

Gestión semafórica basada en aprendizaje por refuerzo: Q-Learning y SARSA

Traffic light management based on reinforcement learning: Q-Learning and SARSA

José Joaquín Vergara Cartagena¹, Leober Arturo Duran Raudales², Juan Pablo Hoyos Sánchez³

¹ Mechatronic Engineer, Universidad Nacional de Colombia, Sede de La Paz, La Paz, Colombia.

 <https://orcid.org/0009-0005-0650-9056>

² Mechatronic Engineer, Universidad Nacional de Colombia, Sede de La Paz, La Paz Colombia.

 <https://orcid.org/0009-0002-1609-565X>

³ Doctor in Electronic Sciences, Universidad Nacional de Colombia, Sede de La Paz, La Paz, Colombia.

 <https://orcid.org/0000-0002-1844-9127>

jhoyoss@unal.edu.co

Cite this article as: J. Vergara Cartagena, L. Duran Raudales, J. Hoyos Sánchez, “*Gestión semafórica basada en aprendizaje por refuerzo: Q-Learning y SARSA*” *Prospectiva*, Vol. 24 N° 2 2026

Recibido: 20/11/2025 / Aceptado: 24/06/2026

RESUMEN

El crecimiento acelerado del tráfico urbano ha puesto de manifiesto las limitaciones de los semáforos tradicionales con temporización fija, que no se adaptan a la variabilidad de los patrones de tráfico. En respuesta a ello, este trabajo propone el uso de algoritmos de aprendizaje por refuerzo, concretamente Q-Learning y SARSA, para implementar sistemas de control de semáforos más eficientes y dinámicos. Se desarrollaron simulaciones en el entorno SUMO (Simulación de Movilidad Urbana) para comparar el rendimiento de estos modelos con el de los semáforos tradicionales basados en ciclos fijos, tomando como métrica principal el tiempo total de espera por vehículo. Los resultados permiten evaluar la adaptabilidad y superioridad de los modelos inteligentes en la mayoría de los escenarios de tráfico, obteniendo reducciones en los tiempos promedio de espera entre el 20% y el 50%, y en los escenarios de máxima congestión de 2400 veh/h se encontró que el desempeño se degrada, mostrando un mejor desempeño que el control tradicional. Los resultados del análisis sistemático demuestran su potencial uso como solución eficaz para gestionar el flujo de vehículos en entornos urbanos.

Palabras clave: Control de tráfico vehicular, SUMO, TraCI, Q-learning, SARSA, SUMO-RL.

ABSTRACT

The accelerated growth of urban traffic has revealed the limitations of traditional traffic lights with fixed timing, which do not adapt to the variability of traffic patterns. In response, this work proposes the use of reinforcement learning algorithms, specifically Q-Learning and SARSA, to implement more efficient and dynamic traffic light control systems. Simulations were conducted in the SUMO (Urban Mobility Simulation) environment to compare the performance of these models with that of traditional traffic lights based on fixed cycles, using average waiting time per vehicle as the primary metric. The results allow for an evaluation of the adaptability and superiority of intelligent models in most traffic scenarios, yielding reductions in average wait times ranging from 20% to 50%; however, in scenarios of maximum congestion (2,400 vehicles per hour), performance was found to degrade, with traditional control systems demonstrating better performance. The results of the systematic analysis demonstrate their potential use as an effective solution for managing vehicle flow in urban environments.

Keywords: Vehicle traffic control, SUMO, TraCI, Q-learning, SARSA, SUMO-RL.

1. INTRODUCCIÓN

El crecimiento urbano y el aumento de vehículos han hecho que el flujo vehicular sea cada vez más complejo, especialmente en las ciudades con un crecimiento acelerado y con infraestructuras viales limitadas; como consecuencia, la eficiencia de los semáforos tradicionales es cada vez menor. Este tipo de semáforos con tiempos fijos ya no cumple con las exigencias en ciudades que cambian constantemente, por tanto, se hace necesario empezar a explorar alternativas más eficientes [1].

En las últimas décadas, la evolución de los sistemas inteligentes para el control del tráfico ha sido sorprendente, métodos como la lógica difusa, el aprendizaje por refuerzo y los semáforos autoorganizados (SOTL) ofrecen distintas formas de afrontar la complejidad vial, adaptándose a escenarios que van desde cruces sencillos hasta ciudades densamente pobladas.

