a que se obtienen resultados similares con un trabajo matemático mucho más simple, además de que el algoritmo tiene espacio para seguir mejorando.

Esto último, nos indica que utilizar aprendizaje reforzado en este tipo de problemas, nos permite llegar a una solución óptima. Esto bajo la restricción de formular una función de recompensa que sea una guía para el agente de aprendizaje reforzado, de modo que solo con esta y su interacción con el sistema pueda determinar qué acciones tomar y cuales estados son los deseados.

Luego, analizando el rendimiento de los algoritmos implementados, tenemos que DDPG y TD3 se comportan de manera similar. Esto era lo esperado pues TD3 deriva de DDPG. Tenemos en primer lugar que DDPG actúa mejor que TD3 en los casos más sencillos, sin embargo, en dichos casos la diferencia es pequeña, no así en los casos más complejos, donde TD3 actúa mejor que DDPG y las diferencias son mayores, tanto cuantitativamente como cualitativamente. Sin embargo, tenemos que considerar que TD3 es un algoritmo más demandante computacionalmente, es más robusto y entrena de una manera más estable, pero a su vez entrena considerablemente más lento que DDPG, por lo que, al hacer un balance con esto en consideración, podemos llegar a que el uso del uno u el otro es circunstancial y dependerá netamente de lo que necesitemos en el instante, donde ambos algoritmos entregarán buenos resultados con los recursos que tengamos a mano.

Por otro lado, PPO no logró controlar de buena manera ninguna de las variantes del problema, donde en los casos iniciales se notaba una cercanía con los estados deseados, pero luego obtuvo unos resultados realmente deficientes. Esto puede deberse a varios factores, como el no utilizar hiper parámetros correctos o no haber formulado de manera correcta la red neuronal. De todas formas, teniendo a los otros dos algoritmos al alcance, no se recomendaría utilizar PPO en vez de alguno de los otros dos.

Además, se realizó una prueba del algoritmo DDPG, utilizando predicciones, donde claramente se obtuvieron peores resultados que al no utilizarlas. Este resultado no nos permite descartar el uso de predicciones completamente, pero, si es un buen indicador para tomar

en cuenta la cantidad a utilizar, puede ser que en este caso se tomaran demasiados pasos y por ello se viera mermado el rendimiento del algoritmo. Por otro lado, no solo se obtienen peores resultados, sino que también se utiliza más potencia computacional, pues las redes neuronales son de mayor tamaño.

Finalmente, los códigos utilizados fueron estructurados de tal forma de tener un código principal, donde se pueden modificar los parámetros a utilizar, el modelo de simulink, si queremos guardar el agente entrenado, si queremos comenzar el entrenamiento con un agente ya entrenado anteriormente, entre otros. Además, el código se deja documentado de tal forma que sea de fácil lectura para cualquier persona que quiera utilizarlo y/o mejorarlo.

Por otro lado, los bloques de simulink usados se dejan ordenados de tal forma que sea de fácil entendimiento dónde modificar para incluir un nuevo sistema.

Estas dos características son las que definen al principio de *toolbox* realizado en este trabajo, el cual fue utilizado para constante iteración entre distintos modelos a lo largo de este trabajo, evitando así una gran cantidad de tiempo al tener que cambiar solo una cantidad de parámetros para escoger otro sistema.

6.2. Trabajo Futuro

Luego de finalizado este trabajo, podemos identificar puntos desde lo cuales podemos continuar estudiando y mejorando. En primer lugar, un paso a seguir para entregarle mayor universalidad al código es, el poder implementar una función que, tomando los limites mínimos y máximos de las acciones, pueda determinar la magnitud de la desviación estándar del ruido para TD3 y DDPG, de tal forma de que partamos con un ruido de $x\,\%$ en relación a la amplitud máxima de la acción y terminemos en un $y\,\%$, con $x > y$. Esto es importante, dado que la magnitud del ruido está dado por el modelo de Ornstein-Uhlenbeck, y no es trivial cual es la desviación estándar necesaria para llegar a una cierta magnitud.

