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

OPTIMIZACIÓN DE UN PROBLEMA INTEGRAL DE CONSTRUCCIÓN DE CAMINOS, PLANIFICACIÓN Y DISTRIBUCIÓN ESPACIAL DE COSECHA FORESTAL MEDIANTE LA APLICACIÓN DE UN MODELO DE DEEP REINFORCEMENT LEARNING

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL

CRISTÓBAL NICOLÁS SILVA VERA

PROFESOR GUÍA:
Andrés Weintraub Pohorille

PROFESOR CO-GUÍA:
Jaime Carrasco Barra

COMISIÓN:
Juan Velásquez Silva

Este trabajo ha sido parcialmente financiado por:
Fondo Nacional de Desarrollo Científico y Tecnológico (FONDECYT)

SANTIAGO DE CHILE
2023

RESUMEN DE LA MEMORIA PARA OPTAR AL TITULO DE
INGENIERO CIVIL INDUSTRIAL
POR: CRISTÓBAL NICOLÁS SILVA VERA
FECHA: 2023
PROFESOR GUÍA: ANDRÉS WEINTRAUB POHORILLE
PROFESOR CO-GUÍA: JAIME CARRASCO BARRA

OPTIMIZACIÓN DE UN PROBLEMA INTEGRAL DE CONSTRUCCIÓN DE CAMINOS, PLANIFICACIÓN Y DISTRIBUCIÓN ESPACIAL DE COSECHA FORESTAL MEDIANTE LA APLICACIÓN DE UN MODELO DE DEEP REINFORCEMENT LEARNING

La administración forestal es esencial para garantizar un manejo sostenible y eficiente de los recursos forestales. Este estudio se centra en la optimización de la toma de decisiones en la administración forestal, específicamente en la cosecha de rodales y la construcción de caminos, considerando la maximización de las utilidades de los administradores forestales y la satisfacción de las demandas de madera. La investigación aborda la comparación de dos enfoques: modelos matemáticos de programación entera mixta (MIP) y agentes de aprendizaje por refuerzo profundo o deep reinforcement learning(DRL).

Se desarrolló un modelo MIP para determinar el calendario óptimo de cosechas y construcción de caminos en un bosque pequeño. Posteriormente, se diseñó un entorno personalizado que representa los estados del bosque para entrenar agentes de DRL utilizando Deep Q-learning (DQN). Los agentes DQN y el modelo MIP se compararon en términos de convergencia y rendimiento en un ejemplo de bosque pequeño, demostrando que los agentes de DRL pueden alcanzar soluciones similares a la obtenida mediante el modelo MIP con vastos episodios de entrenamiento.

Los resultados de esta investigación sugieren que los agentes de DRL, especialmente aquellos basados en el algoritmo DQN, pueden ser una alternativa viable a los modelos MIP en la resolución de problemas de optimización en la administración forestal. Se espera que los hallazgos de este estudio contribuyan al desarrollo de nuevas metodologías y enfoques en la gestión de los recursos forestales y proporcionen una base para futuras investigaciones en el campo.

Dedicado a,
mi.

Como muestra de dónde me puede llevar cada acción

Agradecimientos

Siempre hay que agradecer a quienes te apoyan y quienes están disponibles a contribuir contigo. A mi me apoya mi familia, amada, amigos y los profesionales de la universidad que me dieron el espacio y las herramientas para realizar este trabajo. Hay tantos nombres que se repiten, pero si lees el tuyo, siéntete identificado en este agradecimiento por tu apoyo y simpatía conmigo:

Marcela, Malcom, Rafael, Emilia, Peka, **Titina**, Dona, Ursus, Paula, Ramón, Purísima, Claudia, Matilde, Patricio, Cristian, Margarita, Claudio, Francisco, Marina, Martín, Catalina, Francisca, Magdalena, Maggaly, Patricio, Maritza, Carlos, César, Emiliano Luis Alberto, Matías, Fran, Juan Pablo, Camila, Dania, Álvaro, Pancho, Dani, Liss, Diego, Rodrigo, Alonso, Stefan, Gonzalo, Nicolás, Víctor, Tatiana, Lucas, Andrés, David, Jaime, Juan, Fernando.

Muchas gracias

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

Los bosques son recursos naturales críticos que requieren una gestión eficiente y sostenible. La sociedad espera que a través de la industria maderera las administradoras forestales sean capaces de cubrir las actuales necesidades relacionadas con servicios forestales y recursos forestales sustentables para que las futuras generaciones puedan disfrutar de los mismos beneficios forestales que coexisten con la sociedad de hoy en día.[1]

La industria forestal es importante desde la perspectiva regional y nacional en varios países dado que proporcionan una amplia variedad de productos tales como papel, envoltorios, materiales de construcción y muebles y también proporcionan servicios tales como recreación, hábitat de vida silvestre, agua limpia, amenidades escénicas y almacenamiento de carbono [2]. Constituye una proporción significativa de las exportaciones netas, por ejemplo, en Chile. Según el Anuario INFOR 2022, la participación del sector forestal en el monto total exportado a nivel nacional disminuyó por tercer año consecutivo, llegando a un 6,3 %, después de haber representado el 9,1 % el 2018 y 8,2 % el 2019 [3].

Año/ *Year*	Total/ *Total*	Minería/ *Mining*	Agropecuario, Silvícola y Pesquero/ *Agriculture, Forestry and Fishing*	Industria/ *Industry*	Participación del sector forestal/ *Share or forestry sector* (1)
	Millones de US$ FOB/ *million of US$ FOB*				
2007	68.716	42.387	3.287	23.042	7,2%
2008	64.695	34.228	4.066	26.402	8,4%
2009	55.624	31.820	3.667	20.137	7,5%
2010	71.422	44.477	4.371	22.575	6,9%
2011	81.857	49.038	4.969	27.851	7,2%
2012	78.284	46.212	5.019	27.052	6,9%
2013	77.070	43.646	5.647	27.777	7,4%
2014	75.324	40.377	5.621	29.327	8,1%
2015	62.120	32.309	5.194	24.617	8,8%
2016	60.769	30.698	5.883	24.189	8,7%
2017	68.904	37.135	5.743	26.027	7,8%
2018	74.838	39.148	6.487	29.204	9,1%
2019	68.792	35.377	6.798	26.618	8,2%
2020	74.086	42.485	6.411	25.189	6,7%
2021	94.677	58.630	6.640	29.407	6,3%

Figura 1.1: Exportacion chilena de bienes por sector económico y porcentaje de participación del sector forestal chileno en el monto total exportado.[3]

Sin embargo, esta disminución responde al notorio aumento del sector minero desde 2020 respecto del total de exportaciones antes que a una dismución significativa en la producción

de exportación del sector forestal. Además, Chile exporta el 7 % del total mundial de pulpa para papel, posicionándose en uno de los principales lugar, luego de Brasil (24 %), Canadá (15 %), y Estados Unidos (11 %) [4].

1.1. Investigación de operaciones en planificación de industria forestal

Dada su envergadura, el sector forestal chileno requiere funcionar sujeto a bajos niveles de riesgo y en la vía de administración sustentable. En la Figura 1.2 se aprecia como la administración forestal sustentable surge como demanda social para preservar un equilibrio entre la estraccion de madera para fines comerciales y el balance de recursos naturales de las plantaciones forestales. Para cumplir con este funcionamiento, las administradoras forestales realizan una planificación óptima que garantiza el uso adecuado de los recursos en el tiempo. Una gestión forestal inadecuada puede conducir a la deforestación, la degradación de la biodiversidad [5], los desastres naturales como deslizamientos de tierra e inundaciones [6] y erosión del suelo [7].

La planificación involucra problemáticas de uso de tierra, regeneración de la plantación, construcción y mejora de caminos, cosecha, transporte de producto primario, entre otras. En cada problemática se debe prestar cuidadosa atención tanto a las cuestiones sociales y ambientales, como a las organizaciones forestales relacionadas con la planificación, que tienen restricciones y objetivos específicos distintos entre sí.

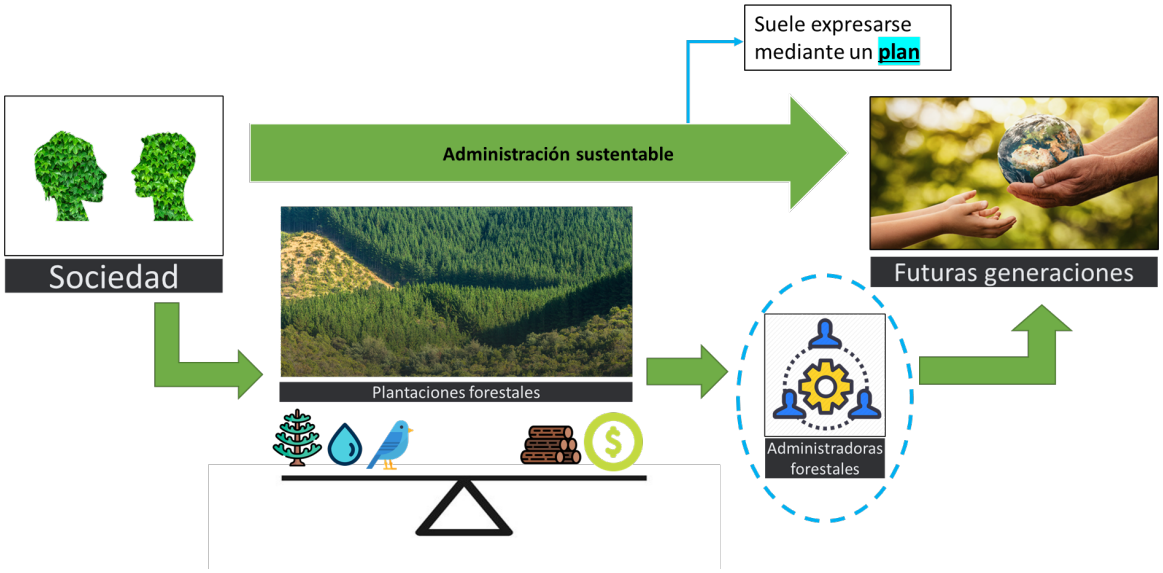

Figura 1.2: Planificación forestal como expresión de demanda social por administración sustentable de plantaciones forestales.

Desde la década de los 60, la comunidad forestal ha incorporado herramientas de investigación operativa (IO) para respaldar su planificación. La introducción de modelos de programación lineal (LP) para apoyar la toma de decisiones de cosecha y regeneración a largo plazo, se ha aplicado en una amplia gama de problemas de toma de decisiones, que abarcan a la planificación estratégica, táctica, y operacional.[2]

La planificación táctica sirve como puente entre el nivel estratégico y el nivel operativo y debe garantizar que las decisiones operativas posteriores no sean sub optimizadas debido a un horizonte de planificación más corto, y sigan la dirección que ha sido establecida en el plan estratégico. [8]

Este trabajo de memoria se enmarca en el ámbito de gestión forestal, y se enfoca en el problema táctico de **planificación de cosecha** y **construcción de caminos**. Este problema consiste, básicamente, en un planificador que tiene bosques destinados a explotación, los cuales son divididos en unidades de cosecha denominadas "rodales"[1]. En cada periodo de tiempo, el planificador debe decidir qué rodales se cortarán y qué caminos se construirán para llevar a cabo esta explotación en orden de maximizar la utilidad esperada neta.[9]

1.2. Deep reinforcement learning en problemas de alta complejidad

Los modelos de programación entera mixta (MIP) han sido ampliamente utilizados para abordar este problema; sin embargo, pueden enfrentar dificultades para encontrar soluciones óptimas en problemas de gran escala debido a la explosión del espacio de soluciones y los tiempos de cómputo requeridos debido a que la escala y la complejidad del problema aumentan con el tamaño del bosque y el número de rodales y caminos a considerar, convirtiendose en un desafío significativo debido a la gran cantidad de variables y restricciones involucradas, como las condiciones del terreno, las regulaciones ambientales y las fluctuaciones en la demanda de madera[10].

Por otro lado, el aprendizaje por refuerzo profundo (DRL) ofrece un enfoque alternativo para abordar problemas de optimización complejos y de gran escala [11]. Aunque los algoritmos de DRL han demostrado ser prometedores en una variedad de aplicaciones, su aplicación en la administración forestal sigue siendo relativamente nueva y requiere una investigación exhaustiva. Además, los algoritmos de DRL pueden enfrentar desafíos en términos de estabilidad y convergencia, especialmente en entornos con estados y acciones de alta dimensión[12].

Este trabajo de memoria aborda estos desafíos utilizando métodos de optimización, con un enfoque específico en MIP y DRL para mejorar la toma de decisiones en la administración forestal. En específico, el caso de estudio es el predio Los Copihues tal como en [13].

1.3. Objetivos general y específicos de la investigación

El propósito de esta investigación es desarrollar y comparar modelos matemáticos de programación entera mixta (MIP) y agentes de aprendizaje por refuerzo profundo (DRL) para la optimización de la administración forestal, específicamente en la toma de decisiones sobre la cosecha de rodales y la construcción de caminos en un bosque.

[1] Agrupación de árboles que ocupando una superficie de terrenos determinada, es suficientemente uniforme en especies, edad, calidad o estado, lo cual permite distinguirlo del arbolado contiguo

1.3.1. Objetivo general

Desarrollar y evaluar un modelo MIP específico y agentes de DRL en la resolución del problema de optimización forestal de cosecha y construcción de caminos del predio Los Copihues considerando la maximización de las utilidades de los administradores forestales y la satisfacción de las demandas por volumen de madera como objetivo de optimización.

1.3.2. Objetivos específicos

1. Formular un modelo MIP que permita determinar el calendario óptimo de cosechas y construcción de caminos en un bosque, considerando las restricciones y objetivos de la administración forestal.

2. Diseñar entorno personalizado que ejecute la transición entre estados del bosque e interactúen con los agentes de DRL entregandole señales de recompensa y nuevo estado.

3. Implementar y entrenar agentes de DRL en entorno personalizado.

4. Comparar el desempeño de los agentes de DRL con la solución óptima obtenida a través del modelo MIP respecto a la secuencia de decisiones de construcción de caminos y cosecha de rodales, y por ende comparar las recompensas acumuladas obtenidas por cada modelo.

Al lograr estos objetivos, se espera obtener una comprensión más profunda de las ventajas y desventajas de cada enfoque, así como identificar oportunidades para futuras investigaciones en el campo.

Capítulo 2

Marco teórico

Este capítulo proporciona los fundamentos conceptuales y teóricos que respaldan el estudio de la optimización forestal utilizando deep reinforcement learning y programación entera mixta.

2.1. Administración forestal

La administración forestal es el proceso de planificación y toma de decisiones para alcanzar una gestión sostenible y eficiente de los recursos forestales. La maximización de las utilidades, la conservación de la biodiversidad y el cumplimiento de las demandas de madera son algunos de los objetivos clave en la administración forestal [10].

2.1.1. Cadena de producción

En la Figura 2.1 se resume visualmente la cadena de producción forestal chilena propuesta en [14]. El análisis simplificado de la cadena de producción forestal propone tres etapas:

- Etapa **bosque**, como el lugar donde se toman decisiones estratégicas que abarcan el ciclo de vida del bosque, desde la plantación hasta la cosecha. Se planifica por periodos, el destino de la madera cosechada, la cual puede ser almacenada en las áreas de cosecha, transportada a aserraderos o plantas de producción de pulpa y electricidad. O bien, la madera puede ser vendida directamente desde el bosque.

- Etapa **plantas**, como aserraderos, plantas de pulpa, generadoras de electricidad y papeleras, donde los productos pueden permanecer en la empresa, como venta interna o ser distribuidos a otros actores de la industria, como venta externa.

- Finalmente etapa **distribución**, donde los productos son distribuidos tanto al retail como a los clientes externos a la (o las) empresas estudiadas en la cadena de valor. También asume la producción de Pulpa, Aserrables y Electricidad como productos finales de la cadena de producción, sin distinguir clientes internos de externos.

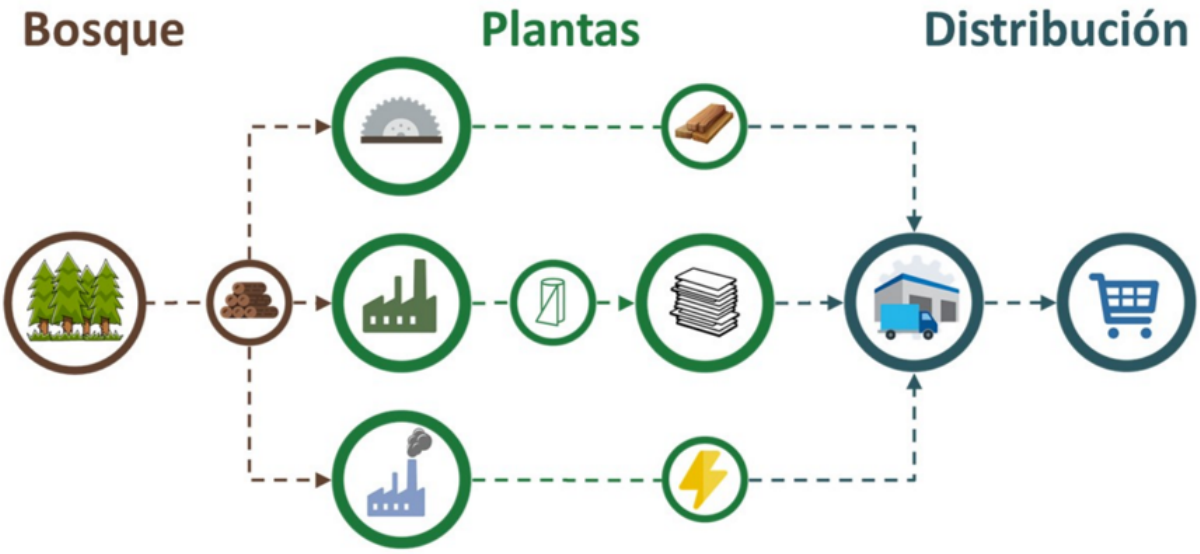

Figura 2.1: Cadena de valor simplificada de producción forestal en Chile[14]

2.1.1.1. Cadena de producción - Definiciones

- Rodal: Unidad de producción del bosque, cada rodal representa un área, en la cual se obtiene madera en trozos, la cual se mide según su volumen estimado.
- Trozos: Distintos tipos de troncos obtenidos al cosechar un rodal. En este trabajo se usa el término trozo y tronco para referir a este concepto. Los trozos son identificados según su destino.
- Co-productos: Se denota como co-producto al volumen de desecho en un aserradero o planta, que es destinado a ser insumo en otra planta, como astillas y aserrín.
- Aserrable: refiere tanto al tipo de trozo destinado a ser procesado en aserraderos, como a los productos obtenidos de ellos.
- Pulpable: refiere tanto al tipo de trozo como co-productos aserrables destinados a ser insumos en la producción de celulosa para Pulpa.

2.1.2. Cadena de producción verticalmente integrada

Cuando una empresa se encuentra en una situación de integración vertical completa, la controladora tiene la facilidad para coordinar cada una de las componentes de su cadena de producción. En Chile, las dos empresas con mayor participación en el mercado forestal (Arauco y CMPC) se encuentran integradas verticalmente, sin embargo, históricamente la planificación ha seguido una estrategia basada en la disponibilidad de insumos, en la cual, la planificación de operaciones en el bosque es el motor de la planificación industrial, y por lo tanto, lo que dicta las decisiones tácticas de toda la cadena [15]. En [15] demuestran que una estrategia que integra las decisiones del bosque y de la producción industrial en el mismo problema de optimización puede aumentar el valor presente neto de toda la cadena hasta en un 5 %.

2.1.2.1. Cadena de producción verticalmente integrada - Definiciones

Para una cadena de producción verticalmente integrada se toman en cuenta los siguientes elementos en la operación del bosque:

- Vivero Forestal: se encarga principalmente del proceso de multiplicación de las plantas. Su ubicación es cercana a las áreas a reforestar.
- Cosecha: cada rodal tiene una edad mínima de cosecha (dado por la madurez de la especie plantada, según el volumen de madera que puede entregar por unidad de área), por ejemplo, la especie pinus radiata puede ser cosechada cuando el rodal cumple 20 años.
- Caminos: la construcción de caminos es parte importante de la planificación forestal. Se modelan como una red de caminos existentes anteriores al inicio del horizonte de planificación y otros caminos potenciales que pueden ser construidos como parte de la operación forestal.

En tanto los elementos a tomar en cuenta en la operación de las plantas:

- Horizonte de planificación: cantidad de periodos en que se divide la planificación.
- Parámetros del bosque: El problema toma en cuenta un tipo de árbol (especie), los cuales son modelados según volumen anual por hectárea. De esta forma cada rodal queda definido por el volumen de madera esperado en cada periodo.
- Parámetros de transporte: El problema modela el transporte mediante restricciones de flujo entre cada rodal y aserraderos o plantas. Mientras el coste de transporte es calculado como un coste promedio unitario de transporte multiplicado la distancia y volumen transportado en cada periodo. De la misma forma, cada volumen de co-producto transportado entre aserraderos y plantas es multiplicado por el coste unitario de transportar 1 m^3 por kilómetro. Estos costes dependen del tipo de tronco o co-producto transportado.
- Ingresos: La venta de productos es calculada mediante el volumen total obtenido multiplicado por el precio en cada periodo. Además, es considerado como ingreso el valor potencial final del bosque, asignando un valor a cada m3 de bosque en el final del horizonte. Estos valores son modelados como parte de la función objetivo, junto a los costes antes mencionados.

2.2. Programación entera mixta

La programación entera mixta es una técnica de optimización matemática que combina variables de decisión enteras y continuas con restricciones lineales. En el ámbito de la administración forestal, los modelos MIP se han utilizado ampliamente para optimizar la programación de cosechas y construcción de caminos.

$$\begin{aligned} \max \quad & c^T x + d^T y \\ \text{s.a} \quad & Ax + By \leq b \\ & x \geq 0, \quad y \in \{0,1\}^n \end{aligned}$$

Donde:

- x y y son vectores de variables continuas y binarias, respectivamente.
- c,d, y b son vectores de coeficientes.
- A y B son matrices de coeficientes.

.

2.3. Procesos de Decisión de Markov (MDP)

Un Proceso de Decisión de Markov (MDP) es una tupla $< \mathcal{S}, \mathcal{A}, \mathcal{P}^a_{s,s'}, \mathcal{R}^a_{s,s'}, \gamma >$. Un MDP se define por un conjunto de estados $\mathcal{S}$, un conjunto de acciones $\mathcal{A}$, una función de transición $\mathcal{P}$, una función de recompensa $\mathcal{R}$ y un factor de descuento γ. Se utiliza para modelar situaciones en las que un agente toma decisiones secuenciales con el objetivo de maximizar la recompensa acumulada a lo largo del tiempo.

2.3.1. Modelo

- **Espacio de estados** $\mathcal{S}$: conjunto que representa todas las configuraciones posibles del sistema.
- **Espacio de acciones** $\mathcal{A}$: conjunto que representa todas las acciones posibles que el agente puede tomar.
- **Función de transición** $\mathcal{P}^a_{s,s'}$: probabilidad de ir al estado s' dado que el sistema está en el estado s y el agente toma la acción a.

  Los MDPs, como el nombre lo indica, son procesos markovianos; es decir, que el siguiente estado es independiente de los estados anteriores dado el estado actual:

$$\mathcal{P}^a_{s,s'} = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$$

- **Función de recompensa** $\mathcal{R}^a_{s,s'}$: asigna un valor numérico a la transición entre estados bajo ciertas acciones.

$$\mathcal{R}^a_{s,s'} = \text{recompensa al pasar del estado } s \text{ al estado } s' \text{ tras realizar la acción } a$$

- **Factor de descuento** γ: Un valor entre 0 y 1 que indica cuánto valora el agente las recompensas futuras respecto a las inmediatas.
- **Función de término** $IsEnd(s)$: función binaria que indica si el estado s es terminal o no.

2.3.2. Política

En los MDPs, la política se denota como $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. La función π asigna una probabilidad a cada acción $a \in \mathcal{A}$ para un estado dado $s \in \mathcal{S}$. Formalmente es una distribución de probabilidad de acciones dados los estados de la siguiente manera:

$$\pi(a|s) = \mathbb{P}(a_t = a|s_t = s) \quad \text{donde } a \in \mathcal{A} \text{ y } s \in \mathcal{S}$$

Iniciar en un estado s_0 y seguir la política π durante n pasos produce un flujo de estados, acciones y recompensas que finalmente conforman un episodio como:

$$episodio^{\pi} = (s_0; a_1, \mathcal{R}(s_0, a_1, s_1), s_1; a_2, \mathcal{R}(s_1, a_2, s_2), s_2; a_3, \mathcal{R}(s_2, a_3, s_3), s_3; ...; a_n, \mathcal{R}(s_{n-1}, a_n, s_n), s_n)$$

2.3.3. Recompensas acumuladas

Definición 2.1 *Recompensas acumuladas*
"El retorno G_t es la recompensa descontada acumulada total desde el tiempo $t+1$ en adelante. Se define como:"

$$G_t = r_{t+1} + \gamma \cdot r_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

2.3.4. Funciones de valor

Definición 2.2 *Función de valor de estado $v^{\pi}(s)$*
"La función de valor de estado es la esperanza de la recompensa acumulada comenzando desde el estado s para luego continuar con la política π"

$$v_{\pi}(s) = \mathbb{E}[G_t|s_t = s, \pi]$$

Definición 2.3 *Función de valor estado-acción $q^{\pi}(s, a)$*
"La función de valor estado-acción es la expectativa del retorno si se empieza en s, se toma la acción a y luego se sigue la política π"

$$q_{\pi}(s, a) = \mathbb{E}[G_t|s_t = s, a_t = a, \pi]$$

En resumen, las ecuaciones de valor $v_{\pi}(s)$ y $q_{\pi}(s, a)$ son herramientas esenciales en la planificación de acciones, ya que permiten evaluar, comparar y tomar decisiones óptimas cuando las recompensas son inciertas y se toman a lo largo del tiempo. En un MDP se busca resolver los problemas de predicción y control del sistema, que son fundamentales para encontrar políticas que maximicen la recompensa acumulada a largo plazo y, por lo tanto, son vitales para la toma de decisiones secuenciales en entornos dinámicos:

- **Predicción**: dada una política π, se busca determinar v_{π}.
- **Control**: se busca determinar la política óptima π_* y v_*.

2.3.5. Ecuación de expectativa de Bellman

Desde un estado s se busca representar sus funciones de valor considerando todas las posibles acciones a tal como se muestra en la Figura 2.2:

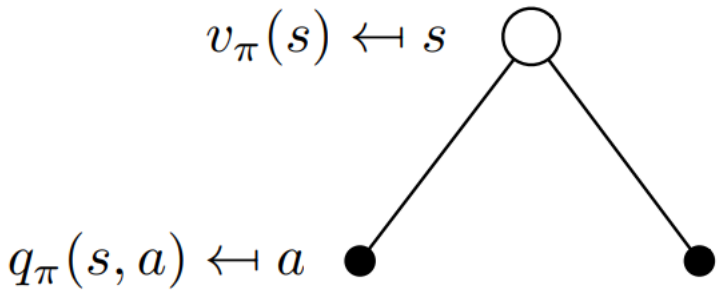

Figura 2.2: Descripción de funciones de valor en la selección de acciones

La ecuación de expectativa de Bellman para $v^\pi(s)$ en términos de $q^\pi(s,a)$:

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a)$$

Desde un estado s se selecciona una acción a y se busca representar sus funciones de valor considerando todos los posibles estados s' a los que se podría llegar como se muestra en la Figura 2.3:

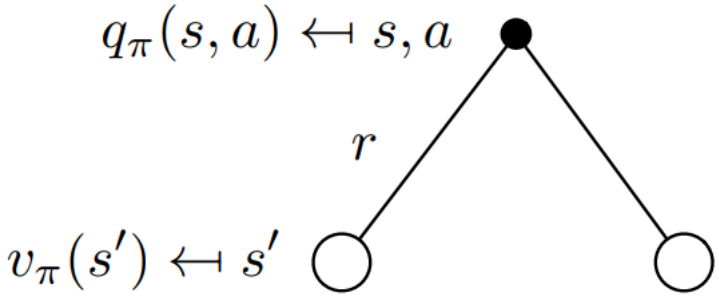

Figura 2.3: Descripción de funciones de valor luego de seleccionar una acción

La ecuación de expectativa de Bellman de $q_\pi(s,a)$ en términos de $v_\pi(s')$ es:

$$q_\pi(s,a) = \mathcal{R}^a_{s,s'} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{s,s'} v_\pi(s')$$

2.3.6. Ecuaciones de optimalidad de Bellman

La forma estándar del operador de Bellman implica la maximización sobre acciones:

$$v_\pi(s) = \sum_a \pi(a|s) \left[\mathcal{R}^a_{s,s'} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{s,s'} v_\pi(s') \right]$$

Pero, cuando se considera una política fija π, el operador se convierte en un operador de expectativa de Bellman T^π [16] como:

$$(T^\pi v)(s) = \sum_a \pi(a|s) \left[\mathcal{R}^a_{s,s'} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{s,s'} v(s') \right]$$

Teorema 1 (Mapeo Contractivo [16])**.** Para cualquier espacio métrico $\mathcal{V}$ que es completo (i.e. cerrado) bajo un operador $T(v)$, donde T es una contracción-γ:

- T converge a un punto fijo único.
- Posee una tasa de convergencia lineal de γ.

En Anexo A y Anexo B puede encontrarse las definiciones completas del operador de Bellman.

El hecho clave aquí es que el operador de expectativa de Bellman es una contracción en este espacio con la norma infinito. Dado que el espacio de funciones de valor (bajo la norma infinito) es completo, la aplicación iterativa del operador de expectativa de Bellman a cualquier función de valor inicial garantiza la convergencia a una función de valor única.

Por lo tanto, aunque no se dice comúnmente que el operador de expectativa de Bellman es "cerrado", el espacio en el que opera es completo. Y es esta completitud, combinada con la propiedad contractiva del operador, lo que garantiza la convergencia a una solución única.

Por consecuencia, para cualquier MDP se cumple que:

1. Existe una política óptima π_* que es mejor o igual que el resto de políticas, $\pi_* \geq \pi, \forall \pi$
2. Todas las políticas óptimas alcanzan la función óptima $v_{\pi_*}(s) = v_*(s)$.
3. Todas las políticas óptimas alcanzan la función de valor óptima $q_{\pi_*}(s,a) = q_*(s,a)$.

Una política óptima es aquella que, para cualquier estado dado, selecciona la acción que maximiza el valor esperado:

$$\pi_*(a|s) = \begin{cases} 1, & \text{si } a = \arg\max_{a \in \mathcal{A}} q_*(s,a) \\ 0, & \text{en otro caso} \end{cases}$$

Definición 2.4 *Función de valor de estado óptima v_**
"La función de valor óptima de estado corresponde al máximo valor de estado $v_\pi(s)$ entre todas las políticas"

$$v_*(s) = \max_\pi v_\pi(s)$$

Definición 2.5 *Función de valor estado-acción óptima q_**
"La función de valor estado-acción corresponde al máximo valor de estado-acción $q_\pi(s,a)$

entre todas las políticas"

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

Un MDP se considera resuelto una vez que se obtiene el valor de las funciones de valor óptimas.

Conociendo la ecuación de expectativa de Bellman de v_π en función de q_π, es directo que $v_*(s) = \max_a q_*(s,a)$. Por lo tanto la ecuación de optimalidad de Bellman para q_* es:

$$q_*(s,a) = \mathcal{R}^a_{s,s'} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{s,s'} \max_{a'} q_*(s',a')$$

Dada su naturaleza no lineal y la falta de soluciones cerradas en general, se pueden aplicar métodos iterativos para resolverla:

- **Iteración de valor**
- **Iteración de política**
- **Q-learning**

2.4. Programación dinámica

La programación dinámica es un enfoque matemático utilizado para resolver problemas de optimización descomponiéndolos en subproblemas más simples. En el contexto del aprendizaje por refuerzo, se utiliza para calcular funciones de valor y encontrar políticas óptimas. El término "dinámico" refiere a la componente de secuencialidad temporal del problema, en tanto "programación" hace referencia a optimizar un comportamiento, es decir, optimizar una política [16].

Programación dinámica tiene 2 propiedades:

1. Subestructura óptima: solución óptima puede descomponerse en subproblemas.
2. Subproblemas solapados
    - subproblemas ocurren varias veces
    - soluciones son almacenadas y usadas en cada iteración

Los MDP satisfacen ambas propiedades que solicita la programación dinámica:

1. Ecuación de Bellman entrega una descomposición recursiva del problema: la mayor recompensa por presente ($\mathcal{R}^a_{s,s'}$) y la mayor recompensa para el resto de la trayectoria ($\gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{s,s'} \max_{a'} q_*(s',a')$).
2. Las funciones de valor q_π, v_π almacenan información y la reusan en cada iteración.

Adicionalmente, la programación dinámica asume conocimiento completo del MDP, vale decir $\mathcal{P}^a_{s,s'}$ y $\mathcal{R}^a_{s,s'}$.

2.4.1. Evaluación de la Política

La evaluación de la política se refiere al proceso de calcular la función de valor v_π para una política dada π. Iterativamente se aplica la ecuación de expectación de Bellman para evaluar la política:

$$v_{k+1}(s) = \sum_a \pi(a|s) \left(\mathcal{R}^a_{s,s'} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{s,s'} v_k(s') \right)$$

Este proceso se repite hasta que la función de valor converja, es decir, hasta que $|v_{k+1}(s) - v_k(s)| < \theta$ para un pequeño θ.

2.4.2. Mejora de la política

Una vez que se evalúa una política y se obtiene su función de valor, puede mejorarse usando un política *greedy* π' respecto a esta función de valor (eligiendo las acciones que llevan a estados con mayores recompensas acumuladas). La idea es actualizar la política basándose en el valor de las acciones en cada estado:

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} q_\pi(s, a)$$

Esto mejora el valor desde cualquier estado s en un paso:

$$q_\pi(s, \pi'(s)) = \max_{a \in \mathcal{A}} q_\pi(s, a) \geq q_\pi(s, \pi(s)) = v_\pi(s)$$

Por lo tanto, utilizar la política π' garantiza que la función de valor será igual o mejor que la función de valor de otra política π:

$$\begin{aligned}
v_\pi(s) &\leq q_\pi(s, \pi'(s)) \\
&\leq \mathbb{E}\left[r_{t+1} + \gamma v_\pi(s_{t+1}) \mid s_t = s \, , \pi' \right] \\
&\leq \mathbb{E}\left[r_{t+1} + \gamma q_\pi(s_{t+1}, \pi'(s_{t+1})) \mid s_t = s \right] \\
&\leq \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 q_\pi(s_{t+2}, \pi'(s_{t+2})) \mid s_t = s \right] \\
&\vdots \\
&= \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \mid s_t = s \right] = v_{\pi'}(s)
\end{aligned}$$

En el momento que la política π deje de mejorar ocurrirá:

$$q_\pi(s, \pi'(s)) = \max_{a \in \mathcal{A}} q_\pi(s, a) = q_\pi(s, \pi(s)) = v_\pi(s)$$

Lo cual significa que ahora π es la política óptima, ya que se satisface la ecuación de optimalidad de Bellman:

$$v_\pi(s) = \max_{a \in \mathcal{A}} q_\pi(s, a) \rightarrow v_\pi(s) = v_*(s), \forall s \in \mathcal{S} \rightarrow \pi = \pi^*$$

2.4.3. Iteración de la política

La iteración de la política combina la evaluación y la mejora de la política, alternándolas hasta obtener una política óptima:

1. Inicializar una política π de manera arbitraria.
2. Evaluar la política.
3. Mejorar la política basándose en la función de valor obtenida.
4. Repetir los pasos 2 y 3 hasta que la política converja y $\pi' = \pi$.

2.4.4. Iteración de Valor

La iteración de valor es una optimización sobre la iteración de la política con el objetivo de encontrar una política óptima de un MDP. En lugar de esperar a que la función de valor converja completamente en la evaluación de la política, la iteración de valor combina la evaluación y la mejora en cada paso:

$$v_{k+1}(s) = \max_{a \in \mathcal{A}} \left(\mathcal{R}^a_{s,s'} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{s,s'} v_k(s') \right)$$

Este proceso se repite hasta su finalización, donde la función de valor converge. Esta iteración de valor se mueve en el espacio de funciones de valor, lo cual implica que los valores de la función de valor en etapas intermedias podrían no corresponder a ninguna política en particular, ya que son una construcción ideada para llegar a la política óptima.

2.5. Reinforcement learning

El aprendizaje por refuerzo (RL, por sus siglas en inglés) representa una formulación rigurosa para abordar problemas de toma de decisiones en entornos que pueden ser parcial o completamente desconocidos. A diferencia de los paradigmas de aprendizaje supervisado y no supervisado, RL opera en un marco en el cual un agente aprende mediante la interacción con su entorno sin necesitar un conjunto de datos etiquetado. En este contexto, un agente es la entidad computacional encargada de interactuar con el entorno para aprender la política óptima. El agente toma decisiones basadas en la observación del estado del entorno y su política de toma de acciones π y recibe recompensas como retroalimentación como lo muestra la Figura 2.2.

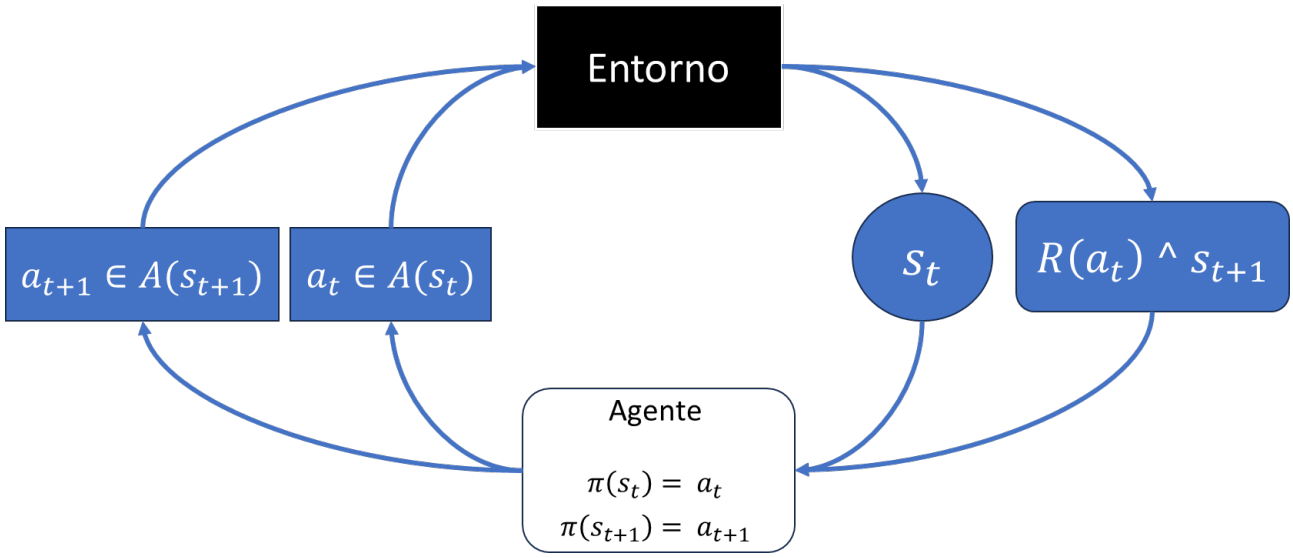

Figura 2.4: Esquema de flujo estado - acción - recompensa - nuevo estado

RL está basado en la `hipótesis de recompensa`.

Definición 2.6 *Hipótesis de recompensa*
"Todo objetivo/meta puede describirse por la maximización de la esperanza de recompensa acumulada"

Un agente de RL opera mediante un conjunto de principios matemáticos y computacionales que se describen a continuación.

- **Marco teórico de procesos de decisión de Markov (MDP)**: un MDP completamente conocido proporciona una descripción precisa de la dinámica del sistema mediante funciones de transición de probabilidad y funciones de recompensa. Resolver el MDP implicaría encontrar la política óptima π_* que maximiza la recompensa esperada acumulada.

- **Desconocimiento del Sistema**: en el caso de RL, el agente no tiene acceso a las funciones $\mathcal{P}$ ni $\mathcal{R}$. Este escenario representa un MDP bajo incertidumbre, donde el agente tiene que aprender la política óptima interactuando con el entorno.

- **Temporalidad y secuencialidad**: en RL, la toma de decisiones se efectúa a lo largo de una serie de pasos temporales discretos. En cada paso de tiempo t, el agente observa el estado s_t, ejecuta una acción a_t, recibe una recompensa r_t y el sistema transita al estado s_{t+1}. La secuencia de interacciones se denota generalmente como $[s_0, a_0, r_1, s_1, a_1, r_2, \ldots]$.

- **Política de decisiones y maximización de recompensa acumulada**: el objetivo final del agente en RL es aprender una política de decisiones, denotada generalmente como $\pi(s)$, que maximice alguna medida de recompensa acumulada a lo largo del tiempo. Esto se suele formalizar mediante la función de valor $v_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, \pi\right]$, donde γ es el factor de descuento [17].

En el campo del RL, el objetivo es encontrar una política óptima que maximice la recompensa acumulada esperada en un MDP. Hay diversas formas de abordar este problema. Los métodos de Monte Carlo que emplean muestreo aleatorio para estimar funciones de valor y

actualizar políticas, por ejemplo, pueden ser basados en modelo o libres de modelo. En la versión basada en modelo, se necesita un modelo explícito del entorno para realizar simulaciones, mientras que en la versión libre de modelo, las estimaciones se realizan directamente a partir de la experiencia.

En el contexto del aprendizaje por refuerzo, los términos "on-policy" y "off-policy" refieren a diferentes enfoques para la actualización de la política y/o función de valor de un agente. En métodos on-policy, el agente aprende sobre la política que está siguiendo actualmente, para aprender, el agente usa la experiencia generada por su política actual, lo cual implica que la política que se está evaluando y mejorando es la política que el agente está utilizando para interactuar con el entorno.Un ejemplo clásico de un método on-policy es SARSA. Por otro lado, en métodos off-policy el agente puede aprender sobre una política diferente a la que está siguiendo para explorar. Esto implica que el agente puede aprender sobre la política óptima mientras sigue una política exploratoria, permitiendo el uso de experiencias pasadas o experiencias generadas por otras políticas para el aprendizaje. Un ejemplo clásico de un método off-policy es Q-Learning.

2.5.1. Métodos libres de modelo

Como lo indica su nombre, este método estima el valor de $v_\pi(s)$ directamente de la trayectoria generada por π en cada episodio, sin necesidad de funciones de transición y recompensa del MDP.

2.5.1.1. Método Monte Carlo libre de modelo

$v^\pi(s)$ es el valor esperado por estar en el estado luego continuar escogiendo acciones según la política π. Un buen estimador de $v^\pi(s)$ es G_t:

$$\widehat{v_\pi(s)} = G_t \text{ , donde } s_{t-1} = s$$

Para plantear la interacción del método Monte Carlo con el entorno, se define $N(s)$ como el número de veces que se visita el estado s para implementar "averaging with a step-size" o "incremental averaging". La idea es asegurarse de que, a medida que se gana más experiencia (es decir, se visita el par estado-acción más veces), el impacto de una única observación se reduce. En otras palabras, al principio, con poca experiencia, se confía en gran medida en las nuevas observaciones. Pero a medida que se gana experiencia, las actualizaciones se vuelven más conservadoras.

Un planteamiento análogo a calcular un promedio es una combinación convexa que actualiza el antiguo valor de $\widehat{v_\pi(s)}$ con cada transición usando "incremental averaging":

$$\eta = \tfrac{1}{N(s)}$$

$$\widehat{v_\pi(s)} \leftarrow (1-\eta) \cdot \widehat{v_\pi(s)} + \eta \cdot G_t$$

Algebraicamente, la última formulación equivale a:

$$\widehat{v_\pi(s)} \leftarrow \widehat{v_\pi(s)} + \eta \cdot (G_t - \widehat{v_\pi(s)})$$

Esta última formulación indica que la aproximación se realiza calculando el error de la estimación y luego moviendose en la dirección de ese error. Si la estimación se aproxima al valor real de recompensa entonces se converge al valor real de $v_\pi(s)$.

2.5.1.2. Métodos de diferencia temporal

Los métodos de diferencia temporal [16] son libres de modelo. La temporalidad está representada por la cantidad de pasos que se utilizan para estimar la función de valor. El ejemplo básico es una aproximación de dos pasos como en las ecuaciones de Bellman. Se utiliza una ecuación similar a la ecuación derivada en los métodos de Monte Carlo libres de modelo:

$$\widehat{v_\pi(s_t)} \leftarrow \widehat{v_\pi(s_t)} + \alpha \cdot (r_{t+1} + \gamma \widehat{v_\pi(s_{t+1})} - \widehat{v_\pi(s_t)})$$

Se definen:

- TD target : $r_{t+1} + \gamma \widehat{v_{\pi(s_{t+1})}}$
- TD error : $\delta_t = (r_{t+1} + \gamma \widehat{v_\pi(s_{t+1})} - \widehat{v_\pi(s_t)})$

Esta aproximación utiliza bootstraping para estimar el valor de un estado como la recompensa inmediata por estar un estado más la función de valor del estado siguiente descontada por γ, obteniendo un sesgo en la estimación a cambio de un **aprendizaje intermedio** con cada paso que se realice, sin tener que esperar el final del episodio para estimar como en el caso de Monte Carlo.

α es una tasa de aprendizaje constante que determina en qué medida una nueva estimación afecta a la anterior. Dado que TD realiza actualizaciones basándose en estimaciones parciales (y no en recompensas de episodios completos como MC), es útil tener un factor de aprendizaje constante para mezclar la nueva información con la vieja, en lugar de promediarla directamente. Entonces mientras que MC utiliza $\frac{1}{N}$ para calcular el promedio verdadero de las recompensas retornadas para un estado a lo largo del tiempo, los métodos de diferencia temporal utilizan α para mezclar efectivamente la información nueva y vieja de las estimaciones parciales. La elección de α a menudo se basa en la experiencia y puede ser ajustada para optimizar el aprendizaje.

2.5.1.2.1. Q-learning

La función $q(s, a)$ proporciona directamente información sobre la calidad de acciones específicas en estados específicos, lo que la hace especialmente útil para algoritmos como el Q-learning.

Q-learning es un algoritmo de RL que fue introducido por Chris Watkins en su tesis doctoral en 1989 [18].Es un algoritmo de RL que se utiliza para encontrar una política óptima en un MDP.

El algoritmo básicamente sigue los siguientes pasos:

1. Inicializa la función $q(s, a)$ de manera arbitraria.

2. Repite para cada episodio:

   a) Inicializa el estado inicial s_0.

   b) Repite para cada paso del episodio hasta llegar a un estado terminal:

      i. Escoge una acción a a partir del estado s usando alguna política derivada de q, como la política ε-greedy.

      ii. Realiza la acción a y observa la recompensa r y el nuevo estado s'.

      iii. Actualiza q utilizando la siguiente fórmula de actualización:

$$q(s,a) \leftarrow (1-\alpha)q(s,a) + \alpha\left(r + \gamma \max_{a'} q(s',a')\right)$$

$$v_*(s') = \max_{a'} q(s',a')$$

Donde:

- α es la tasa de aprendizaje.
- r es la recompensa recibida.
- γ es el factor de descuento.
- $\max_{a'} q(s',a')$ es el máximo Q-valor para el nuevo estado s'

Si cada par (s,a) es visitado infinitamente y ciertas condiciones técnicas se cumplen (como tasas de aprendizaje decrecientes de forma adecuada), entonces **Q-Learning converge a $q_*(s,a)$, la función óptima de acción-valor, independientemente de la política seguida**.

El uso de Q-learning implica:

- **Sin modelo**: Q-Learning puede operar en un entorno de caja negra, sin necesidad de modelar la dinámica del entorno.
- **Off-Policy**: es un método "off-policy", lo que significa que puede aprender la política óptima mientras sigue otra política. Esto es especialmente útil para la exploración eficaz del espacio de estado-acción.
- **Convergencia**: la función q converge a q_*, la cual se puede utilizar para derivar la política óptima π_*.
- **Aproximación funcional**: en espacios de estado y/o acción muy grandes, se pueden utilizar aproximadores funcionales como las redes neuronales para estimar $q(s,a)$.

En Q-learning, la información sobre la calidad o utilidad de tomar ciertas acciones desde ciertos estados se almacena en una tabla de valor-acción, comúnmente conocida como Q-table. La Q-table es típicamente una matriz donde las filas representan todos los estados posibles y las columnas representan todas las acciones posibles. El valor en la celda correspondiente al estado $q(s,a)$, se actualiza en función de la recompensa recibida y las estimaciones de la función de valor estado-acción para el nuevo estado.

2.6. Política de toma de decisiones ϵ-greedy

En el contexto de RL la toma de decisiones son de importancia para hallar el camino hacia la política óptima. Si bien el método basado en un modelo y Q-learning estiman el comportamiento observado óptimo del sistema, **no se hacen cargo de los estados no explorados ni de las correspondientes recompensas por las acciones que llevan a esos estados no explorados.**

Una de las estrategias más comunes para explorar el espacio de acciones es la política de *epsilon-greedy*. En esta política, un agente toma generalmente la acción que tiene el máximo valor esperado de recompensa, con una probabilidad de $1 - \epsilon$. Sin embargo, con una probabilidad de ϵ, el agente elige una acción al azar. Este enfoque asegura un equilibrio entre la *exploración* de nuevas acciones y la *explotación* de las acciones que se sabe son beneficiosas. El valor de ϵ puede ser fijo o disminuir con el tiempo, dependiendo del problema específico y de cómo se desee balancear la exploración y la explotación.

"Exploración" se refiere al comportamiento del agente cuando busca descubrir nuevas estrategias, tomando acciones que no ha evaluado anteriormente y "explotación", por otro lado, se refiere a cuando el agente selecciona la acción que, según su conocimiento actual, tiene la mayor recompensa esperada.

En esta política, el agente toma la acción que considera óptima con probabilidad $1 - \epsilon$ y elige una acción aleatoria con probabilidad ϵ. Considerando $n \in (0, 1)$:

$$\pi(a|s) = \begin{cases} \text{argmax}_{a \in \mathcal{A}} \widehat{q_*(s, a)}, & \text{si } n > 1 - \epsilon, \\ \text{acción aleatoria } a \in \mathcal{A}(s), & \text{si } n \leq \epsilon, \end{cases}$$

$\mathcal{A}(s)$ es el conjunto de acciones que corresponden a las acciones posibles en el estado s.

Si un ambiente tiene un gran número de estados y acciones posibles la exploración se vuelve difícil incluso utilizando una estrategia como ϵ-greedy.

2.7. Aproximación de funciones

En problemas de RL con un gran número de estados y/o acciones, mantener una tabla de valor Q (Q-table) para cada par estado-acción puede ser inviable en términos de memoria y computación. Aquí es donde entra la aproximación de funciones. En lugar de mantener una entrada para cada par estado-acción, se utiliza una función aproximada $q(s, a; w)$ para estimar el valor de una acción a en un estado s como un modelo de regresión lineal.

$$\widehat{q(s, a; w)} = w \cdot \phi(s, a)$$

Esta función está parametrizada por w, que podría ser un vector de pesos en el caso de una red neuronal y ϕ que es un vector de caracteristicas de un par (s, a).

Q-learning puede expresarse como:

$$\widehat{q_\pi(s,a)} \leftarrow \widehat{q_\pi(s,a)} + \eta \cdot (\underbrace{u}_{\text{target}} - \underbrace{\widehat{q_\pi(s,a)}}_{\text{predicción}})$$

En el caso de Q-learning y la política óptima, la aproximación de función se define para cada (s, a, r, s'):

$$w \leftarrow w + \eta \left(\underbrace{(r + \gamma \widehat{v_*(s')}}_{\text{target}} - \underbrace{q(s,a;w))}_{\text{predicción}} \right) \phi(s,a)$$

El aprendizaje de los parámetros w generalmente se realiza mediante métodos de optimización como el descenso de gradiente estocástico (SGD). La idea es minimizar la diferencia entre Q-valor estimado y Q-valor objetivo, que se puede calcular a partir de la recompensa recibida y el Q-valor estimado para el próximo estado en el caso de Q-learning.

La función de pérdida $\mathcal{L}(w)$ implícita puede definirse como:

$$\mathcal{L}(w) = \mathbb{E}\left[\left((r + \gamma \widehat{v_*(s')}) - Q(s,a;w) \right)^2 \right]$$

Para minimizar $\mathcal{L}(w)$, se puede usar descenso de gradiente estocástico (SGD). El gradiente de $\mathcal{L}(w)$ respecto a w se calcula como:

$$\nabla_w \mathcal{L}(w) = 2 \left((r + \gamma \widehat{v_*(s')}) - Q(s,a;w) \right) \phi(s,a)$$

En consecuencia con SGD, la aproximación ya no debe ser respecto a $q(s, a)$ sino a los pesos w:

$$w \leftarrow w - \alpha \nabla_w \mathcal{L}(w)$$
$$= w - 2\alpha \left((r + \gamma \widehat{v_*(s')}) - q(s,a;w) \right) \phi(s,a)$$

La razón por la que hay un signo negativo en la actualización con SGD es porque el objetivo de SGD es minimizar la función de pérdida. Entonces, el movimiento es en la dirección opuesta al gradiente para reducir la pérdida.

Cuando se usan redes neuronales para aproximar la función q, generalmente se permite que la red aprenda una representación interna que es equivalente a aprender una función $\phi(s, a)$ óptima. En el caso de RL, ϕ **no se define explícitamente, sino que se aprende a partir de los datos.**

A partir de esta formulación η deja de tener la interpretación relacionada a la cantidad de veces que se visita un particular (s, a) y pasa a ser la tasa de aprendizaje α con la que se pondera la actualización de w en un SGD.