Por otra parte, la optimización del flujo vehicular mediante el uso de simulación ha sido una estrategia usada con frecuencia para enfrentar el aumento de la congestión en las intersecciones urbanas. El uso de herramientas como Simulation of Urban MObility (SUMO) permite construir escenarios con el fin de evaluar distintos esquemas de control para facilitar el análisis de políticas de semaforización en entornos complejos [2]. Algunos estudios muestran resultados positivos, evidenciando así la efectividad del uso de estas herramientas, como es el caso de la intersección Samsat en Bandung, donde se validó un sistema de semáforo adaptativo con base en datos reales, y el desarrollo de entornos urbanos simulados como el de Bangalore; en estos se obtuvieron resultados que lograron replicar la geometría urbana y patrones de movilidad [1, 2].

En estos escenarios simulados se han probado algoritmos de aprendizaje reforzado (RL) buscando mejorar el comportamiento de estos sistemas de control. Los métodos clásicos como Q-Learning (QL) y State Action Reward State Action (SARSA) han demostrado su eficacia al superar estrategias tradicionales, como la política de cola más larga primero, incluso en escenarios donde no se tiene control total del entorno [3].

Algunas aplicaciones evidencian mejoras en la duración de los viajes, como la realizada en el sistema urbano de Taipéi, destacando así el potencial que tienen estos algoritmos en contextos reales [4].

Debido a que muchas ciudades no cuentan con infraestructura para la detección de vehículos, es necesario diseñar sistemas que operen en condiciones donde solo se tiene información de los vehículos en espera. Teniendo esto en cuenta, se han desarrollado enfoques que son capaces de aprender políticas óptimas aun cuando sólo se dispone de una información parcial del entorno, estos han demostrado mejoras frente a los métodos tradicionales [5]. Para mejorar la interpretación del entorno en estas condiciones, se han propuesto modelos basados en ontologías que incorporan razonamiento semántico, esto les permite a los agentes mantener un rendimiento óptimo en entornos dinámicos y con detección incompleta [6].

Algunas revisiones han documentado la evolución del aprendizaje por refuerzo enfocado al control de intersecciones semafóricas, destacando que algoritmos como Q-Learning y SARSA han sido fundamentales. Estas revisiones identifican desafíos persistentes, como la dificultad para trasladar modelos desde simulación hacia entornos reales [7-9]. En este contexto, SUMO ha sido ampliamente usado como el entorno por excelencia, debido a su capacidad para representar de manera precisa cómo se comporta el tráfico urbano, su compatibilidad con diversos algoritmos y su capacidad para hacer evaluaciones reproducibles [9, 10].

Así mismo, investigaciones que exploran estructuras más complejas han aportado nuevas ideas de gran utilidad, como el uso de recompensas compuestas para equilibrar múltiples objetivos simultáneamente, minimizando colas y tiempos de espera [11]. También el análisis de intersecciones críticas dentro de una red, que permite aplicar control inteligente solo en zonas estratégicas y así reducir la complejidad [12]. Aunque estas propuestas emplean técnicas avanzadas, sus principios se pueden utilizar en variantes más simples compatibles con algoritmos como SARSA y Q-Learning. Por otro lado, revisiones más amplias sobre sistemas de transporte inteligente han identificado la utilidad de estos algoritmos, junto con otros como Proximal Policy Optimization (PPO) y Asynchronous Advantage Actor-Critic (A3C), y destacan el desafío que existe al transferir modelos entrenados en simulación a condiciones reales [13].

En resumen, los enfoques basados en aprendizaje por refuerzo han demostrado ser herramientas útiles para el control de semáforos en entornos simulados, especialmente cuando no se tiene conocimiento completo del entorno [11]. Si bien investigaciones recientes han propuesto estructuras más complejas y técnicas avanzadas, sus conceptos pueden ser adaptados a variantes más simples que sean compatibles con entornos como SUMO. La literatura también reconoce la importancia de enfrentar desafíos como la escalabilidad y la transferencia a contextos reales, lo que refuerza la necesidad de seguir explorando soluciones prácticas, interpretables y reproducibles dentro de marcos experimentales controlados [12, 13].

Así, el objetivo de este artículo es realizar un análisis sistemático del comportamiento de los algoritmos Q-Learning y SARSA en la gestión dinámica del tráfico, evaluando su desempeño frente a un esquema de control tradicional de semáforos de ciclos fijos en una intersección de la ciudad de Valledupar. A diferencia de trabajos previos, este estudio compara ambos algoritmos bajo una misma configuración experimental utilizando funciones de recompensa basadas en métricas de tráfico relevantes, como el tiempo de espera, el número de vehículos detenidos y la fluidez general, permitiendo un análisis más consistente de su comportamiento en distintos niveles de congestión. El estudio se centra en una intersección urbana crítica y

