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

DESARROLLO DE UNA RED NEURONAL ENTRENADA CON APRENDIZAJE REFORZADO APLICADA AL PROBLEMA DE RUTEO DE UNA FLOTA DE VEHÍCULOS ELÉCTRICOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

ANDRÉS ANTONIO URRUTIA ALIAGA

PROFESOR GUÍA:
DIEGO MUÑOZ CARPINTERO

MIEMBROS DE LA COMISIÓN:
FRANCISCO RIVERA SERRANO
CLAUDIO BURGOS MELLADO

SANTIAGO DE CHILE
2024

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: ANDRÉS ANTONIO URRUTIA ALIAGA
FECHA: 2024
PROF. GUÍA: DIEGO MUÑOZ CARPINTERO

DESARROLLO DE UNA RED NEURONAL ENTRENADA CON APRENDIZAJE REFORZADO APLICADA AL PROBLEMA DE RUTEO DE UNA FLOTA DE VEHÍCULOS ELÉCTRICOS

El problema de ruteo de vehículos (VRP, por su sigla en inglés) consiste en un problema de optimización combinatorial que busca encontrar la respuesta a la pregunta: "¿Cuál es el conjunto óptimo de rutas para una flota que debe satisfacer las demandas de un conjunto dado de clientes?". Existen tres elementos principales involucrados en el VRP, que son: los clientes, la bodega o depósito del cual se extraen los productos y la flota, que puede considerar 1 o más vehículos. En los problemas reales de VRP, se aplican diversas restricciones, entre las que destacan: capacidad limitada de los vehículos, varios puntos de suministro (múltiples depósitos) y los clientes deben ser visitados en una ventana de tiempo determinada. La importancia de la solución a este problema radica en los significativos ahorros relacionados al costo de transporte y despacho de productos que pueden significar hasta el 10 % del costo total[1].

En el presente trabajo, se realizó una investigación bibliográfica sobre las soluciones del VRP propuestas en la literatura. Estas propuestas son explicadas y analizadas para entender posteriormente las soluciones desarrolladas en este trabajo. Además, se realizó una investigación sobre el algoritmo de aprendizaje reforzado y su aplicación en redes neuronales para el VRP.

A raíz de lo anterior, la instancia del VRP abordada en este trabajo, consiste en una flota de 3 vehículos eléctricos que deben satisfacer la demanda en los destinos, considerando las siguientes restricciones: pausas de reabastecimiento de batería para mantener el estado de carga (SoC, por su sigla en inglés) entre ciertos límites, capacidad limitada del vehículo y número máximo de pausas de reabastecimiento.

En este trabajo se desarrollaron tres soluciones basadas en redes neuronales, que atacan el problema de distintos enfoques. La primera solución corresponde a una red neuronal que considera cargas completas de batería en cada pausa de reabastecimiento. El segundo modelo consiste en un modelo con estaciones de carga ficticias con distintos porcentajes de carga. Por último, la tercera solución hace uso de una red neuronal adicional con salida continua, la cual es interpretada como el porcentaje a cargar en las pausas de reabastecimiento. Las tres redes neuronales fueron programadas en Python utilizando principalmente la librería Pytorch, la cual está optimizada para la implementación y entrenamiento de redes neuronales profundas.

A partir de resultados obtenidos en este trabajo, se concluye la correcta implementación tanto de las redes neuronales, como del algoritmo de entrenamiento reforzado. Además, en términos de desempeño general, se concluye que el mejor rendimiento lo obtiene el modelo de carga continua, seguido por el modelo de carga completa y por último el modelo de estaciones de carga ficticias.

Dedicado a mis padres, hermanas y polola.
Gracias por su apoyo incondicional.
Los amo.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

El problema de ruteo de vehículos, o VRP por su siglas en inglés, es un problema de optimización combinatorial que busca minimizar los costos de transporte a partir de encontrar la mejor ruta para recorrer un número finitos de destinos, considerando múltiples restricciones de operación[2]. El VRP puede considerar flotas de uno o más vehículos.

La importancia de estudiar este problema, reside en sus múltiples aplicaciones en la industria. Las soluciones del VRP pueden significar ahorros significativos en el transporte de productos, que corresponde hasta el 10 % del costo total[1]. Por otro lado, dada la crisis ambiental que se ha exacerbado en los últimos años como plantea la Convención Marco de las Naciones Unidas sobre el Cambio Climático (CMNUCC)[3], es importante fomentar la utilización de energías renovables para ayudar a disminuir el impacto ambiental.

La complejidad del VRP varía dependiendo de la configuración en la cual se está trabajando. Un ejemplo de variante del VRP, es el problema de ruteo de vehículos con capacidad (CVRP), donde se considera una carga máxima que pueden cargar los vehículos. Otra variante, es el problema de ruteo de vehículos con ventanas de tiempo (VRPTW), que consiste en restringir las visitas a los destinos a una ventana de tiempo determinada. El problema de ruteo que considera vehículos eléctricos, es denominado E-VRP (*Electric Vehicle Routing Problem*).

En la literatura, la estrategia que parece ser más explorada y más utilizada, es la resolución del problema de optimización como tal. Esta estrategia asegura la obtención de la ruta óptima dadas las restricciones impuestas, sin embargo, sus extensos tiempos de resolución han motivado la exploración de otras estrategias como lo es el uso de redes neuronales. La utilización de redes neuronales simplifica considerablemente los tiempos de resolución *online*, sin embargo, requieren un entrenamiento prolongado y con muchos ejemplos. En particular hay un tipo de red neuronal llamada *Pointer Network* [4] que ha sido ampliamente utilizada para resolver algunas variantes del VRP. No obstante, las propuestas que utilizan este tipo de red solo cubren las componentes discretas de la solución final, es decir, encuentran la secuencia óptima de destinos a visitar pero no dan cuenta de algunos aspectos continuos como lo es el tiempo total de viaje o estado de carga del vehículo. Otra alternativa, es la resolución del problema de optimización pero con la utilización de heurística. Algunas publicaciones

utilizan heurística para simplificar la resolución del problema y así mejorar sus tiempos, pero el gran problema con esto es que al considerar estas simplificaciones no se puede asegurar que el resultado obtenido corresponda al óptimo.

En este trabajo de memoria, se busca crear una red neuronal que sea capaz de solucionar; para todos los vehículos de la flota, tanto la secuencia óptima para visitar los destinos, como también los aspectos continuos como lo es la cantidad de carga en las pausas de reabastecimiento. Es por esto que se utiliza como base a la *pointer network* pero se incluyen modificaciones para resolver los aspectos continuos de la solución.

Por otra parte, la red neuronal es entrenada con aprendizaje reforzado, pues dada sus características, nos permite llegar a la solución óptima aún cuando no se conoce a priori dicha solución. Esto, lo logra mediante la interacción de la red neuronal con el sistema. De esta forma, no es necesario resolver el problema de optimización como tal, lo cual conlleva mucho tiempo y recursos computacionales.

1.2. Formulación del problema

Para proporcionar un mejor entendimiento del problema, a continuación se presenta la formulación formal de un VRP clásico extraído de [5]. Considerando el grafo completo $G = (V, A)$, donde $V = \{0, ..., n\}$ es el set de vértices y $A = \{(i, j) : \forall i, j \in V\}$ es el set de arcos. Los vértices, también llamados nodos, son puntos geográficos tales que los vértices *i = 1,...,n* representan clientes y el vértice 0 representa el depósito. Cada arco $(i, j) \in A$ es asociado a un costo no negativo c_{ij}. Este costo representa el gasto de un vehículo al viajar de i a j. Si G es un grafo dirigido, entonces $c_{ij} \neq c_{ji}$, i.e, la matriz de costos es asimétrica, y el problema es llamado VRP asimétrico. De lo contrario, el problema es llamado VRP simétrico. En la mayoría de casos prácticos, la matriz de costos satisface la desigualdad triangular:

$$c_{ij} + c_{kj} \geqslant c_{ij} \forall i, j, k \in V. \tag{1.1}$$

Lo anterior establece que el arco *(i,j)* es siempre el camino más conveniente de viajar de i a j.

Cada cliente i para cada $i = 1, ..., n$ requiere una demanda conocida no negativa d_i, mientras que el depósito tiene una demanda imaginaria $d_0 = 0$. En el depósito hay un set de m vehículos idénticos. Estos vehículos entregan toda la demanda requerida por los clientes. En los escenarios más realistas, los vehículos tienen una capacidad máxima C tal que $C \geqslant d_i$ para cada $i = 1, ..., n$. Cuando la restricción anterior es considerada, el VRP es denominado problema de ruteo de vehículos con capacidad (CVRP).

Considerando todo lo anterior, el VRP es definido como el problema de encontrar una colección de exactamente m rutas (una ruta para cada vehículo) de mínimo costo. El costo total de la ruta corresponde a la suma de todos los costos de los arcos en esa ruta. Además, las rutas son tales que cada vehículo empieza y termina en el depósito; cada cliente es visitado exactamente una vez por un vehículo; y la suma de todas las demandas que cada vehículo transporta no excede la capacidad C.

La formulación de VRP anterior puede ser extendida añadiendo más restricciones operacionales, las cuales producen una nueva variante del VRP. Por ejemplo, una variante popular del VRP es el VRP con ventanas de tiempo (VRPTW). En esta variante, los clientes tienen un tiempo de servicio (adicional a su demanda) y un intervalo de tiempo donde la operación se puede llevar a cabo. En algunos casos, los vehículos llegan al cliente antes de que esta ventana de tiempo inicie. En ese caso, el vehículo debe esperar hasta que la operación se puede ejecutar.

Considerando todo lo anterior, para este trabajo de memoria, se trabajó el problema de ruteo de una flota de vehículos eléctricos con capacidad. La variación del VRP de vehículos eléctricos añade al problema las restricciones propias de los vehículos, como lo son:

- Capacidad máxima de la batería
- Pausas de reabastecimiento de carga
- Tiempo total de viaje

Por lo tanto, se busca que el sistema reciba una serie de coordenadas correspondiente a la ubicación geográfica de los destinos que los vehículos deben visitar para satisfacer cierta demanda, considerando, además, restricciones de operación como: carga máxima de vehículos, capacidad de la batería, pausas de reabastecimiento y tiempo total de viaje. Como se mencionó anteriormente, el problema es resuelto mediante la utilización de redes neuronales entrenadas con aprendizaje reforzado con una cantidad considerable de ejemplos. Además, es importante mencionar que el resolver el VRP, considerando la información de una red de transporte real, puede ser muy complejo en términos de capacidad computacional, por lo que se hace necesario la utilización de simplificaciones en la red de transporte durante el entrenamiento del sistema. Sin embargo, se busca que el sistema sea capaz de continuar su entrenamiento durante la operación, considerando la información de la red de transporte real. A continuación, en la Figura 1.1, se presenta un ejemplo de la solución esperada por el sistema para 3 vehículos, extraída de [2].

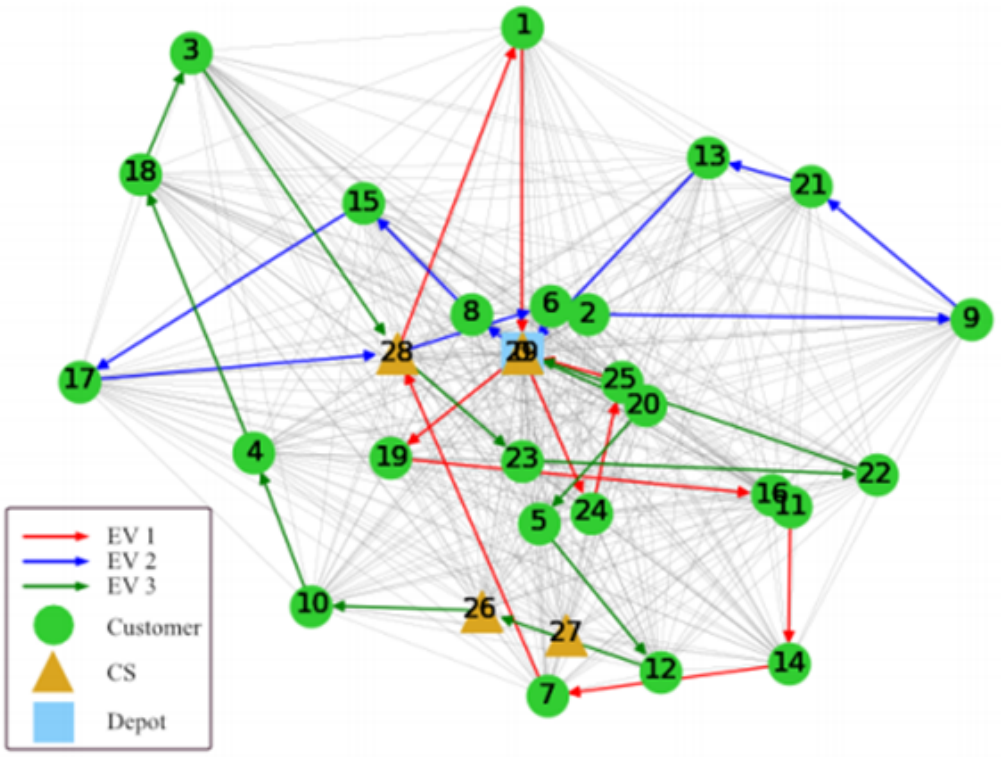


Figura 1.1: Rutas generadas para 3 vehículos eléctricos[2]

1.3. Objetivos

1.3.1. Objetivo General

El objetivo general del trabajo de memoria es el diseño e implementación de una red neuronal entrenada con aprendizaje reforzado para solucionar una instancia del problema de ruteo de vehículos eléctricos, considerando una flota de 2 o más vehículos.

1.3.2. Objetivos Específicos

Para lograr llevar a cabo el objetivo principal, se consideran los siguientes objetivos específicos:

1. Modelar una red de transporte simplificada, la interacción de la flota con la misma y formular el problema de decisión.

2. Proponer distintas estructuras de redes neuronales que, entrenadas con aprendizaje reforzado, resuelvan el problema planteado.

3. Análisis y comparación del desempeño entre las estructuras propuestas mediante simulaciones.

Capítulo 2

Marco Teórico y Estado del Arte

A continuación, se explicarán los conceptos de redes neuronales y aprendizaje reforzado que son fundamentales para entender el trabajo de memoria. Además, se explicarán distintas arquitecturas de redes neuronales utilizadas para resolver el VRP.

2.1. Redes Neuronales

Las redes neuronales consisten en redes distribuidas y paralelas de procesadores simples denominadas neuronas. Recibe su nombre pues nacen a partir del intento de replicar el funcionamiento del cerebro humano.

Estas redes presentan la capacidad de adquirir conocimiento del ambiente a través de distintos tipos de aprendizaje que pueden ser: supervisado, no supervisado y reforzado. Además, son capaces de almacenar el conocimiento adquirido en las conexiones sinápticas, que corresponden a las interacciones entre las neuronas. Estas conexiones pueden cambiar su estructura según la arquitectura de red que se utiliza.

Un gran inconveniente que tienen las arquitecturas clásicas de redes neuronales, es que estas no trabajan muy bien con secuencias de datos. Esto, debido principalmente a que no mezclan información entre las ejecuciones, además de que tratan una secuencia de datos de una sola vez y no elemento a elemento. A raíz de este problema surgen las redes neuronales recurrentes (RNN).

Las redes neuronales recurrentes son una arquitectura de red para analizar datos de series temporales. Este tipo de red tiene las características de tratar secuencias de datos de manera eficiente. Esto, gracias a que recuerda las entradas y salidas anteriores. Además, las RNN pueden tratar secuencias de datos muy largas, elemento a elemento. Sin embargo, cuando las redes recurrentes trabajan con secuencias demasiado extensas, se provoca lo que se conoce como el problema de desvanecimiento de gradiente que, en simples palabras, consiste en la incapacidad de la red de recordar dependencias temporales muy largas. Es por esta razón que nacen las celdas *Long-Short Term Memory* (LSTM).

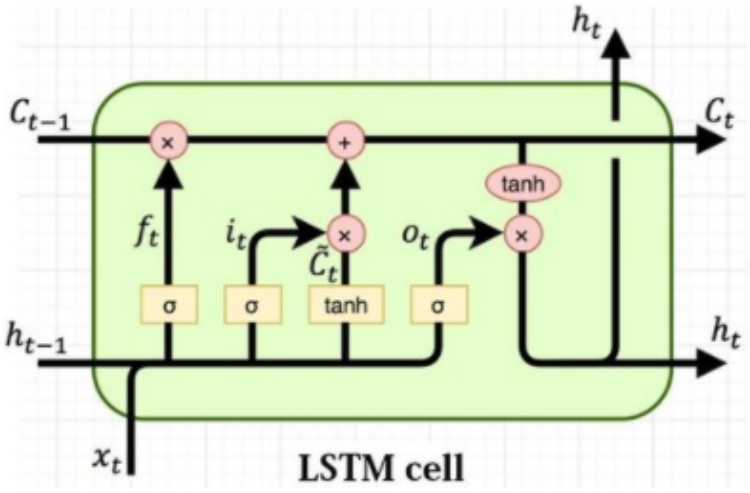


Figura 2.1: Estructura de una celda LSTM[6]

Las celdas LSTM permiten almacenar información de todos las entradas anteriores independiente de su posición. Esto, gracias a que estas celdas poseen un vector denominado vector de estado que se actualiza cada vez que ingresa un elemento a la red. En el diagrama de la Figura 2.1, el vector de estado corresponde a la línea horizontal que atraviesa la parte superior (C_t). Además, la celda tiene como salida un vector de estado oculto o vector de salida h_t. Estos vectores de estado se actualizan con cada entrada y se entregan a la siguiente celda, manteniendo así, la información de entradas pasadas.

El funcionamiento de una celda LSTM se puede dividir en 3 etapas. La primera etapa, consiste en determinar el porcentaje de información que se olvida y se mantiene de los estados anteriores. Esta etapa la realiza la denominada puerta del olvido y se calcula de la siguiente manera:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.1}$$

donde W_f y b_f son parámetros entrenables de la red.

Luego de la etapa de olvido, se define qué nueva información se almacenará en el vector de estado de la celda. Esto, se realiza en dos etapas. Primero, una capa sigmoide, llamada puerta de entrada, decide que valores se van a actualizar. Luego, una capa *tanh* crea un vector de nuevos estados candidatos $\tilde{C}_t$ que podrían agregarse al estado. Con esto se actualiza el estado, multiplicando el estado del paso anterior por f_t y sumando la nueva información.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.2}$$
$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.3}$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.4}$$

Al igual que en la etapa anterior, W_i, W_c, b_i y b_c son parámetros entrenables.

Finalmente, la última etapa consiste en definir lo que será la salida de la celda. Esta salida

se basará en el estado de la celda pero filtrado, de modo que se producen solo las partes que se desea. Para realizar esto, primero se ejecuta una capa sigmoide que define que las partes del estado que se van a generar. Luego, se pasa el estado de la celda por una capa *tanh* y se multiplica por la salida de la capa sigmoide.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.5}$$
$$h_t = o_t * tanh(C_t) \tag{2.6}$$

W_o y b_o son parámetros entrenables.

Como se mencionó anteriormente, las celdas LSTM fueron diseñadas para trabajar con secuencias de datos y es, por esta razón, que han sido muy utilizadas para buscar soluciones al VRP. Una arquitectura de red neuronal basada en LSTM, que ha sido muy utilizada en la resolución del problema de ruteo de vehículos, es la *pointer network*[4]. Este tipo de red neuronal permite resolver problemas de optimización combinatorial donde la salida tenga una dependencia de la entrada.

Este tipo de red, se compone de dos redes neuronales: una codificadora o *encoder*, que toma las coordenadas del vector de entrada y las codifica, y una red decodificadora o *decoder*, que entrega un vector que corresponde a una secuencia del vector de entrada. Ambas redes (*encoder* y *decoder*) corresponden a celdas LSTM.