En RL, especialmente en entornos con un gran número de estados, no siempre es posible visitar cada estado lo suficiente como para aprender un valor q exacto para cada acción en cada estado. Aquí es donde los conceptos de aproximación de funciones y generalización son particularmente útiles. Cuando se usa aproximación de funciones, **implícitamente se generaliza la experiencia de estados y acciones visitados a estados y acciones no visitados**. Dicho de otra manera, si la función de aproximación está bien entrenada, debería

ser capaz de dar estimaciones razonables de $q(s, a)$ incluso para pares que no han visitado. Esto se debe a que los parámetros w se ajustan de forma que minimizan el error entre la función de aproximación $q(s, a; w)$ y los valores de Q que se han observado realmente.

2.8. Deep reinforcement learning

El objetivo del RL es encontrar una política π_* que maximice los retornos esperados totales, es decir, una política óptima. Las definiciones anteriormente explicitadas servirán para construir algoritmos que permitan encontrar políticas que generen soluciones incluso para ambientes con gran dimensión de estados y acciones. Sin embargo, antes de aquello se hablará de una estructura de funciones mediante las cuales se piensa modelar los conceptos. Estas estructuras corresponden a las redes neuronales profundas, las que al integrarse a la teoría de RL dan como resultado el deep reinforcement learning (DRL), que ha sido popular en los últimos años gracias a sus buenos resultados en desafíos de aplicación.
.

2.8.1. Deep learning

La inteligencia artificial (IA) ha experimentado un crecimiento exponencial en los últimos años, impulsada por avances en áreas como el aprendizaje automático (ML) [19]. Dentro de ML, el aprendizaje profundo (DL) ha ganado prominencia por su capacidad para abordar tareas complejas en campos como el procesamiento del lenguaje natural, la visión por computadora y el reconocimiento de voz [20].

El núcleo del aprendizaje profundo es la red neuronal artificial, que toma inspiración de la estructura del cerebro humano. Matemáticamente, una red neuronal se define como una composición de funciones de activación f y parámetros θ (pesos y sesgos):

$$y = f(x; \theta)$$

A través de un algoritmo llamado retropropagación [21], la red ajusta θ para minimizar una función de pérdida $\mathcal{L}$:

$$\theta^* = \arg\min_{\theta} \mathcal{L}(y, f(x; \theta))$$

Los orígenes de DL se remontan a los modelos neuronales propuestos por McCulloch y Pitts en la década de 1940 [22]. Sin embargo, no fue hasta la era de las unidades de procesamiento gráfico (GPUs) y grandes conjuntos de datos cuando el entrenamiento de modelos de redes neuronales profundas se hizo factible [23].

En el panorama actual, el DL supera regularmente a métodos más tradicionales en tareas como la clasificación de imágenes [23], y se ha expandido para incluir arquitecturas especializadas como las redes neuronales convolucionales (CNNs) para análisis de imágenes [24] y redes neuronales recurrentes (RNNs) para secuencias temporales [25]. La rápida evolución del DL también ha catalizado investigaciones en inteligencia artificial explicativa [26].

2.8.1.1. Redes Neuronales

Las redes neuronales constituyen una clase de algoritmos de aprendizaje profundo diseñados para modelar relaciones complejas en conjuntos de datos [19]. Estos modelos computacionales emulan ciertas características del sistema nervioso humano, especialmente la forma en que las neuronas se interconectan y transmiten señales [22]. Las redes neuronales artificiales se componen de unidades, o "neuronas", dispuestas en distintas capas, trabajando en conjunto para realizar tareas específicas.

Una red neuronal típica consiste en una capa de entrada, múltiples capas ocultas y una capa de salida. Las neuronas de cada capa están conectadas densamente con las de la capa subsiguiente, facilitando el flujo de información [27].

Matemáticamente, una red neuronal realiza una serie de transformaciones lineales seguidas de activaciones no lineales. Para una entrada $x \in \mathbb{R}^n$, una capa $\mathcal{L}$ ejecuta la transformación $\mathcal{L}(x) = \phi(Wx+b)$, donde W y b son la matriz de pesos y el vector de sesgos, respectivamente, y ϕ es la función de activación [19].

El término “profundo” en aprendizaje profundo se refiere a la arquitectura de múltiples capas de la red. Un modelo de red neuronal con N capas se representa como [19]:

$$y = f_\theta(x) = \mathcal{L}_N \circ \cdots \circ \mathcal{L}_1(x),$$

Las redes neuronales pueden evolucionar hacia arquitecturas más avanzadas, como las redes neuronales recurrentes y las redes de atención [28, 29].

2.8.1.2. Funciones de Activación

Las funciones de activación juegan un papel crucial al introducir no linealidades en la red, permitiendo que aprenda patrones más complejos [30].

Las funciones de activación pueden ser similares en que son diferenciables (como Sigmoide y Tanh), pero difieren en aspectos como el rango de sus salidas. Por ejemplo, la función Sigmoide tiene un rango entre 0 y 1, mientras que Tanh varía entre -1 y 1. ReLU, por otro lado, es eficiente computacionalmente pero no es diferenciable en el origen [31]. Las ecuaciones y características de las funciones de activación se resumen en Tabla 2.1.

Tabla 2.1: Comparación de funciones de activación comunes.

Función de Activación	Ecuación	Características
Sigmoide	$f(z) = \frac{1}{1+e^{-z}}$	Acotada, diferenciable
ReLU	$f(z) = \max(0, z)$	No acotada, no diferenciable en 0
Tanh	$f(z) = \tanh(z)$	Acotada, diferenciable

Las ecuaciones que definen el comportamiento de una red neuronal se pueden expresar matemáticamente como una serie de operaciones matriciales. Sea X una matriz de entrada

con dimensiones (n, m), donde n es el número de observaciones y m es el número de características, W una matriz de pesos con dimensiones (m, p) y b un vector de sesgos con dimensiones $(1, p)$. La salida de una capa se puede calcular como:

$$Z = XW + b$$

La salida Z se pasa a través de una función de activación $f(Z)$ para producir la salida de la capa siguiente. La salida de la última capa se utiliza como la salida final de la red neuronal.

2.8.1.3. Algoritmo de Backpropagation

El algoritmo de backpropagation es un algoritmo utilizado en el aprendizaje supervisado para entrenar redes neuronales. El objetivo del algoritmo es ajustar los pesos de la red neuronal para minimizar una función de costo que mide la diferencia entre la salida predicha y la salida real.

Antes de explicar el algoritmo de backpropagation, es importante comprender el proceso de propagación hacia adelante (forward propagation) utilizado en el cálculo de la salida de la red neuronal. En la propagación hacia adelante, la entrada se pasa a través de la red neuronal capa por capa, calculando las salidas intermedias de cada capa hasta llegar a la capa de salida.

Suponga que se tiene una red neuronal con una capa de entrada, dos capas ocultas y una capa de salida. La entrada se representa como una matriz X con dimensiones (n, m), donde n es el número de observaciones y m es el número de características. Cada capa i de la red neuronal tiene una matriz de pesos $W^{(i)}$ con dimensiones (m_{i-1}, m_i) y un vector de sesgos $b^{(i)}$ con dimensiones $(1, m_i)$. La salida de cada capa i se calcula como:

$$Z^{(i)} = XW^{(i)} + b^{(i)}$$

A continuación, se pasa la salida $Z^{(i)}$ a través de una función de activación $f^{(i)}$ para producir la salida $A^{(i)}$ de la capa i:

$$A^{(i)} = f^{(i)}(Z^{(i)})$$

La salida de la última capa se utiliza como la salida final de la red neuronal.

Backpropagation

El algoritmo de backpropagation utiliza el gradiente descendente para ajustar los pesos de la red neuronal. El gradiente descendente es un método de optimización que busca ajustar los pesos de la red neuronal para minimizar una función de costo J.

En backpropagation, el gradiente descendente se aplica a través de la red neuronal en sentido inverso, comenzando desde la capa de salida y retrocediendo hacia la capa de entrada. El algoritmo utiliza la regla de la cadena para calcular la derivada parcial de la función de costo con respecto a los pesos de la red neuronal.

Sea J la función de costo y $w^{(i,j)}$ el peso de la conexión entre la neurona i de la capa

anterior y la neurona j de la capa actual. Entonces, la derivada parcial de J con respecto a $w^{(i,j)}$ se puede expresar como:

$$\frac{\partial J}{\partial w^{(i,j)}} = \frac{\partial J}{\partial z^{(j)}} \frac{\partial z^{(j)}}{\partial w^{(i,j)}}$$

Donde $z^{(j)}$ es la entrada a la neurona j de la capa actual. La primera derivada parcial se refiere a la contribución de la neurona j a la función de costo J, y la segunda derivada parcial se refiere a la contribución de la neurona i a la entrada de la neurona j.

La derivada parcial $\frac{\partial J}{\partial z^{(j)}}$ se puede calcular utilizando la regla de la cadena:

$$\frac{\partial J}{\partial z^{(j)}} = \sum_k \frac{\partial J}{\partial z^{(k)}} \frac{\partial z^{(k)}}{\partial z^{(j)}}$$

Donde la suma se realiza sobre todas las neuronas k que están conectadas a la neurona j. Esto significa que la contribución de la neurona j a la función de costo J depende de la contribución de todas las neuronas k a la entrada de la neurona j.

La derivada parcial $\frac{\partial z^{(j)}}{\partial w^{(i,j)}}$ se puede expresar como el valor de la entrada de la neurona i de la capa anterior multiplicado por el gradiente de la función de activación $f^{(j)}$:

$$\frac{\partial z^{(j)}}{\partial w^{(i,j)}} = a^{(i)} \frac{\partial f^{(j)}}{\partial z^{(j)}}$$

El gradiente de la función de activación $f^{(j)}$ se puede calcular a partir de la derivada de $f^{(j)}$ con respecto a su entrada $z^{(j)}$.

Finalmente, el gradiente descendente se utiliza para actualizar los pesos de la red neuronal en la dirección opuesta del gradiente. El proceso se repite hasta que se alcanza una convergencia satisfactoria de la función de costo J.

2.8.2. Modelo DQN

El algoritmo Deep Q-Network (DQN) combina Q-learning con deep learning para resolver problemas de RL con grandes espacios de estado-acción. En DQN, la red neuronal actúa como la función de aproximación $q(s, a; w)$, donde w son los pesos de la red.

2.8.2.1. Replay Buffer

Las experiencias de aprendizaje se almacenan en el replay buffer desde donde se muestrean al azar experiencias para aprender. Esta técnica se llama "Experience Replay" y permite descorrelacionar las experiencias, mejorando así el aprendizaje del agente.

El Replay Buffer $\mathcal{D}$ es una colección de transiciones pasadas $(s, a, r, s', \text{done})$ que se utilizan para entrenar la red neuronal[11]. Durante el entrenamiento, se muestrea un mini-lote de transiciones del Replay Buffer para calcular el Q-valor objetivo q_{target} mediante la ecuación:

$$q_{\text{target}} = r + \gamma \max_{a'} q(s', a'; w_{\text{target}}) \times (1 - \text{done})$$

done representa a la función $IsEnd(s)$ que hereda RL de MDP.

2.8.2.2. Target Network

La Target Network, con pesos w^-, se utiliza para estabilizar el aprendizaje[11]. La actualización de los pesos w_{target} puede hacerse de dos maneras:

1. **Hard Update**: Copiar los pesos de la red principal a la red objetivo después de $\mathcal{N}$ pasos.
$$w_{\text{target}} = w$$
2. **Soft Update**: Actualizar gradualmente los pesos de la red objetivo.
$$w_{\text{target}} = \tau w + (1 - \tau) w_{\text{target}}$$

Finalmente, el error de pérdida $\mathcal{L}$ que se minimiza generalmente mediante el Descenso de Gradiente Estocástico (SGD) se define como:

$$\mathcal{L}(w) = \mathbb{E}_{(s,a,r,s',\text{done})\sim\mathcal{D}}\left[(q_{\text{target}} - q(s,a;w))^2\right]$$

Capítulo 3

Metodología

3.1. Caso de estudio

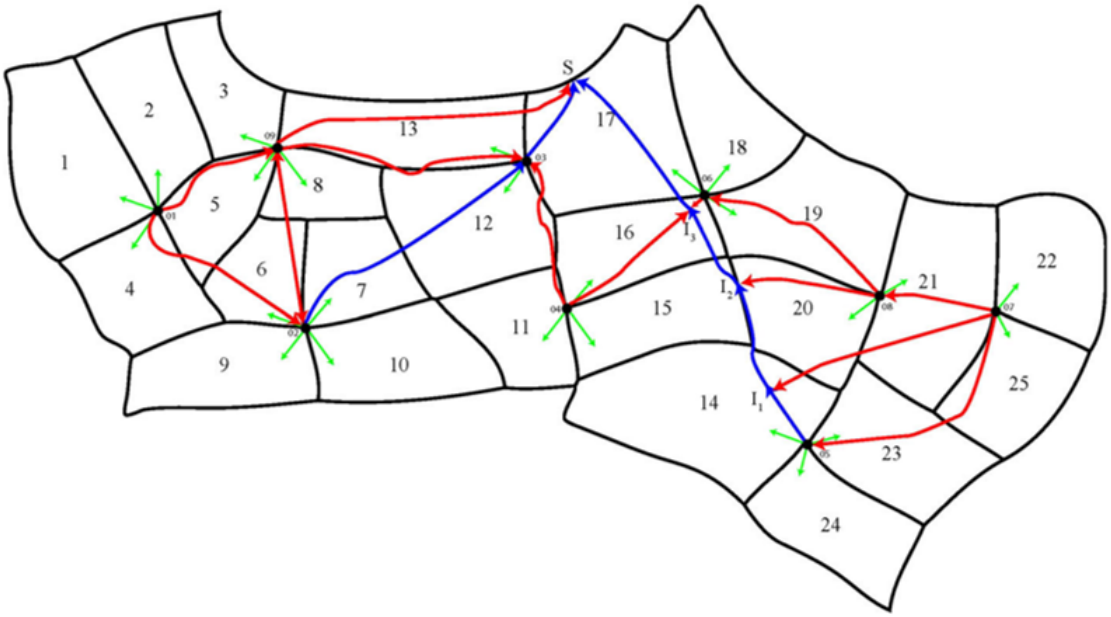

Figura 3.1: Representación gráfica del predio Los Copihues[13]

En la Figura 3.1 se muestra una descripción del bosque Los Copihues, que es el caso de estudio de este trabajo, y está compuesto por 25 rodales a ser cosechados. Los caminos existentes se representan con flechas azules, los caminos potenciales se representan con flechas rojas y el acceso a cada rodal se representa mediante flechas verdes. Finalmente, el punto S representa el nodo de salida del bosque, que se conecta con una carretera pública.

3.1.1. Descripción del problema

El problema consiste en modelar la red de caminos forestales que conectan nodos de origen (donde se originan las flechas verdes), intersección (señalados como I_1, I_2, I_3 en Figura 3.1) y salida, y los flujos que se transportan desde los nodos de origen hasta el nodo de salida debido a las acciones de cosecha de los rodales. Las acciones básicas deben ser construir caminos y

cosechar rodales. Con la acción de construir caminos se asocia un costo total agregado por la construcción, y con la acción de cosecha de un rodal se asocia un beneficio por la venta del volumen cosechado, que es básicamente el producto de un precio por unidad de volumen y la cantidad de volumen producido por la cosecha, un costo total agregado por la operación de cosecha y un costo por transporte hasta la salida que es el producto entre los km que se transporta el volumen producido y un costo fijo de transportar una unidad de volumen por un km. Este es un problema determinista, entonces la demanda de madera en cada periodo de tiempo es fija. El objetivo es maximizar las utilidades del administrador forestal a lo largo del horizonte de planificación que son 7 periodos.

3.2. Supuestos de todos los modelos

Los supuestos de mayor importancia para comprender la formulación del modelo MIP y la formulación de los entornos de aprendizaje por refuerzo son:

- La cadena de producción funciona con la lógica verticalmente integrada que incorpora al bosque y las plantas de producción: se conocen los parámetros de costos, precios, demandas, volumen de producción.
- Solamente se distingue un tipo de producto forestal producto de la cosecha. Además, no se hacen distinciones de la planta productiva a la que llega la madera, simplemente se cosecha y la madera es vendida directamente desde el bosque.
- No se almacena madera en la operación.
- Una vez que se decide cosechar un rodal, se cosecha la totalidad del volumen y no vuelve a plantarse, es decir, no existe vivero forestal.
- Son 7 los periodos que conforman el horizonte de planificación. En el primer periodo nunca se toman decisiones y se considera que es cuando se construyen los caminos de la red inicial de caminos, entonces la toma de decisiones comienza en el periodo 2 y son 6 los periodos en que se toman decisiones.
- Se debe satisfacer la demanda en todos los periodos.
- No se puede cosechar más de la mitad de los rodales antes de la mitad del horizonte de planificación.
- No pueden cosecharse rodales que sean adyacentes entre si en el mismo periodo. La adyacencia de cada uno de los rodales se encuentra en la sección Conjuntos del Modelo MIP.
- Una vez que se cosecha un rodal, su volumen debe ser enviado por la ruta más corta hacia la salida del bosque para ser vendido.
- Como ya se mencionó, este es un problema determinista y por lo tanto **los parámetros de distancia en kilómetros entre los nodos i y j, precio de venta por metro cúbico de madera en el periodo t, costo fijo por transportar un metro cúbico de madera por kilómetro, costo de construcción de un camino entre los nodos i y j en el periodo t, costo de corte de un rodal r en el periodo t, producción del rodal r en el periodo t, demanda de volumen de madera en el periodo t son fijos y son compartidos por todos los modelos.**

- La construcción del camino (i, j) implica la construcción inmediata del camino (j, i) y se aplica el costo de solo uno de los caminos, no su suma de costos.

3.3. Modelo MIP y solución óptima

En esta sección se detalla un modelo de optimización programado con la librería Pyomo en lenguaje Python para maximizar las utilidades del administrador forestal en la gestión del transporte de madera desde los nodos de origen hasta el nodo de salida en una red de caminos forestales. Se abarca el caso del predio Los Copihues de la región del Maule, Chile. El modelo tiene en cuenta múltiples factores, como costos de transporte, construcción de caminos, demanda y producción de rodales. Además, se enfatiza la importancia de la relación entre las diferentes partes del modelo para garantizar que las restricciones se cumplan y se logre el objetivo deseado.

A continuación, se describen los conjuntos, parámetros y variables utilizados en el modelo:

3.3.1. Conjuntos

- N = conjunto de todos los nodos en el grafo: $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]$
- N^O = conjunto de nodos de origen: $[1, 2, 3, 4, 5, 6, 7, 8, 9]$
- N^I = conjunto de nodos de intersección: $[10, 11, 12]$
- N^S = conjunto de nodos de salida: $[13]$
- A = conjunto de aristas del grafo. En total, la cantidad total de aristas es 38, dado que el conjunto de todos los nodos y los caminos no forman un grafo completo. Se asume que la existencia de (i, j) implica la existencia de (j, i)

Nodo 1	Nodo 2
1	2
1	9
2	3
2	9
3	4
3	9
3	13
4	12
5	7
5	10
6	12
6	8
7	8
7	10
8	11
9	13
10	11
11	12
12	13

- R = conjunto de rodales en el bosque: $[1, 2, 3, ..., 23, 24, 25]$
- T = conjunto de periodos de tiempo: $[1, 2, 3, 4, 5, 6, 7]$
- $RA(n)$ = conjunto de rodales asociados al nodo origen n:

Nodo n	rodales asociados
1	$1, 2, 4$
2	$6, 7, 9, 10$
3	$12, 13$
4	$11, 15, 16$
5	$14, 23, 24$
6	$17, 18, 19$
7	$22, 25$
8	$20, 21$
9	$3, 5, 8$

- $RAR(r)$ = conjunto de rodales adyacentes al rodal r:

Rodal r	rodales adyacentes
1	2, 4
2	1, 3, 5
3	2, 5, 13
4	1, 5, 6, 9
5	2, 3, 4, 6, 8
6	4, 5, 7, 8, 9
7	6, 7, 10, 12
8	6, 7, 5, 12, 13
9	4, 6, 10
10	9, 7, 11
11	10, 12, 15, 16
12	7, 8, 11, 13, 16, 17
13	3, 8, 12, 17
14	11, 15, 20, 23, 24
15	11, 16, 20, 14
16	11, 12, 15, 17, 19
17	12, 13, 16, 18
18	17, 19
19	18, 16, 20, 21
20	14, 15, 19, 23, 25
21	19, 22, 25
22	21, 25
23	20, 21, 25, 24, 14
24	14, 23
25	21, 22, 23

- $N_{predecesor}(n)$ = conjunto de nodos predecesores del nodo n en la red de caminos:

Nodo n	Predecesores
1	–
2	$1, 9$
3	$2, 4, 9$
4	–
5	7
6	8
7	–
8	7
9	$1, 2$
10	$5, 7$
11	$8, 10$
12	$4, 6, 11$
13	$3, 9, 12$

- $N_{sucesor}(n)$ = conjunto de nodos sucesores del nodo n en la red de caminos:

Nodo n	Sucesores
1	$2, 9$
2	$3, 9$
3	$9, 13$
4	$3, 12$
5	10
6	12
7	$5, 8, 10$
8	$6, 11$
9	$2, 3, 13$
10	11
11	12
12	13
13	–

- CI = conjunto de aristas que representan la red de caminos inicial. En total son 12 los caminos iniciales, se asume la existencia de los caminos (3,2),(13,3),(10,5),(11,10),(12,11),(13,12):

Nodo 1	Nodo 2
2	3
3	13
5	10
10	11
11	12
12	13

3.3.2. Parámetros

- $d_{i,j}$: distancia en kilómetros entre los nodos i y j.
- p_t: precio de venta por metro cúbico de madera en el periodo t. Se consideran precios entre 180 y 200 USD que están cercanos al valor por metro cúbico del pino radiata[3].
- k: costo fijo por transportar un metro cúbico de madera por kilómetro. Para este caso se considera igual a 12,02 USD.
- $c_{i,j,t}^{build}$: costo de construcción de un camino entre los nodos i y j en el periodo t. Se consideran precios de valores enteros entre 58.000 y 60.000 USD. Los costos en el periodo 1 son nulos.
- $c_{r,t}^{harvest}$: costo de corte de un rodal r en el periodo t. Se consideran costos con valores enteros entre 10.000 y 12.000 USD
- $P_{r,t}$: producción del rodal r en el periodo t. Está medida en decenas de metros cúbicos y se simularon valores de producción entre 510 y 950 decenas de metros cúbicos. El bosque total posee 160.000 m^3, entonces se modela una cantidad total de magnitud similar al original.
- D_t: demanda de volumen de madera en el periodo t. Se consideran demandas por decenas de m^3 con valores entero entre 1.304 y 1.346 decenas de m^3

3.3.3. Variables

- $E_{r,t}$: variable binaria que indica si se corta el rodal r en el periodo t.
- $W_{i,j}^t$: variable binaria que indica si se construye el camino entre los nodos i y j en el periodo t.
- $X_{i,j}^t$: variable binaria que indica si el camino entre los nodos i y j está disponible en el periodo t.
- $Y_{n,t}$: volumen enviado desde el nodo origen n en el periodo t.
- $F_{i,j}^t$: flujo de volumen enviado entre los nodos i y j en el periodo t.

3.3.4. Función objetivo

La función objetivo busca maximizar las utilidades del administrador forestal en cada periodo de tiempo t. Estas utilidades se calculan como la diferencia entre los ingresos y los costos asociados al transporte, la construcción de caminos y el corte de rodales.

$$\max \Omega = \sum_{t \in T} \left(\sum_{n \in N^O} p_t \cdot Y_{n,t} - \sum_{(i,j) \in A} k \cdot d_{i,j} \cdot F_{i,j}^t - \frac{1}{2} \cdot \sum_{(i,j) \in A} c_{i,j}^{build} \cdot W_{i,j}^t - \sum_{r \in R} c_{r,t}^{harvest} \cdot E_{r,t} \right) \quad (3.1)$$

3.3.5. Restricciones

3.3.5.1. Restricción para periodo inicial

$$\sum_{(i,j)\in A} W^1_{i,j} + X^1_{i,j} + \sum_{r\in R} E_{r,1} = 24, \tag{3.2}$$

Esta restricción garantiza que en el primer periodo solamente se construyen los caminos que pertecen a CI. En total son 13 caminos que deben ser construídos y 6 caminos que conforman la red de caminos inicial, es decir, la cantidad de variables XyW que deben tener valor 1 son 24 (6 que indican que (i, j) se construyen más 6 que indican que (j, i) se construyen más 6 que indican que (i, j) está disponible para ser transitado más 6 que indican que (j, i) está disponible para ser transitado.

3.3.5.2. Restricción de demanda

$$\sum_{r\in R} E_{r,t} \cdot P_{r,t} \geq D_t, \quad \forall t \in T \tag{3.3}$$

Esta restricción asegura que la demanda de madera en cada periodo se cumpla. La cantidad total de madera cosechada debe ser al menos igual a la demanda en ese periodo.

3.3.5.3. Restricción de máximo de rodales cortados

$$\sum_{r\in R} E_{r,t} \leq 5, \quad \forall t \in T \tag{3.4}$$

Esta restricción limita la cantidad de rodales que se pueden cortar en cada periodo a 5. Representa una restricción de capacidad de producción en el sistema forestal.

3.3.5.4. Restricción de disponibilidad de caminos

$$X^t_{i,j} \geq X^{t-1}_{i,j} + W^t_{i,j}, \quad \forall (i, j) \in A, t \geq 1 \tag{3.5}$$

Esta restricción garantiza que un camino (i, j) será parte de la red de caminos construída en el periodo t solo si es construida en el periodo t, es decir, que la suma de la variable de construcción debe ser igual a cero para todos los periodos anteriores o iguales a t.

3.3.5.5. Restricción de disponibilidad desde la construcción

$$X^t_{i,j} \leq \sum_{k=1}^{t} W^k_{i,j}, \quad \forall (i, j) \in A, \forall t \in T \tag{3.6}$$

Esta restricción garantiza que $X^t_{i,j}$ no será parte de la red de caminos hasta que se haya construído con $W^t_{i,j} = 1$.

3.3.5.6. Restricción caminos no aislados e inicialización de X y W

$$\begin{cases} W_{i,j,1} = 1 & \forall (i,j) \in A, t = 1, (i,j) \in CI \\ W_{i,j,1} = 0 & \forall (i,j) \in A, t = 1, (i,j) \notin CI \\ W_{i,j,t} \leq \sum_{(p,q) \in CE(t)} X_{p,q,t-1} & \forall (i,j) \in A, t > 1, (p = i \vee p = j \vee q = i \vee q = j) \wedge (p,q) \neq (i,j) \end{cases}$$

Donde $CE(t)$ es un conjunto auxiliar que contiene a todos los nodos que conforman la red de caminos existentes hasta el periodo t, es decir, $CE(1) = CI$.
En palabras, esta restricción indica que una arista (i,j) puede ser construida en la etapa t si no ha sido construida antes (en las etapas anteriores) y si existe al menos una arista en la etapa anterior que comparta un nodo con la arista (i,j).

3.3.5.7. Restricción de flujo de volumen

$$F_{i,j}^t \leq 10^6 \cdot X_{i,j}^t, \quad \forall (i,j) \in A, \forall t \in T \tag{3.7}$$

Esta restricción limita el flujo de volumen de madera en un camino (i,j) según si el camino está construido o no. Si el camino no está construido, el flujo de volumen será cero.

3.3.5.8. Restricción de flujo en intersecciones

$$\sum_{j \in N_{sucesor}(n)} F_{n,j}^t = \sum_{i \in N_{predecesor}(n)} F_{i,n}^t, \quad \forall n \in N^I, \forall t \in T \tag{3.8}$$

Esta restricción garantiza la conservación del flujo de volumen de madera en las intersecciones de la red de caminos. El volumen de madera que llega a una intersección debe ser igual al volumen de madera que sale de ella.

3.3.5.9. Restricción de flujo en nodos de origen

$$\sum_{j \in N_{sucesor}(n)} F_{n,j}^t = Y_{n,t} + \sum_{i \in N_{predecesor}(n)} F_{i,n}^t, \quad \forall n \in N^O, \forall t \in T \tag{3.9}$$

Esta restricción garantiza que el flujo de volumen de madera en los nodos de origen sea igual al volumen de madera cosechado en ese nodo más el flujo de volumen que llega al nodo.

3.3.5.10. Restricción de volumen cosechado

$$Y_{n,t} = \sum_{r \in RA(n)} E_{r,t} \cdot P_{r,t}, \quad \forall n \in N^O, \forall t \in T \tag{3.10}$$

Esta restricción garantiza que el volumen de madera cosechado en cada nodo de origen sea igual a la suma de los volúmenes cosechados en los rodales asociados.

3.3.5.11. Restricción de flujo hacia nodo de salida

$$\sum_{i \in N_{predecesores}(m)} F_{i,m}^t = \sum_{n \in N^O} \sum_{r \in RA(n)} E_{r,t} \cdot P_{r,t}, \quad \forall m \in N^S, \forall t \in T \tag{3.11}$$

Esta restricción garantiza que la cantidad de madera que llega al nodo de salida m sea igual a la suma de la producción de todos los rodales cosechados.

3.3.5.12. Restricción de flujo desde el nodo de salida

$$\sum_{i \in N_{predecesores}(n)} F_{n,i}^{t} = 0, \quad \forall n \in N^{S}, \forall t \in T \tag{3.12}$$

La restricción garantiza que no haya flujo de madera desde el nodo de salida hacia sus nodos antecesores.

3.3.5.13. Restricción de caminos iguales en ambos sentidos

$$W_{i,j}^{t} = W_{j,i}^{t}, \quad \forall (i,j) \in A, t \in T \tag{3.13}$$

La restricción establece que los caminos i, j y j, i son el mismo camino, y por lo tanto, deben tener los mismos valores de construcción de camino en el modelo.

3.3.5.14. Restricción de disponibilidad de caminos en ambos sentidos

$$X_{i,j}^{t} = X_{j,i}^{t}, \quad \forall (i,j) \in A, t \in T \tag{3.14}$$

La restricción establece que los caminos i, j y j, i son el mismo camino, y por lo tanto, deben tener los mismos valores de disponibilidad en el modelo.

3.3.5.15. Restricción de cosecha única

$$\sum_{t \in T} E_{r,t} \leq 1, \quad \forall r \in R \tag{3.15}$$

La restricción establece que un rodal solo puede ser cortado una vez a lo largo de todos los períodos de tiempo.

3.3.5.16. Construcción única de cada camino

$$\sum_{t \in T} W_{i,j}^{t} \leq 1, \quad \forall (i,j) \in A \tag{3.16}$$

La restricción establece que un camino solo puede ser construido una vez a lo largo de todos los períodos de tiempo.

3.3.5.17. Restricción de adyacencia entre rodales

$$E_{r,t} + E_{s,t} \leq 1, \quad \forall r \in R, \forall s \in RAR(r), \forall t \in T \tag{3.17}$$

La restricción establece que no se pueden cortar rodales adyacentes en el mismo período de tiempo.

3.3.6. Discusión

Las restricciones presentadas en este informe son coherentes y garantizan que se cumplan las condiciones operativas del sistema forestal. Sin embargo, no garantizan explícitamente que

se siga la ruta más corta hacia el destino. A pesar de ello, la función objetivo busca maximizar las utilidades del administrador forestal, lo que implica minimizar los costos de transporte de la madera. Por lo tanto, el modelo de optimización tenderá a seleccionar rutas más cortas y económicas para satisfacer la demanda. Además, las restricciones actuales obligan a tomar decisiones a partir del periodo 2, las únicas decisiones que se toman en el periodo 1 son la construcción e incorporación a la red de los caminos $(i, j) \in CI$.

3.3.7. Resultados

El resultado óptimo de utilidad es negativo (pérdida) $\Omega = -2.535.260USD$.
El calendario de corte de los rodales es la representación gráfica de los valores de la variable $E_{r,t}$.
A su vez, las decisiones de construcción de caminos para expandir la red de caminos inicial está ocurriendo. En las gráficas de solución óptima las conexiones de color azul representan los valores distintos de cero de $X^t_{i,j}$, las conexiones de color amarillo representan los valores distintos de $W^t_{i,j}$ y las conexiones de color rojo representan los valores distintos de cero de $F^t_{i,j}$. También puede apreciarse cuales son los rodales que se cortan en cada periodo, con color rojo si fue cortado en el periodo ($E_{r,t} = 1$) y color verde si no. Adicionalmente se mencionan los valores de la demanda D_t y el costo k de transportar $1m^3$ de madera por $1km$.

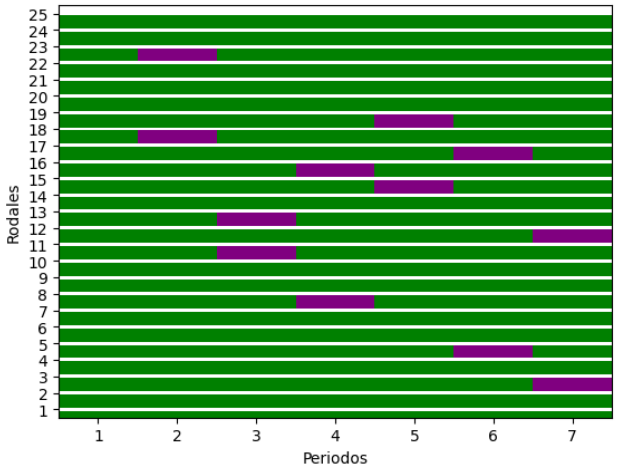

Figura 3.2: Calendario óptimo de cosecha de rodales

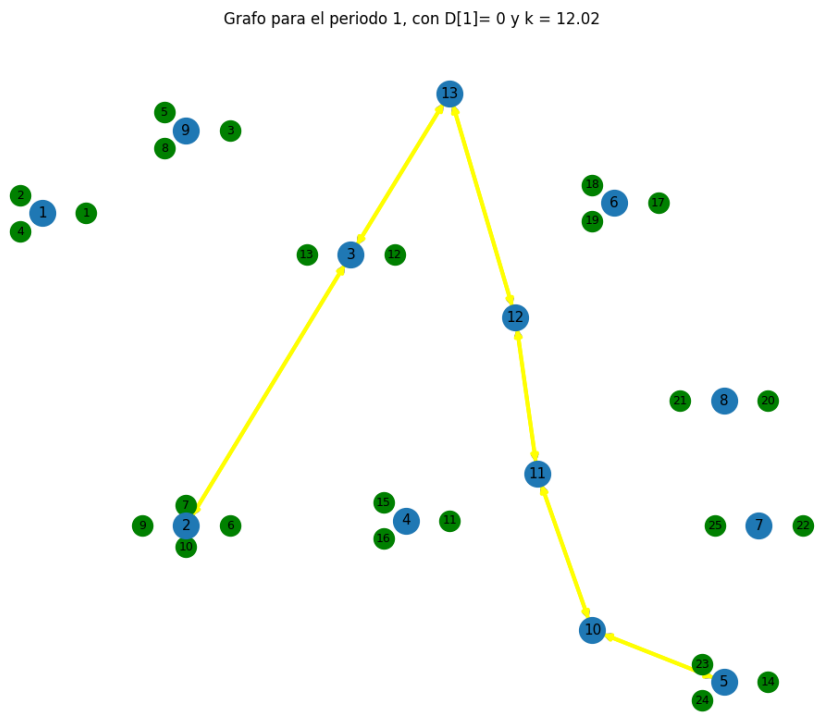

Figura 3.3: Solución óptima del periodo 1

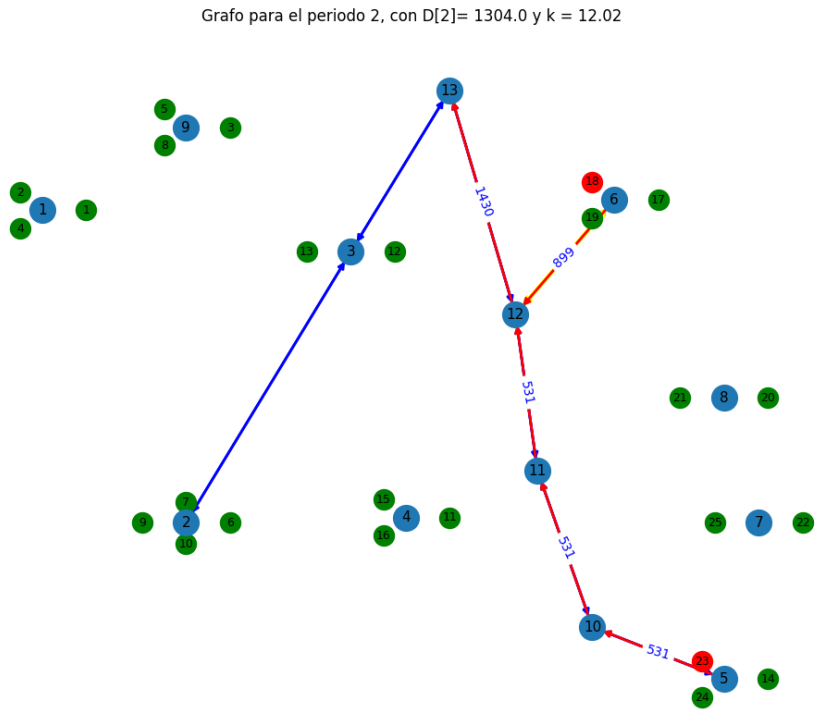

Figura 3.4: Solución óptima del periodo 2

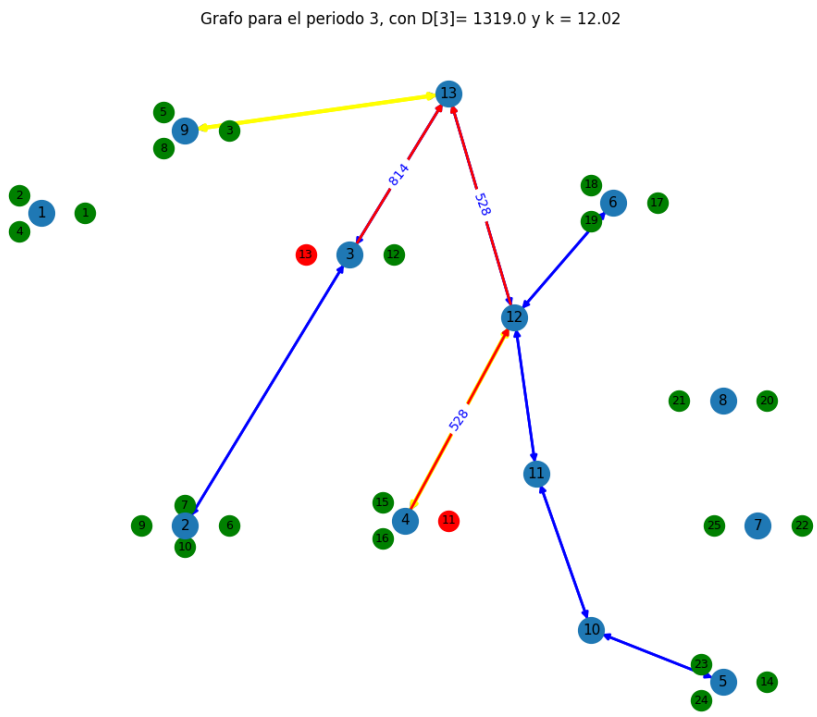

Figura 3.5: Solución óptima del periodo 3

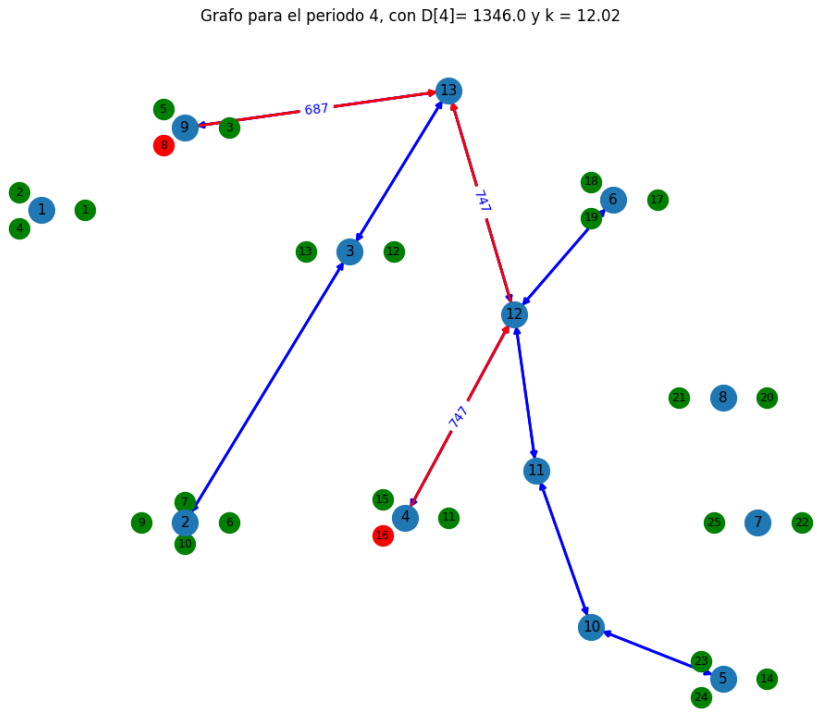

Figura 3.6: Solución óptima del periodo 4

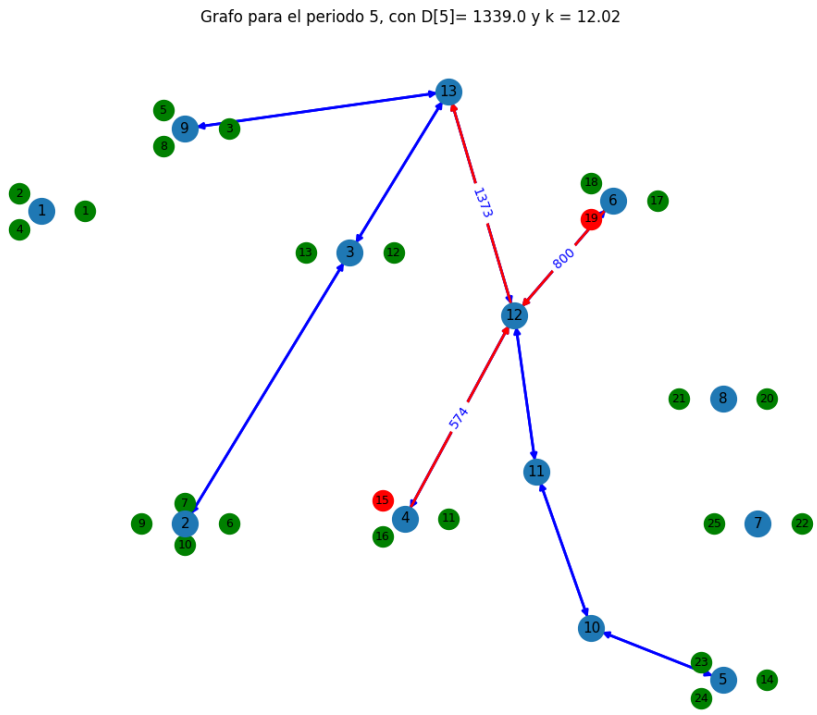

Figura 3.7: Solución óptima del periodo 5

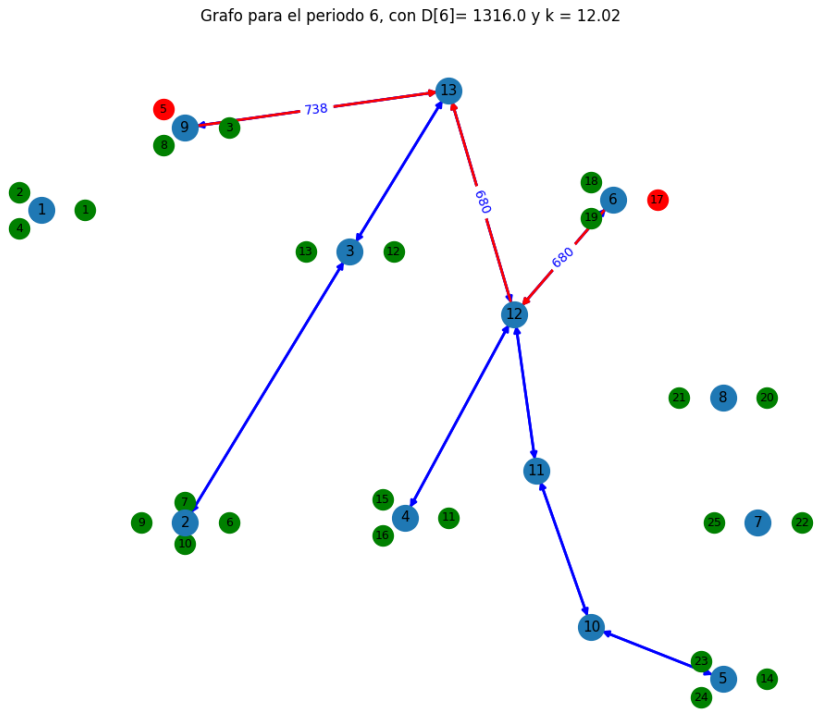

Figura 3.8: Solución óptima del periodo 6

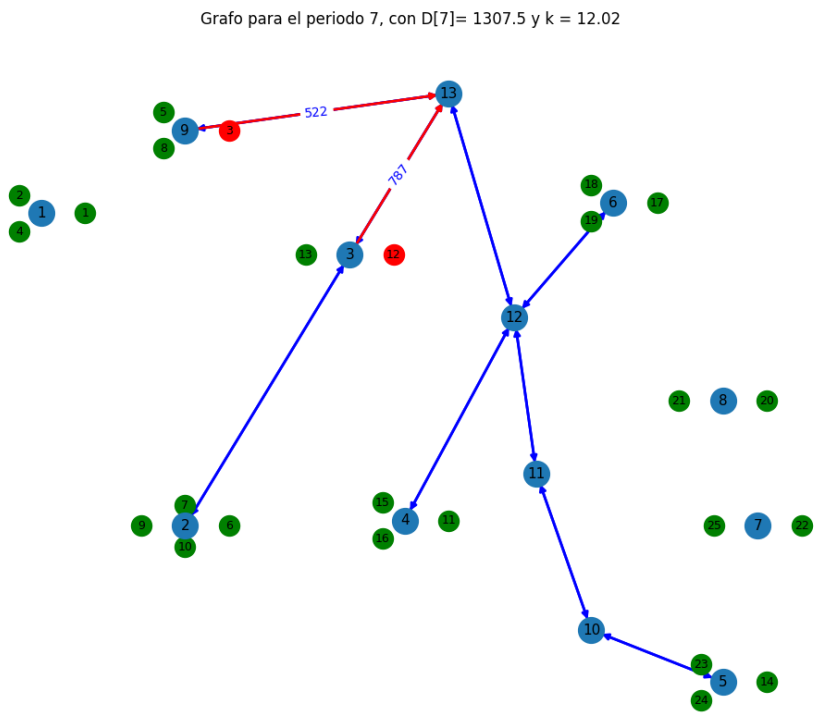

Figura 3.9: Solución óptima del periodo 7

3.3.8. Conclusiones

El calendario óptimo de cosecha considera como máximo la cosecha de dos rodales para satisfacer la demanda en cada periodo. La solución óptima no considera la construcción de los caminos hacia los nodos de orígen número 1, 7 y 8 y por ende tampoco considera la cosecha de rodales asociados. Por su parte la construcción de caminos se da hasta el periodo 3.

La formulación de este problema se puede extrapolar a casos de mayor tamaño, solamente deben expandirse los conjuntos y agregar restricciones acorde a la expansión del modelo.

3.4. Diseño del entorno de aprendizaje

Al modelar el problema mediante MIP se obtiene la política óptima de acciones π^* para el caso de estudio. Sin embargo, la hipótesis principal de este trabajo es que se puede aprender la política óptima utilizando técnicas y modelos de DRL. Entonces el entorno de aprendizaje debe recrear el problema con restricciones y recompensas equivalentes a las del MIP, pero en el marco teórico de un RL para así utilizar técnicas que permitan aproximar $q^*(s, a)$ directamente de la experiencia con el entorno.

3.4.1. Representación del Estado

Para efectos prácticos de resolver exactamente el mismo problema de optimización que con el modelo MIP, el entorno de aprendizaje por refuerzo reúne información de cada rodal, cada camino, periodo de tiempo, volumen cosechado, nivel de demanda y cantidad total de

rodales cosechados. También se añade una variable binaria que funciona como verificador de que se satisface la demanda con el volumen cosechado en ese estado. Entonces cada estado s_t del sistema se representa como un vector de 49 características ordenadas:

$$s_t = \Bigg(\underbrace{(s^1, ..., s^{25},}_{\text{Estado rodales}} \mid \underbrace{s^{26}, ..., s^{44},}_{\text{Estado caminos}} \mid \underbrace{s^{45}}_{\text{Periodo } t} , \mid \underbrace{s^{46}}_{\text{Volumen } V_t} , \mid \underbrace{s^{47}}_{\text{Demanda } D_t} , \mid \underbrace{s^{48}}_{V_t \geq D_t} , \mid \underbrace{s^{49}}_{\text{N° rodales cosechados}} \Bigg)$$

- En los primeros 25 valores el estado de cada uno de los rodales. El estado de cada rodal puede ser:
  - 1: Si el rodal ya fue cosechado
  - 0: Si el rodal está disponible para ser cosechado.
  - -1: Si el rodal no está disponible para ser cosechado.

  Para que un rodal esté disponible para ser cosechado significa que:
  - Existe un camino que permite acceder al rodal.
  - Ningún rodal adyacente al rodal en cuestión ha sido cosechado en el mismo periodo de tiempo.
- En los siguientes 19 valores el estado de cada uno de los caminos. El estado de cada caminos puede ser:
  - 1: Si el camino ya fue construido
  - 0: Si el camino está disponible para ser construido.
  - -1: Si el camino no está disponible para ser construido.

  Para que un camino esté disponible para ser cosechado significa que posee conexión directa con la red de caminos construidas hasta ese instante, es decir, que no es un camino aislado de la red.
  El conjunto A de aristas planteado en el MIP muestra todos los caminos posibles. Para reconocer cada camino en esta formulación de MDP se agrega el nombre con que se reconoce cada camino, el parámetro de distancias y, sin perder el orden de la sección de MIP, se presentan de la siguiente manera:

Camino	Nodo 1	Nodo 2	$d_{Nodo1,Nodo2}[km]$
Camino 7	1	2	30
Camino 8	1	9	20
Camino 1	2	3	40
Camino 9	2	9	30
Camino 12	3	4	30
Camino 11	3	9	45
Camino 2	3	13	17
Camino 13	4	12	23
Camino 19	5	7	20
Camino 6	5	10	20
Camino 14	6	12	5
Camino 15	6	8	35
Camino 17	7	8	10
Camino 18	7	10	45
Camino 16	8	11	30
Camino 10	9	13	50
Camino 5	10	11	18
Camino 4	11	12	15
Camino 3	12	13	6

Los caminos que conforman la red en el estado inicial son Camino 1, Camino 2, Camino 3, Camino 4, Camino 5 y Camino 6.

- A continuación del último valor de caminos se encuentra el valor del periodo de tiempo del entorno. El valor mínimo es 2 y máximo es 7.
- Luego del periodo de tiempo se encuentra el valor de volumen cosechado hasta ese instante. El valor mínimo es cero y el valor máximo se establece en 6500 decenas de m^3, que representa aproximadamente la cosecha de 8 rodales en un mismo periodo.
- Después del valor de volumen cosechado está el valor de la demanda en ese periodo de tiempo. Los valores van desde 1304 a 1346 decenas de m^3.
- En la penútima posición del vector se encuentra un verificador binario que indica si se satisface la demanda con el volumen cosechado.
- Finalmente se encuentra el valor de la cantidad total de rodales cosechados hasta ese instante, que se calcula internamente por el ambiente, aunque también se puede deducir de los primeros 25 valores de s_t. En este caso el mínimo es cero y el máximo 25 que es el total de rodales.

El cálculo combinatorio para determinar el número total de estados posibles se realiza multiplicando las posibilidades de cada uno de los componentes del vector. Para el vector de 49 valores descrito:

- Los primeros 44 valores (rodales y caminos) pueden tener 3 estados posibles: -1, 0 o 1, representando respectivamente si el rodal/camino no está disponible, está disponible o ya ha sido cosechado/construido.
- El término 45 puede tener 6 valores distintos.
- El término 46 puede tener, suponiendo que el volumen cosechado tenga 200 valores distintos, 200 valores distintos.
- El término 47 puede tener 6 valores distintos.
- El término 48 puede ser 1 o -1, es decir, tiene 2 valores distintos.
- El término 49 puede tener 26 valores distintos.

Entonces, el número total de estados posibles es:

$$\text{Número de Estados Posibles} = 3^{44} * 6 * 200 * 6 * 2 * 26$$

Este valor nos da una dimensión de la complejidad y la magnitud del espacio de estados con el cual estamos trabajando, lo que subraya la importancia de utilizar técnicas avanzadas como DQN para abordar problemas de este tipo y escala.

En la Figura 3.10 se resume la estructura que define a un estado s_t antes de normalizar los datos de entrada.