Por otro lado, con respecto al controlador entrenado se propone instaurar horarios de uso, donde se puede implementar un reloj en la recompensa y a su vez como observación, donde podemos definir horarios que tienen un uso reducido, como lo es la noche, o inclusive días como feriados o fin de semanas. Así el sistema tendrá uso eficiente de la energía en aquellos momentos, para lograr esto podemos expandir el rango de las restricciones (por ejemplo en vez de $[22, 25]°C$, aumentar a $[15, 30]°C$), entregándole al sistema mayor holgura para tomar (o no tomar) acciones que minimicen el uso de energía.

También se propone mejorar la función de recompensa para los casos con 4 zonas.

Finalmente, se propone estudiar una posible falla en la estructura de la red neuronal en el algoritmo de PPO, dado que los resultados obtenidos difieren mucho con los mostrados en los estudios revisados.

Bibliografía

[1] L. Yu, S. Qin, M. Zhang, C. Shen, T. Jiang, and X. Guan, "A review of deep reinforcement learning for smart building energy management," *IEEE Internet of Things Journal*, vol. 8, p. 12046–12063, Aug 2021.

[2] L. Pérez-Lombard, J. Ortiz, and C. Pout, "A review on buildings energy consumption information," *Energy and Buildings*, vol. 40, no. 3, p. 394–398, 2008.

[3] Y. Ma, J. Matusko, and F. Borrelli, "Stochastic model predictive control for building hvac systems: Complexity and conservatism," *IEEE Transactions on Control Systems Technology*, vol. 23.1, p. 101–116, 2015.

[4] A. Parisio, M. Molinari, D. Varagnolo, and K. H. Johansson, "Scenario-based predictive control approach to building hvac management systems," *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, 2013.

[5] Z. Wu, Q.-S. Jia, and X. Guan, "Optimal control of multiroom hvac system: An event-based approach," *IEEE Transactions on Control Systems Technology*, 2015.

[6] B. Sun, P. B. Luh, Q.-S. Jia, Z. Jiang, F. Wang, and Chen, "Building energy management: Integrated control of active and passive heating, cooling, lighting, shading, and ventilation systems," *IEEE Transactions on Automation Science and Engineering*, vol. 10.3, 2013.

[7] E. B. Kraus and J. A. Businger, *Atmosfere-Ocean Interaction*. 1994.

[8] T. Gonsalves and J. Upadhyay, "Artificial intelligence for future generation robotics," 2021.

[9] H. Alashwal, M. El Halaby, J. J. Crouse, A. Abdalla, and A. A. Moustafa, "The application of unsupervised clustering methods to alzheimer's disease," *Frontiers in Computational Neuroscience*, vol. 13, 2019.

[10] R. S. Sutton, F. Bach, and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press Ltd, 2018.

[11] A. Burkov, *The hundred-page machine learning book*. Andriy Burkov, 2019.

[12] Matlab, "Reinforcement learning diagram," 2021.

[13] G. Gao, J. Li, and Y. Wen, "Energy-efficient thermal comfort control in smart buildings via deep reinforcement learning," *CoRR*, vol. abs/1901.04693, 2019.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

[15] Matlab, "Train biped robot to walk using reinforcement learning agents," 2021.

[16] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in

actor-critic methods," *CoRR*, vol. abs/1802.09477, 2018.

[17] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms.," *In International Conference on Machine Learning*, p. 387–395, 2014.

[18] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, p. 2094– 2100, 2015.

[19] H. Hasselt, "Double q-learning," *Advances in neural information processing systems*, vol. 23, pp. 2613–2621, 2010.

[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, 2016.

[21] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Reinforcement Learning*, vol. 8, p. 5–32, 1992.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[23] Mathworks, "Ppo agent." URL:https://la.mathworks.com/help/reinforcement-learning/ug/ppo-agents.html.

[24] Z. Wu, Q.-S. Jia, and X. Guan, "Energy-efficient buildings facilitated by microgrid," *IEEE Transactions on Smart Grid*, 2010.

[25] J. Leitão, P. Gil, B. Ribeiro, and A. Cardoso, "A survey on home energy management," *IEEE Access*, vol. 8, pp. 5699–5722, 2020.

[26] G. Serale, M. Fiorentini, A. Capozzoli, D. Bernardini, and A. Bemporad, "Model predictive control (mpc) for enhancing building and hvac system energy efficiency: Problem formulation, applications and opportunities," *Energies*, vol. 11.3, 2018.