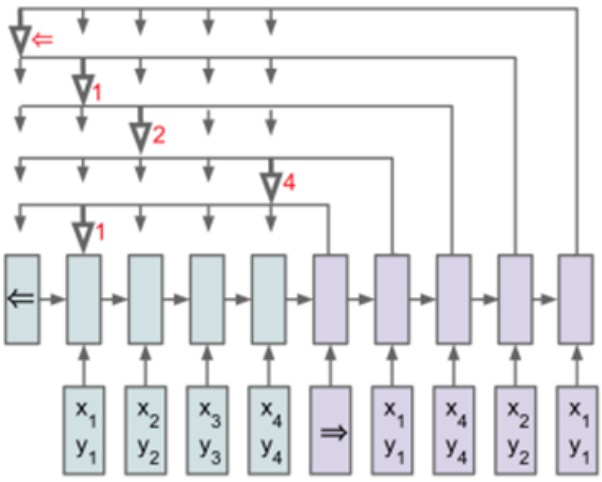


Figura 2.2: Funcionamiento de una *Pointer Network* [4]

La *pointer network* se inicializa con un estado arbitrario entrenable representado por el token (⇐) (ver Figura 2.2); este símbolo marca el comienzo del ingreso de datos. Una vez inicializada la red, comienzan a ingresar los datos, elemento a elemento, a la red *encoder*, actualizando el estado de la misma. Es decir, ingresa el primer elemento de la secuencia, se actualiza el estado de la red y se entrega este estado actualizado al siguiente paso; luego, ingresa el segundo elemento de la secuencia, actualiza el estado y lo entrega al siguiente paso. Así se procede sucesivamente hasta que se ingresa el token (⇒), el cual marca el fin

de la etapa de ingreso de datos y se inicializa la fase de decodificación. En el contexto del problema de ruteo, los elementos de la entrada (pares x_i, y_i) corresponden a representaciones de los destinos, por ejemplo, su coordenada geográfica y la demanda requerida.

Cuando se recibe el token ($\Rightarrow$) comienza a funcionar la red *decoder*, cuyo estado se inicializa como el estado de la red *encoder* hasta ese momento. El proceso de decodificación consiste en encontrar el orden óptimo de elementos de entrada. Para ello, se hace uso de otra red que efectúa un mecanismo de atención, en el cual, se definen pesos para cada elemento de la entrada, los cuales son normalizados con una función *softmax* y este resultado se interpreta como la probabilidad de cada elemento de ser elegido como siguiente destino. Luego, mediante *Beam Search*, se elige el destino a visitar.

$$\begin{aligned} u_j^i &= v^T tanh(W_1 e_j + W_2 d_i) \qquad j \in (1, ..., n) \\ p(C_i | C_1, ..., C_{i-1}, \mathcal{P}) &= softmax(u^i) \end{aligned} \tag{2.7}$$

donde $\mathcal{P}$ es la secuencia de n destinos, y $C^{\mathcal{P}}$ es la secuencia de índices, e_j corresponde al estado oculto del *encoder*, y d_i al estado oculto del *decoder*. v, W_1 y W_2 son parámetros entrenables de la red.

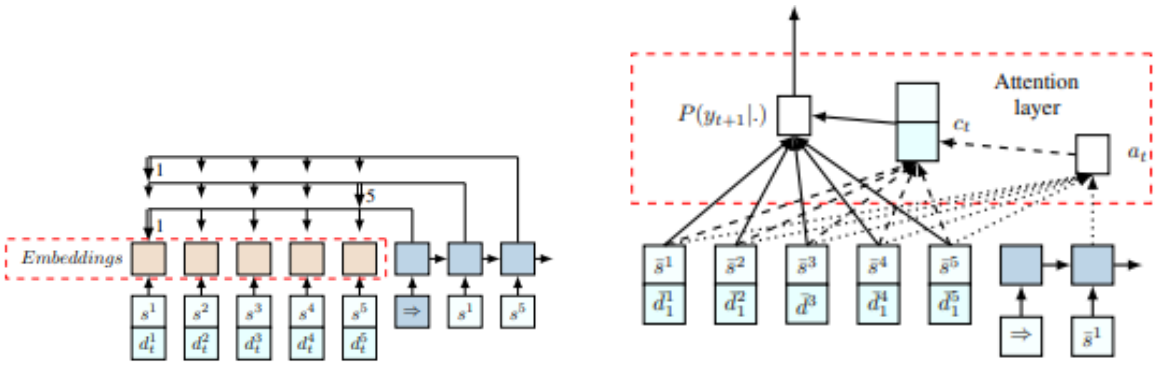
Volviendo al diagrama de la figura 2.2, se recibe el token $\Rightarrow$ y comienza el proceso de decodificación. Entonces, el mecanismo de atención calcula el vector u utilizando el estado oculto de la red *encoder* y el estado oculto de la red *decoder* hasta ese momento. Luego, normaliza el vector y se elige el destino a visitar apuntando su posición en la secuencia de entrada. En este caso, el primer paso del *decoder* apunta hacia el primer elemento (x_1, y_1), luego de esto, se actualiza el estado del decoder y se entrega como entrada el elemento seleccionado en el paso anterior. En el siguiente paso, se vuelve a repetir el proceso: se calcula la probabilidad para cada elemento, se elige el destino (x_4, y_4) y este destino es utilizado como entrada para el siguiente paso. Esto se repite hasta que se recibe el token $\Leftarrow$, el cual marca el fin de la etapa de decodificación.

Los problemas que presenta este tipo de red, indican que la supervisión durante el entrenamiento la limita de encontrar mejores soluciones a la obtenida durante esta etapa y, por otro lado, esta estructura solo se puede aplicar a problemas cuya salida sea discreta. Por lo tanto, al utilizar una estructura *pointer network* solo podremos obtener la secuencia de destinos pero no tendremos los aspectos continuos de la solución, como el tiempo empleado en cada parada, por ejemplo. Es por esta razón que se busca realizar una modificación a esta estructura para poder considerar los aspectos continuos. Además, este tipo de estructura aplica para la obtención de una ruta para un solo vehículo. Por lo tanto, en un principio, nos limitaremos a resolver el problema para un vehículo eléctrico.

En base a esta estructura de red neuronal, se han desarrollado modificaciones para mejorar su desempeño. Una de estas modificaciones se presenta en [7]. En este trabajo se busca resolver el VRP considerando suplir demanda en los destinos, por lo tanto, cada nodo tiene una representación compuesta por una parte estática s (ubicación geográfica) y una parte dinámica d (demanda). El considerar elementos dinámicos hace poco eficiente el uso de la *pointer network* original, pues, cuando un vehículo visita un nodo, se suple la demanda o

parte de ella. Esto provoca que cambie el estado del sistema y, por ende, se debería realizar el proceso de codificación nuevamente, ahora considerando la nueva demanda.

Es por la razón mencionada anteriormente que en [7] realizan una modificación a la estructura de *pointer network*. Esta modificación consiste en la eliminación de la capa *encoder*, utilizando directamente cada elemento de la secuencia de entrada; lo que ellos llaman *"embeddings"*. Esta modificación soluciona el problema de tener que realizar nuevamente la codificación, pues en cada paso se estarán mirando los elementos directamente. Sin embargo, para que esto funcione, se realiza un cambio en el mecanismo de atención. Estas modificaciones se presentan en la Figura 2.3.



(a) Estructura propuesta en [7]

(b) Mecanismo de atención modificado

Figura 2.3: Modificaciones propuestas en [7]

Como en la nueva estructura no se tiene estado *encoder*, se debe modificar el mecanismo de atención para poder incorporar la información de cada entrada. Esta modificación, consiste en calcular primero un vector de *"alineamiento"* a_t, el cual indica qué tan relevante es cada elemento en el paso de decodificación t.

$$a_t = a_t(\bar{x}_t^i, h_t) = softmax(u_t) \tag{2.8}$$

$$u_t^i = v_a^T tanh(W_a[\bar{x}_t^i; h_t]) \tag{2.9}$$

donde $\bar{x}_t^i = (\bar{s}^i, \bar{d}_t^i)$ es la i-ésima entrada y h_t el estado de la red *decoder* en el paso de decodificación t, además, W_a y v_a son parámetros entrenables.

Calculado el vector de alineamiento, se calcula la probabilidad para cada elemento de la entrada de ser elegido como siguiente destino de la siguiente manera:

$$c_t = \sum_{i=1}^{M} a_t^i \bar{x}_t^i \tag{2.10}$$

$$\bar{u}_t^i = v_c^T tanh(W_c[\bar{x}_t^i; c_t]) \tag{2.11}$$

$$P(y_{t+1}|Y_t, X_t) = softmax(\bar{u}_t^i) \tag{2.12}$$

donde W_c y v_c son parámetros entrenables.

La modificación propuesta, disminuye los tiempos de resolución comparado a la estructura *pointer network* original, a la vez que mantiene el desempeño del sistema. Sin embargo, en ese trabajo se resuelve el problema de ruteo de vehículos considerando solo demanda en los destinos. Además, la estructura propuesta solo da solución a los aspectos discretos del problema; nos entrega el orden para visitar los destinos, pero no da cuenta de elementos continuos como sería el tiempo en cada detención. Es por esta razón que se buscaron modificaciones a esta estructura, con el fin de incorporar los elementos continuos de la solución.

2.1.1. Otras arquitecturas de redes aplicadas al VRP

Otra solución propuesta al VRP, se presenta en [8]. Corresponde a una solución basada en el uso de *pointer network*, pero añade las restricciones de operación de vehículos eléctricos y emplea un entrenamiento reforzado profundo. Además, en dicho trabajo, utilizan un algoritmo llamado STRUCT2VEC, el cual consiste en un algoritmo que permite representar un grafo como un vector, manteniendo su identidad estructural. Por lo tanto, la red recibe como entrada la información de los nodos del grafo (x_i), los cuales son procesados por STRUCT2VEC para obtener su representación vectorial μ_i. Posteriormente, estos vectores son utilizados como entrada para la *Pointer network*. La arquitectura propuesta se presenta a continuación en la figura 2.4.

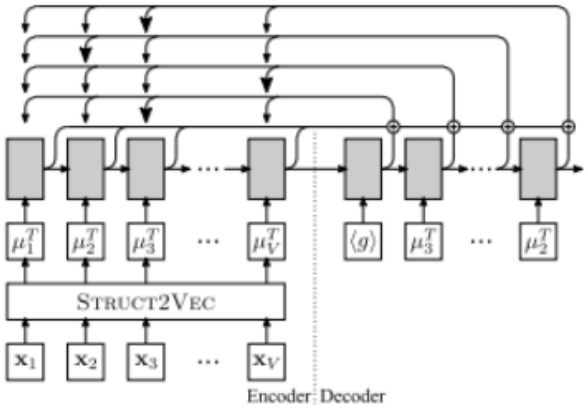


Figura 2.4: Arquitectura de red neuronal propuesta en[8]

En la figura 2.4, g corresponde al estado inicial del *decoder*, el cual es un parámetro entrenable de la red.

Si bien este trabajo resuelve el problema de ruteo para una flota de vehículos eléctricos, el gran problema que tiene esta solución, es que en los resultados presentados, se contem-

plan solo las variables discretas, es decir, el orden de los destinos, la distancia recorrida y la demanda servida, pero no se describe cómo se resuelven las variables continuas como: el tiempo de viaje, tiempo de reabastecimiento, estado de carga del vehículo, etc., cuyo cálculo es importante para el trabajo de memoria que se desea realizar.

2.2. Aprendizaje Reforzado

Otro concepto que es necesario explicar para la comprensión de las soluciones del estado del arte, es el aprendizaje reforzado. El aprendizaje reforzado es un método de aprendizaje, cuya estrategia se basa en un esquema de premios y castigo, donde se evalúa el desempeño del sistema y en base a qué tan cercana del óptimo es la solución encontrada, se le premia o castiga. Este proceso se repite hasta encontrar los parámetros de la red que conducen a la solución óptima, por lo tanto, su entrenamiento conlleva una alta demanda computacional. De esta forma, se asegura encontrar la solución óptima al problema sin necesidad de resolver la optimización como tal y, además, da la oportunidad de continuar el entrenamiento durante la operación del sistema.

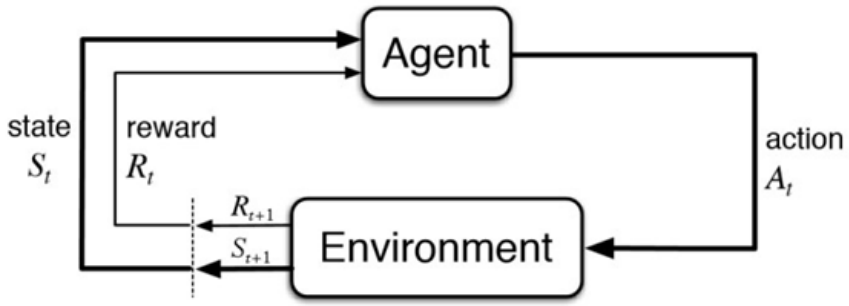


Figura 2.5: Esquema de aprendizaje reforzado[9]

En el esquema de la Figura 2.5, el agente es quien realiza la acción. Esta acción se evalúa en el ambiente y se obtiene un estado, que es el resultado de la acción aplicada y, una recompensa, que es una métrica que indica que tan cercano al óptimo fue el resultado obtenido. Posteriormente, las señales del estado y recompensa se retroalimentan al agente, quien realiza cambios en base a sus resultados y ejecuta otra acción. De esta forma, se procede iterativamente.

Es importante mencionar que el agente puede estar conformado por dos redes. Una red, denominada actor, la cual elige la acción a realizar y, una red llamada crítica, la cual predice la recompensa que se obtendrá con la acción e informa al actor sobre los cambios que debería realizar.

2.2.1. Algoritmo de aprendizaje reforzado aplicado al VRP

Haciendo una analogía de los elementos presentados en la figura 2.5 al problema de ruteo de una flota de vehículos eléctricos, el agente sería un sistema de control inteligente encargado de tomar decisiones sobre las rutas que deben seguir los vehículos para minimizar los costos

operativos y cumplir con las restricciones establecidas. El ambiente representaría el entorno en el cual operan los vehículos eléctricos, incluyendo los destinos a ser visitados, la capacidad de carga de cada vehículo, estado de carga de batería de cada vehículo y cualquier otra variable relevante. La recompensa se definiría como una medida cuantitativa de la eficiencia y el rendimiento del sistema, considerando factores como la duración del viaje, el consumo de energía y, posiblemente, la optimización de recursos.

En este contexto, las acciones del agente serían las decisiones que toma para determinar las rutas específicas de cada vehículo eléctrico. Estas acciones podrían incluir la selección de destinos, la asignación de vehículos a rutas particulares y la gestión de la carga de batería para optimizar la autonomía. La retroalimentación del ambiente en forma de recompensas, permitiría al agente aprender y ajustar sus estrategias a lo largo del tiempo, con el objetivo de mejorar continuamente la eficiencia del ruteo de la flota de vehículos eléctricos.

En [7], además de la arquitectura de red explicada en la sección anterior, se hace uso de aprendizaje reforzado para entrenar la red, implementando el algoritmo que se muestra a continuación, denominado REINFORCE. Este algoritmo corresponde a un tipo de aprendizaje reforzado Actor-Crítico, donde la red Actor es la que realiza las acciones y, en base a las observaciones del estado y la recompensa obtenida a partir de esa acción, la red Crítica retroalimenta los resultados obtenidos al Actor para que se actualice la política. El objetivo del entrenamiento en este *paper* consiste en encontrar la ruta más corta utilizando la siguiente función objetivo:

$$J(\theta|s) = \mathbb{E}_{\pi \sim p_\theta(.|s)} L(\pi|s) \tag{2.13}$$

Para el caso del trabajo de memoria, en esta función objetivo se pueden incluir las restricciones operacionales, como pesos, de tal forma que penalice fuertemente las rutas que no cumplan con alguna de ellas. La implementación de este algoritmo se presenta a continuación en el Algoritmo17.

El algoritmo comienza inicializando aleatoriamente los pesos θ y ϕ, asociados a las redes actor y crítica, respectivamente. Luego, se realizan iteraciones sobre las cuales se muestrean N instancias de problemas de acuerdo a una distribución de probabilidad sobre la familia de problemas de ruteo $\mathcal{M}$. Posteriormente, para cada instancia, se calcula la ruta óptima y se calcula su recompensa R^n. Luego, con los valores de recompensa calculadas (R^n) y predichas por la red crítica ($V(X_0^n; \phi)$), se calcula $d\theta$ y $d\phi$ con los cuales se actualizan los pesos θ y ϕ. Este es el algoritmo que se utilizó en este trabajo de memoria.

Algorithm 1 Algoritmo de REINFORCE

Se inicializan las redes actor y crítica con pesos aleatorios θ y ϕ, respectivamente
for iteración = 1, 2, ... **do**
    se reinician los gradientes: $d\theta \leftarrow 0$, $d\phi \leftarrow 0$
    se muestrean N instancias de la familia de problemas $\Phi_{\mathcal{M}}$
    for n = 1, ..., N
        se inicializa el conteo de pasos $t \leftarrow 0$
        repetir
            se escoge y^n_{t+1} de acuerdo a la distribución $P(y^n_{t+1}|Y^n_t, X^n_t)$
            se observa el nuevo estado X^n_{t+1}
            $t \leftarrow t + 1$
        hasta cumplir condición de término
        se calcula el reward $R^n = R(Y^n, X^n_0)$
    end for
    $d\theta \leftarrow \frac{1}{N}\sum_{n=1}^{N}(R^n - V(X^n_0;\phi))\nabla_\theta log P(Y^n|X^n_0)$
    $d\phi \leftarrow \frac{1}{N}\sum_{n=1}^{N}\nabla_\phi(R^n - V(X^n_0;\phi))^2$
    actualizar θ usando $d\theta$ y ϕ usando $d\phi$
end for

2.3. Otras soluciones al VRP

Las soluciones presentadas hasta ahora, hacen uso de redes neuronales para resolver el VRP, sin embargo, la estrategia más utilizada es la resolución del problema de optimización como tal. En [2], por ejemplo, se resuelve el problema de ruteo de una flota de vehículos eléctricos calculando la optimización empleando algoritmos genéticos. La estrategia utilizada en dicho trabajo consiste en dividir la operación en pre-operación y operación *online*. En pre-operación, se calculan las rutas iniciales resolviendo un E-VRP *offline*. Por otra parte, en operación *online*, se actualizan las rutas según realizaciones del tráfico y mediciones del estado de los vehículos eléctricos resolviendo un E-VRP *online*, que permite mejorar la operación y calidad de servicio. Los resultados presentados en este trabajo dan cuenta de un buen desempeño general del sistema, logrando completar el *tour* con un mínimo de violación a la restricciones. Además, los tiempos de cómputo no superan los 45 minutos para la operación *offline* y los 5 minutos para la operación *online*. Sin embargo, el gran inconveniente de esta estrategia es que para cada instancia se deberá resolver las etapas *offline* y *online*, lo que conlleva mucho tiempo y recursos pensando en una aplicación durante la operación.

Otro ejemplo de resolución del problema de optimización, se presenta en [10], donde abordan el VRP para vehículos eléctricos (E-VRP), añadiendo un modelo de consumo energético realista y aplicándolo a un modelo realista de la ciudad de Beijing. En este trabajo logran encontrar soluciones en tres horas, aproximadamente. Nuevamente, si se piensa en una aplicación durante la operación, la estrategia presentada en este trabajo se hace imposible de aplicar por los tiempos de cómputo.

Otros métodos de solución hacen uso de heurísticas para simplificar parte del problema y se asumen algunos aspectos que aminoran el tiempo de resolución [11]. En [12], por ejemplo, se aborda el problema de VRP separando las rutas de cada vehículos para ser tratados como un Problema del Vendedor Viajero (TSP, por si nombre en inglés) simple. De esta forma, se

simplifica el problema original logrando optimizar los tiempos de resolución. Sin embargo, estos métodos tienen el problema que no se puede comprobar que las soluciones obtenidas sean las óptimas.

Capítulo 3

Diseño e Implementación de las Soluciones Propuestas

3.1. Metodología

Para desarrollar el trabajo de memoria, primero se realiza la formulación de la instancia del problema que se desea resolver y, luego, se utiliza una metodología basada en una iteración de dos actividades: la primera, es idear una estructura de red neuronal y, la segunda, es implementar el entrenamiento y estudiar el desempeño de la red propuesta. Este proceso se repite hasta alcanzar un desempeño adecuado (ver Figura 3.1).

