ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Un método de redes neuronales para el problema del vendedor viajero

PROYECTO INTEGRADOR

Previo a la obtención del Título de:

Matemático

Presentado por:

Mario Raúl Ruiz Córdova

David Andrés Montalván Poppe

GUAYAQUIL - ECUADOR

Año: 2023

DEDICATORIA

Este trabajo está dedicado a todos los caídos en el camino de esta carrera.

Mario Ruiz C.

DEDICATORIA

A nadie en especial.

David Montalván P.

AGRADECIMIENTOS

Agradezco a mi Tutor Domingo Quiroz por la oportunidad de trabajar en este proyecto, a la Profesora Elimar Marchan por su gran paciencia y guía en la elaboración de este trabajo, a todos mis compañeros por ser dignos de admiración y a mi familia por el soporte físico y emocional.

Mario Ruiz C.

AGRADECIMIENTOS

Agradezco a Taylor Swift, Olivia Rodrigo, Lady Gaga, The Beatles, Bullet For My Valentine y Gojira quienes me acompañaron On Repeat durante todo este proceso. Agradezco a Ángel Guale con quien pude estudiar matemática de verdad este último año y a todos los amigos que hice en EMALCA; por ustedes no abandoné. Agradezco a mis estudiantes de IRM y los demás de la carrera quienes me enseñaron que sí hay futuro para la matemática del país. Finalmente, agradezco a NoEm por carrearme en este trabajo.

David Montalván P.

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; *Mario Raúl Ruiz Córdova, David Andrés Montalván Poppe,* y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual”

Mario Ruiz C.

David Montalván P.

EVALUADORES

Firmado electrónicamente por:
LUZ ELIMAR MARCHAN MENDOZA

Luz Elimar Marchan Mendoza

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
DOMINGO ALBERTO QUIROZ RODRIGUEZ

Domingo Alberto Quiroz Rodríguez

TUTOR

RESUMEN

En este trabajo, se resuelve un problema del vendedor viajero (TSP) usando un método de redes neuronales y aprendizaje reforzado combinado con una heurística simple para mostrar cómo este método puede ser empleado en una instancia real. Para el desarrollo, se utilizó una aplicación del modelo Pointer-network para redes neuronales en TensorFlow versión 2.11.0. El entrenamiento se realizó con datos generados de manera ficticia con 50 ciudades y la heurísitca 2opt para hallar las soluciones, además, fue probado con 100 ciudades. Finalmente, se procesaron los datos de la instancia real para que puedan ser aplicados al modelo. En la fase de prueba se encontró que con pocas iteraciones (5000), el modelo depende mucho de la heurística escogida. También se resolvió el problema usando software comercial para comparar los resultados obtenidos. Se obtuvo los mismos resultados para la solución con el método planteado y la solución por medio de software comercial. En esta tesis, se muestra como una instancia real del TSP puede ser resuelta por medio de aprendizaje reforzado y la manera en que pueden combinarse heurísticas simples y métodos de redes neuronales para obtener soluciones competitivas de problemas de optimización.

Palabras Clave: TSP, optimización, aprendizaje reforzado, redes neuronales.

ABSTRACT

In this work, the traveling salesman problem (TSP) is solved using neural networks and reinforcement learning combined with a simple heuristic to show how this method can be utilized in a real case. For the development, the model Pointer-network method was used for neural networks in TensorFlow version 2.11.0. The network was trained with artificially generated data with 50 nodes and the heuristic 2opt to find the solution. In addition, the network was then tested with a 100 nodes. Finally, the real data was processed and applied to the model. During testing phase, it was found that the proposed method depended heavily on the chosen heuristic when using few iterations (5000). The real problem was also solved using commercial software and compared the results. Both methods obtained the same solution. In this thesis, we show how a real TSP can be solved with reinforcement learning and how simple heuristics and neural networks can be combined to get competitive solutions of optimization problems.

Keywords: *TSP, optimization, reinforcement learning, neural networks.*

ÍNDICE GENERAL

ABREVIATURAS

ESPOL	Escuela Superior Politécnica del Litoral
TSP	Traveling salesman problem
RL	Reinforcement learning
MDP	Markov decision process

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. INTRODUCCIÓN

1.1 Descripción del problema

EXPOTUNA S.A es una empresa dedicada a la exportación de alimentos marinos que, desde sus inicios, no cuenta con un modelo logístico que le permita encontrar el trayecto que minimice el tiempo y costo para recorrer distintos puntos de distribución en Guayaquil. Hasta el momento ha encargado a los conductores encontrar la ruta a seguir, quienes lo han hecho de manera empírica.

