UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE CIENCIAS

ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

Resolución del Problema de Rutas de Vehículos usando CNN, GNN y el Algoritmo de Clarke y Wright

PROYECTO DE TESIS III

Autor: Jesús Miguel Yacolca Huamán

Asesor: Jaime Osorio Ubaldo

Octubre, 2022

Resumen

En esta tesis se realizará una comparación entre las Redes Neuronales basadas en Grafos y las Redes Neuronales Convolucionales, GNN y CNN respectivamente por sus siglas en inglès, en el problema de Rutas de Vehículos. Se tendrá en cuenta la rapidez en la convergencia para la obtención de parámetros que den resultados óptimos. A su vez se comparará la calidad de resultados de ambas arquitecturas de Redes Neuronales.

Índice general

Resumen III

1. Introducción 1

1.1. Motivación 2

1.2. Objetivos de la Investigación 3

1.2.1. Objetivo General 3

1.2.2. Objetivos Específicos 3

2. Planteamiento del Problema 4

2.1. Descripción del Problema 4

2.2. Formulación del Problema 4

2.3. Justificación del Estudio 5

3. Hipótesis 7

3.1. Hipótesis General 7

3.2. Hipótesis Específica 7

4. Estado del Arte 8

4.1. Marco Histórico 8

4.2. An Overview of Vehicle Routing Problems [1] 9

4.2.1. Resumen 9

4.3. An Investigation Into Graph Neural Networks [2] 9

4.3.1. Objetivos 9

4.3.2. Resumen 9

4.3.3. Conclusiones 10

4.4. An overview of convolutional neural network [3] 10

4.4.1. Objetivos 10

4.4.2. Resumen 10
4.4.3. Conclusiones 10
4.5. A Gentle Introduction to Graph Neural Networks [4] 11
Objetivos 11
4.5.1. Resumen 11
4.5.2. Conclusiones 11
4.6. Understanding Convolutions on Graphs [5] 11
4.6.1. Objetivos 11
4.6.2. Resumen 12
4.6.3. Conclusiones 12
4.7. Finding Solutions to the Vehicle Routing Problem using a Graph Neural Network [6] 12
4.7.1. Objetivos 12
4.7.2. Resumen 12
4.7.3. Conclusiones 13

5. Marco Teórico **14**
5.1. Conceptos Previos 14
5.1.1. Problema de Rutas de Vehículos 14
5.1.2. Problema de Rutas de Vehículos con limitaciones de capacidad (CVRP) 15
5.1.3. Algoritmo de Clarke y Wright 17
5.1.4. Redes Neuronales (NN) 18
5.1.5. Red Neuronal Convolucional (CNN) 20
5.1.6. Red Neuronal basada en Grafos (GNN) 23
5.1.7. Red Convolucional en Grafos (GCN) 26

6. Metodología de Trabajo **29**
6.1. Problema 29
6.2. Datasets 30
6.3. Elección del modelo 30
6.4. Métricas 30

7. Diseño e Implementación **32**

7.1. Herramientas . . . 32

7.1.1. Software . . . 32

Python [7] . . . 32

PyTorch [8] . . . 32

PyTorch Geometric [9] . . . 33

Matplotlib [10] . . . 33

Torchvision [11] . . . 33

NumPy [12] . . . 33

NetworkX [13] . . . 33

OR-Tools [14] . . . 34

scikit-learn [15] . . . 34

7.1.2. Hardware . . . 34

7.2. Métodos de solución . . . 35

7.2.1. Algoritmo de Clarke y Wright . . . 35

VRP . . . 35

CVRP . . . 36

7.2.2. Redes Neuronales . . . 36

7.3. Diseño de los modelos de Redes Neuronales . . . 37

7.3.1. FNN . . . 37

7.3.2. CNN . . . 37

7.3.3. GNN . . . 38

7.4. Tratamiento del dataset . . . 38

7.4.1. FNN . . . 39

7.4.2. CNN . . . 39

7.4.3. GCN . . . 40

7.5. Implementación de los modelos . . . 40

7.5.1. FNN . . . 40

7.5.2. CNN . . . 40

7.5.3. GCN . . . 41

7.6. Entrenamiento . . . 41

7.6.1. Optimizador Adam 42
7.6.2. Función de perdida Divergencia de Kullback-Leibler . . . 42
7.7. Evaluación 44
7.7.1. Métricas 44

8. Presentación de Resultados **45**
8.1. VRP 45
8.1.1. Algoritmo de Clarke y Wright 45
8.1.2. Red Neuronal Feed-Forward 46
8.1.3. CNN 47
8.1.4. GCN 47
8.1.5. Tabla comparativa de Redes Neuronales 49
8.1.6. Comparativa del Algoritmo de Clarke y Wright y GCN . . 49
8.2. CVRP 50
8.2.1. Algoritmo de Clarke y Wright 50
8.2.2. Red Neuronal Feed-Forward 51
8.2.3. CNN 52
8.2.4. GCN 52
8.2.5. Tabla comparativa de Redes Neuronales 53
8.2.6. Comparativa del Algoritmo de Clarke y Wright y GCN . . 53
8.3. Resultados 54

9. Conclusiones y Trabajos Futuros **56**
9.1. Conclusiones 56
9.2. Trabajos Futuros 57

A. Implementación realizada **60**

Índice de figuras

5.1. Feed-Forward Neural Network 20
5.2. Proceso de una Capa Convolucional sobre una entrada 22
5.3. Proceso de una Capa de tipo Max Pooling sobre una entrada . . . 23
5.4. Funcionamiento de una CNN 23
5.5. Representación de la lista de adyacencia de un grafo 25
5.6. Representación del funcionamiento de una GNN 25

7.1. Grafo solución de un problema de CVRP 39

8.1. Resultados del Algoritmo de Clarke y Wright para el VRP 46
8.2. Resultados de la Red Feed-Forward para el VRP 47
8.3. Resultados de la Red Convolucional para el VRP 47
8.4. Resultados de la GCN para el VRP 48
8.5. Predicciones de la GCN para el VRP 48
8.6. Comparativa del Algoritmo de Clarke y Wright y GCN para el VRP 50
8.7. Resultados del Algoritmo de Clarke y Wright para el CVRP . . . 51
8.8. Resultados de la Red Feed-Forward para el CVRP 51
8.9. Resultados de la Red Convolucional para el CVRP 52
8.10. Resultados de la GCN para el CVRP 52
8.11. Predicciones de la GCN para el CVRP 53
8.12. Comparativa del Algoritmo de Clarke y Wright y GCN para el CVRP 54

Índice de cuadros

8.1. Comparativa problema VRP 49
8.2. Comparativa problema CVRP 53

Índice de Acrónimos

VRP	Vehicle Routing Problem
CVRP	Capacited Vehicle Routing Problem
GNN	Graph Neural Network
GCN	Graph Convolutional Network
CNN	Convolutional Neural Network
FNN	Feed-Forward Neural Network
FC	Fully-connected layer
MLP	Multi-layer Perceptron
NN	Neural Network
DNN	Deep Neural Network
DL	Deep Learning
ETC	Etcétera

Agradecimientos

Para la realización de esta tesis se agradece al asesor de tesis por la ayuda en la creación de la misma. A su vez debo agradecer a mi familia por haberme apoyado en el transcurso de mi estadía en la universidad. Al mismo tiempo dar las gracias a los profesores que me han impartido clases que han sido buenas y dieron lo mejor de sí para transmitir conocimientos a lo largo de mi carrera profesional.

Capítulo 1

Introducción

Las Redes Neuronales basadas en Grafos (GNN) son una arquitectura de Red Neuronal (NN) la cuál ha sido recientemente introducida en el campo del Deep Learning. Estas están especializada en el tratamiento de datos con una estructura de Grafo. Esta estructura es fundamental para representar muchos problemas del campo de la Ciencia de la Computación. También es usado en otras áreas como la química y la optimización. Es en este último donde se encuentra el problema de Rutas de Vehículos, el cual utiliza grafos para la representación del problema del siguiente modo: los caminos que unen los nodos, son las aristas, y siendo la distancia de estos caminos los pesos que toman las aristas del grafo.

A su vez, otra conocida arquitectura, las Redes Neuronales Convoluciones (CNN) que ha demostrado ser de las que mejores resultados da en los problemas relacionados a imágenes, las cuales son un tipo especial de grafo conocido como grid; aunque demostrando problemas al tratar con grafos no euclídeos, también podría ser usada para tratar el Problema de Rutas de Vehículos (VRP).

También, se puede tratar este problema utilizando heurísticas. Una de las más conocidas es el Algoritmo de Clarke y Wright. Este es del tipo codicioso; es decir, realiza su búsqueda de la solución buscando maximizar el objetivo en cada iteración. Este enfoque no suele llevar a la solución óptima pues obvia soluciones que puedan no no mejorar la solución actual al principio, pero a largo plazo es mejor.

El Problema de Rutas de Vehículos es importante en el campo de

la optimización. Este tiene diversas variantes en las cuales se agregan restricciones. En su versión más sencilla se busca minimizar el coste, que puede ser la distancia a recorrer, que toma recorrer un número dado de nodos con una cantidad de vehículos a disposición, esto partiendo de un nodo y llegando a este mismo. Este nodo es llamado depot. Una variante que también se buscará resolver en esta tesis, es el Problema de Rutas de Vehículos Capacitivo, CVRP por sus siglas en inglés, en esta variante se añade demandas a cada nodo y las restricciones de capacidad a los vehículos, la forma en que se dan estas capacidades da a su vez otras variantes de este mismo tipo, en el caso de esta tesis se tomará todos los vehículos de igual capacidad.

1.1. Motivación

La motivación para este trabajo es la de dar una revisión e implementación de esta arquitectura que está creciendo en popularidad como son las GNN, para así conocer los alcances que pueda tener. También se busca probar su eficacia en un problema que utiliza la estructura en la que se especializa, el grafo. A su vez se comparara con un tipo de arquitectura más conocido y con una gran cantidad de estudios como son las CNN. Esta última ha demostrado su valía para el procesamiento de imágenes. Sin embargo, este tipo de Red Neuronal no da buenos resultados sobre grafos que no son euclídeos. Es por ello que se escogió el problema de Rutas de Vehículos, un problema sobre grafos, el cual es ideal para demostrar la capacidad de las GNN para tratar este tipo de estructura. También se compararan con un método más conocido, el Algoritmo de Clarke y Wright. Esto para comparar su rapidez y capacidad para alcanzar los resultados.

1.2. Objetivos de la Investigación

1.2.1. Objetivo General

Evaluar las arquitecturas GNN y CNN en dos variantes del Problema de Rutas de Vehículos. VRP y CVRP, para de esta forma comprobar la efectividad de las GNNs sobre el mismo, y comparando estos resultados con un método heurístico como es el Algoritmo de Clarke y Wright.

1.2.2. Objetivos Específicos

- Conseguir mejores resultados con las GNNs con respecto a un modelo más simple como es el de Red Neuronal Feed-Forward básico.
- Determinar la eficiencia de ambas arquitecturas al tratar los problemas de Rutas de Vehículos.
- Determinar la diferencia en la dificultad de resolver el Problemas de Rutas de Vehículos simple y el Problemas de Rutas de Vehículos Capacitivo.
- Interpretar los resultados obtenidos por ambas arquitecturas sobre los problemas de Rutas de Vehículos.
- Examinar la rapidez con la que se alcanza la convergencia del modelo a una elección de pesos que den resultados que puedan ser aceptables.
- Comparar la eficiencia tanto en tiempo como en resultados de las Redes Neuronales con el Algoritmo de Clarke y Wright.

Capítulo 2

Planteamiento del Problema

2.1. Descripción del Problema

El problema a tratar es el de Rutas de Vehículos. Este problema trata de hallar la ruta que optimiza el recorrido de las mismas. Estas rutas son ciclos que parten y llegan a un punto en común que se conoce como centro de distribución o depot. Este problema puede ser representado mediante un grafo. Este es un grafo no euclídeo. Los pesos en las aristas suelen ser las distancias entre los nodos, aunque estos pesos pueden simbolizar algun otro valor relacionado al recorrido entre dos puntos de entrega, los nodos. Existen varias variantes del mismo, en esta tesis se tratará el problema básico y el que tiene restricciones de capacidad de unidades que pueden llevar por vehículo, CVRP en inglés, en esta variante los nodos de entrega tienen una cantidad de unidades como demanda asociada que debe ser cubierta por los vehículos de la flota que tienen una capacidad de transporte limitada.

2.2. Formulación del Problema

Para poder abordar este problema se usará dos tipos de representaciones de las rutas. Estas serán la representación en forma de grafo y la representación en forma de matriz de adyacencia. La primera se usará para entrenar a la arquitectura de Red Neuronal GNN, al subtipo GCN, mientras que la segunda, para la CNN. Esto se debe a que la CNN solo trabaja con grafos del tipo malla. Transformando el problema a uno de trazos sobre una imagen. Después de

entrenar estas arquitecturas se obtendrá la ruta que minimice la suma de los pesos en las aristas. También se resolverá usando el Algoritmo de Clarke y Wright. Este algoritmo utilizará la misma representación que utiliza la GNN.