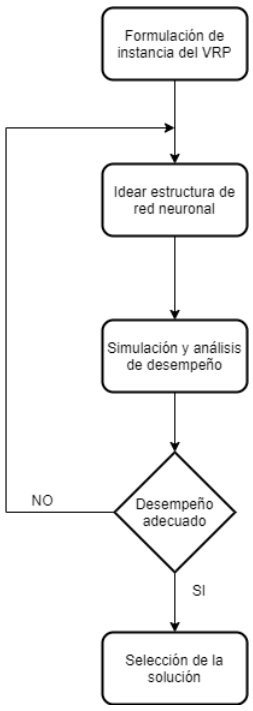


Figura 3.1: Esquema de la metodología propuesta

Para definir el desempeño de las soluciones simuladas, se realiza un análisis técnico-económico que considera las siguientes variables:

- Largo de la secuencia
- Tiempo de la secuencia
- Energía consumida
- Costo económico

La metodología descrita anteriormente es elegida, pues, dado que las estructuras propuestas en el estado del arte no presentan solución a los aspectos continuos de la resolución del VRP, surge la necesidad de realizar modificaciones a estas estructuras de redes y, en base a prueba y error, determinar, a través del desempeño en las simulaciones, si se logran incorporar exitosamente estos aspectos.

3.2. Instancia de VRP a solucionar

Para poder presentar las redes neuronales diseñadas, es fundamental caracterizar la instancia del problema de ruteo que se está abordando. Para este trabajo, se considera una flota de 3 vehículos eléctricos los cuales deben satisfacer la demanda de cierta cantidad de clientes. Además, los vehículos tienen limitaciones de capacidad de cargamento y batería, por lo tanto, si durante el tour algún vehículo de la flota se queda sin cargamento o con batería insuficiente para visitar algún cliente, este deberá reabastecerse en el depósito o en alguna estación de carga de batería, respectivamente. Es importante mencionar que también se define una cota máxima y una cota mínima para el estado de carga de batería ($SoC_{max} y SoC_{min}$), de esta forma los vehículos deberán mantener su estado de carga en este rango durante la operación y cualquier violación a estos límites se traducirá en una penalización en la función de recompensa.

A continuación, en la figura 3.2 se presenta un diagrama de cómo sería una disposición de los elementos en la instancia mencionada anteriormente, considerando 10 clientes y 3 estaciones de carga.

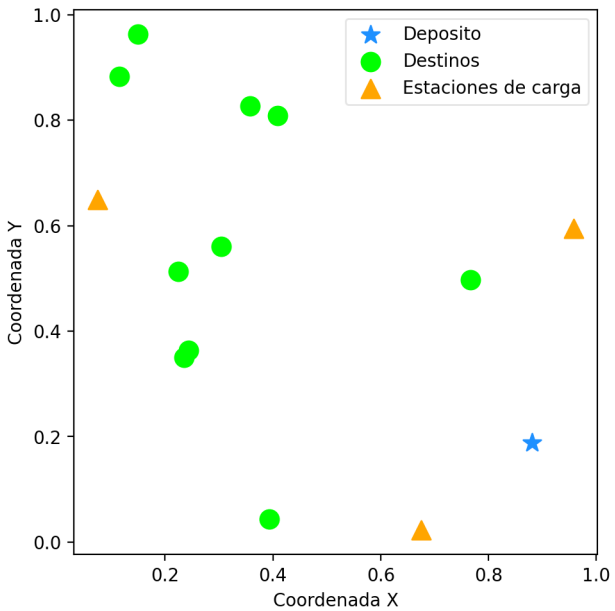


Figura 3.2: Ejemplo de una instancia del VRP a resolver

Como se observa en la figura 3.2, las ubicaciones geográficas del depósito, los destinos y las estaciones de carga, son generadas aleatoriamente en un cuadrado unitario. Sin embargo, para poder realizar una simulación más realista, se considera una conversión de escalas de $1:20km$.

También es importante destacar que esta instancia del VRP, considera una simplificación bastante potente, que considera que desde un punto se puede llegar a todos los demás con una linea recta. Esta simplificación evidentemente se aleja de la realidad pero ayuda a tener un buen punto de partida en caso de querer resolver un problema considerando datos de una red de transporte real.

Como se mencionó anteriormente, en este trabajo se utilizará el algoritmo REINFORCE, de aprendizaje reforzado, y como se vio en la sección 2.2.1, para utilizar este algoritmo se hace necesario definir la función de recompensa.

3.2.1. Diseño de función de recompensa

La función *reward* utilizada incorpora los siguientes aspectos: largo total de viaje, tiempos de reabastecimiento y penalizaciones al violar las restricciones del SoC. La función de recompensa diseñada se presenta a continuación:

$$\begin{gathered} J_1 = \frac{1}{v_{\text{promedio}}} \sum_{i=1}^{N_{\text{Flota}}} \text{distancia del tour}_i \\ J_2 = \begin{cases} 0 & \text{if SoC} > 0 \\ 1000 & \text{if SoC} = 0 \end{cases} \\ J_3 = \,\% \text{ reabastecimiento} \cdot \text{tiempo de carga completa} \\ J_4 = \begin{cases} 75 \cdot SoC^2 + 3 \cdot SoC & \text{if SoC} > soc_{max} \\ 150 \cdot SoC^2 + 3 \cdot SoC & \text{if SoC} \leq soc_{min} \end{cases} \\ R = -(J_1 + J_2 + J_3 + J_4) \end{gathered} \tag{3.1}$$

Donde, en la ecuación de J_4 , $x = |SoC - SoC_{min}|$, $v_{promedio} = 30\ [km/h]$ y tiempo de carga completa $= 2\,[h]$.

En la ecuación 3.1, J_1 corresponde a la suma del tiempo de viaje de cada vehículo de la flota, J_2 corresponde a una penalización que se activa en caso de llegar a $SoC = 0$, es decir, en caso que algún vehículo quede en mitad del tour sin batería suficiente para ir a ningún destino. J_3 es el tiempo invertido en recargas de batería a lo largo del tour y, por último, J_4 corresponde a penalizaciones en caso que se viole la restricción de SoC mínimo y SoC máximo. Es importante mencionar que se penaliza más fuertemente las violaciones al SoC mínimo por el riesgo que existe a que los vehículos queden sin carga al violar esta restricción. Además, se utiliza el *reward* como la suma negativa de los costos (J), pues durante el entrenamiento, se busca que el sistema maximice la función de recompensa.

Hasta ahora, se ha definido la instancia del VRP a resolver y la función de recompensa que será utilizada en el algoritmo de aprendizaje reforzado, sin embargo, aún falta explicar las estructuras de las redes neuronales diseñadas para la resolución del problema.

3.3. Diseño e implementación de las redes neuronales

En esta sección se presentan las arquitecturas de red de las soluciones desarrolladas durante este trabajo; las cuales corresponden a modificaciones de la arquitectura de red presentada en [7]. Se utiliza como base la *Pointer Network*, pues como se mencionó en el capítulo anterior, esta arquitectura de red resuelve muy bien el problema de la secuencia de destinos, por lo que solo basta realizarle modificaciones que resuelvan el problema de pausas de reabastecimiento. A continuación, se presenta la arquitectura de la *Pointer Network* de [7]. Es importante mencionar que la arquitectura que se presenta en la Figura 3.3 es la misma que la Figura 2.3 pero presentada en forma de diagrama de flujo para explicar de mejor manera las operaciones realizadas desde la entrada hasta la salida.

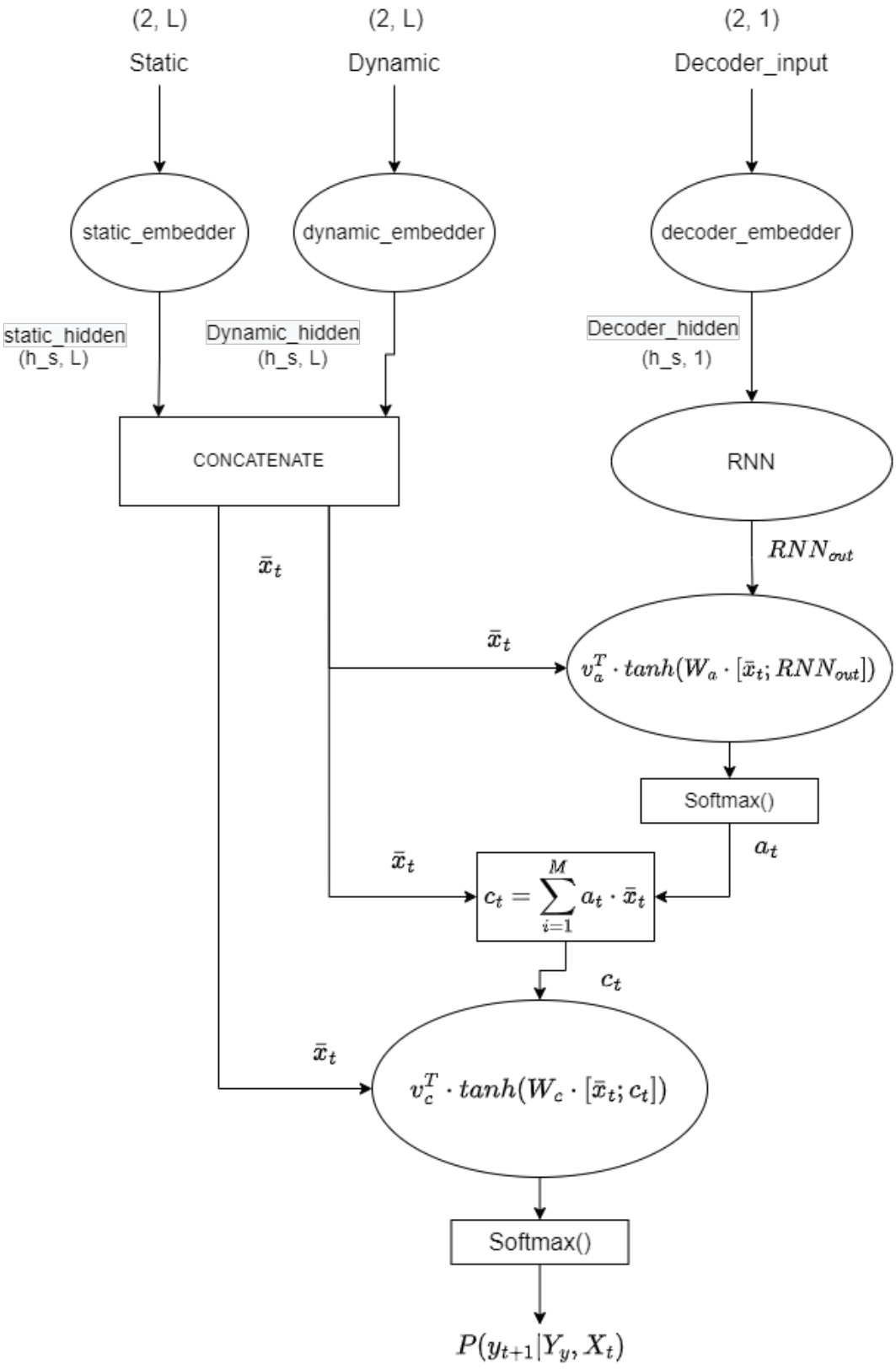


Figura 3.3: Arquitectura de red propuesta en [7]

En la Figura 3.3, "**;**" significa la concatenación de los vectores, L corresponde a la cantidad de destinos (incluyendo el depósito [1+N°clientes]), los óvalos corresponden redes neuronales

(módulos con parámetros entrenables) y los rectángulos, a operaciones sobre los vectores.

Como se observa en la Figura 3.3, la red tiene 3 inputs:

- ***Static:*** corresponde a la información del sistema que se mantiene invariable a lo largo del tour. En este caso, la ubicación geográfica del depósito y los destinos a visitar (coordenadas x,y) (ver Figura 3.4).

x	x_0	x_1	x_2	$\dots$	$\dots$	x_{L-1}
y	y_0	y_1	y_2	$\dots$	$\dots$	y_{L-1}

Figura 3.4: Static

- ***Dynamic:*** corresponde a la información que va evolucionando durante el trayecto del vehículo (recordar que en [7] se considera el VRP de un solo vehículo). En esta implementación, tiene la información de cargamento del vehículo y demanda en los destinos.

Cargamento	l	l	l	$\dots$	$\dots$	l
Demandas	d_0	d_1	d_2	$\dots$	$\dots$	d_{L-1}

Figura 3.5: Dynamic

  Como la información del cargamento es propia del vehículo, en el *dynamic* se repite L veces para coincidir con las dimensiones de las demandas, tal como se observa en la Figura 3.5. Además, como en el depósito no se debe satisfacer demanda, se cumple siempre $d_0 = 0$.

- ***Decoder Input:*** corresponde a la información del *Static* del último destino visitado, es decir, la última ubicación geográfica visitada. Como el tour comienza en el depósito, en $t = 0$ el *decoder input* es la ubicación geográfica del depósito.

Continuando con el diagrama de la Figura 3.3, estos 3 *inputs* ingresan a sus respectivos *embedders* , los cuales corresponden a una red neuronal sencilla que realiza la siguiente operación:

$$y = W^T x + b$$

Con W y b parámetros entrenables.

Los *embedders* cumplen la función de aumentar la dimensión de los *inputs*, de forma de caracterizar de mejor manera esa información. En este caso, los *embedders* llevan a los *inputs* a una dimensión $h_s = 128$. Luego de los *embedders*, se tiene una RNN correspondiente a una celda LSTM, que procesa la información del *decoder hidden*. Las 2 redes neuronales restantes presentes en el diagrama de la Figura 3.3, realizan lo que sería el mecanismo de atención y poseen la misma estructura, de la forma:

$$v^T \cdot tanh(W \cdot x)$$

.

con v y W parámetros entrenables.

En la figura 3.3, además de las redes neuronales, destaca la utilización de dos funciones *Softmax*. La función *softmax* convierte un vector de K números reales en una distribución de probabilidad de K resultados posibles, realizando la siguiente operación:

$$Softmax(x_i) = \frac{exp(x_i)}{\sum jexp(x_j)}$$

En el diagrama de la figura 3.3, el resultado de la primera función *Softmax* es interpretado como un vector de alineamiento, el cual indica qué tan relevante es cada elemento para la decodificación. Por otra parte, la salida de la segunda función *softmax* se interpreta como la probabilidad de cada elemento de ser elegido como siguiente destino.

Para considerar la existencia de una flota de vehículos e incluir la restricción de pausas de reabastecimiento de batería, se modificó tanto la arquitectura de la red como los *inputs*, de manera de incorporar la información necesaria para resolver el problema bajo estas nuevas condiciones.

3.3.1. Solución 1: Carga completa

La primera solución implementada considera pausas completas de reabastecimiento de batería, es decir, cada vez que un vehículo se detenga en una estación de carga, se cargará lo suficiente para llegar a un $SoC = SoC_{max}$.

La primera modificación necesaria a la arquitectura presentada anteriormente, fue la incorporación de las ubicaciones geográficas de las estaciones de carga. Como las ubicaciones no varían durante el tour, esta información se agregó al *Static*. Es importante mencionar que estos nodos de carga tienen demanda nula y se consideran nodos opcionales, es decir, no es obligación que los vehículos los visiten, sino que solo los visitarán en caso de tener que cargar batería.

La segunda modificación a la arquitectura, consistió en la incorporación de la información del SoC de cada vehículo. Como el SoC es un elemento que va cambiando a lo largo del tour (luego de visitar cada destino el SoC vendrá dado por $SoC_t = SoC_{t-1} - Consumo$), esta información se agregó al *Dynamic* y, como es propio de los vehículos, se repite L veces tal como el cargamento. Recordemos que L ahora también considera las estaciones de carga (tiene tamaño [1 + Nºclientes + Nºestaciones de carga]).

Luego de algunas pruebas al modelo, se consideró apropiado que además del SoC, el sistema cuente con la información del consumo de batería que significa el visitar todos los destinos desde el último destino visitado. Por lo tanto, un ejemplo de la estructura del *dynamic* para

una flota de 3 vehículos quedaría definida en la Figura 3.6.

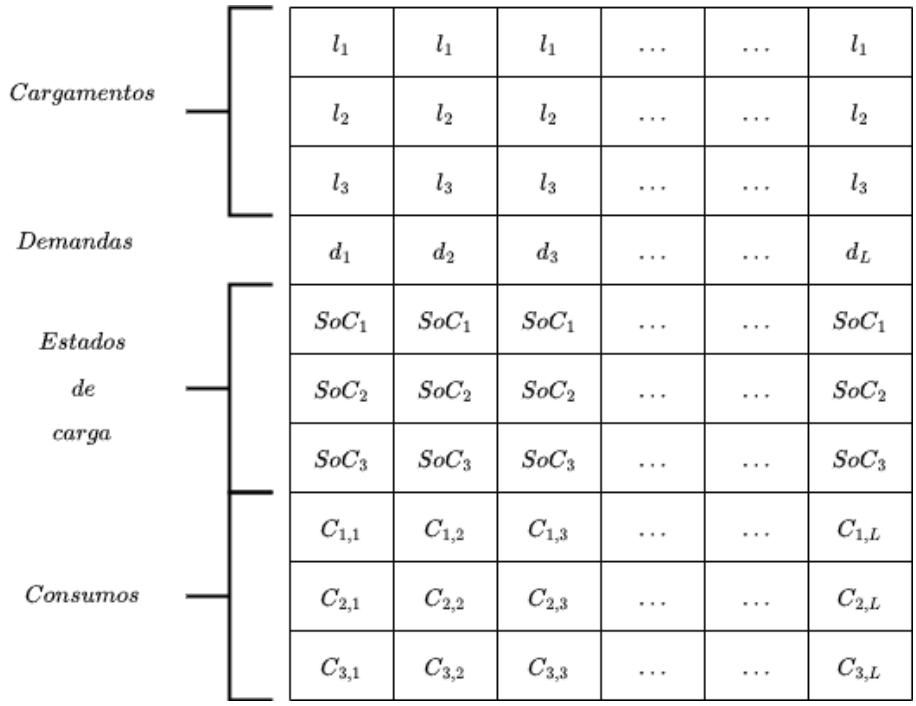


Figura 3.6: *Dynamic* modificado

Como estamos añadiendo más información al *dynamic*, se hace necesario el aumento de dimensionalidad del *dynamic embedder*, seteando $h_s = 256$ (el h_s del *static* y *decoder input* permanecen igual, $h_s = 128$). Considerando las modificaciones al *Static*, *Dynamic* y tamaño del *dynamic embbeders*, la arquitectura de la red neuronal queda representada en la Figura 3.7