1.2 Justificación del problema

La decisión de no actuar frente a este problema de enrutamiento genera costos adicionales que pueden ser evitados tomando una ruta más óptima. La solución del problema del vendedor viajero es esencial para disminuir gastos innecesarios en una empresa, sin embargo, muchas de ellas suelen tener dificultades para implementar alguna solución según Khalil et al. (2017), es por esto que los resultados de este trabajo contribuirán a obtener métodos más sencillos de aplicar al estar enfocados en datos y no en otros parámetros como los demás métodos heurísticos.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar un plan de ruteo para la distribución eficiente de una empresa comercializadora de productos marinos.

1.3.2 Objetivos Específicos

- Compilar la base de datos del cliente para la identificación de los puntos de distribución y su respectiva ubicación.

- Identificar el modelo que se adapta al problema para implementar el método pointer network.
- Elaborar una solución al problema usando el método pointer network para redes neuronales.

j

1.4 Marco teórico

El problema del vendedor viajero (TSP por sus siglas en Inglés) es un problema que es formulado de la siguiente manera: *dada una lista de ciudades y las distancias entre cada par de ellas ¿Cuál es la ruta más corta que visita cada ciudad una única vez y regresa a la ciudad de origen?* (Quispe et al., 2021).

1.4.1 Historia y estado del arte del TSP

La primera vez que se registra el problema del vendedor viajero fue en 1832 en un folleto donde se plantea y se dan algunos ejemplos de rutas que pasan por Alemania y Suiza, no obstante, no tiene un carácter matemático (Voigt, 1831). El TSP vuelve a aparecer de una manera más clara en el *juego Icosian*, hasta que en 1930 se obtiene una formulación matemática por Merril M. Flood intentando resolver un problema de ruteo.

La primera publicación donde el título contiene ”el problema del vendedor viajero” se realizó en 1949 tratando el *Icosian*. En la siguiente década, George Dantzig, Delbert Ray Fulkerson y Selmer M. Johnson contribuyeron enormemente al formular el TSP como un problema de optimización lineal entera, además, logran desarrollar el método de planos de corte para solucionarlo (Lawler, 1985). El planteamiento de Dantzig y sus colaboradores fue el siguiente: Enumere las ciudades del 1 al n y defina:

$$x_{ij} = \begin{cases} 1 & \text{si el camino va de la ciudad } i \text{ a la ciudad } j \\ 0 & \text{en cualquier otro caso} \end{cases}$$

$$
\begin{aligned}
\text{minimizar } & \sum_{j=1}^{m} w_j x_j \\
\text{sujeto a } & \sum_{j:\, e_i \in S_j}^{m} x_j \geqslant 1, && i = 1, \ldots, n \\
& \sum_{j=1}^{m} w_j = 1, \\
& x_j \in \{0, 1\}, && j = 1, \ldots, m
\end{aligned}
$$

Después de esto, los estudios se enfocaron en hallar cotas para la solución óptima y en heurísticas para resolver el problema, por ejemplo, el estado del arte en 1976 era el algoritmo Christofides-Serdyukov el cual encuentra una solución que es a lo sumo 1.5 veces más larga que la solución óptima (van Bevern and Slugina, 2020).
Finalmente, en la década de 1990 Applegate, Bixby, Chvátal, y Cook desarrollan el algoritmo *Concorde* del cual se han obtenido algoritmos derivados que mantienen las mejores soluciones actuales para el TSP simétrico (Rego et al., 2011).

1.4.2 Conceptos de inteligencia artificial

Definición 1. *Un **par de entrenamiento** es un par ordenado $(\mathcal{I}, \mathcal{O})$ donde $\mathcal{I} = \{i_1, ..., i_n\}$ es una secuencia de entradas y $\mathcal{O} = \{o_1, ..., o_n\}$ una secuencia de salidas de tal manera que se espera que el modelo mapee $i_k \in \mathcal{I}$ al respectivo $o_k \in \mathcal{O}$.*

De esta manera, se puede explicar el aprendizaje reforzado como el aprendizaje que a partir de un par de entrenamiento, crea una función que mapee los datos de entrada a los datos de salida (Yadav et al., 2020).