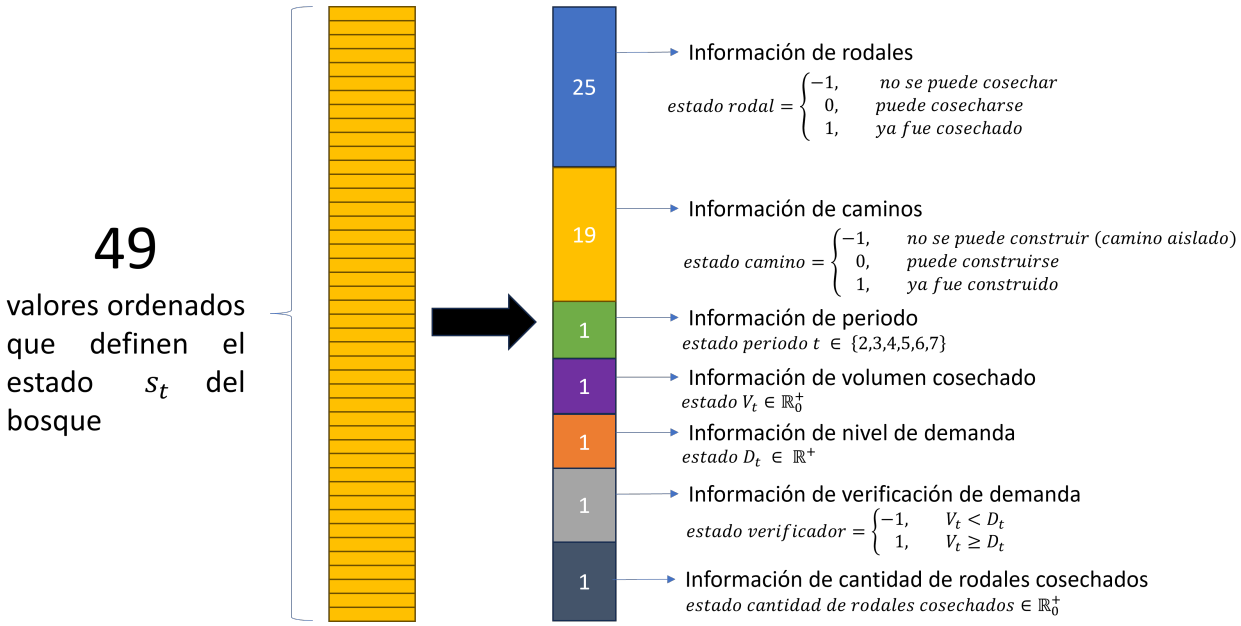

Figura 3.10: Representación gráfica del estado con 49 valores y la naturaleza de cada uno de sus valores antes de la normalización

3.4.1.1. Normalización de datos de entrada

La normalización de los datos del estado es un paso crucial en la preparación de datos para muchos algoritmos de aprendizaje automático. Los algoritmos de aprendizaje a menudo son sensibles a la escala y la distribución de los datos de entrada. Al normalizar los datos, es

decir, al escalar los atributos para que estén en rangos comparables, se mejora la estabilidad numérica del modelo y se acelera el proceso de entrenamiento. Exceptuando los estados de rodales, caminos y del verificador de demanda que ya se encuentran normalizados, el resto de valores x de un estado se modifican según:

$$x_{\text{normalizado}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

3.4.2. Condición Terminal

Las transiciones entre estados ocurren luego de que el agente toma una acción. Por las restricciones del modelo puede darse que el agente transite hacia estados donde no tenga más opciones que escoger. Para anticipar la ocurrencia de estos casos se programa una respuesta de fin del episodio. Existen dos transiciones que llevarán al agente a estos casos:

- El agente llega a un estado donde no hay acciones disponibles.
- El agente llega al límite de cosecha permitida justo en la mitad de los periodos de toma de decisiones (periodo 4).

En ambos casos se evalúa si se satisface la demanda ($\frac{D_t}{V_t} \leq 1$)en el estado al que llegó. En caso de satisfacer la demanda, se avanza al siguiente periodo automáticamente y en caso contrario se finaliza el episodio, ya que no puede avanzar de periodo si no satisface la demanda con el volumen cosechado hasta ese instante. En el último caso se finaliza el episodio instantáneamente ya que, si avanza de periodo, de todas formas se violaría la restricción de límite de cosecha tomando una acción de cosecha en el periodo 4 buscando satisfacer la demanda.

Este ambiente está diseñado para responder a todos las acciones que se tomen. En este caso existe una máscara de acciones, pero en caso de no querer usarla y **el agente escoja una acción inválida, entonces no se realiza transición de estados y el episodio finaliza instantáneamente.**

3.4.3. Acciones

Las acciones que puede tomar un agente son de tres tipos:

- Cosecha de rodal
- Construcción de camino
- Ir al siguiente periodo

Estas acciones están representadas como índices en el espacio de acciones A, y se ordenan en un vector de acciones como se aprecia en la Figura 3.11. En total son 45 acciones:

- Las 25 primeras están relacionadas con la acción de corte para cada rodal.
- Las siguientes 19 acciones están relacionadas con la acción de construir un camino. Los primeros 6 caminos representan la red inicial de camino.
- Finalmente la última acción es la de ir al siguiente periodo. Esta acción es la **única que puede ser escogida más de una vez.**

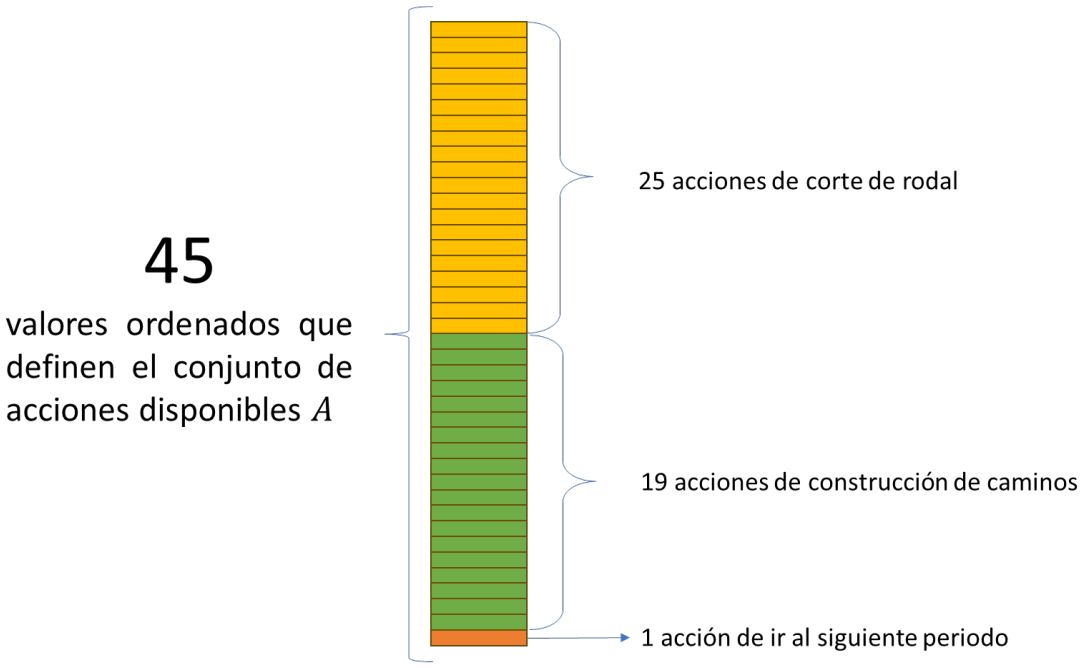

Figura 3.11: Representación gráfica del espacio de acciones con 45 valores

3.4.3.1. Máscara de acciones

Para la implementación de agentes inteligentes se utiliza una **máscara de acciones**, que es básicamente un vector con dimensiones iguales al espacio de acciones aunque poblado con variables binarias donde 0 representa una acción inválida y 1 representa una acción válida. El propósito de una máscara de acciones es filtrar únicamente las acciones disponibles en un estado dado, asegurando que el agente solo elija acciones válidas dada la naturaleza de la mayoría de variables que pueden estar disponibles hasta que son escogidas y en adelante pasan a ser acciones restringidas.
Identificar las acciones válidas/inválidas es simple con el diseño de estados propuesto:

- Si un rodal ya se cosechó, entonces esta acción pasa a ser restringida por la máscara.
- Si se cosechó un rodal, entonces todos los rodales adyacentes disponibles para ser cosechados pasan a no estar disponibles y se restringe su elección por la máscara de acciones.
- Si un camino ya se construyó, entonces esta acción pasa a ser restringida por la máscara de acciones.
- Si un camino se encuentra aislado de la red de caminos, entonces pasa a estar restringido por la máscara de acciones.
- La acción de ir al siguiente periodo es válida solo si el verificador de satisfacción de demanda es igual a 1.
- Con la elección de la acción de ir al siguiente periodo, todos los rodales disponibles para ser cosechados que quedaron restringidos dejan de estarlo.

El conjunto resultante aplicar la máscara de acciones se denomina $\mathcal{M}$ y contiene todas las acciones no restringidas en función de s_t. Entonces el conjunto de acciones disponibles en un

estado s_t está definido por la función de la máscara de acciones:

$$A(s_t) = \mathcal{M}$$

3.4.4. Recompensas

La función de recompensa depende exclusivamente de las acciones que se escojan dado un periodo t y son independientes del estado al que se llega cada acción exceptuando los casos en que las acciones lleven al agente no acabe un episodio exitósamente, es decir, que finalice el episodio antes de llegar al horizonte de planificación.

Las recompensas están incluídas en la señal que envía el entorno al agente luego de ejecutar una acción. Las recompensas representan los ingresos y costos asociados con las acciones del agente.

- **Recompensas asociadas a cosechas:** ocurren por acciones de cosecha de rodales. Equivalen a los ingresos por la venta de volumen de madera cosechada. Con la cosecha de rodales se asume el costo total por la cosecha del volumen total del rodal sumado al costo de transportar el volumen desde su orígen hasta la salida del bosque.
  Si a_t es una acción de corte del rodal z entonces la recompensa se define como:

$$R(s_t, a_t) = \underbrace{p_t \cdot P_{z,t}}_{\text{Ingresos}} - \underbrace{c_{z,t}^{harvest}}_{\text{Costo de corte}} - \underbrace{k \cdot Dijkstra(\mathcal{S}, z^{\text{nodo origen}})}_{\text{Costo de transporte}}$$

  El nodo de salida está simbolizado por $\mathcal{S}$ al igual que en el mapa de Los Copihues. $Dijkstra(x)$ es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de los vértices en un grafo con pesos en cada arista.

  En el grafo de caminos construídos los pesos están definidos por la distancia entre nodos $d_{i,j}$ En este caso los parámetros para calcular la distancia mínima son el nodo de salida $\mathcal{S}$ y el nodo de orígen asociado al rodal z denominado $z^{\text{nodo origen}}$.

- **Recompensas asociadas a construcción de caminos:** ocurren por acciones de construcción de nuevos caminos o la cosecha de rodales. Con la construcción de nuevos caminos se asume el costo total de construcción del camino.

  Si a_t es una acción de construcción de camino definido por el los nodos (i, j):

$$R(s_t, a_t) = \underbrace{-c_{i,j,t}^{build}}_{\text{Costo de construcción}}$$

- **Recompensas asociadas a transición exitosa entre periodos:** si el agente elige la opción de ir al siguiente periodo A^{45}, entonces recibirá una recompensa equivalente a la razón $\frac{Demanda}{Volumen}$. Si esta razón es mayor a 0.8, entonces la recompensa será de $1000 \cdot \frac{D_t}{V_t} USD$ y en caso contrario la recompensa será el valor de la razón $1 \cdot \frac{D_t}{V_t} USD$.

  Esta recompensa está pensada para guiar mediante un pequeño incentivo al agente a que tome la acción de transitar entre periodos, y será mayor entre más cercana sea la

razón $\frac{Demanda}{Volumen}$ de 1:

$$R(s_t, a_t = A^{45}) = \begin{cases} 100.000 \cdot \frac{D_t}{V_t} & \text{para } \frac{D_t}{V_t} \geq 0.8 \\ 10.000 \frac{D_t}{V_t} & \text{de lo contrario} \end{cases}$$

Por otro lado están las **recompensas en caso que el agente acabe un episodio exitosamente o fallidamente**:

- Si el agente logra finalizar con éxito un episodio recibe una recompensa de $R(s_7 = IsEnd(s_t) = 1, a_T = A^{45}) = 100.000USD$.
- Si el agente llega al límite de rodales que pueden cosecharse antes de la mitad de los periodos que tiene para tomar decisiones, entonces finaliza el episodio y recibe una penalización equivalente al valor absoluto de la recompensa acumulada óptima. Si a_t es la acción que lo lleva a violar el límite: $R(s_t = IsEnd(s_t) = 1, a_t) = -2.535.260USD$.
- Para el caso en donde no se utilice una máscara de acciones y el agente escoge un acción que no es válida en el estado en que se encuentra recibe la penalización más alta. Si a_t es restringida, entonces la recompensa es $R(s_t = IsEnd(s_t) = 1, a_t) = -10.000.000USD$.

3.4.5. Validación del Diseño

Para testear que el diseño transita y recompensa como se modela, se ejecuta la política óptima, obteniendo la recompensa total acumulada cercana a la política óptima π^*. La diferencia se debe a las recompensas intermedias por transitar entre periodos que recibe el agente.

La secuencia de acciones que ejecuta la política óptima π^* se deriva de la solución óptima del MIP. Cada acción se representa por su posición en el espacio de acciones:

Secuencia de acciones $\pi_* = (39, 18, 23, 45, 35, 38, 11, 13, 45, 8, 16, 45, 19, 15, 45, 5, 17, 45, 3, 12, 45)$

Se aprecia que la trayectoria óptima tiene 21 pasos. La recompensa acumulada por la política óptima π^* en el entorno diseñado es $\Omega = -1.538.473USD$ y difiere del valor observado en el MIP debido a las recompensas asociadas a transiciones exitosas entre periodos.

La secuencia podría variar en orden para las acciones de corte. En estricto rigor la mejor forma de elegir las acciones en el ambiente es:

1. En primer lugar las acciones de construcción de caminos, que a su vez dejan disponible mayor cantidad de rodales.
2. En segundo lugar las acciones de cosecha, donde el orden de ejecución de cosecha podría intercambiarse siempre que escojan después de las acciones de construcción de caminos.
3. Y en tercer lugar está la acción 45 para transitar al siguiente periodo.

3.4.6. Buenas prácticas de implementación del entorno de aprendizaje

Este entorno se implementa mediante métodos y sigue el modelo propuesto por Open AI en su librería de entornos **gym**. Según esos entornos:

- Debe existir un método para seleccionar acciones cuya respuesta sea una señal que incluya la información del estado al que llegó, la acción que escogió, la recompensa que recibió y una señal binaria que indica si el episodio finalizó. Este método es popularmente denominado `step(·)`.
- Debe existir un método que se accione cada vez que el episodio inicia y cuya respuesta sea el estado inicial del bosque. Este método es popularmente denominado `reset(·)` y en este caso **siempre regresa al mismo** s_0**.**
- Deben definirse las dimensiones del estado del entorno y también las dimensiones del espacio de acciones. Ambas dimensiones deben ser de tamaño fijo. Típicamente estos términos se definen respectivamente como `observation_space(·)` y `action_space(·)`.

3.5. Diseño de agente DQN

Los agentes constituyen el núcleo instrumental para la interacción con el entorno. Estos agentes asimilan señales del ambiente y, con base en ellas, adaptan su estrategia de toma de decisiones con el objetivo de optimizar la suma acumulativa de recompensas a lo largo de episodios sucesivos. La métrica de rendimiento de estos agentes se establece comparando la calidad de su política aprendida con un estándar de excelencia, a representado por la política óptima π^*. Concretamente, el criterio de evaluación será eficiencia con la que los agentes logran satisfacer la demanda de volumen de madera en intervalos temporales específicos, en relación con una política de referencia π^*.

3.5.1. Redes Neuronales e implementación de optimización

Dos redes neuronales se utilizan para aproximar la función $Q(s, a; w) = w \cdot \phi(s, a)$. Ambas redes se basan en la misma arquitectura de red densa (fully-connected): una red principal Q y una red objetivo Q_{target}.

Entonces w se aproxima como:

$$w \leftarrow w - \alpha \nabla_w \mathcal{L}(w)$$

Y q_{target} es:

$$q_{\text{target}} = r + \gamma \max_{a'} q(s', a'; w_{\text{target}}) \times (1 - \text{done})$$

Mediante Soft Update se actualizan los pesos de la red objetivo con una frecuencia de $\mathcal{N}$ pasos:

$$w_{\text{target}} \leftarrow \tau w + (1 - \tau) w_{\text{target}}$$

donde τ es el factor de actualización.

La implementación del proceso de optimización utiliza Adam, que es un acrónimo de "Adaptive Moment Estimation", combina las ventajas de dos otros métodos de optimización populares: RMSProp (Root Mean Square Propagation) y Momentum. Las 3 principales motivaciones de la elección son:

- **Convergencia rápida**: gracias a la acumulación del momento del gradiente (momentum), Adam suele converger rápidamente. Esto es especialmente útil en entornos de RL, donde el objetivo es entrenar un agente en el menor tiempo posible.
- **Eficacia computacional**: Adam es computacionalmente eficiente y requiere poca memoria, lo cual es crítico en aplicaciones en tiempo real o cuando se dispone de recursos limitados.
- **Amigable para el usuario**: aunque ofrece una gran cantidad de hiperparámetros que se pueden ajustar, los valores predeterminados de Adam suelen funcionar bastante bien en una amplia variedad de problemas. En este caso se ajustan la tasa de aprendizaje α y la tasa de descuento γ.

La arquitectura de la red que comparten las redes principal y objetivo es la siguiente:

$$\text{Capas: Entrada} \xrightarrow{\text{ReLU}} \text{Oculta}_1 \xrightarrow{\text{ReLU}} \cdots \xrightarrow{\text{ReLU}} \text{Oculta}_7 \xrightarrow{\text{ReLU}} \text{Salida}$$

- Una capa de entrada con tantas neuronas como el tamaño del estado.
- Una capa oculta con tantas neuronas como el tamaño del espacio de acción.
- Cinco capas ocultas sucesivas, cada una con 25 neuronas.
- Una capa oculta con tantas neuronas como el tamaño del espacio de acción.
- Una capa de salida con tantas neuronas como el tamaño del espacio de acción.

ReLU se utilizan después de cada capa oculta como función de activación.

3.5.2. Política de toma de decisiones

Se adopta una política ϵ-greedy, donde ϵ disminuye con una tasa constante hasta un valor mínimo de epsilon ϵ_{min}. La acción se selecciona de acuerdo con:

$$a_t = \begin{cases} \text{argmax}_{a \in A(s_t)} q(s_t, a; w) & \text{con probabilidad } 1 - \epsilon \\ \text{acción aleatoria} & \text{con probabilidad } \epsilon \end{cases}$$

La disminución de ϵ ocurre con cada finalización de un episodio y está descrita por:

$$\epsilon = max(\epsilon_{min}, \epsilon - \Delta\epsilon)$$

3.5.2.1. Función de pérdida

En lugar de utilizar la pérdida de error cuadrático medio (MSE), se opta por la función de pérdida Huber, más robusta a los valores atípicos.

$$\mathcal{L}(w) = \mathbb{E}_{(s,a,r,s',\text{done})\sim\mathcal{D}}\left[(Huber(q_{\text{target}} - q(s,a;w))^2)\right]$$

$$Huber(\delta) = \begin{cases} \frac{\delta^2}{2} & \text{para } |\delta| \leq 1 \\ |\delta| - \frac{1}{2} & \text{de lo contrario} \end{cases}$$

donde $\delta = r + \gamma \max_{a'} q(s',a') - q(s,a)$. Por ende el cómputo del gradiente es:

$$\nabla_w \mathcal{L}(w) = \begin{cases} -\delta\phi(s,a) & \text{para } |\delta| \leq 1, \\ -\text{sign}(\delta)\phi(s,a) & \text{para } |\delta| > 1. \end{cases}$$

3.5.3. Parámetros

Para implementar un agente DQN con *experience replay*, *política ϵ-greedy* y una red objetivo para actualizar la política frecuentemente, el agente recibe como parámetros:

- **Tamaño del estado**. Que debe ser fijo y bien definido para todas las entradas del vector de estados.
- **Tamaño del espacio de acciones**. También es fijo y bien definido.
- Capacidad de almacenamiento de episodios de $\mathcal{D}$ de donde se samplea cada volumen de datos.
- Volumen del lote de datos (batch size) utilizado para experience replay.
- Epsilon inicial ϵ_{max} y final ϵ_{min} para la política ϵ-greedy.
- Tasa de decaimiento de epsilon Δ en política ϵ-greedy.
- Frecuencia $\mathcal{N}$ de pasos para actualizar los pesos de la red objetivo mediante soft update.
- τ, que es es el factor de actualización en el soft update de la red objetivo q_{target}.

3.5.4. Implementación mediante métodos

La eficiencia del agente DQN reside en la interacción y complementariedad de sus métodos, que trabajan en conjunto para resolver problemas de aprendizaje por refuerzo. A continuación se describen estas interacciones:

- **choose_action**: este método decide qué acción tomar en base al estado s_t en que se encuentra el agente mediante la máscara de acciones $\mathcal{M}$. Utiliza la red principal para obtener los valores Q del estado actual. Dependiendo del valor de ϵ, elige una explotar o explorar el ambiente. Este método es invocado en cada paso del episodio para tomar una decisión.

- **step**: este método se invoca después de que el agente ha tomado una acción y ha observado la recompensa y el nuevo estado. Aquí se almacenan las transiciones (s, a, r, s') en el buffer de experiencia. Si el buffer está lleno de datos, se invoca el método `_learn` para entrenar la red principal.

- **_learn**: este método es el núcleo del aprendizaje. Utiliza un mini-batch de experiencias del buffer para actualizar los pesos de la red neuronal Q. Este aprendizaje se realiza mediante el optimizador Adam. El método `_learn` también invoca el método `_update_target_net` para actualizar los pesos de la red objetivo con una cierta frecuencia.

- **_update_target_net**: este método actualiza con una frecuencia de $\mathcal{N}$ pasos los pesos de la red objetivo utilizando soft update. Esto es crucial para la estabilidad del aprendizaje.

- **_sample_memory**: este método auxiliar es invocado dentro del método `_learn` para muestrear experiencias del buffer de memoria $(s, a, r, s', \text{done}) \sim \mathcal{D}$. Los datos muestreados son usados para calcular la pérdida y actualizar los pesos de la red principal.

En resumen, el método `choose_action` permite al agente interactuar con el entorno, el método `step` almacena estas interacciones, y el método `_learn` utiliza estas interacciones almacenadas para mejorar la política del agente. Los métodos `_update_target_net` y `_sample_memory` son operaciones auxiliares que facilitan el proceso de aprendizaje. Estos métodos se resumen en la Figura 3.12:

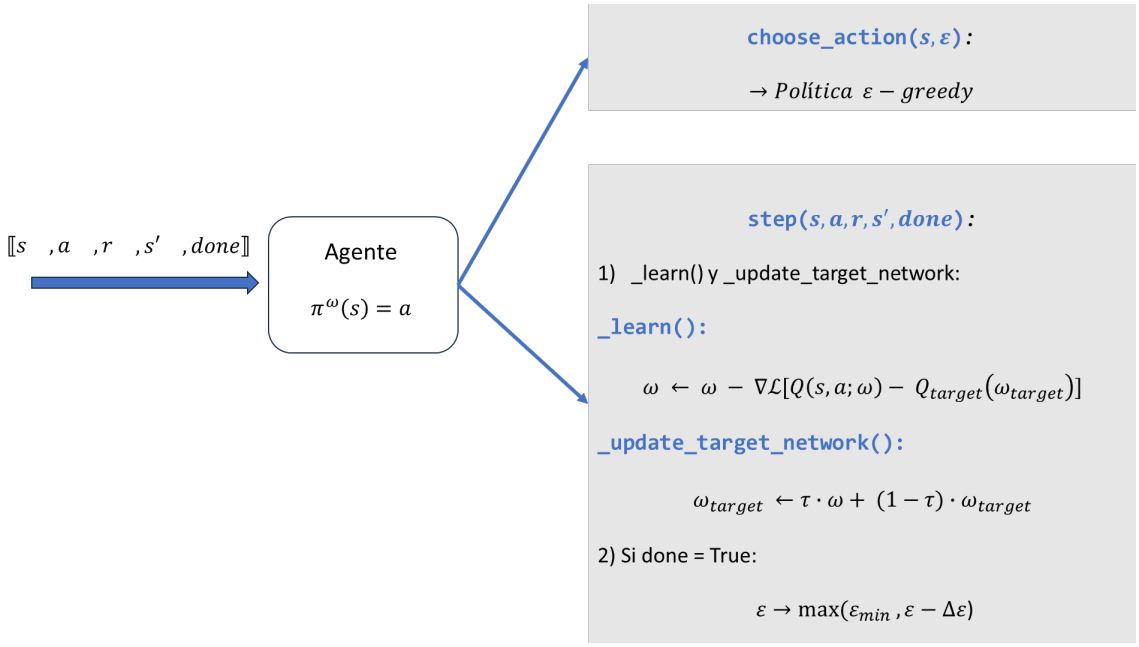

Figura 3.12: Métodos del agente DQN implentado para seleccionar acciones y actualizar su política de toma de decisiones

3.6. Implementación entorno-agente

La implementación de agente descrita mediante métodos se relaciona con los métodos de implementación del entorno en un algoritmo iterativo que hace transitar al agente por el

entorno en varios episodios, y en en cada episodio da una cantidad de pasos para finalizar el episodio de forma exitosa o fallida. Globalmente se definen `learning_step_counter`, un contador de pasos que sirve como referencia para actualizar mediante soft update y también se define $\epsilon_{max} = 1$ para comenzar el aprendizaje.

El elemento "`Entorno`" corresponde a una instancia del entorno diseñado y el elemento "`Agente`" corresponde a una instancia de agente DQN diseñado. Para generar el ciclo de aprendizaje definido por las señales de estados y recompensas del marco teórico de RL se utilizan los métodos de Entorno y Agente.

Se debe definir la cantidad de episodios que se entrenará al agente. Cada episodio se inicia Entorno en el estado s_0 invocando su método `reset()`. A continuación el estado es normalizado mediante `state_to_array()`, un método auxiliar creado exclusivamente para recibir y normalizar el estado de Entorno.

Luego, cada paso de toma de decisiones en el episodio sigue el ciclo:

1. Se selecciona una acción en función del estado normalizado y el valor de ϵ que recibe mediante el método `choose_action()` de Agente.
2. A continuación Entorno realiza la transición de estado y retorna una señal compuesta por el siguiente estado, la recompensa y la variable done mediante su método `step()`.
3. Luego el siguiente estado es normalizado mediante `state_to_array()`.
4. Después, Agente ejecuta el método `step()`, que desencadena una serie de eventos internos cruciales para el aprendizaje y la actualización del modelo.
5. Antes de cerrar el ciclo de aprendizaje se sobreescribe el elemento que representa al estado actual en que se encuentra el agente con la información del último estado al que se transitó.
6. Finalmente si la variable done indica que el episodio finalizó, entonces se vuelve a ejecutar el método `reset()` de Entorno para iniciar un nuevo episodio. Por otro lado, si el episodio aún no finaliza, entonces se vuelve a seleccionar una acción en función del estado, es decir, volver al inicio del ciclo.

Cada vez que el agente DQN ejecuta una acción utilizando el método `step()`, se desencadena una serie de eventos internos cruciales para el aprendizaje y la actualización del modelo. A continuación se describen estos eventos en detalle:

1. Almacenamiento de la Transición en el Replay Buffer: inmediatamente después de ejecutar la acción, el agente guarda la transición compuesta por el estado actual, la acción realizada, la recompensa obtenida, el nuevo estado y la variable `done`, en su Replay Buffer para futuras referencias.
2. Disminución del Valor de ϵ: si la variable `done` se establece en `True`, lo que generalmente significa que se ha llegado al final del episodio, el agente disminuye el valor de ϵ conforme a su estrategia de exploración ϵ-greedy.
3. Aprendizaje y Actualización:

- Aprendizaje: si el Replay Buffer ha alcanzado su capacidad mínima, el agente invoca el método `Agente._learn()` para llevar a cabo el aprendizaje.
- Actualización de la red neuronal principal: a continuación, el agente invoca el método `Agente._sample_memory()` para muestrear un lote de transiciones del Replay Buffer. Utiliza este lote para actualizar los parámetros de su red neuronal principal.
- Luego dentro del método `Agente._learn()` se calcula la función de pérdida y actualiza los parámetros de la red utilizando el algoritmo de optimización Adam.
- Contador de Pasos de Aprendizaje: tras cada actualización, el contador interno de pasos de aprendizaje (`learning_step_counter`) se incrementa en uno.
- Actualización de la Red Objetivo: si `learning_step_counter` es divisible exactamente por la frecuencia de actualización $\mathcal{N}$, el agente actualiza los parámetros de su red objetivo utilizando el método `Agente._update_target_network()`.

Estos eventos aseguran que el agente aprende de manera efectiva a partir de sus experiencias pasadas y actuales, adaptándose para mejorar su rendimiento en episodios futuros.

La Figura 3.13 resume el proceso iterativo que se da, en cada episodio y cada paso del episodio, entre los métodos de Entorno y Agente para llevar a cabo el aprendizaje basado en DRL. En casillas negras se encuentran los métodos aplicados, en casillas blancas breves descripciones de los métodos asociados y en casillas grises las condiciones para transitar entre métodos. Por su parte, las flechas azules unen el recorrido principal del ciclo de aprendizaje descrito que comienza con la aplicación del método `reset()` mientras que las flechas de color rojo indican el orden de ejecución de métodos que resulta en la actualización de Agente al ejecutar su método `step()`.

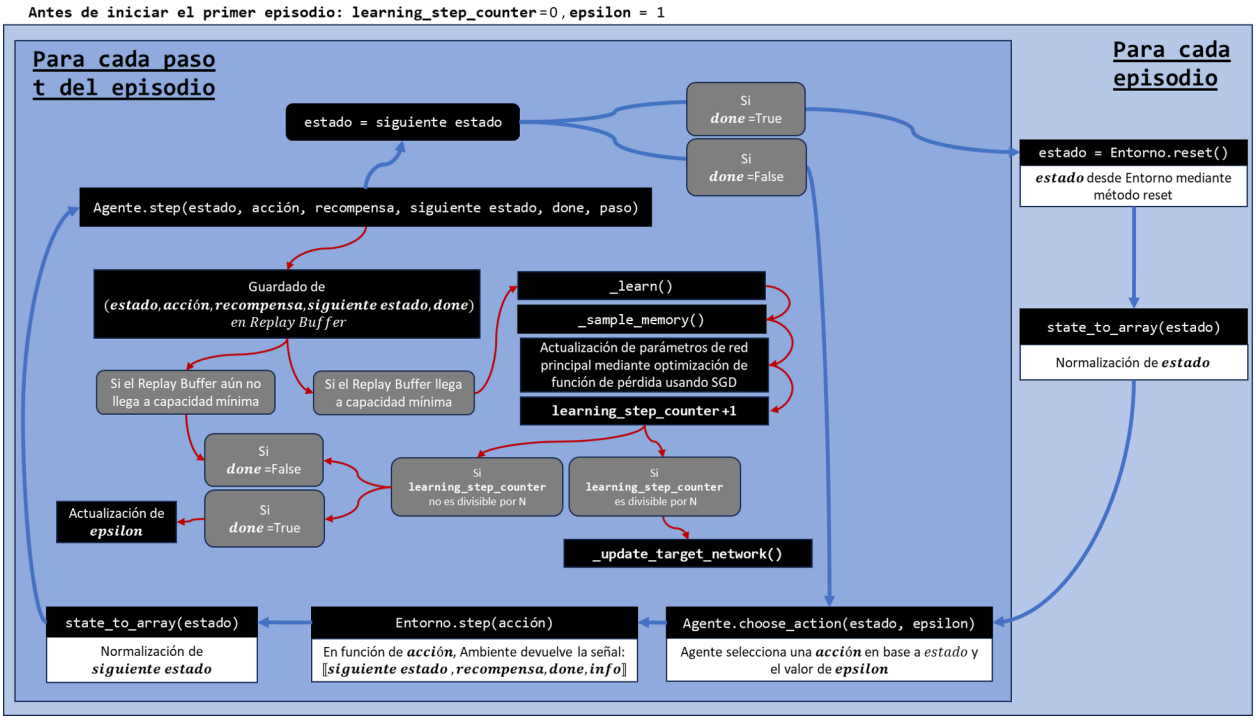

Figura 3.13: Proceso iterativo de métodos de entorno y agente para generar aprendizaje de DRL.

Capítulo 4

Resultados

Los resultados están enfocados en la recompensa acumulada por los agentes a lo largo de su entrenamiento y su testeo. Se presentan las gráficas de:

1. Recompensas acumuladas por el agente conforme avanza en realización de episodios y disminuye el valor de ϵ.
2. Promedio móvil de la recompensa acumulada.
3. Largo de episodios. Medido en pasos por episodio.
4. Valor promedio de $q(s, a; w)$ cada $\mathcal{N}$ pasos.
5. Valor de la función de pérdida $\mathcal{L}$ cada $\mathcal{N}$ pasos.
6. Etapa de prueba del aprendizaje. En esta etapa $\epsilon = 0$, forzando al agente a seleccionar siempre la acción que cree que es la mejor basándose en su conocimiento actual, es decir, se desactiva la exploración y el agente solo explotará su conocimiento.

Todos los resultados se encuentran en la sección de Anexo C y pueden reproducirse con la semilla de datos `seed_value = 17`.

4.1. Parámetros de caso base

Los parámetros se ajustan para entrenamientos de 1000 episodios:

- Tamaño del estado: 49
- Tamaño del espacio de acciones: 45
- Factor de descuento γ : 0.7
- Tasa de aprendizaje: 0.004
- Tamaño del buffer de memoria: 512
- Tamaño de batch para el aprendizaje: 128
- ϵ_{max}: 1

- ϵ_{min}: 0
- Δ: 0.995
- Frecuencia $\mathcal{N}$ de actualización de los pesos de la red objetivo: 33. **Que es una estimación de duración promedio de episodio en base a la observación del entorno en diferentes trayectorias de prueba.**
- τ: 0.2

4.2. Configuraciones experimentales del Agente DQN

4.2.1. Variaciones en la arquitectura de la red neuronal

Se plantean ocho configuraciones de arquitectura basadas en un caso base. La arquitectura base tiene 25 neuronas y 7 capas ocultas. Las variaciones son las siguientes:

1. Caso Base: 25 neuronas, 7 capas ocultas, función de activación ReLU. (Configuración A)
2. Menos neuronas: 15 neuronas, 7 capas ocultas. (Configuración B)
3. Más capas ocultas: 25 neuronas, 10 capas ocultas. (Configuración C)
4. Más neuronas: 35 neuronas, 7 capas ocultas. (Configuración D)
5. Menos capas ocultas: 25 neuronas, 4 capas ocultas. (Configuración E)
6. Más capas ocultas y más neuronas: 35 neuronas, 10 capas ocultas. (Configuración F)
7. Menos capas Ocultas y menos neuronas: 15 neuronas, 4 capas ocultas. (Configuración G)
8. Más capas ocultas y menos neuronas: 15 neuronas, 10 capas ocultas (Configuración H)
9. Menos capas ocultas y más neuronas: 35 neuronas, 4 capas ocultas (Configuración I)

Todas las configuraciones, excepto la Configuración H, mostraron un descenso en la función de pérdida $\mathcal{L}$ y aproximaron de forma coherente los valores q en términos de magnitud y signo. Además, todas las configuraciones convergieron en la duración de los episodios a un rango entre 33 y 35 pasos. Se seleccionó la Configuración F como la mejor debido a su capacidad para alcanzar niveles superiores de recompensa acumulada, superando el umbral de -3000000, nivel que las otras configuraciones, excepto la configuración C, no pudieron alcanzar.

4.2.1.1. Variaciones en las funciones de activación

Se prueban las siguientes funciones de activación intercaladas con ReLU para la mejor de las configuraciones de arquitectura anteriores (configuración F):

- Tangente Hiperbólica (Configuración J)
- Sigmoide (Configuración K)

Posteriormente, se experimentó con diferentes combinaciones de funciones de activación utilizando la arquitectura de la Configuración F. La Configuración J intercalaba funciones de activación ReLU y tangente hiperbólica, mientras que la Configuración K intercalaba ReLU y sigmoide. Ambas configuraciones mostraron una disminución con menor varianza en la función de valor $\mathcal{L}$, y los valores q estimados fueron coherentes en magnitud y signo. Sin embargo, ni la Configuración J ni la Configuración K superaron a la Configuración F en términos de nivel de recompensa acumulada, manteniendo a la Configuración F como la configuración óptima.

4.2.2. Variaciones en los parámetros del agente DQN

Para la configuración F, se prueban por separado las siguientes configuraciones de parámetros que contrastan con el caso base:

- γ: [$\gamma_1 = 0.3$, $\gamma_2 = 0.5$, $\gamma_3 = 0.9$] (Configuraciones L1, L2 y L3).

  Estas configuraciones no lograron un mejor desempeño que el parámetro $\gamma = 0.6$ del caso base. La configuración $L3$ obtuvo peores resultados en comparación a $L2$ y $L1$. Los valor Q promedio de $L1$ son estables y el valor de su función de perdida tiene un comportamiento que converge hacia cero en comparación a $L2$ que presenta un aumento en la función de perdida hacia el final del entrenamiento. $L3$ muestra una función de pérdida que no converge hacia el final del entrenamiento. Las recompensas acumuladas que logran estas configuraciones se ordenan: $L1 > L2 > L3$

- α: [$\alpha_1 = 0.002$, $\alpha_2 = 0.007$, $\alpha_3 = 0.01$] (Configuraciones M1, M2 y M3).

  Estas configuraciones no lograron mejor desempeño que el parámetro $\alpha = 0.004$ del caso base para aprender una política óptima con recompensas acumuladas mayores a -3000000. Las recompensas acumuladas que logran estas configuraciones se ordena: $M3 > M2 > M1$. Las configuraciones $M1, M2, M3$ resultaron con curvas de aprendizajes muy parecidas y los valores q promedio tienen magnitudes y signo coherente con las recompensas que recibe el agente (generalmente perdidas). El comportamiento de la función de perdida de las tres configuraciones demuestra una tendencia a converger. Sin embargo, $\alpha = 0.002$ de configuración $M1$ presenta menos variabilidad en su convergencia y las configuraciones de F, $M2$, $M3$ presentan mayor variabilidad, aunque esto no llega a afectar las estimaciones de los valores Q.

- τ para Soft Update: [$\tau_1 = 0.3$, $\tau_2 = 0.5$, $\tau_3 = 0.7$] (Configuraciones N1, N2 y N3).

  Estas configuraciones no lograron mejor desempeño que el parámetro $\tau = 0.2$ del caso base para aprender una política óptima con recompensas acumuladas mayores a -3000000. Las recompensas acumuladas que logran estas configuraciones se ordena: $N3 > N1 > N2$. La función de perdida de $N2$ parece haber tenido un desaprendizaje casi a la mitad del entrenamiento, lo que generó una estimación de los q valores promedio imprecisa, tal como lo muestra su función de perdida, que alcanza un pico en su valor casi a la mitad del entrenamiento. La función de perdida de $N3$ también posee un pico en su valor en la mitad del entrenamiento, pero logra estabilizar las estimaciones

de los valores q. Por su parte de la configuración $N1$ se aprecia una convergencia con poca variabilidad en la función de perdida.

- Tamaño de batch de aprendizaje: [$\text{batch}_1 = 32$, $\text{batch}_2 = 64$, $\text{batch}_3 = 256$] (Configuraciones Ñ1, Ñ2 y Ñ3)

  Estas configuraciones no lograron mejor desempeño que el parámetro `tamaño de batch = 128` del caso base para aprender una política óptima con recompensas acumuladas mayores a -3000000.Las recompensas acumuladas que logran estas configuraciones se ordena: Ñ1 > Ñ3 > Ñ2. Se aprecia claramente que la variación en la estimación de Ñ1 es la más estable, ya que Ñ2 experimenta mayor variación en la estimación de valores q promedio hacia el final del entrenamiento y Ñ3 presenta una leve sobre estimación en algunos mustreos de los valores q promedio. La variación en la estimación de la configuración Ñ2 se refleja en la función de perdida que no logra converger hacia el final del entrenamiento, mientras que las funciones de perdida de Ñ1yÑ3 convergen respectivamente con mayor y menor variabilidad en la estimación.

- Cantidad de episodios: [$\text{Episodios}_1 = 2000$, $\text{Episodios}_2 = 5000$, $\text{Episodios}_3 = 10000$] (Configuraciones O1, O2 y O3).

  Este es el parámetro que logró hallar políticas que superaran el desempeño de la configuración F. Con la realización de $O1$ se aprecia una mayor recompensa acumulada, aunque la recompensas obtenidos por el agente no parecieran haber explorado episodios con este nivel de recompensa.

Capítulo 5

Discusión

La aplicación de DRL en la optimización de estrategias de corte forestal ha planteado diversos desafíos y reflexiones críticas en este estudio. La formulación de un modelo que incorpora un agente DQN ha revelado *insights* en cuanto a la adaptabilidad y potencial de la metodología DRL en campos interdisciplinarios, como la gestión forestal sostenible.

5.1. Formulación del estado y representación

Uno de los aspectos más destacados de este trabajo es la conceptualización del estado. Se sugiere que la formulación del estado podría beneficiarse de una representación más concisa, omitiendo los estados de los caminos y gestionándolos internamente en el entorno. Esto no solo reduce la dimensionalidad del espacio de estados, sino que también podría mitigar problemas relacionados con la maldición de la dimensionalidad, un punto crucial en la investigación de DRL. Una representación de estado más eficiente y efectiva podría significar mejor convergencia y generalización del agente, resultando en políticas más robustas y adaptativas.

5.2. Reinicialización del entorno

El método `reset()` ha introducido un desafío significativo, reiniciando el entorno a su estado inicial en cada invocación. Esto, en conjunción con los niveles de dificultad inherentes en el aprendizaje del agente desde un conocimiento nulo de las consecuencias de sus acciones, subraya la necesidad de un análisis profundo en el diseño de entornos y la adaptabilidad del agente.

5.3. Niveles de dificultad en el aprendizaje

El proceso de aprendizaje del agente se enmarcó en distintos niveles de dificultad:

1. Adquisición de conocimiento desde un estado de ignorancia inicial.
2. Transición óptima entre periodos post-satisfacción de la demanda.
3. Orden lógico en la toma de decisiones: rodales, caminos y cambio de periodo.
4. Exploración adecuada del espacio de estados previo a una explotación predominante.

Cada nivel requirió del agente un entendimiento progresivo y adaptativo del entorno, promoviendo una comprensión más profunda del equilibrio entre exploración y explotación, crucial en el desarrollo de políticas efectivas en DRL.

5.4. Exploración y convergencia

Pese a que la convergencia está garantizada teóricamente, el equilibrio entre exploración y explotación es crucial en el aprendizaje por refuerzo profundo (DRL). El valor de epsilon debe asegurar una adecuada convergencia del agente y su capacidad para deducir estrategias óptimas de manejo forestal sostenible.

Por otro lado la aproximación de funciones mediante redes neuronales en el algoritmo DQN es vital para entender la dinámica de aprendizaje del agente en la estimación de valores q para pares de estado-acción que no se han explorados previamente.

Las redes neuronales utilizadas en DQN permiten la aproximación de funciones que es fundamental para tratar con la alta dimensionalidad y complejidad en los espacios de estados y acciones presentes en problemas de gestión forestal. La efectividad de la convergencia del agente está intrínsecamente ligada a la precisión de estas aproximaciones. Un entendimiento riguroso de estas dinámicas es crucial para interpretar cómo las imprecisiones en la aproximación de funciones pueden afectar la calidad de la política aprendida.

Dada la extensión del espacio de estados y acciones, una exploración exhaustiva es computacionalmente intratable, resaltando la importancia de estrategias de exploración y técnicas de aproximación efectivas. Los errores de aproximación y una exploración insuficiente pueden obstaculizar el desarrollo de políticas robustas y coherentes, poniendo de relieve la necesidad de investigaciones futuras centradas en optimizar estos aspectos en el contexto de la gestión forestal sostenible.

La identificación y mitigación de las limitaciones asociadas con el uso de redes neuronales para aproximaciones de funciones y la exploración en entornos de manejo forestal avanzan el campo de DRL y sugieren rutas prometedoras para la implementación de estas tecnologías en el ámbito de la sostenibilidad ambiental.

Capítulo 6

Conclusiones

En la presente investigación, se ha explorado la aplicación de deep reinforcement learning en el contexto de la gestión forestal, con un enfoque especial en el uso de agentes DQN para abordar problemas de decisiones secuenciales. Se ha implementado una metodología de diseño de entorno y agentes personalizados, buscando evaluar su capacidad de aprendizaje desde un estado de conocimiento nulo.

Los resultados del estudio indican que, cuando se entrenan durante una cantidad suficiente de episodios, los agentes son capaces de aprender políticas efectivas que les permiten acumular recompensas significativas. Esto se manifiesta en una exploración más exhaustiva de escenarios de estado y una probable aproximación hacia la política óptima. Este hallazgo pone de manifiesto la importancia crítica de la duración del entrenamiento en el proceso de aprendizaje de los agentes DQN.

Se ha descubierto también que el diseño del entorno, en términos de definición del estado, tiene una importancia equivalente, si no superior, a la calibración de los agentes de aprendizaje. Para este estudio, se definieron 49 valores y 45 acciones para representar el estado, resaltando la adaptabilidad y flexibilidad del DRL. Sin embargo, el estudio sugiere la posibilidad de explorar diferentes conjuntos de valores y acciones para la representación del estado.

La función de pérdida implicada es de una complejidad notable, constituyendo un reto significativo en la aplicación de la metodología de DRL. Se observó que, en varias configuraciones de agentes DQN, el valor muestreado tiende a converger a mínimos locales, implicando que la política aprendida no es óptima. Este fenómeno subraya la necesidad de desarrollar estrategias de optimización y aproximación avanzadas para superar las dificultades intrínsecas en la función de pérdida en escenarios de DRL.

Respecto a la arquitectura de la red neuronal, se concluye que las arquitecturas más grandes, con un mayor número de capas ocultas y neuronas, mostraron un rendimiento superior en el aprendizaje para la recolección de recompensas. Sin embargo, este mejor rendimiento se logra principalmente al extender la duración del entrenamiento, corroborando la importancia del tiempo de entrenamiento en el aprendizaje efectivo de DQN.

En conclusión, a pesar de los desafíos y complejidades presentados, los agentes diseñados en este estudio son prometedores y demuestran potencial para alcanzar políticas óptimas.

Esto valida el potencial de los métodos DQN como herramientas viables y efectivas para abordar problemas de decisiones secuenciales en el ámbito de la gestión forestal, resaltando los desafíos y oportunidades inherentes a la implementación de técnicas de DRL en este y otros campos. Los hallazgos de esta investigación sugieren un futuro prometedor para la aplicación e investigación de DRL en problemas complejos de toma de decisiones, estableciendo un precedente para futuras investigaciones en el campo.

Bibliografía

[1] Bettinger, P., "Forest management and planning 2nd edition," 2016.

[2] Rönnqvist, M., "Operations research challenges in forestry: 33 open problems," 2015.

[3] INFOR, "Anuario 2021 infor," 2022.

[4] FAO, "Porcentaje de las exportaciones mundiales de pulpa de madera.".

[5] Lindenmayer, D. *et al.*, "Value of long-term ecological studies," Austral Ecology, vol. 37, no. 7, pp. 745–757, 2012.

[6] Bradshaw, C. J. *et al.*, "Global evidence that deforestation amplifies flood risk and severity in the developing world," Global Change Biology, vol. 13, no. 11, pp. 2379–2395, 2007.

[7] Gayoso, J., A. D., "Guía de conservación de suelos forestales," 1999.

[8] D'Amours, S., R. M. . W. A., "Using operational research for supply chain planning in the forest product industry," 2008.

[9] Quinteros, M., "Una aplicación de programación estocástica en un problema de gestión forestal," 2005.

[10] Bettinger, Boston, K., "Spatial forest plan development using heuristic," Forest Science, vol. 63, pp. 518–528, 2017.

[11] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., y et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[12] Henderson, P., "Deep reinforcement learning that matters," 2018.

[13] Badilla, F., "Stochastic-optimization-models-in-forest-planning," 2014.

[14] Fernández, N., "OptimizaciÓn estocÁstica bajo incertidumbre de un modelo de planificaciÓn forestal integrado usando estrategias de decomposiciÓn," 2020.

[15] Troncoso, J., "A mixed integer programming to evaluate integrate strategies in the forest value chain - a case study the chilean forest industry," 2015.

[16] Silver, D., "Lectures on reinforcement learning," 2015.

[17] Sutton, R. S., . B. A. G., "Reinforcement learning: An introduction," 2018.

[18] Watkins, C. J. C. H., Learning from Delayed Rewards. PhD thesis, University of Cambridge, England, 1989.

[19] Goodfellow, I., Bengio, Y., y Courville, A., Deep Learning. MIT Press, 2016.

[20] LeCun, Y., Bengio, Y., y Hinton, G., "Deep learning," Nature, vol. 521, pp. 436–444,

2015.

[21] Rumelhart, D. E., Hinton, G. E., y Williams, R. J., "Learning representations by back-propagating errors," Nature, vol. 323, pp. 533–536, 1986.

[22] McCulloch, W. S. y Pitts, W., "A logical calculus of the ideas immanent in nervous activity," Bulletin of Mathematical Biology, vol. 5, pp. 115–133, 1943.

[23] Krizhevsky, A., Sutskever, I., y Hinton, G. E., "Imagenet classification with deep convolutional neural networks," en Advances in Neural Information Processing Systems, pp. 1097–1105, 2012.

[24] Zhang, S., "Constructing deep sparse coding network for image classification," 2017.

[25] Hochreiter, S. y Schmidhuber, J., "Long short-term memory," Neural Computation, vol. 9, pp. 1735–1780, 1997.

[26] Doshi-Velez, F. y Kim, B., "Towards a rigorous science of interpretable machine learning," arXiv preprint arXiv:1702.08608, 2017.

[27] Haykin, S., "Neural networks: A comprehensive foundation," Prentice Hall, 1998.

[28] Schmidhuber, J., "Deep learning in neural networks: An overview," Neural Networks, vol. 61, p. 85–117, 2015.

[29] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., y Polosukhin, I., "Attention is all you need," en Advances in Neural Information Processing Systems, p. 5998–6008, 2017.

[30] Glorot, X., Bordes, A., y Bengio, Y., "Deep sparse rectifier neural networks," en Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 315–323, 2011.

[31] Nwankpa, C., Ijomah, W., Gachagan, A., y Marshall, S., "Activation functions: Comparison of trends in practice and research for deep learning," arXiv preprint arXiv:1811.03378, 2018.

Anexos

Anexo A. Teorema de Contracción

En el aprendizaje por refuerzo, el operador de Bellman juega un papel fundamental en la definición y solución de problemas de decisión secuencial. En este informe, demostramos que el operador de Bellman es una contracción en el espacio de funciones de valor bajo la norma infinito y, por lo tanto, garantiza la convergencia de los algoritmos de programación dinámica a la función de valor óptima. El operador de Bellman T es una contracción en el espacio de funciones de valor bajo la norma infinito.

Un operador T es una contracción con factor γ (donde $0 \leq \gamma < 1$) si para todas las funciones f y g:

$$||Tf - Tg||_\infty \leq \gamma ||f - g||_\infty$$

Demostración. Consideremos dos funciones de valor arbitrarias V_1 y V_2. Sin pérdida de generalidad, suponemos que V_1 es mayor que V_2 en algún estado.

Definimos la diferencia entre estas funciones de valor en la norma infinito como $\Delta = ||V_1 - V_2||_\infty$.

Para un estado s y la acción que maximiza el valor esperado bajo V_1, denotado por $a = \arg\max_a Q^{V_1}(s, a)$, tenemos:

$$(TV_1)(s) - (TV_2)(s) = Q^{V_1}(s, a) - Q^{V_2}(s, a)$$

Expandiendo esta diferencia, obtenemos:

$$Q^{V_1}(s, a) - Q^{V_2}(s, a) = \gamma \sum_{s'} P^a_{ss'}(V_1(s') - V_2(s')) \leq \gamma\Delta$$

Por lo tanto, el operador de Bellman T satisface la definición de una contracción con factor γ en la norma infinito. □

Anexo B. Convergencia a la Función de Valor Óptima

Dada la propiedad de contracción del operador de Bellman, podemos invocar el Teorema del Punto Fijo de Banach:

[Teorema del Punto Fijo de Banach] Si un operador es una contracción en un espacio métrico completo, entonces dicho operador tiene un único punto fijo en ese espacio y, comenzando desde cualquier punto en ese espacio, la sucesión de aplicaciones iteradas del operador convergerá a ese punto fijo.

Aplicando este teorema, concluimos que:

1. Existe una única función de valor óptima V^* que es un punto fijo del operador de

Bellman, es decir, $TV^* = V^*$.

2. Los algoritmos de programación dinámica convergerán a V^* desde cualquier inicialización.

La propiedad de contracción del operador de Bellman asegura la convergencia de los métodos de programación dinámica a la función de valor óptima en problemas de decisión secuencial, lo que hace de la programación dinámica una herramienta poderosa en el aprendizaje por refuerzo.

