Aprendizaje por refuerzo multiagente para la optimización de rutas del transporte público.

Ahumada García, Fermin y Durazo Duarte, Chelsea.

Licenciatura en ciencias de la computación de la Universidad de Sonora.

A 20 de mayo de 2024.

Resumen: Este proyecto nace con el objetivo de rediseñar las rutas actuales de la ciudad de Hermosillo, Sonora. Este problema ha sido abordado en la literatura utilizando técnicas de programación dinámica o algoritmos genéticos. Un enfoque reciente de resolución sería el de aplicar aprendizaje por refuerzo para encontrar las soluciones adecuadas y equilibrar la ganancia entre operador y usuarios, el cual será el utilizado en este proyecto para resolver el problema del diseño de rutas en una ciudad como Hermosillo y posteriormente el establecimiento de frecuencias de las unidades.

1. INTRODUCCIÓN

El transporte público es de suma importancia en la movilidad urbana, siendo el principal medio de desplazamiento para millones de personas en áreas urbanas densamente pobladas como Hermosillo, que según el Censo de Población y Vivienda 2020, levantado por el Instituto Nacional de Estadística y Geografía (INEGI), Hermosillo contaba con 936,263 habitantes en 2020.Sin embargo, la calidad y eficiencia de este servicio pueden variar considerablemente debido a diversos factores, desde la planificación de rutas hasta el estado de las unidades y el comportamiento de los conductores.

En esta situación, el aprendizaje por refuerzo emerge como una herramienta poderosa para mejorar la optimización de las rutas del transporte público. Esta técnica de aprendizaje automático permite a un agente tomar decisiones secuenciales en un entorno dinámico, buscando maximizar una señal de recompensa a largo plazo. Aplicar el aprendizaje por refuerzo al diseño y operación de rutas de transporte público puede significar mejoras notables en la eficiencia, puntualidad y satisfacción de los usuarios.

Esta técnica viene siendo innovadora ya que en la literatura usualmente se ha abordado el problema con técnicas convencionales como metaheurísticas, algoritmos genéticos, recocido simulado, búsqueda tabú, GRASP, o colonia de hormigas [1].

MARCO TEÓRICO

Aprendizaje por Refuerzo:

El aprendizaje por refuerzo (RL) es una técnica de aprendizaje automático orientada a objetivos desarrollada para guiar a un agente autónomo a maximizar la recompensa del agente a lo largo de repeticiones de procedimientos computacionales en un determinado entorno

1. Agente: El agente aprende y realiza acciones interactuando con un entorno.
2. Entorno: Todo lo que está fuera del agente y con el que interactúa.
3. Acción a: El conjunto de acciones disponibles para el agente.
4. Estado (S): La representación del entorno en un momento dado.
5. Factor de descuento (γ): El factor de descuento se utiliza para ponderar la importancia de las recompensas a largo plazo frente a las recompensas inmediatas.
6. Recompensa (R): La retroalimentación recibida por el agente por sus acciones, basada en cuánto se acerca a los objetivos de optimización predefinidos.

Q-learning

A medida que el agente interactúa con el entorno en una secuencia de pasos, el entorno se formula en base al proceso de decisión de Markov.

La asignación del agente de estado a acción se llama política π, que se mejora de manera iterativa a medida que el agente gana experiencia. Durante la fase de entrenamiento, se ejecutan numerosas pruebas para que el agente adquiera experiencia y encuentre, la mejor política, que determina la mejor acción cuando el agente está en un estado particular.

La actualización de la tabla de valores Q se realiza utilizando la siguiente ecuación de un paso del Q-learning:

$Q_{new}(s,a) \leftarrow Q_{old}(s,a) + \alpha(r + \gamma maxQ(s',a') - Q_{old}(s,a))$

Q_{new} es el nuevo valor de Q del siguiente estado s' bajo la acción a en el tiempo t

Q_{old} es el valor Q registrado previamente del estado s' bajo la acción a en el tiempo t

α_t es la tasa de aprendizaje en el tiempo $t(0 \leq \alpha_t \leq 1)$

γ_t es la tasa de aprendizaje en el tiempo $t(0 \leq \gamma_t \leq 1)$

$maxQ(s',a')$ representa el máximo valor de Q entre las acciones posibles del siguiente estado s'

Q table

La tabla Q es la estructura de datos que mantiene la estimación de la utilidad de tomar una determinada acción en un estado particular. La tabla se actualiza de manera iterativa a medida que el agente interactúa con el entorno y recibe recompensas.

La representación de la tabla Q se puede entender como una matriz donde las filas representan los estados posibles y las columnas representan las acciones posibles que el agente puede tomar en esos estados. Cada celda de la tabla contiene el valor Q, que es una estimación de la utilidad de tomar esa acción en ese estado específico.