2.3. Justificación del Estudio

Las redes neuronales (NN) han demostrado ser uno de los métodos del Deep Learning que mejores resultados está dando. Un tipo particular de red neuronal es la Red Neural Profunda (DNN). Esta se compone por capas, dos de las más conocidas son la capa convolucional y la capa recurrente.

Además muchos de los tipos de datos en importantes componentes de la vida actual, como las redes sociales o ramas de la ciencia como la biología, tienen una estructura de grafo.

Esta estructura esta relacionada con muchos problemas de optimización. Problemas como el que se abordará a lo largo de esta tesis que viene a ser el de la Ruta de Vehículos. Este problema es de gran relevancia en el campo de la logística.

En las imágenes las CNN han demostrado tener gran performance gracias a características tales como la compartición de pesos y el uso de múltiples capas para la localización de conexiones (LeCun, Yoshua Bengio, y G. Hinton, 2015). En contraste, suele tener problemas al momento de tratar con grafos no euclídeos.

Sin embargo, otro tipo de red neuronal conocido como GNN que está especializado en el tratamiento de datos con estructura de grafo, da mejores resultados en el tratamiento de datos representados como tal.

El tratamiento de este tipo de datos como grafos suele ser complicado debido a la cantidad de aristas no homogénea que cada nodo en estas contienen. Aunque la importancia de su tratamiento es crucial pues está vinculado a importantes áreas de la matemática como es la optimización, incluso a otras ramas de la ciencia como lo son la medicina o la biología.

Es en el área de la optimización donde esta ubicado el Problema de Ruta de Vehículos, siendo este importante para optimizar el recorrido de rutas. De la solución óptima y rápida de este problema dependen varias aplicaciones del mismo en la actualidad. Una de estas es el planeamiento de la entrega de pedidos por parte de restaurantes. Este consiste en recorrer unos puntos de reparto con una flota de vehículos que parten y llegan al centro de distribución. Lo que se busca minimizar es un coste asociado con este viaje, como puede ser la suma de distancias de recorrido de cada vehículo.

Para comprobar la eficacia de las Redes Neuronales, se les comparará con uno de los métodos más conocidos para la resolución del Problema de Rutas de Vehículos, el Algoritmo de Clarke y Wright. Este es un algoritmo codicioso, es por esta misma característica que para una gran cantidad de nodos, su uso resulta computacionalmente complejo.

Capítulo 3

Hipótesis

3.1. Hipótesis General

La GNN obtendrá mejores resultados que la CNN en la predicción de la ruta que optimice el costo en un problema de VRP y CVRP; sin embargo, el Algoritmo de Clarke y Wright obtendrá mejores predicciones pero con un consumo de tiempo y considerablemente mayor.

3.2. Hipótesis Específica

- Los modelos de Redes Neuronales tardan significativamente menos en tiempo que el Algoritmo de Clarke y Wright.
- La cantidad de iteraciones necesarias para alcanzar resultados aceptables durante el entrenamiento de las CNN será mayor al de las GNN.
- Las predicciones para la variante básica de Problema de Rutas de Vehículos serán mejores que las de la variante que toma en cuenta las restricciones de capacidad, CVRP.
- La robustez de las predicciones no será tan alta debido a la alta complejidad del problema a tratar para las Redes Neuronales.
- El Algoritmo de Clarke y Wright obtendrá los valores más óptimos

Capítulo 4

Estado del Arte

Se dará un marco histórico que da el contexto en el cual se esta realizando esta tesis. A su vez se realizará una sumilla de los libros, artículos o tesis que sustentan este trabajo dando los resúmenes, los objetivos y las conclusiones de cada uno.

4.1. Marco Histórico

El área del Aprendizaje de Máquina, Machine Learning en inglés (ML), ha tenido un crecimiento pronunciado en la última década. En especial la área que refiere al Aprendizaje Profundo, Deep Learning en inglés (DL), gracias al desarrollo de un tipo de algoritmo conocido como las Redes Neuronales (NN). Inicialmente se utilizaba el Perceptrón Multicapa (MLP), una de las versiones más básicas del tipo de Red Neuronal Feed-Forward, para los problemas de clasificación. Luego se dio un nuevo avance en este tipo de problemas gracias a las Redes Neuronales Convolucionales (CNN) que se especializan en los problemas referidos a imágenes. En el presente ha surgido un tipo de arquitectura nuevo especializada en los problemas con datos estructurados como grafos. Esta arquitectura se llama Graph Neural Network (GNN). Esta arquitectura fue presentada en 2009 en el paper titulado "The Graph Neural Network Model"[16]. Ahora las GNN son potenciadas gracias al uso de la convolución tomando el nombre de Convolutional Graph Network (GCN).

Estas mismas podrían dar solución a uno de los problemas más importantes en lo que a optimización se refiere, como es el Problema de Rutas de Vehículos.

Esta tradicionalmente es resuelto mediante heurísticas o métodos exactos, ahora puede ser resuelto por este tipo de red neuronal que se especializa en este tipo de estructura como es el grafo.

4.2. An Overview of Vehicle Routing Problems [1]

4.2.1. Resumen

Este libro da una revisión a los Problemas de Rutas de Vehículos, VRP por sus siglas en inglès, dando el marco teórico necesario para la elaboración de esta tesis. Escrito en 2014, da soluciones a estos problemas mediante algoritmos heurísticos o exactos. Sin embargo, no aborda su resolución mediante algoritmos del campo del Machine Learning o del Deep Learning.

4.3. An Investigation Into Graph Neural Networks [2]

4.3.1. Objetivos

- Dar una revisión de la arquitectura de Red Neuronal GNN.
- Describir las librerías y frameworks necesarios para crear una GNN y como implementarla.

4.3.2. Resumen

En esta tesis desarrollada por V. Kumar se da una revisión de las GNN. Dando una sustentación teórica de las mismas. Además, se da un resumen de las librerías y frameworks que son necesarios para el desarrollo de las GNNs. También se dan las limitaciones de las mismas. Luego se implementan para resolver distintos tipos de tareas sobre datasets conocidos como lo son MNIST y Cora.

4.3.3. Conclusiones

- Es complicado encontrar la librería correcta para cada experimento.
- A pesar de que las GNN no daban resultados satisfactorios en muchas condiciones, estas logran conseguir resultados destacables en otras.

4.4. An overview of convolutional neural network [3]

4.4.1. Objetivos

- Realizar una revisión de los fundamentos detrás de la arquitectura CNN.
- Describir las aplicaciones de las CNN.

4.4.2. Resumen

En este paper se realiza una revisión de la arquitectura de las CNN. Primero se presenta los fundamentos de la arquitectura. Luego se presenta algunas revisiones de la arquitectura más complejas. Finalmente se dan algunos casos de aplicaciones.

4.4.3. Conclusiones

- Se revisó la arquitectura de las CNN y sus aplicaciones.
- Las CNN han demostrado ser de las mejores arquitecturas para las tareas relacionadas al campo de la Visión Computacional.

4.5. A Gentle Introduction to Graph Neural Networks [4]

Objetivos

- Explorar y explicar la arquitectura de las modernas GNN.

4.5.1. Resumen

En este trabajo se detalla de una manera simple y entendible los conceptos principales concernientes a las GNN. Se divide en 4 partes. En la primera se da una explicación y ejemplos de los tipos de datos que son mejor representados por grafos. En la segunda se da una explicación de que hace diferente a los grafos de otros tipos de datos. Para la tercera se construye una GNN explicando en este proceso los conceptos de la misma. En la última parte se provee una representación animada de una GNN en la cual se puede ajustar sus pesos para un mayor entendimiento de su funcionamiento.

4.5.2. Conclusiones

- Los grafos son una estructura de datos poderosa que genera retos diferentes a los vistos con imágenes y texto.
- Se revisó en el artículo algunas de las decisiones de diseño importantes que se deben tener en cuenta al momento de diseñar una GNN.

4.6. Understanding Convolutions on Graphs [5]

4.6.1. Objetivos

- Ilustrar los retos de la computación relacionada a grafos.
- Explorar las variantes más recientes de las GNN.

4.6.2. Resumen

En este artículo se da una revisión de la arquitectura de las GNN. Además, se da una explicación de la problemática del uso de la convolución convencional en el tratamiento de grafos. En particular se trata el problema de la falta de orden en los nodos. A esto se muestra una solución mediante la extensión de la noción de laplaciano a grafos. También se muestran variantes de las GNN como lo son las GCN.

4.6.3. Conclusiones

- En el artículo se da una muestra del campo amplio de las GNN a la vez de mostrar varias técnicas e ideas relacionadas a este campo.
- Se comunicaron las ideas principales para entender las GNN y algunas de sus variantes.

4.7. Finding Solutions to the Vehicle Routing Problem using a Graph Neural Network [6]

4.7.1. Objetivos

- Explorar GNN para resolver el VRP.
- Uso de GNN, en especifico, Recurrent Relational Network, para obtener las probabilidades de cada arista y luego usar Beam Search para obtener el grafo.

4.7.2. Resumen

En esta tesis se utilizó un subtipo de GNN, conocido como RRN o Red Recurrente Relacional, para obtener las probabilidades de elección de cada arista y así mediante el uso de Beam Search obtener el grafo que de solución

al Problema de Rutas de Vehículos sin restricciones, con 20 y 50 nodos, y al Problema del Vendedor Viajero con 20 nodos.

4.7.3. Conclusiones

- El modelo tiene mejor performance en VRP 20 y VRP 50, pero no en TSP 20 que OR-Tools. Esto puede deberse a que esta herramienta maneja eficientemente TSP, pero para VRP con muchos nodos tarda en alcanzar una solución.
- Aunque RRN no es bueno prediciendo datos de test, solo de entrenamiento, es un buen punto de partida para Beam Search.
- Los mejores resultados son dados en las iteraciones iniciales, esto junto a la irregularidad progresión sugiere que el carácter relacional de RRN no aporta mejoras significativas, por lo que, otras arquitecturas de GNN, pueden dar mejores resultados.
- Mejor data ayudaría a obtener mejores resultados, pero esto es complicado, siendo la obtención de datasets una de las debilidades del aprendizaje supervisado.

Capítulo 5

Marco Teórico

5.1. Conceptos Previos

5.1.1. Problema de Rutas de Vehículos

La definición para este problema dada por el libro An Overview of Vehicle Routing Problem [1] es la siguiente: "Dado un conjunto de requerimientos de transporte y una flota de vehículos se pide determinar un conjunto de Rutas de vehículos que satisfaga todos (o la mayoría) de requerimientos de transporte con la flota de vehículos dada con el mínimo coste; en particular, decidir que vehículos manejaran cuales requerimientos y en que orden de tal forma que todas las rutas de vehículos puedan ser factibles."

Así como se menciona en el libro citado anteriormente, este problema tiene implicaciones directas en el mundo real, tales como la planificación de rutas para vehículos auto-guiados.

Existen muchas versiones de este problema como el Problema de Rutas de Vehículos con Ventanas de Tiempo (Vehicle Routing Problem with Time Windows VRPTW en inglés) o la versión estocástica conocida como Problema de Rutas de Vehículos Estocástico (Stochastic Vehicle Routing Problem SVRP en inglès). Sin embargo la versión que se abordará en esta tesis será la conocida como Problema de Rutas de Vehículos con limitaciones de capacidad, CVRP por sus siglas en inglès.

5.1.2. Problema de Rutas de Vehículos con limitaciones de capacidad (CVRP)

Esta es la variante más estudiada de las que existen en la familia de problemas de Rutas de Vehículos. Además de ser fácil de entender

Antes de definir el problema formalmente, es necesario conocer la terminología que se usa. En este modo se define un "*depot*" o centro de distribución, usualmente denotado como "0", desde donde se parte a realizar las entregas a los "*consumidores*" denotados como una lista $N = \{1, 2, 3, ...\}$ de los nodos que los representa y lo que solicita cada consumidor $i \in N$ se denomina "*demanda del consumidor*" el cual es un número $q_i \geq 0$ que podría ser el peso de las entregas solicitadas. La "*flota de vehículos*" de la que se dispone en el *depot* para realizar las entregas es representada por la lista $K = \{1, 2, ..., |K|\}$, cada vehículo puede llevar una carga $Q > 0$. Un vehículo que responda a la *demanda* de un subconjunto $S \in N$ empieza su recorrido en el *depot* recorre cada consumidor en S una vez y retorna a donde partió inicialmente. Un vehículo que recorra el tramo entre el nodo i y j conlleva un "*costo de viaje*" c_{ij}.

Definiendo el conjunto de vértices de un grafo que represente este problema como $V = \{0\} \cup N$ y $q_0 = 0$. A partir de esto se puede construir un grafo dirigido o no dirigido dependiendo de como se defina c_{ij} y c_{ji},)el camino contrario entre estos dos nodos.

Si se toma como iguales; es decir, la ida cuesta igual que la vuelta, entonces se tiene un grafo no dirigido. En este el conjunto de aristas es $E = \{e = \{i, j\} = \{j, i\} : i, j \in V, i \neq j\}$. Así se tiene el grafo no dirigido $G = (V, E)$.