Definición 2. *Un **proceso de decisión de Markov** (Wrobel, 1984) es una 4-tupla $(\mathcal{S}, \mathcal{A}, P_a, R_a)$ donde:*

- *$\mathcal{S}$ es un conjunto de estados llamado el espacio de estados*
- *$\mathcal{A}$ es el conjunto de acciones llamado el espacio de acciones.*
- *$P_a(s, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ es la probabilidad de que la acción a en el estado s, en el tiempo t llevará al estado s' en el tiempo $t + 1$.*

- $R_a(s, s')$ *es el incentivo intermedio o inmediato esperado recibido al cambiar del estado* s *a* s' *debido a la acción* a.

Es importante notar que el objetivo es encontrar un modo de escoger las acciones que resulten en el retorno más largo a partir de las trayectorias que surgen. El retorno se define como:

$$\mathcal{R} = r_{a_0}(s_0) + \gamma r_{a_1}(s_1) + \gamma^2 r_{a_2}(s_2) + ... \tag{1.1}$$

Donde $\gamma \in [0, 1)$ es un **factor de descuento**.

Definición 3. *Dado un conjunto* X*, denote por* $\mathcal{M}_1(X)$ *el conjunto de distribuciones de probabilidades sobre* X *y dado el entero* $t > 0$*, defina* $\mathcal{H}_t = (\mathcal{S} \times \mathcal{A})^{t-1} \times \mathcal{S}$ *(*$\mathcal{H}_0 = S$*) una* ***política*** *es una sucesión* $\pi = (\pi_t)_{t \geqslant 0}$ *donde cada* π_t *está definida como:*

$$\pi_t : \mathcal{H}_t \to \mathcal{M}_t(X)$$

Un política es ***sin memoria*** *si dada una función* $m : \mathcal{S} \to \mathcal{M}_1(X)$*, se puede escribir cada término* π_t *como:* $\pi_t(a|s_0, a_0, ..., s_{t-1}, a_{t-1}, s_t) = m(a|s_t)$ *y una* ***política es estocástica*** *cuando* π *puede ser representado como una familia de distribuciones de probabilidad condicional.*

Definición 4. *Defina* $v^\pi(s) = \mathbb{E}_s^\pi[\mathcal{R}]$*, donde* $\mathcal{R}$ *está definido igual que en (1.1), decimos que una política es óptima si maximiza la esperanza* $\mathbb{E}_s^\pi[\mathcal{R}]$*, es decir, una política es óptima si se cumple que* $v^* = v^\pi$ *donde:*

$$v^* = \sup_\pi v^\pi(s)$$

El objetivo en los procesos de decisión de Markov, y por tanto el del aprendizaje reforzado, es encontrar una política óptima, en la práctica se usan métodos para aproximarlos. En este trabajo se usará el método de política de gradiente como el trabajo de Williams (1992).

La justificación del uso de la política del gradiente depende del teorema de la política del gradiente cuyo enunciado se incluye en el apéndice A.

1.4.3 Enfoques de inteligencia artificial en el TSP

Como fue expuesto, el TSP ha sido abordado por algoritmos exactos y por heurísticas, sin embargo, el trabajo de Vinyals et al. (2015) abrió la posibilidad de tratar problemas de

optimización combinatoria con redes neuronales usando el modelo *Pointer-network* que es un modelo de atención el cual selecciona parte de la entrada como salida. El modelo es planteado de la siguiente manera: Dado un par de entrenamiento $\left(\mathcal{P}, \mathcal{C}^{\mathcal{P}}\right)$ donde $\mathcal{P}$ es una sucesión de n vectores y $\mathcal{C}^{\mathcal{P}} = \{\mathcal{C}_1, ..., \mathcal{C}_{m(\mathcal{P})}\}$ una sucesión de $m(\mathcal{P})$ índices, se modela la probabilidad condicional $p(\mathcal{C}_i|\mathcal{C}_1, ..., \mathcal{C}_{i-1}, \mathcal{P})$ con un vector de atención u^i:

$$u_j^i = v^T \tanh W_1 e_j + W_2 d_i \quad j \in (1, ..., n) \tag{1.2}$$

$$p(\mathcal{C}_i|\mathcal{C}_1, ..., \mathcal{C}_{i-1}, \mathcal{P}) = softmax(u^i) \tag{1.3}$$

donde $softmax$ normaliza el vector de atención u^i para convertirse en una distribución sobre el diccionario de salidas y v, W_1, W_2 son parámetros *aprendibles* del modelo de salida.

