

## Proyecto 3 - Aprendizaje por refuerzo

### Antecedentes - Definición del Problema

#### Contexto

A medida que las ciudades van creciendo, se va haciendo necesario optimizar el tráfico para garantizar una vida más armoniosa y organizada. Uno de los elementos a revisar es la optimización en los tiempos de cambio de los semáforos, para los cuales se pueden definir procedimientos de aprendizaje con el fin de minimizar trancones generados por semáforos en rojo.

En Bogotá, por ejemplo, hay instaladas cámaras sobre algunos semáforos cuya función es detectar qué tan lleno de carros está un carril, y de acuerdo a esos datos, cambiar en tiempo real los tiempos de cambio de luz en cada semáforo. [1]

#### Definición del Problema

Queremos explorar la posibilidad de que un semáforo aprenda, con la ayuda de los algoritmos de aprendizaje por refuerzo, los tiempos óptimos de cambio de luz. Para ello dedicamos bastante esfuerzo modelando una intersección vial con su respectivo sistema de semáforos que nos simule la manera en que interactúa cada semáforo con los carros van transitando por la intersección teniendo en cuenta las horas en que el tráfico aumenta y disminuye.

El modelado del problema nos implicó pensar en particularidades que hiciesen que el problema fuese más interesante.

Uno de estos factores es nuestro modelado del tiempo: usamos ‘ticks’ o pulsos como unidades, en principio abstractas, de tiempo. En el problema establecíamos que por cada tick que el semáforo estaba en verde, 4 carros cruzarían el semáforo. Esto nos dio una conversión en la que establecimos que un tick equivaldría a 10 segundos, por lo cual, establecimos que un semáforo debía demorar al menos dos ticks en cambiar y a lo más 9 ticks. Entre ellas propusimos que a lo largo del día la densidad del tráfico se comportara con una curva senoidal para simular las horas pico y valle, y le dimos a esta forma senoidal un período de 12 horas.

También establecimos que en el sentido norte-sur cruzarían 1.5 veces la cantidad de carros que cruzan en el sentido este-oeste, simulando que el sentido norte-sur es una avenida mientras que el sentido este-oeste es una calle secundaria.

#### Definición de algoritmos

##### Estados

Los estados sobre los que vamos a trabajar los definimos a partir de

- El sentido en que el semáforo está en verde (norte-sur, este-oeste)
- Cuántos carros se encuentran por cruzar el semáforo por sentido
- La cantidad de ticks o el tiempo que ha pasado desde el último cambio de semáforo
- La hora del día medida como la cantidad de ticks desde que empieza el periodo de la función seno

## Acciones Legales

Las acciones que el agente puede ejecutar son sencillamente cambiar de semáforo *SWITCH* y no cambiar *STAY*. En principio colocamos un máximo y un mínimo de ticks que el semáforo debe permanecer sin cambiar, puesto que, por un lado, no queremos que el semáforo cambie demasiado rápido, y por otro lado, no queremos que se demore demasiado en cambiar.

## Recompensas y penalizaciones

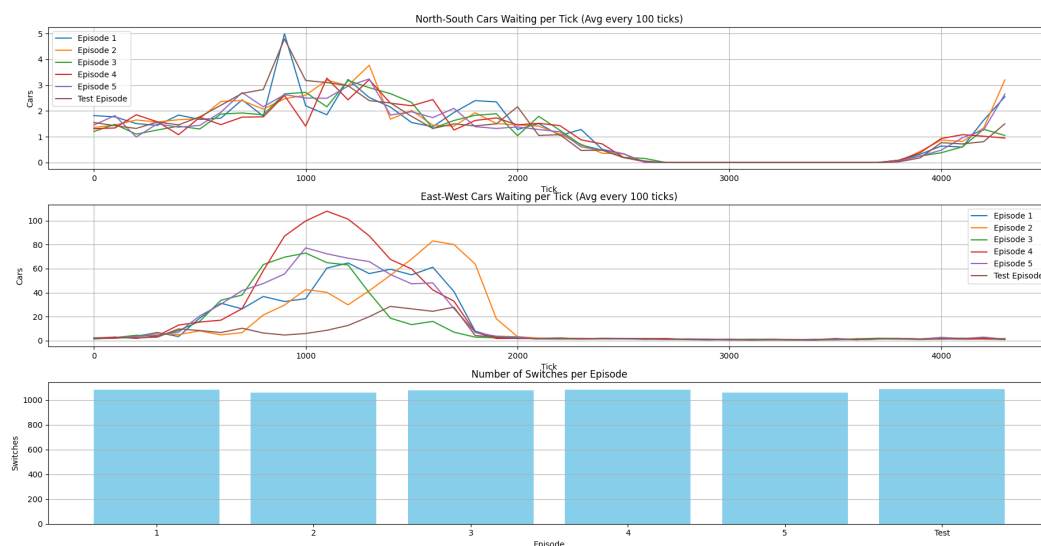
Para *Q-learning* y *Q-learning con epsilon-greedy* establecimos algunas funciones de recompensa distintas esperando que alguna nos arrojara resultados realistas que funcionaran bien especialmente para horas pico. Propusimos algunas variaciones, entre ellas penalizar el que se le diera mucho más tiempo en verde a una dirección que a otra, el contar cuántos carros esperan cruzar la intersección por cada tick, jugando con esos valores y el penalizar que se cambie la luz demasiado rápido.

