Hierarchical Reinforcement Learning: Maze with Tasks

Rubén Cid Costa[1], **Aimar Nicuera Usandizaga**[2], **Daniel Obreo Sanz**[3]

[1, 2, 3]Universidad Carlos III de Madrid
[1]100538592@alumnos.uc3m.es, [2]100537352@alumnos.uc3m.es, [3]100451058@alumnos.uc3m.es

Abstract

La realización de objetivos secuenciales en un entorno es una tarea compleja de modelar y aprender. Para poder representar estos escenarios, una de las técnicas más usadas es el Aprendizaje por Refuerzo Jerárquico (*Hierarchical Reinforcement Learning (HRL)*). En este contexto, este trabajo se enfoca en Feudal Learning, una variante de HRL que organiza las tareas en una estructura jerárquica de niveles de abstracción. Este documento detallará las bases teóricas y su aplicación sobre un ambiente de desarrollo con tareas de navegación y obtención de subobjetivos.

Introducción

En muchos dominios, como la robótica o sistemas autónomos, las tareas implican la realización de objetivos secuenciales en entornos complejos. La capacidad de modelar y aprender estos escenarios es un desafío crucial para la inteligencia artificial.

El aprendizaje por refuerzo jerárquico (HRL) se presenta como un enfoque para la resolución de estos problemas. Mientras que otras técnicas previas enfrentan dificultades para escalar con el número de tareas y su complejidad, el Aprendizaje por Refuerzo Jerárquico (HRL) organiza el proceso en niveles de abstracción. Dentro de este marco, el Aprendizaje Feudal se presenta como un enfoque que permite modelar las tareas mediante una jerarquía de abstracción.

Este trabajo se centra en el estudio de *Feudal Learning*, una variante de HRL. Se presentan las bases teóricas de esta técnica como diferentes algoritmos o métodos de aprendizaje que se han desarrollado en este campo. También, se mostrará su aplicación sobre un entorno simulado diseñado para tareas de navegación en laberintos y obtención de subobjetivos.

Marco Teórico y Estado del Arte

El aprendizaje por refuerzo jerárquico (HRL por sus siglas en inglés) se ha consolidado como una estrategia para abordar problemas complejos que involucran secuencias de tareas. Este enfoque se originó como una extensión de los modelos de decisión de Markov para tareas de largo horizonte de decisión mediante el uso de modelos semi-MDP (Sutton, Precup, and Singh 1999). Con ellos, se establecen las bases teóricas para la abstracción temporal.

Posteriormente, con los años, han surgido una serie de algoritmos que implementan y abordan las limitaciones de HRL. Dietterich (Dietterich 2000) propuso el modelo MaxQ, que descompone las tareas en jerarquías de subtareas para la optimización y planificación de diferentes funciones de valor. Por otro lado, en (Barto and Mahadevan 2003) se exploran los avances en estructuras jerárquicas para diferentes dominios y la relevancia de HRL para la resolución de problemas de gran complejidad.

Dentro de este marco teórico, el Aprendizaje Feudal se presenta como una técnica que organiza las tareas en niveles jerárquicos, donde cada nivel es operado con distintos grados de abstracción tanto en el estado como en temporalidad. Este enfoque (Dayan and Hinton 1992) divide la jerarquía en *managers*, que operan con abstracciones del estado y dan órdenes de acción y en *workers* que realizan las acciones. Esta colaboración jerárquica permite aprender y ejecutar tareas de manera más eficiente, aprovechando la modularidad y la capacidad de escalar a problemas complejos.

Otro enfoque es el propuesto por A. S. Vezhnevets et al. (**?**), que plantea las redes feudales como una arquitectura de aprendizaje por refuerzo jerárquico. Este sistema emplea la estructura *manager-worker* mencionada anteriormente, pero añade redes neuronales recurrentes (específicamente LSTMs) como mecanismo de funcionamiento tanto del *manager* como del *worker*. Este enfoque se ha mostrado efectivo en la resolución de problemas complejos; por lo que puede ser de utilidad en la resolución del problema propuesto en este trabajo.