Si existe $c_{ij} \neq c_{ji}$ entonces un camino tiene un costo diferente al ir y volver. Debido a esto se tiene un grafo dirigido donde el conjunto de arcos es $A = \{(i, j) \in V \times V : i \neq j\}$

Ahora el grafo para el CVRP esta completamente definido, teniendo una versión no dirigida $G = (V, E, c_{\{ij\}}, q_i)$ y una dirigida $G = (V, A, c_{\{ij\}}, q_i)$. Junto a la cantidad de vehículos disponible $|K|$ y su capacidad Q.

Ahora se define el concepto de "*ruta*" como $r = \{i_0, i_1, i_2, ..., i_s, i_{s+1}\}$ con

$i_0 = i_{s+1} = 0$ donde el subconjunto $S = \{i_1, ..., i_s\} \subseteq N$ es visitado. El costo de esta ruta es $c(r) = \sum_{p=0}^{s} c_{i_p i_{p+1}}$. Esta ruta es factible si cumple las restricciones de capacidad del vehículo que la realiza, $q(S) = \sum_{i \in S} q_i \leq Q$ y además un nodo es visitado solo una vez. De esta forma $S \subseteq N$ se denomina como un conjunto factible.

Una solución al CVRP consta de $|K|$ rutas factibles, cada una por cada vehículo $k \in K$, teniendo así las rutas $r_1, r_2, ..., r_{|K|}$ y sus conjuntos $S_1, S_2, ..., S_{|K|}$ proveyendo una "*solución factible*" si todas las rutas son factibles y los conjuntos forman una partición en N.

Para definir el problema como uno de programación entera mixta se tiene la siguiente notación. Sea $S \subseteq N$ arbitrario. Para la forma de grafo no dirigido se tiene el "*conjunto de corte*" $\delta(S) = \{\{i, j\} \in E : i \in S, j \notin S\}$ el conjunto de aristas con uno o ambos puntos terminales en S. Para la forma como grafo dirigido se tiene $\delta^-(S) = \{(i, j) \in A : i \notin S, j \in S\}$ y $\delta^+(S) = \{(i, j) \in A : i \in S, j \notin S\}$.

Para un conjunto de compradores $S \subseteq N$, $r(S)$ es el número mínimo de rutas necesario para cubrir S. A la vez, se denota como x_{ij} la frecuencia con la cual un vehículo cruza la arista i, j.

Ahora se puede modelar para la forma como grafo dirigido como sigue:

$$\text{minimizar} \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{5.1}$$

$$\sum_{j \in \delta^+(i)} x_{ij} = 1 \quad \forall i \in N, \tag{5.2}$$

$$\sum_{i \in \delta^-(j)} x_{ij} = 1 \quad \forall j \in N, \tag{5.3}$$

$$\sum_{j \in \delta^+(0)} x_{0j} = |K|, \tag{5.4}$$

$$\sum_{(i,j) \in \delta^+(S)} x_{ij} \geq r(S) \quad \forall S \subseteq N, S \neq \emptyset, \tag{5.5}$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \tag{5.6}$$

Para la versión no dirigida se tiene:

$$\text{minimizar} \sum_{\{i,j\}\in E} c_{ij}x_{ij} \tag{5.7}$$

$$\sum_{j\in\delta(i)} x_{ij} = 2 \quad \forall i \in N, \forall j \in N \tag{5.8}$$

$$\sum_{j\in\delta(0)} x_{0j} = 2|K|, \tag{5.9}$$

$$\sum_{(i,j)\in\delta(S)} x_{ij} \geq 2r(S) \quad \forall S \subseteq N, S \neq \emptyset \tag{5.10}$$

$$x_{ij} \in \{0,1,2\} \quad \forall (i,j) \in \delta(0) \tag{5.11}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in E \setminus \delta(0) \tag{5.12}$$

5.1.3. Algoritmo de Clarke y Wright

En 1964 Clarke y Wright proponen un algoritmo llamado Algoritmo de Ahorros. Este algoritmo hace uso de un enfoque codicioso; esto significa que prioriza la solución que sea mejor en cada iteración. Este enfoque suele evitar soluciones que puedan parecer peores al momento de ser elegidas, pero terminan siendo las soluciones óptimas. Es una de las heurísticas más conocidas para resolver el problema de Rutas de Vehículos. El algoritmo es de tipo constructivo; esto significa que la ruta se va construyendo en cada iteración del algoritmo. El nombre hace referencia a la técnica que se utiliza, la de calcular el ahorro de combinar 2 rutas. Se tiene una ruta del depot 0 a un nodo i, la ruta $0 \to i$ y de regreso al depot, $i \to 0$, esta ruta se conoce como ruta $0 \to i \to 0$. De esta misma forma se tiene otra ruta $0 \to j \to 0$. Entonces el ahorro es calculado como sigue:

$$(0 \to i + i \to 0 + 0 \to j + j \to 0) - (0 \to i + 0 \to j + i \to j) \tag{5.13}$$

Esto es el costo de recorrer las rutas para llegar a ambos nodos por separado menos el costo de recorrer la ruta que los unes dando solo un recorrido. Si se tiene que el camino de ida y de regreso tienen igual costo, entonces se puede

calcular el ahorro como sigue:

$$0 \to i + 0 \to j - i \to j \tag{5.14}$$

Si al calcular el ahorro se obtiene un valor positivo, esto significa que la combinación de estas rutas es mejor que recorrer ambas por separado. Este calculo del ahorro se realiza para cada combinación de nodos sin incluir al depot. Se juntan las que produzca mayor ahorro. El principal problema del algoritmo es la cantidad de combinaciones de nodos que se necesitan para ejecutarlo. Siendo computacionalmente difícil de aplicar para más de 25 nodos.

5.1.4. Redes Neuronales (NN)

Una Red Neuronal es un algoritmo del Aprendizaje Supervisado que, en su forma más simple, es una serie de capas compuestas por neuronas que están conectadas entre sí. Esto viene inspirado de la biología donde las neuronas transmiten información entre si mediante pulsos eléctricos en sus conexiones. Esta transmisión de información en las Redes Neuronales se da de manera similar asignando un peso a cada conexión, que en una Red Neuronal simple se conecta cada neurona en una capa con todas las neuronas en la capa siguiente. Los pesos se pueden colocar de manera simplificada en un vector para cada neurona. La unión de estos pesos en forma de vector de cada neurona en una capa forma una matriz conocida como "Matriz de pesos" representada con la letra W.

Este algoritmo realiza un proceso de aprendizaje el cual le permite ajustar sus parámetros en base a los datos que se le proporcionen. Estos parámetros son las matrices de peso de cada capa. De esta forma el proceso de aprendizaje consiste en actualizar estas matrices de pesos. Existen varios algoritmos que realizan esta tarea. Sin embargo, el más usado es el algoritmo "Back-propagation". Este realiza un proceso de optimización sobre los parámetros para minimizar una función de coste. Esta función mide la diferencia entre las salidas que se desean obtener y las que se obtienen

al culminar el proceso del cálculo hacia adelante o "Forward calculation". El proceso de optimización se conoce como "Backward propagation" o propagación hacia atrás debido a que este construye una función en base a las derivadas parciales de las capas anteriores.

Las capas de las Redes Neuronales se pueden dividir en tres grupos de acuerdo a su función y relación con los datos que se le proporcionen:

- **Capa de entrada:** Es la que esta conformada por los datos de los que se dispone para realizar la predicción.
- **Capas ocultas:** Estas dan la robustez a la red para procesar los datos mediante la agregación de múltiples capas conformadas por un número de neuronas no fijo para cada una de estas, tanto la cantidad de neuronas por capa como la cantidad de capas se determina de tal forma que se alcance un nivel de predicción aceptable.
- **Capa de salida:** De esta capa salen los valores predichos por la Red Neuronal.

Aunque con el pasar del tiempo y la realización de más estudios se han creado tipos de capas especializadas en ciertas tareas como la predicción de texto (RNNs) o clasificación de imágenes (CNNs). A continuación, se presenta tres tipos que son de vital importancia para este estudio:

- **Feed-Forward Neural Network (FNN):** Es el tipo más simple de Red Neuronal. Consiste en las 3 capas ya mencionadas. Además, posee la restricción de que una neurona de una capa solo puede conectarse a las neuronas de las capas vecinas. Un ejemplo de esto se puede ver en la Figura 5.1

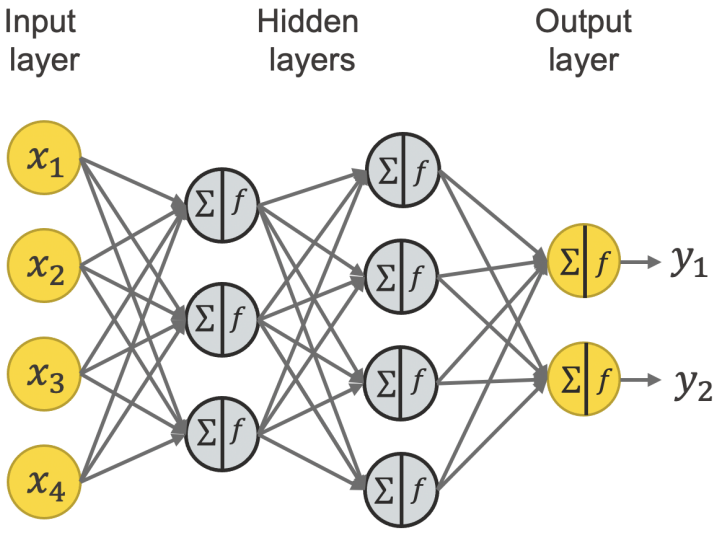

FIGURA 5.1: Feed-Forward Neural Network

- **Red Neuronal Convolucional (CNN):** Esta es un tipo particular de la anteriormente mencionada. Pero a diferencia de esta la conectividad local se preserva. [2]. Este tipo de capa también está compuesta por subcapas como:

  - Capa Convolucional
  - Capa de reducción de muestra o Pooling Layer
  - Capa simple donde todas las neuronas están conectadas con las de la capa inmediatamente anterior y posterior.

- **Red Neuronal basada en Grafos (GNN):** Esta capa esta especialmente diseñada para tratar con tipos de datos estructurados como grafo que pueden ser no euclídeos. Tiene subtipos que pueden aprovechar la operación de convolución. Uno de estos se conoce como GCN.

5.1.5. Red Neuronal Convolucional (CNN)

Una Red Neuronal Convolucional está compuesta, como se mencionó en la sección anterior, por tres tipos diferentes de capas: una capa convolucional, una capa Pool y una capa completamente conectada o Fully-conncected (FC).

Las CNNs se centran en la idea primaria que la entrada contiene elementos que se relacionan entre si de forma espacial, como lo es una imagen, esto enfoca la arquitectura a construir en buscar un método que de solución a la necesidad de lidiar con esta forma particular de la información. Este método es la operación de convolución. También, una de las características de las CNNs es que las capas intermedias están compuestas por neuronas organizadas en 3 partes conocidas como la dimensión espacial de la entrada. [3]

A continuación se realizará una revisión de los 3 tipos de capas que componen esta arquitectura, especializada en el procesamiento de imágenes, las cuales son cruciales para el entendimiento de esta.

Capa convolucional

Esta capa es la más importante que posee esta arquitectura. Además, es la que mayor coste computacional genera. Esta realiza el proceso de la convolución sobre la entrada, que es una rejilla como lo puede ser una imagen, siguiendo unos parámetros que se centran en el empleo de lo que se conoce como kernels de aprendizaje. Estos suelen ser de una dimensionalidad reducida. Estos "recorren" completamente la entrada. La capa convolucional opera de la siguiente forma sobre las entradas. A cada entrada se le aplica una multiplicación entre los pesos, que son los componentes del kernel, y una región asociada de la entrada para luego sumarlos dando un mapa en forma de matriz de la misma. Una imagen que ilustra este procedimiento es la Figura 5.2. Esta capa posee tres hiperparámetros que ayudan a controlar las dimensiones de la salida de la misma los cuales son el padding (P) que es un "marco" de números que rodean a la entrada; el stride (S), que es la cantidad de pasos que "salta" el kernel en su recorrido por una dimensión de la entrada; y la dimensión del kernel. Estos junto a las dimensiones de la entrada (V) permite calcular las dimensiones de la salida. A partir del stride y padding es posible calcular las dimensiones del kernel necesario para generar una salida

de tamaño determinado (R) usando la siguiente fórmula:

$$\frac{(V-R)+2P}{S+1} \tag{5.15}$$

El proceso de entrenamiento de este tipo de red también hace uso del algoritmo de Backpropagation.

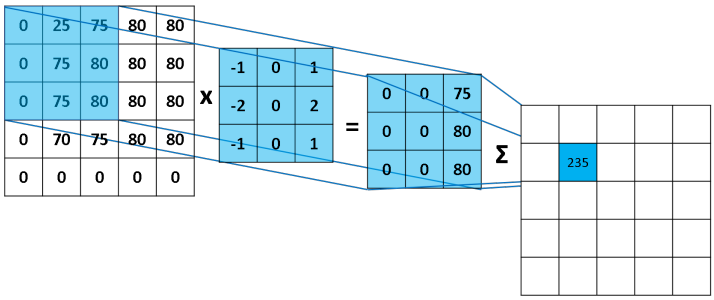

FIGURA 5.2: Proceso de una Capa Convolucional sobre una entrada

Capa de reducción de muestra o Pooling Estas capas también hacen parte de la arquitectura de una CNN y normalmente se le puede encontrar después de una capa convolucional. Siendo así la salida de la capa convolucional la entrada de la capa de Pooling. La función de este tipo de capas es realizar una reducción de la dimensión de sus entradas mediante el uso de funciones como calcular el valor más común o el mayor en una región donde se aplique. La importancia de esta capa radica en el hecho que estas pueden reducir la complejidad del modelo. Existen varios tipos de estas capas como son Max pooling basado en hallar el valor máximo de la región u otro tipo llamado Average pooling basado en hallar la media de la región. La imagen que se puede ver en la Figura 5.3 ofrece una explicación visual de cómo funciona una capa de Pooling, en concreto una Max Pooling.