[27] M. Han, R. May, X. Zhang, X. Wang, S. Pan, D. Yan, Y. Jin, and L. Xu, "A review of reinforcement learning methodologies for controlling occupant comfort in buildings," *Sustainable Cities and Society*, vol. 51, 08 2019.

[28] D. Zhang, X. Han, and C. Deng, "Review on the research and practice of deep learning and reinforcement learning in smart grids," *CSEE Journal of Power and Energy Systems*, vol. 4, no. 3, pp. 362–370, 2018.

[29] T. Wei, Y. Wang, and Q. Zhu, "Deep reinforcement learning for building hvac control," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2017.

[30] Y. Ma, G. Anderson, and F. Borrelli., "Distributed predictive control approach to building temperature regulation," *Proceedings of the 2011 American Control Conference*, 2011.

[31] D. M. Carpintero, "(trabajo no publicado)."

[32] Y. Ma, A. Kelman, A. Daly, and F. Borrelli, "Predictive control for energy efficient buildings with thermal storage," *IEEE Control Systems*, vol. 32.1, p. 44–64, 2012.

[33] A. Kelman and F. Borrelli., "Bilinear model predictive control of a hvac system using sequential quadratic programming," *IFAC Proceedings Volumes*, vol. 44.1, p. 9869–9874, 2011.

[34] R. Muszynski, "The role of reward signal in deep reinforcement learning," 08 2020.

[35] D. Azuatalam, W.-L. Lee, F. de Nijs, and A. Liebman., "Reinforcement learning for whole-building hvac control and demand response," *Energy and AI*, vol. 2, 2020.

[36] Jatin, "Which activation function for output layer?." Cross Validated. URL:https://stats.stackexchange.com/q/255115 (version: 2017-01-08).

[37] MathWorks, "Train ppo agent to land rocket - matlab example." URL:https://www.mathworks.com/help/reinforcement-learning/ug/train-ppo-agent-to-land-rocket.html.

[38] S. Kobayashi, "Train ppo agent to swing up and balance pendulum - matlab forum," 2021. URL:https://it.mathworks.com/matlabcentral/answers/707088-train-ppo-agent-to-swing-up-and-balance-pendulum.

[39] M. Boschini, L. Bonicelli, P. Buzzega, A. Porrello, and S. Calderara, "Class-incremental continual learning into the extended der-verse," *CoRR*, 2022.

[40] G. Shi, J. Chen, W. Zhang, L. Zhan, and X. Wu, "Overcoming catastrophic forgetting in incremental few-shot learning by finding flat minima," *CoRR*, vol. abs/2111.01549, 2021.

[41] Dirección General De Aeronáutica Civil, "Datos de 48 horas recientes," 2021.

Anexos

Anexo A. Perturbaciones utilizadas y MPC completo

A continuación se muestran los gráficos (A.1, A.2 y A.3) que contienen las perturbaciones utilizadas para cada una de las instancias descritas en este trabajo.