Vinyals et al. (2015) usan el modelo *Pointer-network* para resolver tres problemas de optimización combinatoria donde el TSP es uno de ellos y consiguen resultados competitivos para instancias pequeñas ($n \leqslant 50$). Un trabajo más enfocado al TSP basado en el modelo *Pointer-network* de Vinyals es el de Bello et al. (2016) el cual se dedica exclusivamente al TSP con aprendizaje reforzado y política de gradiente, reporta resultados más veloces y entrena a las redes con heurísticas más simples, además, explica que las ventajas de usar modelos de redes neuronales para resolver el TSP radican en que el modelo puede ser reutilizado en distintos problemas y requiere menos pasos para su implementación.

En el caso del TSP, el primer componente del par de entrenamiento $\mathcal{P} = \{x_1, ..., x_n\}$ está formado por las coordenadas cartesianas de las n ciudades, mientras que $\mathcal{C}^{\mathcal{P}}$ es una permutación de enteros del 1 al n representando el camino óptimo (Vinyals et al., 2015). De este modo, la longitud de un camino $\mathcal{C}_k$ está definido de la siguiente manera según Khalil et al. (2017):

$$L(\mathcal{C}_k|\mathcal{P}) = \|x_{\mathcal{C}_k(n)} - x_{\mathcal{C}_k(1)}\|_2 + \sum_{i=1}^{n-1} \|x_{\mathcal{C}_{k(i+1)} - x_{\mathcal{C}_k(i)}}\|_2 \tag{1.4}$$

De esta forma, se busca aprender los parámetros estocásticos de la probabilidad condicional $p(\mathcal{C}_k|\mathcal{P})$ que asigne probabilidades altas a caminos largos y probabilidades bajas a caminos cortos, esta optimización se logra por medio de política de gradiente como lo explican Khalil et al. (2017).

1.5 Aprendizaje reforzado en el TSP

En el contexto de procesos de decisiones de Markov, como lo indican Deudon et al. (2018), un estado es una solución parcial y una acción es la siguiente ciudad a visitar, el incentivo es la longitud del camino y la política mapea de acciones a estados.
Con este paradigma Deudon et al. (2018), resuelve el TSP entrenando una adaptación del modelo *pointer-network* con aprendizaje reforzado.

1.5.1 Arquitectura neural

En esta sección se explica la adaptación del modelo *pointer-network*, esta variante consta de tres partes: El codificador, el decodificador y el apuntador.
El codificador tiene como propósito conseguir una familia de vectores de acciones donde cada uno representa a una ciudad interactuando con otras ciudades. El codificador toma como entrada el arreglo de ciudades convolucionado y normalizado según el lote, así, la entrada está formada por N capas (N es la longitud de la entrada), cada capa tiene dos sub-capas, una llamada *feed-forward* en la cual se realiza una convolución unidimensional seguida de una activación $ReLu$. La sub-capa llamada *Multi-head attention*, la cual toma las entradas, aplica una activación $ReLu$, formando tres arreglos de pares clave-valor, $Q = [q_1, ..., q_n], K = [k_1, ..., k_n], V = [v_1, ..., v_n]$, luego, se aplica el mecanismo de atención definido de la siguiente manera:

$$Attention(Q; K; V) = softmax\left(\frac{QK^T}{\sqrt{d}}V\right) \tag{1.5}$$

d es la dimensión común de cada q_i, k_i, v_i. La salida es una nueva representación de las ciudades, calculada como la suma ponderada de los valores de las ciudades, donde los pesos se dan a partir de las *afinidades* entre cada ciudad obtenidas en la fase de entrenamiento.
Las salidas de ambas capas, dada una entrada x, es Layer norm$(x+$Sublayer$(x))$, donde *Sublayer* es la función aplicada a cada sub-capa.
El decodificador, similar al de Bello et al. (2016), usa la regla de la cadena para obtener la probabilidad de un camino como:

$$p_\theta(\pi|s) = \prod_{t=1}^{n} p_\theta(\pi(t)|\pi < t), s) \tag{1.6}$$

Cada término del producto es calculado secuencialmente con activaciones $softmax$. Para cada tiempo t de salida, mapeamos las últimas tres ciudades visitadas al siguiente vector de búsqueda:

$$q_t = ReLu(W_1 a_\pi(t-1) + W_2 a_\pi(t-2) + W_3 a_\pi(t-3)) \tag{1.7}$$

Finalmente, el apuntador se usa para predecir una distribución sobre las ciudades dadas acciones codificadas y un vector de búsqueda. Apuntando a una parte específica de de la secuencia de entradas, permite adaptar el modelo a longitudes variables.

1.5.2 Entrenamiento

La red neuronal es entrenada con la política del gradiente usando la regla REINFORCE de Williams (1992).