Anexo C. Resultados de todas las configuraciones de modelo DQN sobre entorno de RL diseñado

C.1. Recompensas acumuladas y disminución de ϵ

C.1.1. Configuración A

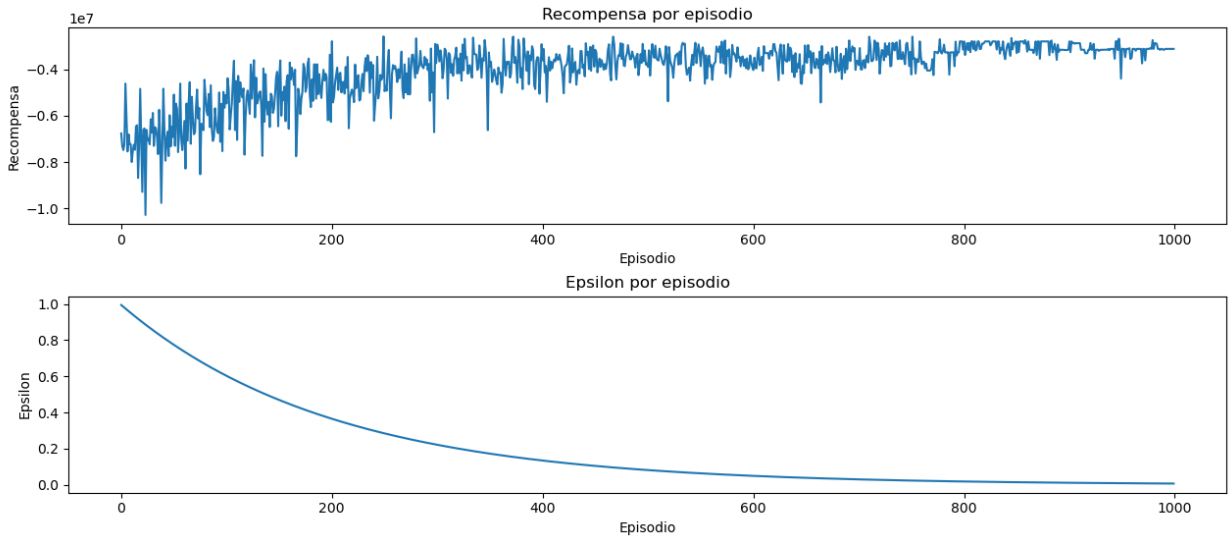

Figura C.1: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración A

C.1.2. Configuración B

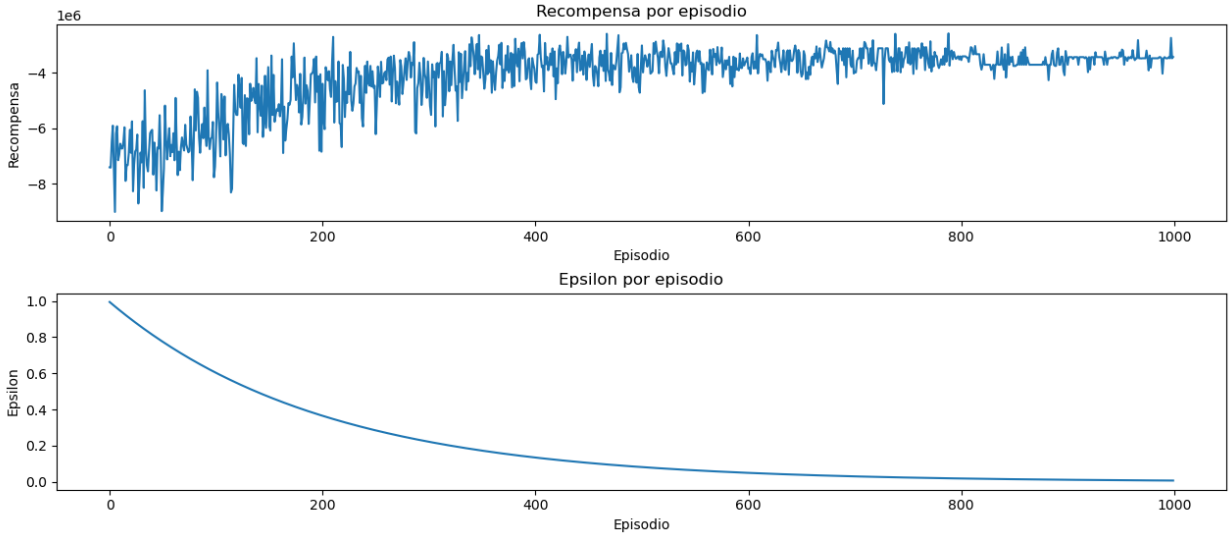

Figura C.2: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración B

C.1.3. Configuración C

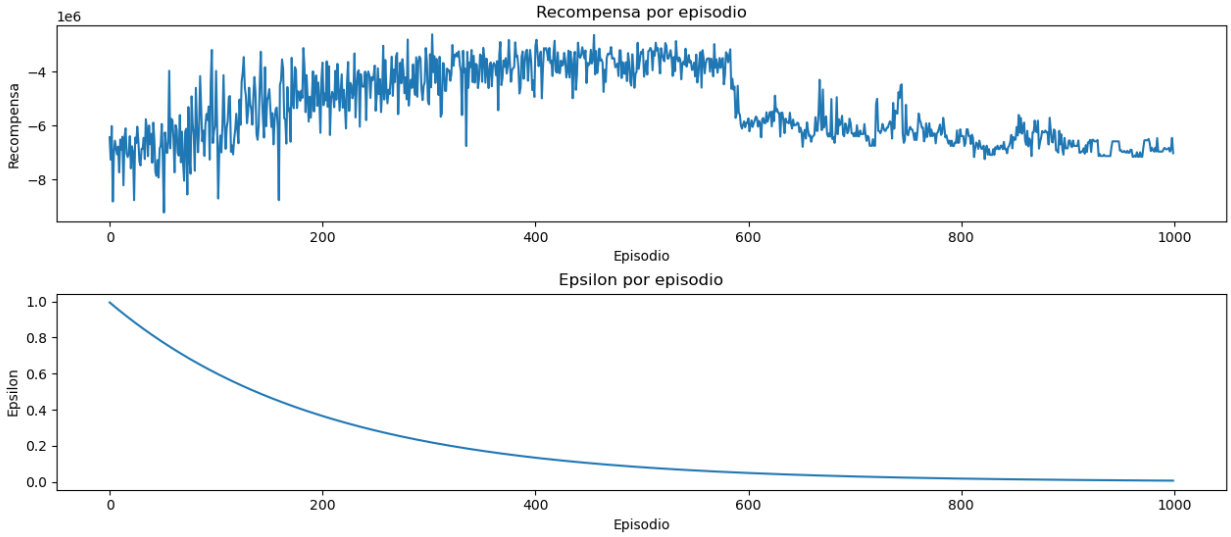

Figura C.3: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración C

C.1.4. Configuración D

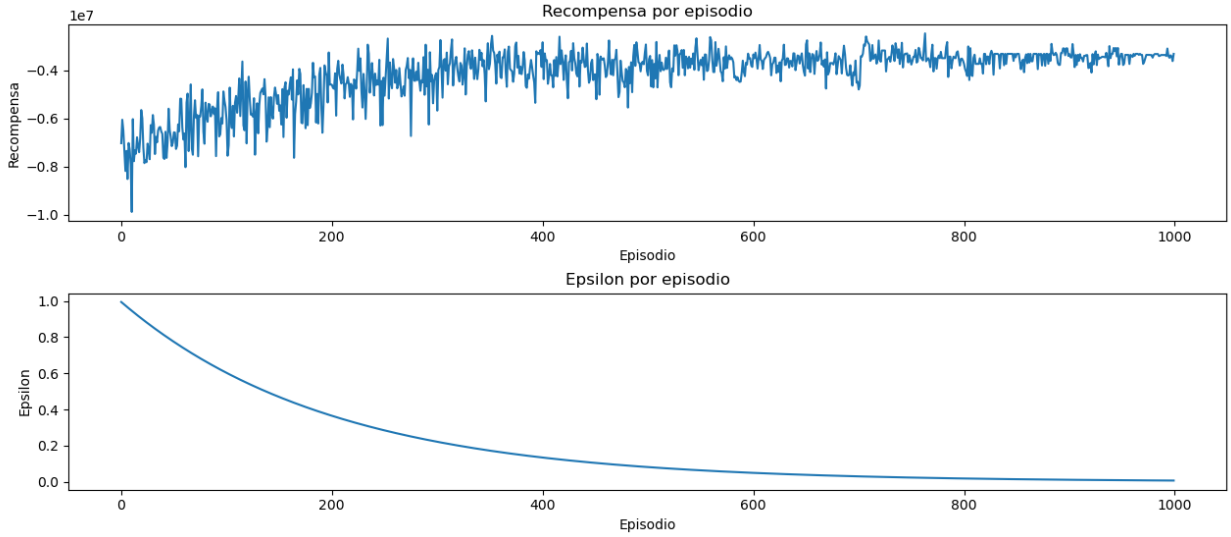

Figura C.4: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración D

C.1.5. Configuración E

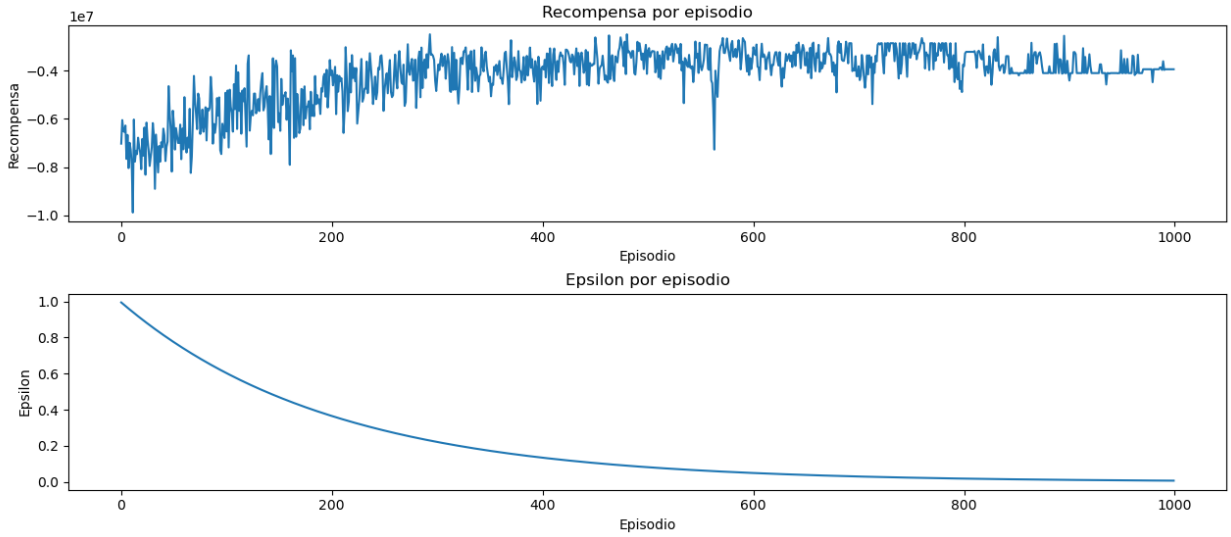

Figura C.5: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración E

C.1.6. Configuración F

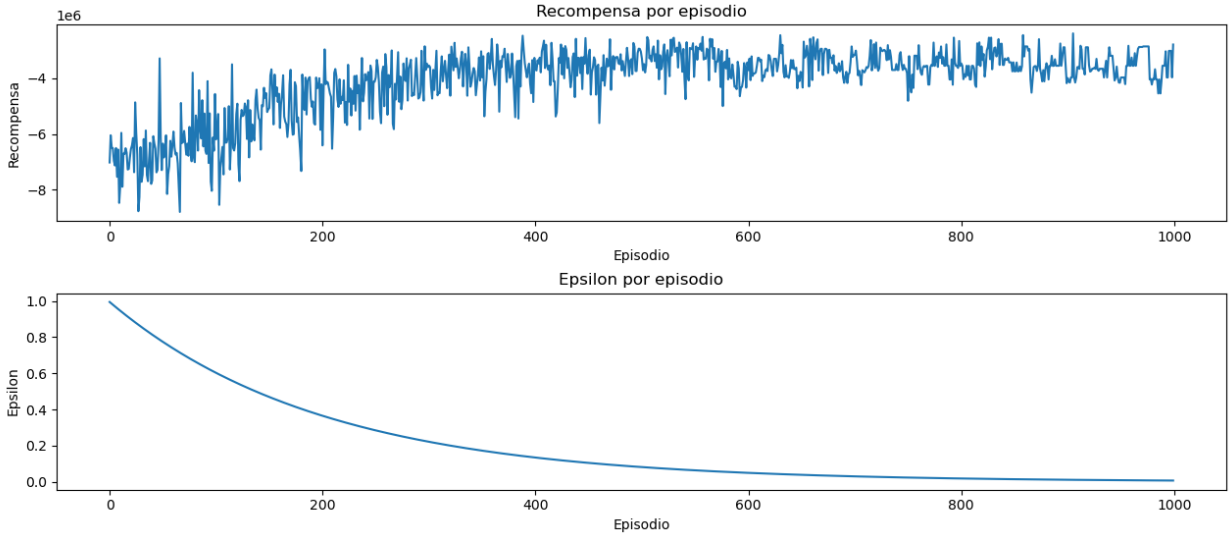

Figura C.6: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración F

C.1.7. Configuración G

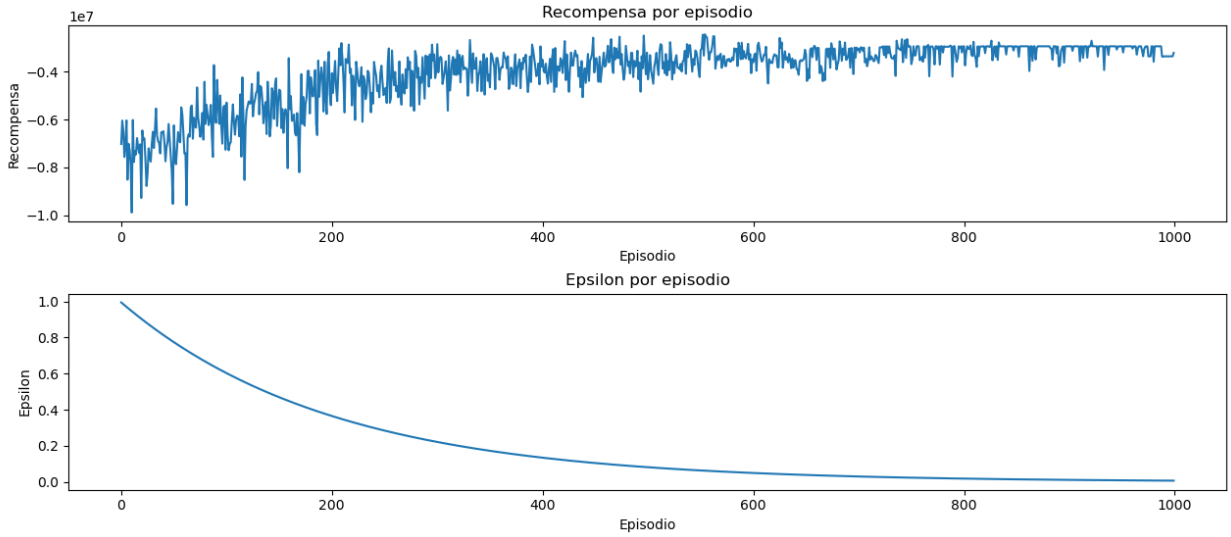

Figura C.7: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración G

C.1.8. Configuración H

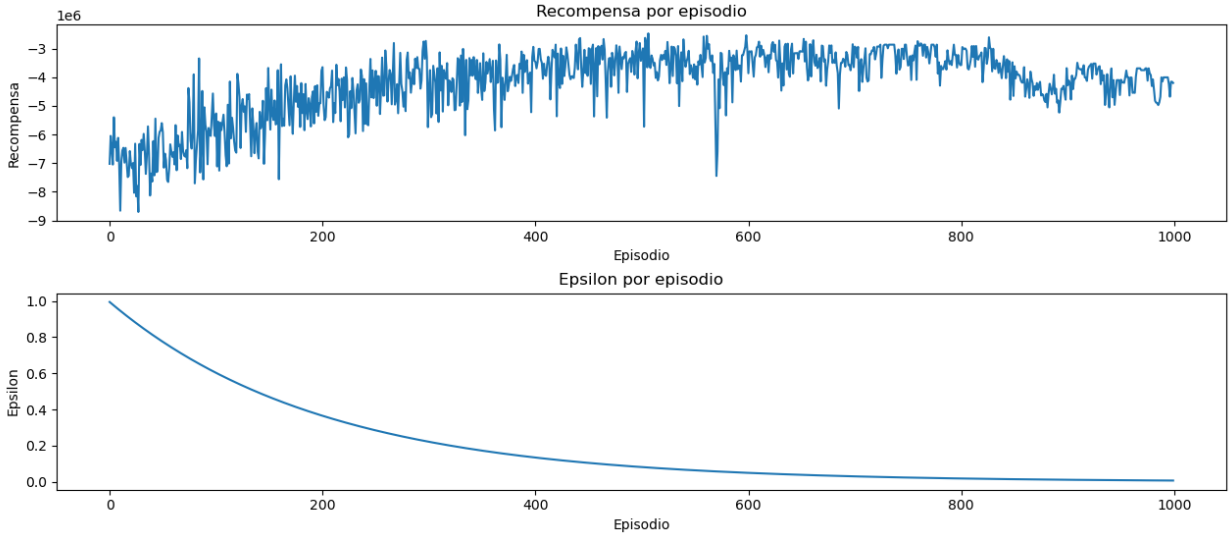

Figura C.8: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración H

C.1.9. Configuración I

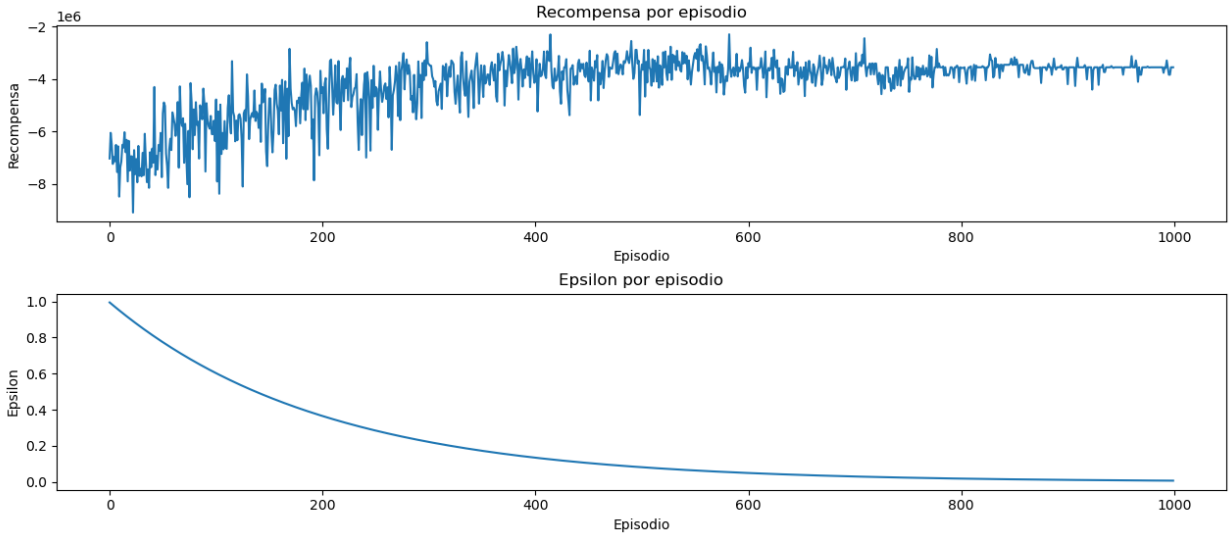

Figura C.9: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración I

C.1.10. Configuración J

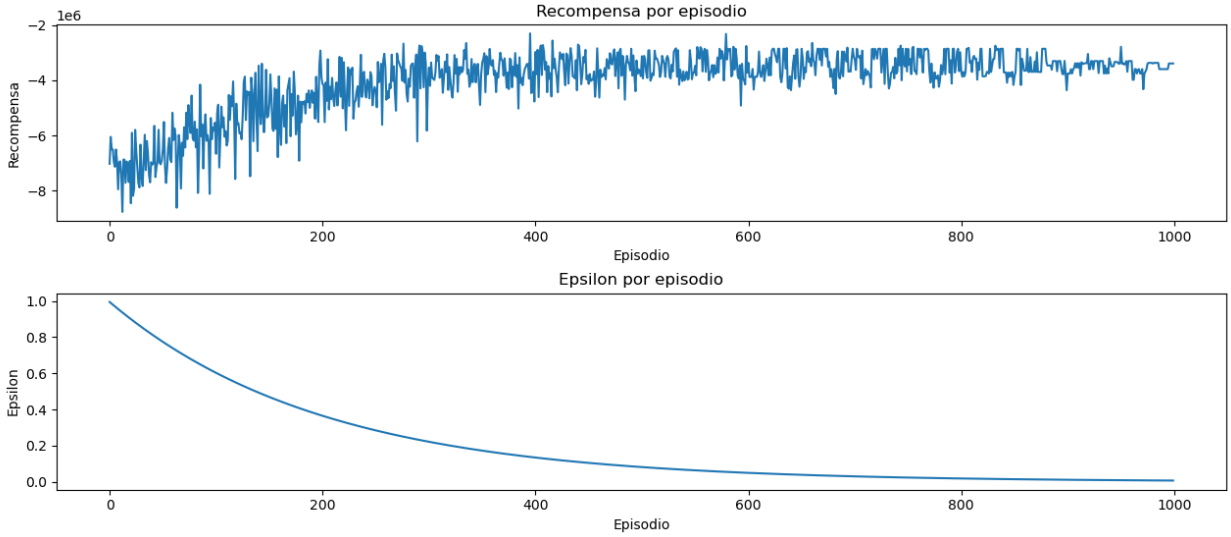

Figura C.10: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración J

C.1.11. Configuración K

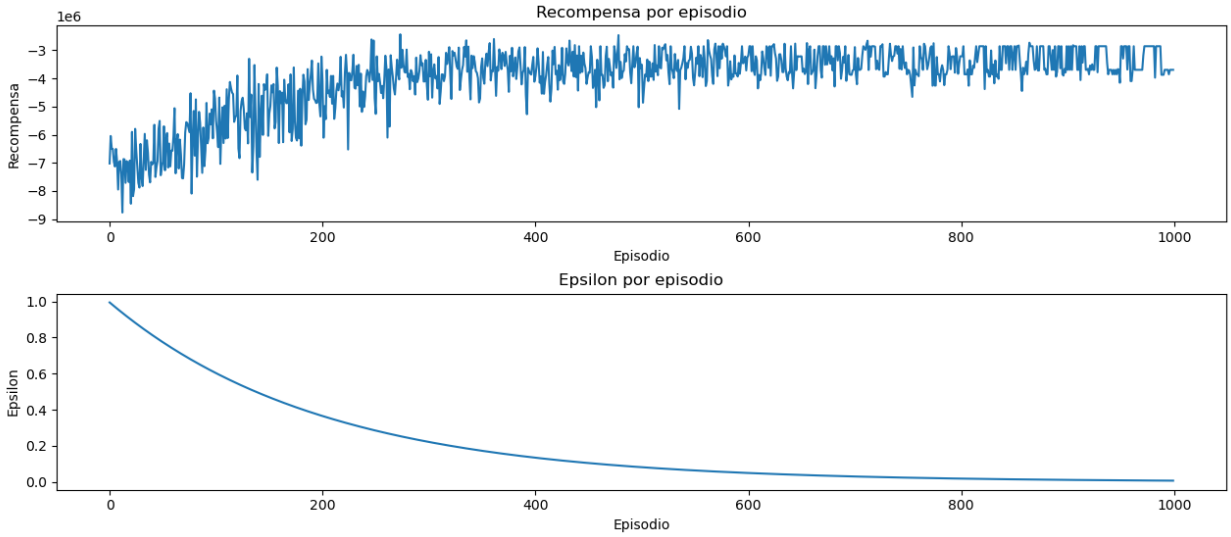

Figura C.11: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración K

C.1.12. Configuración L1

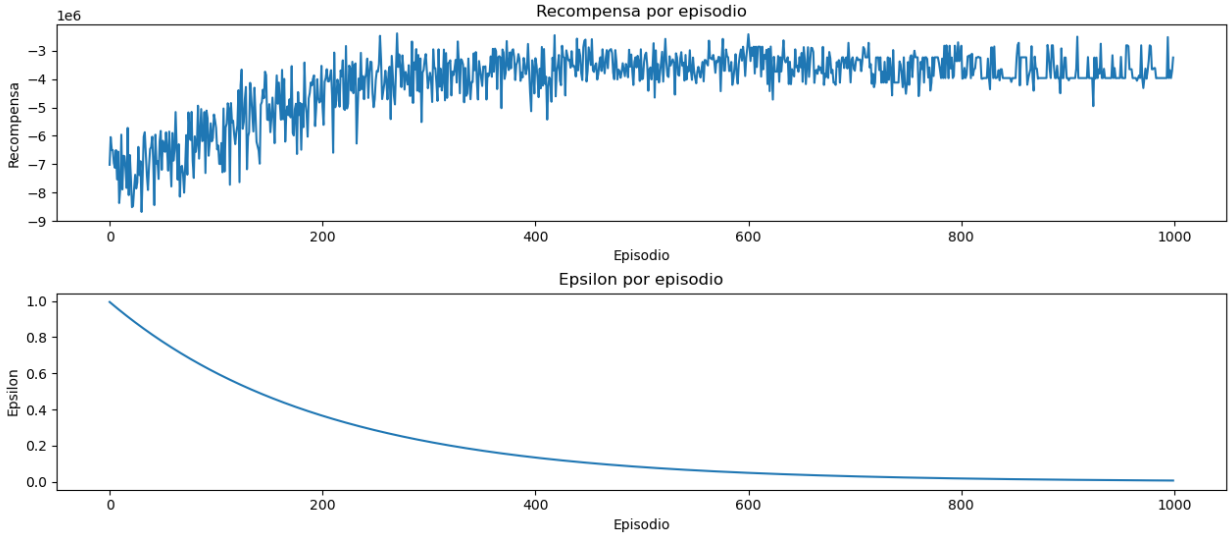

Figura C.12: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración L1

C.1.13. Configuración L2

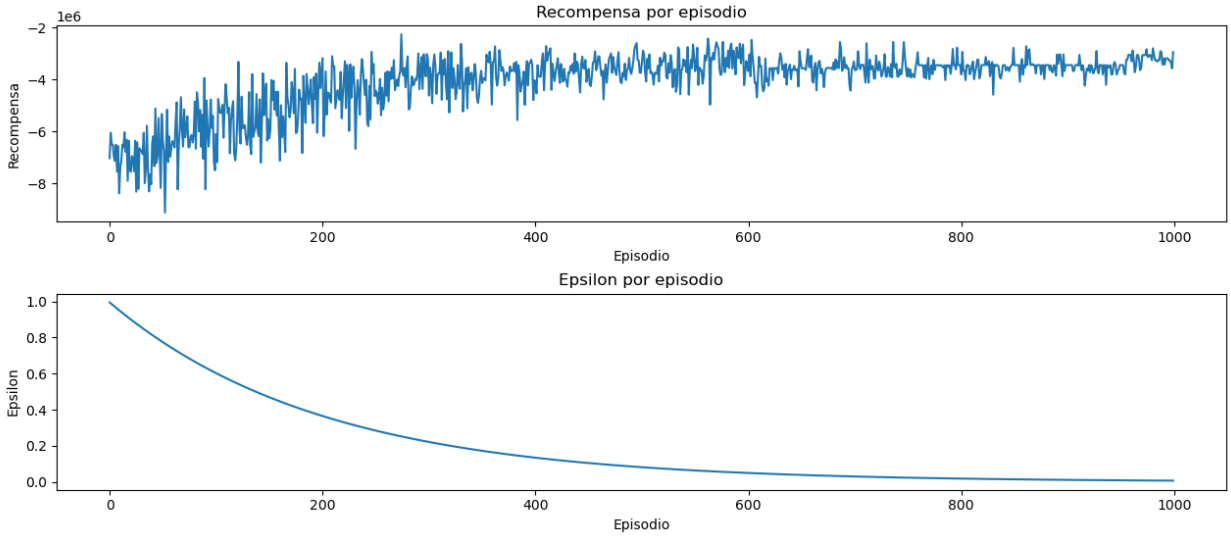

Figura C.13: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración L2

C.1.14. Configuración L3

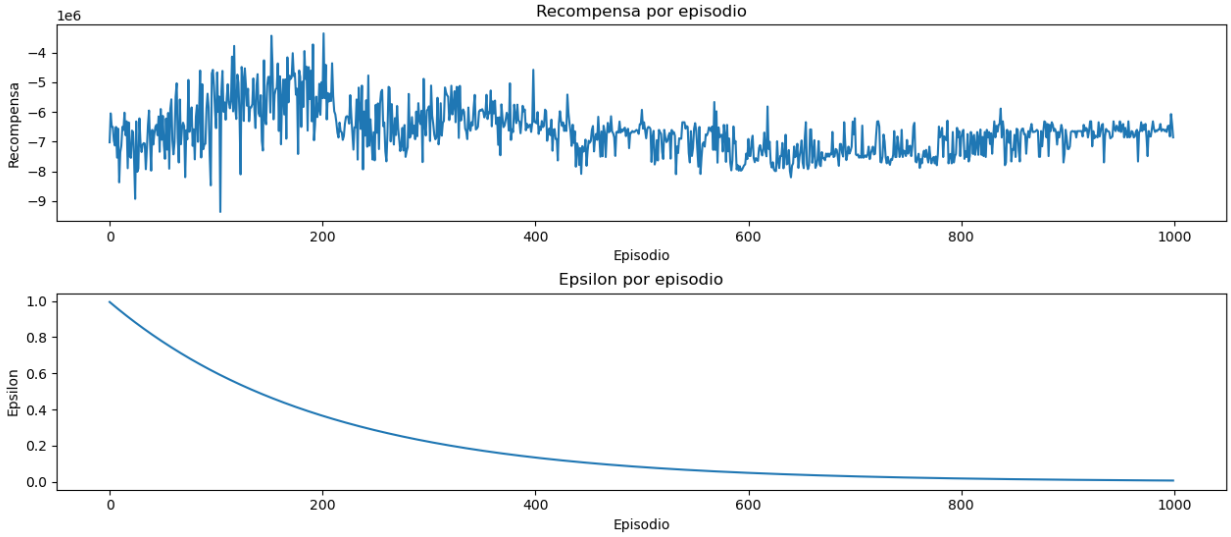

Figura C.14: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración L3

C.1.15. Configuración M1

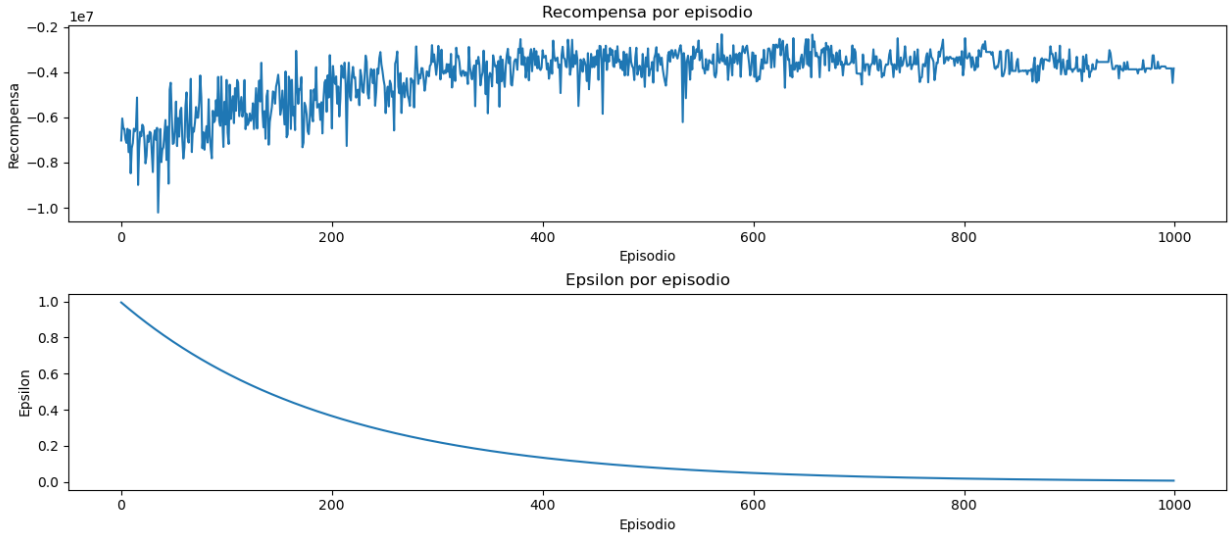

Figura C.15: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración M1

C.1.16. Configuración M2

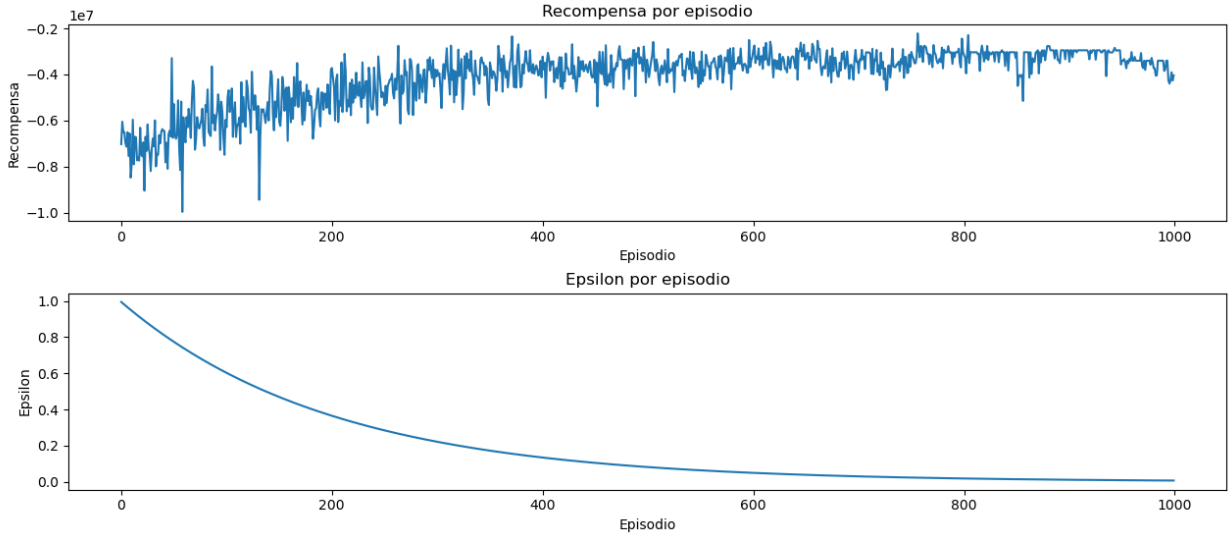

Figura C.16: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración M2

C.1.17. Configuración M3

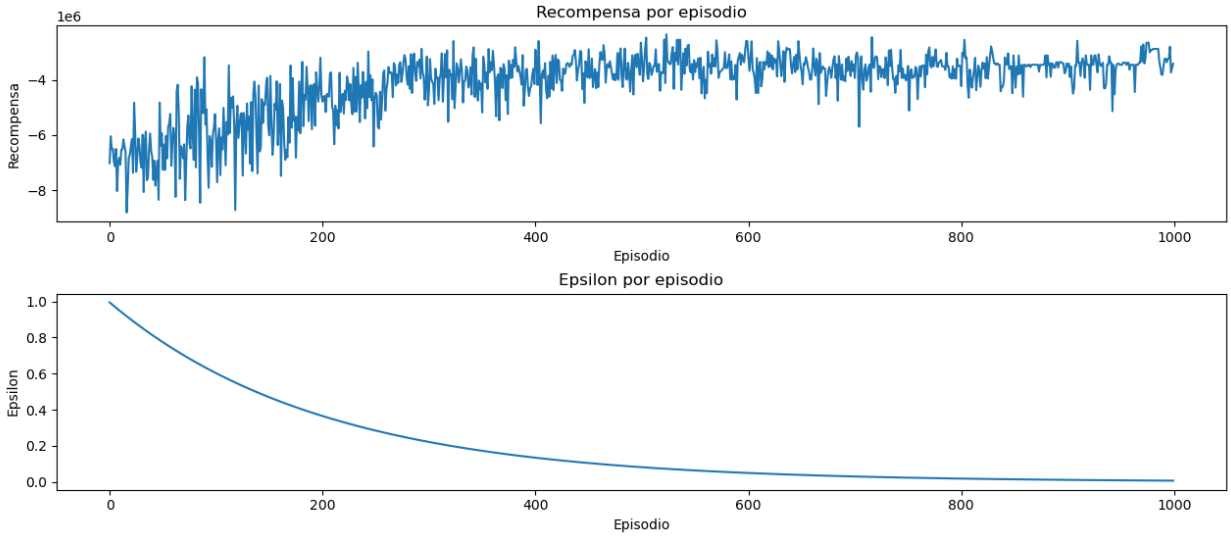

Figura C.17: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración M3

C.1.18. Configuración N1

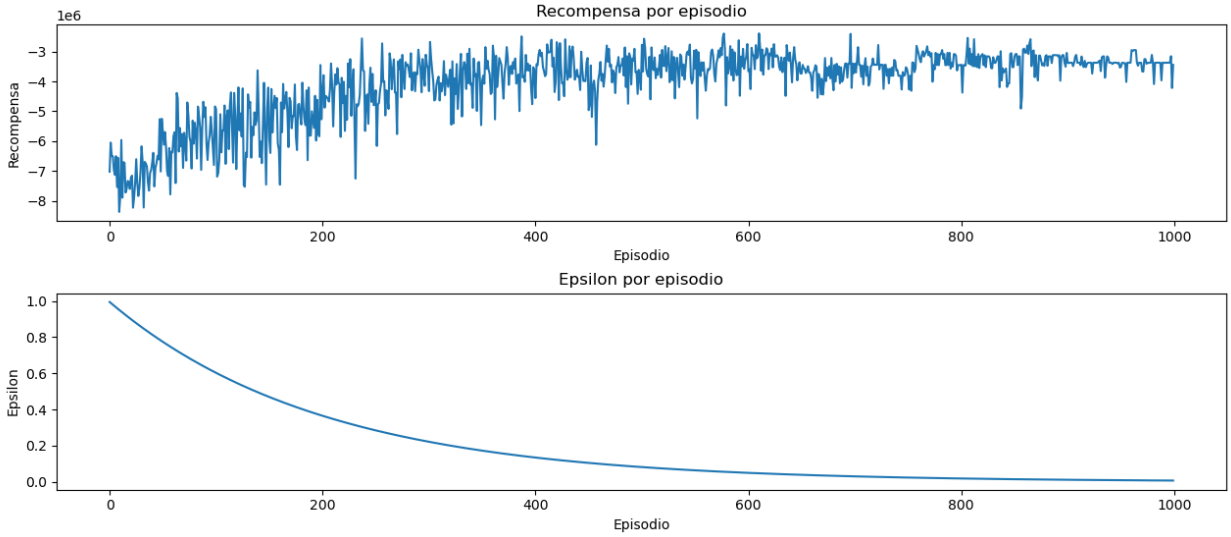

Figura C.18: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración N1

C.1.19. Configuración N2

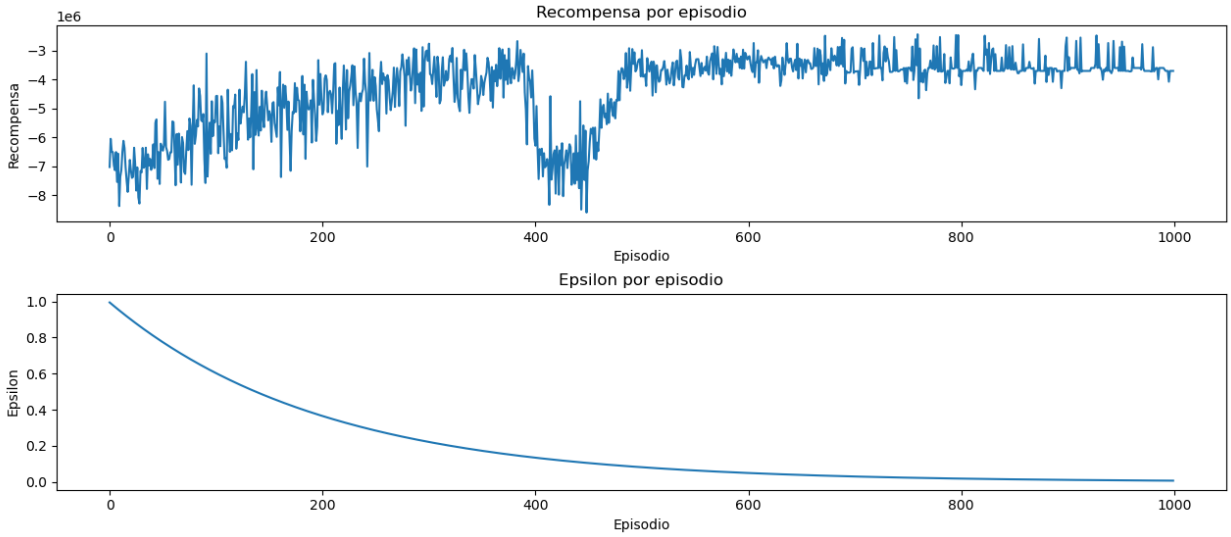

Figura C.19: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración N2

C.1.20. Configuración N3

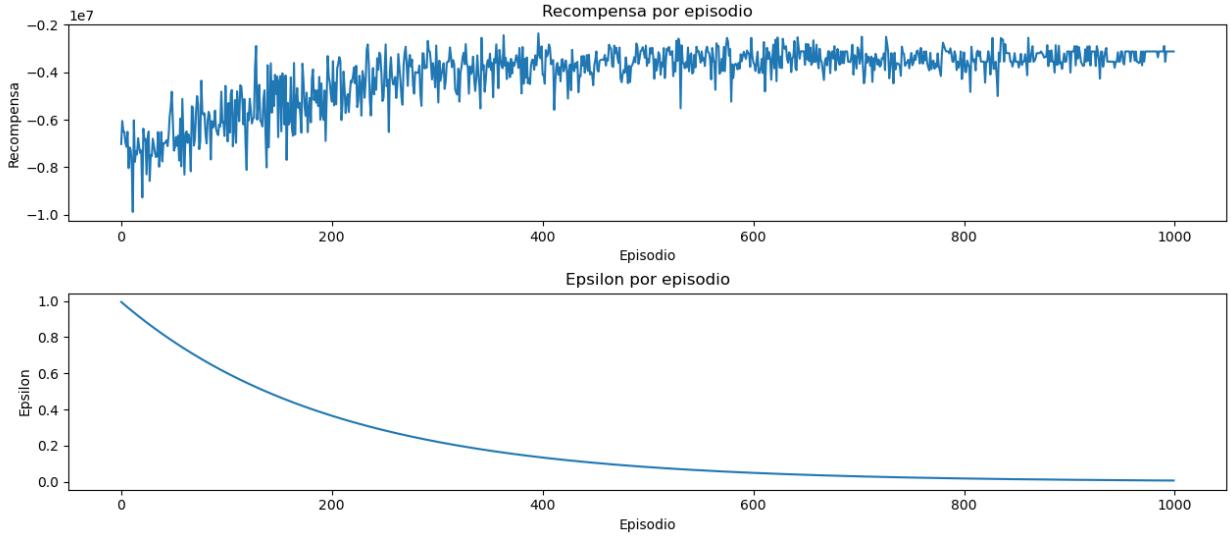

Figura C.20: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración N3

C.1.21. Configuración Ñ1

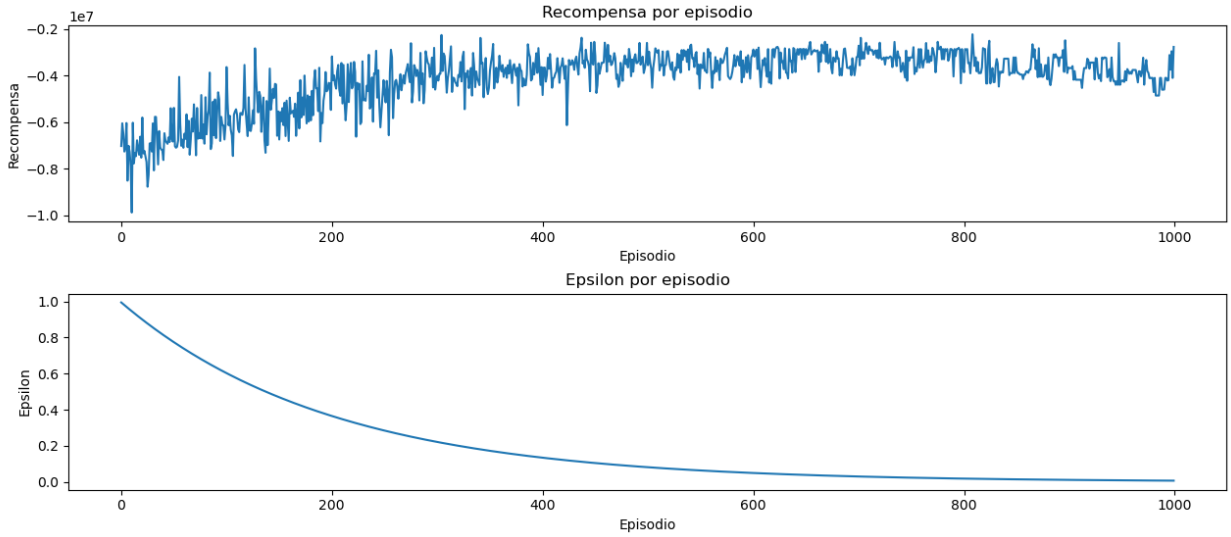

Figura C.21: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración Ñ1

C.1.22. Configuración Ñ2

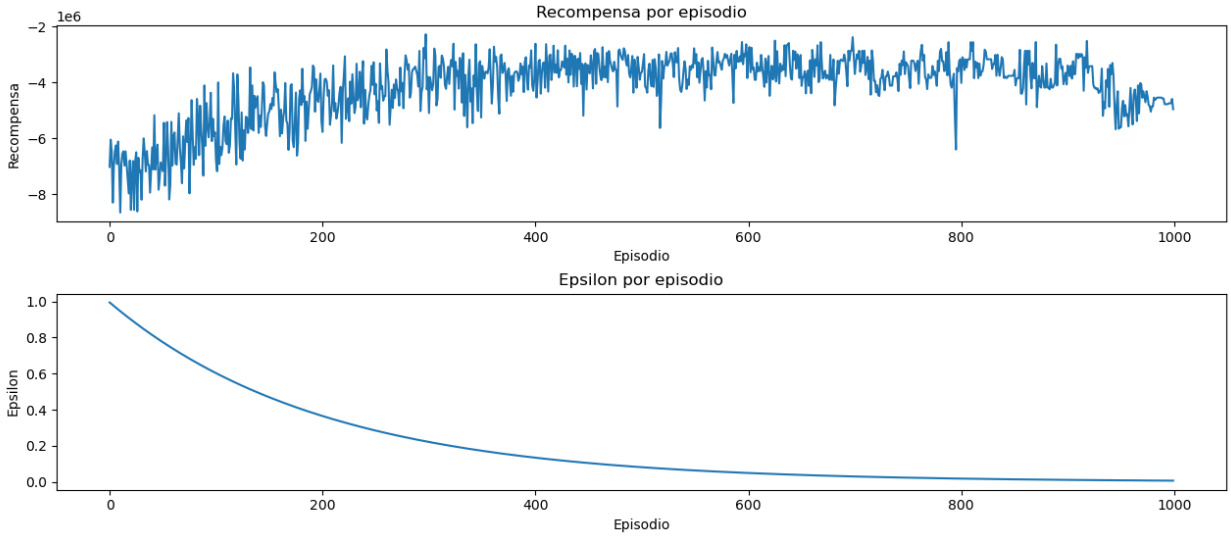

Figura C.22: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración Ñ2

C.1.23. Configuración Ñ3

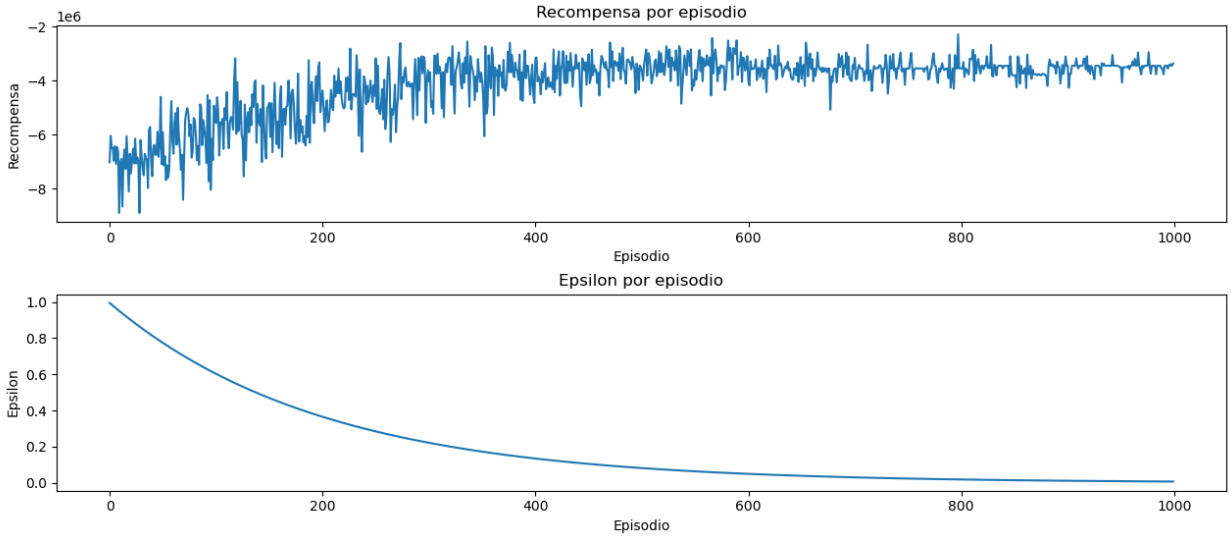

Figura C.23: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración Ñ3

C.1.24. Configuración O1

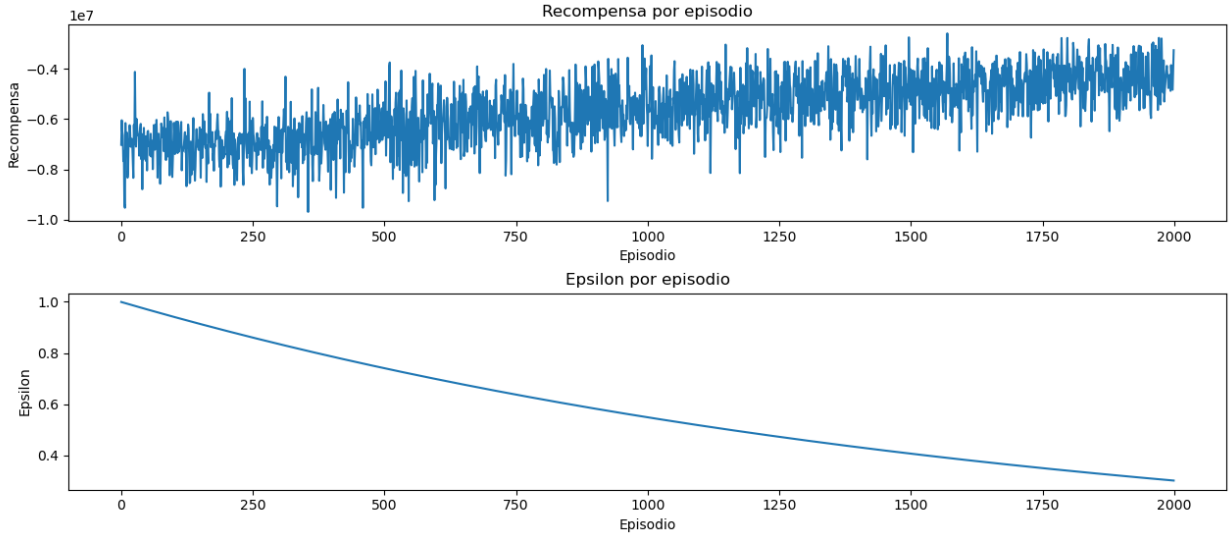

Figura C.24: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración O1

C.1.25. Configuración O2

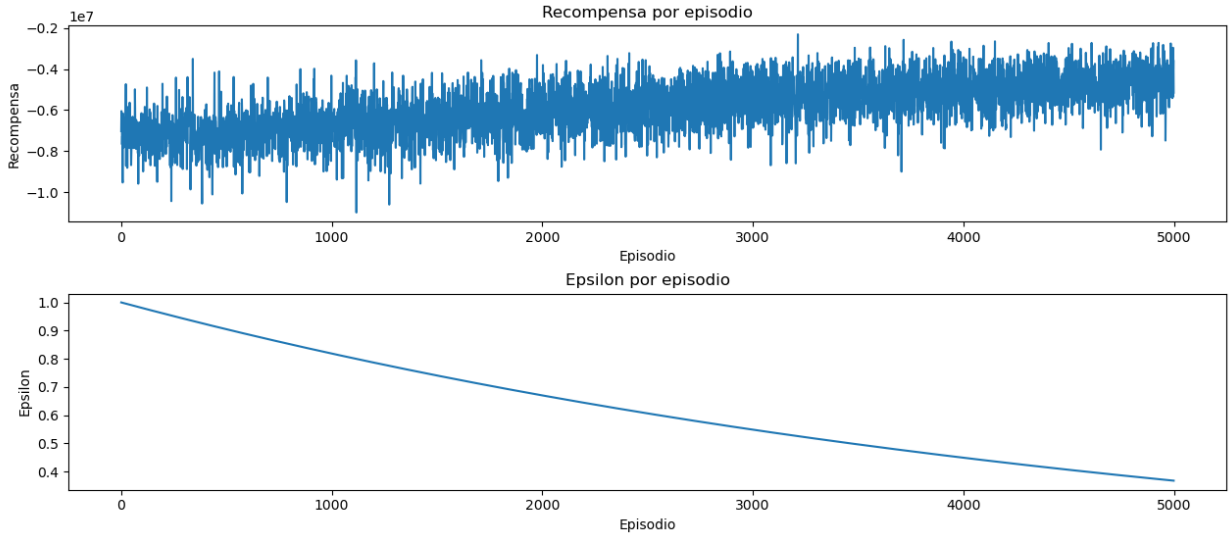

Figura C.25: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración O2

C.1.26. Configuración O3

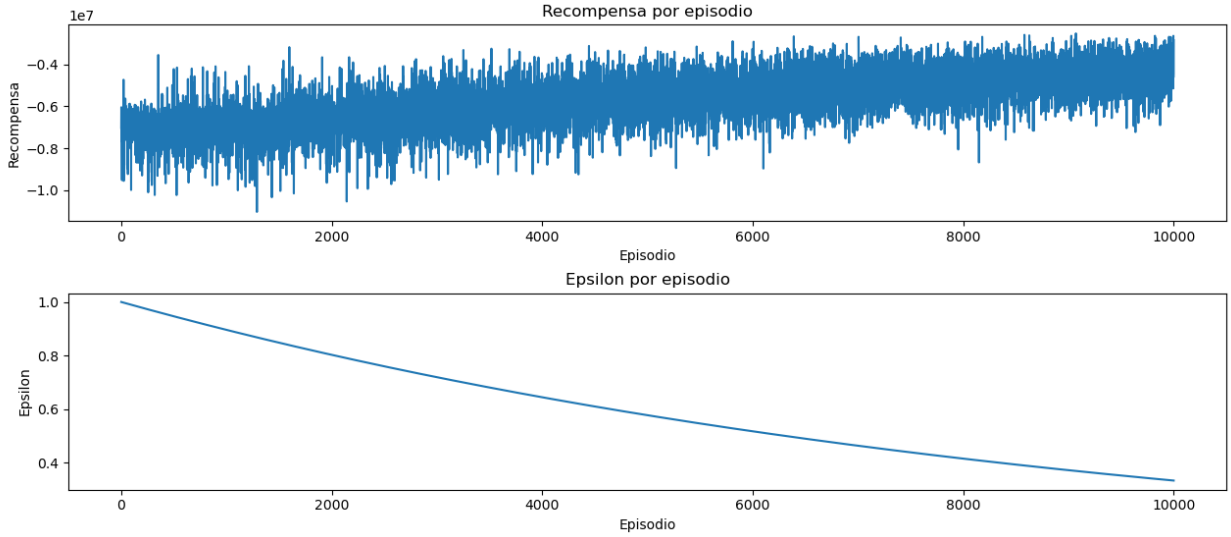

Figura C.26: Recompensa acumulada y disminución de ϵ obtenidos de modelo basado en configuración O3