Si tienes N estados y M acciones posibles, la tabla Q tendrá $N \times M$. cada celda (s, a) de la tabla Q representa el valor Q de un estado s y la acción a.

$$\boldsymbol{Q(s,a)} = \begin{bmatrix} \boldsymbol{Q(s_1,a_1)} & \boldsymbol{Q(s_1,a_2)} & \dots & \boldsymbol{Q(s_1,a_M)} \\ \boldsymbol{Q(s_2,a_1)} & \boldsymbol{Q(s_2,a_2)} & \dots & \boldsymbol{Q(s_2,a_M)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{Q(s_N,a_1)} & \boldsymbol{Q(s_N,a_2)} & \dots & \boldsymbol{Q(s_N,a_M)} \end{bmatrix}$$

$s_1, s_2, \dots\dots, s_N$ representan los estados posibles

$a_1, a_2, \dots\dots, a_M$ representan las acciones posibles

$Q(s_i, a_j)$ es el valor Q para el estado s_i y la a_j

MDP

Procesos de decisión de Markov (MDP)

Un proceso de decisión de Markov es un marco matemático de toma de decisiones utilizado para modelar problemas estocásticos en los que se toma decisiones basándose en un modelo predefinido (o política). Tal decisión llevaría al agente de un estado a otro basándose en una distribución de probabilidad, que depende en gran medida de cuándo se tomó la decisión.

Formalmente, un MDP es una tupla (S, A, P_t, R_a) donde:

• S es un conjunto finito de estados

• A es un conjunto finito de acciones

• $P_t$es la probabilidad de pasar del estado s al s' después de una acción a en el tiempo t

• R_a es la recompensa inmediata recibida debido a la transición de un estado a otro.

2. METODOLOGÍA

Abordaremos el problema de TNDFSP (Transit network design and frequency setting problem – Diseño de la red de tránsito y establecimiento de frecuencias).

El problema se dividirá en dos partes; primero el diseño de la red de transporte y segundo el establecimiento de frecuencias. El problema de establecimiento de frecuencias requiere de una ya existente red de tránsito sobre la cual se asignarán demandas entre las paradas y se intentará minimizar el tiempo total del viaje para cada pasajero.

1. Diseño de la Red de Transporte:

Se busca crear una red de rutas conectadas que satisfaga las necesidades de conectividad de la ciudad, y se puede expresar como

$$G(N, E)$$

donde N es un conjunto de nodos que representan las paradas, y E es un conjunto de arcos que representan los caminos disponibles.

Establecimiento de Frecuencias:

Se asignarían frecuencias de servicio para minimizar el tiempo total de viaje de los pasajeros. Esto puede formularse como la optimización de una función de coste que tenga en cuenta el tiempo de espera en las paradas y el tiempo de viaje en autobús.

2. RESTRICCIONES DEL PROBLEMA

Tomaremos algunas de las restricciones utilizadas por Ahmed Darwish et. al. en su artículo *"Optimising Public Bus Transit Networks Using Deep Reinforcement Learning"* [2]

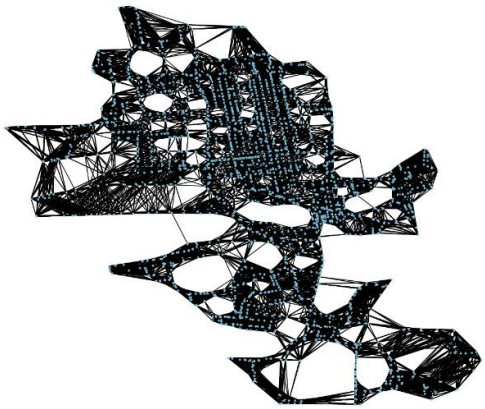
1. La red resultante es un grafo conectado, refiriendo que cualquier parada puede conectar con cualquier otra dentro de la red.
2. Todas y cada una de las paradas deberán pertenecer al menos a una de las rutas.
3. Las rutas deben ser diferentes entre sí.
4. El número de las rutas totales es predefinido para cada ejecución.
5. Todos los autobuses tienen la misma capacidad.
6. No se repiten paradas dentro de una ruta. Estas deben comenzar en un punto y terminar en otro.
7. Cada ruta tendrá un límite de distancia máxima para resguardar el tiempo de conducción sano de los conductores.

3. DISEÑO DE LA CIUDAD

Para probar el modelado del problema, se empleará un grafo que representa las paradas de Hermosillo. Para obtener este grafo, se utilizó información detallada de la ciudad para calcular las distancias precisas entre las paradas en las rutas actuales del sistema de transporte público.

El mapa de las paradas actuales del transporte público fue proporcionado por el Instituto de Movilidad y Transporte del Estado de Sonora.

Como resultado, obtuvimos el siguiente grafo, sobre el cual se explorarán las diferentes combinaciones posibles de rutas.



4. DEMANDA

La matriz de origen destino para la demanda de pasajeros se puede expresar como:

$$D = \{d_{ab} | a, b \in N\}$$

Donde d_{ab} es el número de pasajeros que viajan de la parada a a la parada b.