El objetivo del entrenamiento es el incentivo esperado, el cual corresponde con la longitud del camino que dada una secuencia de entrada s, se define de la siguiente manera:

$$J(\theta|s) = \mathbb{E}_{\pi \to p_\theta(.|s)}[r(\pi|s)] \tag{1.8}$$

Inicialmente se toma una política aleatoria que se va mejorando con la regla REINFORCE de Williams (1992), el gradiente es aproximado con integración de Montecarlo, de la siguiente manera:

$$\nabla_\theta J(\theta) \approx \frac{1}{B} \sum_{k=1}^{B} r(\pi_k|s_k) - b_\varphi(s_k) \nabla_\theta \log(p_\theta(\pi_k|s_k)) \tag{1.9}$$

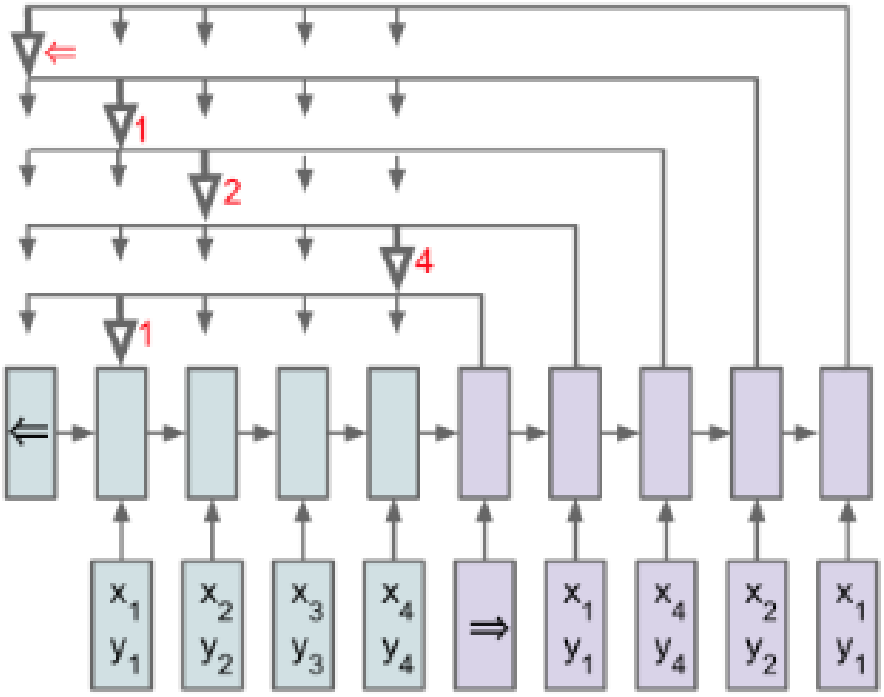

Figura 1.1. Pointer network tomado Vinyals et al. (2015)

CAPÍTULO 2

2. METODOLOGÍA

A continuación, se describe la metodología utilizada para la resolución del problema. La metodología consistió en implementar el método pointer-network, entrenar el modelo, realizar pruebas para verificar la precisión de los resultados a obtener, luego resolver el problema particular de Expotuna y finalmente realizar una comparación en términos de optimalidad sobre el problema particular con otro método. Con el primer método se resolvió el problema migrando el código provisto en el artículo de Deudon et al. (2018) de *TensorFlow 1.3.0* a *TensorFlow 2.11.0* y el segundo, con el fin de realizar una comparación de resultados, se resolvió el problema usando el algoritmo recocido simulado.

2.1 Método pointer-network

Este método se implementó siguiendo los pasos que se detallan posteriormente: creación de datos de entrenamiento, fase de entrenamiento, fase de prueba y aplicación del modelo al problema real.

2.1.1 Creación de datos de entrenamiento

Los datos fueron creados a partir de una semilla aleatoria usando la función *random.rand* de la biblioteca numpy, obteniendo coordenadas en el intervalo $[0, 1]$. Posteriormente se usó el análisis de componentes principales (PCA por sus siglas en inglés) en dos dimensiones para rotar y centrar los datos, aunque no hay reducción de dimensionalidad, se obtiene una mejor representación de los datos, de manera que son centrados en el origen y los componentes principales se alinean a los ejes coordenados. El método *get_instance* de la clase llamada *Data_Generator* encargada de generar los datos aleatorios fue implementada de la siguiente manera:

```
1
2 def gen_instance(self, max_length, dimension, seed=0):
3         if seed!=0: np.random.seed(seed)
```

```
        sequence = np.random.rand(max_length, dimension)
        pca = PCA(n_components=dimension)
        sequence = pca.fit_transform(sequence)
        return sequence
```