C.2. Promedio móvil de recompensas acumuladas

C.2.1. Configuración A

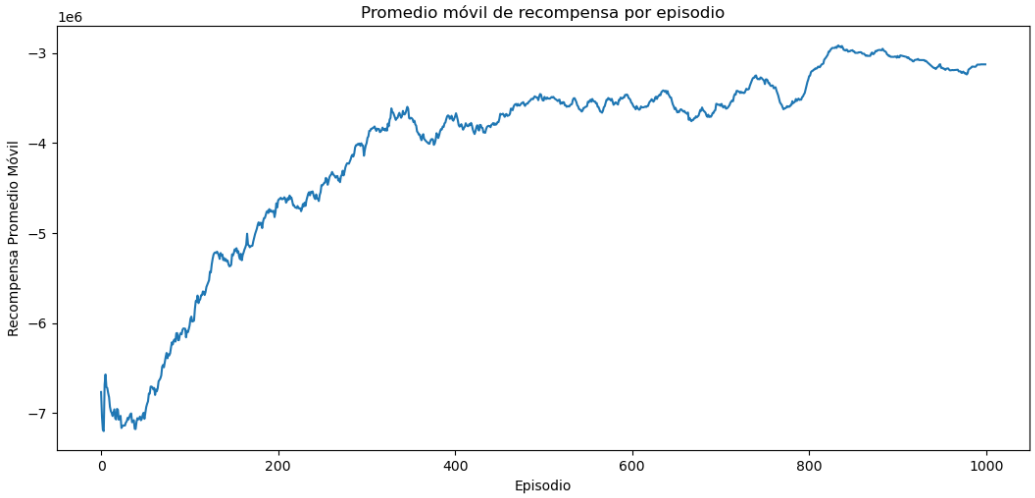

Figura C.27: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración A

C.2.2. Configuración B

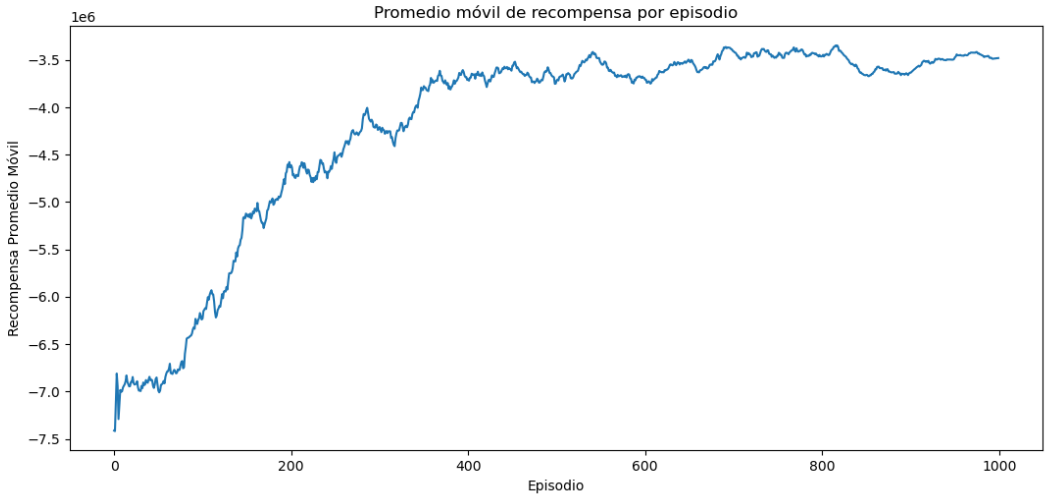

Figura C.28: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración B

C.2.3. Configuración C

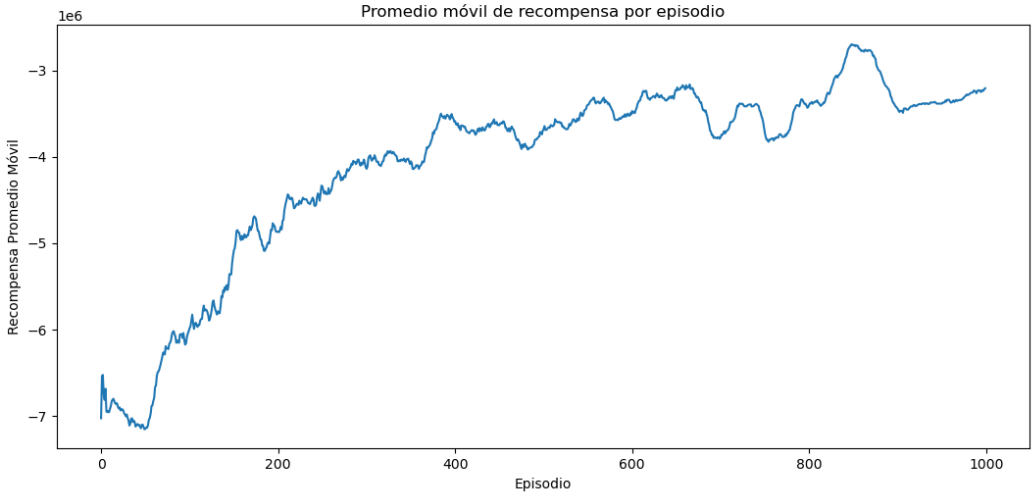

Figura C.29: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración C

C.2.4. Configuración D

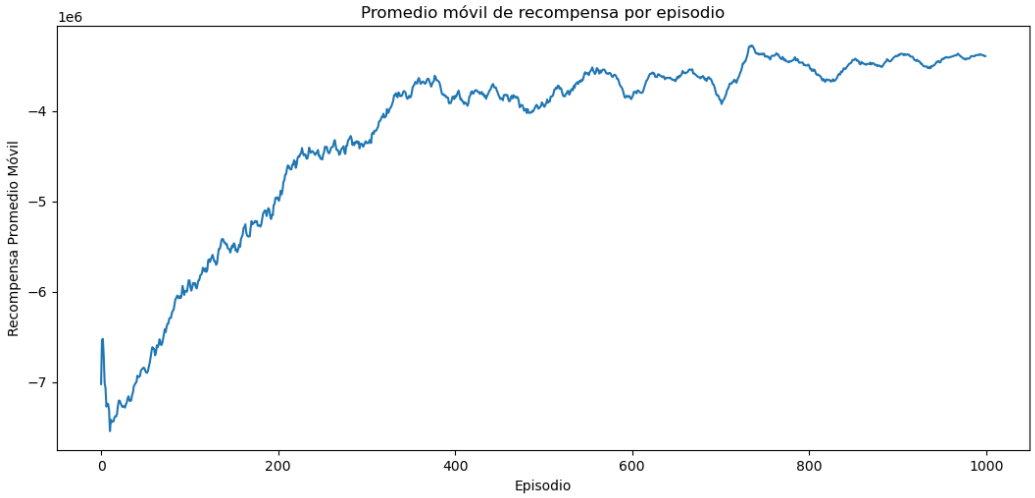

Figura C.30: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración D

C.2.5. Configuración E

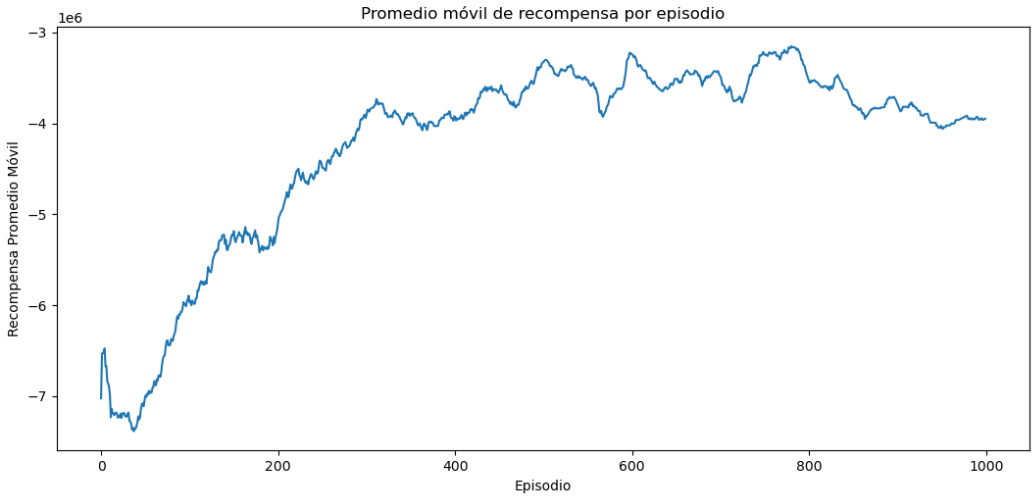

Figura C.31: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración E

C.2.6. Configuración F

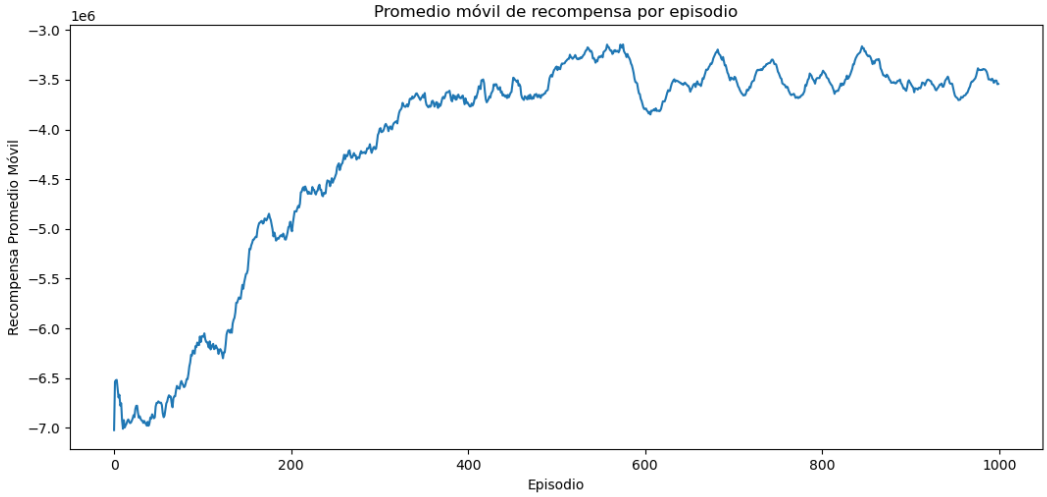

Figura C.32: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración F

C.2.7. Configuración G

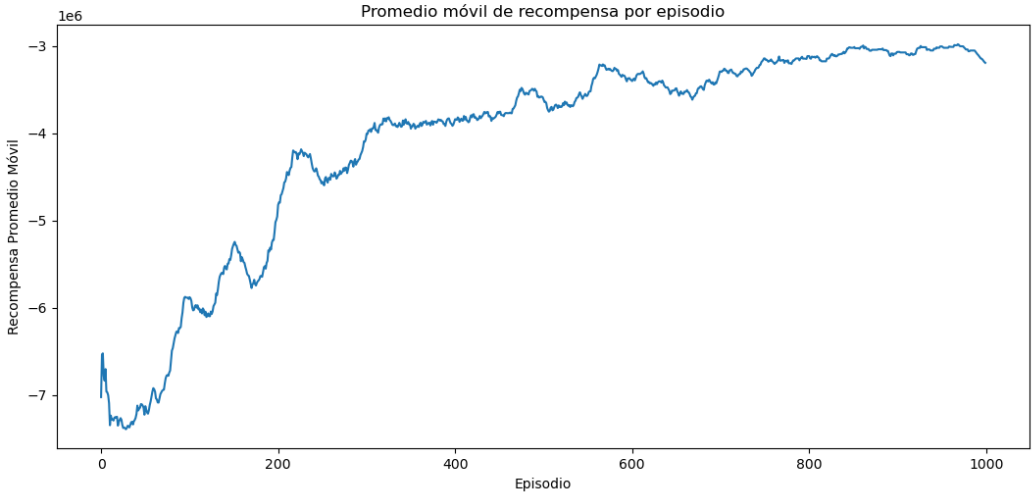

Figura C.33: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración G

C.2.8. Configuración H

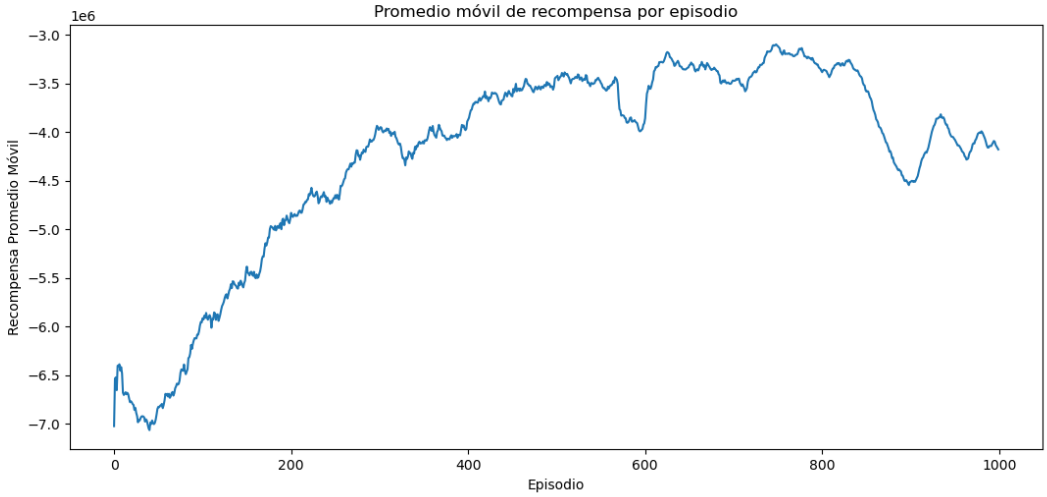

Figura C.34: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración H

C.2.9. Configuración I

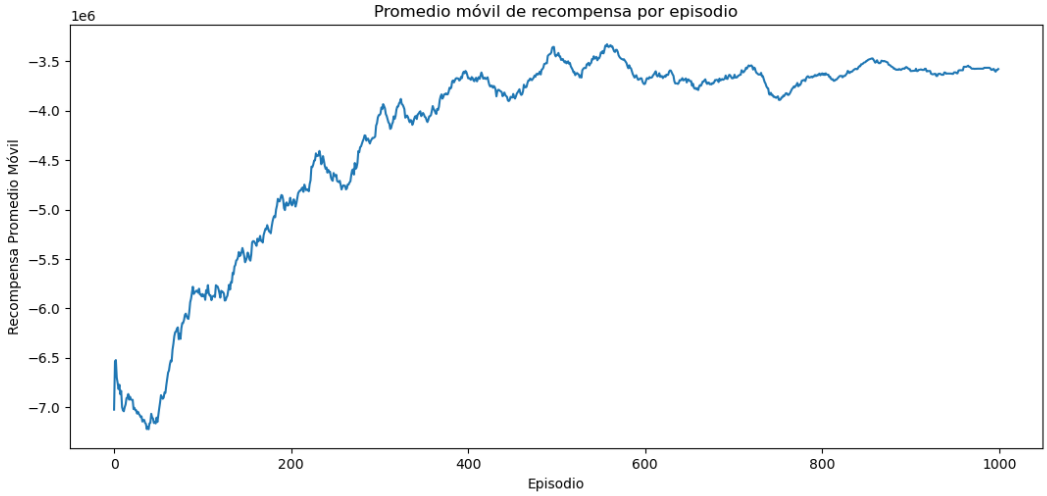

Figura C.35: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración I

C.2.10. Configuración J

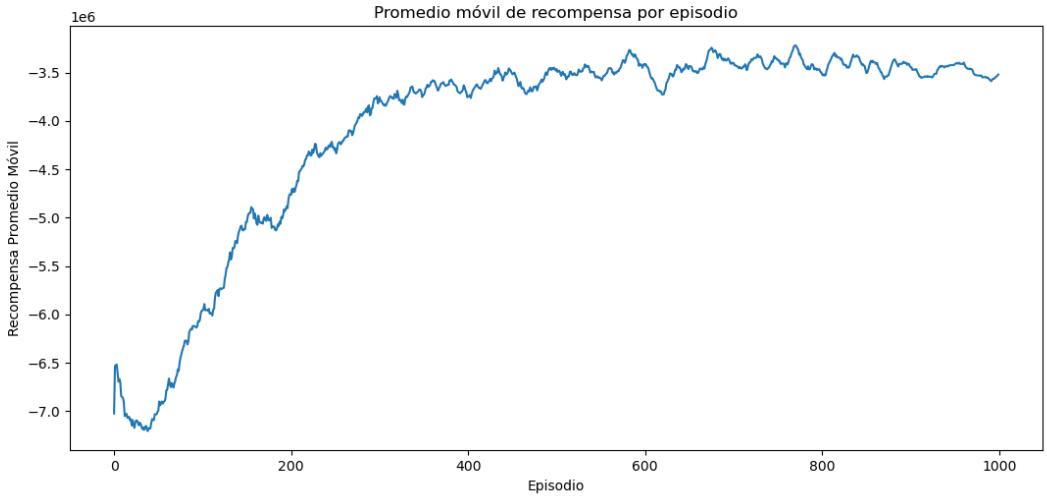

Figura C.36: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración J

C.2.11. Configuración K

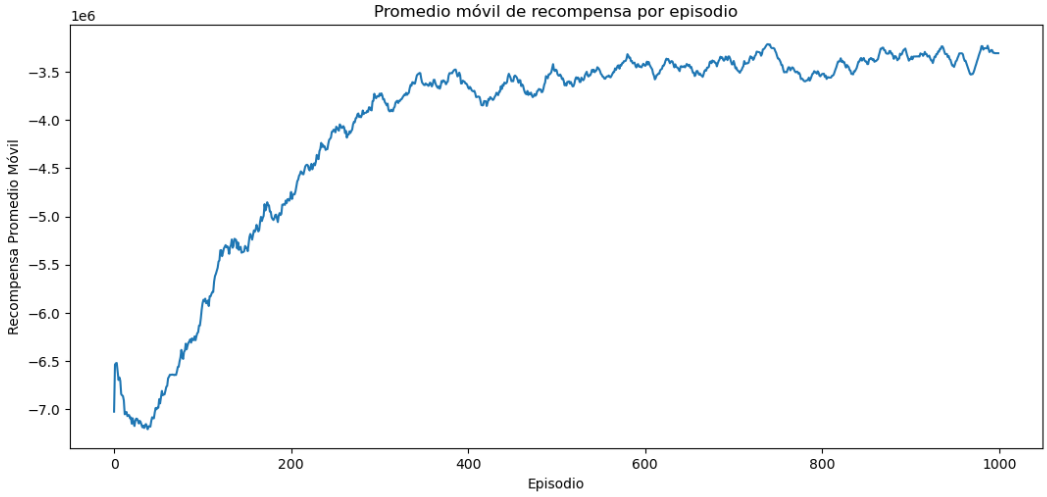

Figura C.37: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración K

C.2.12. Configuración L1

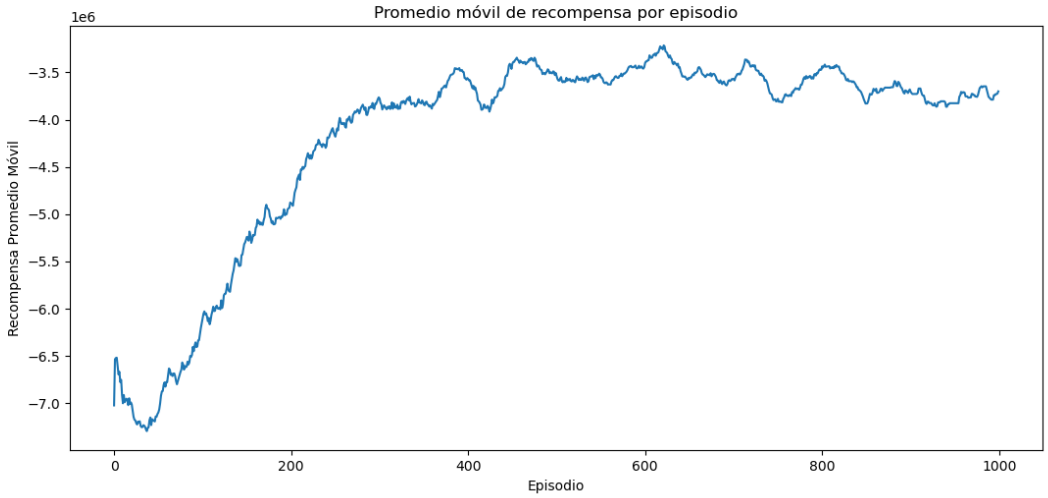

Figura C.38: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración L1

C.2.13. Configuración L2

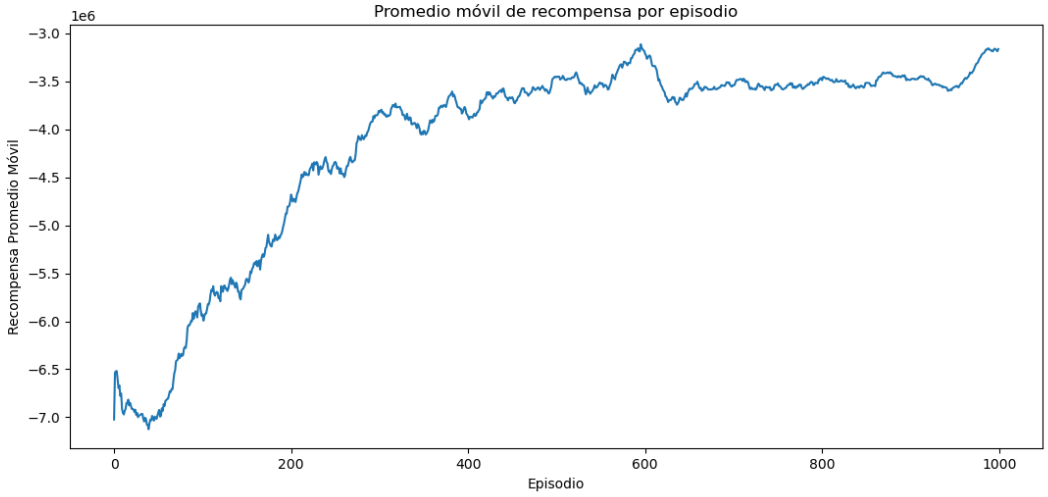

Figura C.39: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración L2

C.2.14. Configuración L3

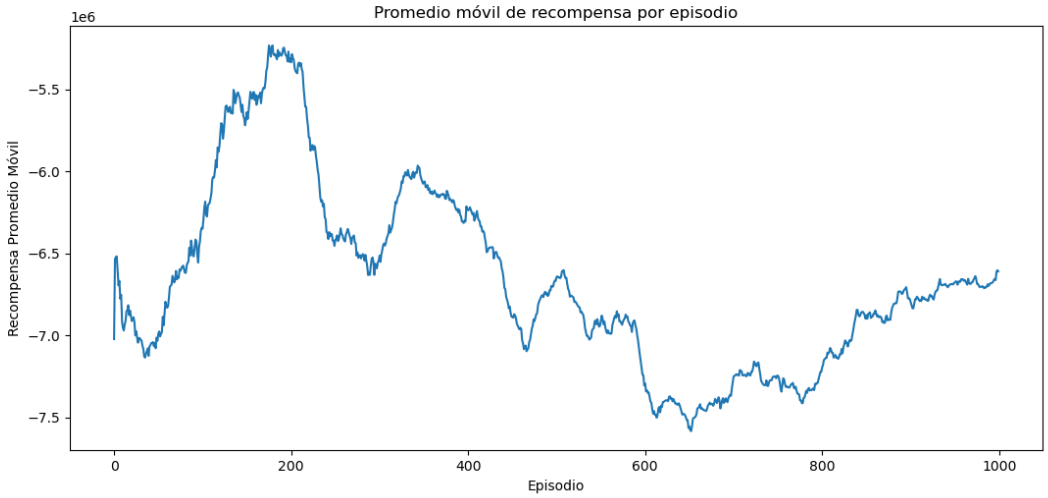

Figura C.40: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración L3

C.2.15. Configuración M1

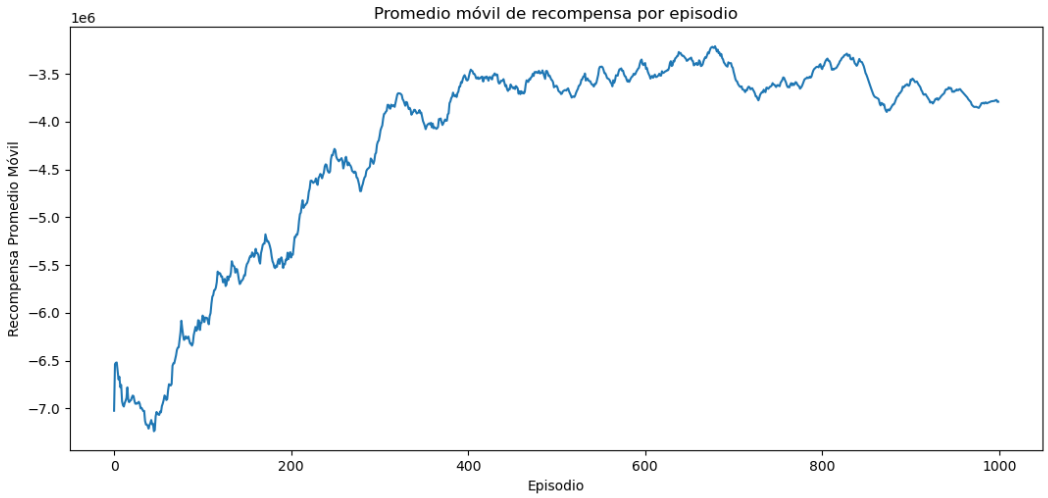

Figura C.41: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración M1

C.2.16. Configuración M2

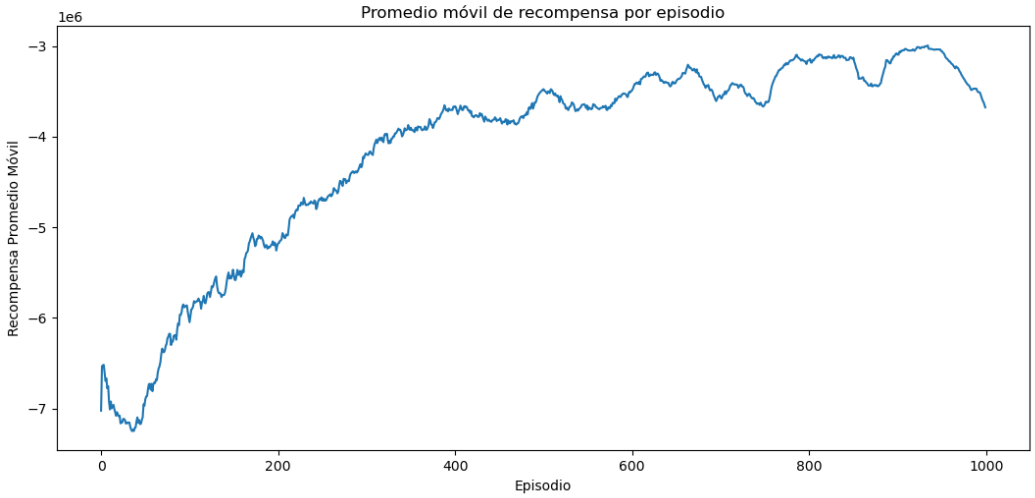

Figura C.42: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración M2

C.2.17. Configuración M3

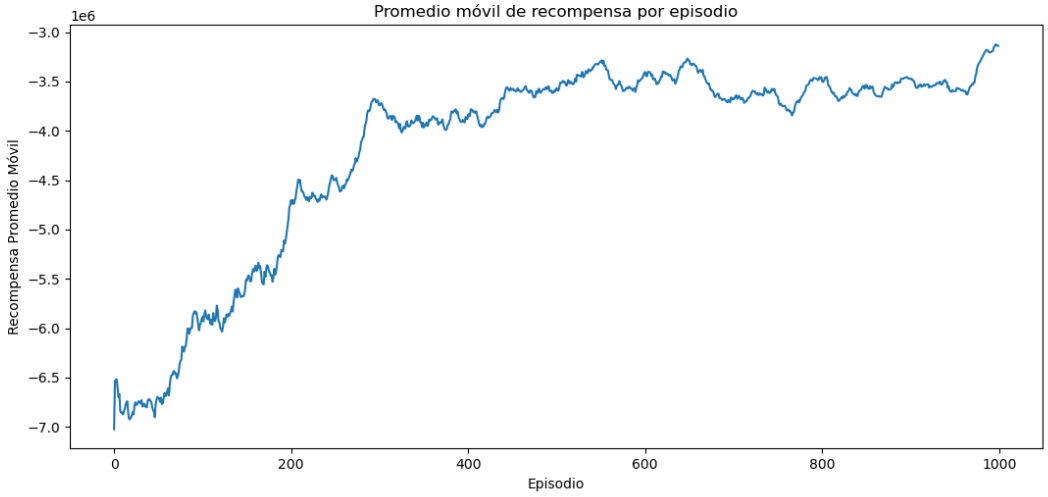

Figura C.43: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración M3

C.2.18. Configuración N1

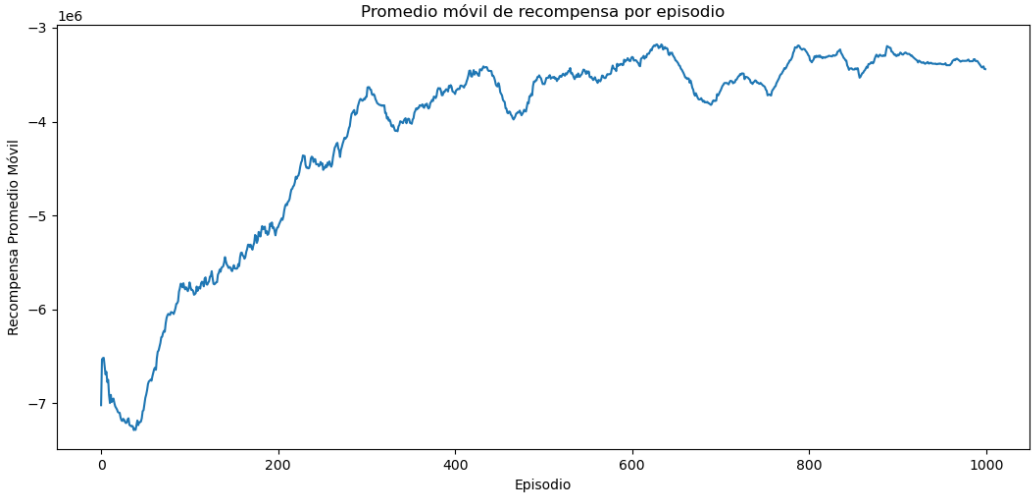

Figura C.44: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración N1

C.2.19. Configuración N2

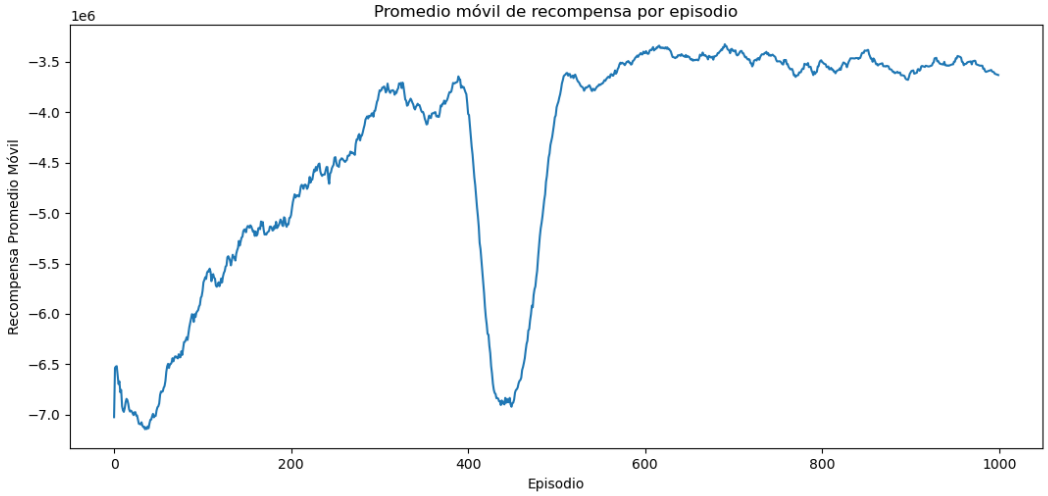

Figura C.45: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración N2

C.2.20. Configuración N3

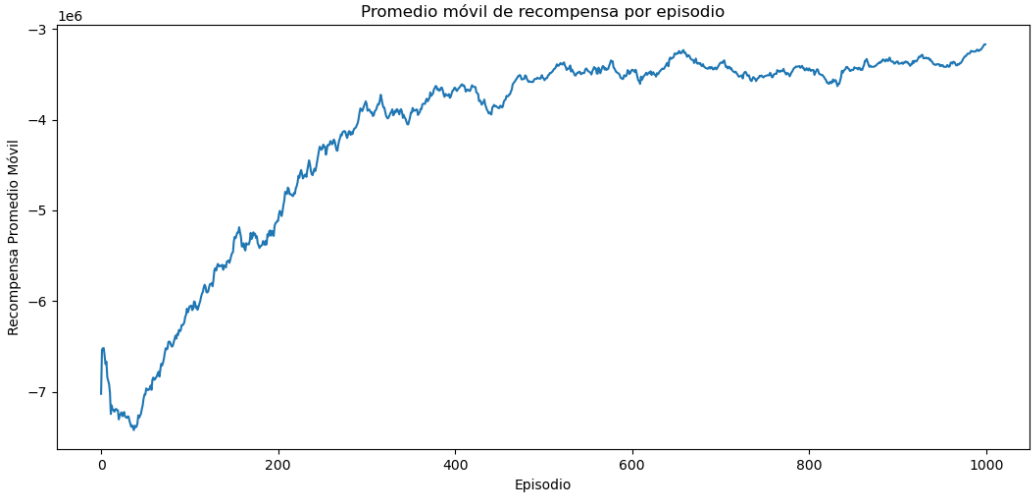

Figura C.46: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración N3

C.2.21. Configuración Ñ1

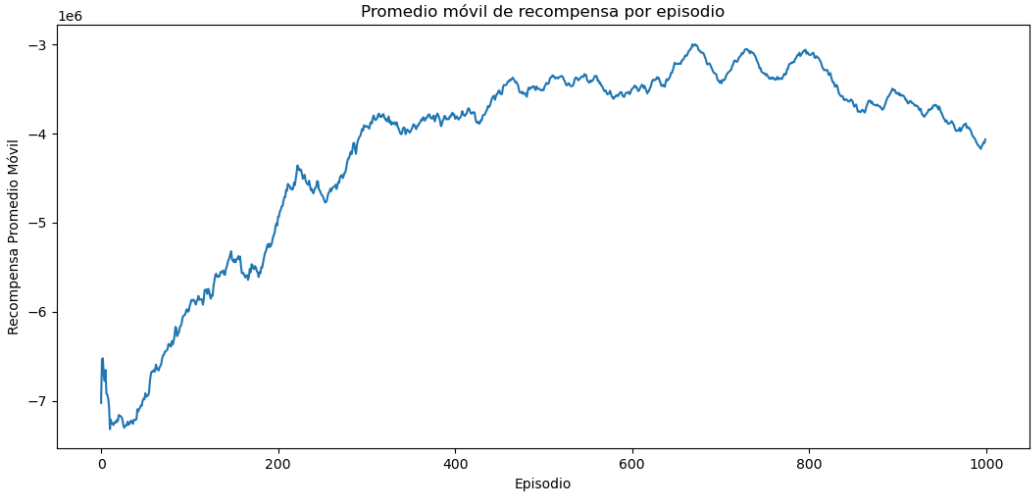

Figura C.47: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración Ñ1

C.2.22. Configuración Ñ2

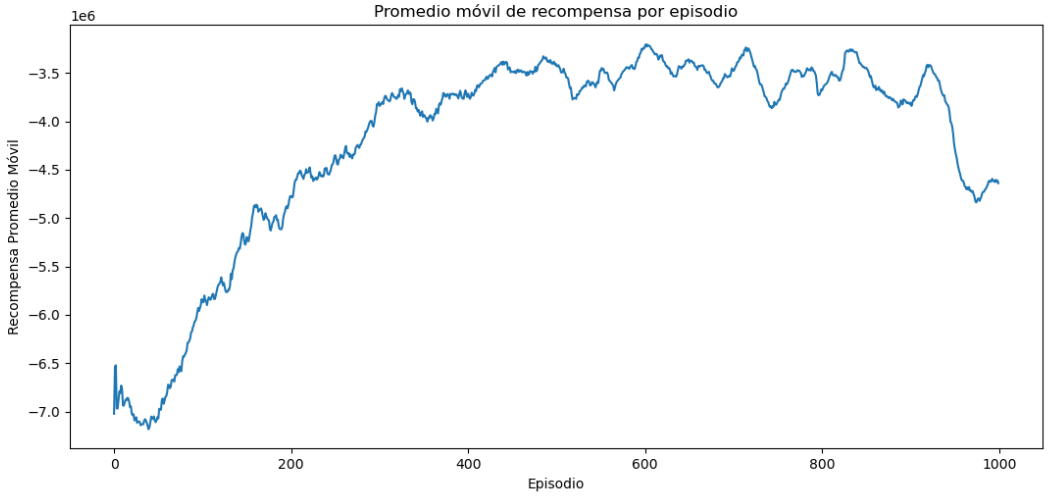

Figura C.48: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración Ñ2

C.2.23. Configuración Ñ3

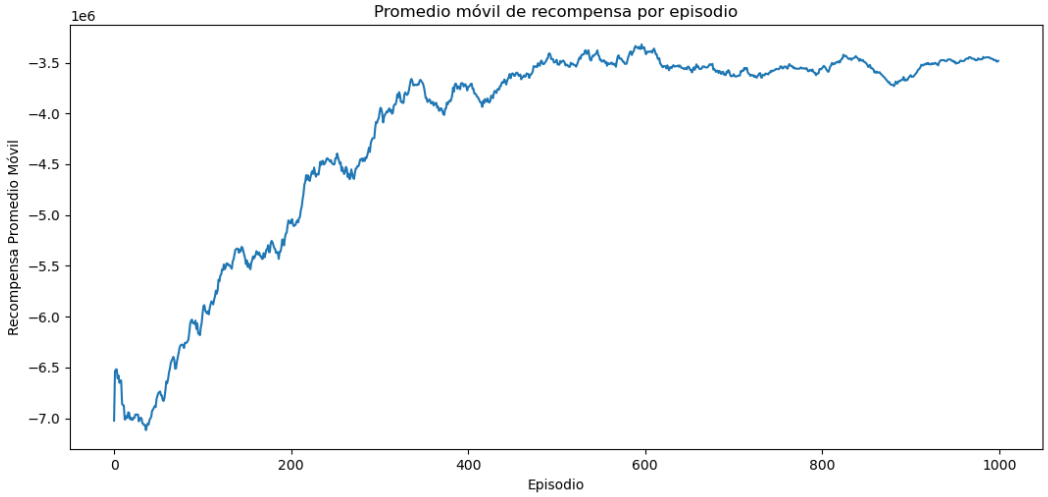

Figura C.49: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración Ñ3

C.2.24. Configuración O1

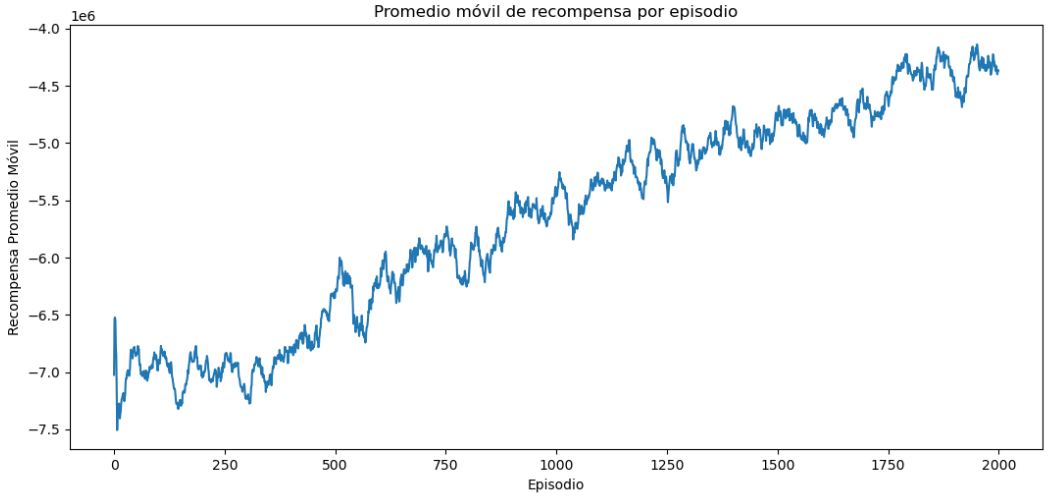

Figura C.50: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración O1

C.2.25. Configuración O2

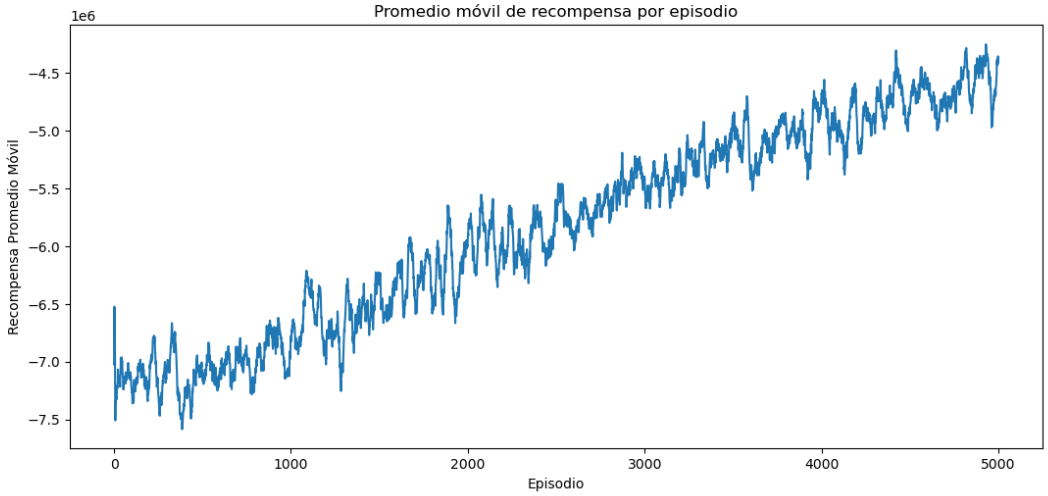

Figura C.51: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración O2

C.2.26. Configuración O3

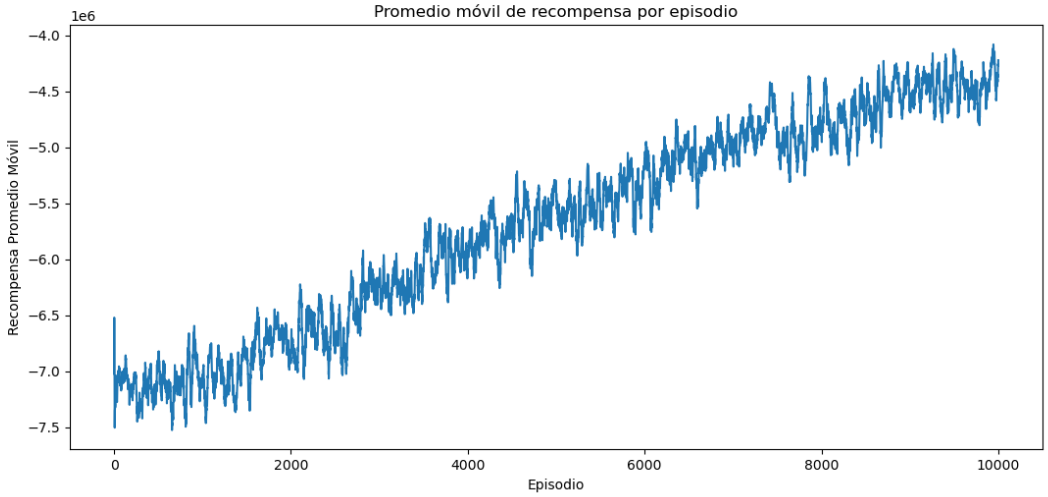

Figura C.52: Promedio móvil cada 33 episodios de recompensa acumulada obtenidos de modelo basado en configuración O3