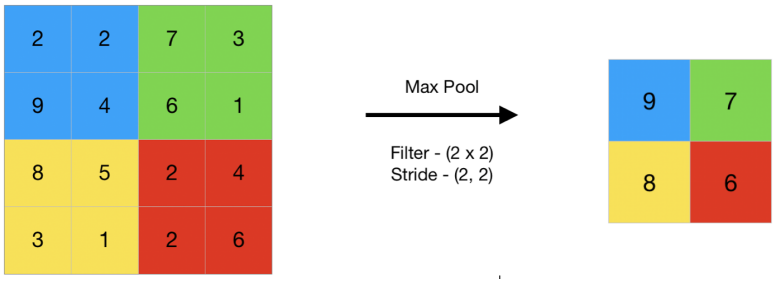

FIGURA 5.3: Proceso de una Capa de tipo Max Pooling sobre una entrada

Capa Fully-Connected Esta capa ya se vio en la sección anterior. Esta capa como parte de la arquitectura de las CNNs se coloca al final pues esta capa no encapsula bien la información espacial. También es usada para "aplanar" la salida de las capas de Pooling que llegan en una dimensión superior a 1. Sin embargo, esta no es indispensable para la creación de la arquitectura de las CNNs pues recientemente algunos diseños reemplazan esta con otra capa llamada General Average Pooling [17].

Una visualización de cómo funcionan estas capas conjuntamente se puede ver en la imagen de la Figura 5.4

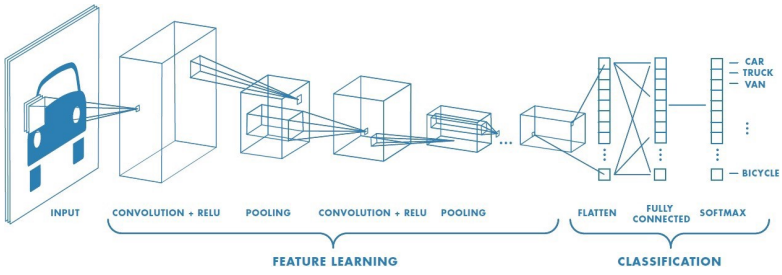

FIGURA 5.4: Funcionamiento de una CNN

5.1.6. Red Neuronal basada en Grafos (GNN)

Los grafos son una estructura de datos que está muy presente en la vida real pues esta sirve para representar la conexión entre grupos de objetos. Y es por esto que se han realizado estudios sobre redes neuronales que operan sobre grafos. El tipo de red neuronal que opera con este tipo de entrada recibe el nombre de GNN. Los estudios de esta arquitectura de red neuronal llevan realizándose más de una década. Sin embargo, son los avances recientes

los que han aumentado sus capacidades en el poder de representación de la data. Gracias a esto se está empezando a ver utilidades en campos como la simulación de sistemas físicos o la detección de noticias falsas. [4]

Un grafo puede ser representado mediante un conjunto de nodos y las aristas que los unen. Esto a su vez puede representarse con una matriz conocida como "Matriz de Adyacencia". En este sentido las imágenes también pueden ser representadas como grafos. En esta representación cada pixel es un nodo y se coloca una arista entre este y los vecinos que posee o un texto como un grafo donde cada palabra es un nodo y esta conectada con la anterior y siguiente, si estas están. Aunque esta representación puede ser muy redundante pues solo se crea una banda en la matriz de adyacencia, esto en el caso de las imágenes. Además esta representación es sensible a las etiquetas que tengan los nodos.

Existen 3 tipos de problemas a resolver en grafos. Estos son a nivel de grafo, nodo o arista. Un ejemplo de un problema a resolver a nivel de grafo relacionado a imágenes es la clasificación de estas. En cuanto a una tarea a nivel de nodo es la segmentación de imágenes. Y un ejemplo a nivel de arista es establecer relaciones entre los objetos de una imagen. En el caso de esta tesis se realizará un problema a nivel de arista como es la predicción de las conexiones que dan el grafo que da solución al problema de Rutas de Vehículos.

Para representar un grafo se puede utilizar la Matriz de Adyacencia, pero esta no es eficiente pues ocupa mucha memoria y no es única para un mismo grafo llegando a tener varias de estas. Una alternativa a esto es una lista de adyacencia donde el elemento $K - esimo$ es la representación de la arista que conecta un nodo i con uno J de la forma (i, j). Un ejemplo de esto se puede ver en la Figura 5.5.

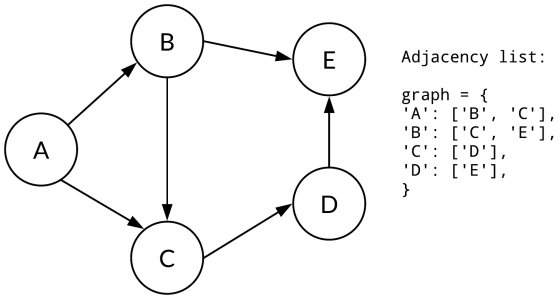

FIGURA 5.5: Representación de la lista de adyacencia de un grafo

Un modelo simple de GNN es pasar los nodos, las aristas o el contexto global del grafo a través de un MLP (Tipo especial de Red Feed-Forward). De esta forma se puede resolver las distintas problemáticas que se mostraron anteriormente y así realizar predicciones. Por ejemplo, si queremos realizar predicciones en el problema a nivel de nodo, solo es necesario aplicar el MLP a los nodos de este. Una forma de visualizar esto se puede ver en la Figura 5.6.

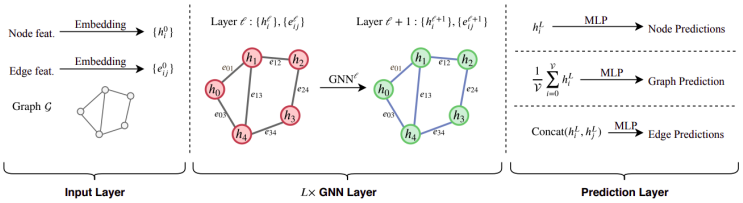

FIGURA 5.6: Representación del funcionamiento de una GNN

A veces no se disponen de la información de todas las partes que se utilizan para representar el grafo, como la de los nodos o aristas, afortunadamente este problema puede solventarse gracias a una técnica conocida como la transferencia de mensajes. Esta consiste en, generalmente, sumar la información contenida en los objetos adyacentes al que necesitamos suministrar información. Por ejemplo, si necesitamos información de los nodos, pero solo tenemos la de los vértices se puede obtener esta información como la suma de los vectores de las aristas de las cuales este nodo es parte.

El proceso de transferir mensajes también se puede realizar entre distintas capas de la GNN. Por ejemplo, se puede transferir información de los nodos

de la capa anterior sumando a los vectores de los nodos de la nueva capa los vectores de los nodos vecinos a este en la capa anterior por cada nodo en la nueva capa.

Existen diversos subtipos de GNNs. Sin embargo, esta tesis se centrará en las GCNs.

5.1.7. Red Convolucional en Grafos (GCN)

Esta red es parte de las GNNs. Esta hace uso de la generalización de la operación de convolución de las CNNs, que opera sobre un grafo en forma de rejilla que es como se puede representar a una imagen como grafo, a un tipo más general de grafo. Esto puede resultar complicado pues un grafo, en general, es invariante bajo el orden de los nodos; es decir, no importa el orden en el cual se etiqueten los nodos del mismo. Esta última propiedad otorga flexibilidad en la representación de datos, pero la convolución depende de que el grafo sea de un tipo especial conocido como rejilla. Además de esto la estructura de un nodo a otro cambia sustancialmente pues puede tener diferente número de nodos vecinos. [5]

Una forma de conseguir utilizar la operación de convolución en grafos es mediante la definición de filtros sobre estos como se hace en las CNNs.

Primero se mostrará el Laplaciano de un grafo que se puede definir como sigue:

$$L = D - A \tag{5.16}$$

Donde A es la Matriz de Adyacencia del grafo y D es una matriz diagonal que se define como sigue:

$$D_v = \sum_u A_{vu} \tag{5.17}$$

Esto esencialmente viene a ser que para el nodo v el valor es la suma de esa fila en la matriz de adyacencia. Este Laplaciano representa la misma información que A. Además, a partir de A se puede hallar L y viceversa.

Ahora se puede formar polinomios con estos de la siguiente forma:

$$p_w(L) = \sum_{i=0}^{d} w_i L^i \tag{5.18}$$

Ahora este polinomio de grado d puede ser un filtro tal y como se tienen en una CNN con los coeficientes w como los pesos. Ahora si consideramos a la información asociada a un nodo como un valor único real; entonces, podemos concatenar estos para cada nodo obteniendo un vector x. De esta forma aplicar el filtro obteniendo un vector x' es equivalente a:

$$x' = p_w(L)x \tag{5.19}$$

Un dato importante a considerar sobre el grado d del polinomio es que este afecta a como se realiza el filtro en un nodo v, este solo se realiza entre nodos que no estén separados más de d pasos.

Se dice que un algoritmo f es equivariante si dada una permutación P sobre la entrada x se cumple que:

$$f(Px) = Pf(x) \tag{5.20}$$

Esta propiedad es indispensable para conseguir un operador que actúe como la convolución sobre grafos. Y, en efecto, se puede probar que el polinomio obtenido anteriormente es equivariante del orden de los nodos. Por tanto, esta forma de definir la convolución es equivariante tal y como se buscaba. Además, se puede ver que con un polinomio de grado d un nodo v solo afecta otro nodo u como ya se mencionó antes. Esto es similar a una transferencia de información entre los nodos. Si este paso se repitiese k veces, entonces el mensaje se propagará a todos los nodos del grafo que estén a una distancia k entre sí.

Una manera de diferenciar a los diferentes tipos de GNNs es mediante el cálculo del embedding. Como esta tesis se centrará en el uso de las GCN este

será el que se presentará a continuación [5]:

$$h_v^{(0)} = x_v \qquad \forall\, v \in V \tag{5.21}$$

Aquí $h_v^{(0)}$ es el embedding inicial del nodo v. Para K iteraciones se pueden calcular los enbeddings como sigue:

$$h_v^{(k)} = f^{(k)} \left(W^{(k)} \cdot \frac{\sum\limits_{u \in \mathcal{N}(v)} h_u^{(k-1)}}{|\mathcal{N}(v)|} + B^{(k)} \cdot h_v^{(k-1)} \right) \qquad \forall\, v \in V \tag{5.22}$$

Tanto $f^{(K)}$ como $W^{(K)}$ y $B^{(K)}$ son los mismos para todos los nodos. Las predicciones pueden obtenerse como sigue:

$$y_v = \text{PREDICT}\left(h_v^{(K)}\right) \tag{5.23}$$

Aquí PREDICT suele ser otra red neuronal que trabaja junto a la GCN. Esta formulación de la normalización del embedding es diferente a la dada en el paper original donde se presentaron las GCN.

Capítulo 6

Metodología de Trabajo

En esta sección se mostrará la forma en la que se procedió a resolver el problema de Rutas de Vehículos con las CNN, GNN y el Algoritmo de Clarke y Wright. En este sentido se dividirá este capítulo en 4 secciones. En la primera se detallará la forma de abordar el problema de obtener el grafo que de solución al problema. Seguidamente se especificará como se obtuvieron los datasets a usar. Luego se detallará la elección de los modelos. Finalmente se mostrará la forma de comparar los resultados de las implementaciones, tanto con CNN como GNN, en base a determinadas mètricas. Lo cual se comparará con los resultados obtenidos por el Algoritmo de Clarke y Wright y el tiempo que tardan en obtenerlos.

6.1. Problema

Como se ha ido mencionando en secciones anteriores, el problema a tratar es el de Rutas de Vehículos. El motivo de esto se debe a la gran importancia de este en la optimización y con aplicaciones varias en la industria. Además de dar una solución que implique el uso de Redes Neuronales y sea eficiente. El resultado que se busca es el grafo que de solución al mismo. Se tratará dos variaciones de este problema. Se resolverá la forma básica del problema y la que presenta variaciones de capacidad, conocido como CVRP. Además ambos problemas son resueltos para 25 nodos, incluyendo el nodo depot. La razón de usar 25 nodos es que este es el umbral que se considera para el cual el problema es tratable con

el Algoritmo de Clarke y Wright. También se considera que todos los vehículos cuentan con igual capacidad.

6.2. Datasets

Se generaran 10000 instancias de VRP y otras 10000 de CVRP, mediante el uso de una librería especializada en resolver problemas con restricciones conocida como OR-Tools, la cual proporcionará el respectivo grafo solución de cada problema.