Código 2.1. Generación de datos aleatorios

2.1.2 Modelo y entrenamiento

El modelo que fue implementado en la clase *Actor* empieza definiendo atributos como la longitud máxima, tamaño de muestra, número de dimensión, datos de entrada, número de neuronas, tamaño del paso y *ratio* de aprendizaje. Además, contiene cuatro subrutinas que implementan el modelo *in se*. El primer método, *encode_decode* define el codificado y decodificado de los datos en las capas de la red neuronal siguiendo el modelo *Pointer-network*. Se inicializó la red usando el inicializador *Glorot uniforme*. El segundo método en ser implementado fue *build_reward* el cual implementa el incentivo del MDP como se definió en (1.4). El tercer método *build_critic* se encargó de manejar las capas de predicción y el último método *build_optim* optimizó las predicciones usando la politica del gradiente.

Para reproducir los resultados obtenidos en Deudon et al. (2018), se utilizó una semilla de 123, 512 neuronas, 20000 pasos y un ratio de aprendizaje de 0.001. Luego se inició una sesión de TensorFlow y se guardó el modelo entrenado para ser usado en la fase de entrenamiento. Cabe aclarar que el entrenamiento es auxiliado por la heurística 2opt en el sentido que se partió de una solución parcial generada por el 2opt, la cual se va mejorando.

2.1.3 Fase de prueba y resolución

Para la fase de prueba se inició una nueva sesión de TensorFlow, se aplicó el modelo, se graficó las permutaciones en distintas operaciones y al final se mostró la media de las longitudes. Los detalles se muestran en el siguiente bloque de código:

```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

```

```
    save_path = "save/"+dir_
    saver.restore(sess, save_path+"/actor.ckpt")

    predictions_length, predictions_length_w2opt = [], []
    for i in tqdm(range(1000)): # test instance
        seed_ = 1+i
        input_batch = dataset.test_batch(1, actor.max_length, actor.
            dimension, seed=seed_, shuffle=False)
        feed = {actor.input_: input_batch} # Get feed dict
        tour, reward = sess.run([actor.tour, actor.reward],
            feed_dict=feed) # sample tours

        j = np.argmin(reward)
        best_permutation = tour[j][:-1]
        predictions_length.append(reward[j])
        print('reward (before 2 opt)',reward[j])
        #dataset.visualize_2D_trip(input_batch[0][best_permutation])
        #dataset.visualize_sampling(tour)

        opt_tour, opt_length = dataset.loop2opt(input_batch[0][
            best_permutation])
        predictions_length_w2opt.append(opt_length)
        print('reward (with 2 opt)', opt_length)
        dataset.visualize_2D_trip(opt_tour)

    predictions_length = np.asarray(predictions_length)
    predictions_length_w2opt = np.asarray(predictions_length_w2opt)
    print("Testing COMPLETED ! Mean length1:",np.mean(
        predictions_length), "Mean length2:",np.mean(
        predictions_length_w2opt))

    n1, bins1, patches1 = plt.hist(predictions_length, 50, facecolor
```

```
        ='b', alpha=0.75)
31      n2, bins2, patches2 = plt.hist(predictions_length_w2opt, 50,
           facecolor='g', alpha=0.75)
32      plt.xlabel('Tour length')
33      plt.ylabel('Counts')
34      plt.axis([3., 9., 0, 250])
35      plt.grid(True)
36      plt.show()
```

Código 2.2. Fase de prueba

Después, se convirtieron los datos de los puntos de distribución de Expotuna a coordenadas cartesianas por medio de la biblioteca pyproj asumiendo geografía WGS84. Posteriormente, se procedió a resolver el problema usando el mismo esuquema que en la fase de prueba.

2.2 Método clásico

El mismo problema fue resuelto usando la librería *python-tsp* mediante el algoritmo de recocido simulado con la finalidad de comparar los resultados del nuevo método de redes neuronales con las heurísticas clásicas.

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

En esta sección, se describen los resultados obtenidos en el entrenamiento y prueba del modelo y posteriormente la aplicación al problema real con la respectiva comparación de optimalidad.

3.1 Resultados de entrenamiento

El modelo fue entrenado con un tamaño de lote de 256, con un máximo de 50 ciudades en 2 dimensiones. Con respecto a la red. Se usaron 512 neuronas, de las cuales se usaron 256 para la capa crítica.