Sistemas de Control Feudal

En 1992, P. Dayan y G. Hinton (Dayan and Hinton 1992) definen los sistemas de control feudal como un reflejo de las sociedades feudales. En ellos, se define una jerarquía de *managers*, *supermanagers* y *workers*. Cada uno opera con un grado de abstracción de estados y temporal distinto. Los *managers* tienen poder absoluto sobre sus subordinados. Pueden dar órdenes que deben ser seguidas, dan tareas, recompensas y castigos si las órdenes no son seguidas. Esta estructura jerárquica de mando permite aprender y comprender tareas complejas con el fin de maximizar el refuerzo.

Este sistema de control se basa en dos principios:

- **Ocultamiento de la Recompensa (Reward Hiding)**. Los *managers* recompensan a los *submanagers* únicamente si operan en consonancia con las órdenes dadas. Los *submanagers* deben aprender a obedecer y aprender que debe hacer el siguiente agente para cumplir de manera eficiente. De la misma manera, el *submanager* es recompensado si su subordinado ejecuta las órdenes a pesar de no llegar a cumplir las ordenes propias del *submanager*.
- **Ocultamiento de la Información (Information Hiding)**. Los *managers* solamente deben conocer la información necesaria con un nivel de granularidad distinto a la de los *workers*. Las decisiones se toman en base a un espacio de estados menor pero más denso en información. También, las órdenes dadas a un *submanager* no son propagadas hacia el resto de los niveles para no afectar la toma de decisiones. Tampoco, se propagan hacia arriba las acciones del *worker* hacia el resto de los *managers*.

Estos principios siguen una dinámica de poder y gestión similar a la de sociedades feudales con diferente interés en cada estamento de la organización. No obstante, todos se alinean para la obtención y maximización de una recompensa (o refuerzo) dado por el problema.

Para el aprendizaje e implementación, se propuso el uso de una versión modificada del Aprendizaje Q-Learning. Cada *manager*, *submanager* y *worker* tiene asociada una función acción-valor (*Q Tabla*). La principal diferencia con respecto al algoritmo original es la modificación de la función de recompensa. En este caso, se modifica en función de las acciones tomadas por cada nivel de la jerarquía en consonancia con su nivel de obediencia.

FeUdal Networks: FuN

Las redes feudales se basan en la arquitectura de aprendizaje por refuerzo feudal (**?**), una arquitectura del aprendizaje por refuerzo jerárquico. Esta arquitectura emplea un sistema de control, conocido como *manager*, que asigna tareas a un subsistema conocido como *worker* que debe aprender a ejecutarlas de manera óptima.

La arquitectura del sistema se muestra en la imagen siguiente(Science 2024).

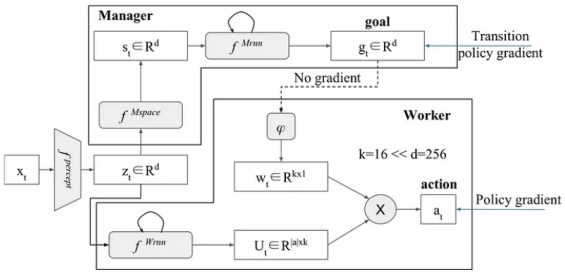

Figure 1: Arquitectura de una red feudal.

La entrada de esta red es procesada por una capa de percepción, que emplea capas convolucionales para extraer características de la imagen de entrada. A continuación, estas características son procesadas tanto por el *worker* como por el *manager*, cada uno de manera distinta; el *manager* extrae objetivos y el *worker* aprende a alcanzar esos objetivos.

El objetivo principal del *manager* es generar metas que el *worker* debe cumplir. Recibe la percepción del entorno, proporcionada por el módulo de percepción, ese estado es procesado por una red recurrente LSTM para mantener un estado interno y poder capturar información relevante en horizontes temporales largos. El *manager* emplea esta información para predecir un objetivo direccional en el espacio latente, este objetivo es un vector unitario, lo que asegura que el *worker* se enfoque en la dirección y no en la posición absoluta.