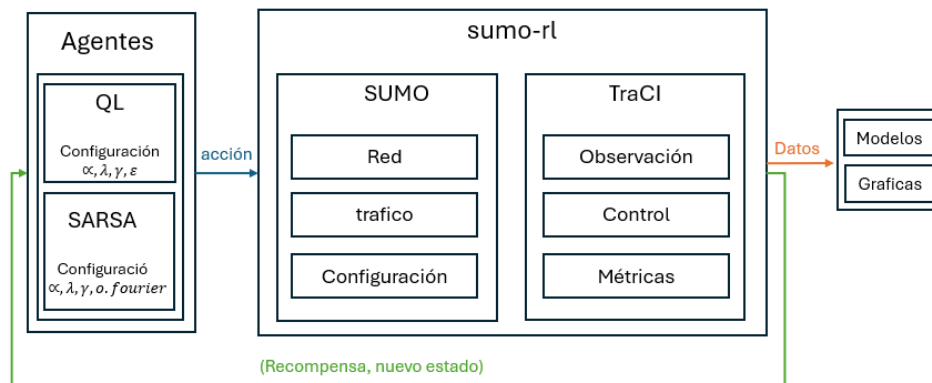
asimétrica de la ciudad de Valledupar, modelada a partir de un entorno real. Para ello se simularon tres escenarios de tráfico: bajo, medio y alto, y se evaluó el tiempo de espera de los vehículos obtenido con cada enfoque, comparándolo con un sistema tradicional de ciclos fijos. El estudio se limitó a la evaluación mediante una simulación microscópica en SUMO sobre una intersección urbana crítica y asimétrica modelada a partir de un entorno real.

Los resultados obtenidos en este entorno simulado demuestran que los algoritmos de aprendizaje por refuerzo pueden superar el control tradicional de semáforos, con un promedio de mejora del 20% y máximos de 50% en algunos escenarios, aunque en ciertos escenarios de alto tráfico el control tradicional supera a los algoritmos. No obstante, el estudio presenta limitaciones intrínsecas a su naturaleza computacional, como no poder replicar el comportamiento exacto del tráfico en la intersección; por esto los modelos fueron entrenados bajo flujos vehiculares homogéneos y estacionarios, omitiendo factores de la dinámica real como el comportamiento de peatones y conductores, giros prohibidos o imprevistos viales.

2. METODOLOGÍA

Para evaluar el desempeño de semáforos tradicionales frente a modelos de aprendizaje por refuerzo, se diseñó una serie de experimentos utilizando el simulador SUMO, en conjunto con la librería sumo-rl, que facilita la interacción entre el entorno de simulación y agentes inteligentes en Python a través de TraCI. En la Figura 1 se resume el flujo de interacción entre los agentes y el entorno de simulación. Los agentes generan acciones con base en políticas, que se envían al simulador SUMO a través de TraCI. Este entorno simulado incluye una red vial, tráfico y parámetros de configuración definidos. A través de TraCI, se recopilan observaciones, métricas y se ejerce control sobre la simulación; todo esto se encarga de llevarlo a cabo la librería sumo-rl. Estos datos retroalimentan al agente, permitiendo la actualización de sus políticas. Además, los resultados se almacenan como modelos aprendidos y gráficas para su posterior análisis.

Figura 1. Diagrama de flujo de la metodología propuesta



2.1. Simulador SUMO

El Simulador de Movilidad Urbana (SUMO) es un paquete de simulación de tráfico de código abierto, altamente portátil, microscópico y continuo, diseñado para manejar grandes redes. También permite la

simulación intermodal que incluye peatones y viene con un gran conjunto de herramientas para la creación de escenarios. Es desarrollado principalmente por el equipo del Institute of Transportation Systems del German Aerospace Center. SUMO está licenciado bajo la EPL 2.0. Y el código fuente también puede estar disponible bajo las siguientes licencias cuando se cumplen las condiciones para dicha disponibilidad establecidas en la EPL 2.0: GPL2 o posterior.

Dentro de nuestro proyecto, SUMO proporciona el entorno físico sobre el cual se ejecutan las simulaciones de tráfico. Los modelos de red vial son definidos a través de archivos XML que especifican intersecciones, calles, carriles y rutas de los vehículos. El simulador opera con un motor de física que gestiona el movimiento de los vehículos de acuerdo con parámetros como velocidad máxima, aceleración, tiempo de seguimiento y reglas de prioridad.