En general lo que más nos sirvió fue contar los carros que estaban a la espera. Puesto que las demás variaciones generaban un desbalance muy grande teniendo en cuenta la diferencia en el tránsito que establecimos entre los dos sentidos.

Por otro lado, para *Q-Learning aproximado*, los features que consideramos también contabilizan los carros que esperan en cada sentido y el desbalance entre los tiempos en verde de los distintos sentidos.

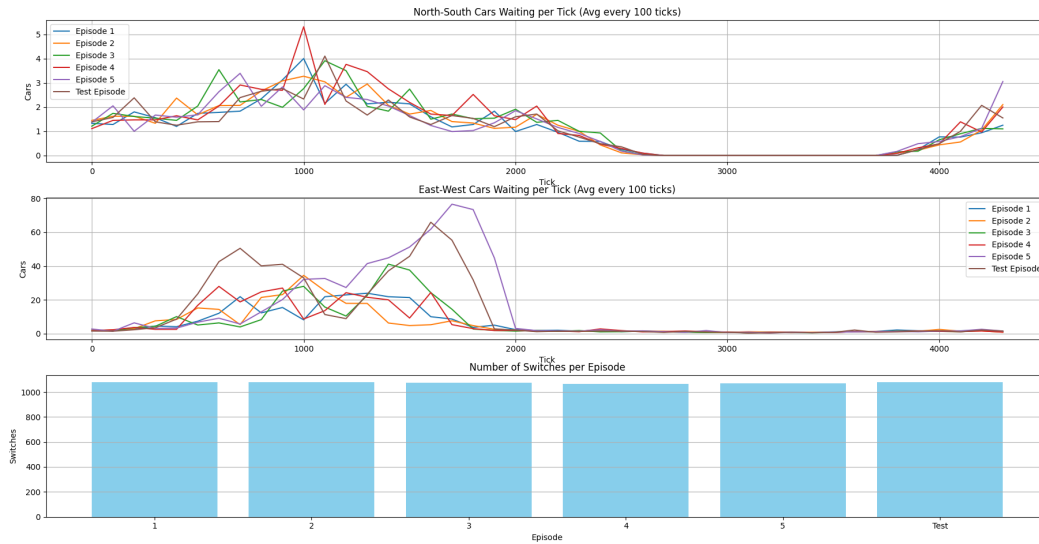
## Resultados

A continuación algunos resultados que obtuvimos al correr simulaciones con distintas variaciones. Corrimos 5 episodios (periodos de 12 horas) de entrenamiento y uno de prueba para verificar qué tan bueno fue el entrenamiento:



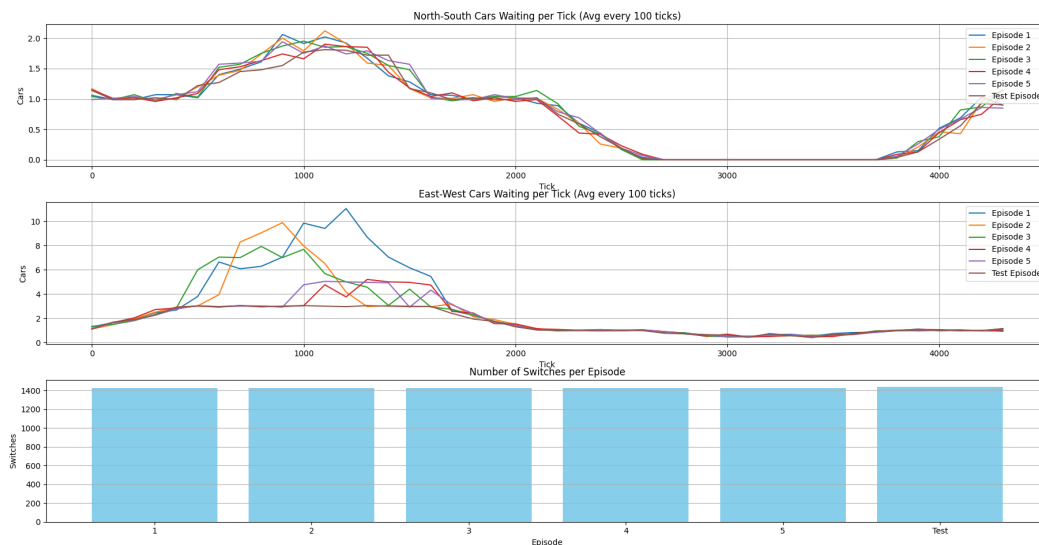
Entrenamiento (5 episodios o *épocas*) cada una con 4320 ticks con el agente QLearning. La función de recompensa que usamos es el total de carros esperando.

Aquí vemos que el episodio de prueba fue muy favorable para la dirección este-oeste mientras que no fue muy buena para la dirección norte-sur. Lo que no logramos entender es por qué si la dirección este-oeste tiene menos tráfico.



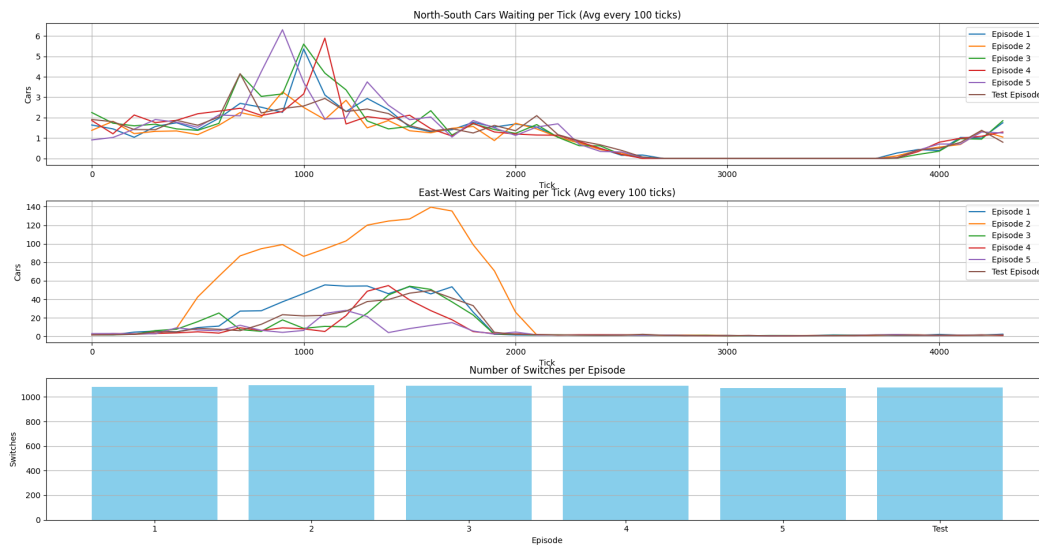
Entrenamiento (5 episodios o *épocas*) cada una con 4320 ticks con el agente QLearning usando epsilon greedy para la exploración. La función de recompensa que usamos es el total de carros esperando.

Aquí no vemos que haya habido un aprendizaje notablemente bueno y el problema con la gran cantidad de carros esperando persiste.