Para entrenar el *manager*, se emplea la recompensa obtenida, y emplea la similitud coseno entre la dirección en la que se movió el *worker* y la compara con el objetivo establecido, empleando la similitud coseno como función de pérdida. Esta pérdida incentiva al *manager* a emitir objetivos que maximicen el progreso hacia estados ventajosos.

El vector de objetivos se envía al *worker* sin propagar gradientes, esto garantiza que los objetivos mantengan un significado semántico independiente, en lugar de ser simples variables latentes optimizadas de manera conjunta.

En el caso del *worker*, también se emplea una red LSTM para mantener un estado interno y poder capturar información relevante, pero en este caso, el *worker* recibe tanto la percepción del entorno como el objetivo del *manager*. El *worker* emplea esta información para predecir la acción que debe realizar para alcanzar el objetivo. La acción se predice en el espacio de acciones, y se ejecuta, lo que proporciona un nuevo estado y un refuerzo que se emplea para entrenar las redes.

Evaluación práctica

Descripción del Experimento

Para la realización de pruebas, se ha planteado el problema de navegación en un conjunto de laberintos. En cada uno de ellos, el agente tiene como tarea secundaria (obligatoria) la obtención de un conjunto de monedas (*tokens*) para poder abrir la meta. Estas monedas están repartidas alrededor del mapa.

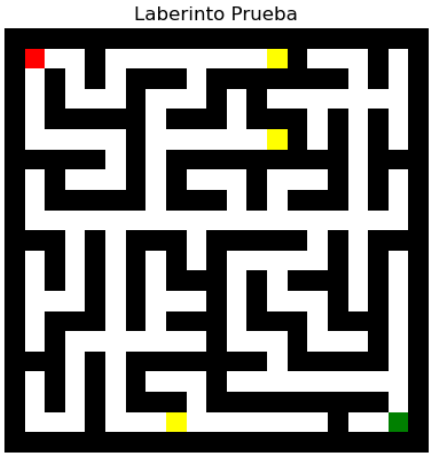

Figure 2: Ejemplo de Laberinto con 3 monedas (cuadrado amarillo) y una meta (cuadrado verde)

En términos de implementación, los laberintos se han creado usando el algoritmo de Wilson. Las monedas se posicionan aleatoriamente en casillas libres del mapa. Para poder ser usados en CNN y RNN, por cada celda original se obtienen 4 celdas con las paredes. Esto genera un laberinto con el doble del tamaño original, como se ve en Fig. 2.

Para los experimentos, se iniciaba con 0 monedas y, en base a la capacidad de aprender el camino y obtener los resultados, se añadían monedas o se cambiaba el experimento. De esta manera, se podía estudiar de manera directa el efecto del incremento en la complejidad en la capacidad de adaptarse y aprender el laberinto.

Se debe destacar que, en cada experimento, el laberinto generado es distinto. No se ha podido mantener el mismo laberinto en todas las ocasiones. No obstante, se pueden comparar los resultados debido a que todos comparten las mismas características y son válidos.

Para la experimentación, los agentes tienen como estado base (previo a transformaciones propias del método) la vista global del laberinto y la posición como coordenadas. Como acciones base, los agentes tienen las acciones de movimiento cardinales estándar NSWE (North-South-West-East). En algunos métodos, también se añade la acción $*$ para los *managers*.

Finalmente, aunque inicialmente se partió de episodios partiendo del punto más alejado a la meta, se decidió implementar una política de inicio aleatorio en favor de la exploración y capacidad del agente para solventar el problema desde diferentes áreas del laberinto.

Experimento: Feudal Q-Learning

Para comprobar la efectividad de estos métodos, se han realizado diferentes pruebas e implementaciones. El primer experimento se basa en una implementación que refleja fielmente los alineamientos de los sistemas de control feudales descritos.

Basándose en el ejemplo de la publicación, se han planteado una jerarquía de cuatro niveles en la que cada *manager* tiene la mitad de resolución que el nivel anterior (Véase Fig.3). Además, los *managers* tienen acceso al estado con la presencia de monedas en cada región.