6.3. Elección del modelo

Primero se tendrá como solución base una brindada por un MLP, el cual es una versión simple de una Feed-Forward Neural Network. Este se comparará al modelo más simple de CNN con un subtipo de GNN. Para la elección del subtipo de modelo de GNN se tendrá en cuenta la complejidad del modelo y el que pueda tener en cuenta los nodos vecinos. Para el caso de la Red Feed-Forward y la CNN es un problema de regresión para predecir un vector. En el caso de la GNN es de predicción de aristas/clasificación binaria donde 1 es que hay una arista y 0 que no esta.

6.4. Métricas

Para la comparación se obtendrá los valores de una métrica conocida como "Explained Variance" y la perdida en cada época para los modelos que se implementaran.

La métrica "Explained Variance" se calcula como sigue:

$$\text{explained\_variance\_score}(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}} \tag{6.1}$$

Donde $\hat{y}$ es el valor predicho y y es el valor que se desea. Luego estas se graficarán para la observación de cual obtiene mejores resultados en los datos de evaluación y la rapidez con la cual los obtiene.

Para comparar las Redes Neuronales con el Algoritmo de Clarke y Wright se mostrará en una tabla el tiempo que tardan en obtener la solución. También se comparará el valor obtenido por ambos en otra tabla.

Capítulo 7

Diseño e Implementación

En este capítulo se mostrará las herramientas usadas para la implementación y evaluación de los modelos. Así como el diseño e implementación de los modelos. para ver detalladamente el código implementando los modelos de esta tesis, así como los datasets usados y los modelos entrenados revisar el siguiente apéndice A.

7.1. Herramientas

7.1.1. Software

Python [7]

Como dice en su página: "Python es un lenguaje de programación que le permite trabajar más rápido e integrar sus sistemas de manera más efectiva..Este es el lenguaje que se usa para la implementación en esta tesis.

PyTorch [8]

Es un framework de diferenciación automática que se usó para la implementación de los modelos pues cuenta con funciones que facilitan el proceso de creación de redes neuronal complejas como es el caso de las CNN y la FNN.

PyTorch Geometric [9]

Es una librería basada en el framework PyTorch especializada en el diseño de Redes Neuronales basadas en Grafos o GNN. En la presente tesis se hace uso de este para la implementación de un subtipo de GNN conocido como GCN.

Matplotlib [10]

Es una librería especializada en la creación y visualización de gráficos. En este trabajo se utilizó para realizar la visualización de las gráficas de la evolución de la función de perdida y la precisión a lo largo de las épocas.

Torchvision [11]

Esta librería es una extensión del proyecto PyTorch. Se especializa en el tratamiento de imágenes y tiene datasets relacionados al campo de visión artificial. En este caso se utilizó para el tratamiento de las matrices de adyacencia como si fueran imágenes.

NumPy [12]

Esta librería se especializa en la computación científica en Python. En lo referente a este trabajo se utilizan funciones como la media para calcular las métricas o el reescalado de dimensiones para el tratamiento del dataset. También para el tratamiento de las matrices de datos.

NetworkX [13]

Como dice su página web: "NetworkX es una librería de Python para la creación, manipulación y estudio de la estructura, dinámica y funciones en redes complejas..[En] el caso de esta tesis se utiliza para la representación de los grafos asociados al Problema de Rutas de Vehículos.

OR-Tools [14]

Como dice su página web: .ºR-Tools es una librería de software de código abierto para problemas de optimización, ajustado para abordar los problemas más difíciles del mundo en rutas de vehículos, flujos, programación entera y lineal, y programación con restricciones..En el caso de esta tesis se utiliza para generar las soluciones a los Problemaa de Rutas de Vehículos y CVRP que serán el target del dataset.

scikit-learn [15]

Como dice su página web: "Scikit-learn es una biblioteca de aprendizaje automático de código abierto que admite el aprendizaje supervisado y no supervisado. También proporciona varias herramientas para el entrenamiento de modelos, preprocesamiento de datos, selección de modelos, evaluación de modelos y muchas otras utilidades..En el caso de esta tesis se utiliza las utilidades que ayudan al tratamiento del dataset.

7.1.2. Hardware

Es vital el uso de altos requerimientos de hardware para el proceso de entrenamiento de modelos de aprendizaje automático. Existen varias formas de entrenar usando este tipo de hardware como puede ser ejecutarlo de manera local o remota usando herramientas en la nube como AWS. En este caso se optó por este último mediante el uso de Google Colaboratory. Esta herramienta provee hardware en la nube de manera gratuita bajo limitaciones de tiempo de uso.

7.2. Métodos de solución

7.2.1. Algoritmo de Clarke y Wright

Se implementará este algoritmo para resolver tanto el VRP como el CVRP. Se hará uso del lenguaje de programación Python y de la libreria NumPy del mismo referenciada en la sección anterior.

Se implementará de dos formas diferentes dependiendo del tipo de problema que se este tratando.

VRP

En este caso se proseguirá con el algoritmo de Clarke y Wright implementando el cálculo del ahorro que este usa como sigue:

Algoritmo 1 Cálculo del ahorro

1: **función** CALCULO_AHORRO(M, $ruta$, $tomados$)
2: $costo\_ruta \leftarrow M_{0 \rightarrow ruta_0} + M_{ruta_n \rightarrow 0}$
3: $d\_ahorros \leftarrow \{\}$ (ahorro por nodo escogido)
4: **para** $nodo \leftarrow ruta_0, \dots ruta_n$ **hacer**
5: $costo\_ruta \leftarrow costo\_ruta + M_{nodo \rightarrow siguiente\_nodo}$
6: **fin para**
7: **para** $nodo \leftarrow 1, \dots \#nodos$ **hacer**
8: **si** $nodo \neq 0 y no do no esta en ruta ni tomados$ **entonces**
9: $d\_ahorros \leftarrow costo\_ruta + M_{ruta_0 \rightarrow nodo} - M_{nodo \rightarrow ruta_n}$
10: **fin si**
11: **fin para**
12: **retornar** $d\_ahorros$
13: **fin función**

M: matriz con los costes entre nodos
$ruta$: ruta a la que se le agregará un nodo
$tomados$: lista de nodos que estan tomados
$M_{i \rightarrow j}$: costo de ir de i a j
$ruta_i$: $i-$esimo nodo de la ruta
n: número de nodos en la ruta

Ahora se mostrará el Algoritmo de Clarke y Wright usando el cálculo del ahorro previamente definido.

Algoritmo 2 Algoritmo de Clarke y Wright

```
1: función CLARKE_WRIGHT(M, #vehiculos)
2:     rutas ← {}   (rutas escogidas)
3:     tomados ← [0]
4:     costos_iniciales ← ordenarCostos(M_{0→i})
5:     para ruta ← 0, ... #vehiculos hacer
6:         rutas[ruta].agregar(costos_iniciales_{ruta})
7:         tomados.agregar(costos_iniciales_{ruta})
8:     fin para
9:     mientras tomados ≠ #nodos_totales hacer
10:        para ruta ← 1, ... #nodos hacer
11:            ahorros ← Calculo_Ahorro(M, rutas[ruta], tomados))
12:            si #ahorros ≠ 0 entonces
13:                maximo_ahorros ← argmax(ahorros)
14:                rutas[ruta].agregar(maximo_ahorro)
15:                tomados.agregar(maximo_ahorro)
16:            fin si
17:        fin para
18:    fin mientras
19:    retornar rutas
20: fin función
```

M: matriz con los costes entre nodos
$tomados$: lista de nodos que estan tomados
$rutas_i$: $i-$esima ruta
$M_{i \to j}$: costo de ir de i a j
$\#nodos\_totales$: número de nodos totales a recorrer

CVRP

Se usará el mismo algoritmo del VRP, pero restringiendo la inserción de un nuevo nodo en la ruta al limite de capacidad del vehículo que la este recorriendo.

7.2.2. Redes Neuronales

Se utilizará 3 tipos de Redes Neuronales para la resolución del VRP y CVRP. Se creará una versión de cada una de estas para cada tipo de problema. Las Redes Neuronales a usarse son:

- Red Neuronal Feed-Forward (FNN)
- Red Neuronal Convolucional (CNN)

- Red Neuronal Convolucional basada en Grafos (GCN)

7.3. Diseño de los modelos de Redes Neuronales

7.3.1. FNN

Para este tipo de Red Neuronal se consideró como entrada la misma cantidad de neuronas que de elementos en la matriz de adyacencia que representa al problema concatenado con un vector que representa la cantidad de vehículos en el caso de VRP, y concatenado además con el vector de capacidades y la matriz con las demandas para el CVRP. Y se considera cuatro capas ocultas. La primera de 700, la segunda de 1400, la tercera de 1400 y la cuarta de 700 neuronas en el caso del VRP. En el caso de CVRP, se tienen 4 capas como el de VRP, pero con 1000 en la primera capa, 2000 en la segunda y tercera, y 1000 neuronas en la cuarta. Se utilizan más neuronas en este segundo caso pues posee más neuronas de entrada. Para la salida de VRP y CVRP se consideran 300 neuronas pues solo se esta considerando los elementos ubicados sobre la diagonal de la matriz de adyacencia que muestra las rutas que es de 25 filas y 25 columnas. Todas las capas tienen como función de activación la ReLU.

7.3.2. CNN

Se utilizarán 3 bloques convolucionales: el primer bloque tiene 1 canal de entrada, 64 de salida, un kernel de 3×3, un padding de 1 y un stride de 1, esto para conservar las dimensiones y como activación una capa ReLU; el segundo bloque tiene una capa convolucional que contiene 64 canales de entrada, 128 de salida y con la configuración de padding y stride, y capa Max Pool con stride y padding igual a 1; y activación ReLU anterior; por último, un bloque con una capa de 128 canales de entrada, 256 de salida, padding, stride, y las otras capas igual al primer bloque. La salida de todos estos bloques entra en una capa lineal con la cantidad de neuronas de entrada igual a la salida aplanada en 1 dimensión y de salida 300 neuronas.

7.3.3. GNN

En el caso de esta arquitectura se considera la variante de GNN conocida como GCN. Esta variante implementa la operación de convolución sobre un grafo que es lo que se explicó que se necesitaba en la metodología. Se considera 5 capas convolucionales GCN para el encoder, con capa de activación ReLU para cada una. Para el decoder se multiplica de acuerdo al índice de las aristas y se suma las columnas dando un vector. Las capas convolucionales tienen valores para los canales de entrada de 2 en el caso del VRP y 6 en el del CVRP, 3 y 6 para las capas intermedias y final para el VRP y CVRP respectivamente. Para los de salida es el mismo al de entrada para las capas ocultas para estas mismas e inicial y para la capa final de ambos tipos de problemas es 1.

7.4. Tratamiento del dataset

El dataset utilizado por cada modelo tanto de VRP como CVRP son generados con 25 nodos, distancias, demandas y capacidades aleatorias. Por simplicidad se considera que todos los vehículos de la flota cuentan con la misma capacidad. Luego se generan las soluciones o targets mediante unos Scripts de Python que hacen uso de OR-Tools para obtener la solución. Además se utilizó NumPy para guardan en archivos $.npy$ que son los que guardan los arreglos de NumPy. Se generaron en total 10000 instancias para VRP y otras 10000 más para CVRP.

Como referencia se muestra el grafo solución para un problema de VRP que se muestra en la documentación de OR-Tools, este se construyó como grafo dirigido gracias a la librería NetworkX que permite construirlo a partir de las listas de adyacencia. Se muestra usando Matplotlib.

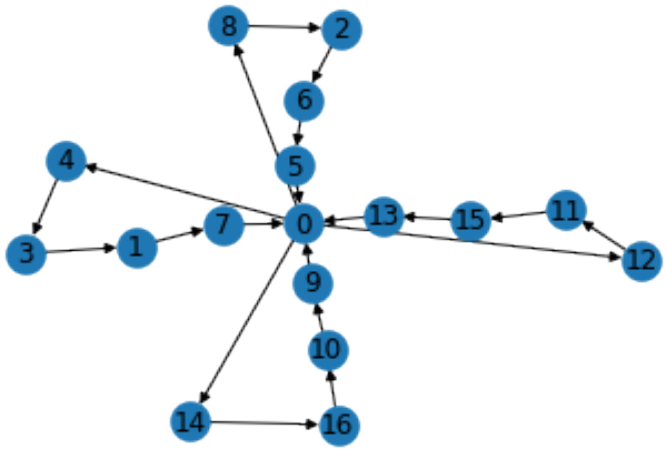

FIGURA 7.1: Grafo solución de un problema de CVRP

7.4.1. FNN

Para el VRP se utiliza una concatenación de la matriz de adyacencia y el vector con el número de vehículos. En el caso del CVRP además se concatena con el vector de demandas y la matriz de capacidades. Los targets, que son la matriz de adyacencia del grafo resultante, son aplanados los elementos que se ubican por encima de la diagonal en el orden que aparecen para que sean un vector de 300 dimensiones. También se aplana la entrada para que pueda ser enviada a la capa lineal que cuenta con tantas neuronas como elementos en las entrada. Las entradas se dividen sobre 1500 para que sus elementos se sitúen en el rango $[0, 1]$. Mientras que las salidas se dividen sobre 8 para el mismo fin.