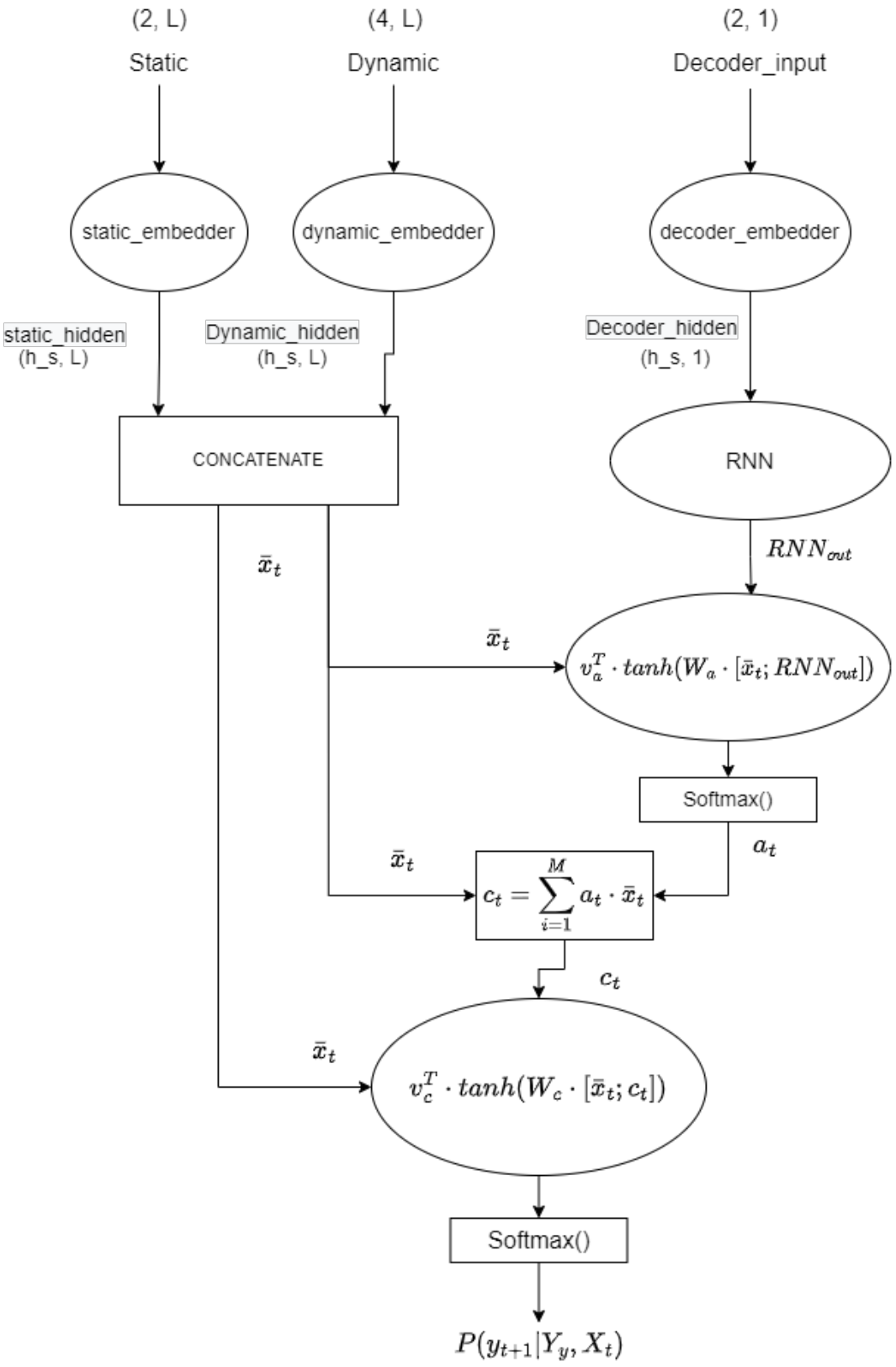


Figura 3.7: Arquitectura de la red neuronal de carga completa

3.3.2. Solución 2: Cargas discretas

Otra solución diseñada, considera estaciones de carga ficticias, que comparten ubicación geográfica, pero con distintos rangos de % de carga disponibles, por lo que los vehículos cargarán lo suficiente para llegar al nivel seleccionado. Como la información de los % de carga son propios de cada estación, y estos no cambian a lo largo del tour, esta información se añadió al *static*. Por ejemplo, si se consideran 2 estaciones de carga con 4 particiones de % carga, la información del *static* correspondiente a esas estaciones de carga viene representada a continuiación en la 3.8:

x	x_i	x_i	x_i	x_i	x_j	x_j	x_j	x_j
y	y_i	y_i	y_i	y_i	y_j	y_j	y_j	y_j
$\%cargas$	25%	50%	75%	SoC_{max}	25%	50%	75%	SoC_{max}

Figura 3.8: Ejemplo de información de estaciones de carga para modelo discreto

De esta forma, el *static* para este modelo viene definido por:

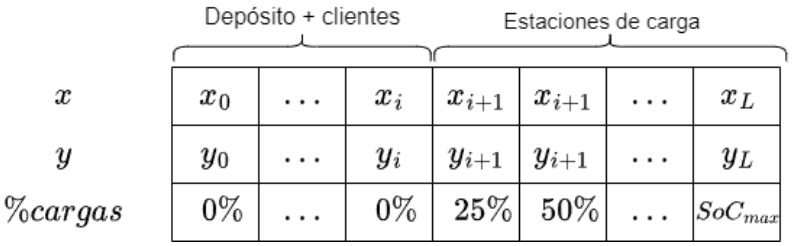


Figura 3.9: *Static* del modelo discreto

Como se observa en la Figura 3.9, los primeros i elementos del *static* corresponden a las ubicaciones geográficas y porcentajes de carga del depósito y los destinos a visitar, es importante notar que estos primeros i elementos tienen $\%carga = 0$ pues los vehículos no cargan batería en esos destinos. Por otra parte, los elementos $i+1$ hasta L corresponden a las estaciones de carga disponibles. Recordar que estos nodos de carga no son paradas obligatorias y los vehículos pueden o no visitarlas de acuerdo a sus necesidades de cargar batería.

Nuevamente, como estamos añadiendo información al *static*, se hace necesario cambiar la dimension del *static embedder* y *decoder input*, definiendo $h_s = 192$. El *dinamic* en este caso permanece igual al utilizado en la arquitectura anterior (ver Figura 3.6). Considerando las modificaciones realizadas al *static* y los cambios de dimension del *static embedder* y *decoder input*, la red resultante queda representada en la Figura 3.10.

La solución implementada en este trabajo considera 3 estaciones de carga con 4 opciones de carga disponible: 25 %, 50 %, 75 % y SoC_{max}.

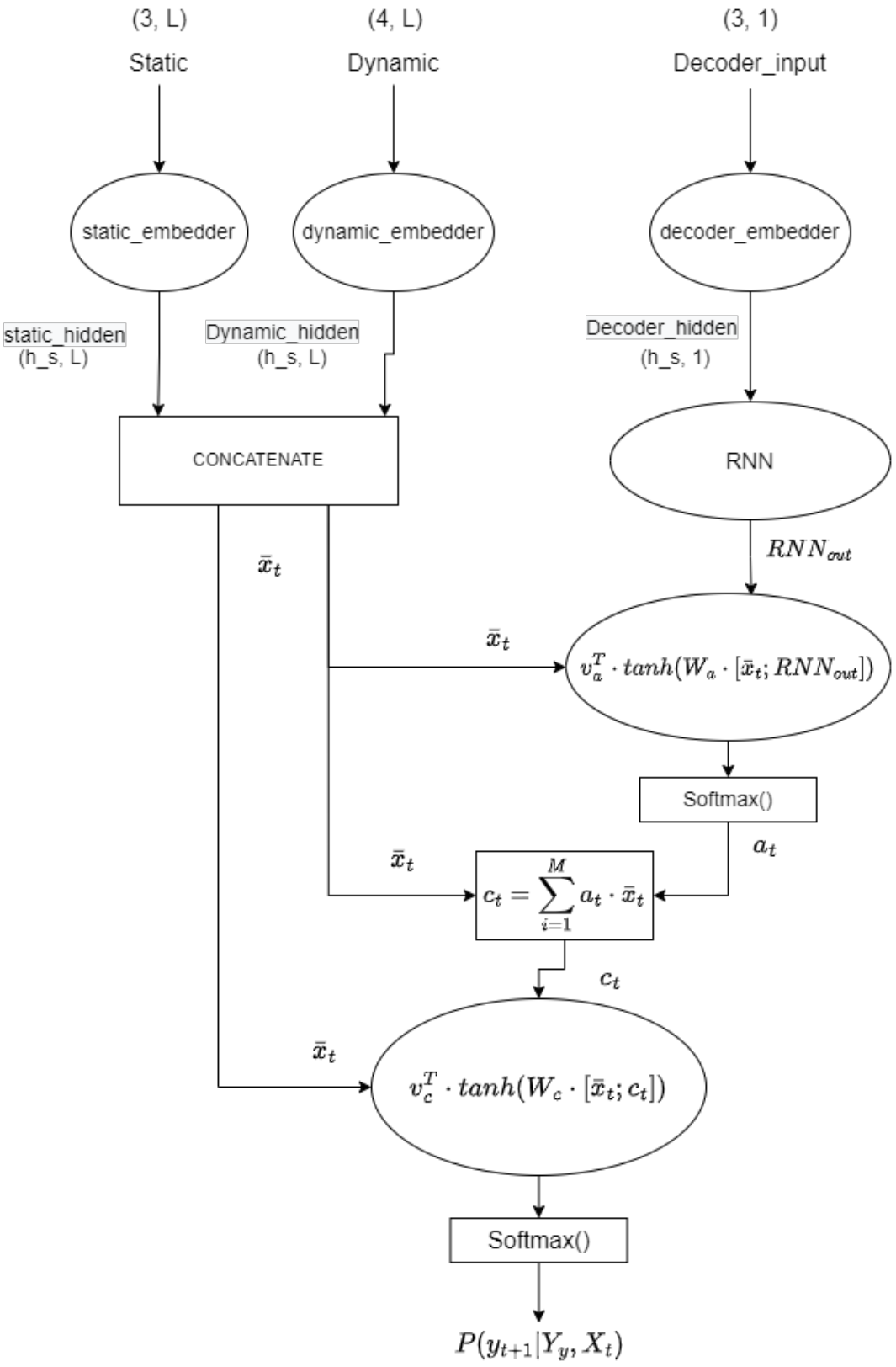


Figura 3.10: Arquitectura de la red neuronal de carga discreta

3.3.3. Solución 3: Carga continua

La tercera solución propuesta busca añadir un nuevo *output* continuo a la red neuronal, el cual nos indicará la cantidad exacta de batería que los vehículos cargarán en caso de detenerse en una estación de carga. Para lograr esto, se implementó una red neuronal extra a las descritas en el modelo de la Figura 3.3. Esta red extra realiza la siguiente operación.

$$sigmoid(W \cdot c_t)$$

Con W un parámetro entrenable.

Esta red consiste en un módulo que a partir de la información del contexto c_t, calcula una variable continua, la cual es interpretada como la cantidad a cargar. Es importante mencionar que, este nuevo *output* es tomado en consideración únicamente cuando algún vehículo de la flota visita una estación de carga. Es decir, en esta solución se obtienen 2 salidas, una es el siguiente destino a visitar y, la otra, es el porcentaje de batería que se quiere cargar, sin embargo, esta segunda salida solo se considera cuando el destino visitado corresponde a una estación de carga y es ignorada en otros casos. Al igual que en las soluciones 1 y 2, los nodos de carga son opcionales y los vehículos pueden elegir si visitarlos o no de acuerdo a sus necesidades.

Debido a lo anterior, la estructura del *static* y *dynamic* son las mismas que para el modelo de la solución 1 (carga completa), por lo tanto, las dimensiones de los *embedders* también son las mismas. La arquitectura de esta red neuronal se presenta a continuación en la Figura 3.11.

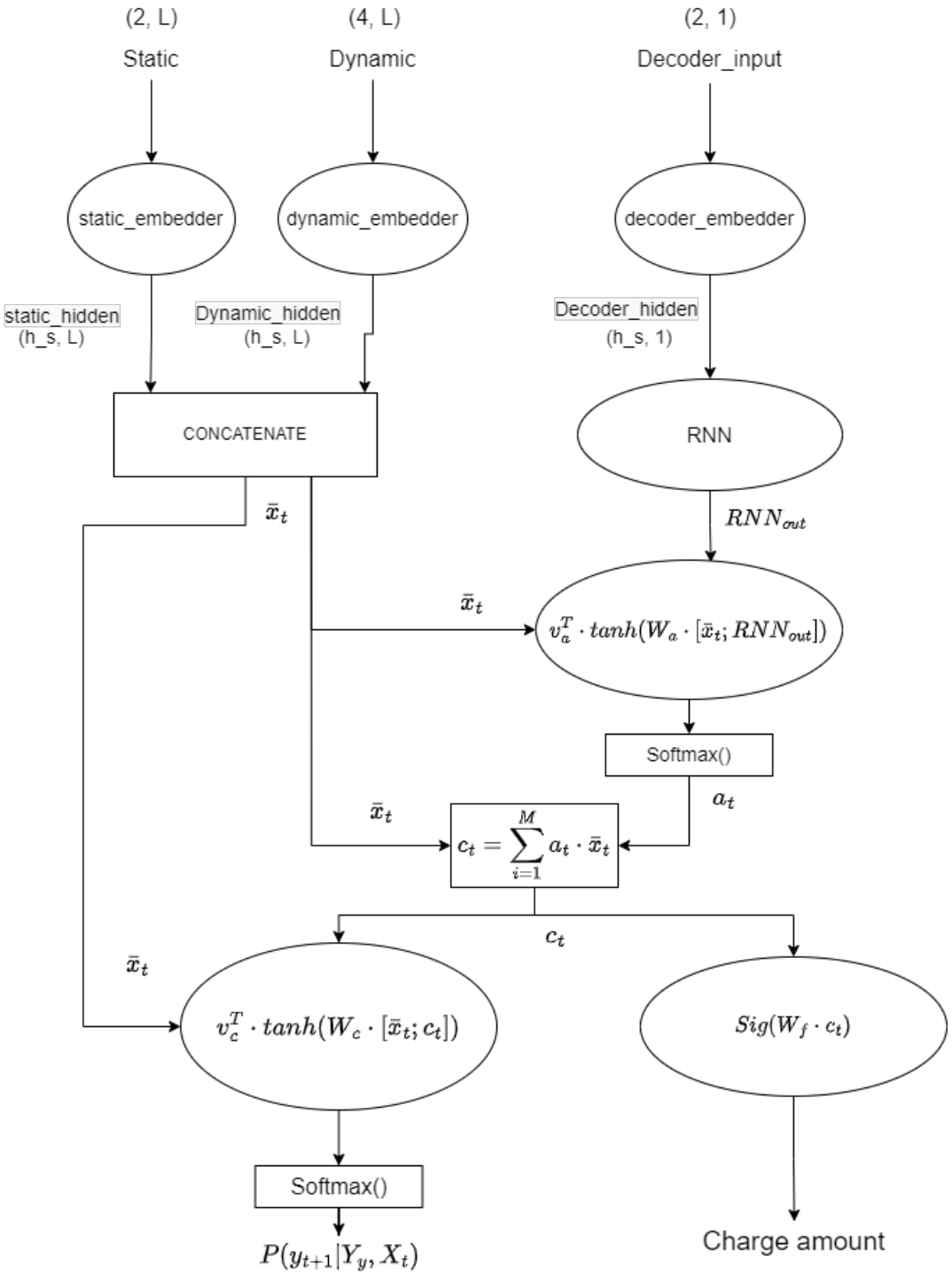


Figura 3.11: Arquitectura de la red neuronal de carga continua

Como se observa en la Figura 3.11, esta red neuronal extra que indica el monto de carga, se añade al final de la red y utiliza la información del contexto c_t al igual que la red que calcula la secuencia.

Capítulo 4

Resultados y Análisis de las Simulaciones

4.1. Configuración

4.1.1. Elección de parámetros de la instancia a resolver

Para este trabajo se utilizaron los siguientes parámetros:

- Nº de vehículos en flota: 3
- Nº de destinos: 10
- Nº de estaciones de carga: 3
- Nº cargas máximas: 2
- Máxima demanda en los destinos: 9
- Máxima capacidad de cargamento del vehículo: 20
- $SoC_{min} = 20\,\%$
- $SoC_{max} = 90\,\%$
- Consumo de batería por unidad: 40 % (corresponde a cuanto nivel de batería se consume al realizar un viaje de largo 1)

Es decir, el problema abordado consiste en una flota de 3 vehículos, los cuales deben satisfacer la demanda de 10 destinos diferentes, considerando un máximo de 2 pausas de reabastecimiento de batería para mantener su estado de carga (SoC) entre 90 % y 20 %.

4.1.2. Entrenamiento y extracción de resultados

Para el entrenamiento de las redes propuestas, como se mencionó anteriormente en este trabajo, se utilizó el algoritmo de REINFORCE y fue realizado en un servidor con las siguientes especificaciones:

- Modelo: Dell PowerEdge R740
- CPU: Intel Xeon Silver 4216 2.10GHz
- Número de Núcleos: 64
- RAM: 32 GB
- GPU: NVIDIA Tesla P40, 24 GB

Vale decir que los modelos fueron entrenados con un conjunto de entrenamiento de 1 millón de instancias (el conjunto fue diferente para cada modelo) y se realizó el entrenamiento en 3 modelos de cada una de las 3 soluciones (9 entrenamientos en total).

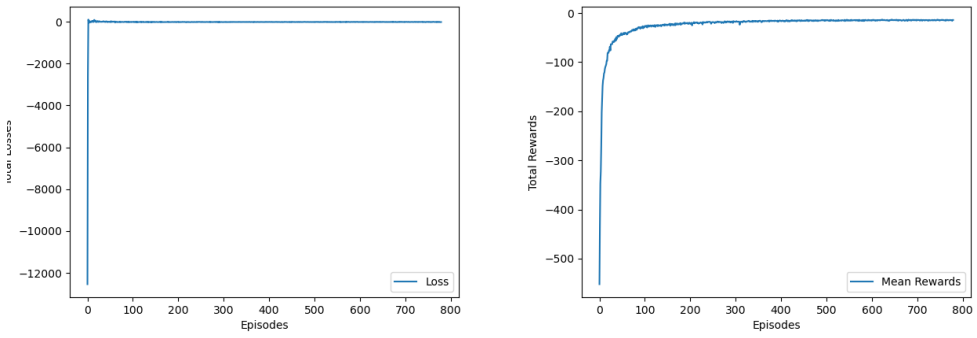
Por otra parte, la obtención de los resultados se realizó sobre un conjunto de prueba de 500 instancias, el cual fue el mismo para los 9 modelos entrenados.

4.2. Resultados

A continuación, se mostrarán los resultados obtenidos por los tres modelos para el problema descrito anteriormente:

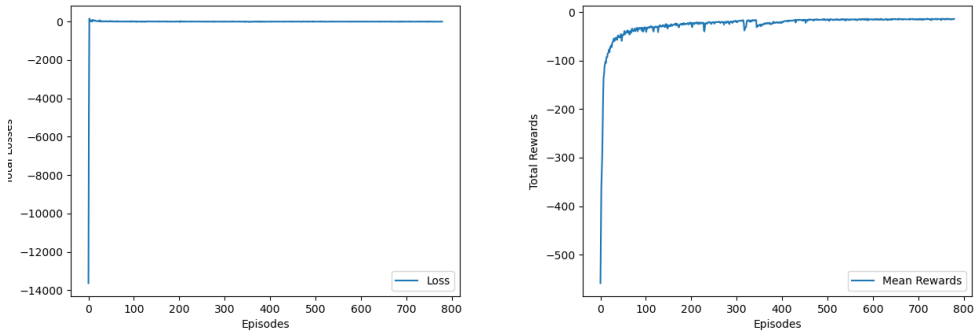
4.2.1. Modelos de carga completa

Para comenzar con los resultados, primero se presentan los gráficos de la evolución de *loss* y *reward* durante los entrenamientos de los modelos. El *loss* corresponde al valor de la función objetivo. El entrenamiento para cada uno de estos modelos tardó aproximadamente 6 horas.