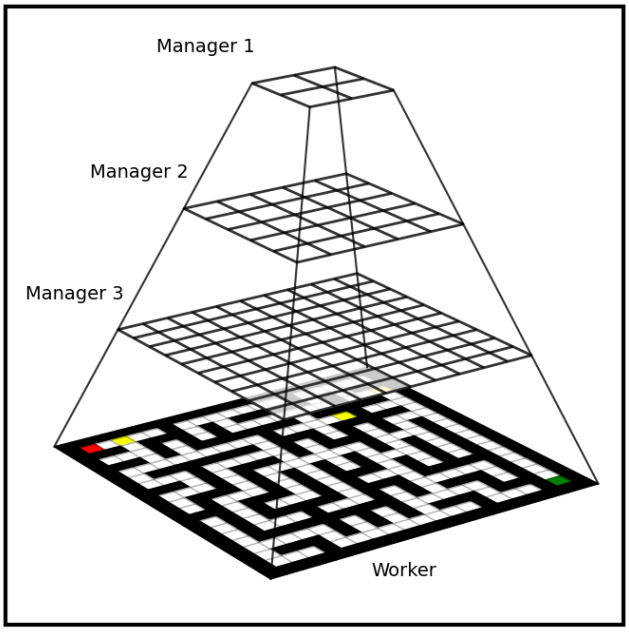

Figure 3: Abstracciones y Niveles de Manager. Cuadros marcados: salida (verde), monedas (amarillo) y entrada (rojo)

A pesar de que este método se presenta prometedor, los resultados obtenidos no son aceptables (Fig 4). Para el problema base de navegación sin tareas intermedias, el algoritmo ha fallado en alcanzar el objetivo reiteradamente. Se destaca la falta de obediencia por parte del *worker* que deriva en una mejora en el número de pasos y recompensa. No obstante, esto indica que el problema se encuentra en los *managers* y sus órdenes.

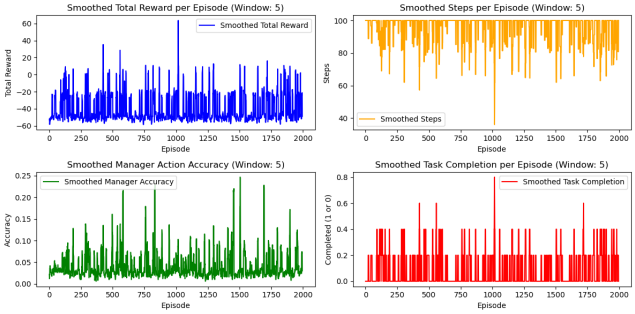

Figure 4: Feudal Q-Learning con 0 monedas.

Tras un análisis de los resultados, se ha detectado un problema de representación que no se tenía en el experimento original. El laberinto contiene una gran cantidad de caminos sin salida y cerrados. En estos casos, los *managers* no tienen percepción de las paredes y su representación puede incluir

intersecciones entre varias zonas paralelas. Este problema deriva en órdenes inválidas por parte de los *manager* que desorientan al *worker*.

Experimento: Feudal Q-Learning Modificado

Debido a la disipación de las paredes que se experimentaba al realizar las abstracciones del mapa de estados a medida que se disminuía el nivel del gerente, el aprendizaje resultaba altamente perjudicado, llegando a transmitirse órdenes imposibles de realizar debido a la naturaleza cerrada del mapa del problema.

Para enfrentar dicha adversidad se probó a disminuir el número de gerentes/subgerentes, limitándose al uso del gerente de menor nivel de abstracción; aquel que veía el mapa de estados en la escala original. Por lo tanto, se realizó una disminución de agentes para quedarse con únicamente 2: un gerente encargado de determinar la siguiente zona destino y otro agente encargado en navegar hasta dicha posición de interés. Dicha estructura contaba con las mismas limitaciones y políticas de recompensa definidas para el proceso anterior.

Para facilitar el arranque y el uso del *manager* desde el principio, se facilitaba al *manager* una pista de la localización de las monedas que debían ser recogidas para que guiara al trabajador (*worker*) hacia dichas zonas sin necesidad de un proceso inicial donde las directrices del *manager* podían ser "ignoradas" debido a su aleatoriedad.