7.4.2. CNN

Los datos de entrada son una concatenación de vectores columna y matrices. En el caso del VRP esta compuesto de la matriz de adyacencia que representa los costes entre cada par de nodos, con 0s en la diagonal, y un vector que representa la cantidad de vehículos. Para el CVRP es igual, con la diferencia que se le añade una matriz que representa las demandas por nodo de dimensión $25 \times$ #nodos. Estos dato son normalizados con media 0 y desviación estándar 1. Los targets son los elementos que se posicionan sobre la diagonal de la matriz

de adyacencia que representa al grafo solución del problema divididos sobre 8 para estar en el rango $[0, 1]$.

7.4.3. GCN

Para esta arquitectura fue necesario un tratamiento especial de los datos obtenidos pues estos deben ser del tipo Data que es un objeto de PyTorch Geometric. Con este fin es necesario crear listas de aristas conformadas por el nodo de entrada, salida y el peso. Los nodos tienen como características asociadas al índice de cada uno y el número de vehículos en el caso del VRP y para el CVRP se tiene como características las demandas de ese nodo y capacidades de los vehículos. Esto se convierte en un objeto de NetworkX para luego crear el ibjeto Data. El target es un vector de 0s y 1s donde 1 representa que hay una arista en esa posición y 0 lo contrario.

7.5. Implementación de los modelos

7.5.1. FNN

Para la implementación de esta arquitectura se utiliza la función Linear del módulo nn de PyTorch que contiene los tipos de capa especificados en el diseño de esta y la función de activación ReLU. Esto se estructura en una clase que hereda la superclase torch.nn.module. Los bloques son creados con la función Sequential del modulo nn de PyTorch.

7.5.2. CNN

Esta arquitectura se implementó gracias al módulo nn de PyTorch usando las funciones Linear para las capas lineales, para la capa convolucional se utiliza Conv2D con stride y padding igual a 1 para conservar las dimensiones de la entrada. y para la capa de pooling se usa MaxPool2D, que es el max pooling con stride y padding con valor 1, por las mismas razones que para la capa convolucional. Para aplanar la salida de estos bloques se utiliza el

método view que posee cada tensor con 2 valores: la primera dimensión de la salida del bloque y -1 para aplanar las otras en una dimensión. Para la función de activación se utiliza la función ReLU del modulo nn. Finalmente, esto se estructura en una subclase de module perteneciente al modulo nn de PyTorch.

7.5.3. GCN

Se implementó usando PyTorch Geometric que contiene el módulo nn con el funcional GCNConv que es la capa de la Red Convolucional basada en Grafos. También se utilizó el funcional ReLU del módulo nn de PyTorch que es la función de activación aplicada a la salida de cada capa GCN, excepto la última. Este par se utiliza en 4 bloques creados gracias al objeto Sequential de PyTorch Geometric. Al final se utiliza solo una capa GCN. También se define los canales de entrada de la primera capa GCN como la cantidad de características de los nodos del grafo que se toma como entrada. Luego los canales de salida de la primera capa, los de entrada y salida de las capas GCN intermedias, y los de entrada de la última capa se definen de acuerdo al tipo del problema; para el caso del VRP es 3 y del CVRP es 6. Se eligen estos valores de acuerdo al número de características de nodos de cada uno, 2 en VRP y 6 en CVRP. Todo esto se estructura en una subclase de Module del modulo nn de PyTorch. Para el paso hacia adelante se pasa las características de los nodos, el peso de las aristas y los índices de las aristas a cada capa GCN. Esta parte se encuentra en la función encode. Esta salida junto a los índices de las aristas se pasan al decodificador que nos da las aristas que se seleccionaron. Por último la función decode$_a llseutilizaparaconseguirlalistadearistasausar.$

7.6. Entrenamiento

Para el entrenamiento se utiliza la técnica de dividir el dataset en minibatches para aumentar la rapidez del entrenamiento.

A continuación se muestra el optimizador a usar para este proceso.

7.6.1. Optimizador Adam

Este optimizador se encuentra en el modulo optim de PyTorch. Según la documentación del mismo [18] se define como se puede ver en el siguiente algoritmo 3.

A su vez para mejorar la convergencia del mismo se utilizó una técnica conocida como learning rate schedulling. Esta consiste en variar los valores del learning rate para conseguir una rápida bajada al inicio y luego un más lenta por la función que se desea optimizar. De esta forma se evita la divergencia por escoger un learning rate alto, pero se gana su rapidez. A la vez se consigue la precisión de un learning rate bajo, pero sin su lentitud.

Para este fin se utilizó el método de schedulling step learning rate que consiste en multiplicar el learning rate después de un número de iteraciones dado. En este caso se multiplicará por 0.2 cada 5 iteraciones comenzando con un learning rate de 0.1. La función que permite realizar esto se encuentra en el submódulo lr_scheduler del módulo optim de PyTorch. Aquí se puede ver la documentación de esta función [19].

A continuación se muestra la función de perdida a usar durante el entrenamiento.

7.6.2. Función de perdida Divergencia de Kullback-Leibler

Para el caso del FNN, puesto que la entrada es un vector y la salida es otro vector, se busca calcular que tan diferentes son estos vectores. Eso es posible gracias a la divergencia de Kullback-Leibler. Esta se calcula como sigue:

$$L(y_{pred}, y_{true}) = y_{true} \cdot \log\left(\frac{y_{pred}}{y_{true}}\right) = y_{true} \cdot (\log(y_{true}) - \log(y_{pred})) \tag{7.1}$$

Esta función se encuentra en el modulo nn de PyTorch. Según la documentación [20] es necesario agregar el parámetro reduction con el valor batchmean para obtener el valor real de esta función. También se agregó el

Algoritmo 3 Algoritmo Adam

función ADAM(γ, β_1, β_2, θ_0, $f(\theta)$, λ, $amsgrad$, $maximize$)
  $m_0 \leftarrow 0$ (first moment)
  $v_0 \leftarrow 0$ (second moment)
  $\hat{v}_0^{max} \leftarrow 0$
  para $t \leftarrow 1, \ldots$ **hacer**
    si $maximize$ **entonces**
      $g_t \leftarrow -\nabla_\theta f_t(\theta_{t-1})$
    sino
      $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
    fin si
    si $\lambda \neq 0$ **entonces**
      $g_t \leftarrow g_t + \lambda\theta_{t-1}$
    fin si
    $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
    $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
    $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1}$
    $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2}$
    si amsgrad **entonces**
      $\hat{v}_t^{max} \leftarrow \max(\hat{v}_t^{max}, \hat{v}_t)$
      $\theta_t \leftarrow \theta_{t-1} - \frac{\gamma \hat{m}_t}{\sqrt{\hat{v}_t^{max}} + \epsilon}$
    sino
      $\theta_t \leftarrow \theta_{t-1} - \frac{\gamma \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$
    fin si
  fin para
  retornar θ_t
fin función

γ: learning rate
β_1, β_2: betas
θ_0: parámetros
$f(\theta)$: función objetivo
λ: weight decay

parámetro log_target con el valor True. Con este último parámetro la fórmula cambia a:

$$L(y_{pred}, y_{true}) = e^{y_{true}} \cdot (y_{true} - y_{pred}) \tag{7.2}$$

Se considera que y_{pred} esta en el espacio logarítmico. Para conseguir esto se utiliza la función torch.nn.functional.log_softmax.

7.7. Evaluación

Para la evaluación de la capacidad de los modelos se consideró un dataset compuesto de 10 000 problemas con su solución de VRP y CVRP. El tratamiento de este dataset varia con el modelo que se usa y la cantidad de épocas dependerá del modelo que se utilice.

También se resolveran estas instancias con el Algoritmo de Clarke y Wright. Esto se realiza con el fin de comparar el resultado dado por los modelos de Redes Neuronales y este.

7.7.1. Métricas

La métrica que se utiliza para comparar estos modelos es la media de la función de perdida en cada batch para saber que tan rápido convergen. Además de “Explained Variance Score", que verifica que tan parecidos son 2 vectores. Esta métrica suele usarse para problemas de regresión, a pesar de ello, también se usará para el caso de la GNN para comparar con los otros tipos de de Red Neuronal.

Se comparara los costes de las rutas obtenidas por los modelos de Redes Neuronales y el Algoritmo de Clarke y Wright.

Capítulo 8

Presentación de Resultados

En este apartado se presentan los resultados obtenidos de la comparativa de modelos. En este sentido se mostrarán las gráficas de la perdida y el "Explained Variance Score.[en] el caso de las Redes Neuronales y los costos para el Algoritmo de Clarke y Wright.

8.1. VRP

8.1.1. Algoritmo de Clarke y Wright

Se muestra los costos obtenidos para las primeras 20 instancias.

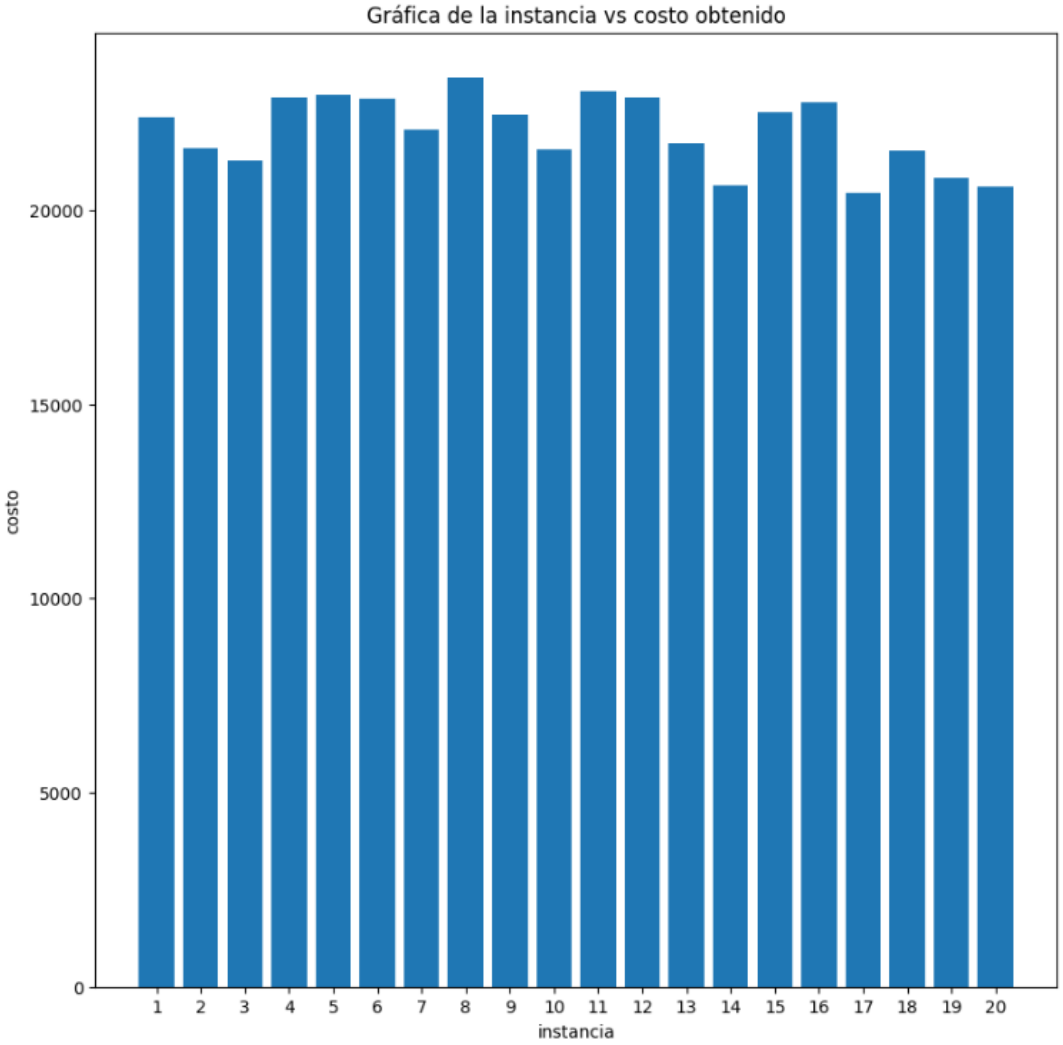

FIGURA 8.1: Resultados del Algoritmo de Clarke y Wright para el VRP

Nota: Esta gráfica se realiza solo para 20 instancias como ejemplo para mostrar los costos obtenidos.

8.1.2. Red Neuronal Feed-Forward

Primero se muestra como base los resultados obtenidos por la Red Feed-Forward como base de comparación.