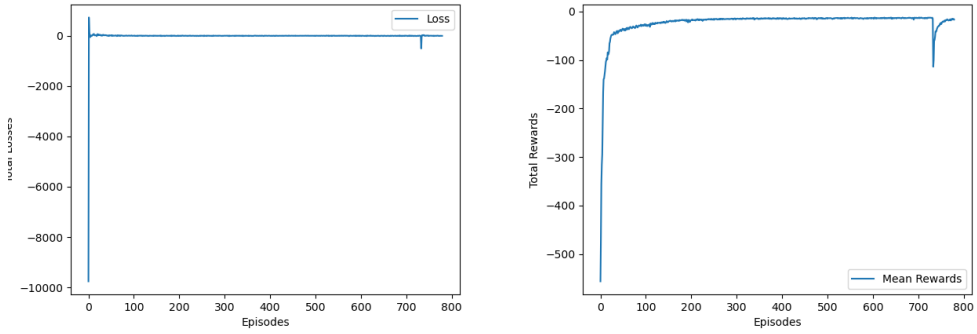
(a) Loss (b) Reward

Figura 4.1: Evolución *loss* y *reward* durante el entrenamiento del modelo 1 de carga completa



(a) Loss (b) Reward

Figura 4.2: Evolución *loss* y *reward* durante el entrenamiento del modelo 2 de carga completa



(a) Loss (b) Reward

Figura 4.3: Evolución *loss* y *reward* durante el entrenamiento del modelo 3 de carga completa

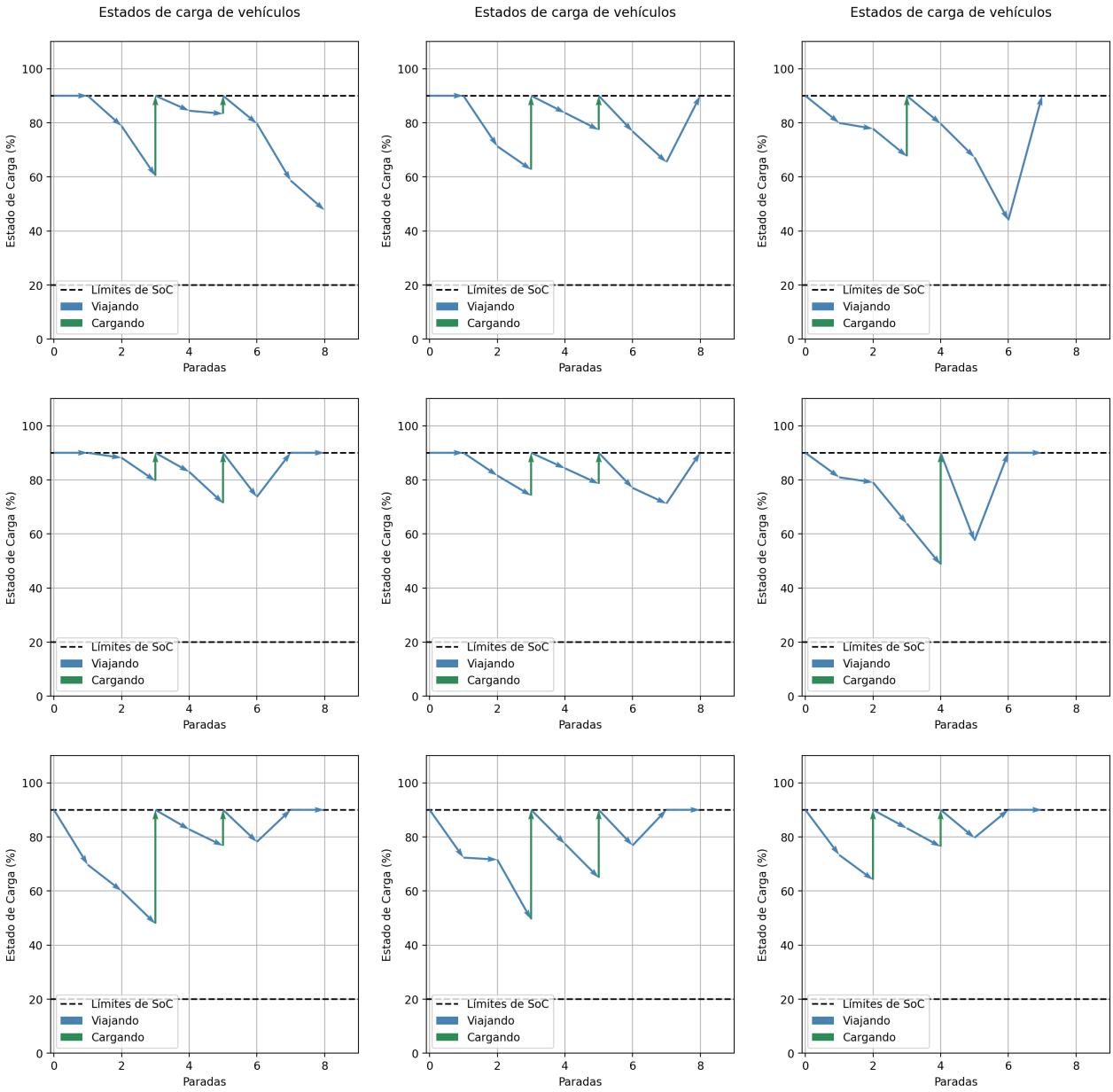
A partir de los gráficos de las Figuras 4.1, 4.2 y 4.3 se puede observar la convergencias tanto de *loss* y *reward* para los 3 modelos, es decir, los modelos lograron alcanzar (o acercarse bastante) a la solución óptima. Ahora bien, aunque se puede apreciar una leve pendiente positiva en los gráficos de *reward* (lo que indica que el modelo aún puede mejorar el *reward*), no se consideró continuar con el entrenamiento por lo insignificante de las mejoras logradas. El aumento del valor del *loss* entre los episodios 600 y 800 es de ≈ 0.5.

A continuación, se presentan los resultados del *reward* obtenidos para los 3 modelos. Las estadísticas fueron calculadas sobre un conjunto de prueba (el mismo para los 9 modelos) de 500 instancias.

Tabla 4.1: Estadísticas del reward para los 3 modelos de carga completa

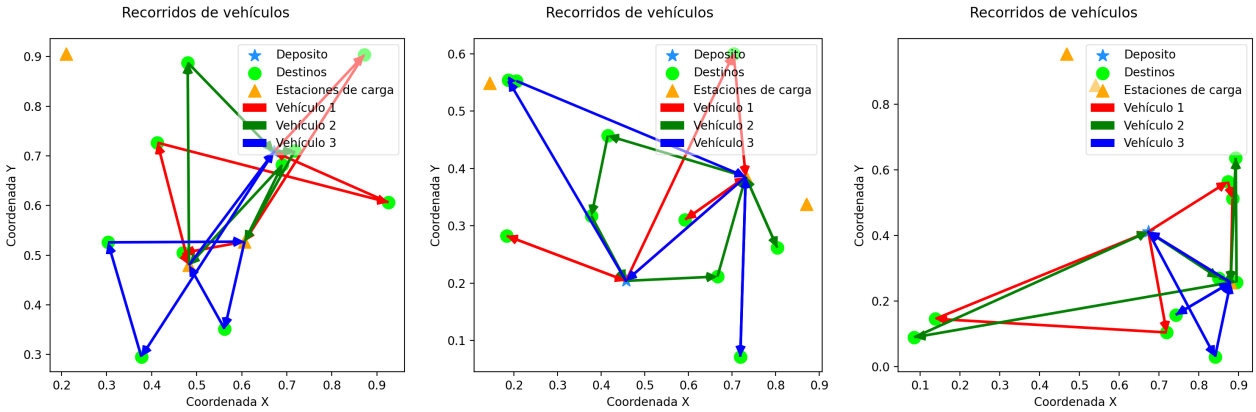
	Modelo 1	Modelo 2	Modelo 3
min	-1005.9664	-1012.2343	-1014.0142
max	-5.7731	-6.0944	-6.1096
avg	-12.2618	-12.2796	-26.0160

A partir de los datos de la Tabla 4.1, se puede observar que el *reward* más alto alcanzado fue logrado por el primer modelo, sin embargo, las diferencias con los otros modelos no es tan significativa. Los valores de *reward* máximo fueron logrados por los tres modelos en instancias diferentes del VRP y se puede explicar observando la Figura 4.4. En dicha figura, se observa que en los modelos 1 y 2 los vehículos realizaron dos pausas de reabastecimiento cada uno (al final del tour, cuando el vehículo vuelve al depósito, se carga la batería hasta SoC_{max}, sin embargo, esta carga está fuera del tour y por eso se representa por una flecha de color azul y no se considera en los costos), por otro lado, el modelo 3 realiza una pausa de reabastecimiento en dos vehículos, sin embargo, el tiempo del recorrido total es mucho mayor a los modelos 1 y 2, lo que explica la diferencia del *reward* con los otros modelos. Además, las restricciones de SoC_{min} y SoC_{max} son respetadas a lo largo del tour, por lo tanto, en esta instancia del VRP, el *reward* total corresponde principalmente al tiempo total de viaje. Ahora bien, si se observan las trayectorias de los 3 modelos para su instancia de mayor *reward* (Figura 4.5), se puede apreciar que los tres modelos, especialmente el modelo 1 y 3, aún pueden optimizar sus rutas para lograr una recompensa aún mayor.



(a) Modelo 1 (b) Modelo 2 (c) Modelo 3

Figura 4.4: Evolución del SoC para la instancia de mayor *reward* de los modelos de carga completa



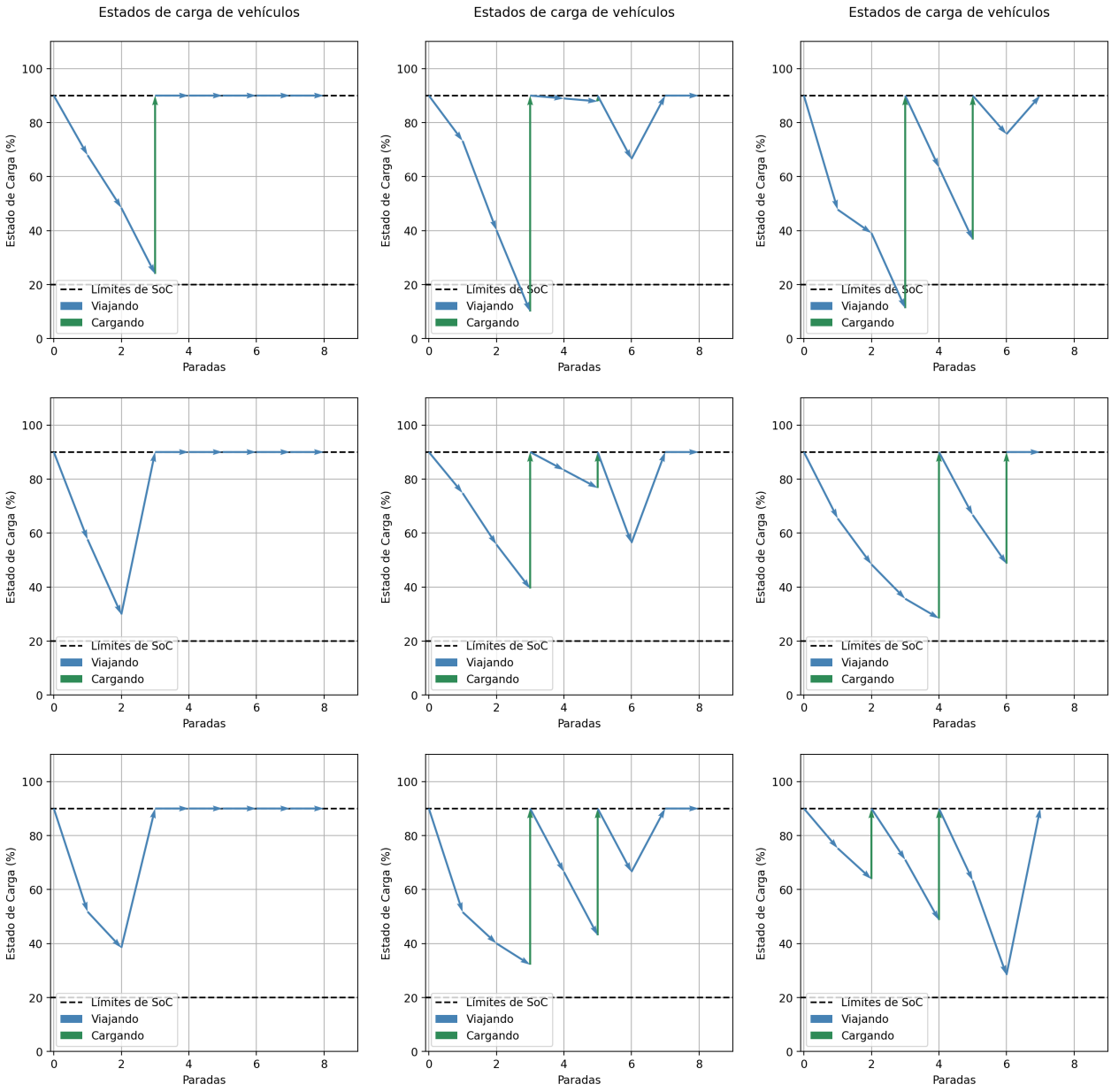
(a) Modelo 1

(b) Modelo 2

(c) Modelo 3

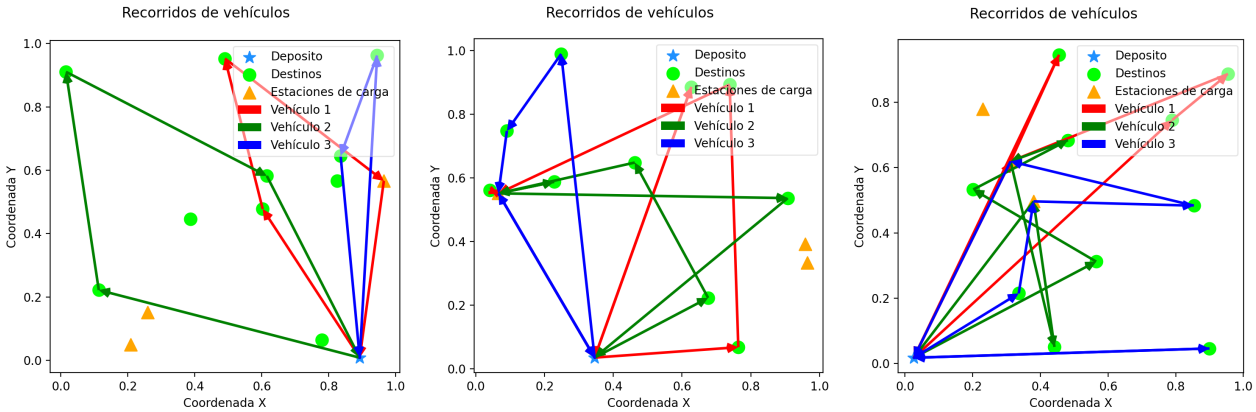
Figura 4.5: Trayectorias para la instancia de mayor *reward* de los modelos de carga completa

Por otra parte, el *reward* mínimo es muy similar en los 3 modelos y se obtienen en instancias diferentes. Como se puede observar en la Figura 4.6 y en la Tabla 4.1, los valores tan bajos en los *rewards* de los modelos se debe a que no lograron finalizar el tour, es decir, algún vehículo de la flota quedó con un SoC que no les permitía llegar a ningún destino. Si ahora analizamos estos resultados a partir de las trayectorias mostradas en la Figura 4.7, primero, se aprecia que en las 3 instancias los destinos están bastante distantes unos de otros, sin embargo, si se cambiara el orden en que se visitan algunos puntos, las rutas podrían ser mucho más corta y se podría haber evitado el quedar sin batería suficiente para viajar a otro destino. Especialmente en el modelo 1, se puede apreciar en la Figura 4.7, los vehículos disponen de dos estaciones de carga cercanas a algunos clientes, si hubiesen visitado alguna de aquellas estaciones, quizás podrían haber finalizado el tour correctamente.



(a) Modelo 1 (b) Modelo 2 (c) Modelo 3

Figura 4.6: Evolución del SoC para la instancia de menor *reward* de los modelos de carga completa



(a) Modelo 1

(b) Modelo 2

(c) Modelo 3

Figura 4.7: Trayectorias para la instancia de menor *reward* de los modelos de carga completa

Para finalizar con los resultados de la Tabla 4.1, se puede observar que el mayor promedio de la función de recompensa para las 500 instancias lo obtiene el modelo 1 con una diferencia notable con respecto a los otros 2 modelos. Esta diferencia se puede explicar con los datos de la Tabla 4.2, donde se observa que el modelo 1 no activa nunca el costo de J_2, por lo que el promedio de los *rewards* es menor.

A continuación, se presentan las estadísticas para cada una de las componentes de la función de recompensa.

Tabla 4.2: Estadísticas de cada componente del *reward* para los 3 modelos de carga completa

		Modelo 1	Modelo 2	Modelo 3
J1	min	3.3603	3.4376	2.8540
	max	8.6004	8.5955	8.0986
	avg	5.5913	5.7349	5.6311
J2	min	0	0	0
	max	1000	1000	1000
	avg	2	2	16
	Nº	1	1	8
J3	min	1.3209	2.5459	0.5623
	max	7.7917	7.4000	6.9486
	avg	4.6542	4.5430	4.3687
J4	min	0	0	0
	max	3.0586	0.4682	3.6167
	avg	0.0162	0.0018	0.0162

A partir de los datos de la Tabla 4.2, se observa que para J_1, los modelos presentan estadísticas bastante similares, lo que es razonable debido a que esta componente corresponde al tiempo total de viaje del tour, y es esperable que este tiempo sea similar para todos los modelos porque cualquier ruta muy diferente aumentaría el tiempo de viaje, disminuyendo el valor del *reward* total.

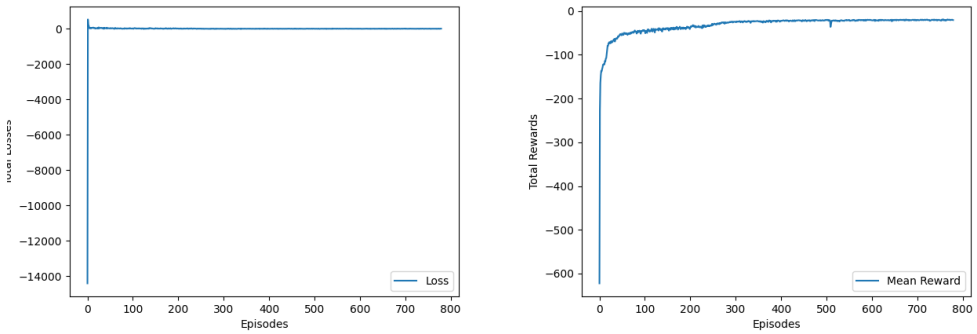
Con respecto a J_2, se observa que esta penalización se activa una vez en los modelos 1 y 2, y ocho veces en el modelo 3, lo que significa que los modelos 1 y 2 "aprendieron" casi completamente a evitar a el agotamiento total del SoC, mientras que el modelo 3 aún tiene inconvenientes en gestionar la carga de los vehículos. Si bien sería deseable que esta restricción no se active nunca, que se active en una de las 500 instancias, no es un mal resultado.

Continuando con los resultados de la Tabla 4.2, para J_3 se tiene que los tres modelos tienen un valor mínimo no nulo, esto quiere decir que en todas las instancias al menos un vehículo debió realizar carga de batería. Por otra parte, el modelo 1 tiene un valor máximo de ≈ 7.8, similar al modelo 2, pero muy distante del valor máximo del modelo 3 (≈ 7.0). Es razonable que los valores sean diferentes con respecto al tercer modelo, pues este último demostró tener inconvenientes para asignar los destinos e impedir que los vehículos queden sin carga.

Finalmente, con respecto a la penalización de violaciones al SoC (J_4), se observa que los 3 modelos tienen un valor promedio bastante bajo. Esto se puede explicar pues dentro de las 500 instancias es muy poco frecuente que se viole la restricción del SoC_{min}.

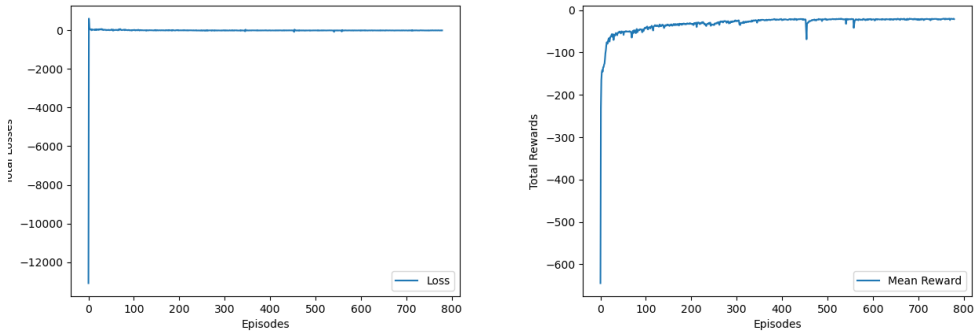
4.2.2. Modelos de carga discreta

Al igual que la sección anterior, comenzamos los resultados mostrando los gráficos de *reward* y función objetivo (*Loss*) obtenidos durante el entrenamiento de los modelos. El entrenamiento de cada uno de los modelos tardó aproximadamente 7 horas.