El trabajador a su vez realizaba un proceso de exploración que se veía recompensado cuando alcanzaba el punto objetivo determinado por el *manager*. Mediante estas exploraciones, iba creando sus caminos o matrices Q que iba actualizando y afinando con el fin de encontrar el camino óptimo a partir de las recompensas obtenidas por obedecer al *manager*.

En cuanto al proceso de aprendizaje, se incluyeron diferentes implementaciones para ver la influencia que estas variaciones o perturbaciones tenían en el proceso de aprendizaje del super-agente. Las perturbaciones más comunes consistían en limitar el conocimiento que tenía el *teacher* (dándole solo el conocimiento de la solución empezando desde las coordenadas (1,1) o el conocimiento completo) o variando la posición de comienzo del agente.

En cuento a los resultados que se obtuvieron sin ayudas externas, el proceso de aprendizaje era más lento al tener que explorar muchos estados y no contar con una ayuda externa en caso de caer en bucles debido a los estrechos caminos que componían el mapa (2). Debido a este desconocimiento y falta de ayuda, las primeras iteraciones oscilaban en refuerzos negativos en su mayoría debido al proceso de exploración y explotación de conocimientos inciertos en dichas iteraciones.

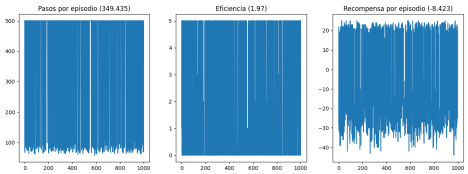

Figure 5: Primeras 1000 iteraciones sin ayuda (16x16)

Sin embargo, la influencia de un *teacher* con conocimiento de un solo camino ante el que siempre tenía la solución óptima suponía una gran diferencia. Mediante esta ayuda, el *worker* era capaz de afinar mejor la ruta a seguir y por lo tanto se vio un incremento en la cantidad de refuerzo medio que se obtenían, pasando este a ser un valor positivo.

Aun así, al no contener información explotable desde todas las zonas del mapa de estados, se observaban procesos de oscilaciones también en los momentos en los que se partía de zonas inexploradas. Esto se debía a que el agente tenía que lograr llegar hasta el punto más cercano de donde tenía opción de explotar conocimiento útil para poder obtener una ayuda, lo que lo ponía en similar situación que cuando no contaba con ayuda externa.

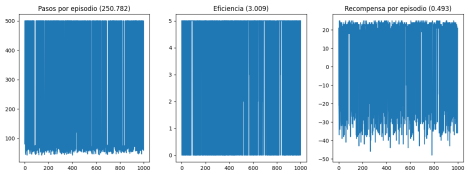

Figure 6: Primeras 1000 iteraciones con ayuda simple (16x16)

La máxima mejora del arranque inicial se logró mediante la introducción de rutas desde 3 zonas diferentes del mapa. Al depender del azar, estas rutas podrían aportar información redundante al tener inicios en lugares aleatorios del mapa, englobaban una mayor cantidad de estados, ofreciendo información explotable y útil en una subsección mayor del mapa. Debido a esto, la explotación del conocimiento y el reajuste ante un paso de exploración indeseado eran corregidos lo que llevaba a evitar bucles o episodios perdidos.[1]

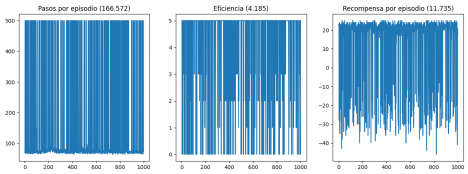

Figure 7: Primeras 1000 iteraciones con ayuda compleja (16x16)

[1] Situaciones en las que, debido a entrar en un callejón, se realizaban pasos hasta llegar al límite de estos por episodio.