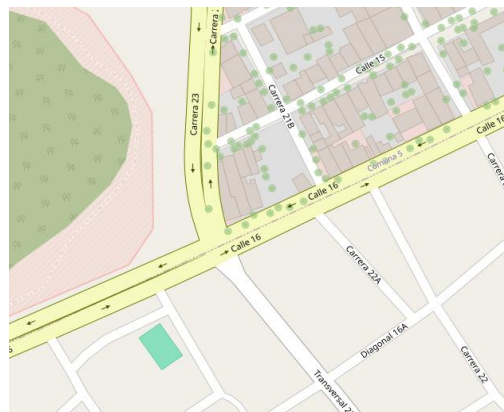
El componente clave que permite la interacción en tiempo real entre el simulador y agentes externos es TraCI (Traffic Control Interface), una interfaz cliente-servidor que habilita la manipulación dinámica de elementos del tráfico durante la simulación. A través de esta interfaz, bibliotecas como sumo-rl pueden modificar las fases semafóricas, observar métricas de tráfico como la longitud de cola, la densidad por carril, o el número de vehículos detenidos, y aplicar políticas de control aprendidas por el agente.

Gracias a su flexibilidad y compatibilidad con códigos en Python, SUMO permite realizar simulaciones complejas, siendo ideal para experimentar con algoritmos de aprendizaje por refuerzo, los cuales requieren una gran cantidad de episodios y variabilidad en las condiciones del entorno para lograr un entrenamiento óptimo.

2.2. Selección y modelado de la intersección

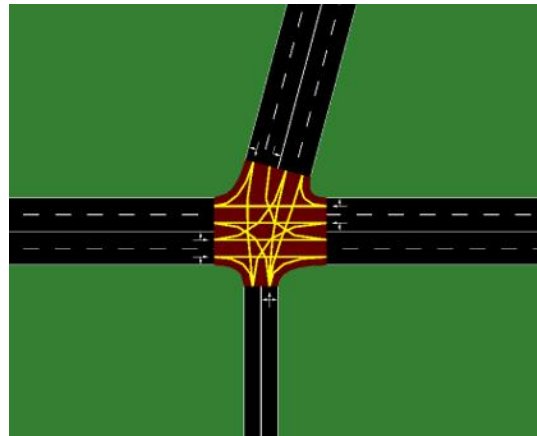
La intersección seleccionada corresponde a un cruce ubicado en la ciudad de Valledupar, Colombia. Este se caracteriza por su asimetría y altos niveles de congestión vehicular. Esta intersección integra un total de ocho vías, de las cuales seis corresponden a vías de doble carril, tres entradas y tres salidas, y dos a vías de un solo carril. La geometría fue extraída de OpenStreetMap como se aprecia en la Figura 2, dicho cruce es la intersección que se encuentra en la Carrera 23 con calle 16.

Figura 2. Geometría de la intersección en OpenStreetMap.



Posteriormente se modeló manualmente utilizando el editor de redes NetEdit, para definir con precisión los carriles, conexiones y fases semafóricas. Este diseño constituye la base de la red vial utilizada en el entorno de simulación SUMO.

Figura 3. Geometría de la intersección modelada en netedit.



Ante la ausencia de datos históricos comerciales sobre el flujo vehicular en la intersección real, se diseñaron tres tipologías de escenarios de tráfico para explorar exhaustivamente los límites operativos de los algoritmos:

- **Flujos Homogéneos (Simétricos):** Escenarios donde todos los accesos de la intersección experimentan una tasa de llegada vehicular idéntica y equilibrada, variando de manera controlada desde demandas bajas hasta flujos saturados.
- **Flujos No Homogéneos (Asimétricos):** Escenarios que replican la asimetría típica de las arterias viales, configurando una carga preferencial alta en los accesos principales (ej. 70 unidades de flujo) y una carga reducida en las vías secundarias (ej. 40 unidades de flujo).
- **Carril sin Flujo:** Escenarios extremos de desequilibrio donde una o más direcciones registran una demanda nula (0 unidades de flujo), diseñados para evaluar la capacidad del agente inteligente de omitir fases verdes innecesarias y mitigar los "tiempos muertos" de los sistemas estáticos.

2.3. Configuración de la librería sumo-rl

La librería sumo-rl es una herramienta de código abierto que se basa en la API de gymnasium, diseñada específicamente para este propósito, entrenar agentes RL en entornos de control de tráfico vehicular. Esta librería permite interactuar de manera eficiente con el simulador SUMO (Simulation of Urban MObility), facilitando la creación de entornos urbanos realistas donde el agente puede observar, actuar y aprender en ciclos. En este proyecto, esta librería es el puente entre el modelo de tráfico simulado y los algoritmos de aprendizaje implementados.

La clase principal del entorno es SumoEnvironment, la cual define la lógica de interacción del agente con la intersección semafórica. Cada paso en el entorno se ejecuta mediante el método step(action), que aplica una acción, cambiar el semáforo, la simulación avanza en SUMO por un intervalo de tiempo, y entrega una

tupla compuesta por el nuevo estado observado, la recompensa obtenida, y una bandera de finalización. La función `reset()` se encarga de reiniciar la simulación al estado inicial del episodio, cargando nuevamente el archivo de rutas y el archivo de red definidos.