C.3. Largo de episodios

C.3.1. Configuración A

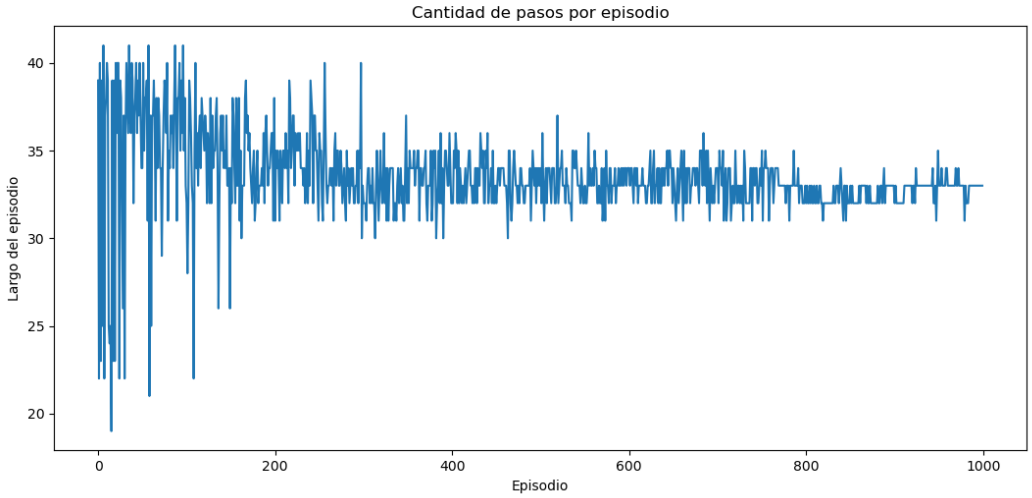

Figura C.53: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración A

C.3.2. Configuración B

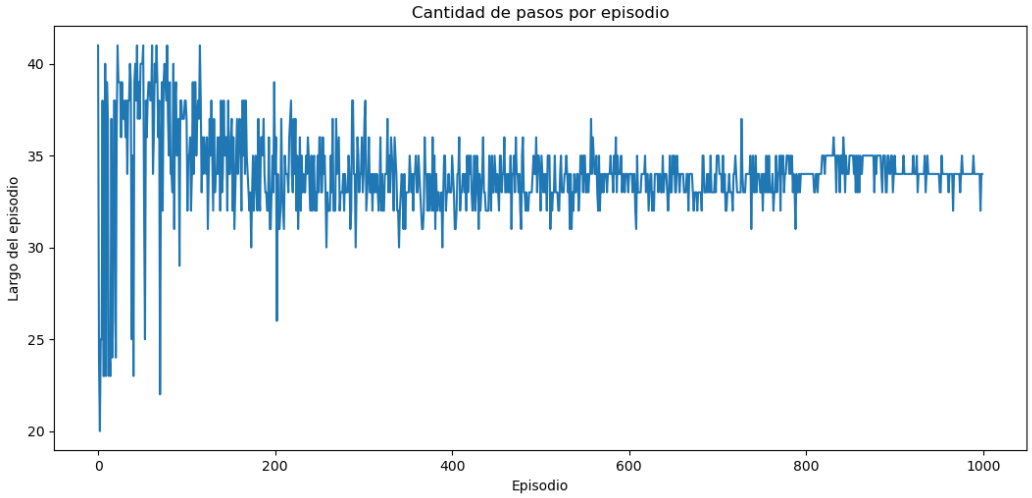

Figura C.54: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración B

C.3.3. Configuración C

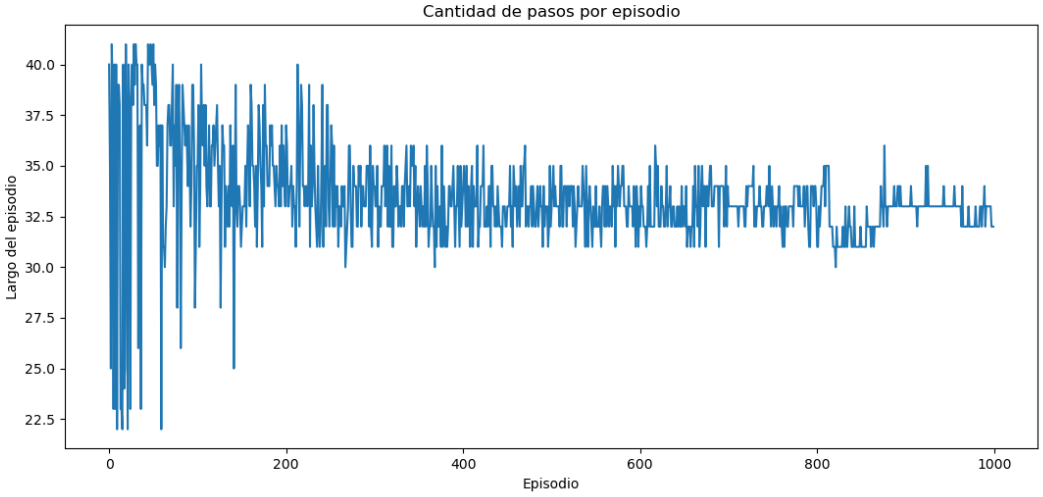

Figura C.55: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración C

C.3.4. Configuración D

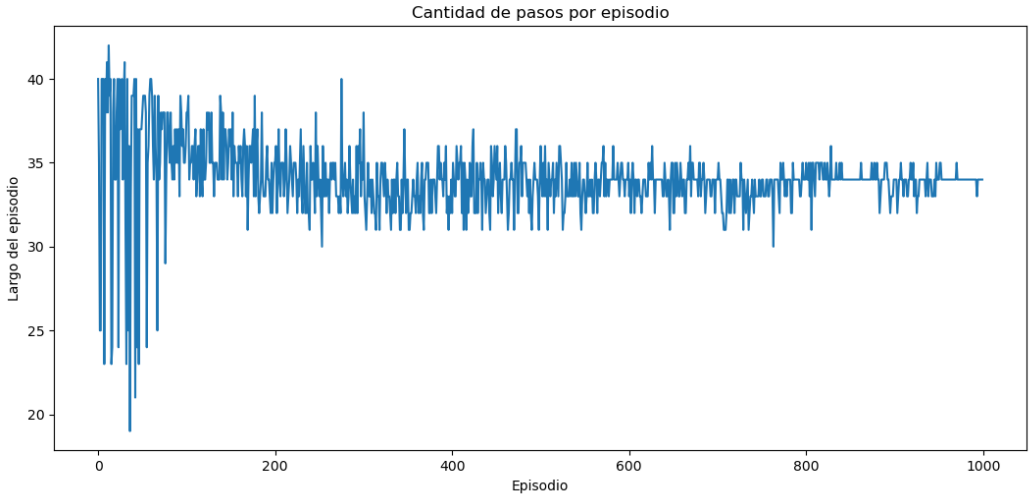

Figura C.56: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración D

C.3.5. Configuración E

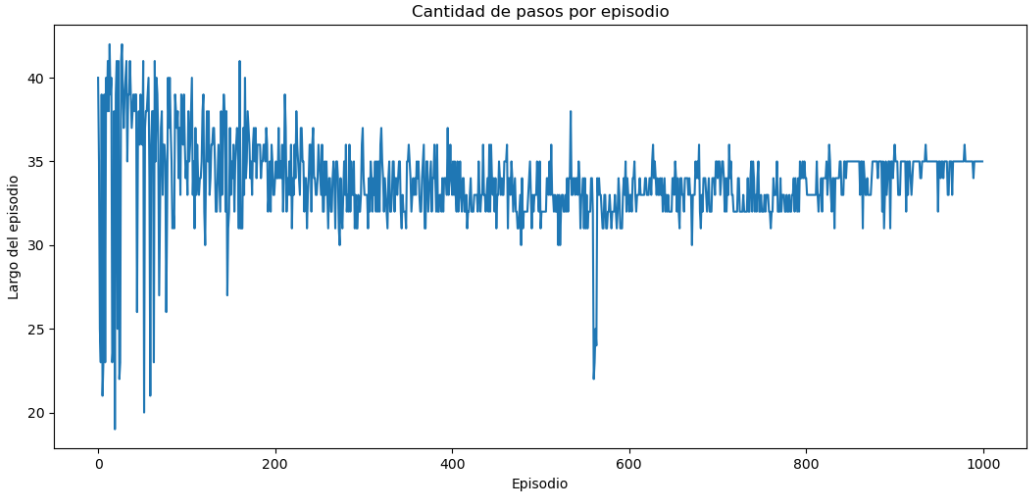

Figura C.57: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración E

C.3.6. Configuración F

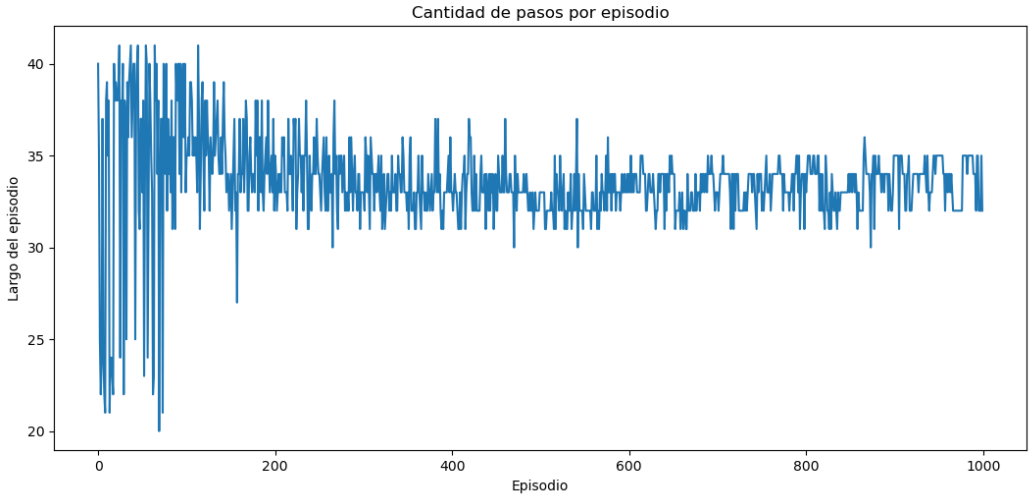

Figura C.58: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración F

C.3.7. Configuración G

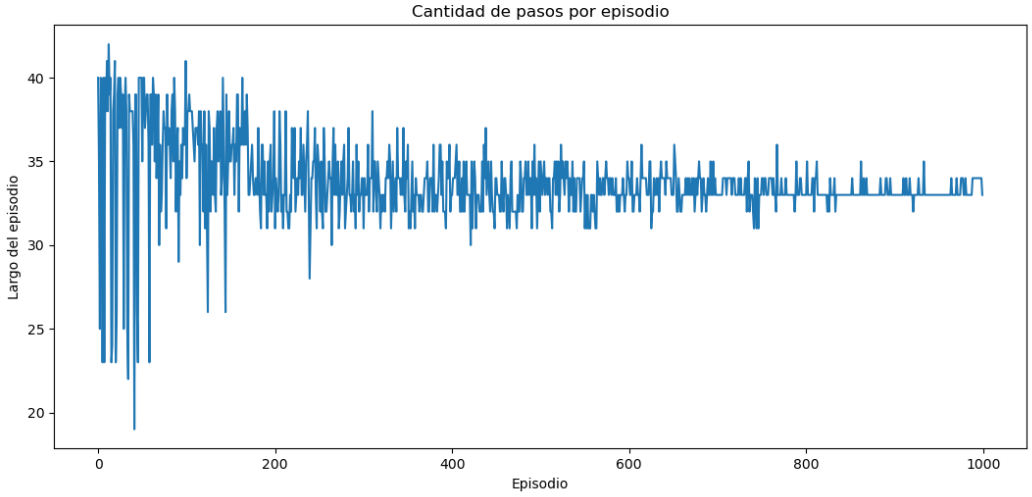

Figura C.59: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración G

C.3.8. Configuración H

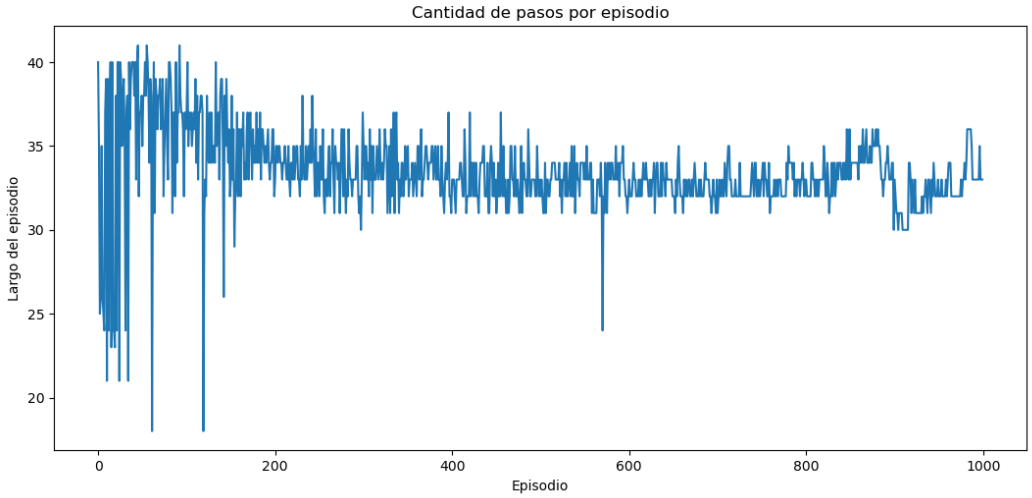

Figura C.60: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración H

C.3.9. Configuración I

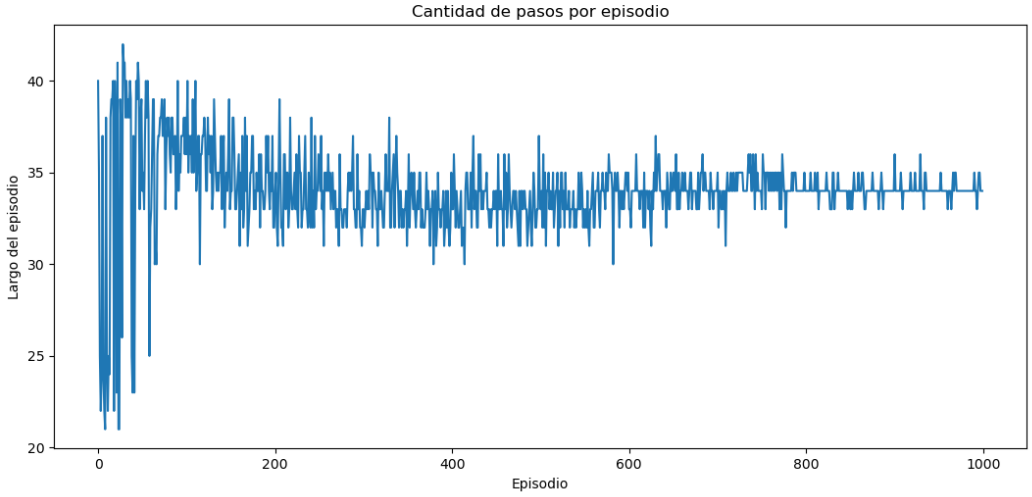

Figura C.61: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración I

C.3.10. Configuración J

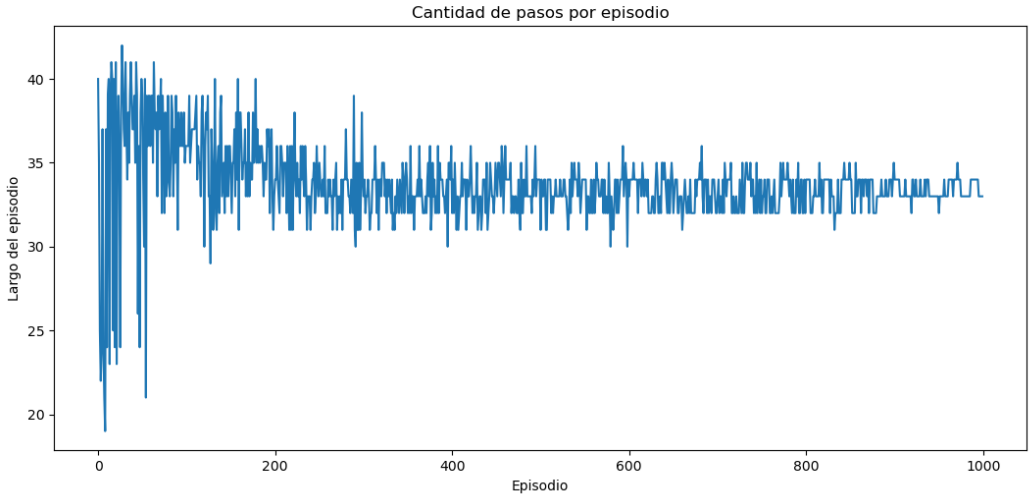

Figura C.62: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración J

C.3.11. Configuración K

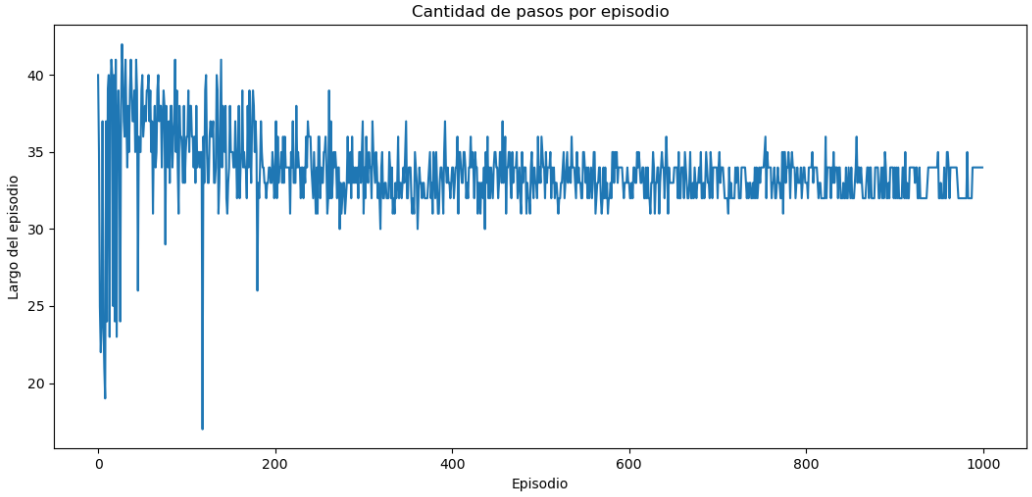

Figura C.63: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración K

C.3.12. Configuración L1

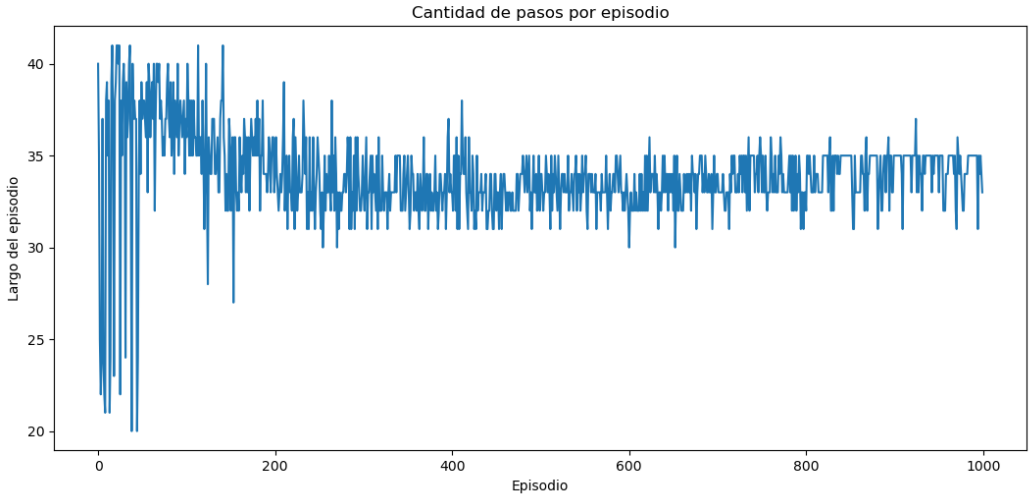

Figura C.64: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración L1

C.3.13. Configuración L2

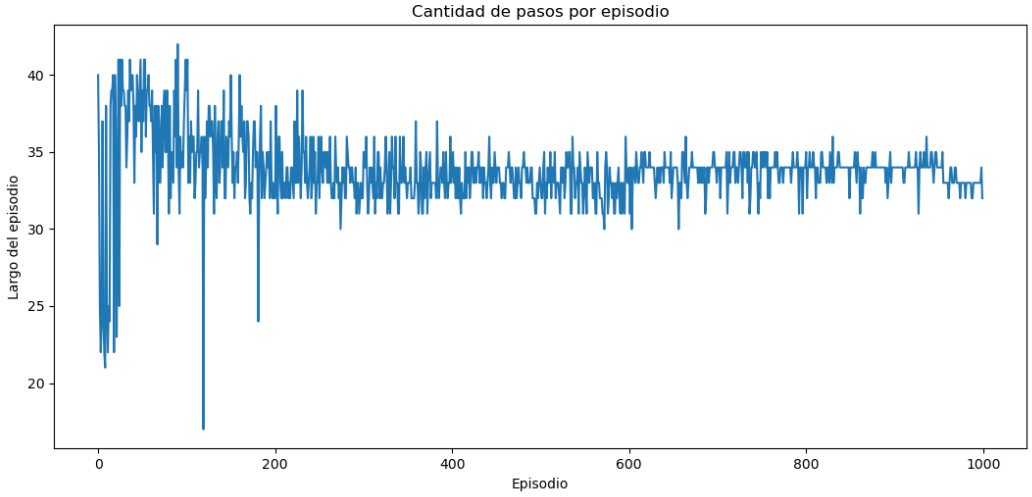

Figura C.65: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración L2

C.3.14. Configuración L3

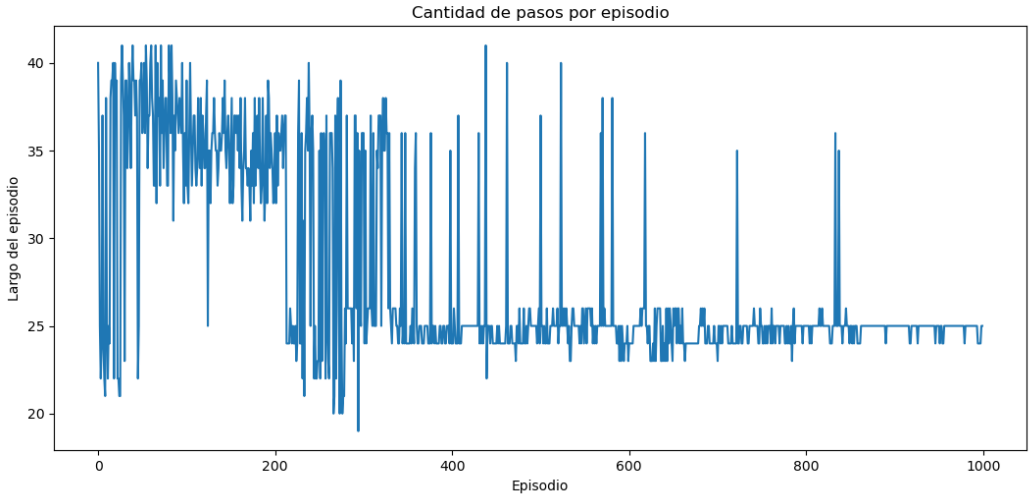

Figura C.66: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración L3

C.3.15. Configuración M1

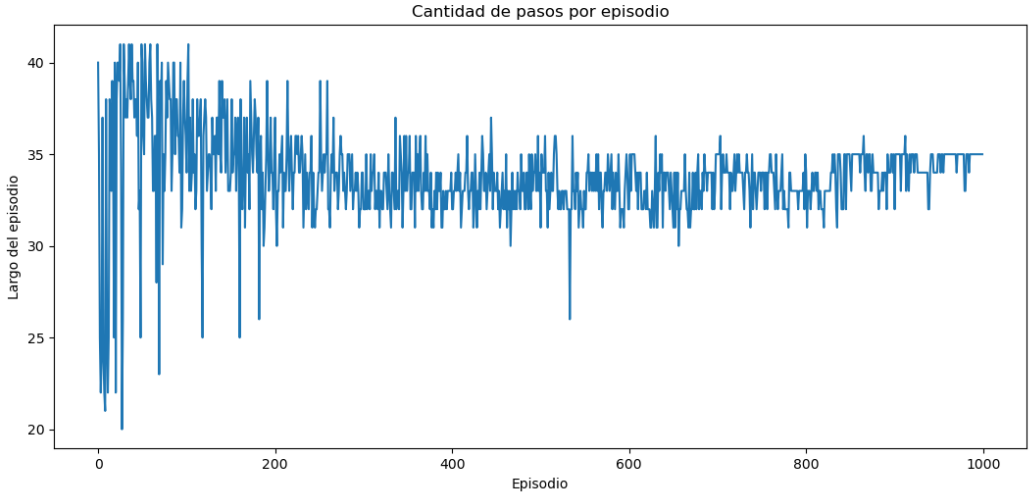

Figura C.67: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración M1

C.3.16. Configuración M2

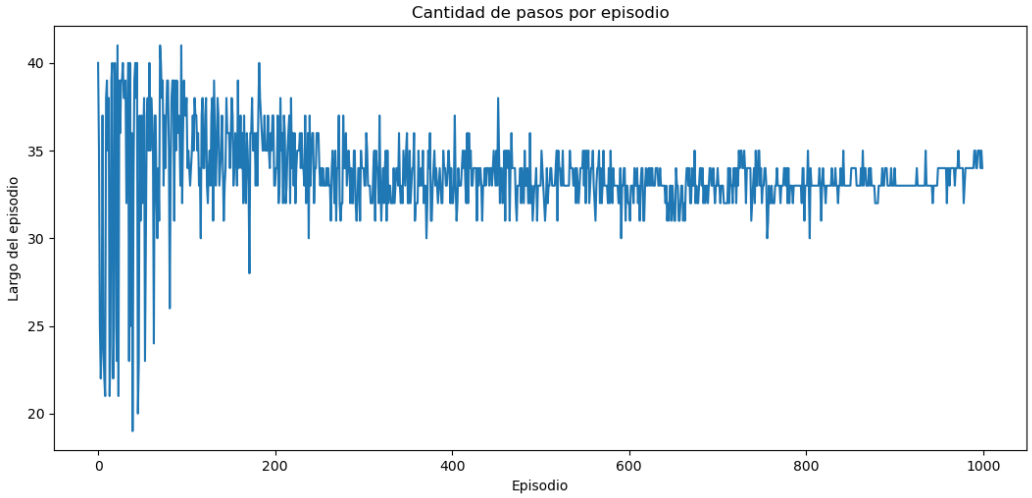

Figura C.68: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración M2

C.3.17. Configuración M3

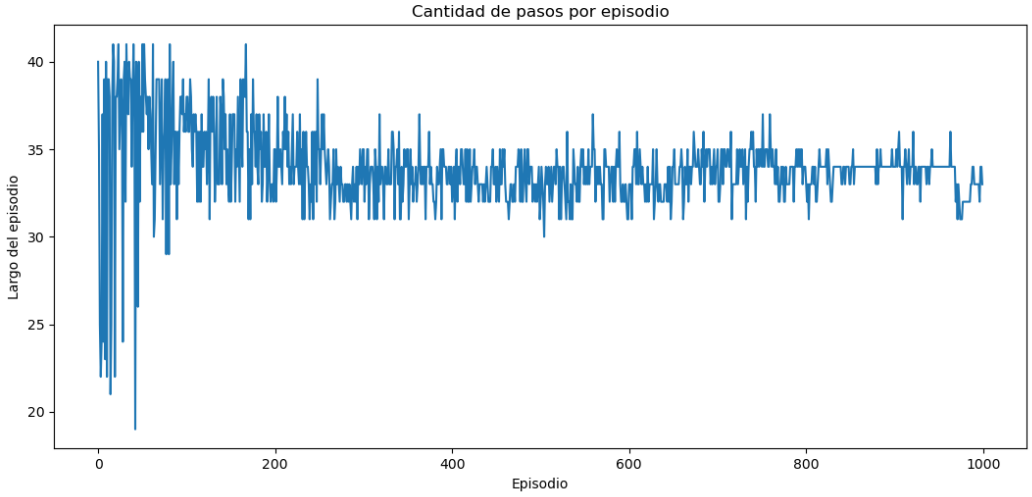

Figura C.69: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración M3

C.3.18. Configuración N1

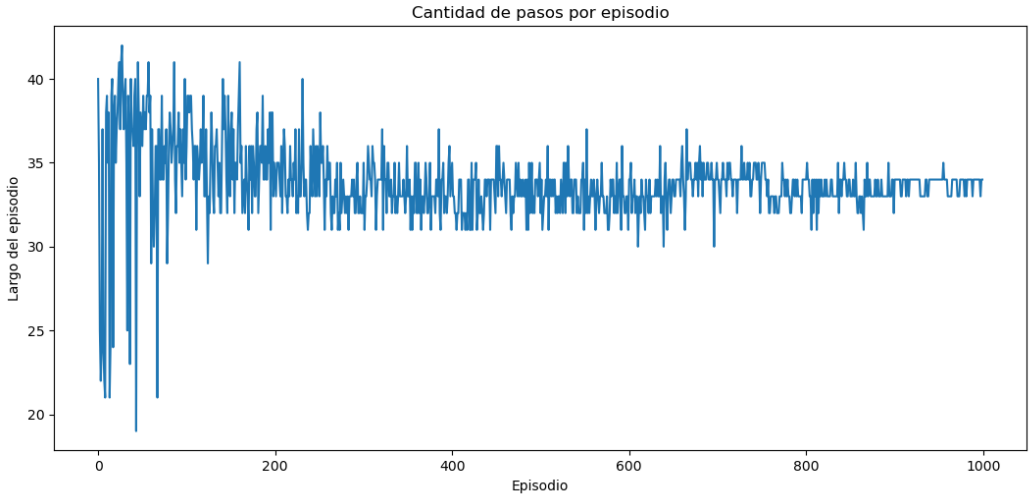

Figura C.70: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración N1

C.3.19. Configuración N2

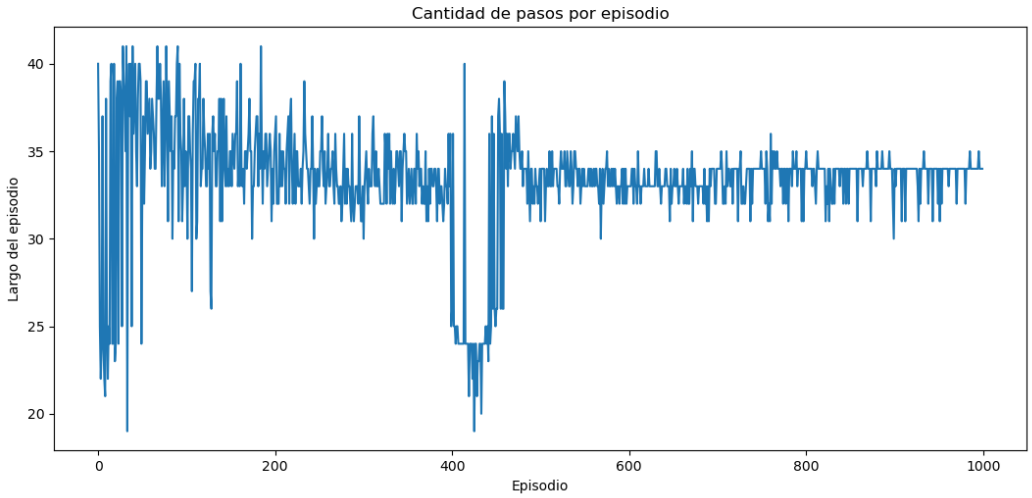

Figura C.71: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración N2

C.3.20. Configuración N3

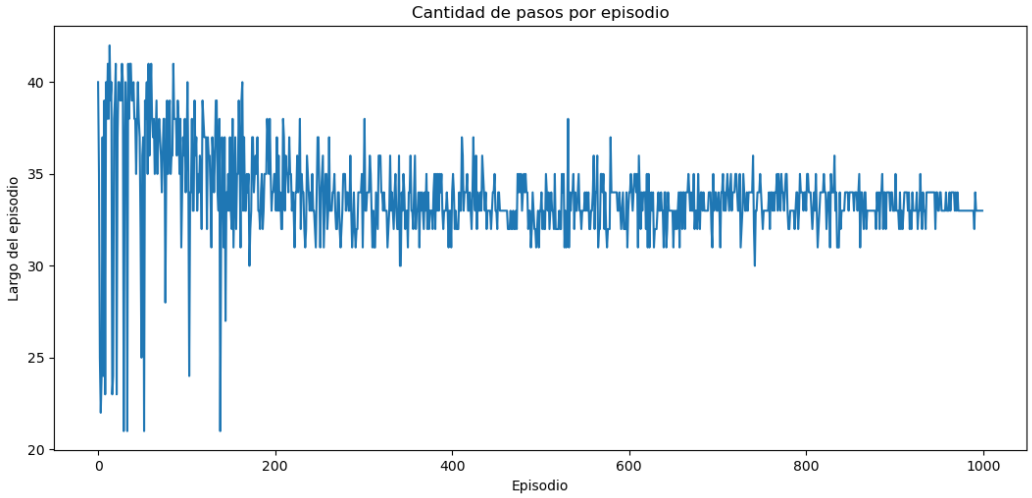

Figura C.72: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración N3

C.3.21. Configuración Ñ1

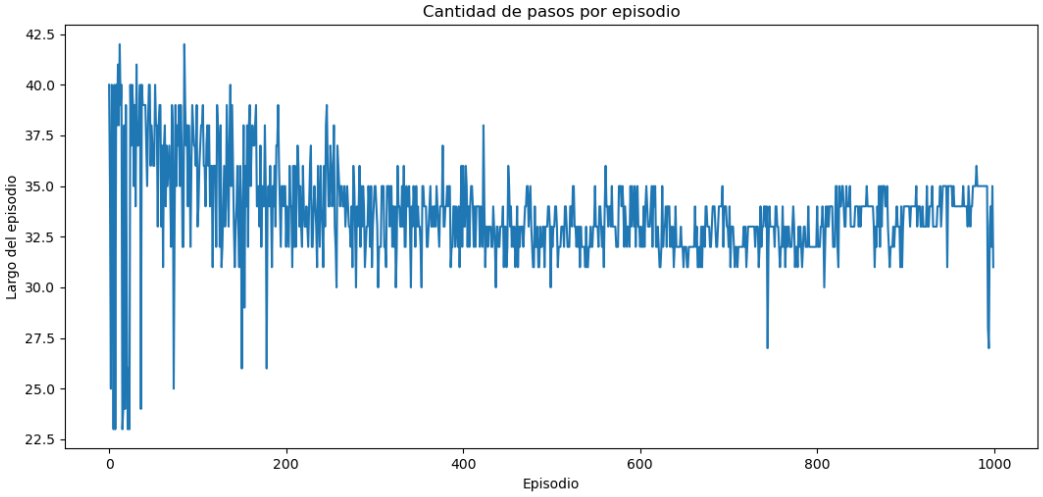

Figura C.73: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración Ñ1

C.3.22. Configuración Ñ2

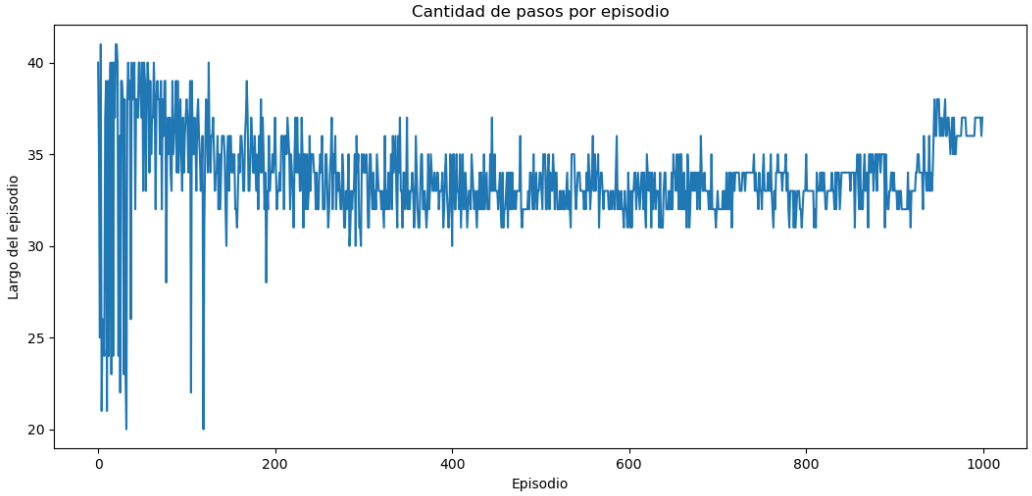

Figura C.74: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración Ñ2

C.3.23. Configuración Ñ3

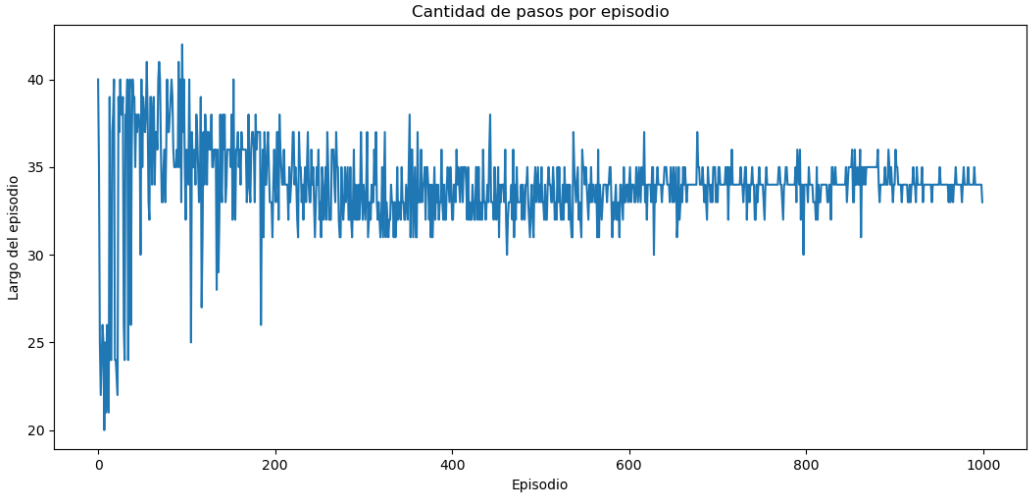

Figura C.75: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración Ñ3

C.3.24. Configuración O1

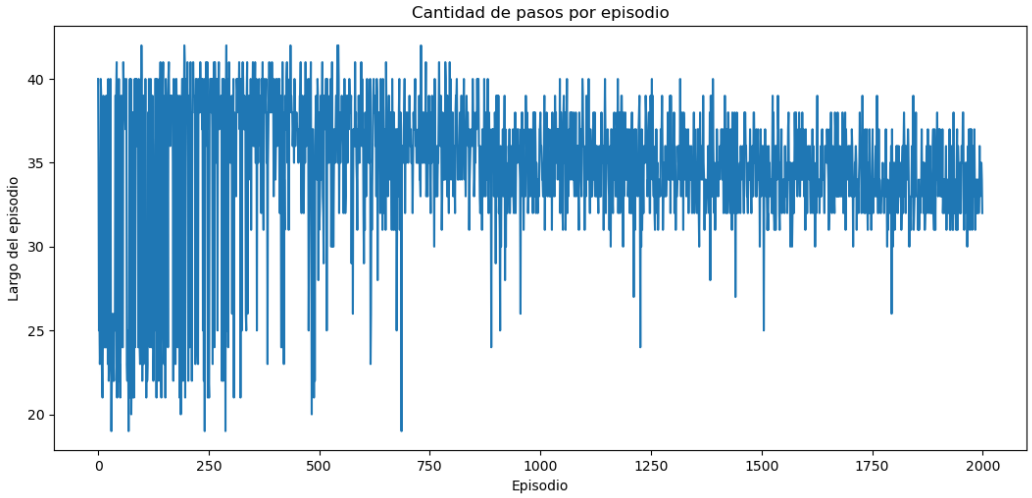

Figura C.76: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración O1

C.3.25. Configuración O2

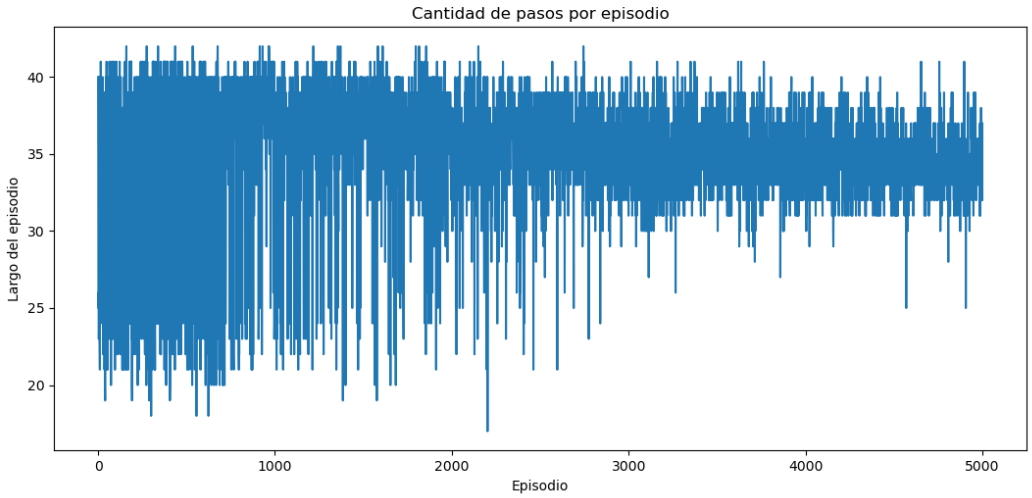

Figura C.77: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración O2

C.3.26. Configuración O3

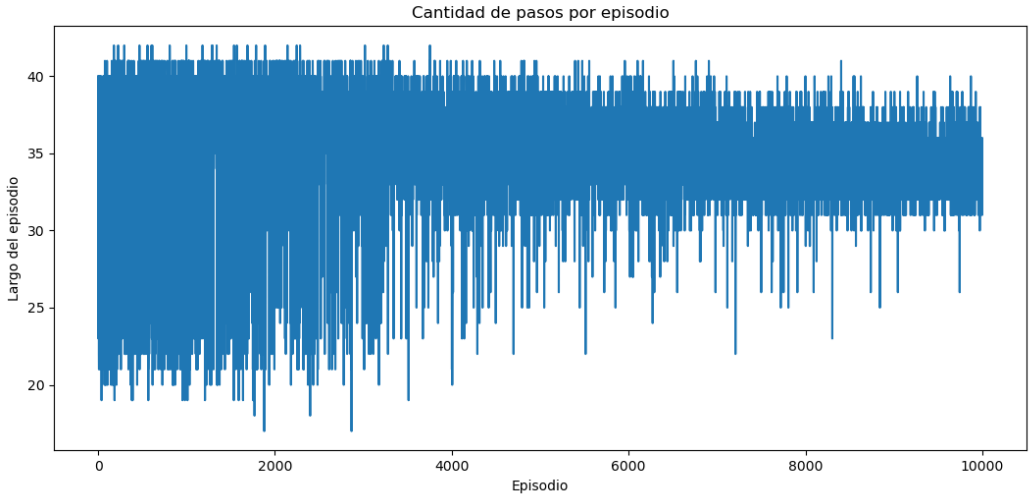

Figura C.78: Largo de episodios medidos en cantidad de pasos obtenidos de modelo basado en configuración O3