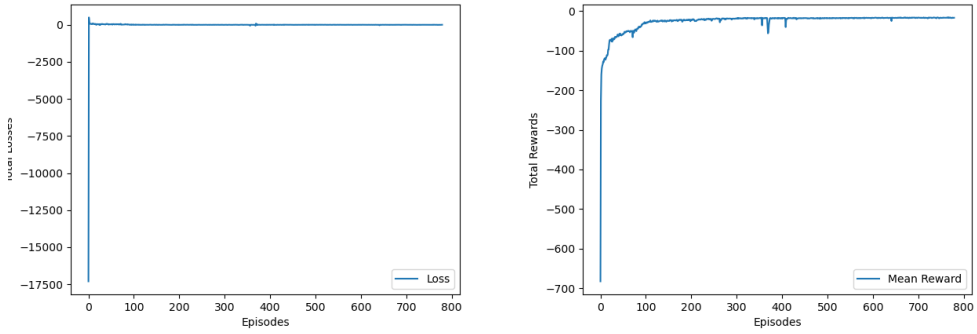
(a) Loss (b) Reward

Figura 4.8: Evolución *loss* y *reward* durante el entrenamiento del modelo 1 de carga discreta



(a) Loss (b) Reward

Figura 4.9: Evolución *loss* y *reward* durante el entrenamiento del modelo 2 de carga discreta



(a) Loss (b) Reward

Figura 4.10: Evolución *loss* y *reward* durante el entrenamiento del modelo 3 de carga discreta

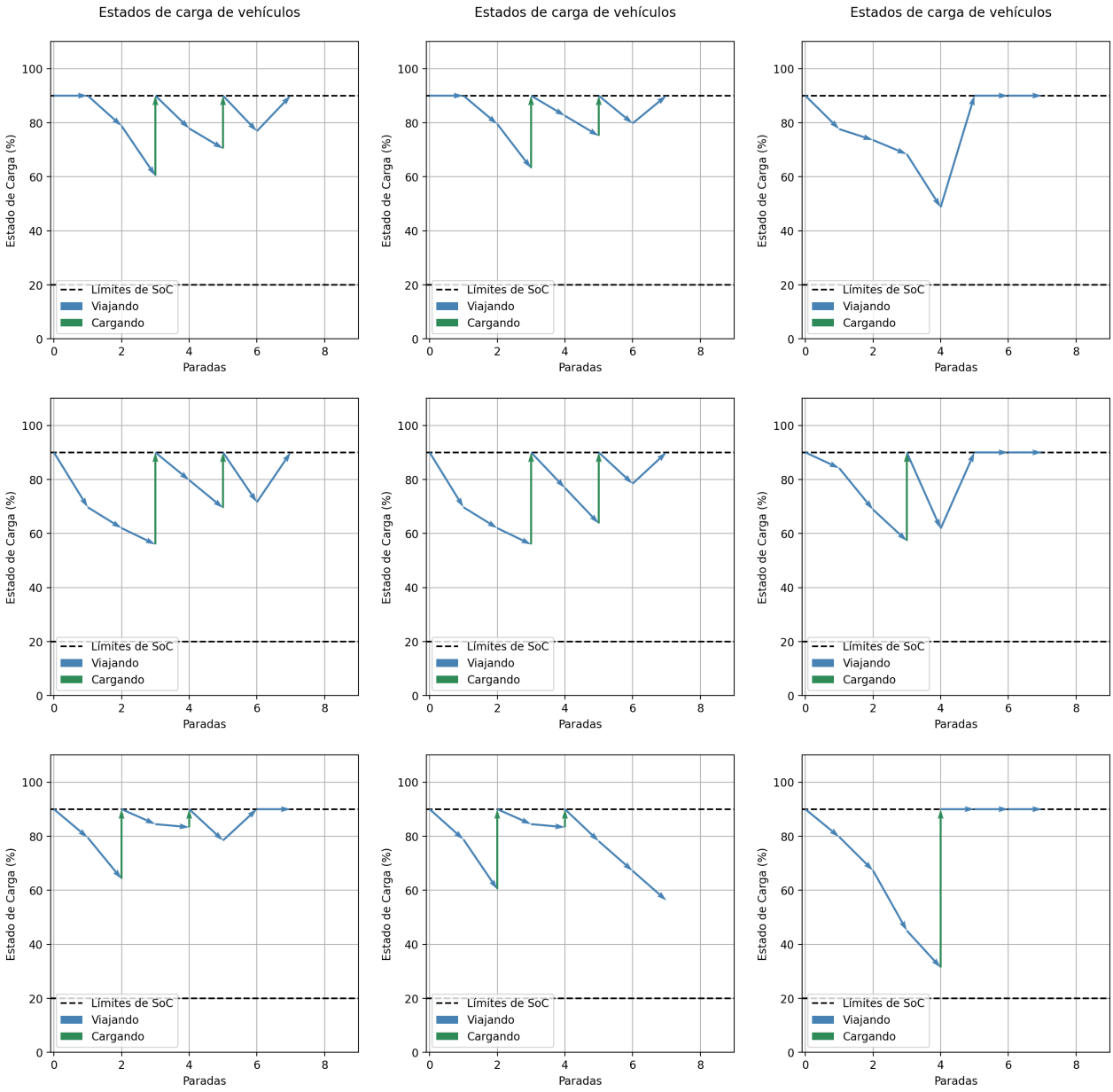
A partir de los gráficos de las Figuras 4.8, 4.9 y 4.10, se puede observar la convergencias tanto de *loss* y *reward* para los tres modelos. Y al igual que el caso de los modelos de carga completa, no se continua con el entrenamiento por la insignificancia de las mejoras. El aumento del valor del *loss* entre los episodios 600 y 800 es de ≈ 0.5

Tabla 4.3: Estadísticas del *reward* para los 3 modelos de carga discreta

	Modelo 1	Modelo 2	Modelo 3
min	-1017.8372	-1020.3420	-1015.0567
max	-12.1294	-12.4591	-6.9454
avg	-18.0693	-22.0158	-16.7996

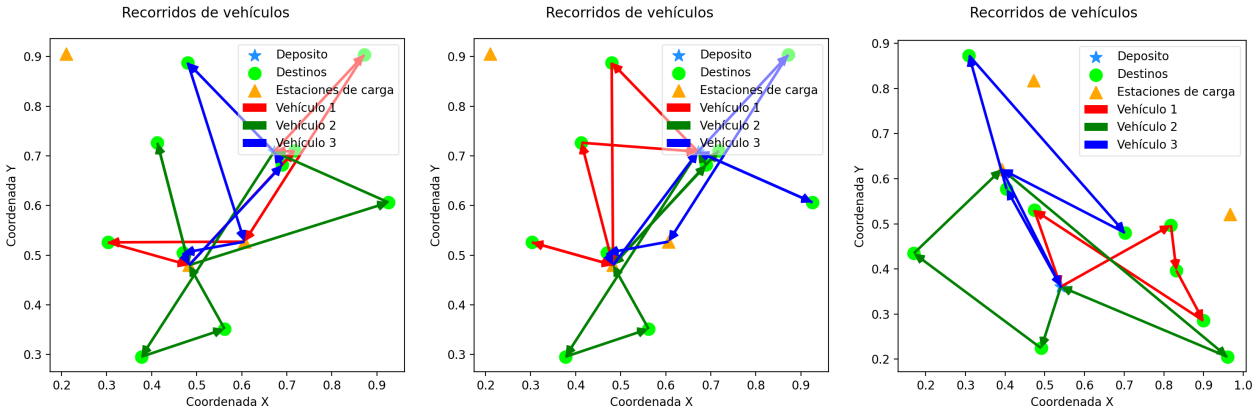
A partir de los datos de la Tabla 4.3, se puede observar que el *reward* más alto alcanzado fue logrado por modelo número 3 con una gran diferencia con respecto a los otros modelos. Los valores de *reward* máximo fueron logrados por los modelos 1 y 2 en la misma instancia,

mientras que el modelo 3 lo logró en una instancia diferente. Observando la Figura 4.11, se aprecia que el modelo 3 pudo visitar todos los destinos realizando solo una pausa de reabastecimiento por vehículo, por su parte, el modelo 1 y 2 realizan dos pausas de reabastecimiento en todos los vehículos, lo que explica la diferencia del *reward* entre estos modelos y el modelo 3. Además, las restricciones de SoC_{min} y SoC_{max} son respetadas a lo largo del tour. Ahora bien, si se observan las trayectorias de los tres modelos para su instancia de mayor *reward* (Figura 4.12), se puede apreciar a simple vista que los tres modelos tienen muchas oportunidades de aumentar sus *reward* con solo cambiar el orden en que se visitan algunos destinos.



(a) Modelo 1 (b) Modelo 2 (c) Modelo 3

Figura 4.11: Evolución del SoC para la instancia de mayor *reward* de los modelos de carga discreta



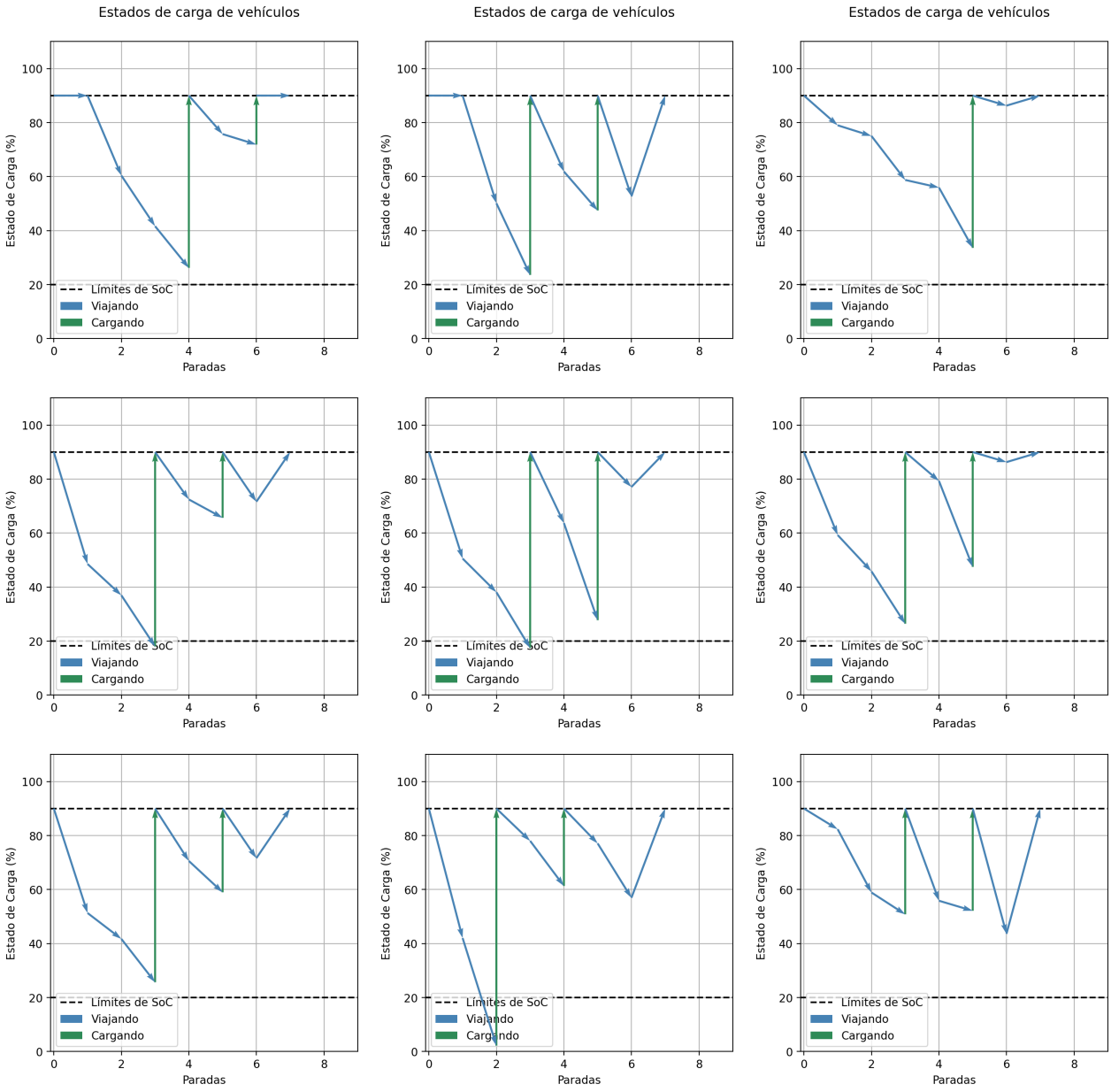
(a) Modelo 1

(b) Modelo 2

(c) Modelo 3

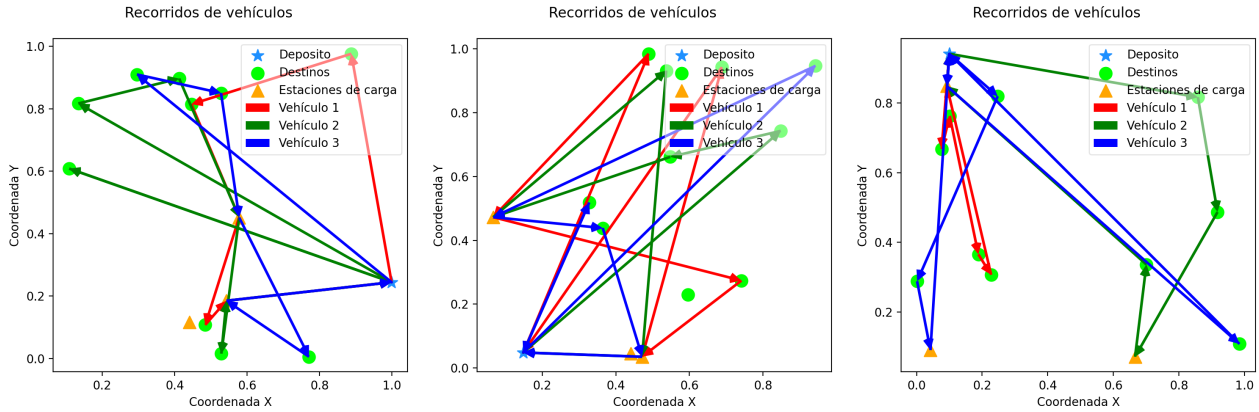
Figura 4.12: Trayectorias para la instancia de mayor *reward* de los modelos de carga discreta

Ahora, mirando los valores de *reward* mínimo obtenido por los modelos y los gráficos de la Figura 4.13, se observa que en los tres casos el valor mínimo corresponde a una instancia en la cual no se completó el tour debido a agotamiento de batería; los valores de *reward* mínimo fueron obtenidos en distintas instancias en los tres modelos. Si ahora analizamos estos resultados a partir de las trayectorias mostradas en la Figura 4.14, se aprecia que en las tres instancias los destinos están bastante distantes unos de otros, pero aún así, si se cambiara el orden de visita se podría haber finalizado el tour en los tres casos.



(a) Modelo 1 (b) Modelo 2 (c) Modelo 3

Figura 4.13: Evolución del SoC para la instancia de menor *reward* de los modelos de carga discreta



(a) Modelo 1

(b) Modelo 2

(c) Modelo 3

Figura 4.14: Trayectorias para la instancia de menor *reward* de los modelos de carga discreta

Si ahora nos fijamos en los datos de *reward* promedio de la Tabla 4.3, se puede observar que el mayor promedio de la función de recompensa para las 500 instancias lo obtiene el modelo 3, seguido por el modelo 2 con un valor similar y, por último, con un valor promedio mucho peor, el modelo 1. Al igual que en los modelos de carga completa, estos resultados se pueden explicar con la cantidad de veces que se activa el costo de J_2 lo que impacta directamente en el promedio.

A continuación, se presentan las estadísticas para cada una de las componentes de la función de recompensa.

Tabla 4.4: Estadísticas de cada componente del reward para los 3 modelos de carga discreta

		Modelo 1	Modelo 2	Modelo 3
J1	min	9.4059	9.6935	5.1195
	max	13.0002	13.1372	13.0237
	avg	11.2124	11.2514	8.8328
J2	min	0	0	0
	max	1000	1000	1000
	avg	2	6	4
	Nº	1	3	2
J3	min	2.3846	2.7656	1.7791
	max	7.2029	7.2048	7.5107
	avg	5.8511	5.7533	5.2437
J4	min	0	0	0
	max	1.6815	4.9795	0.5943
	avg	0.0059	0.0110	0.0031

A partir de los datos de la Tabla 4.4, se observa que para J_1, los modelos presentan estadísticas bastante similares, es decir, los 3 modelos tienen un promedio de tiempos de viajes parecidos en las 500 instancias de prueba.

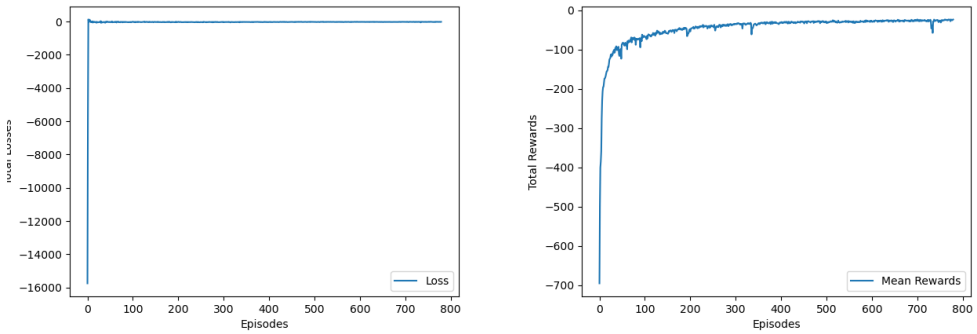
Con respecto a J_2, se observa que esta penalización se activa una vez en el modelo 1, tres veces en el modelo 2 y dos veces en el modelo 3. Estos resultados no son muy buenos, pues es de esperar que, luego del entrenamiento, los modelos aprendan a evitar quedar varados en mitad del tour.

Continuando con los resultados de la Tabla 4.4, para J_3 se tiene que los tres modelos tienen un valor mínimo distinto de cero, esto quiere decir que en todas las instancias hubo detenciones en estaciones de carga. Por otra parte, los tres modelos tienen un valor máximo similar cercano a 7. Al igual que se mencionó en la sección anterior, es esperable que los valores de J_3 sean similares para los tres modelos pues comparten la misma política para las pausas de reabastecimiento.

Para finalizar con los datos de la Tabla 4.4, observando las estadísticas de J_4, se observa que los tres modelos tienen valor promedio bastante bajo. Esto se puede explicar nuevamente por la baja frecuencia con que se viola la restricción del SoC_{min}, sin embargo, cuando se viola la restricción se llegan a valores de penalización bastante significativos, como lo sería en el caso del modelo 2.

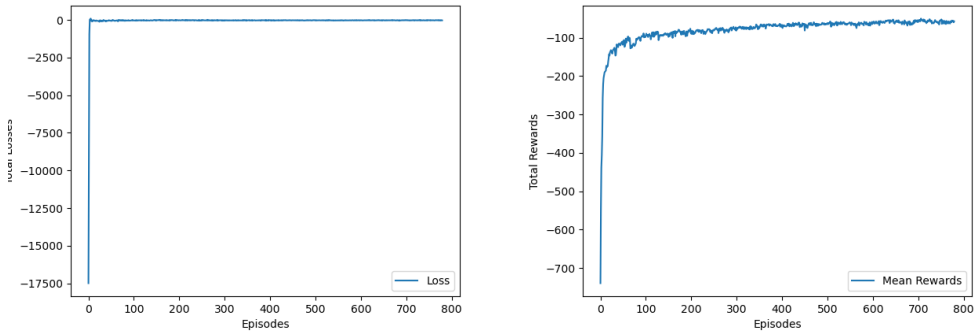
4.2.3. Modelos de carga continua

Al igual como se realizó en las secciones anteriores, comenzamos los resultados mostrando los gráficos de *reward* y *loss* obtenidos durante el entrenamiento de los modelos. El entrenamiento de cada uno de los modelos de carga continua tardó aproximadamente 7 horas.