Esta librería facilita la integración de diferentes estrategias de control a través del parámetro `Agent_type`, permitiendo usar diversos agentes. Además, dispone de una sección para retornar métricas clave como el número total de vehículos detenidos y el tiempo total de simulación, la velocidad de los vehículos, entre otras; son útiles para evaluar el rendimiento del sistema.

Gracias a esta arquitectura modular, el entrenamiento de agentes inteligentes sobre escenarios de tráfico urbano se simplifica notablemente, lo que convierte esta librería en una herramienta fundamental en este tipo de estudios, ya que ayuda a experimentar y comparar distintos esquemas de control semafórico basados en aprendizaje por refuerzo.

El entorno fue configurado con los parámetros de la Figura 4, definidos mediante la clase `SumoEnvironment` que se encuentra en la librería `sumo-rl`. Se estableció un tiempo de simulación de 100.000 pasos, un tiempo de decisión cada 5 segundos, y un tiempo en verde que va desde 5 a 60 segundos. Para la comparación se asignaron ciclos fijos de 38 segundos para cada fase verde. Esta línea base (*baseline*) se determinó de manera empírica mediante una campaña de observación directa *in situ* y aforos de los tiempos de fase reales en la intersección analizada de Valledupar; al representar la configuración operativa actual del entorno real, este ciclo fijo no ha sido optimizado, lo cual se reconoce explícitamente como una limitación metodológica de comparación frente a un óptimo teórico.

Para evaluar el desempeño de los modelos entrenados frente a este ciclo fijo, se ejecutó una única simulación de prueba independiente por cada escenario configurado. La demanda vehicular se parametrizó de forma determinista mediante flujos continuos uniformes definidos en los archivos de rutas (`.rou.xml`), empleando como unidad de medida los vehículos por hora (veh/h) por carril. Debido a que las tasas de llegada se programaron con intervalos fijos de inyección vehicular en el código XML, se prescindió del uso de semillas aleatorias estocásticas, garantizando condiciones de carga idénticas, estables y perfectamente reproducibles para todos los esquemas de control evaluados.

Figura 4. Código de configuración del entorno.

```
entorno = SumoEnvironment(
    net_file='k23c16.net.xml',
    route_file='traffic.rou.xml',
    use_gui=False,
    num_seconds=100000,
    delta_time=5,
    yellow_time=2,
    min_green=5,
    max_green=60,
    single_agent=True
)
```

2.4. Algoritmos

Dentro de los algoritmos clásicos de aprendizaje por refuerzo, Q-Learning y SARSA destacan por la eficiencia computacional y la amplia aplicabilidad que tienen en entornos discretos como el control de

semáforos. Sin embargo, presentan diferencias en el modo en que actualizan sus valores y en cómo se comportan durante el aprendizaje.

2.4.1 Q-Learning

Este es un algoritmo off-policy, es decir, aprende la mejor acción posible en cada estado, pero no necesariamente va a seguirla durante el entrenamiento. Su actualización se basa en la acción que maximiza el valor futuro mediante la ecuación:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)] \quad (1)$$

donde $Q(s, a)$ representa el valor al realizar la acción a en el estado actual s . Inicialmente todos los $Q(s, a)$ son igual a 0, α es la tasa de aprendizaje; controla cuánto influye la recompensa obtenida en la actualización del valor Q actual, su valor se encuentra entre 0 y 1. r es igual a la recompensa obtenida tras ejecutar una acción; γ es el factor de descuento, determina qué tanto valora el agente las recompensas futuras con las inmediatas, con un valor entre 0 y 1; y $Q(s', a')$ es el valor máximo estimado de Q en el siguiente estado s' , considerando todas las acciones posibles a' . Esta actualización permite al agente aprender progresivamente la mejor política a seguir en cada estado, incluso sin seguirla explícitamente durante el entrenamiento [14],[15].

2.4.2 SARSA

Por sus siglas State–Action–Reward–State–Action, por el contrario, es un algoritmo on-policy, que actualiza la función Q utilizando la acción que realmente fue tomada según la política del agente, y su ecuación de actualización viene dada por:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)] \quad (2)$$

donde $Q(s, a)$ representa el valor estimado de tomar la acción a en el estado s ; α es la tasa de aprendizaje, la cual regula qué tanto se ajusta el valor actual de Q en función de la experiencia reciente; r corresponde a la recompensa obtenida tras ejecutar dicha acción; γ es el factor de descuento que pondera las recompensas futuras en relación con las inmediatas. $Q(s', a')$ es el valor estimado de Q para el próximo estado s' y la acción a' que fue realmente ejecutada por el agente; al igual que Q-Learning, los valores de γ y α van de 0 a 1. Esta formulación refleja que SARSA actualiza sus valores basándose en las acciones efectivamente tomadas, lo que le permite adaptarse de forma más natural a comportamientos estocásticos o impredecibles, como los que pueden surgir en entornos dinámicos [15].