En conclusión, esta nueva estructura feudal, de un noble (*manager*) y un subordinado (*worker*), supuso una mejora en comparación con el nivel anterior debida a la simplicidad y mayor énfasis en las limitaciones del mapa de estados que permitía definir. Mediante los experimentos que se realizaron, se observó una mejora tanto en el proceso de aprendizaje como en la capacidad de llegar a soluciones en mapas de 20x20 tras 9000 iteraciones en caso de no incluir la ayuda externa o 1000 iteraciones cuando se le facilitaba el conocimiento explotable en 3 rutas.

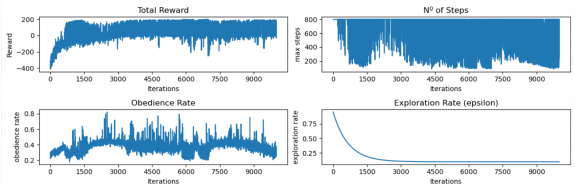

Figure 8: Resultados tras 9000 iteraciones (20x20)

Experimento: FeUdal Networks (FUN)

Otra técnica empleada de aprendizaje por refuerzo feudal son las FeUdal Networks (*FuN*), basadas en LSTMs. Se creó una adaptación de este tipo de red para que pudiera funcionar en el dominio tratado, y se establecieron y probaron distintos refuerzos que proporcionar a la red, evaluando su efectividad a la hora de encontrar soluciones.

La primera versión del modelo FuN, emplea una función de refuerzo muy similar a la usada durante el Q-learning, en donde se da un refuerzo positivo por alcanzar los objetivos y se dan refuerzos negativos a medida que transcurre el tiempo sin alcanzar un objetivo. Esta primera versión no produjo resultados demasiado positivos, el modelo a veces lograba completar el laberinto, pero era común que se quedara atascado y que la pérdida no disminuyera. Todos los modelos probados en esta sección se han entrenado durante 500 épocas, con un límite de 2000 acciones por época (si en esas iteraciones no logra resolver el laberinto, se reinicia) y para comprobar si el modelo aprende correctamente se analizó si la pérdida disminuía a lo largo del tiempo. Se puede observar que no es el caso en la siguiente gráfica, donde el modelo aprende al principio y luego alcanza un punto estable.

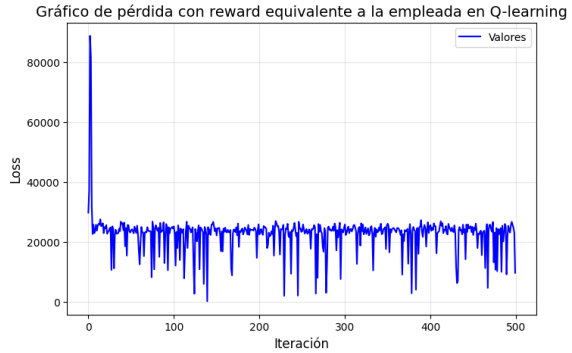

Figure 9: Pérdida de la primera versión.

En la segunda versión, y para intentar mejorar la precisión del modelo, se añadió otra regla que penalizaba al modelo cuanto más lejos estuviera del objetivo definido por el *manager*, para lo que se empleó la distancia de Manhattan. Esta nueva versión produjo resultados más inestables, donde no se llega a apreciar una auténtica mejora del modelo a lo largo del tiempo; los resultados se comparan con los de la versión anterior en la siguiente gráfica. Hay que tener en cuenta que la pérdida no está en la misma escala entre las dos versiones, y lo importante que se debe observar es una disminución de esta a lo largo del tiempo, lo que indica que el modelo está aprendiendo, independientemente de la escala real de la pérdida. En la primera versión se penalizaba menos al modelo y por eso parece que tiene una pérdida menor, pero no quiere decir que sea mejor.