Dado que no tenemos una demanda real, para hacer las pruebas se creo una matriz de unos para asegurar una demanda mínima de uno de cualquier parada a otra.

5. DISEÑO DE LAS SOLUCIONES

Obtendremos como resultado una lista de n listas con una longitud máxima dada en las que cada una de las listas de dentro representará cada una de las n lineas establecidas en el inicio. Por ejemplo, si decidimos crear una red de 3 líneas para una ciudad de 9 paradas, una solución válida podría ser la siguiente:

[[1, 2, 3], [2, 4, 6, 8], [1,6, 7, 5, 4, 9]]

Un estado final dentro de nuestro modelo podría ser cualquier solución valida.

6. FUNCIÓN DE RECOMPENSA

Para recompensar las acciones de cada agente se tomaron en cuanta cada uno de los siguientes puntos:

1. Recompensa si el grafo creado por los agentes está conectado.
2. Recompensa si el estado es final, es decir que todas las paradas pertenecen al menos a un agente.
3. Penalización por la distancia recorrida.
4. Recompensa por la demanda satisfecha.
5. Penalización si hay agentes que comparten una cantidad significable de paradas.
6. Penalización por seleccionar una parada que ya estaba dentro del agente.
7. Penalización si la ruta termina muy cerca del punto de partida
8. Recompensa por visitar un nodo no explorado previamente

7. MULTIAGENT DEEP Q-LEARNING

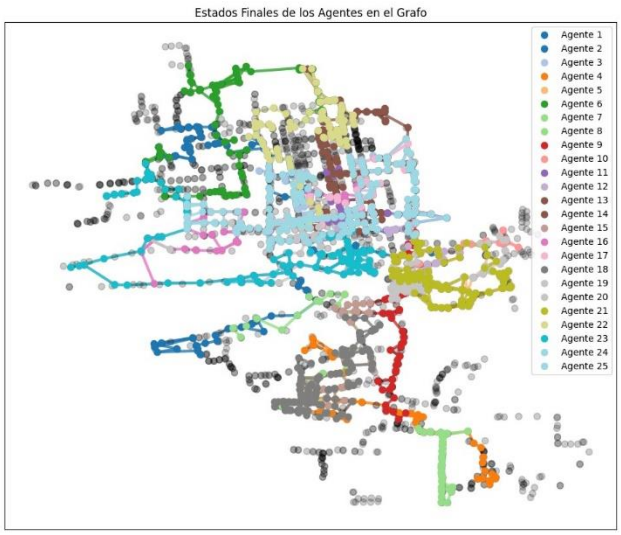
Implementamos una libreta para ejecutar un entorno con las especificaciones dadas previamente junto con una implementación de Deep Q-Learning multiagente. El código se basó en la libreta de GitHub publicada por Moya (2024)[3], que consiste en una implementación de Deep Q-Learning.

Modificamos esta implementación para adaptarla a nuestra necesidad de trabajar sobre un problema multiagente cooperativo, y como resultado obtuvimos una libreta publicada en GitHub [4].

Los resultados obtenidos fueron procesados en un entorno de Google Colab y serán presentados a continuación.

8. RESULTADOS

La siguiente es una ejecución con 30 agentes, 120 steps por agente y un total de 40 episodios.



La elección de estos parámetros se debe al número total de paradas en el grafo. Si queremos que un estado final contenga todas las paradas del grafo, debemos asegurarnos de que haya oportunidad de encontrar este estado final. La cantidad de nodos en el grafo es de un poco menos de 3000.

9. CONCLUSIONES

Este ha sido un buen comienzo como prototipo para abordar un problema real de diseño de rutas. Como trabajo futuro se puede considerar probar el aprendizaje por refuerzo utilizando un modelo individual para cada agente para mejorar las decisiones individuales que toma cada uno. Esto como metaheurística o simplemente para explorar los distintos resultados que podemos obtener con este cambio.

También sería una gran mejora implementar una demanda real para cada una de las paradas.

10. REFERENCIAS

[1] D. Moctezuma García, «Implementación y análisis de estrategias para optimización de redes de transporte público considerando variaciones de precio», Tesis de maestría, Universidad de Colima, Colima, México, 2018.

[2] K. Darwish, A., Khalil, M., & Badawi, «Optimising Public Bus Transit Networks Using Deep Reinforcement Learning.», *IEEE Xplore*, 2018.

[3] R. Moya, «2_02_Deep_Q-Learning.ipynb», GitHub, 2024. [En línea]. Disponible en: https://github.com/RicardoMoya/Reinforcement_Learning_with_Python

[4] C. Durazo Duarte y F. A. Ahumada Garcia, «Multiagent Deep Q-learning For Transport Network Design Problem», GitHub, 2024. [En línea]. Disponible en: https://github.com/chelseadz/MultiagentDeepQ-learningForTransportNetworkDesignProblem