En este trabajo se implementa una versión del algoritmo conocida como True Online SARSA(λ), la cual utiliza funciones base de Fourier para representar el espacio de estados de manera más generalizable. En este enfoque, el valor de Q ya no se representa mediante una tabla, sino que se aproxima como una combinación lineal de características del estado:

$$Q(s, a) = \theta_a^T \phi(s) \quad (3)$$

donde θ_a es el vector de pesos asociado a la acción a , y $\phi(s)$ representa el vector de características obtenido a través de la base de Fourier. Esta aproximación permite que el agente generalice entre estados similares,

lo cual es especialmente útil en entornos continuos o de alta dimensión. La actualización de los pesos θ se realiza mediante aprendizaje por diferencias temporales con trazas de elegibilidad, según la siguiente ecuación:

$$\theta_a \leftarrow \theta_a + \alpha [\delta + Q(s, a) - Q_{old}] e_a - \alpha [Q(s, a) - Q_{old}] \phi(s) \quad (4)$$

donde $\delta = r + \gamma Q(s', a') - Q_{old}(s, a)$ es el error de TD (Temporal Difference) y e_a es la traza de elegibilidad para la acción a . Esta ecuación describe cómo se ajustan los pesos θ_a en función del error de predicción y de la traza de elegibilidad, permitiendo que el agente actualice su política de manera más precisa y estable, lo que resulta en un mejor desempeño en entornos complejos como cruces vehiculares urbanos [16].

2.4.3 Representación de los estados y estructura de la tabla Q

En el caso de agentes que utilizan una tabla Q explícita, como el Q-Learning, los valores asociados a cada par estado-acción se almacenan directamente durante el entrenamiento. Cada estado se representa como un vector de características que resume la situación actual del entorno semafórico.

La observación se genera mediante la función interna *observation_fn_default(self)* de la librería *sumo-rl*, la cual concatena diferentes componentes del entorno en un solo arreglo tipo *np.array*. A continuación, en la Figura 5 se muestra un fragmento del código.

Figura 5. Código que crea el vector de observación (estado).

```
def _observation_fn_default(self):
    phase_id = [1 if self.green_phase == i
                else 0 for i in range(self.
                num_green_phases)] # one-hot encoding
    min_green = [0 if self.
                 time_since_last_phase_change < self.
                 min_green + self.yellow_time else 1]
    density = self.get_lanes_density()
    queue = self.get_lanes_queue()
    observation = np.array(phase_id +
                          min_green + density + queue, dtype=np.
                          float32)
    return observation
```

Este vector se compone de los siguientes elementos:

- Fase activa (*phase_id*): codificación one-hot que indica qué fase verde está actualmente activa.
- Semáforo en tiempo mínimo cumplido (*min_green*): valor binario que vale 1 si el semáforo ya ha permanecido en verde el tiempo mínimo requerido.
- Densidad por carril (*density*): proporción de vehículos en cada carril, normalizada respecto a su capacidad.
- Cola por carril (*queue*): proporción de vehículos detenidos en cada carril, también normalizada.

Un vector típico de estado puede tener la siguiente forma $[0, 1, 0, 0, 1, 0.06, 0.03, 0.06, 0.04, 0.07, 0.06, 0.08, 0.07, 0.00, 0.00, 0.00, 0.01, 0.03, 0.02, 0.03, 0.03]$. Este vector se transforma internamente en una clave *hashable* para indexar la tabla Q. Cada entrada en la tabla tiene la forma:

(estado_vector): 0: Q0, 1: Q1, 2: Q2, 3: Q3

donde las claves representan las posibles acciones (fases *semafóricas*), y los valores Q_i son las estimaciones del valor esperado para cada acción en ese estado.

En contraste con el enfoque basado en tabla, el agente SARSA implementado en este trabajo utiliza una técnica de aproximación funcional basada en *True Online SARSA(λ)* con funciones base de Fourier. En este esquema, el estado observado no se utiliza como clave en una tabla, sino como entrada de una función de aproximación lineal:

$$Q(s, a) = \theta_a^T \phi(s) \quad (5)$$

donde $\phi(s)$ es un vector de características derivado del estado original mediante una transformación de Fourier, y θ_a es el vector de pesos que se ajusta durante el entrenamiento para cada acción a .