C.4. Promedio de Q-values cada $\mathcal{N}$ pasos

C.4.1. Configuración A

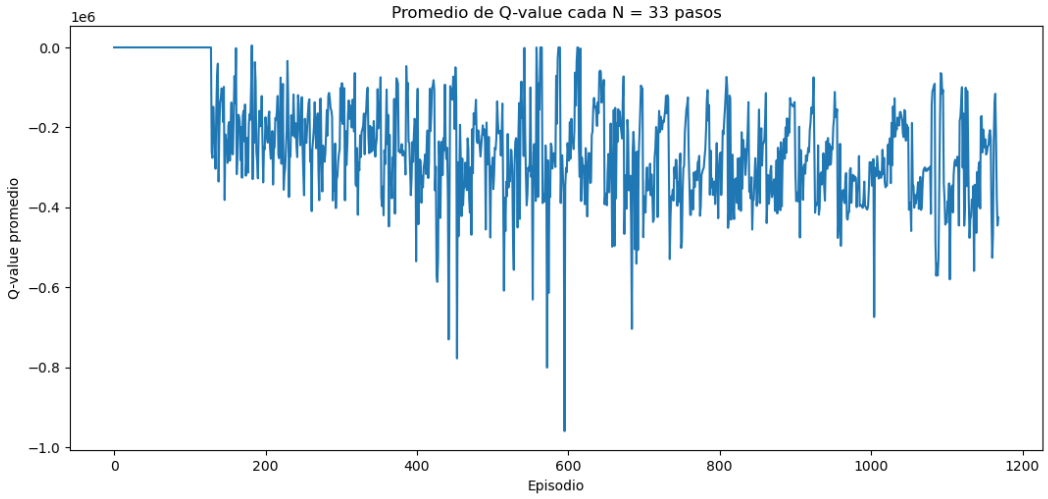

Figura C.79: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración A

C.4.2. Configuración B

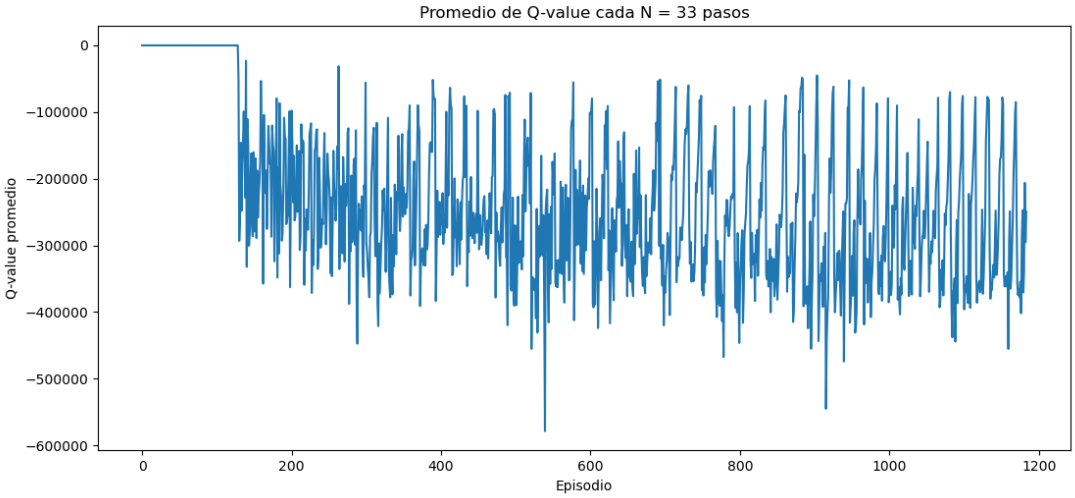

Figura C.80: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración B

C.4.3. Configuración C

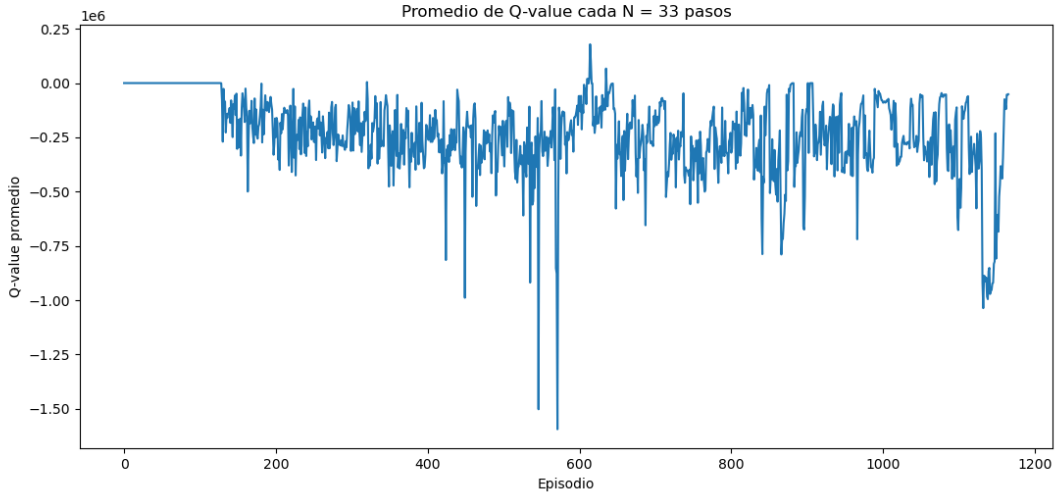

Figura C.81: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración C

C.4.4. Configuración D

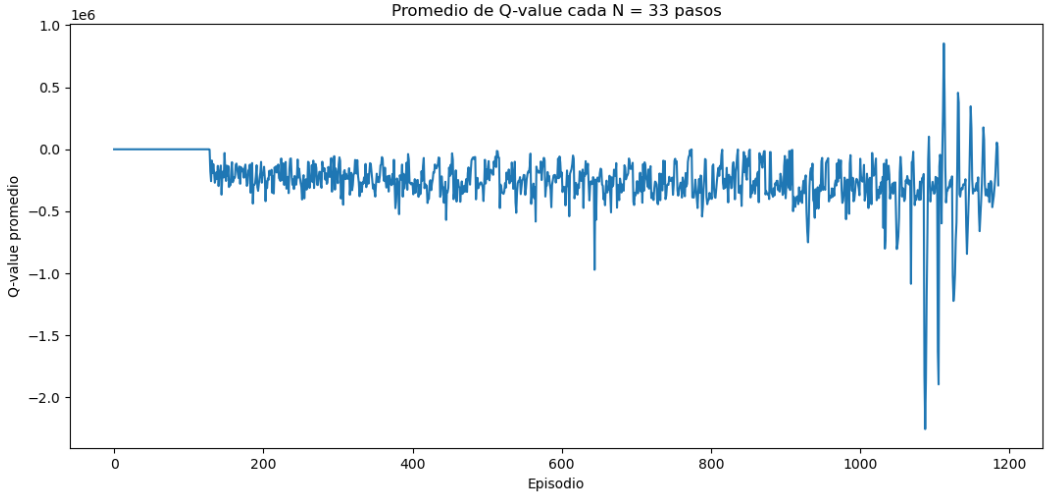

Figura C.82: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración D

C.4.5. Configuración E

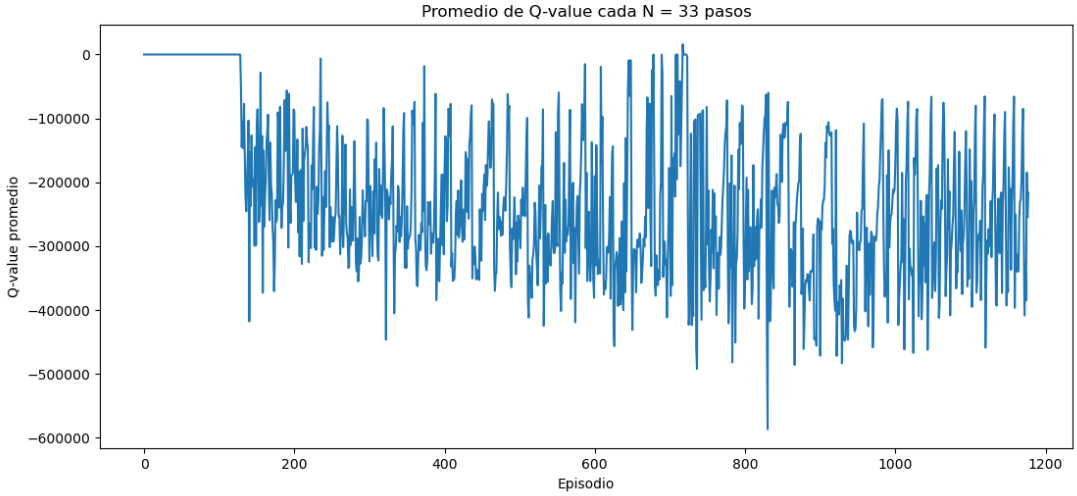

Figura C.83: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración E

C.4.6. Configuración F

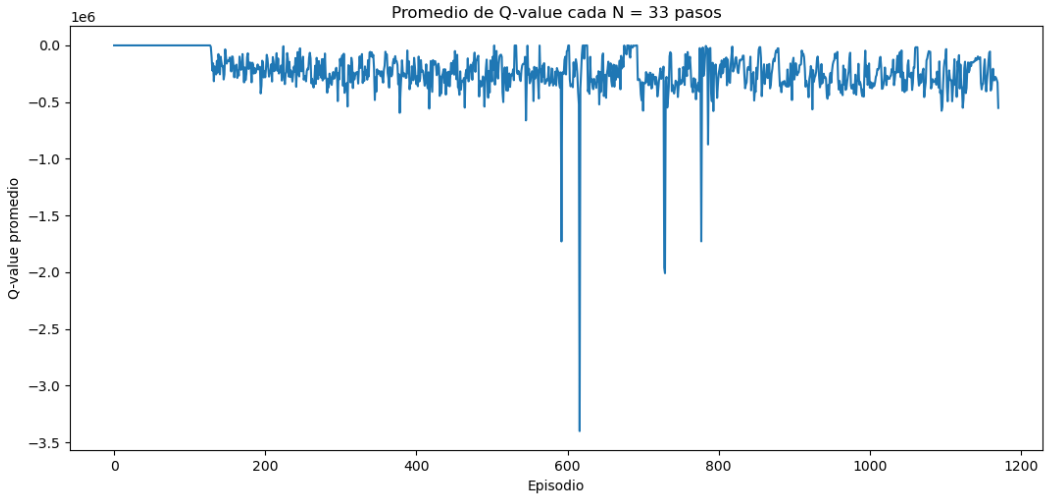

Figura C.84: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración F

C.4.7. Configuración G

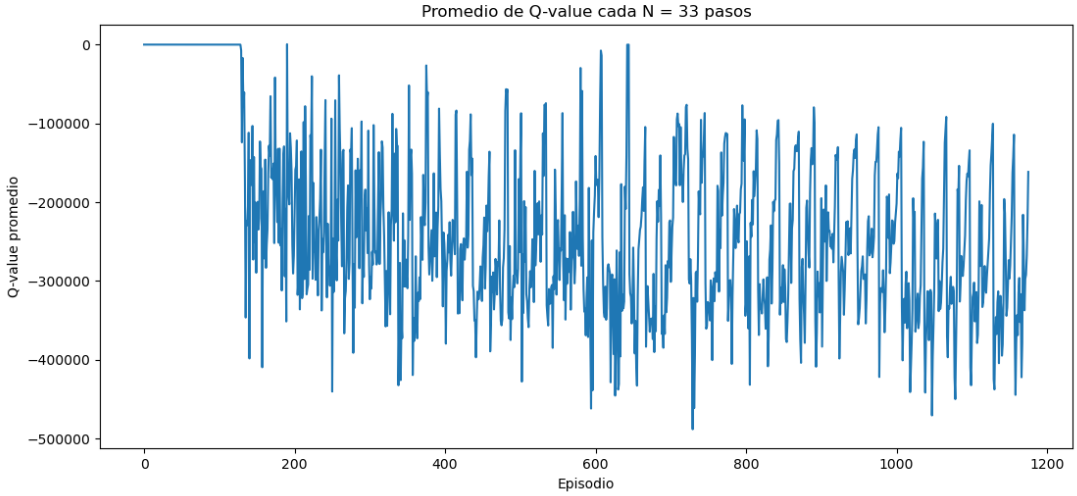

Figura C.85: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración G

C.4.8. Configuración H

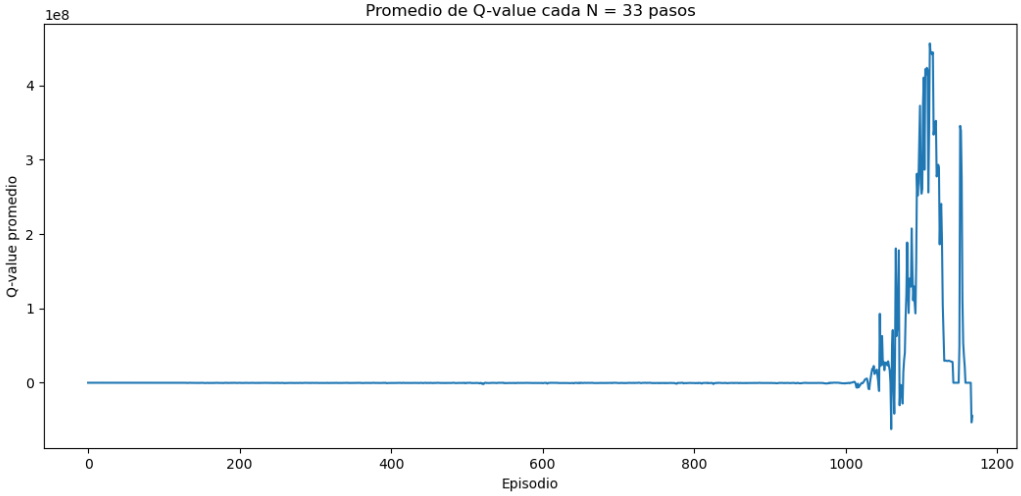

Figura C.86: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración H

C.4.9. Configuración I

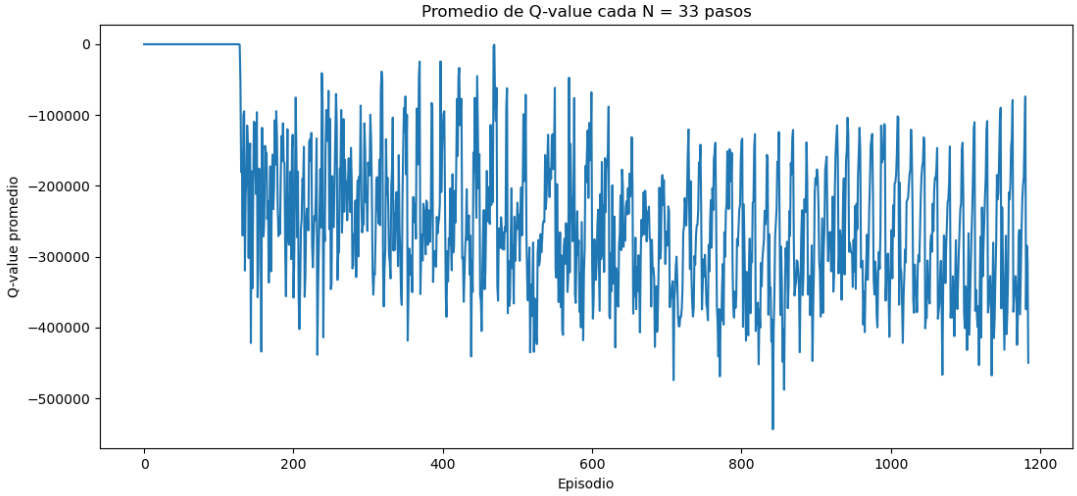

Figura C.87: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración I

C.4.10. Configuración J

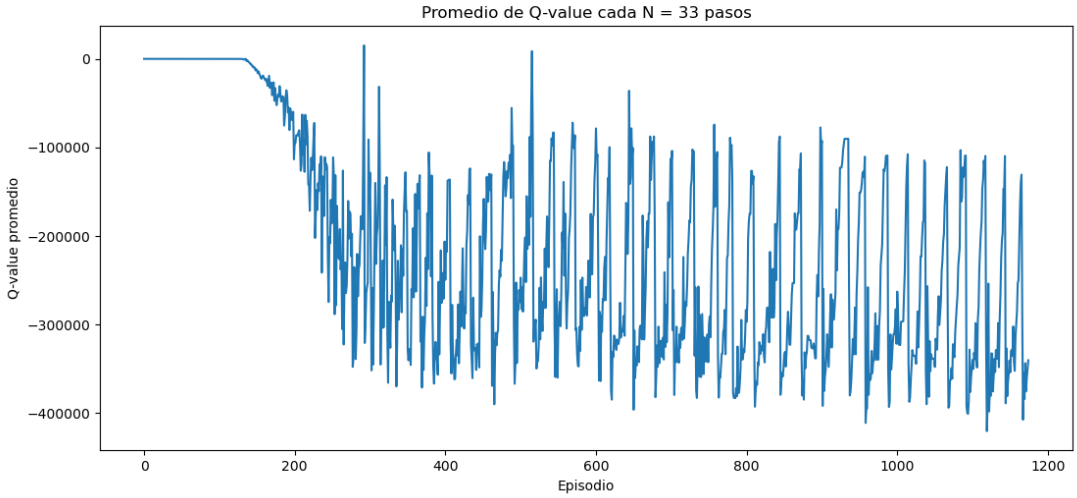

Figura C.88: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración J

C.4.11. Configuración K

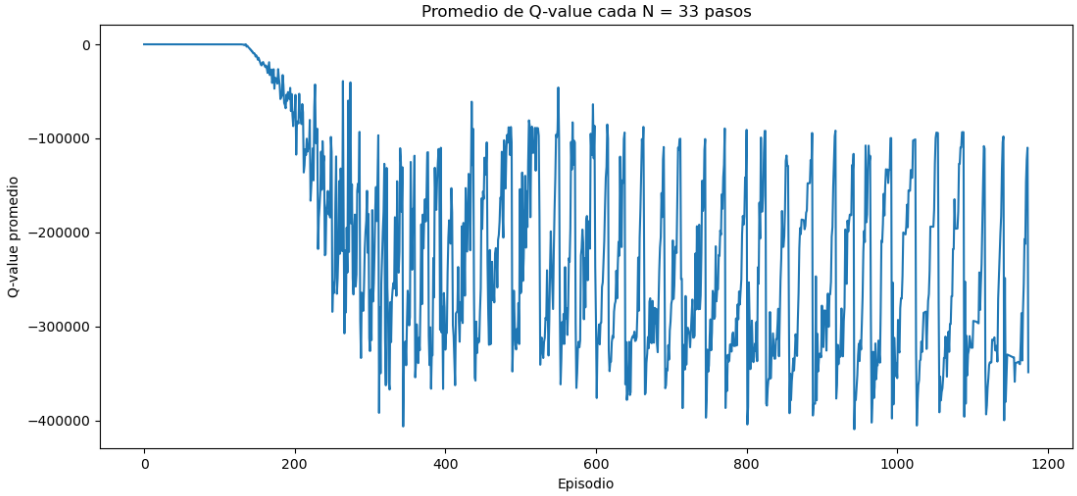

Figura C.89: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración K

C.4.12. Configuración L1

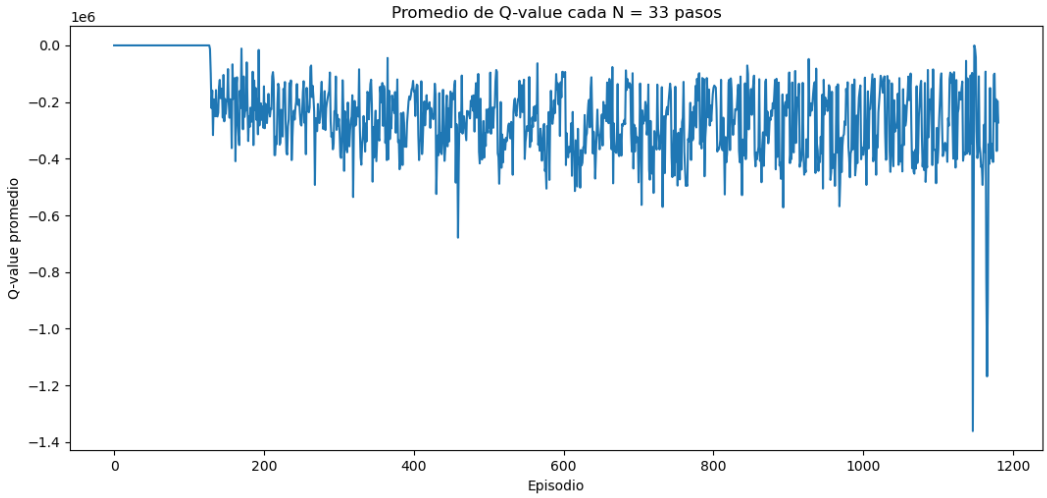

Figura C.90: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración L1

C.4.13. Configuración L2

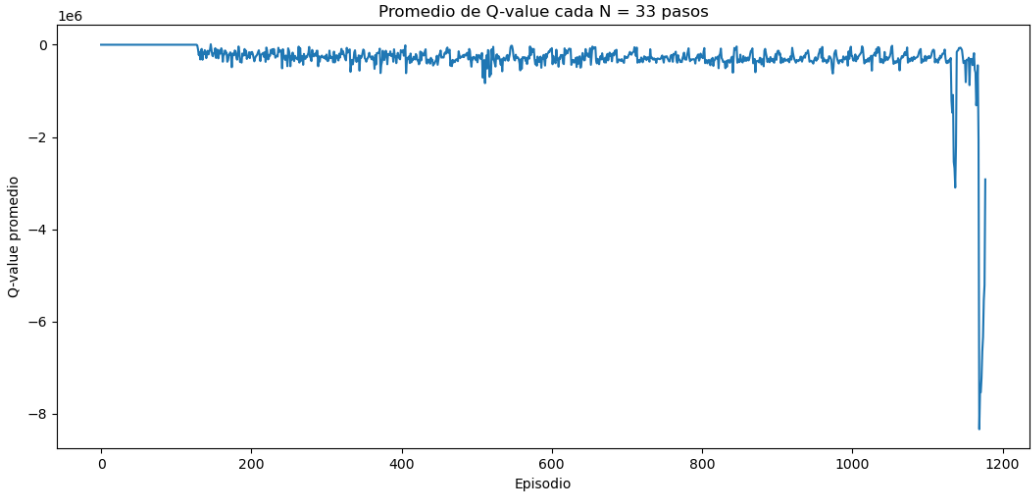

Figura C.91: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración L2

C.4.14. Configuración L3

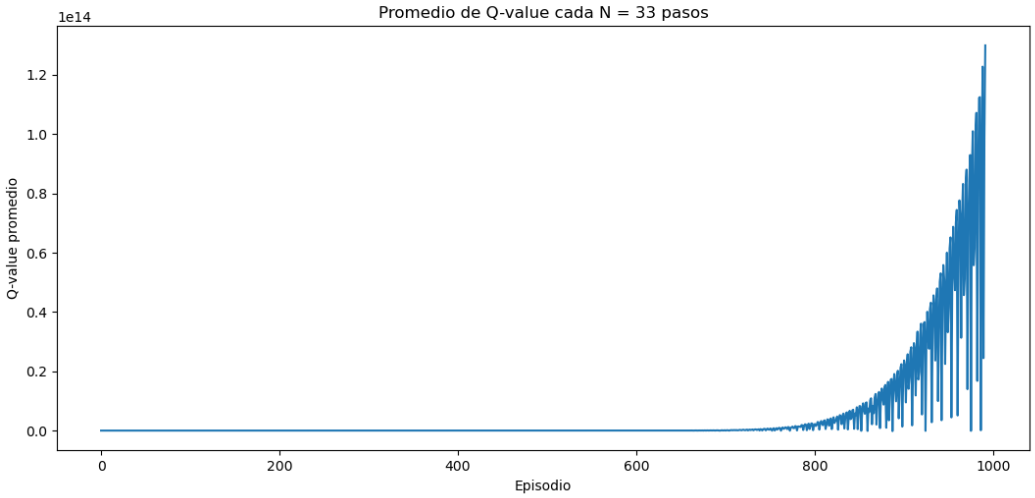

Figura C.92: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración L3

C.4.15. Configuración M1

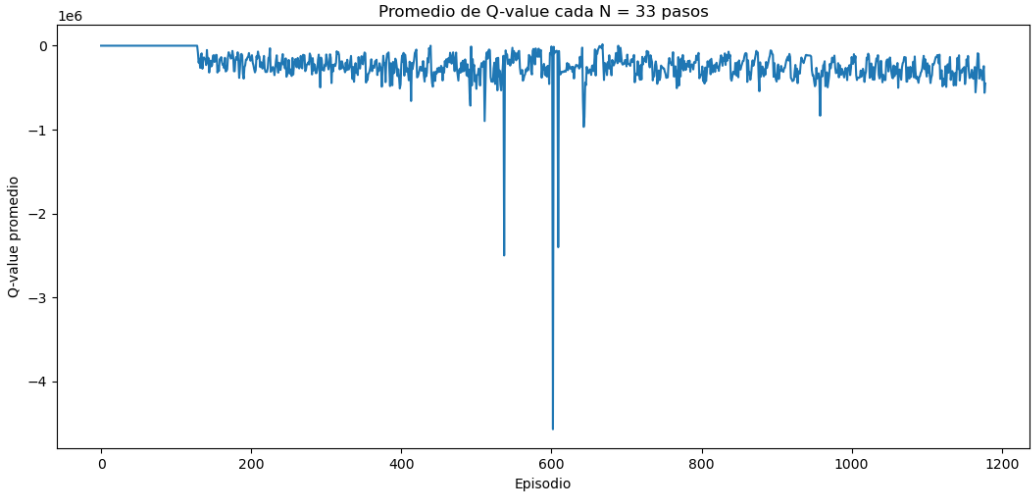

Figura C.93: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración M1

C.4.16. Configuración M2

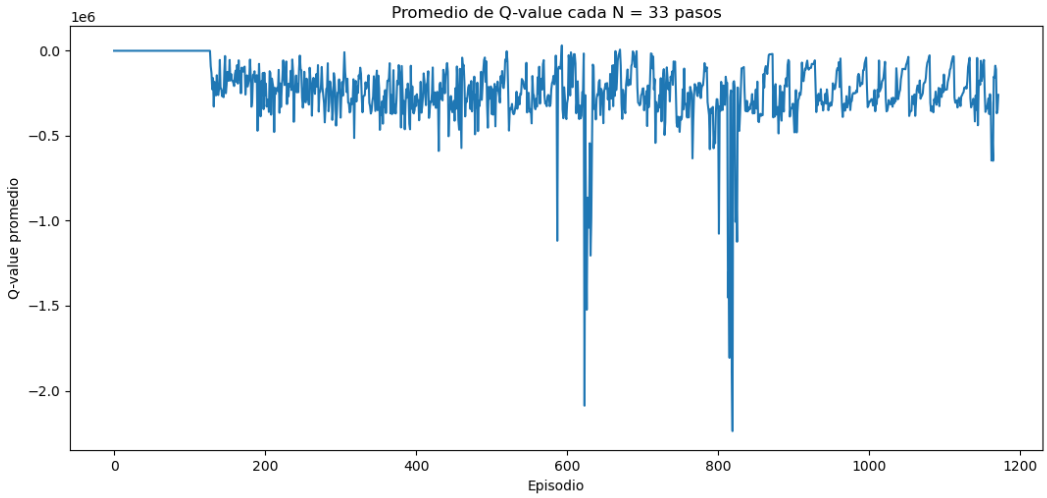

Figura C.94: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración M2

C.4.17. Configuración M3

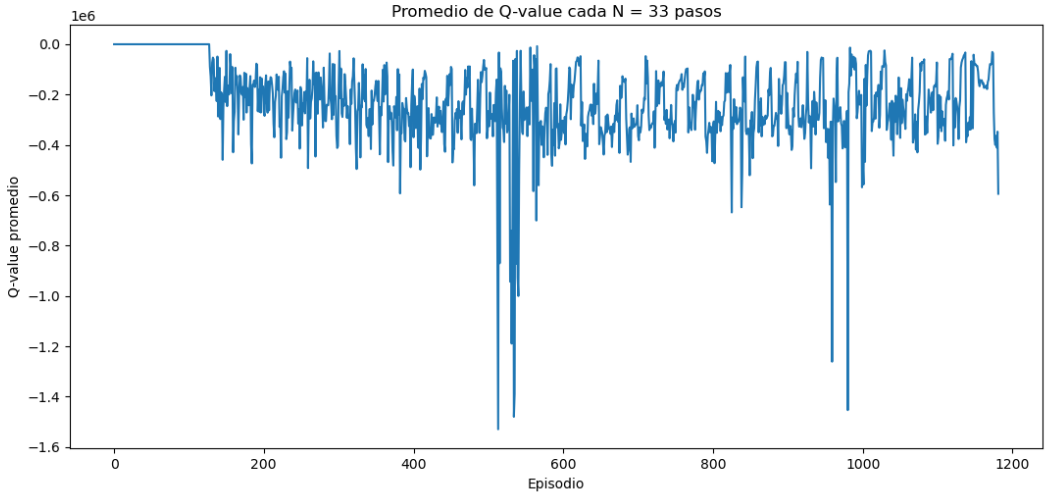

Figura C.95: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración M3

C.4.18. Configuración N1

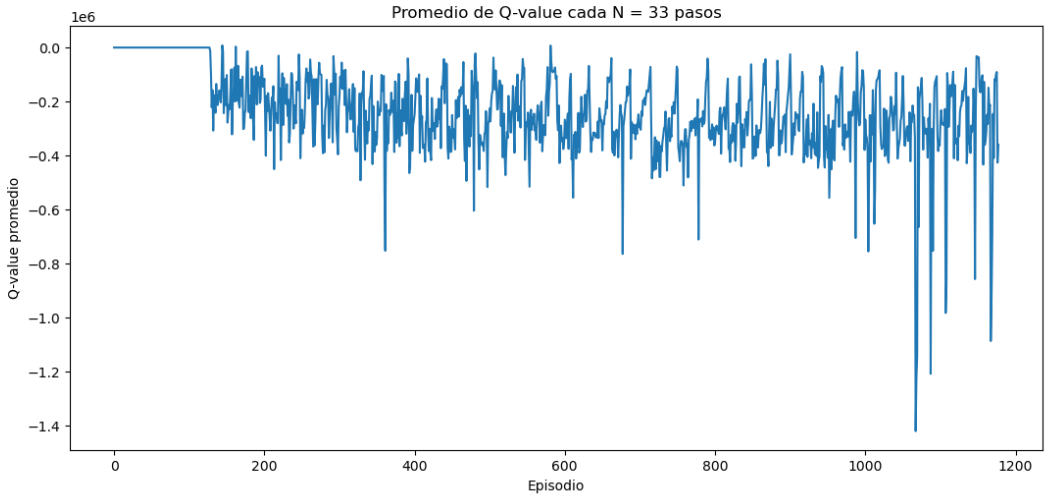

Figura C.96: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración N1

C.4.19. Configuración N2

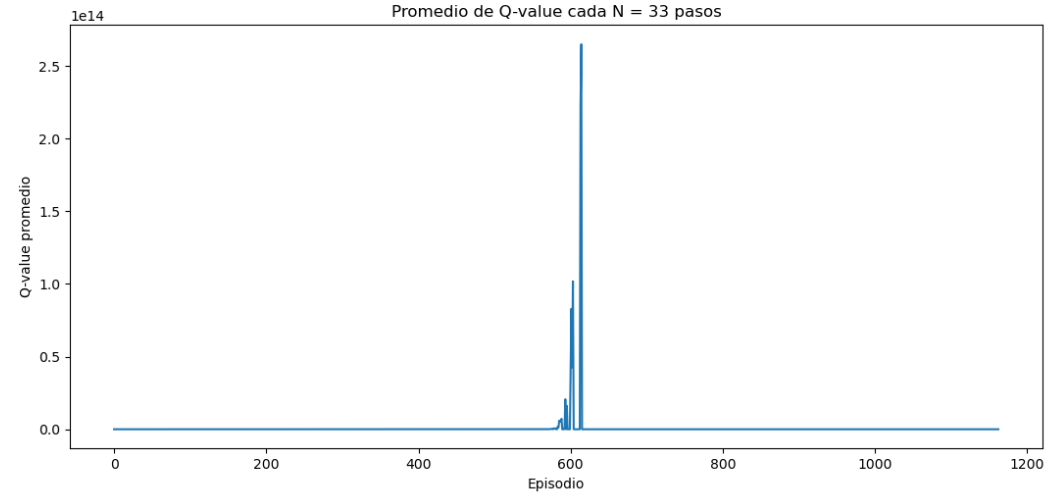

Figura C.97: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración N2

C.4.20. Configuración N3

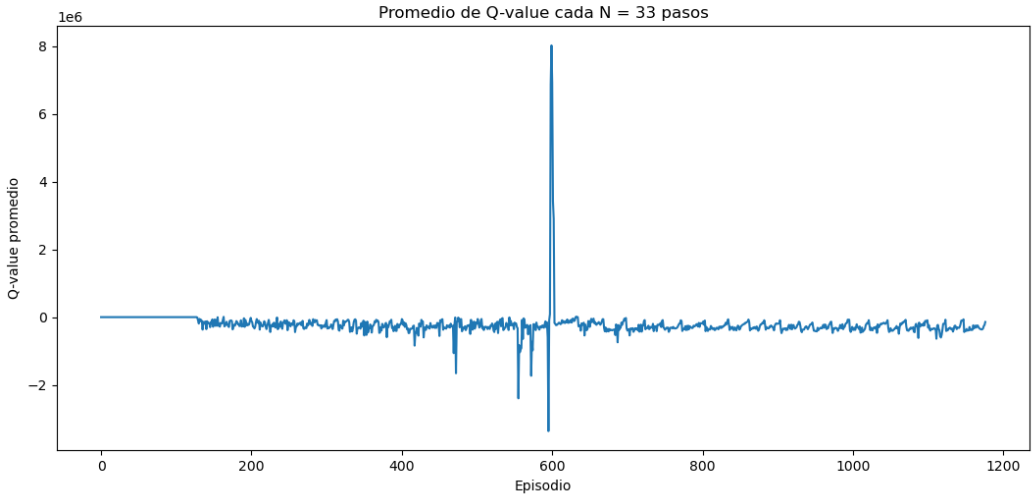

Figura C.98: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración N3

C.4.21. Configuración Ñ1

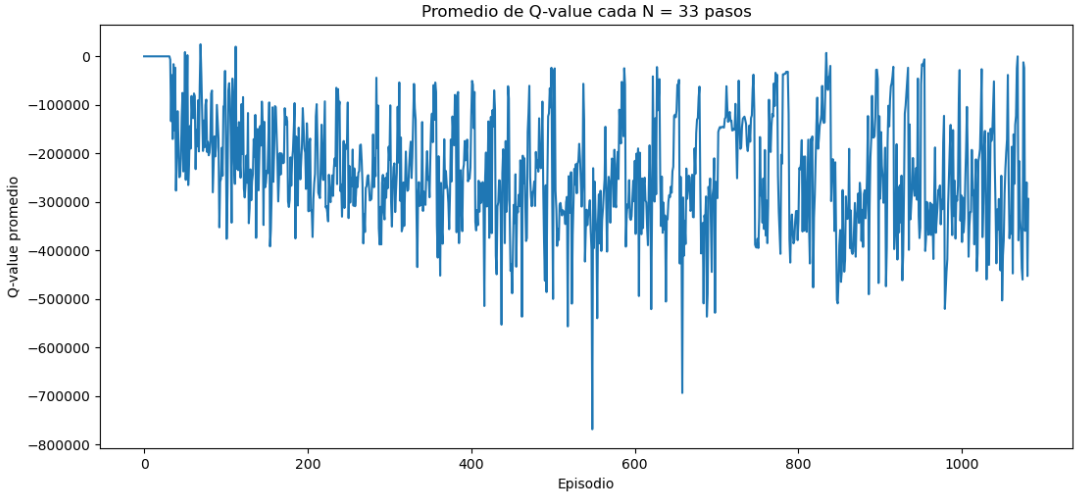

Figura C.99: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración Ñ1

C.4.22. Configuración Ñ2

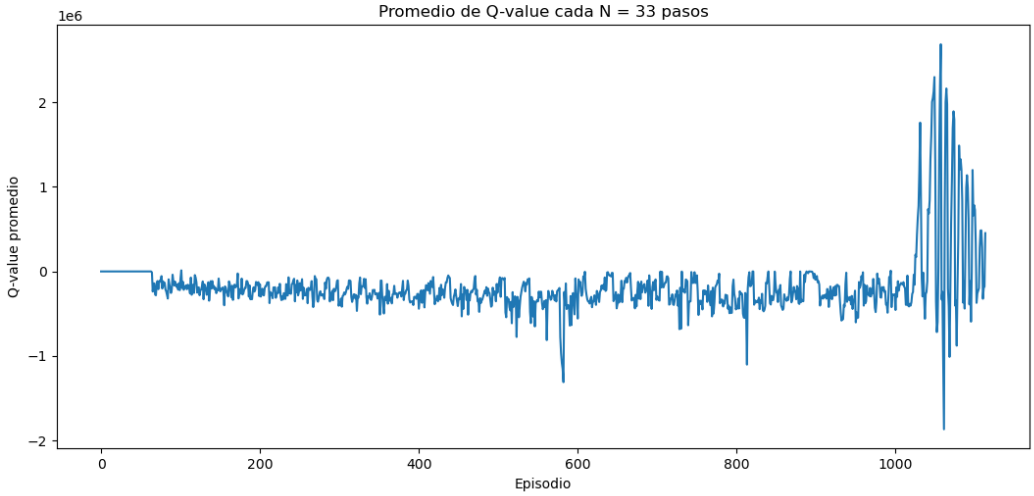

Figura C.100: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración Ñ2

C.4.23. Configuración Ñ3

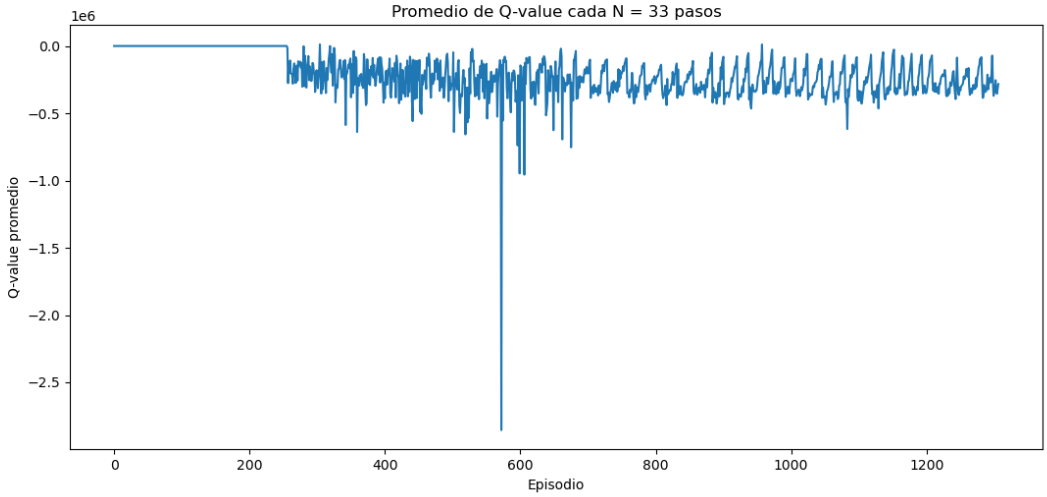

Figura C.101: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración Ñ3

C.4.24. Configuración O1

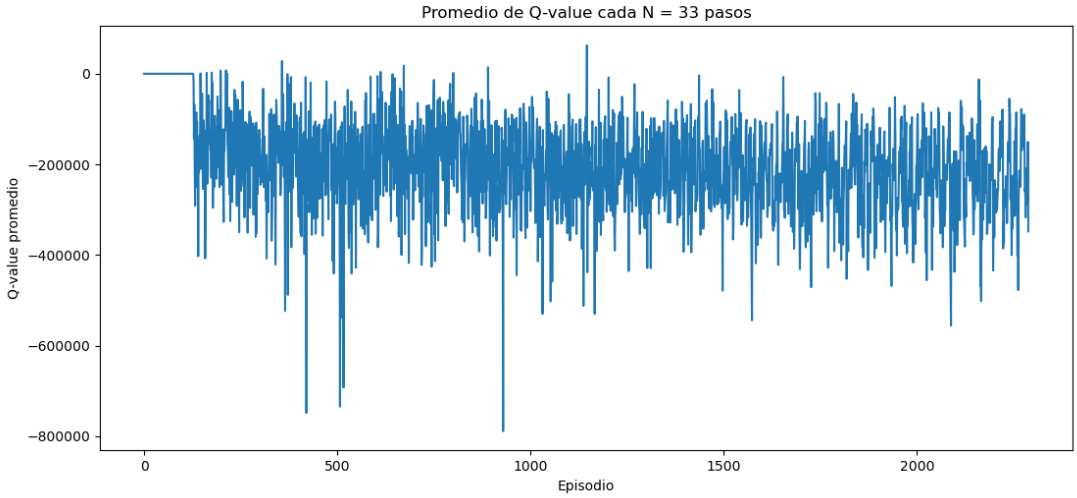

Figura C.102: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración O1

C.4.25. Configuración O2

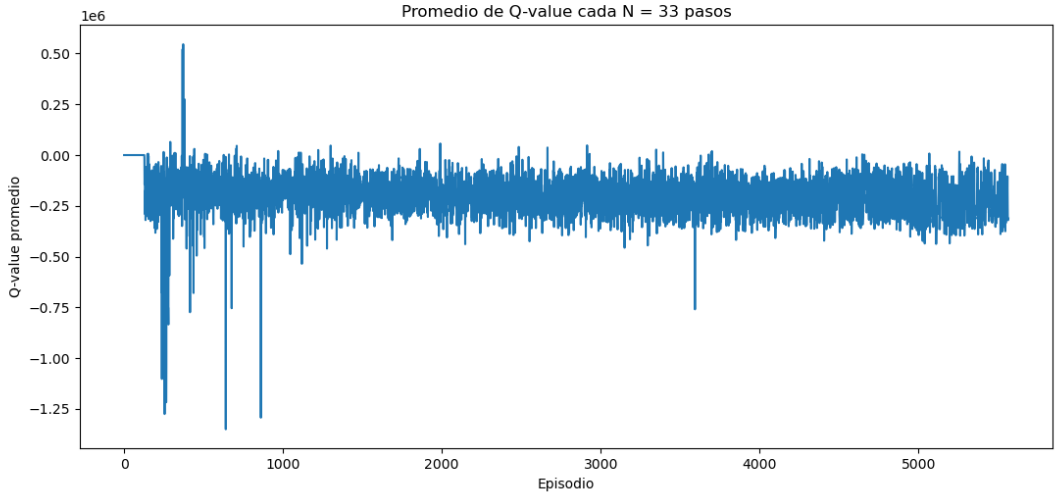

Figura C.103: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración O2

C.4.26. Configuración O3

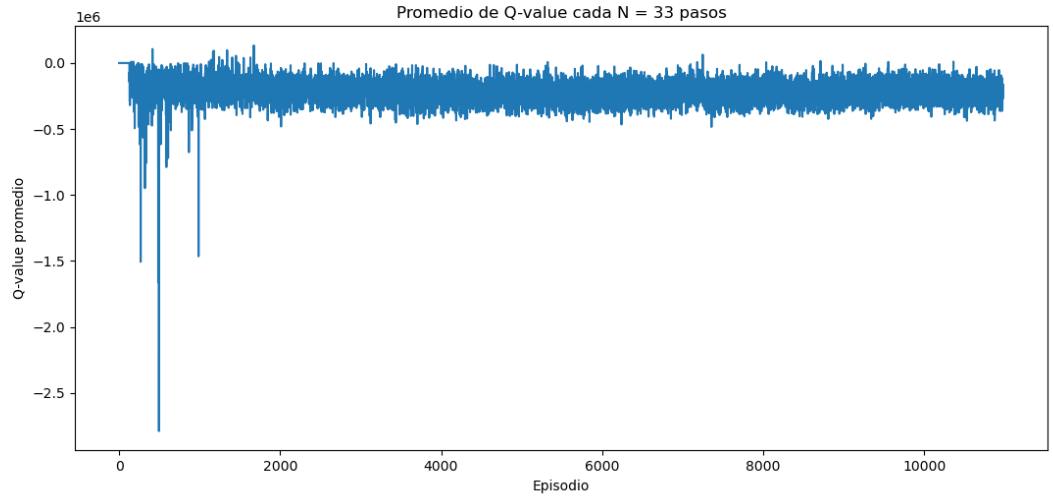

Figura C.104: Q-valor promedio medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración O3