A.1. Perturbaciones para 1,2 y 3 variables

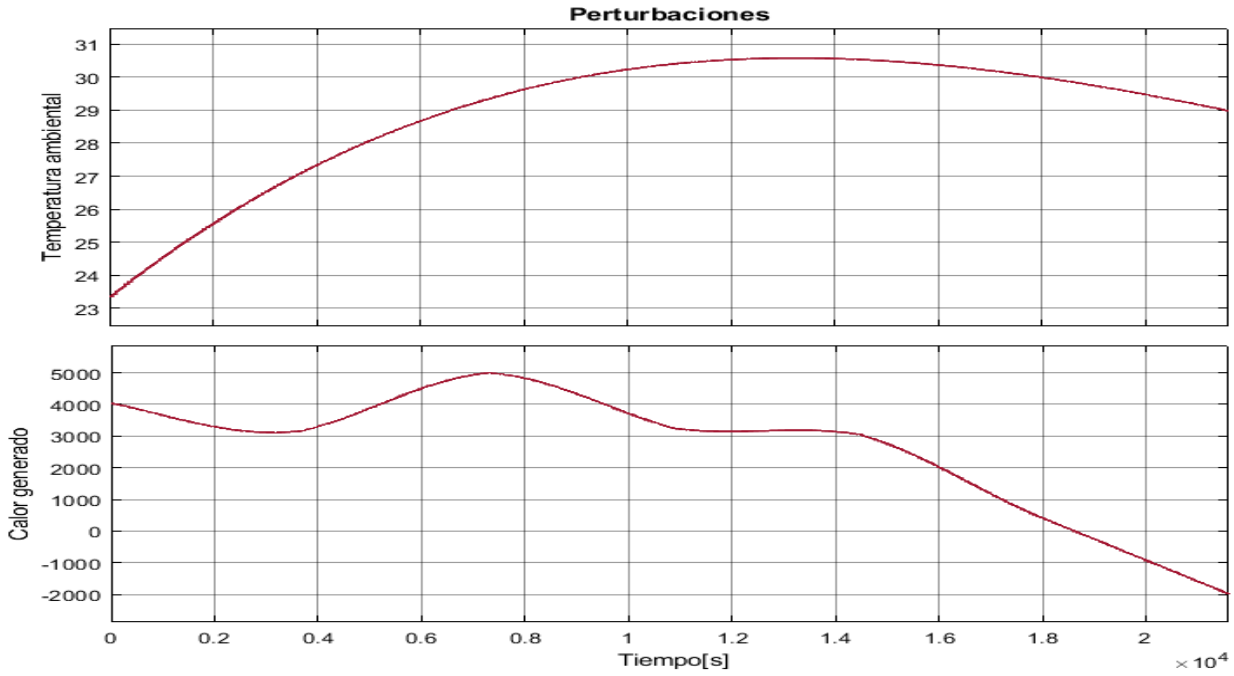


Figura A.1: Gráfico mostrando las perturbaciones utilizadas para entrenar el agente en el caso con restricciones de temperatura.

A.2. Perturbaciones para 4 variables con restricciones de temperatura y CO2

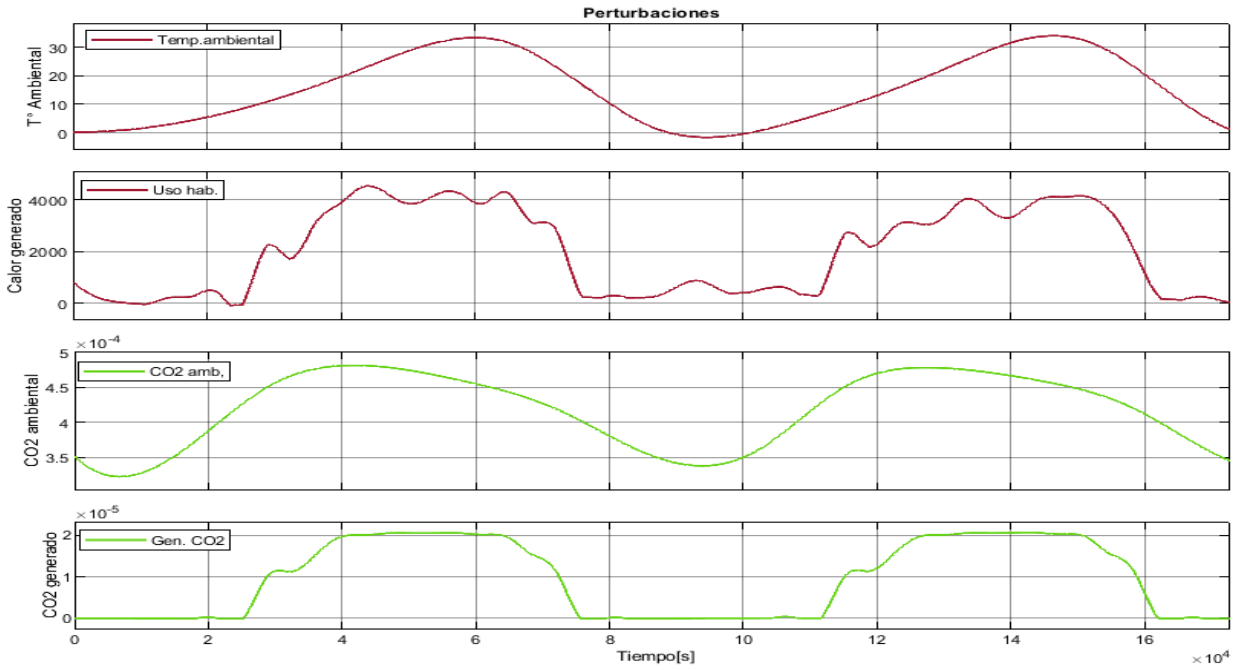


Figura A.2: Gráfico mostrando las perturbaciones utilizadas para entrenar el agente en el caso con restricciones de temperatura y CO2.