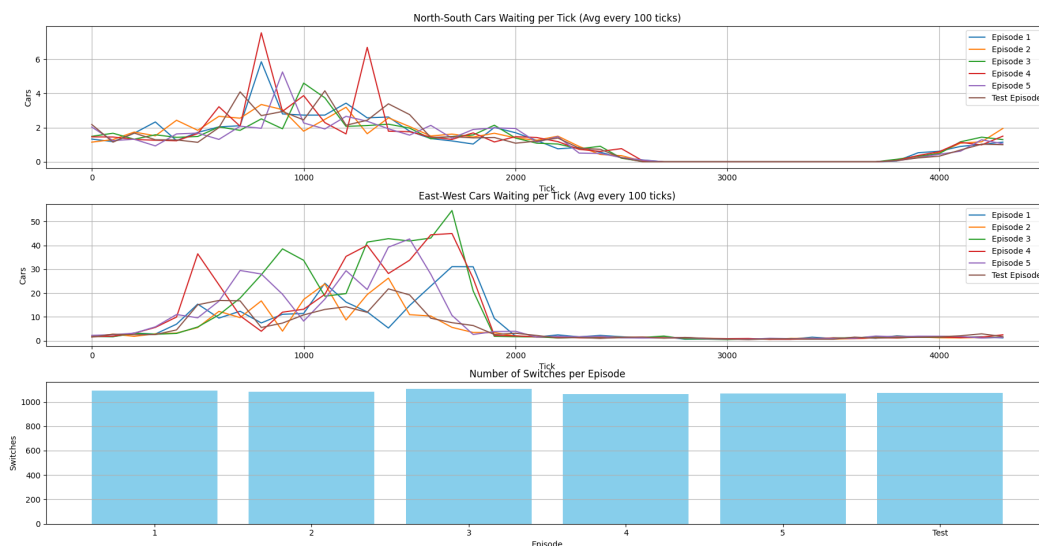
Entrenamiento (5 episodios o *épocas*) cada una con 4320 ticks con el agente QLearning aproximado. La función de recompensa que usamos es el total de carros esperando.

En esta ejecución sí notamos un muy buen aprendizaje y los numeros son mucho mejores y más equilibrados.



Entrenamiento (5 episodios o *épocas*) cada una con 4320 ticks con el agente QLearning usando epsilon greedy para la exploración. La función de recompensa que usamos es el total de carros esperando sumando el desbalance entre los sentidos.

Acá notamos un muy buen aprendizaje pero pésimos números para la dirección este-oeste.



Entrenamiento (5 episodios o *épocas*) cada una con 4320 ticks con el agente QLearning usando epsilon greedy para la exploración.. La función de recompensa que usamos es el total de carros esperando sumando una penalización por cambios muy rápidos.

Acá vemos un buen aprendizaje y no muy malos números.

En general consideramos que hace falta una evaluación más amplia de los distintos parámetros que estamos utilizando. Consideramos posible que nos ayudaría recalibrar los valores de tasa de aprendizaje, del epsilon, del ruido, las iteraciones de episodios de prueba, entre otros. Posiblemente podríamos, como tarea a futuro hacer pruebas exhaustivas de muchas combinaciones de parámetros posibles para obtener resultados más adecuados y aplicables en la vida real.

## Conclusiones

Como nuestro ejercicio está planteado bajo supuestos empíricos, creemos que los resultados podrían ser mucho más notorios y útiles si se basan en datos reales de tráfico, por ejemplo un

ejercicio muy interesante sería tomar una intersección y registrar cuántos vehículos la cruzan en lapsos de tiempo durante todo el día. Además se pueden tener en cuenta otros factores que afectan este tráfico como el día de la semana o incluso la época del año en que se hace esta observación. Sería muy interesante extender el ejercicio a escenarios reales pues esto podría generar una mejoría en la movilidad urbana.

## Enlace al Repositorio

El código de este proyecto puede ser encontrado en el link: [https://github.com/jrodriguezru/traffic\\_lights\\_ai/](https://github.com/jrodriguezru/traffic_lights_ai/)

## Referencias

- [1] Secretaría Distrital de Movilidad de Bogotá. *Con la instalación de contadores regresivos y cámaras de detección vehicular avanza el proyecto de modernización de la semaforización.* [https://www.movilidadbogota.gov.co/web/Noticia/con\\_la\\_instalacion\\_de\\_contadores\\_regresivos\\_y\\_camaras\\_de\\_deteccion\\_vehicular\\_avanza\\_el](https://www.movilidadbogota.gov.co/web/Noticia/con_la_instalacion_de_contadores_regresivos_y_camaras_de_deteccion_vehicular_avanza_el). Consultado el 10 de diciembre de 2025. 2021.