C.5. Valor de función de pérdida $\mathcal{L}$ cada $\mathcal{N}$ pasos

C.5.1. Configuración A

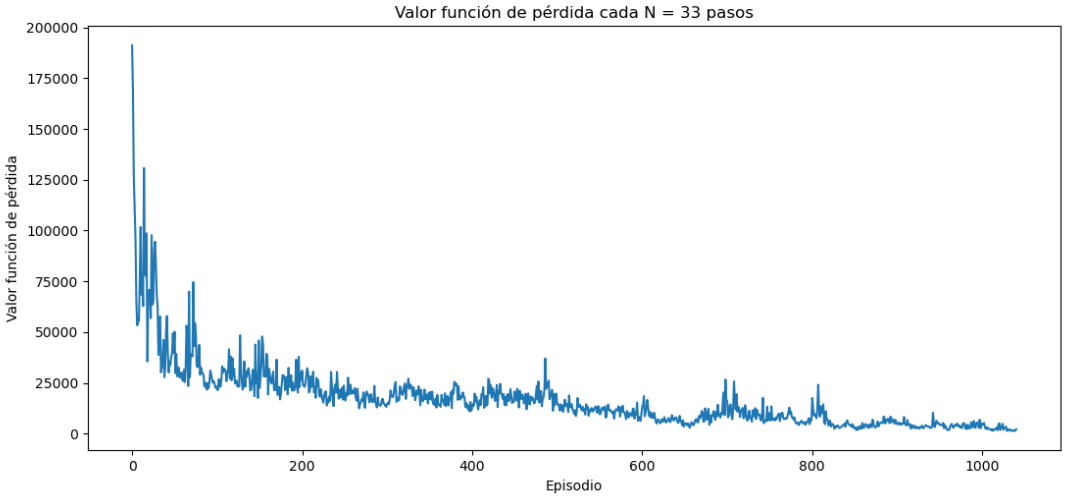

Figura C.105: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración A

C.5.2. Configuración B

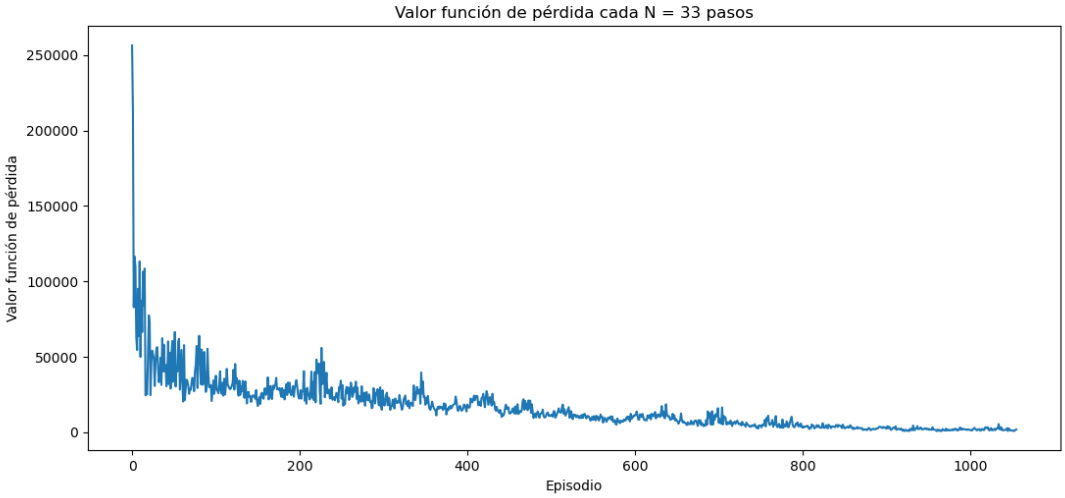

Figura C.106: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración B

C.5.3. Configuración C

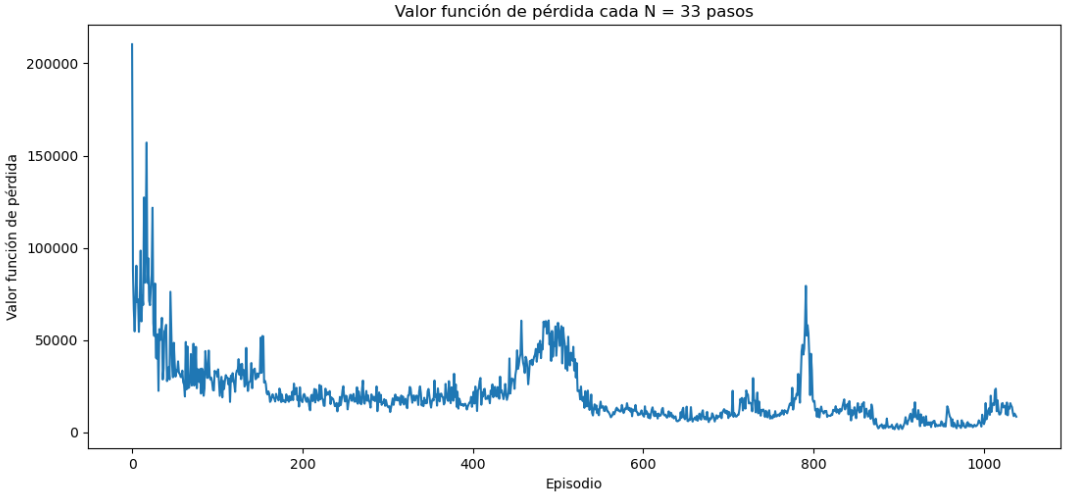

Figura C.107: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración C

C.5.4. Configuración D

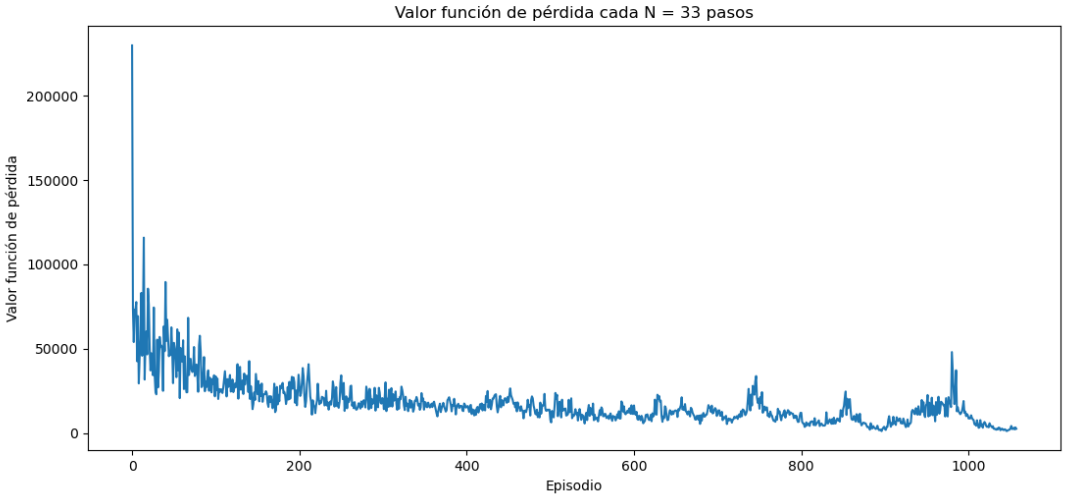

Figura C.108: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración D

C.5.5. Configuración E

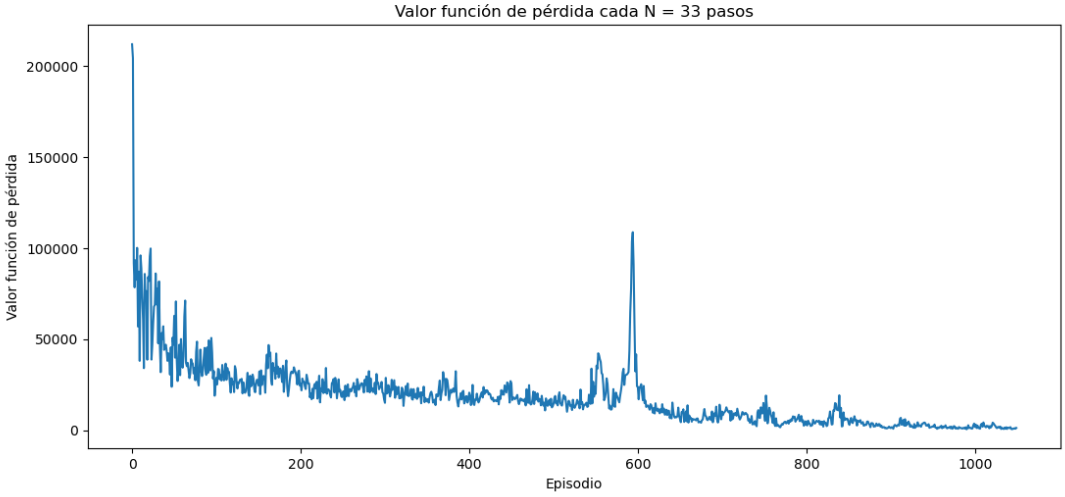

Figura C.109: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración E

C.5.6. Configuración F

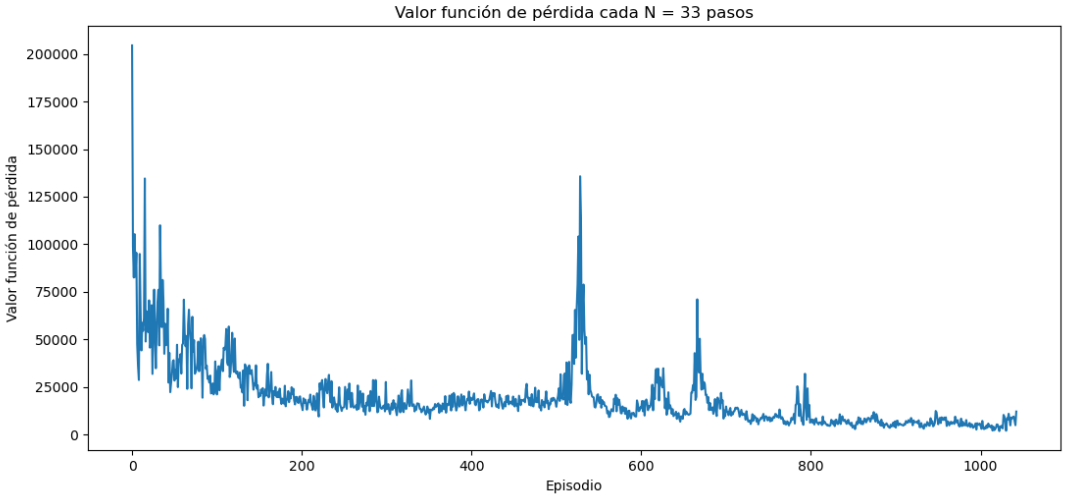

Figura C.110: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración F

C.5.7. Configuración G

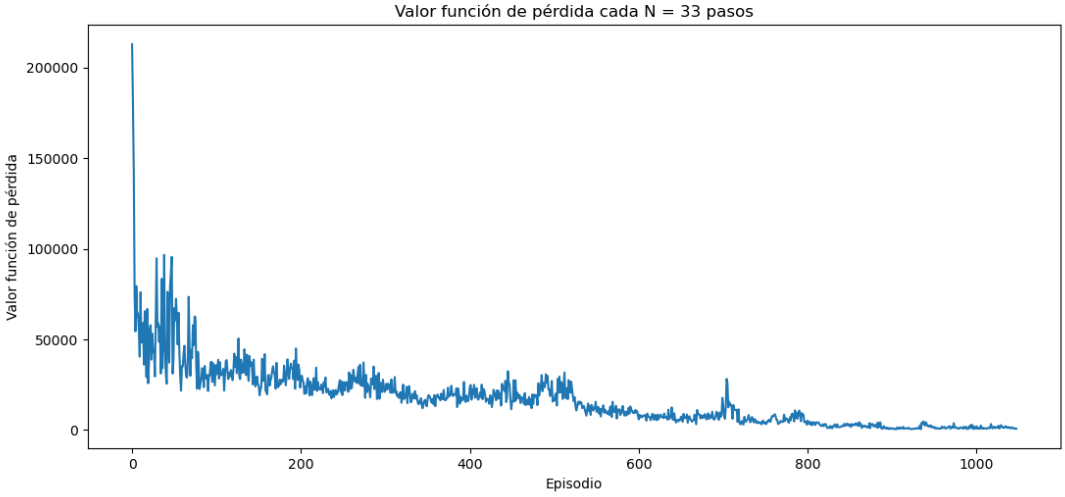

Figura C.111: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración G

C.5.8. Configuración H

Figura C.112: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración H

C.5.9. Configuración I

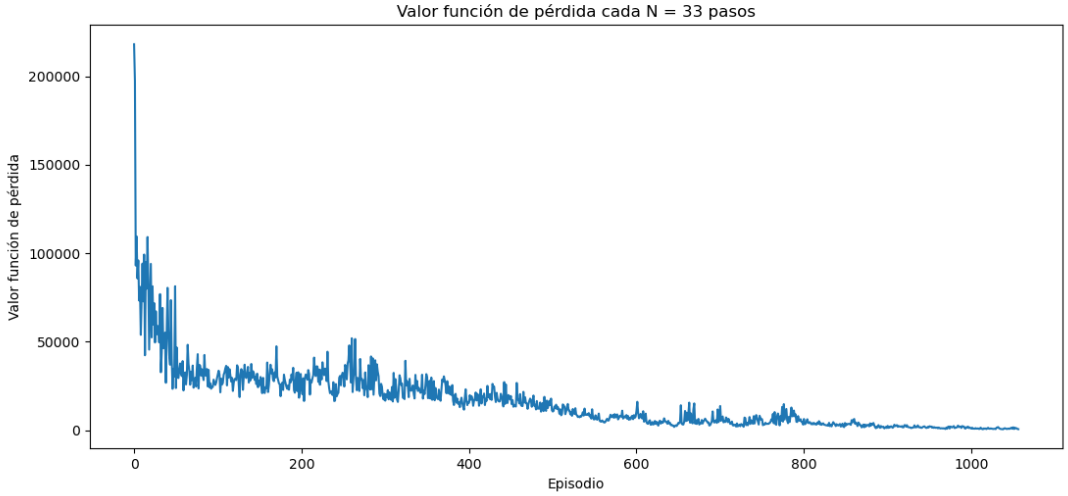

Figura C.113: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración I

C.5.10. Configuración J

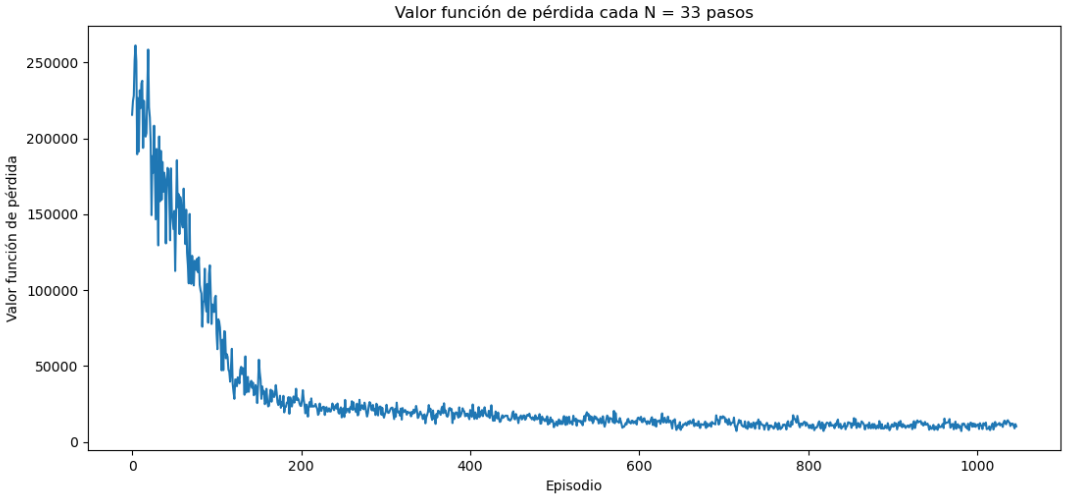

Figura C.114: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración J

C.5.11. Configuración K

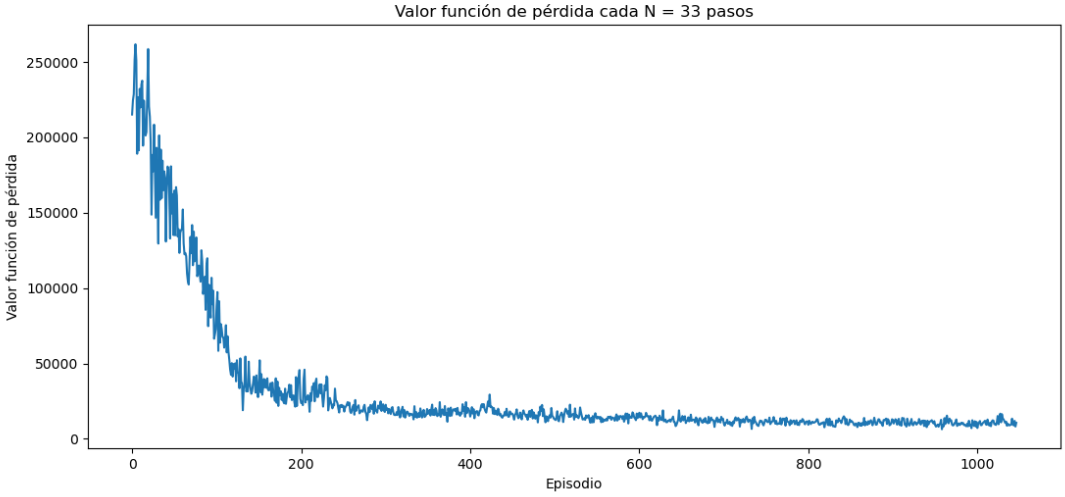

Figura C.115: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración K

C.5.12. Configuración L1

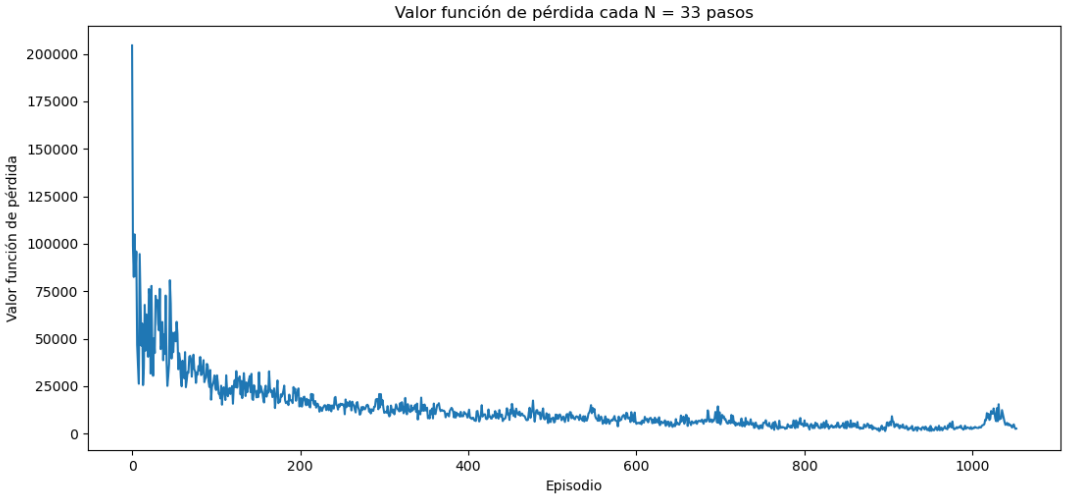

Figura C.116: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración L1

C.5.13. Configuración L2

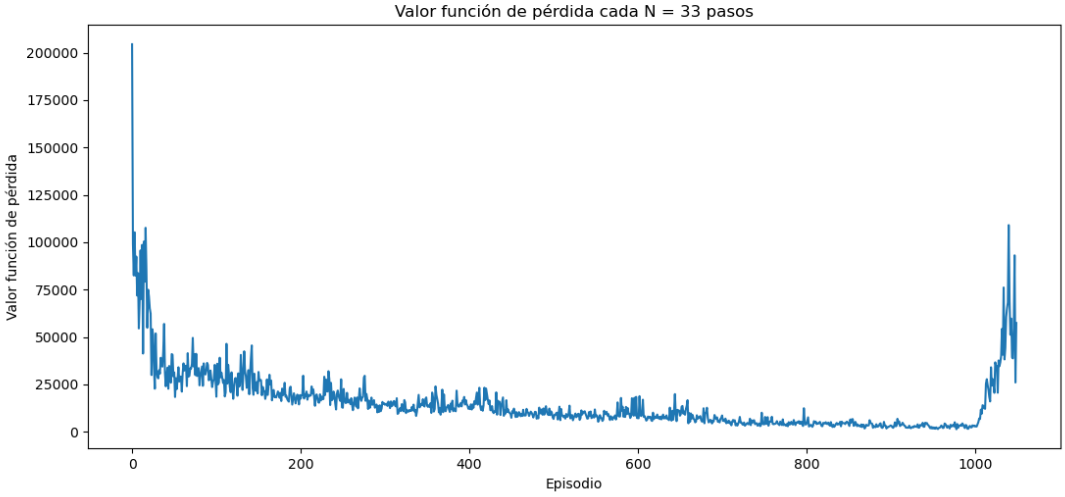

Figura C.117: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración L2

C.5.14. Configuración L3

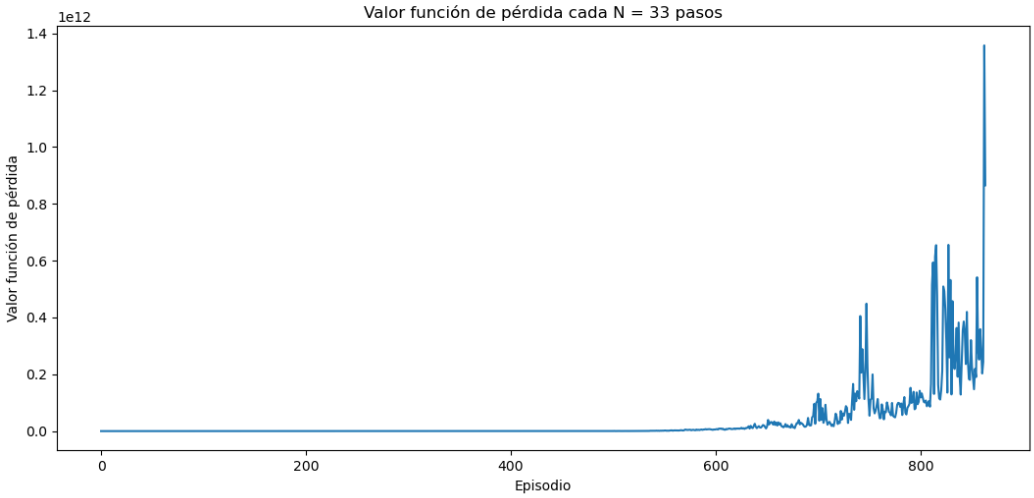

Figura C.118: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración L3

C.5.15. Configuración M1

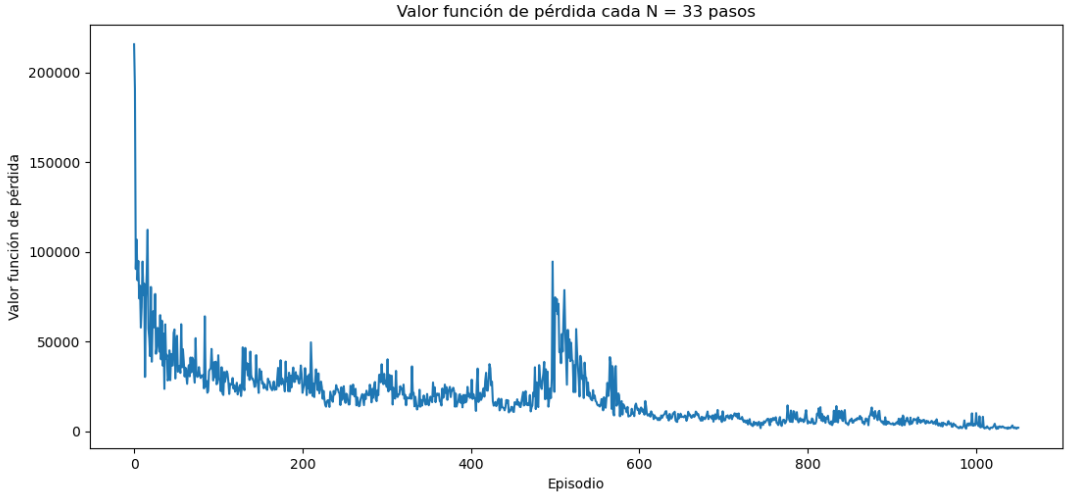

Figura C.119: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración M1

C.5.16. Configuración M2

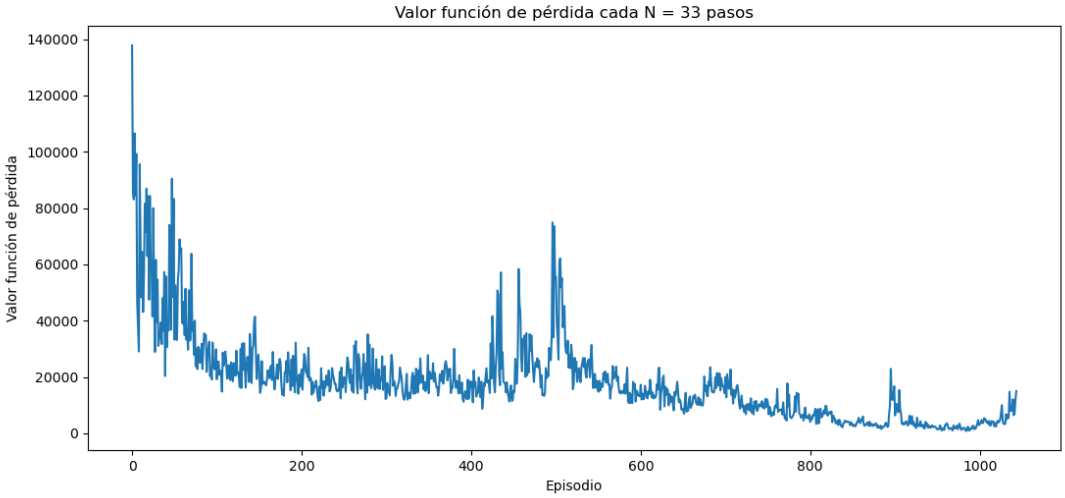

Figura C.120: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración M2

C.5.17. Configuración M3

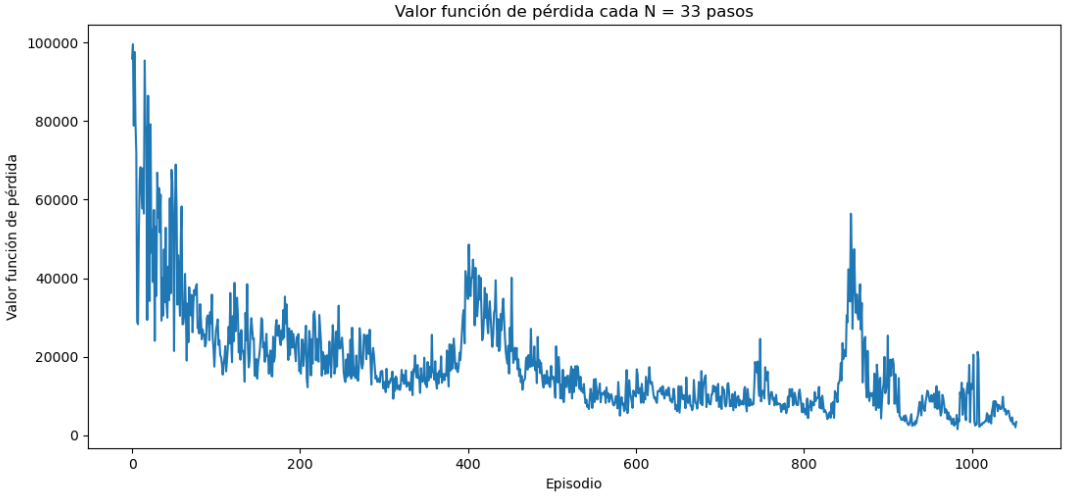

Figura C.121: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración M3

C.5.18. Configuración N1

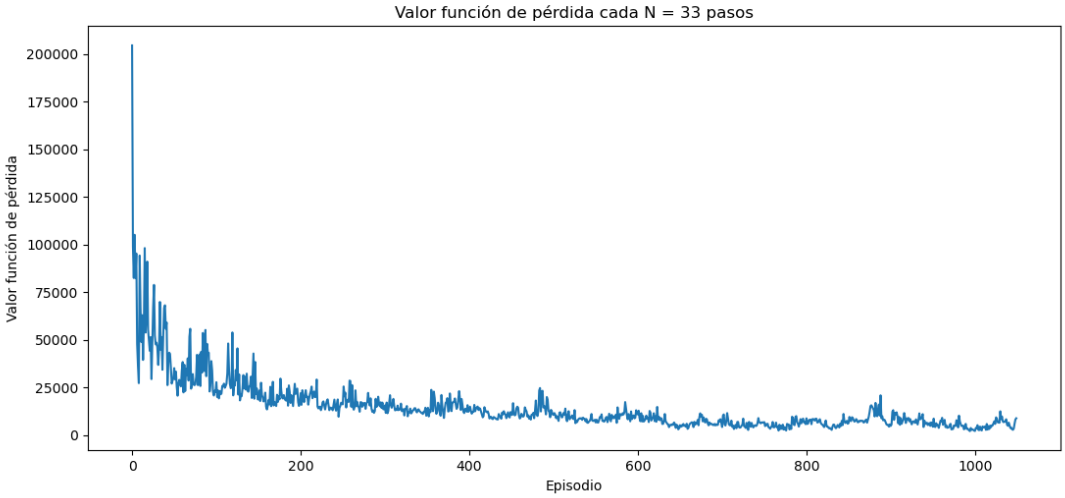

Figura C.122: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración N1

C.5.19. Configuración N2

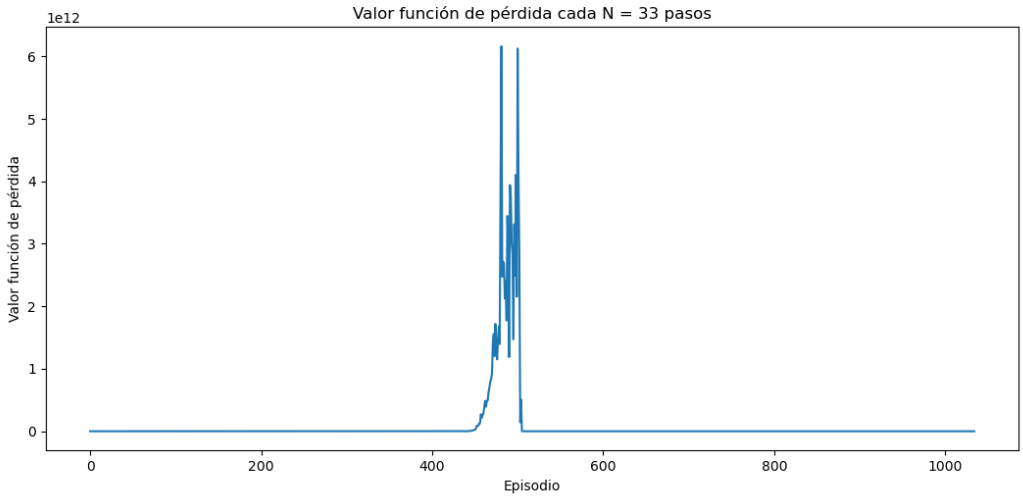

Figura C.123: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración N2

C.5.20. Configuración N3

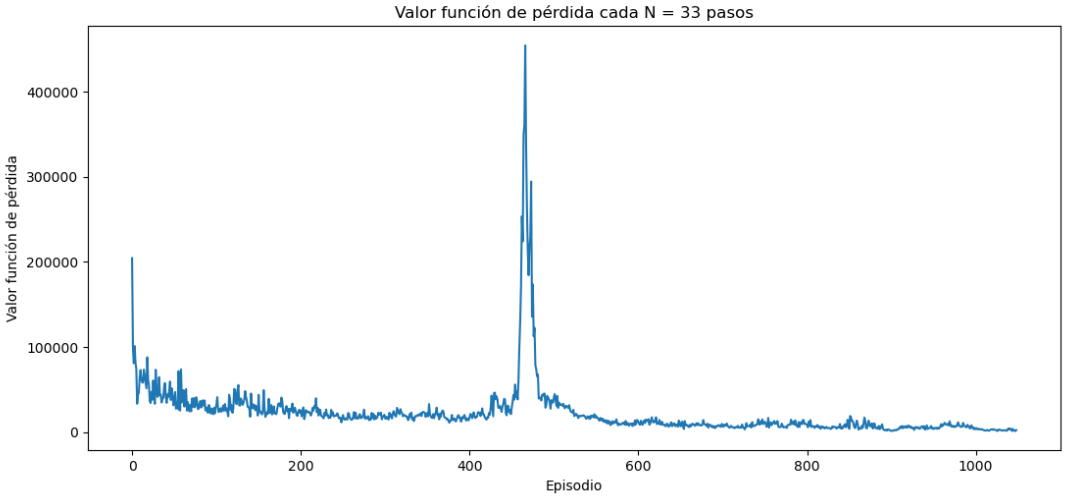

Figura C.124: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración N3

C.5.21. Configuración Ñ1

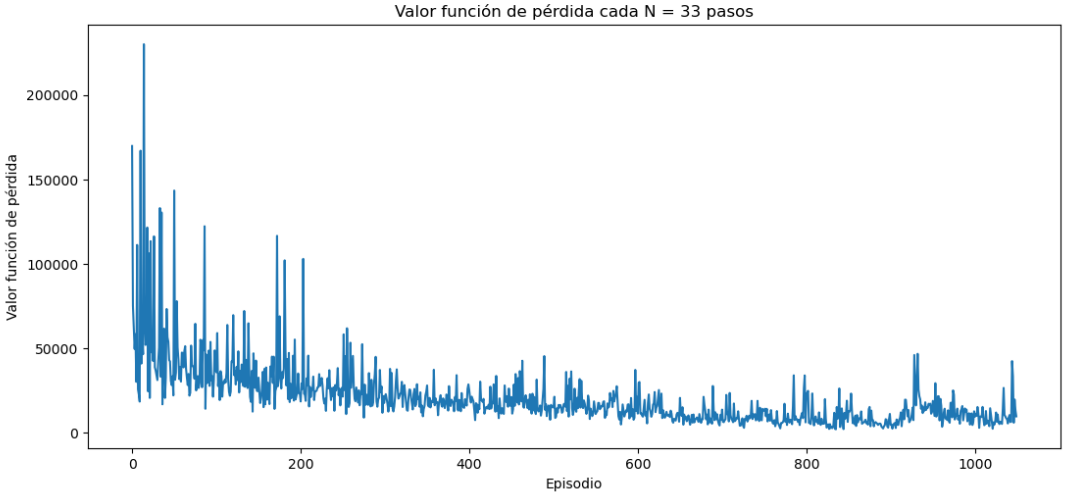

Figura C.125: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración Ñ1

C.5.22. Configuración Ñ2

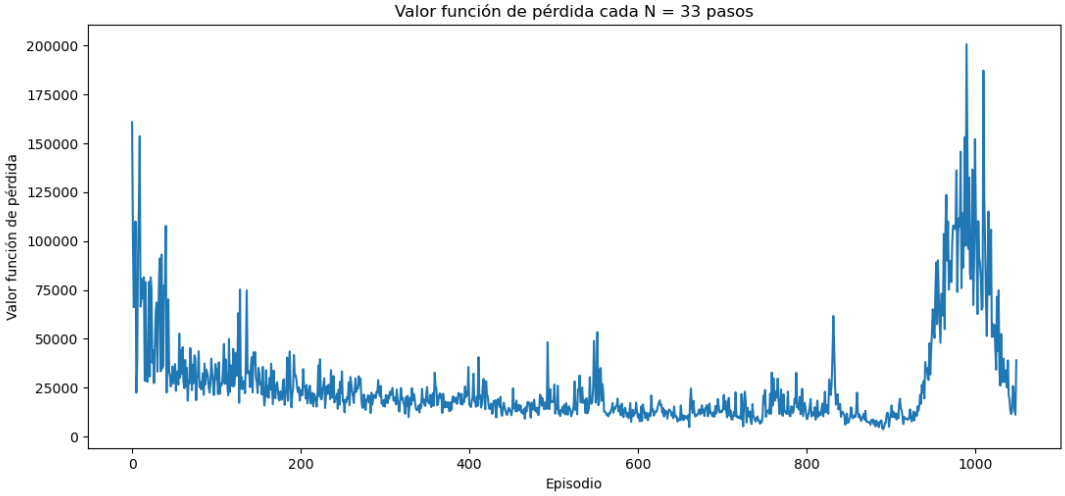

Figura C.126: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración Ñ2

C.5.23. Configuración Ñ3

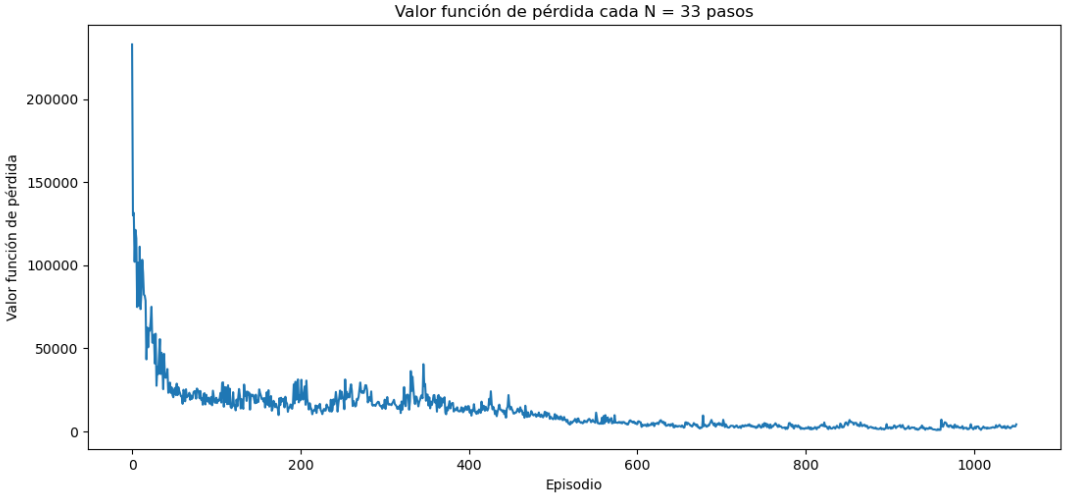

Figura C.127: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración Ñ3

C.5.24. Configuración O1

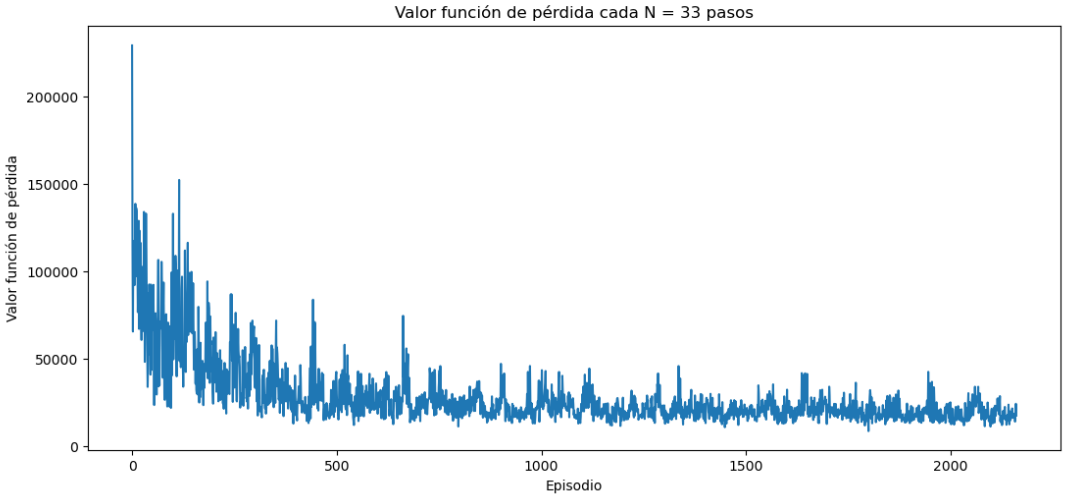

Figura C.128: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración O1

C.5.25. Configuración O2

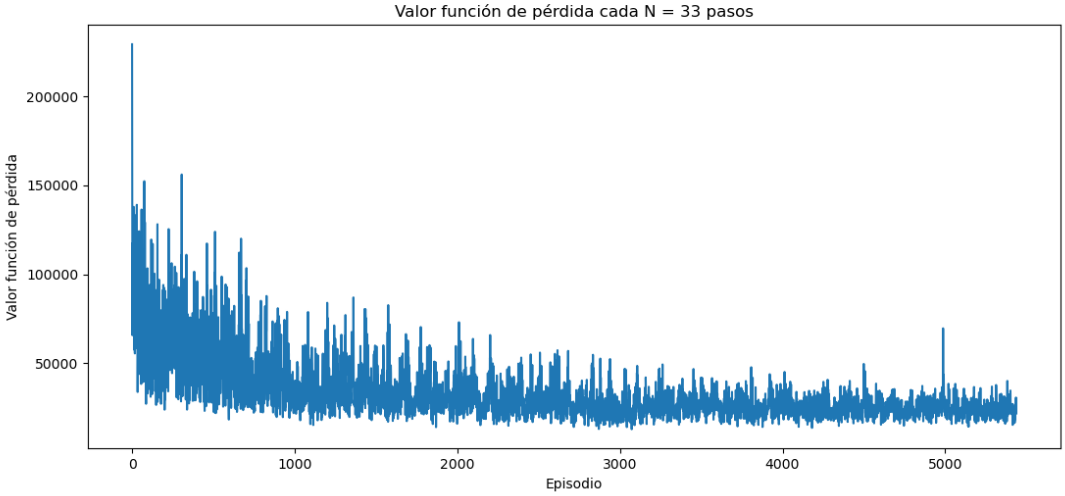

Figura C.129: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración O2

C.5.26. Configuración O3

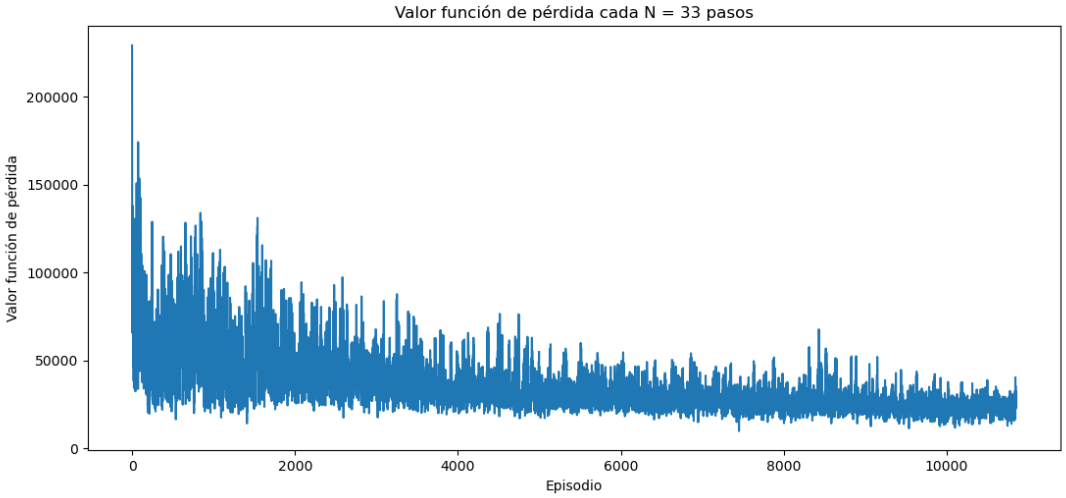

Figura C.130: Valor de función de pérdida $\mathcal{L}$ medidos cada $\mathcal{N} = 33$ pasos obtenidos de modelo basado en configuración O3

C.6. Recompensa acumulada usando explotación exclusivamente ($\epsilon = 0$)

Esto es básicamente el mejor desempeño de los agentes dado que siempre tiene el mismo estado inicial s_0. Se testea en 3 episodios.

C.6.1. Configuración A

Figura C.131: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración A

C.6.2. Configuración B

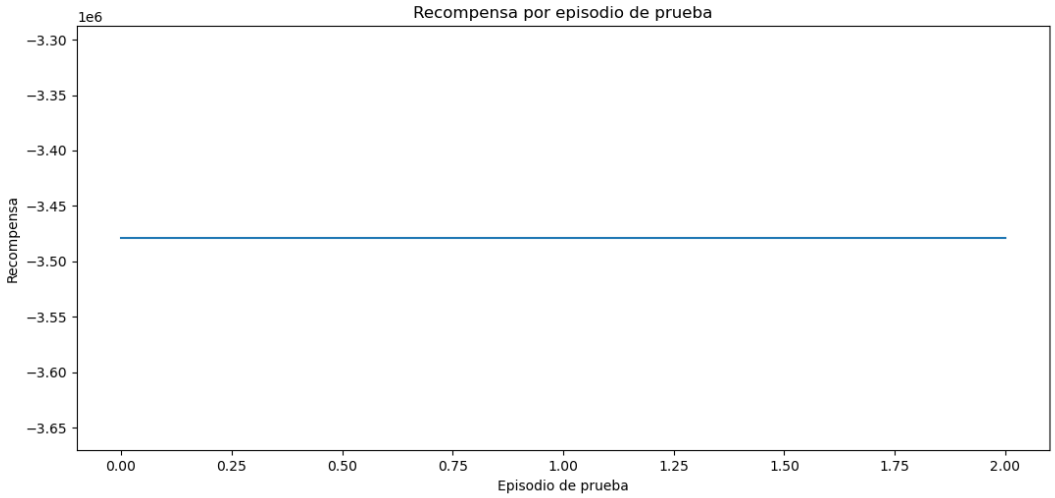

Figura C.132: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración B

C.6.3. Configuración C

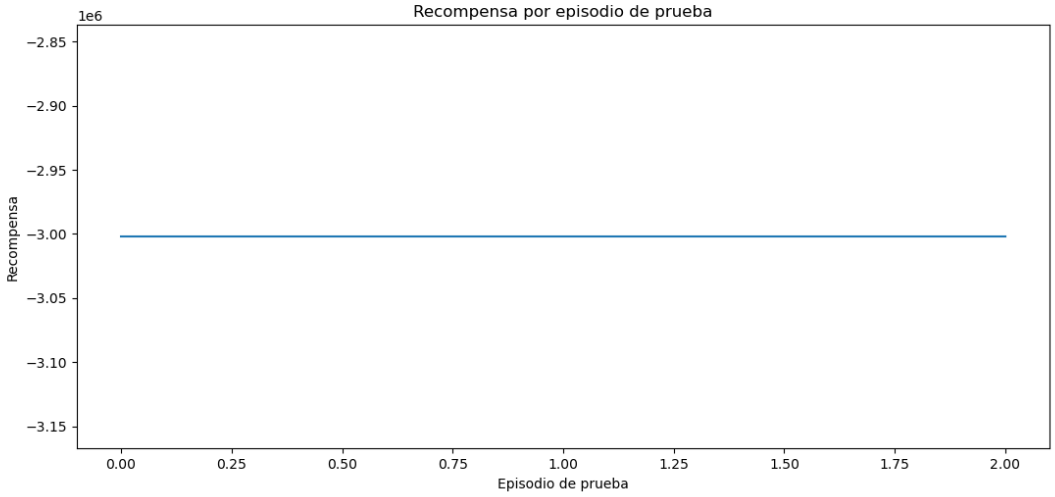

Figura C.133: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración C

C.6.4. Configuración D

Figura C.134: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración D

C.6.5. Configuración E

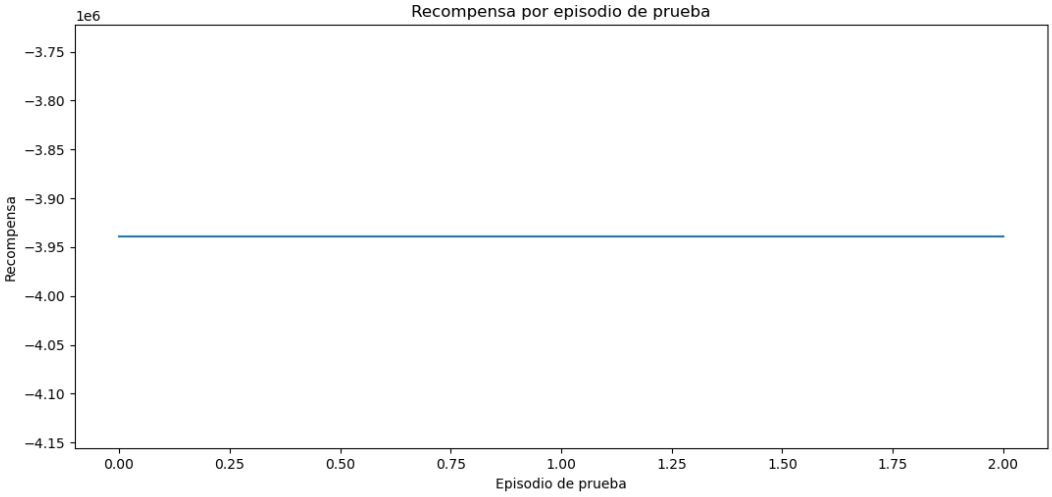

Figura C.135: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración E

C.6.6. Configuración F

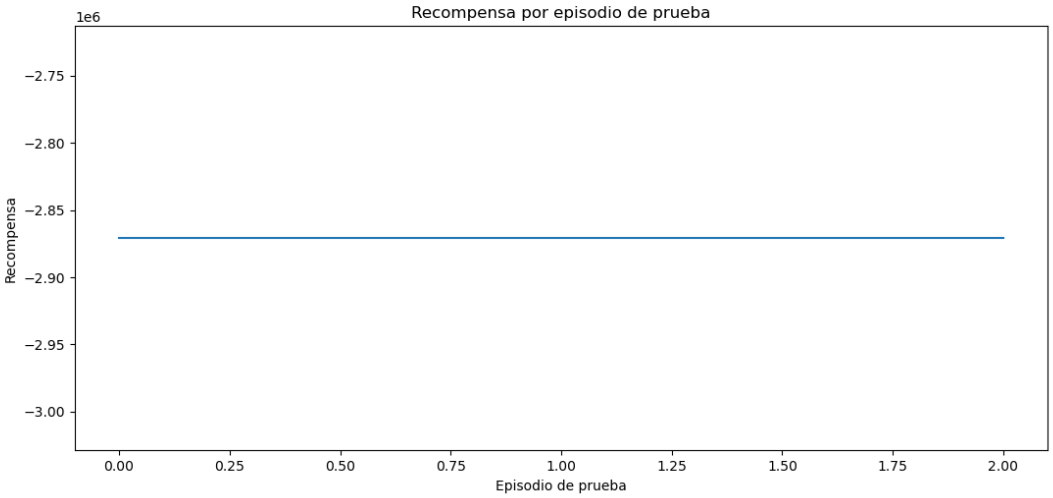

Figura C.136: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración F

C.6.7. Configuración G

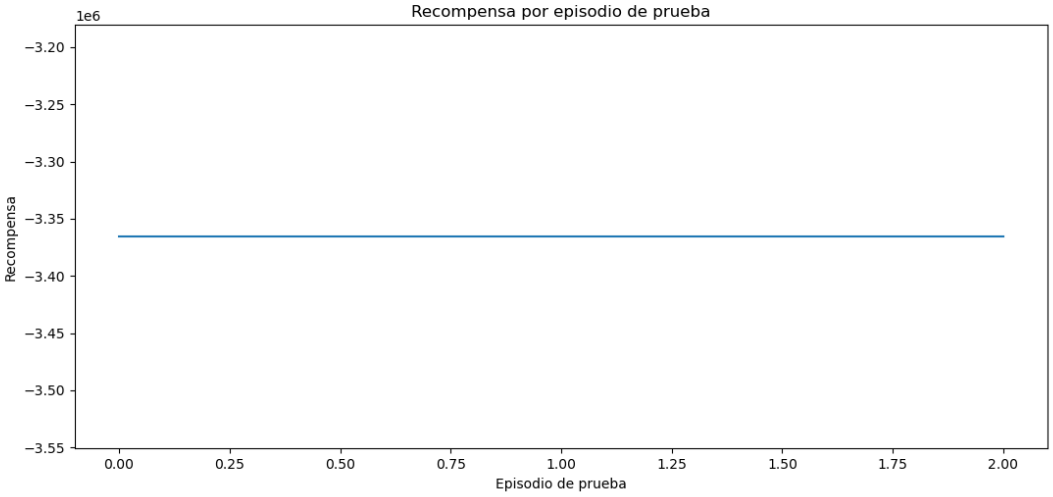

Figura C.137: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración G

C.6.8. Configuración H

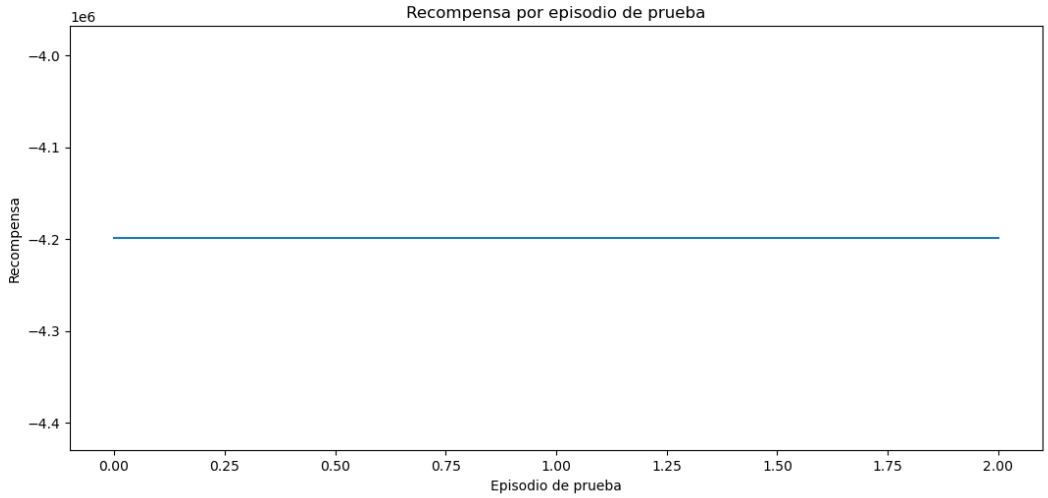

Figura C.138: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración H

C.6.9. Configuración I

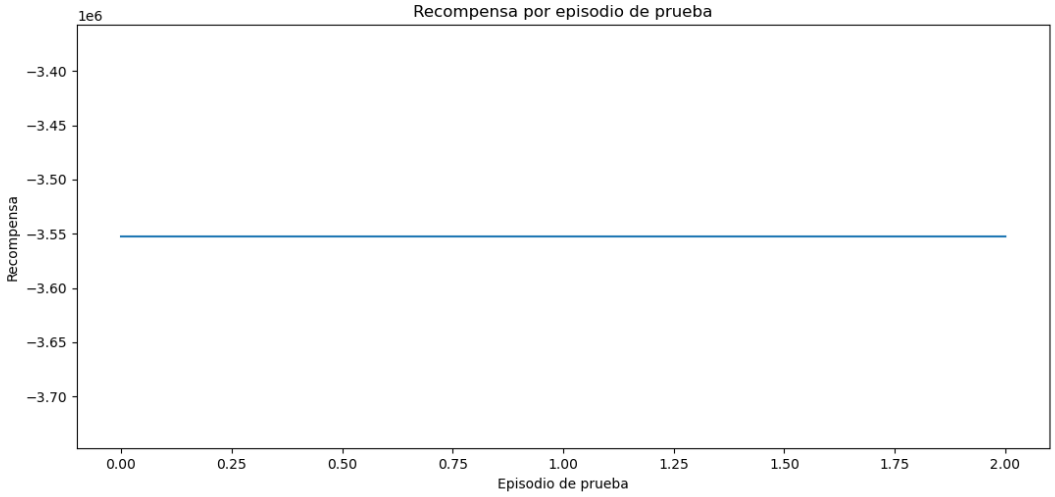

Figura C.139: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración I

C.6.10. Configuración J

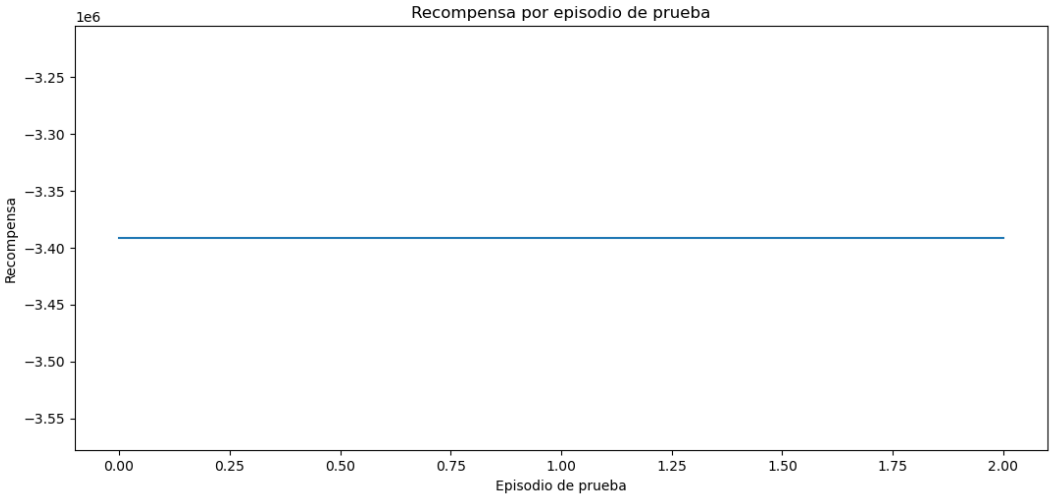

Figura C.140: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración J

C.6.11. Configuración K

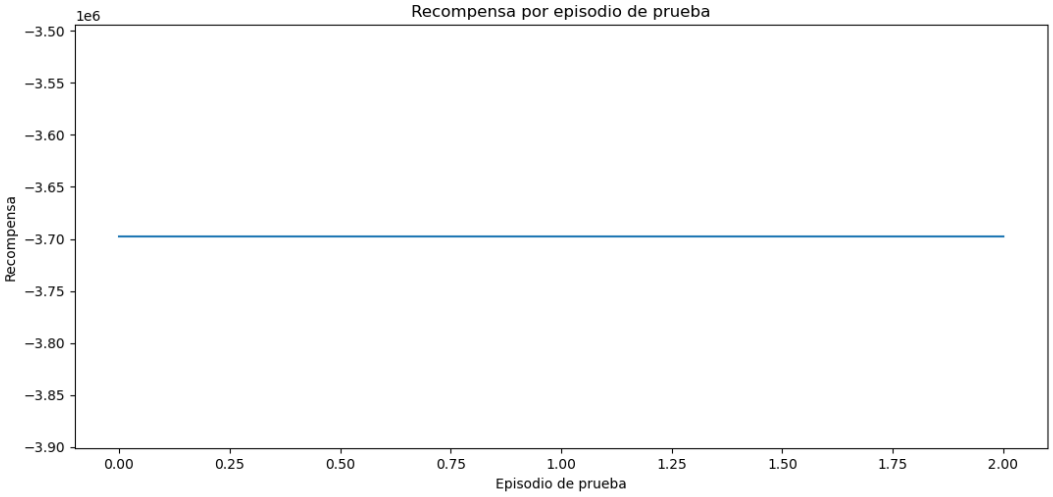

Figura C.141: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración K

C.6.12. Configuración L1

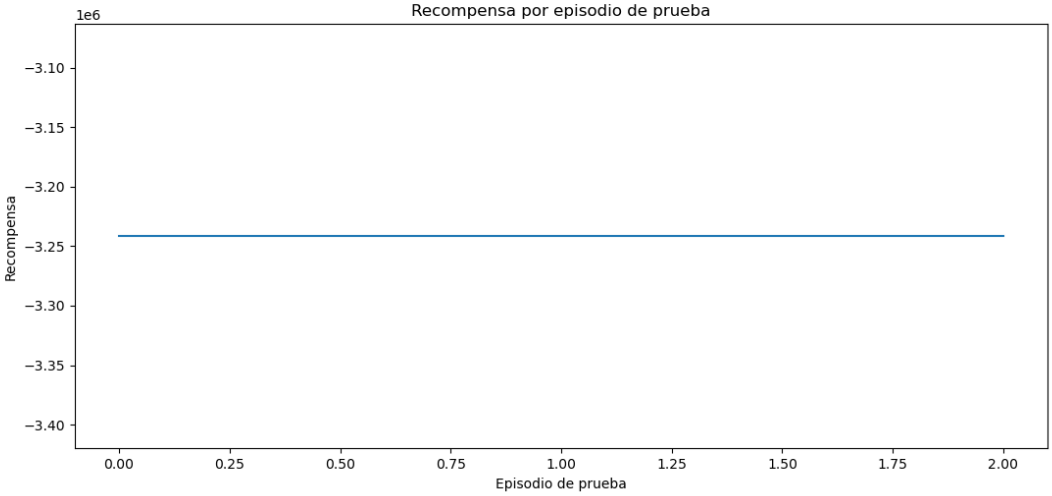

Figura C.142: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración L1

C.6.13. Configuración L2

Figura C.143: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración L2

C.6.14. Configuración L3

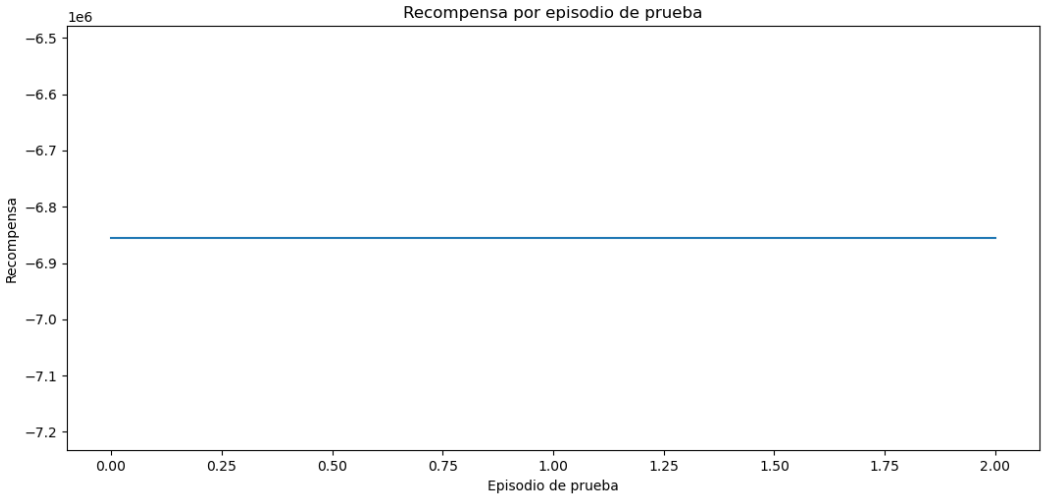

Figura C.144: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración L3

C.6.15. Configuración M1

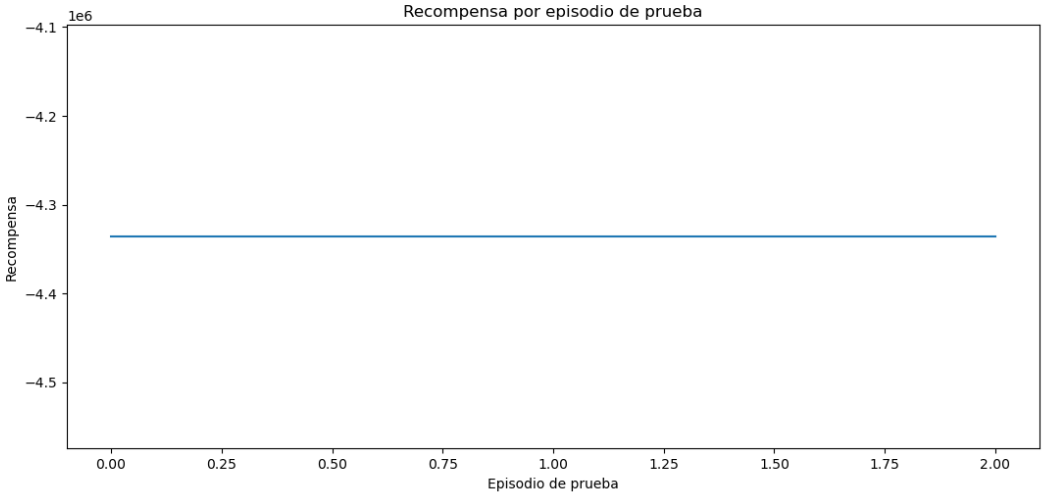

Figura C.145: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración M1

C.6.16. Configuración M2

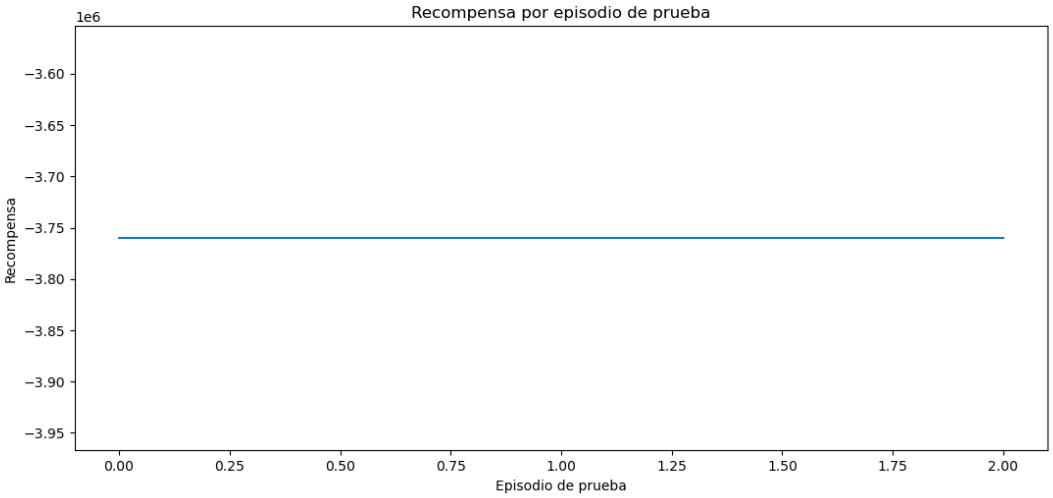

Figura C.146: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración M2

C.6.17. Configuración M3

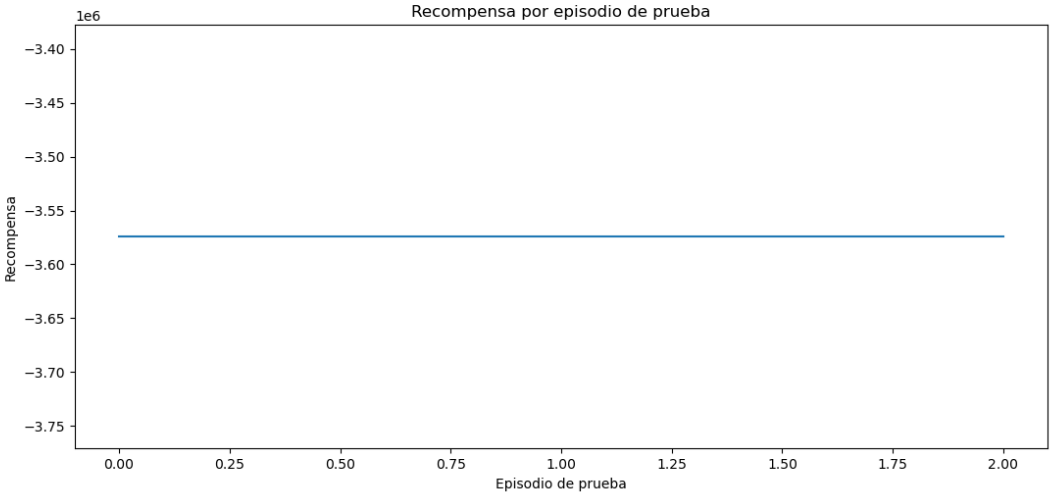

Figura C.147: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración M3

C.6.18. Configuración N1

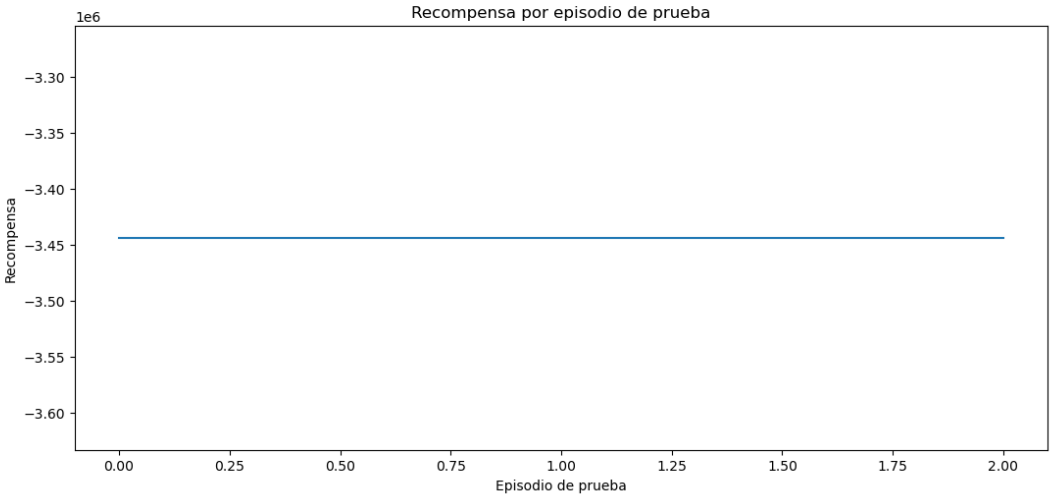

Figura C.148: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración N1

C.6.19. Configuración N2

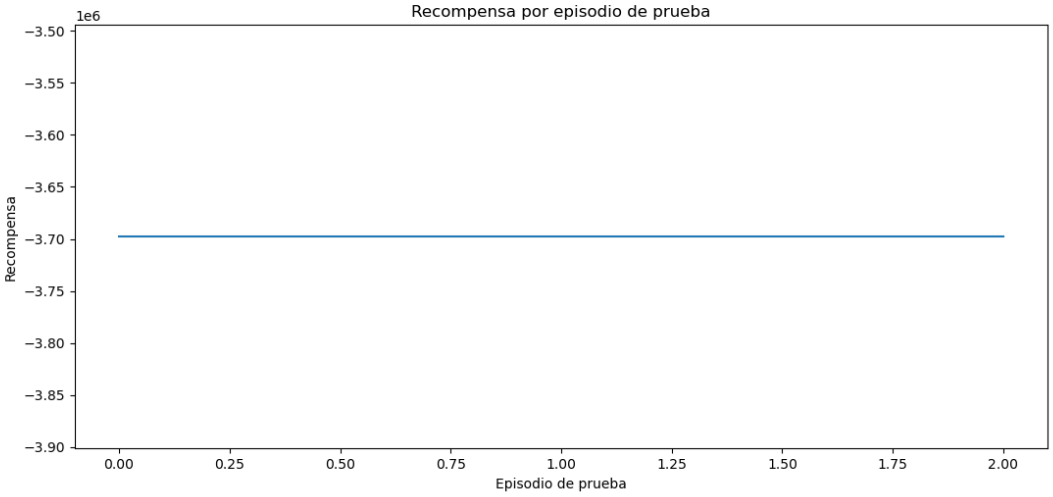

Figura C.149: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración N2

C.6.20. Configuración N3

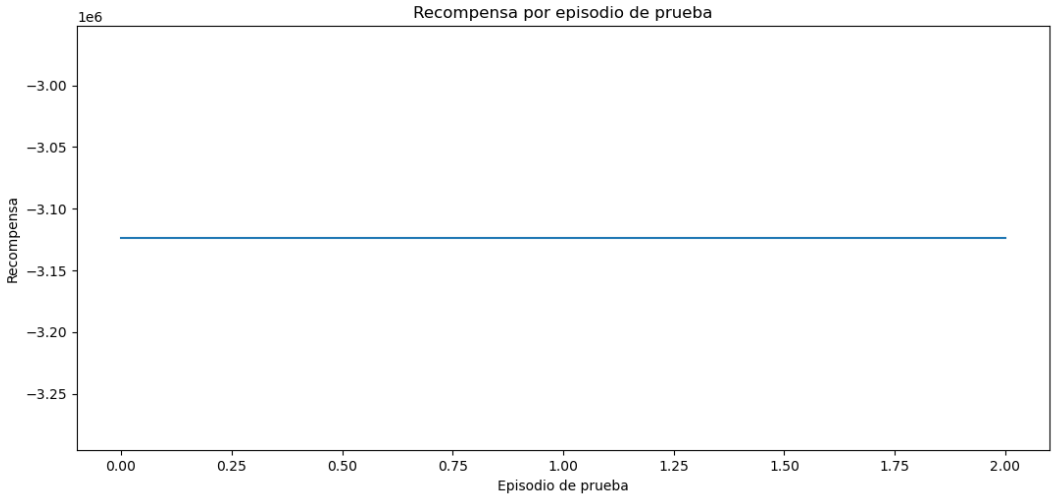

Figura C.150: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración N3

C.6.21. Configuración Ñ1

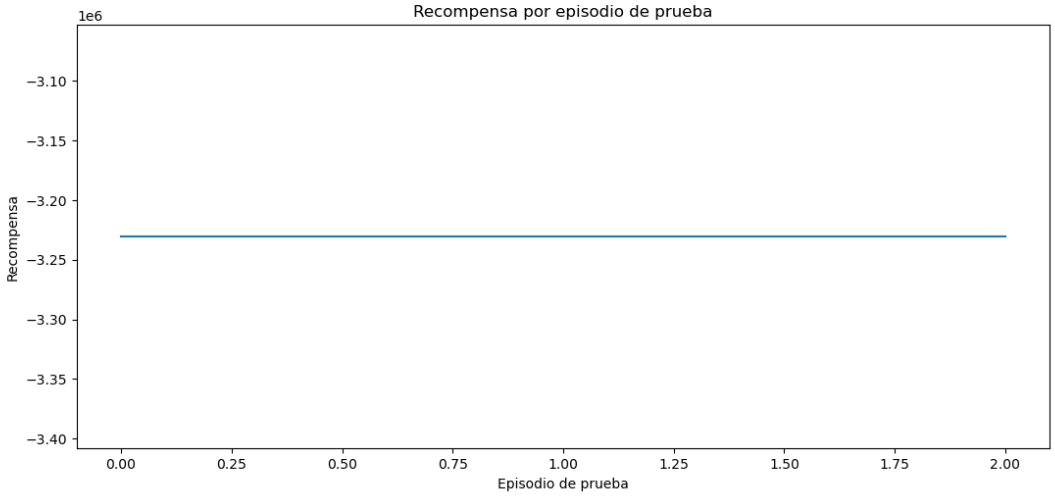

Figura C.151: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración Ñ1

C.6.22. Configuración Ñ2

Figura C.152: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración Ñ2

C.6.23. Configuración Ñ3

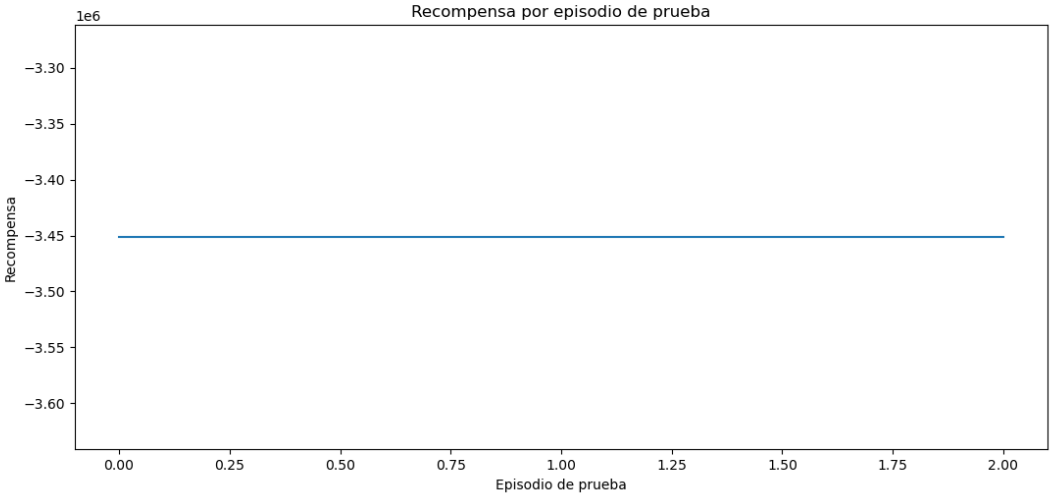

Figura C.153: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración Ñ3

C.6.24. Configuración O1

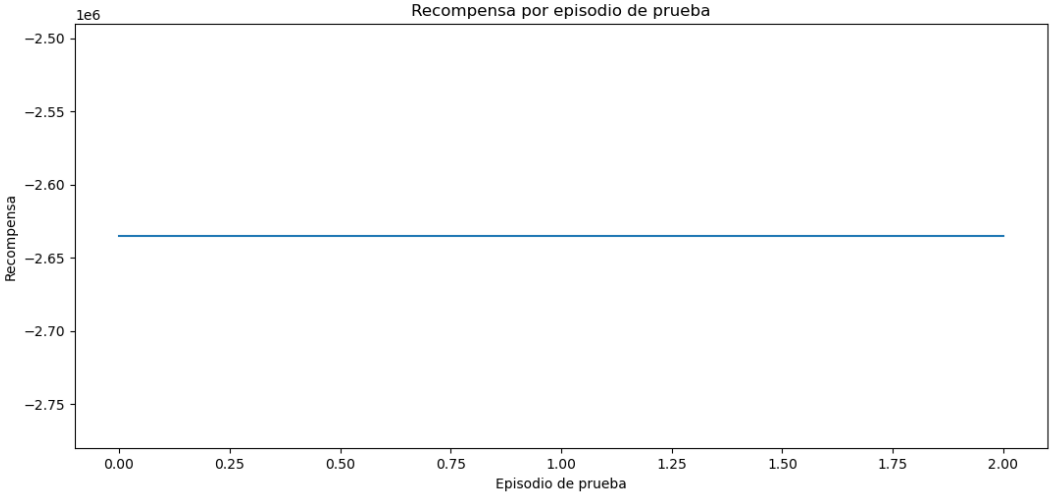

Figura C.154: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración O1

C.6.25. Configuración O2

Figura C.155: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración O2

C.6.26. Configuración O3

Figura C.156: Valor de recompensa obtenidos al explotar la mejor política aprendida por el modelo basado en configuración O3