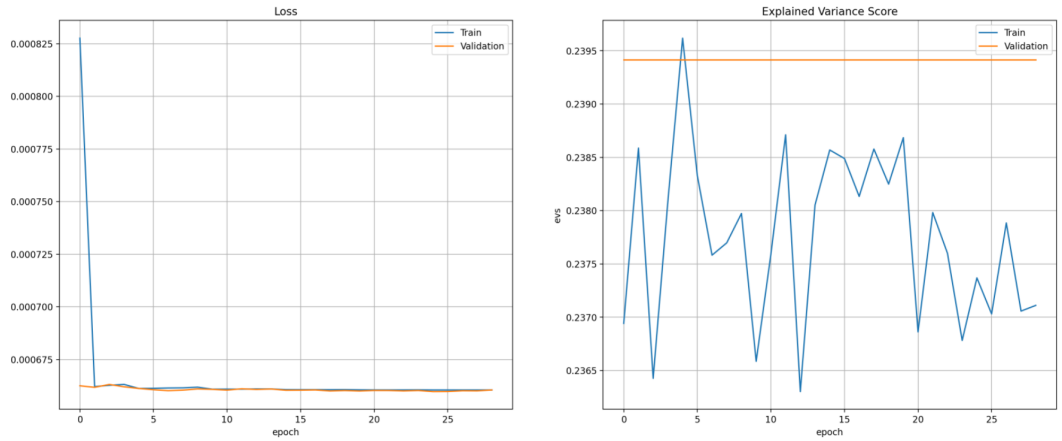

FIGURA 8.2: Resultados de la Red Feed-Forward para el VRP

Nota: Esta gráfica se realiza a partir de la segunda época para que se pueda apreciar la bajada de las funciones, pues las épocas anteriores dan valores de 10^6 los valores posteriores.

8.1.3. CNN

Se muestra los resultados de la CNN.

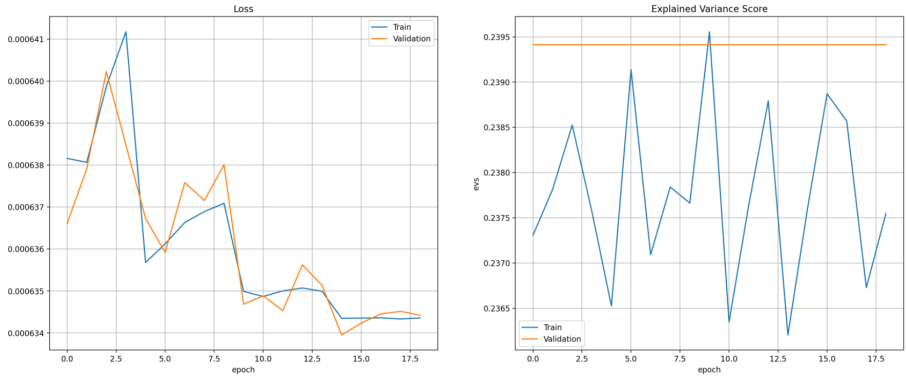

FIGURA 8.3: Resultados de la Red Convolucional para el VRP

Nota: Esta gráfica se realiza a partir de la segunda época para que se pueda apreciar la bajada de las funciones, pues las épocas anteriores dan valores de 10^7 los valores posteriores.

8.1.4. GCN

Se muestran las gráficas de la GCN.

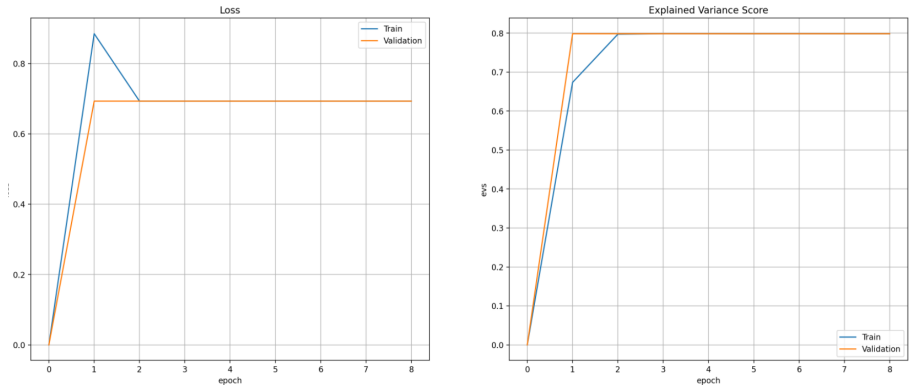

FIGURA 8.4: Resultados de la GCN para el VRP

Nota: Esta gráfica es de todas las épocas pues la diferencia de los valores no es excesivamente alto como en los casos anteriores.

Se muestra las predicciones hechas por la GCN

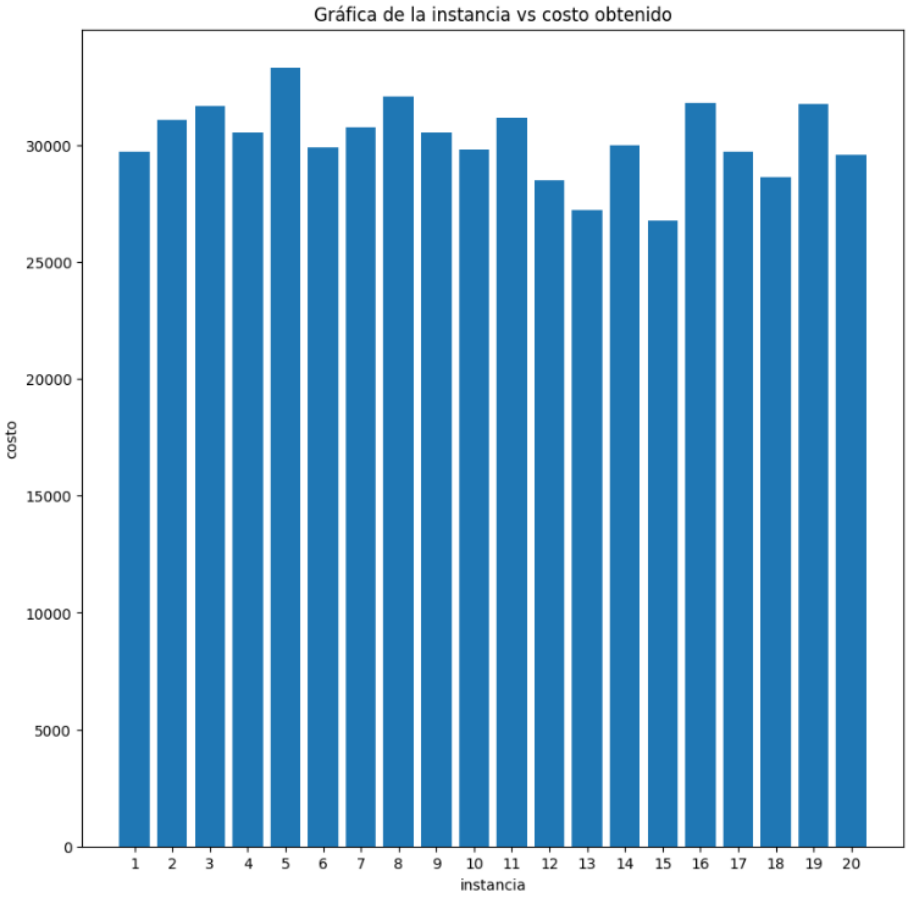

FIGURA 8.5: Predicciones de la GCN para el VRP

Nota: Esta gráfica es de las primeras 20 instancias, solo se gráfica las predicciones de la GCN pues es la que mejores resultados obtuvo.

8.1.5. Tabla comparativa de Redes Neuronales

Se muestra una comparativa entre los valores obtenidos por los diferentes tipos de Red Neuronal usados.

Modelo	Explained Variance Score	
	Entrenamiento	Validación
Red Neuronal Feed-Forward	0.238	0.239
CNN	0.239	0.239
GNN	0.798	0.798

CUADRO 8.1: Comparativa problema VRP

8.1.6. Comparativa del Algoritmo de Clarke y Wright y GCN

Se muestra una comparativa entre el Algoritmo de Clarke y Wright con el modelo de GCN, el cual es el modelo de Red Neuronal con mejores resultados. Esto se realizó mediante la obtención de la media de diferencia en porcentaje relativa al Algoritmo de Clarke y Wright cada 100 instancias, para luego ser graficado.

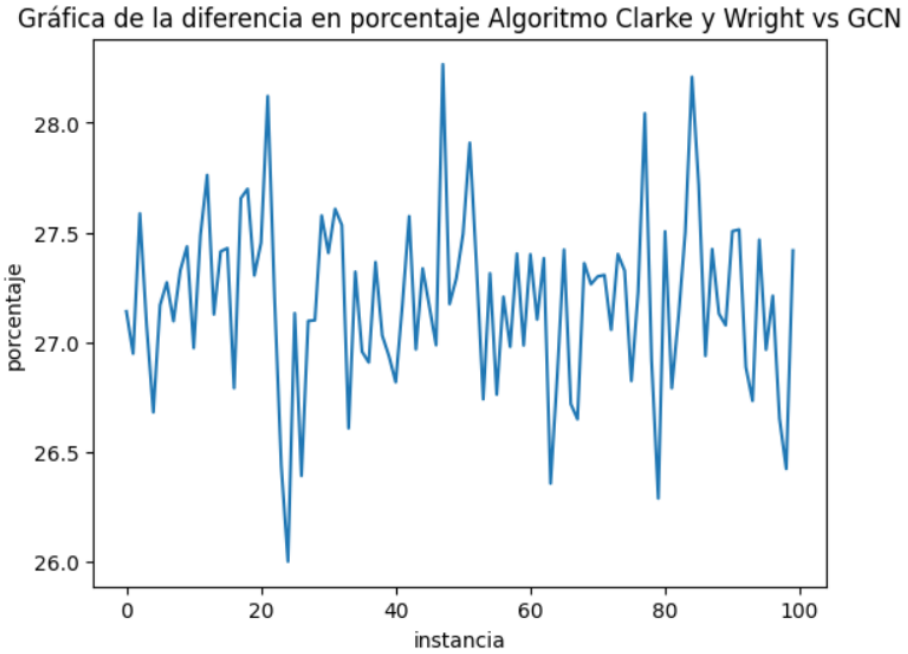

FIGURA 8.6: Comparativa del Algoritmo de Clarke y Wright y GCN para el VRP

8.2. CVRP

8.2.1. Algoritmo de Clarke y Wright

Se muestra los costos obtenidos para las primeras 20 instancias y la cantidad de nodos usada.

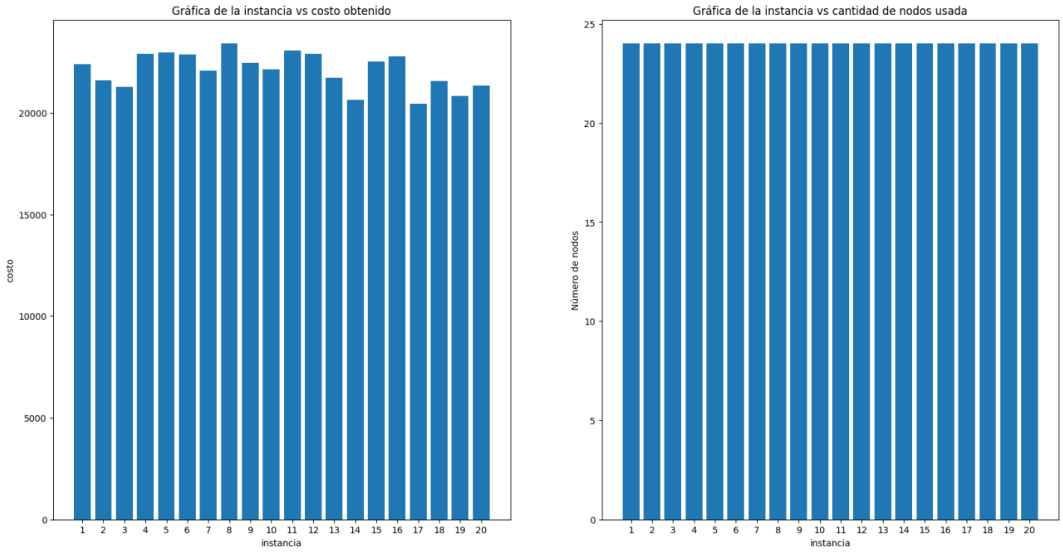

FIGURA 8.7: Resultados del Algoritmo de Clarke y Wright para el CVRP

Nota: Esta gráfica se realiza solo para 20 instancias como ejemplo para mostrar los costos obtenidos.

8.2.2. Red Neuronal Feed-Forward

Primero se muestra como base los resultados obtenidos por la Red Feed-Forward como base de comparación.

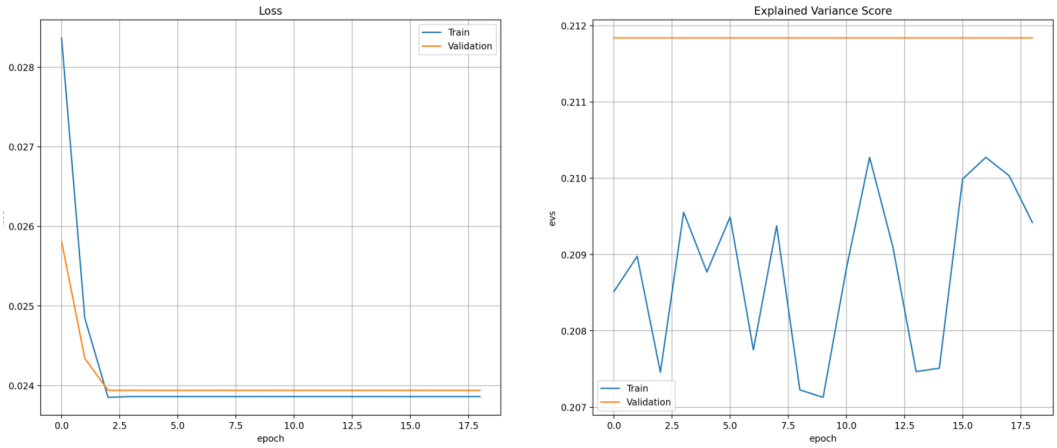

FIGURA 8.8: Resultados de la Red Feed-Forward para el CVRP

Nota: Esta gráfica se realiza a partir de la segunda época para que se pueda apreciar la bajada de las funciones, pues las épocas anteriores dan valores de 10^6 los valores posteriores.