Esta formulación permite representar entornos continuos o de alta dimensionalidad sin necesidad de discretizar el espacio de estados, y favorece la generalización entre situaciones similares. Como resultado, el agente puede aprender una política eficaz en escenarios complejos sin necesidad de almacenar todos los posibles estados observados.

2.5. Función de recompensa

El entorno sumo-rl permite seleccionar entre varias funciones de recompensa predefinidas, diseñadas para optimizar distintos aspectos del tráfico vehicular. Estas funciones ofrecen diversos criterios que afectan directamente el comportamiento que el agente busca aprender. Las principales funciones disponibles en la librería son:

- *queue*: Minimiza la longitud total de las colas en los carriles controlados. Recompensa negativamente la acumulación de vehículos detenidos.
- *delay*: Se enfoca en minimizar el retraso medio de los vehículos respecto a un desplazamiento ideal sin interrupciones.
- *pressure*: Calcula la presión de la intersección como la diferencia entre los vehículos que entran y los que salen, buscando reducir desequilibrios de flujo.
- *diff-waiting-time*: Recompensa las acciones que disminuyen el tiempo total de espera acumulado por los vehículos.

Además de estas opciones, la librería permite definir funciones de recompensa personalizadas, lo que resulta útil para adaptar los objetivos del aprendizaje a contextos específicos, como podrían ser la priorización de vehículos de emergencia, prioridad a los peatones, entre otros. En este trabajo, se optó por utilizar la función de recompensa *diff-waiting-time* debido a que se busca minimizar los tiempos de espera e indirectamente también se reducirían las colas. Esta función evalúa la recompensa en cada paso temporal (t) como la diferencia entre el tiempo acumulado de espera en el instante anterior y el actual:

$$r_t = W_{\{t-1\}} - W_t \quad (6)$$

donde:

- W_t , es el tiempo total acumulado de espera de todos los vehículos en los carriles controlados en el instante (t)
- $W_{\{t-1\}}$, representa el valor en el instante anterior.

Esta aproximación incentiva directamente decisiones que reduzcan la congestión, premiando con valores positivos si el tiempo de espera disminuye y penalizando en caso de que suceda lo opuesto. En la Figura 6 se presenta la implementación en *sumo-rl* realizada en este proyecto.

Figura 6. Código de la recompensa basada en diferencia de tiempos de espera

```
def _diff_waiting_time_reward(self):
    ts_wait = sum(self.
        get_accumulated_waiting_time_per_lane
        ()) / 100.0
    reward = self.last_measure - ts_wait
    self.last_measure = ts_wait
    return reward
```

- *get_accumulated_waiting_time_per_lane()*, es una función que obtiene el tiempo total de espera por carril desde el entorno SUMO.
- La suma se normaliza dividiendo entre 100.0 para mantener valores de recompensa numéricamente estables.
- *last_measure*, es un atributo que almacena el valor anterior para calcular la diferencia.

Este enfoque guía al agente hacia políticas que prioricen la fluidez vehicular en cada decisión, adaptándose dinámicamente a las condiciones cambiantes del tráfico. En consecuencia, la métrica principal de evaluación del rendimiento cuantitativo adoptada en este estudio es el tiempo total de espera acumulado (medido en segundos paso a paso). Además, se seleccionó el número de vehículos detenidos como métrica secundaria, ya que proporciona una representación física, directa y altamente interpretable de la congestión y la longitud de las colas, facilitando la evaluación desde una perspectiva humana.

2.6. Configuración de agentes y fase de entrenamiento

La fase de entrenamiento fue configurada a través de un archivo *YAML*, donde se definieron múltiples instancias de agentes para comparar diferentes estrategias y parámetros de aprendizaje. En particular, se entrenaron agentes de tipo *Q-Learning* y *SARSA*, cada uno con variantes específicas en sus hiperparámetros como se indican en las Tablas 1-4, durante 10 episodios de simulación por experimento.

Tabla 1: Primera configuración de los parámetros del agente Q-Learning (QL_run_1)

Parámetro	Valor
Tasa de aprendizaje α	0.5
Factor de descuento γ	0.5
Política de exploración ϵ	1.0
Decay de ϵ	0.9

ϵ mínimo	0.1
-------------------	-----

Tabla 2: Segunda configuración de los parámetros del agente Q-Learning (QL_run_2)

Parámetro	Valor
Tasa de aprendizaje α	0.7
Factor de descuento γ	0.7
Política de exploración ϵ	1.0
Decay de ϵ	0.8
ϵ mínimo	0.1

Tabla 3: Primera configuración de los parámetros del agente SARSA (SARSA_run_1)

Parámetro	Valor
Tasa de aprendizaje α	5 x 10e-5
Factor de descuento γ	0.5
Factor de elegibilidad λ	0.999
Orden de Fourier	7
Política de exploración ϵ	0.01