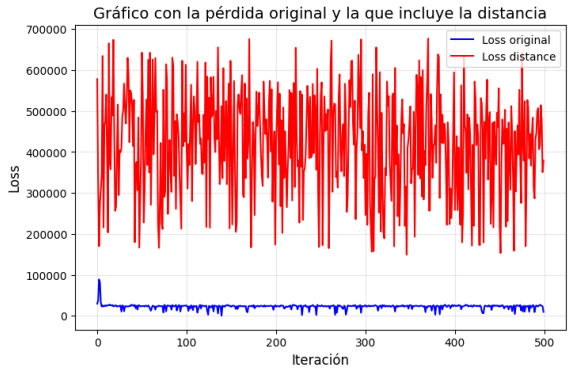

Figure 10: Pérdida de las dos primeras versiones.

Se comprueba que no mejora demasiado a lo largo del tiempo, y los valores son más inestables, lo que no es necesariamente algo negativo, pero tampoco parece estar ayudando mucho al modelo.

Como tercera versión de este método, se añadió una recompensa por explorar estados nuevos y un castigo por repetir estados, lo que resultó en un nuevo modelo que, si bien sí que parece que aprenda, sigue sin producir soluciones mejores que los otros métodos. Es posible que si se entrenara durante el tiempo suficiente, y modificando los parámetros de las recompensas hasta alcanzar los mejores valores, se obtuvieran resultados similares e incluso mejores que los proporcionados por los otros métodos, pero requeriría una gran carga computacional.

Los resultados de esta versión, comparados con las otras dos, son los siguientes.

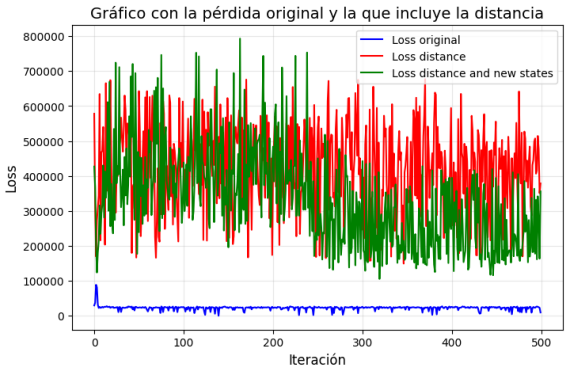

Figure 11: Pérdida de todas versiones.

Se observa que el modelo final, representado por la línea verde, sí que aprende, pero de manera muy lenta e inestable, sin llegar a alcanzar resultados óptimos.

Conclusiones

En base al trabajo realizado, se puede concluir que las técnicas de aprendizaje por refuerzo jerárquico, específicamente el Aprendizaje por Refuerzo Feudal, añaden cierto valor añadido en aquellos problemas que cuentan con un fin complejo que se puede presentar como una suma de tareas a tratar de forma paralela y ordenada; en otras palabras, de forma jerárquica. Este sistema permite determinar agentes para cada nivel de abstracción estrictamente necesaria para la tarea, abstrayendo la decisión de problemas e inconvenientes pertenecientes a otras subtareas del problema.

Por el otro lado, se ha podido observar la influencia de la sobre-división de las tareas. Al dividir una tarea en demasiados niveles de abstracción, se realizaba una sobre-abstracción del espacio, lo que llevaba a tomar decisiones sin la información necesaria para dicha labor. En el caso del estudio realizado, dicho problema se ve bien representado en el caso de la selección de los destinos a navegar, donde debido a la excesiva abstracción del mapa de estados, se perdía la información espacial de las paredes.

Debido a ello, se estima que, aunque dicha decisión se tome como abstraible en otras investigaciones, la naturaleza cerrada en cuestiones de navegación del problema tratado anulaba dicha característica.

Trabajo Fúturo

En vista de las limitaciones presentes del entorno, se plantea como trabajo para continuar el cambio del método de generación del laberinto, la búsqueda de un abstracción más eficiente para el sistema de control feudal. Con estas medidas, se podrían obtener resultados similares o mejores para el método basado en q-learning para el problema propuesto.

References

Barto, A. G.; and Mahadevan, S. 2003. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(1-2): 41–77.

Dayan, P.; and Hinton, G. E. 1992. Feudal reinforcement learning. *Advances in neural information processing systems*, 5.

Dietterich, T. G. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13: 227–303.

Science, T. D. 2024. Hierarchical Reinforcement Learning: Feudal Networks.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2): 181–211.