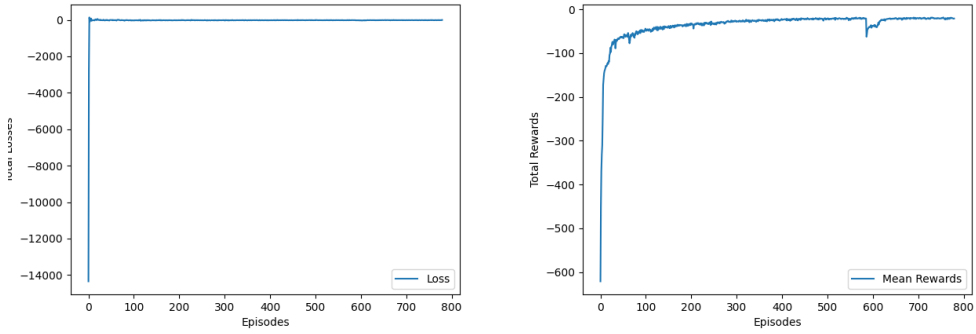
(a) Loss (b) Reward

Figura 4.15: Evolución *loss* y *reward* durante el entrenamiento del modelo 1 de carga continua



(a) Loss (b) Reward

Figura 4.16: Evolución *loss* y *reward* durante el entrenamiento del modelo 2 de carga continua



(a) Loss (b) Reward

Figura 4.17: Evolución *loss* y *reward* durante el entrenamiento del modelo 3 de carga continua

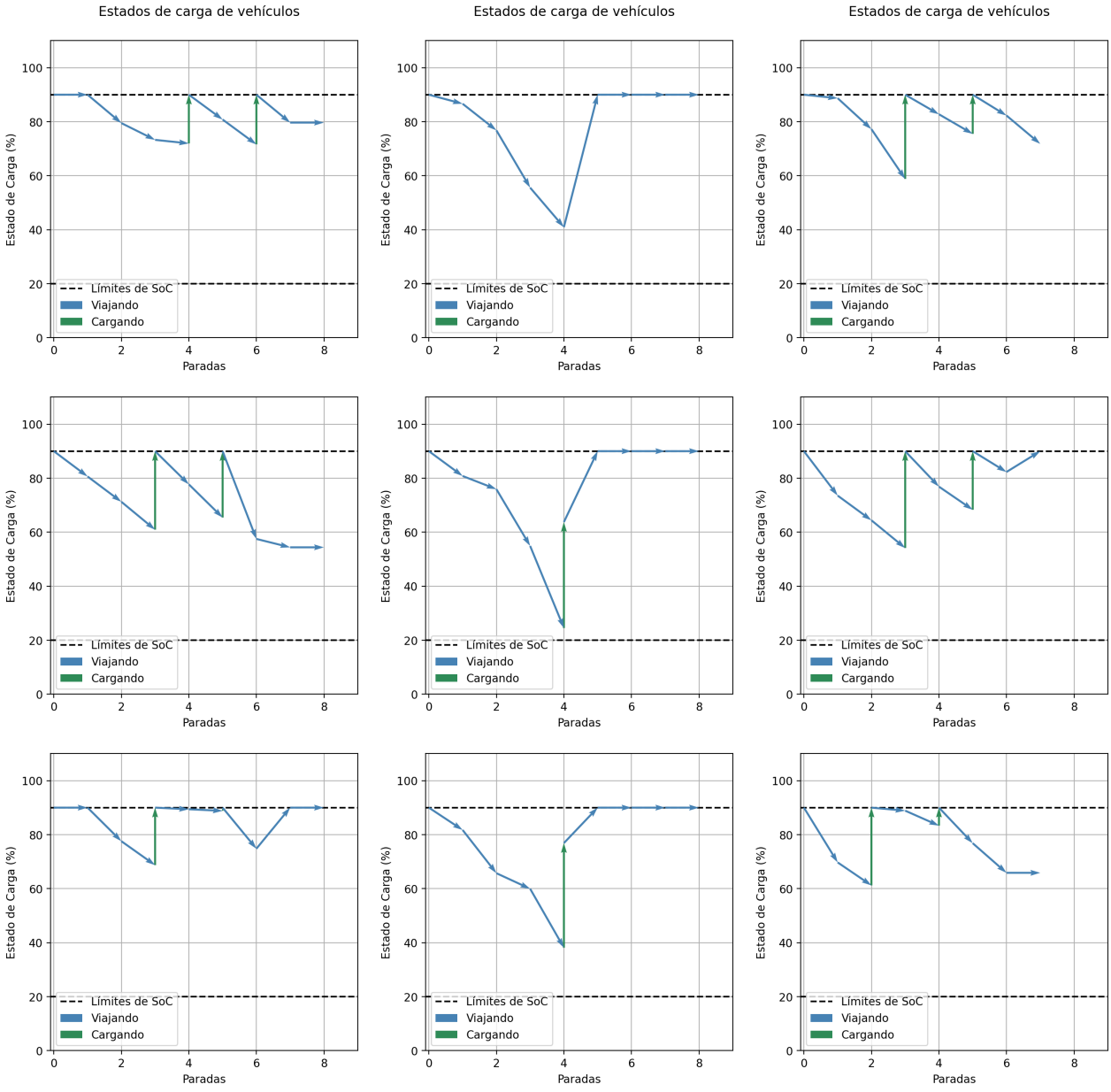
A partir de los gráficos de las figuras 4.15, 4.16 y 4.17 se puede observar la convergencias tanto de *loss* y *reward* para los tres modelos. Nuevamente, tal como se mencionó en el entrenamiento de los modelos de carga completa y discreta, no se continua el entrenamiento debido a la insignificancia de las mejoras (aumento del valor del *loss* de ≈ 0.5 entre los episodios 600 y 800).

Tabla 4.5: Estadísticas del *reward* para los 3 modelos de carga continua

	Modelo 1	Modelo 2	Modelo 3
min	-1012.9801	-1016.8858	1015.3155
max	-5.7538	-5.5073	-6.0946
avg	-29.7389	-59.4235	-20.2194

A partir de los datos de la Tabla 4.5, se puede observar que el *reward* máximo fue alcanzado por el segundo modelo, sin embargo, las diferencias con los otros modelos no son

mayores. Los valores de *reward* máximo fueron logrados por los tres modelos en instancias diferentes. Ahora, observando los gráficos de las Figuras 4.18 y 4.19, se aprecia que en el modelo 2 los vehículos logran recorrer los destinos teniendo que visitar como máximo una vez las estaciones de carga. Por otra parte, los modelos 1 y 3 logran completar el tour pero los vehículos realizan dos visitas a las estaciones de carga. Además, se puede apreciar que los tres modelos finalizan el tour sin violar las restricciones de SoC_{min} y SoC_{max}.

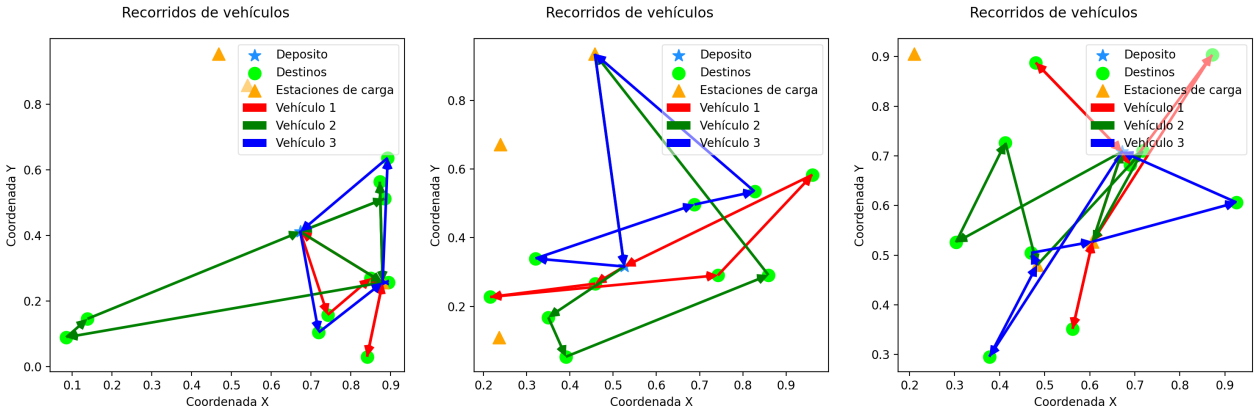


(a) Modelo 1

(b) Modelo 2

(c) Modelo 3

Figura 4.18: Evolución del SoC para la instancia de mayor *reward* de los modelos de carga continua



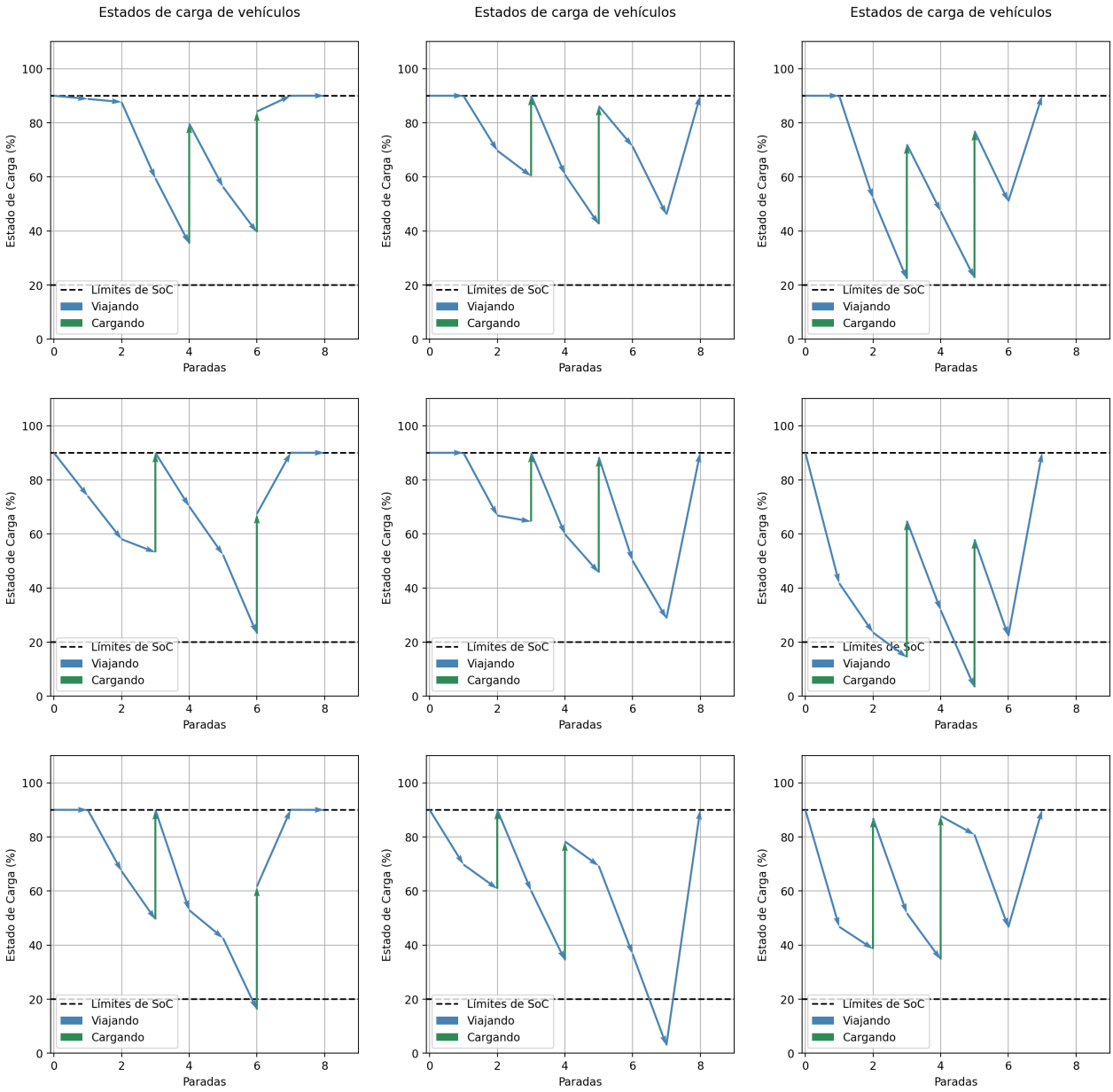
(a) Modelo 1

(b) Modelo 2

(c) Modelo 3

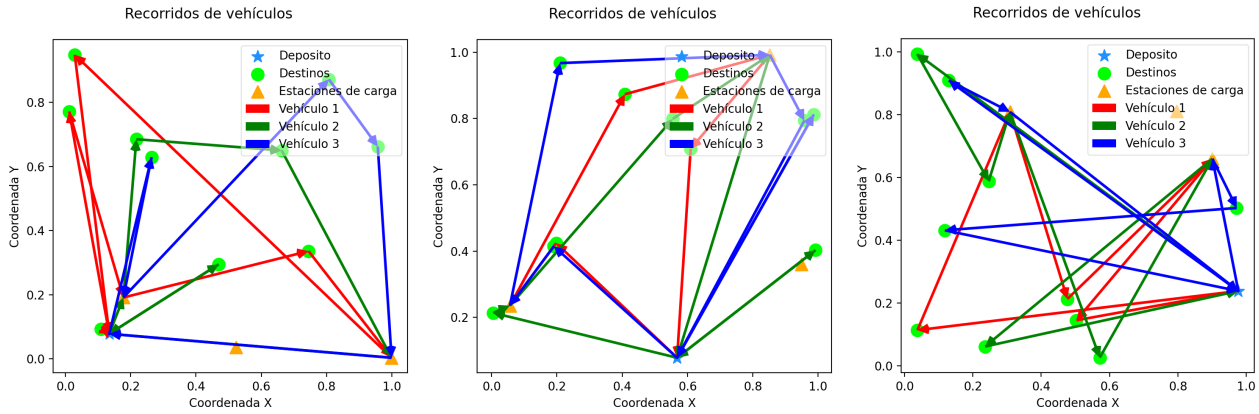
Figura 4.19: Trayectorias para la instancia de mayor *reward* de los modelos de carga continua

Por otra parte, el *reward* mínimo es muy similar en los tres modelos y fueron obtenidos en instancias diferentes. Como se puede observar en la Figura 4.20 y en la Tabla 4.5, los valores tan bajos en los *rewards* se debe a que la flota de los 3 modelos no logró finalizar el tour debido al SoC. Si ahora analizamos estos resultados a partir de las trayectorias mostradas en la Figura 4.7, se aprecia nuevamente que en las tres instancias, los destinos están muy distanciados unos de otros y las rutas realizadas por los vehículos no son las óptimas.



(a) Modelo 1 (b) Modelo 2 (c) Modelo 3

Figura 4.20: Evolución del *SoC* para la instancia de menor *reward* de los modelos de carga continua



(a) Modelo 1 (b) Modelo 2 (c) Modelo 3

Figura 4.21: Trayectorias para la instancia de menor *reward* de los modelos de carga continua

Fijándonos ahora en los datos de *reward* promedio de la Tabla 4.5, se puede observar que el mayor promedio de la función de recompensa para las 500 instancias lo obtiene el modelo 2 , seguido por el modelo 1 con un valor similar, y por último el modelo 3 con un valor un poco mas distante.

A continuación, se presentan las estadísticas para cada una de las componentes de la función de recompensa.

Tabla 4.6: Estadísticas de cada componente del reward para los 3 modelos de carga continua

		Modelo 1	Modelo 2	Modelo 3
J1	min	3.2685	3.2165	3.3207
	max	7.8601	7.7009	9.1006
	avg	5.7518	5.4910	5.6115
J2	min	0	0	0
	max	1000	1000	1000
	avg	20	50	10
	Nº	10	25	5
J3	min	1.9819	1.5228	2.0146
	max	5.4265	5.1287	6.2149
	avg	3.9395	3.9033	4.5932
J4	min	0	0	0
	max	3.6995	4.8924	3.6485
	avg	0.0475	0.0292	0.0147

A partir de los datos de la Tabla 4.6, se observa que para J_1, los modelos tienen estadísticas

similares. Esto significa que los tiempos de viaje son parecidos para los tres modelos, lo cual es un resultado esperable.

Con respecto a J_2, se observa que esta penalización se activa 10 veces por el modelo 1, 25 veces por el modelo 2 y 5 veces por el modelo 3. Estos resultados son bastante malos en comparación a los modelos de carga completa y carga discreta que se revisaron anteriormente, donde la frecuencia de activación de esta penalización no era tan alta.

Continuando con los resultados de la Tabla 4.6, para J_3 se tiene que los tres modelos tienen un valor mínimo no nulo debido a que siempre se cargó batería en al menos un vehículo. Por otra parte, los modelos 1 y 2 tienen un valor máximo de ≈ 5, diferente al valor máximo del modelo 3 (≈ 6). Ahora, observando los promedios, se observa que estos son bastante menores con respecto a lo que ocurría en los modelos de carga completa y discreta. Esto se puede explicar pues los modelos de carga continua tienen completa libertad para escoger los montos de carga, mientras que los otros modelos son más restringidos en ese aspecto.

Con respecto a la penalización de violaciones al SoC (J_4), se observa que los tres modelos presentan promedios bastante similares y bastante pequeños, sin embargo, los valores máximos son mucho mayores a los modelos de carga completa y discreta. Esta situación se puede deber a que, si bien el modelo de carga continua puede elegir exactamente el monto a cargar, la red que decide este monto no "aprendió" a optimizar las cargas violando reiteradamente la limitación de SoC_{min}.

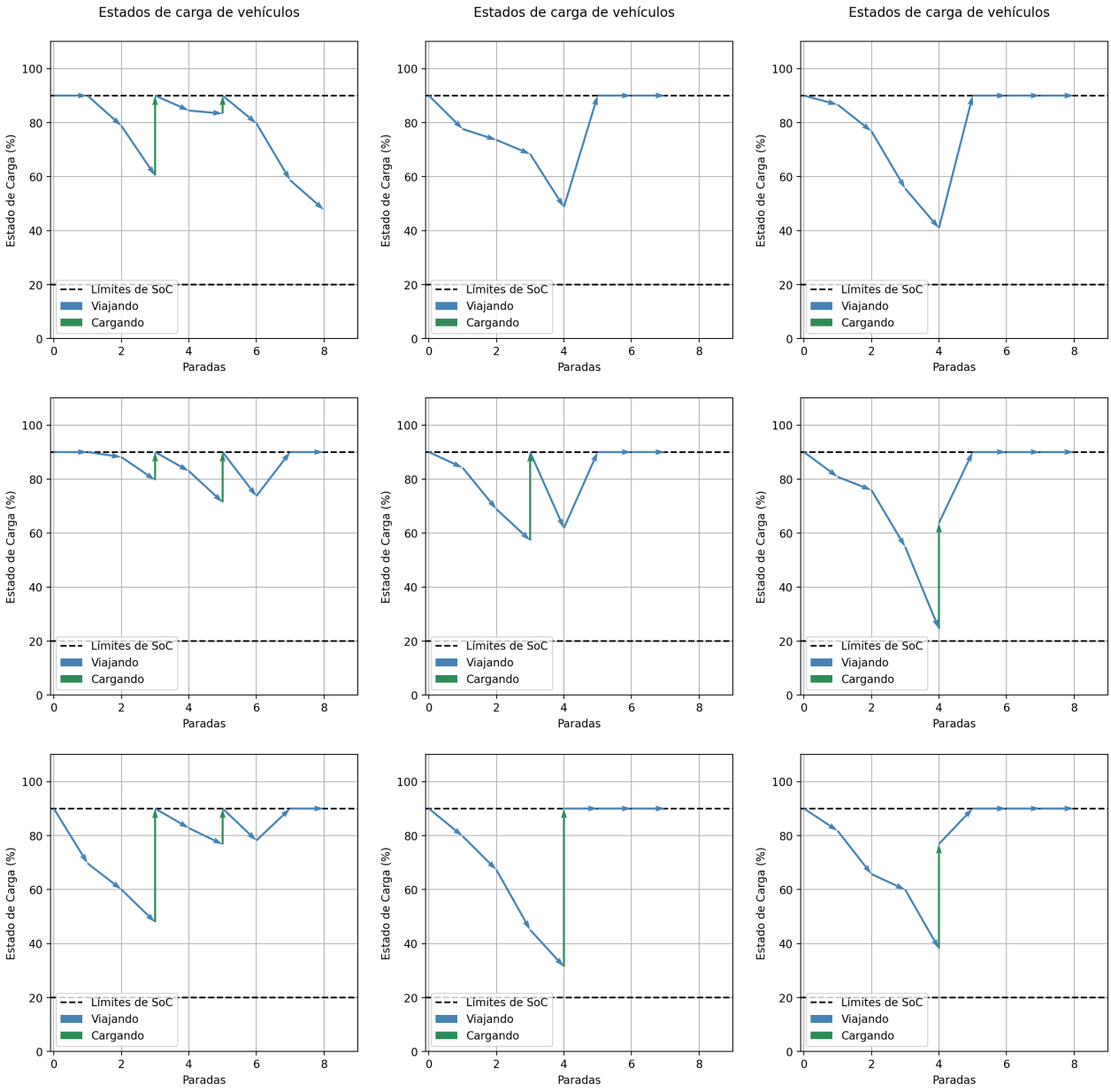
4.2.4. Comparación de los mejores modelos

En esta sección se realiza una comparación entre el mejor modelo de cada solución, para la solución de carga completa se utiliza el modelo 1, para la carga discreta se utiliza el modelo 3 y para la solución de carga continua se utiliza el modelo 2.