A.3. Perturbaciones para 4 variables con restricciones de temperatura, CO2 y humedad

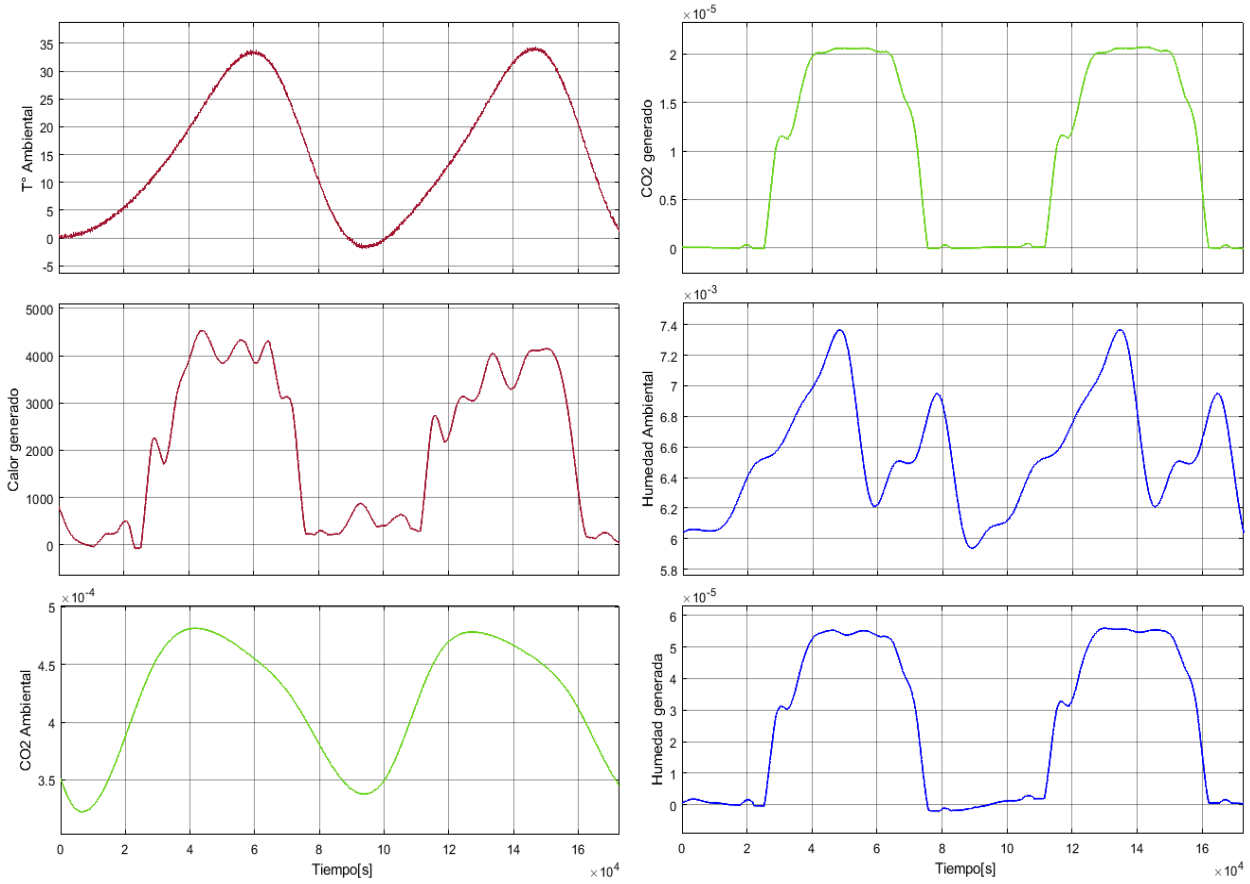
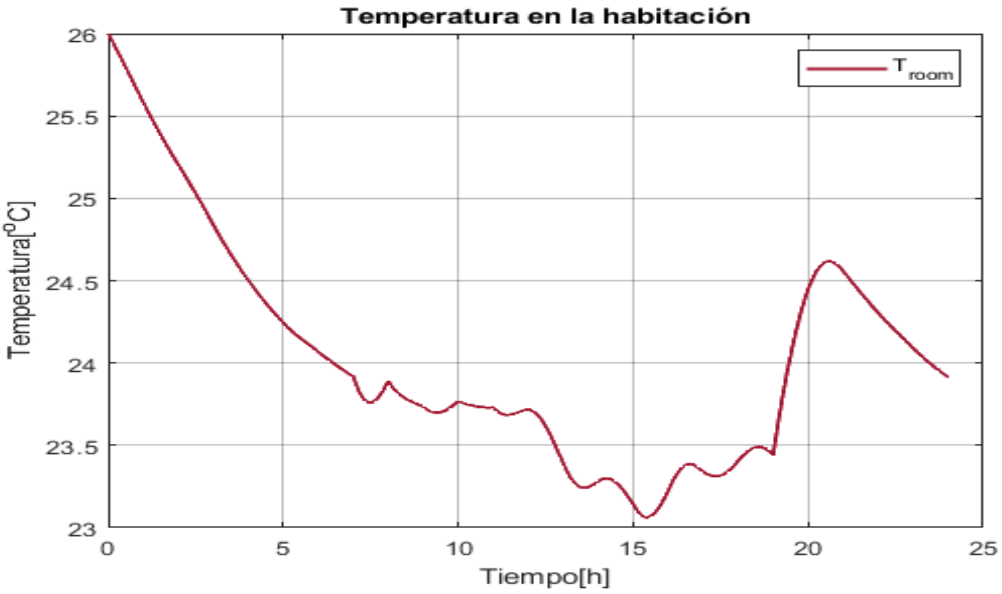
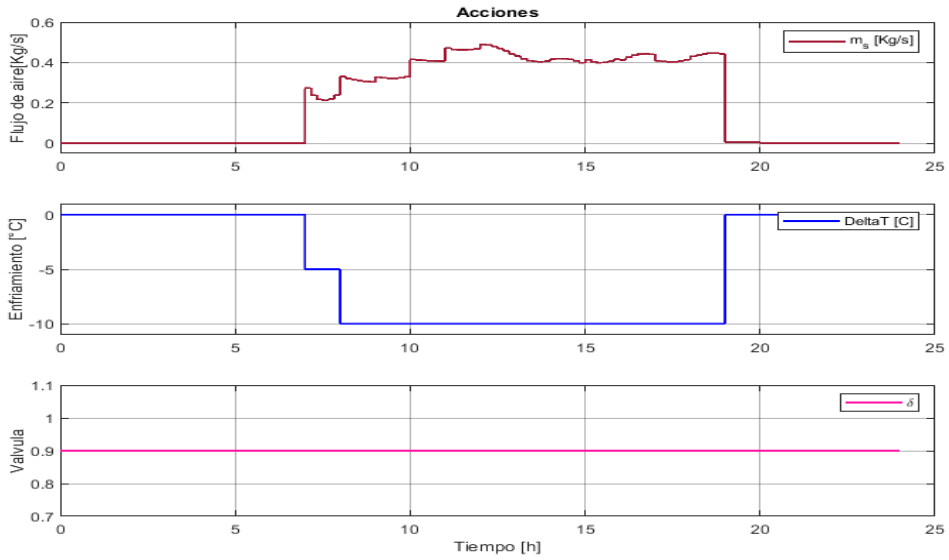


Figura A.3: Gráfico mostrando las perturbaciones utilizadas para entrenar el agente en el caso con restricciones de temperatura, CO2 y humedad.

A.4. Resultado MPC 3 variables día completo



(a) Gráfico mostrando la temperatura a los que llega MPC con un tiempo de simulación de 1 día.



(b) Gráfico mostrando las acciones tomadas por MPC con un tiempo de simulación de 1 día.

Figura A.4: Resultados utilizando MPC para el caso con 3 variables, simulación de un día completo.