8.2.3. CNN

Se muestra los resultados de la CNN.

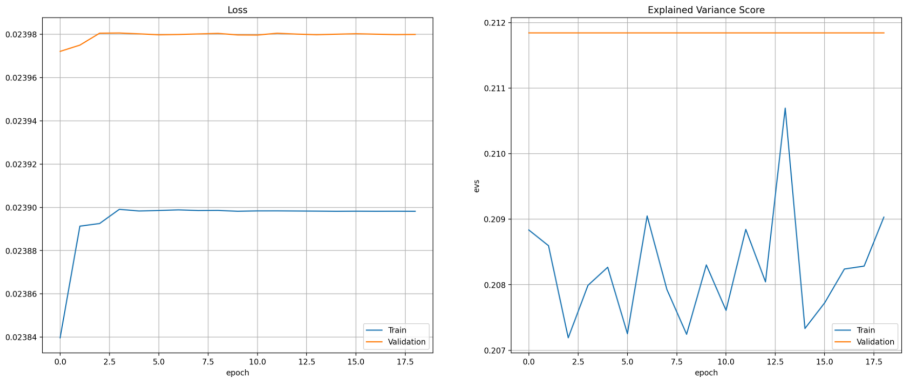

FIGURA 8.9: Resultados de la Red Convolucional para el CVRP

Nota: Esta gráfica se realiza a partir de la segunda época para que se pueda apreciar la bajada de las funciones, pues las épocas anteriores dan valores de 10^7 los valores posteriores.

8.2.4. GCN

Se muestran las gráficas de la GCN.

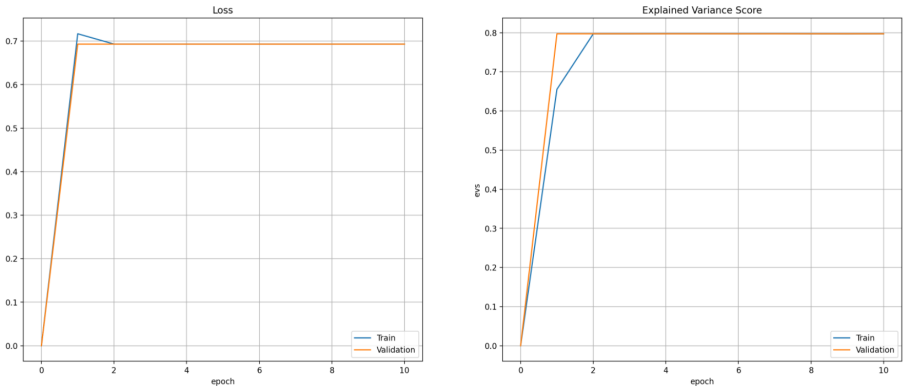

FIGURA 8.10: Resultados de la GCN para el CVRP

Nota: Esta gráfica es de todas las épocas pues la diferencia de los valores no es excesivamente alto como en los casos anteriores.

Se muestra las predicciones hechas por la GCN

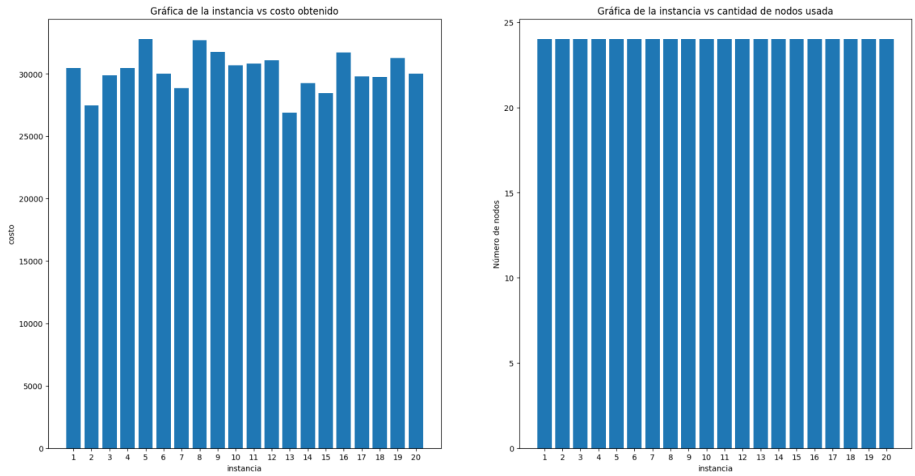

FIGURA 8.11: Predicciones de la GCN para el CVRP

Nota: Esta gráfica es de las primeras 20 instancias, solo se gráfica las predicciones de la GCN pues es la que mejores resultados obtuvo.

8.2.5. Tabla comparativa de Redes Neuronales

Se muestra una comparativa entre los valores obtenidos por los diferentes tipos de Red Neuronal usados.

Modelo	Explained Variance Score	
	Entrenamiento	Validación
Red Neuronal Feed-Forward	0.210	0.211
CNN	0.210	0.211
GNN	0.797	0.797

CUADRO 8.2: Comparativa problema CVRP

8.2.6. Comparativa del Algoritmo de Clarke y Wright y GCN

Se muestra una comparativa entre el Algoritmo de Clarke y Wright con el modelo de GCN, el cual es el modelo de Red Neuronal con mejores resultados. Esto se realizó mediante la obtención de la media de diferencia en porcentaje relativa al Algoritmo de Clarke y Wright cada 100 instancias, para luego ser graficado.

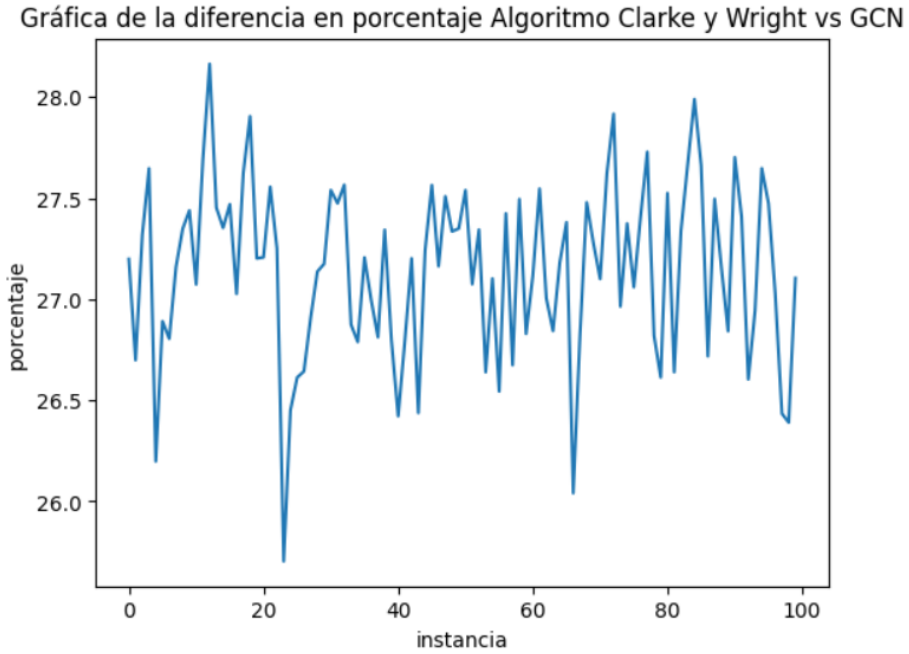

FIGURA 8.12: Comparativa del Algoritmo de Clarke y Wright y GCN para el CVRP

8.3. Resultados

- Se puede notar que en el caso del VRP básico la GCN obtiene cerca del 80 %, mientras que en los casos de la Red Feed-Forward y CNN es de alrededor del 22 %. Estos en la métrica “Explained Variance Score".
- En el caso del CVRP, la GCN consigue un 80 % en la métrica “Explained Variance Score", mientras que la Red Feed-Forward y la CNN se encuentran en torno al 22 %.
- En ambos tipos de problemas la GCN converge más rápido al resultado deseado que la Red Feed-Forward y la CNN.
- La Red Feed-Forward y CNN necesitaron entre 20 y 30 épocas para alcanzar sus mejores resultados, mientras que la GCN necesita entre 8 y 10 épocas para alcanzar los suyos.
- La GCN obtuvo peores resultados que el algoritmo de Clarke y Wright aunque toma menor tiempo en conseguirlos.

- La GCN obtuvo resultados alrededor del 25 % más costosos en comparación al Algoritmo de Clarke y Wright.

Capítulo 9

Conclusiones y Trabajos Futuros

En este apartado se muestran las conclusiones después de haber realizado la experimentación en los modelos de Red Neuronal: Red Feed-Forward, CNN y GCN como con el Algoritmo de Clarke y Wright. También se muestran los posibles trabajos futuros a realizarse a partir de lo obtenido en este trabajo.

9.1. Conclusiones

- El modelo de la GCN consigue resultados considerablemente mejores que la CNN y Red Feed-Forward en ambos tipos de problemas, VRP y CVRP.
- La GCN converge en menos épocas que los modelos Red Feed-Forward y CNN.
- Todos los modelos consiguen resultados parecidos en el VRP básico y en el CVRP respecto a estos mismos.
- Todos los modelos consiguen realizar predicciones en un tiempo considerablemente menor al de la herramienta usada para crear los datasets que es OR-Tools que utiliza métodos clásicos para resolver el problema.
- El Algoritmo de Clarke y Wright obtuvo mejores resultados que las Redes Neuronales, llegando a una media de 25 % menos costo.
- La obtención de datasets con más de 1000 instancias es complicado, la mayoria que se encuentra es de 100 a 200 instancias.

Ahora se presentan los trabajos futuros a realizarse a partir de lo obtenido como conclusión en la presente tesis.

9.2. Trabajos Futuros

- Se propone resolver problemas de VRP y CVRP con más de 25 nodos.
- Se propone utilizar otros tipos de GNN para realizar comparativas.
- Se propone comparar una arquitectura que pueda combinar el uso de las GCNs y CNNs.
- Se propone comparar una arquitectura que pueda combinar el uso de las GCNs y algoritmos clásicos, como el Algoritmo de Clarke y Wright.

Bibliografía

[1] Paolo Toth and Daniele Vigo. *1. An Overview of Vehicle Routing Problems*, pages 1–26.

[2] Vishal Kumar. *An Investigation Into Graph Neural Networks*. PhD thesis, Trinity College Dublin, Ireland, 2020.

[3] Shadman Sakib, Nazib Ahmed, Ahmed Jawad Kabir, and Hridon Ahmed. An overview of convolutional neural network: its architecture and applications. 2019.

[4] Benjamin Sanchez-Lengeling, Emily Reif, Adam Pearce, and Alexander B Wiltschko. A gentle introduction to graph neural networks. *Distill*, 6(9):e33, 2021.

[5] Ameya Daigavane, Balaraman Ravindran, and Gaurav Aggarwal. Understanding convolutions on graphs. *Distill*, 6(9):e32, 2021.

[6] Fredrik Hagström et al. Finding solutions to the vehicle routing problem using a graph neural network. 2022.

[7] Python Software Foundation. Applications for python. `https://www.python.org/about/apps/`, 2022. Accedido 06-07-2022.

[8] PyTorch Team. Pytorch. `https://pytorch.org/`, 2022. Accedido 06-07-2022.

[9] PyG Team. Pyg documentation. `https://pytorch-geometric.readthedocs.io/en/latest/`, 2022. Accedido 06-07-2022.

[10] The Matplotlib development team. Matplotlib. `https://matplotlib.org/`, 2022. Accedido 06-07-2022.

[11] PyTorch Core Team. Torchvision. `https://pytorch.org/vision/stable/index.html`, 2022. Accedido 06-07-2022.

[12] NumPy Developers. Numpy. `https://numpy.org/`, 2022. Accedido 06-07-2022.

[13] NetworkX Developers. Networkx. `https://networkx.org/`, 2022. Accedido 06-07-2022.

[14] Google LLC. About or-tools. `https://developers.google.com/optimization/introduction/overview`, 2022. Accedido 06-07-2022.

[15] Andreas Mueller. scikit-learn. `https://scikit-learn.org/stable/`, 2022. Accedido 06-07-2022.

[16] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[17] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[18] PyTorch Team. Adam. `https://pytorch.org/docs/stable/generated/torch.optim.Adam.html`, 2022. Accedido 07-07-2022.

[19] PyTorch Team. Steplr. `https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.StepLR.html`, 2022. Accedido 07-07-2022.

[20] PyTorch Team. Kldivloss. `https://pytorch.org/docs/stable/generated/torch.nn.KLDivLoss.html`, 2022. Accedido 07-07-2022.

Apéndice A

Implementación realizada

Para visualizar el código, los datasets y los modelos entrenados puede referirse al siguiente repositorio de github:

Click aquí para ver el Repositorio de la tesis