En la tabla 3.1 se muestran los resultados de la prueba realizada al modelo con la fracción de los datos aleatorios descritos en la metodología.

Tabla 3.1. Longitudes del camino en distintas fases de la etapa de prueba

Progreso	Modelo sin 2opt	Modelo con 2opt
10%	18.26219	4.051822
25%	18.152605	4.5684
50%	15.178813	4.1249
75%	17.911282	3.81053
100%	18.17827	4.272433

A partir de los datos mostrados, se puede inferir cómo en este caso, la heurística escogida ayuda a mejorar sustancialmente la optimalidad del modelo, esto se vuelve aún más claro al ver el reporte de longitud promedio. Para el modelo sin la heurística 2opt, se obtuvo una longitud media de **18.816004**, mientras que para el modelo con 2opt fue de **4.269608**.

En el histograma de la figura 3.1 se describe la cantidad de veces que aparece cada longitud entre todas la iteraciones.

Se observa que en la prueba del modelo, las longitudes del camino se distribuyen

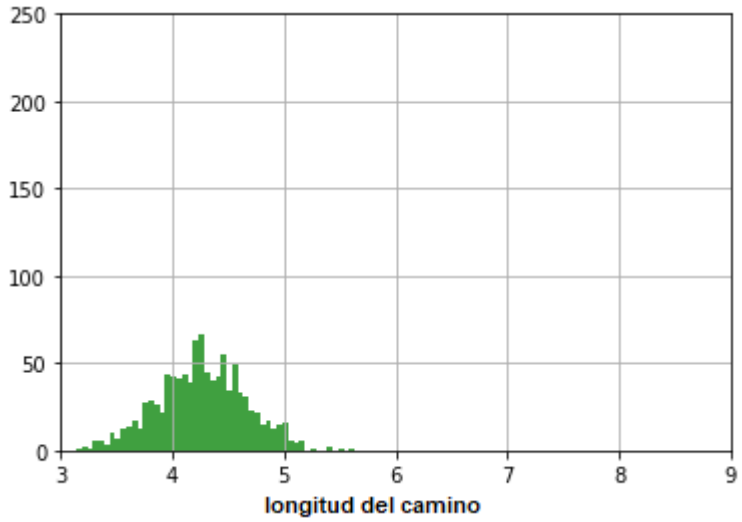

Figura 3.1. Frecuencia absoluta de longitud de camino

normalmente, como se esperaba de datos provenientes de una distribución uniforme.

3.2 Resultados del problema

Para el problema particular de Expotuna, se usó el mismo esquema y configuración que el de la fase de prueba, exceptuando la longitud del arreglo de ciudades, el cual fue cambiado a 141, lo que corresponde al número de puntos de distribución. La menor longitud obtenida fue de **489478.62**, la mejor permutación se encuentra en el Apéndice B.

En el caso del recocido simulado ejecutado con *python-tsp*, se obtuvo una longitud de **684 343.422**. La ruta conseguida por el método apuntador-red supera relativamente al segundo algoritmo por **39.81%**

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

En esta sección se presentan las conclusiones y recomendaciones obtenidas de los resultados de este trabajo, en el cual se implementó un método de redes neuronales y aprendizaje reforzado para resolver una instancia del problema del vendedor viajero.

Conclusiones

- Por el uso de la biblioteca *pyproj*, se compila la base de datos y se transforma a coordenadas cartesianas necesarias para la aplicación del modelo. Usando la geografía wgs84 que coincide con la fuente de los datos.
- Los resultados obtenidos muestran que el método propuesto se adaptó al problema de Expotuna y que compite con otros métodos en cuanto a optimalidad.
- El trabajo constituye una solución al problema y evidencia como la elección de una simple heurísitca mejora sustancialmente el rendimiento del problema, apoyando la propuesta de Deudon et al. (2018).

Recomendaciones

- En vista a los satisfactorios resultados obtenidos, se justifica dirigir estudios que traten las variantes del TSP, como el TSP con ventanas de tiempo, el problema de enrutamiento de vehículos (VRP) por medio del uso de inteligencia artificial.
- Se recomienda probar con distintas heurísticas la fase de entrenamiento para el contraste de resultados.
- Implementar el modelo propuesto a problemas de la vida real, ya que requiere de menos ajuste de parámetros que otras heurísticas.

BIBLIOGRAFÍA

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*.

Deudon, M., Cournut, P., Lacoste, A., Adulyasak, Y., and Rousseau, L.-M. (2018). Learning heuristics for the tsp by policy gradient. In *International conference on the integration of constraint programming, artificial intelligence, and operations research*, pages 170–181. Springer.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., and Song, L. (2017). Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30.