Tabla 4.7: Estadísticas del *reward* para los mejores modelos de cada solución

	Carga completa	Carga discreta	Carga continua
min	-1005.9664	-1015.0567	-1016.8858
max	-5.7731	-6.9454	-5.5073
avg	-12.2618	-16.7996	-59.4235

A partir de los datos de la Tabla 4.7, se observa que el máximo *reward* es obtenido por el modelo de carga continua, seguido de cerca por el modelo de carga completa y, por último, un poco más alejado, el modelo de carga discreta. El valor de *reward* máximo de los modelos fueron obtenidos en instancias diferentes. Estos resultados se pueden explicar observando las Figuras 4.22 y 4.23, en las cuales se observa que, tanto el modelo de carga discreta como el de carga continua, logran completar el tour realizando máximo una detención de carga por vehículo, mientras que el modelo de carga completa realiza 2 pausas de reabastecimiento en cada uno de los 3 vehículos de la flota. Si bien el modelo de carga completa realizó más visitas a estaciones de carga, los montos cargados por los vehículos fueron muy pequeños, lo que reduce los tiempos de carga y por ende el tiempo total de viaje del tour.

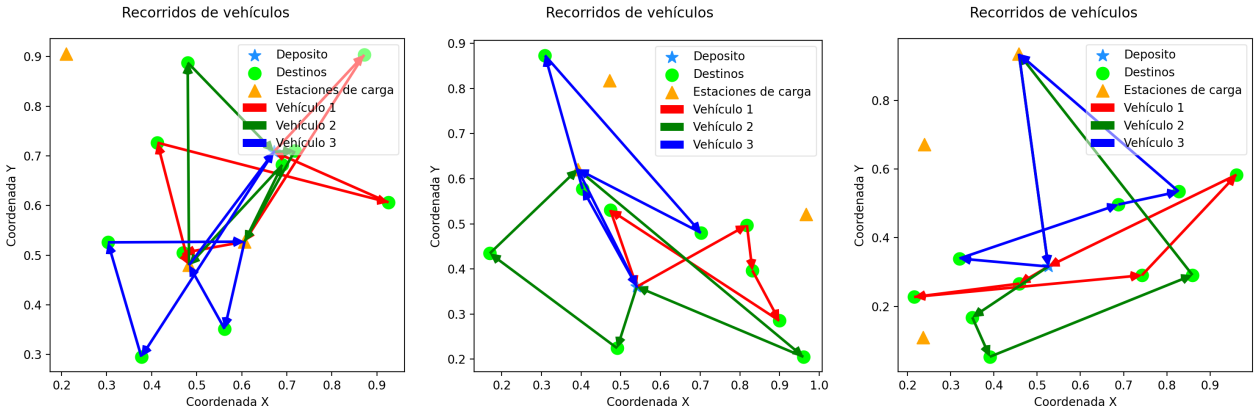


(a) Modelo carga completa

(b) Modelo carga discreta

(c) Modelo carga continua

Figura 4.22: Evolución del SoC para la instancia de mayor *reward* de los mejores modelos



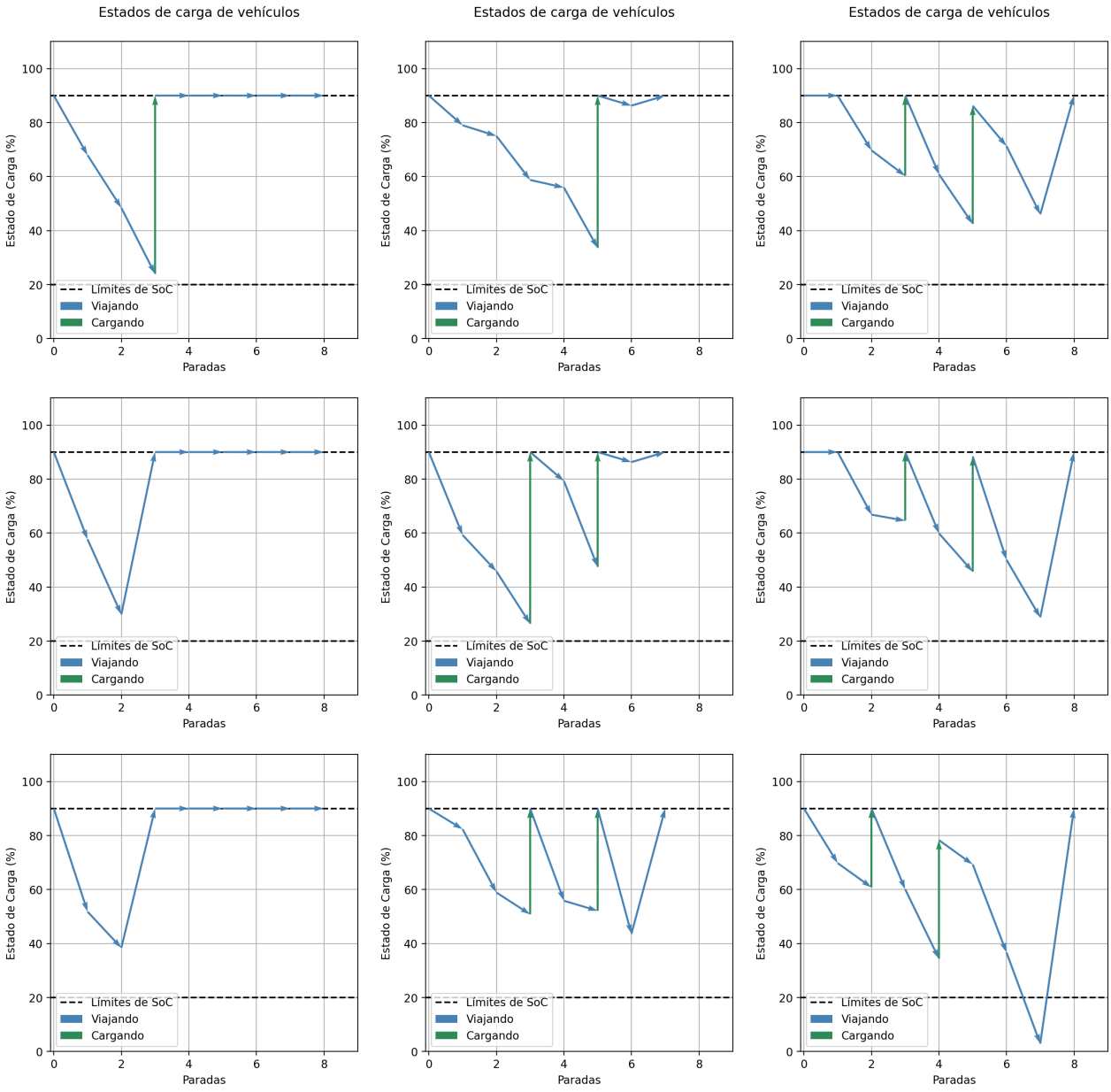
(a) Modelo carga completa

(b) Modelo carga discreta

(c) Modelo carga continua

Figura 4.23: Trayectorias para la instancia de mayor *reward* de los mejores modelos

Con respecto a los valores de *reward* mínimo (ver Figuras 4.24 y 4.25), se observa que los tres modelos no lograron finalizar el tour debido a que algún vehículo quedó sin batería. Nuevamente, estos resultados fueron obtenidos en instancias diferentes para cada modelo.

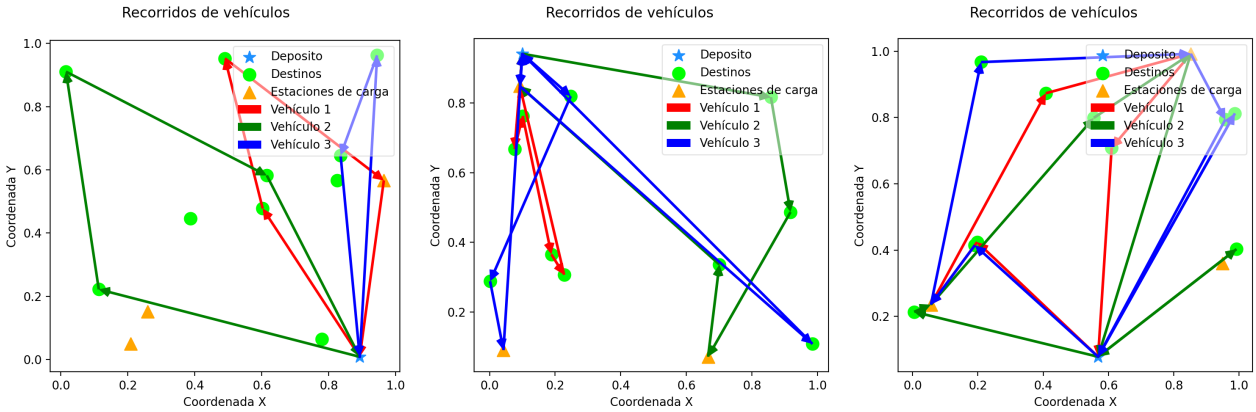


(a) Modelo carga completa

(b) Modelo carga discreta

(c) Modelo carga continua

Figura 4.24: Evolución del SoC para la instancia de menor *reward* de los mejores modelos



(a) Modelo carga completa

(b) Modelo carga discreta

(c) Modelo carga continua

Figura 4.25: Trayectorias para la instancia de menor *reward* de los mejores modelos

Para finalizar con los resultados de la Tabla 4.7, se puede observar que el mayor promedio del *reward* es obtenido por el modelo de carga completa, seguido por el modelo de carga discreta y por último el modelo de carga continua. Esta estadística indica como es el desempeño general de los modelos para resolver la instancia del VRP abordado. Si bien es esperable que el modelo de carga continua sea el que tenga mejor desempeño al tener la libertad de elegir exactamente el monto de carga de batería en las estaciones de carga, estos resultados indican que este modelo aún tiene muchos problemas optimizando los montos de carga, sobrepasando muchas veces la restricción del SoC_{min} y quedando sin batería para finalizar el tour.

A continuación, se presentan las estadísticas para cada una de las componentes de la función de recompensa.

Tabla 4.8: Estadísticas de cada componente del *reward* para los mejores modelos de cada solución

		Carga completa	Carga discreta	Carga continua
J1	min	3.3603	5.1195	3.3207
	max	8.6004	13.0237	9.1006
	avg	5.5913	8.8328	5.6115
J2	min	0	0	0
	max	1000	1000	1000
	avg	2	4	10
	Nº	1	2	5
J3	min	1.3209	1.7791	2.0146
	max	7.7917	7.5107	6.2149
	avg	4.6542	5.2437	4.5932
J4	min	0	0	0
	max	3.0586	0.5943	3.6485
	avg	0.0162	0.0031	0.0147

Al observar los datos de la Tabla 4.8, se puede apreciar que los mejores valores de J_1 son obtenidos por el modelo de carga completa seguido, con valores similares, por el modelo de carga continua. En último lugar está el modelo de carga discreta con estadísticas bastante peores a los otros dos modelos. Esto se debe, a lo poco eficiente que son los recorridos realizados por el modelo discreto lo que afecta enormemente los tiempos totales de viaje.

Continuando con las estadísticas de J_2, se observa que ninguno de los tres modelos logra evitar completamente esta penalización, es decir, aún tienen problemas para aprender a escoger rutas y distribuir las pausas de reabastecimiento de forma de lograr finalizar sus rutas satisfactoriamente. Esta situación quizás pueda deberse al alto consumo de los vehículos (40 % por unidad recorrida).

Para continuar con los resultados de la Tabla 4.8, las estadísticas de J_3 son bastante similares para los tres modelos pero, aún así, el modelo de carga continua logra tener un mejor desempeño. Este resultado puede ser explicado por la flexibilidad de este modelo de escoger los montos de carga, lo que le permite optimizar los tiempos de las pausas de reabastecimiento. Bajo esta premisa, era de esperar que el modelo discreto presente mejores estadísticas que el modelo de carga completa, sin embargo, esto no sucede. Esto se puede deber a que como la solución de carga discreta tiene más nodos que las otras dos soluciones (por la incorporación de estaciones de carga ficticias), su entrenamiento se hace más complejo y es más difícil que aprenda a evitar estas penalizaciones.

Finalizando con los resultados mostrados en la Tabla 4.8, se puede observar que para J_4 los tres modelos logran valores promedios bastante diminutos en promedio, logrando mantener su estado de carga casi siempre en los márgenes establecidos.

A modo de cierre, se puede concluir que en términos de desempeño general, el modelo

de carga continua logra mejores resultados, seguido por el modelo de carga completa y, por último, el modelo discreto. La principal razón de este resultado, es la libertad de elección de los montos de carga en las pausas de reabastecimiento y la elección de rutas que le permiten finalizar el tour en menores tiempos. Ahora bien, sería interesante aumentar la cantidad de opciones de carga para el modelo discreto (en vez de 4 opciones, que sean 10 o más) y realizar nuevamente una comparación entre los 3 modelos para analizar si se logra mejorar el desempeño general de este modelo. Otro estudio que sería interesante, es la comparación de los resultados de estas tres soluciones, basadas en redes neuronales, con los resultados que se obtienen utilizando un método de resolución exacta como el de [2]. Otra posible modificación podría ser la función de recompensa para probar alternativas de penalizaciones y, quizás así, mejorar el desempeño general de las redes. Por otra parte, sería interesante estudiar nuevas estructuras para la red que define el monto de carga en el modelo continuo pues, como se observó en los resultados, este modelo aún no saca provecho al máximo de la flexibilidad de elección de los montos de carga. Finalmente, relacionado al entrenamiento de las redes, se propone una modificación para aumentar la exploración de soluciones, al menos en las primeras etapas, pues se observa que los modelos llegan cerca del óptimo muy rápido y luego tienen mejoras a un ritmo muy lento.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

En el presente trabajo se realizó una revisión bibliográfica del estado del arte de algunas soluciones propuestas al VRP, principalmente, aquellas basadas en implementaciones de redes neuronales y entrenadas con aprendizaje reforzado. A raíz de esto, se decide la utilización de *Pointer Network* como arquitectura base, la cual fue modificada para dar forma a las tres soluciones desarrolladas en este trabajo. Además, se decide utilizar aprendizaje reforzado para el entrenamiento de las redes neuronales pues, este método, nos permite converger a la solución óptima aún cuando no se conoce a priori dicha solución. Más específicamente, se hace uso del algoritmo REINFORCE (1)

Luego, se realizó una caracterización de la instancia del VRP abordado en el trabajo, definiendo las restricciones operacionales de los vehículos de la flota, los elementos considerados para la medición del desempeño de los modelos y las simplificaciones utilizadas. Además, se caracteriza la función de recompensa y la función objetivo diseñadas.

Los resultados obtenidos en este trabajo indican que las tres soluciones desarrolladas logran resolver satisfactoriamente el problema abordado, aunque con algunas diferencias en sus desempeños. Se concluyó que todos los modelos tienen problemas en algunas instancias para lograr finalizar el tour, quedando varados en mitad del recorrido sin tener batería suficiente para llegar a ningún destino. Además, en términos de desempeño general, se concluyó que el modelo de carga continua tenia mejor rendimiento, seguido por el modelo de carga completa y por último el de carga discreta. Este resultado fue explicado a partir de la libertad del modelo continuo de elegir los montos en las estaciones de carga y la elección de rutas óptimas que le permiten finalizar el tour en menor tiempo. Además, se concluye que el mal desempeño del modelo de carga discreta se puede deber a la complejidad extra del entrenamiento al incluir nodos ficticios.

Finalmente, se realizó un análisis de cada uno de los elementos de la función de recompensa, donde se concluyó la similitud de los tiempos de viaje por los tres modelos. Además, se evidenció la dificultad de "aprendizaje" de los modelos para evitar la penalización por quedar con $SoC = 0$. Con respecto a la penalización de los tiempos de carga, se discutió que el mejor desempeño lo obtuvo el modelo de carga continua, seguido por el modelo de carga completa y por último el modelo discreto. Nuevamente, se atribuyó este resultado a la flexibilidad de

la elección de los montos de carga por parte del modelo continuo y a la inclusión de nodos ficticios por parte del modelo discreto, que provocan que el entrenamiento sea más difícil. Por último, con respecto a las penalizaciones por violaciones a los límites de SoC_{min}, se concluye que las tres soluciones presentan bajos valores promedios logrando mantener la carga de los vehículos dentro de los límites la mayor cantidad del tiempo.

5.2. Trabajo Futuro

Finalizado este trabajo se pueden dar puntos para seguir explorando y mejorando. Primero, como se mencionó al final de la sección 3.3.3, se propone el considerar la utilización de un modelo de red de transporte realista que no considere rutas directas entre todos los nodos. Una idea para implementar esto, es la incorporación, en el *dynamic*, de la información del tiempo que demorarían los vehículos en llegar a cada destino. O tal vez, no incluir directamente esa información y esperar que la misma red se dé cuenta mediante los valores en el *reward*. El mayor desafío que plantea esta idea es que los tiempos de entrenamiento aumenten considerablemente debido a la gran cantidad de información extra que esto requiere.

Otra idea de trabajo futuro, es la comparación de los resultados obtenidos por los modelos desarrollados en este trabajo, con los resultados que se obtienen al utilizar un método de resolución exacta como el de [2].

También, como se mencionó en la sección 4.2.4, se propone aumentar la cantidad de opciones de carga del modelo discreto para analizar si se logra mejorar su desempeño general. Además, se propone la modificación de la función de recompensa para probar distintas alternativas de penalizaciones que puedan mejorar el desempeño general de las redes. Finalmente, se propone modificar el entrenamiento para aumentar la exploración en sus primeras etapas, y así evitar que los modelos se acerquen demasiado rápido al óptimo en un comienzo y mejoren muy lentamente en las etapas posteriores.

Bibliografía

[1] Rodrigue, J.-P., "The geography of transport systems," vol. Fifth edition, 2020, doi:https://doi.org/10.4324/9780429346323.

[2] Futalef Gallardo, J. P., "A decision-making system for managing electric vehicle fleets subject to multiple operational constraints," 2021.

[3] de las Naciones Unidas sobre el Cambio Climático, C. M., "¿qué es la triple crisis planetaria?," 2022, https://unfccc.int/es/blog/que-es-la-triple-crisis-planetaria.

[4] Oriol Vinyals, Meire Fortunato, N. J., "Pointer networks," 2017, doi:https://doi.org/10.48550/arXiv.1506.03134.

[5] Toth, P. y Vigo, D., "Vehicle routing: problems, methods, and applications," Society for Industrial and Applied Mathematics, vol. 2nd edition, 2014, doi:https://doi.org/10.1137/1.9781611973594.

[6] Stackoverflow, "Lstm cell diagram," 2018.

[7] Mohammadreza Nazari, Afshin Oroojlooy, L. V. S. M. T., "Deep reinforcement learning for solving the vehicle routing problem," 2018, doi:https://doi.org/10.48550/arXiv.1802.04240.

[8] Yu, J. J. Q., Yu, W., y Gu, J., "Online vehicle routing with neural combinatorial optimization and deep reinforcement learning," IEEE Transactions on Intelligent Transportation Systems, vol. 20, 2019, doi:10.1109/TITS.2019.2909109.

[9] Galatzer-Levy, I., Ruggles, K., y Chen, Z., "Data science in the research domain criteria era: Relevance of machine learning to the study of stress pathology, recovery, and resilience," Chronic Stress, vol. 2, p. 247054701774755, 2018, doi:10.1177/2470547017747553.

[10] S. Shao, W. Guan, B. R. Z. H. y Bi, J., "Electric vehicle routing problem with charging time and variable travel time," Mathematical Problems in Engineering, vol. 2017, 2017, doi:https://doi.org/10.1155/2017/5098183.

[11] Laporte G, R. S. . V. T., "Vehicle routing: Problems, methods, and applications, second edition. chapter 4: Heuristics for the vehicle routing problem," Vehicle Routing, pp. 87–116, 2014, doi:10.1137/1.9781611973594.ch4.

[12] M. Gendreau, J.-Y. Potvin, O. B. G. H. y Løkketangen, A., "Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography," Operations Research/Computer Science Interfaces, vol. 43, p. 143–169, 2008, doi:10.1007/978-0-387-77778-8_7.

3