Lawler, E. (1985). *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley-Interscience series in discrete mathematics and optimization. John Wiley & Sons.

Quispe, G. L., Rodríguez, M. F., and Ontiveros, J. D. (2021). Comparative analysis of aco algorithms for the solution of the travelling salesman problem. In *Handbook of Research on Software Quality Innovation in Interactive Systems*, pages 338–358. IGI Global.

Rego, C., Gamboa, D., Glover, F., and Osterman, C. (2011). Traveling salesman problem heuristics: Leading methods, implementations and latest advances. *European Journal of Operational Research*, 211(3):427–441.

van Bevern, R. and Slugina, V. A. (2020). A historical note on the 3/2-approximation algorithm for the metric traveling salesman problem. *Historia Mathematica*, 53:118–127.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. *Advances in neural information processing systems*, 28.

Voigt, B. F. (1831). Der handlungsreisende, wie er sein soll und was er zu thun hat, um aufträge zu erhalten und eines glücklichen erfolgs in seinen geschäften gewiss zu sein. *Commis-Voageur, Ilmenau*, pages 69–72.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Wrobel, A. (1984). On markovian decision models with a finite skeleton. *Zeitschrift für Operations Research*, 28(1):17–27.

Yadav, S. P., Mahato, D. P., and Linh, N. T. D. (2020). *Distributed artificial intelligence: A modern approach*. CRC Press.

APÉNDICES

APÉNDICE A

A.1 Teorema de la política del gradiente

Teorema 1. *(Teorema de la política del gradiente): Fije un MDP* $(\mathcal{S}, \mathcal{A}, P, r)$*. Para* $x \in \mathbb{R}^d$*, defina los mapeos* $f_\pi : x \mapsto \tilde{\nu}_\mu^\pi M_{\pi_x} q^\pi$ *y* $g_\pi : x \mapsto \tilde{\nu}_\mu^{\pi_x} v^\pi$*. Fije* $\theta_0 \in \mathbb{R}^d$*. Suponga que se cumple al menos una de las siguientes condiciones:* $\theta \mapsto f'_{\pi_\theta}(\theta_0)$ *existe y es continua en una vecindad de* θ_0 *y* $g'_{\pi_{\theta_0}}(\theta_0)$ *exists;* $\theta \mapsto g'_{\pi_\theta}(\theta_0)$ *existe y es continua en una vecindad* θ_0 *y* $f'_{\pi_{\theta_0}}(\theta_0)$ *existe; Entonces,* $x \mapsto J(\pi_x)$ *is diferenciable en* $x = \theta_0$ *y*

$$\left.\frac{d}{dx} J(\pi_x)\right|_{x=\theta_0} = \left.\frac{d}{dx} \tilde{\nu}_\mu^{\pi_{\theta_0}} M_{\pi_x} q^{\pi_{\theta_0}}\right|_{x=\theta_0} = \tilde{\nu}_\mu^{\pi_{\theta_0}} \left.\frac{d}{dx} M_{\pi_x} q^{\pi_{\theta_0}}\right|_{x=\theta_0},$$

Donde la última igualdad se cumple si $\mathcal{S}$ *es finita.*

APÉNDICE B

B.1 Mejor permutación encontrada en la resolución del problema

Enumerando los puntos de distribución desde el 0 al 140, la mejor permutación hallada fue: [0, 34, 57, 46, 14, 97, 47, 39, 99, 84, 24, 85, 22, 64, 5, 132, 1, 21, 41, 77, 33, 31, 111, 2, 16, 4, 82, 27, 38, 127, 62, 135, 69, 88, 11, 87, 123, 91, 86, 49, 124, 44, 45, 20, 36, 75, 73, 95, 96, 140, 119, 80, 60, 13, 117, 106, 100, 90, 50, 55, 112, 93, 89, 40, 43, 52, 107, 101, 54, 76, 81, 3, 19, 48, 42, 26, 128, 126, 66, 115, 12, 94, 35, 65, 98, 28, 59, 134, 37, 137, 131, 102, 104, 105, 129, 130, 138, 136, 139, 122, 10, 113, 17, 53, 103, 15, 78, 92, 67, 125, 120, 56, 71, 108, 74, 18, 79, 114, 72, 32, 6, 83, 121, 63, 23, 118, 61, 109, 7, 8, 9, 110, 25, 133, 51, 70, 29, 30, 116, 68, 58]