Tabla 4: Segunda configuración de los parámetros del agente SARSA (SARSA_run_2)

Parámetro	Valor
Tasa de aprendizaje α	1 x 10e-4
Factor de descuento γ	0.7
Factor de elegibilidad λ	0.0999
Orden de Fourier	5
Política de exploración ϵ	0.01

Cada configuración fue ejecutada en simulaciones de 100.000 segundos sin interfaz gráfica (GUI: False), con tiempos mínimos y máximos de fase verde establecidos en 5 y 60 segundos respectivamente. El tiempo de luz amarilla fue fijado en 3 segundos y el intervalo de acción en 5 segundos Delta_time. Esta variedad de configuraciones permite evaluar el impacto de los valores sobre la eficiencia del aprendizaje y el desempeño del sistema de control metafórico inteligente.

El proceso de entrenamiento constó de 10 episodios independientes de 100.000 pasos cada uno, consolidando un horizonte total de 1.000.000 de pasos de simulación por experimento. La selección de este número de episodios se justificó mediante pruebas piloto previas, las cuales demostraron que los agentes alcanzaban una estabilidad en la recompensa acumulada y una clara convergencia en sus políticas a partir del quinto episodio; por lo tanto, los episodios restantes funcionaron como un margen de seguridad

estadístico para asegurar la explotación del modelo y descartar mínimos locales. El estado final obtenido al concluir el décimo episodio fue seleccionado como el modelo definitivo para la fase de evaluación.

3. RESULTADOS Y DISCUSIÓN

El desempeño de los agentes se evaluó en 3 condiciones de tráfico. Se utilizaron 4 configuraciones de los agentes como se mencionó en la sección Configuración de agentes y fase de entrenamiento: dos configuraciones de Q-Learning y dos configuraciones de SARSA. Estas se compararon con un agente de tiempos fijos, y el entrenamiento se realizó en un entorno con flujo intermedio constante de 1440 vehículos por hora.

Los tres escenarios de tráfico definidos fueron:

- **Flujo homogéneo por carril:** misma densidad de tráfico en todos los carriles.
- **Flujo no homogéneo:** se asigna menor flujo al carril más estrecho de la red vial, simulando condiciones reales donde ciertas vías tienen menor capacidad.
- **Carril sin flujo:** uno de los 4 carriles no recibe flujo vehicular.

Para cada experimento, se registró el tiempo de espera acumulado y el número de vehículos detenidos.

Desde una perspectiva de análisis, estas condiciones permiten evaluar el comportamiento de los algoritmos frente a distintos niveles de complejidad y variabilidad del entorno. En particular, el escenario homogéneo representa un caso controlado donde la carga vehicular es equilibrada, mientras que los escenarios no homogéneos y con carriles sin flujo introducen asimetrías que reflejan condiciones más cercanas a entornos reales.

En este ámbito, la comparación con un sistema de tiempos fijos resulta relevante, ya que permite identificar no solo mejoras en métricas de desempeño, sino también las limitaciones de los enfoques basados en aprendizaje por refuerzo; con esto, se espera que estos algoritmos presenten una mejoría en escenarios donde existe variabilidad en el tráfico gracias a su capacidad de adaptación.

3.1. Entrenamiento

Se registraron los 10 episodios que realizó cada agente y posteriormente se calculó la media con el fin de comparar los resultados obtenidos. De la Figura 7 se pueden observar los resultados de la media de los 10 episodios por agente, destacando el impacto en la reducción del tiempo de espera de ambas configuraciones de SARSA.

Cuando se observa el número de vehículos detenidos, ver Figura 8, sólo la segunda configuración de Q-Learning es el modelo que genera unas mayores detenciones que el ciclo fijo, mientras que SARSA en ambas configuraciones y Q-learning con la primera configuración logran mejorar la movilidad de los automóviles en la intersección.

3.2. Pruebas

Luego de realizar el entrenamiento y una vez almacenados los modelos, se procede con el ensayo de cada agente en entornos diferentes al de entrenamiento, para analizar la adaptabilidad de cada uno y determinar

si su comportamiento es óptimo para usarse en control de tráfico. El tiempo de simulación en cada caso fue de 10.000 segundos.

3.2.1. Tráfico homogéneo

La primera prueba se realizó con un tráfico homogéneo, es decir, con el mismo flujo vehicular en cada carril. y dado que este escenario es el más similar al entorno en el que fue entrenado, las pruebas se realizaron con: 720, 960, 1200, 1440, 1800 y 2400 vehículos por hora.

En las gráficas correspondientes al escenario con flujo de 720 vehículos por hora, ver Figuras 9 A1 y A2, se observa que el mejor desempeño fue obtenido por Q-Learning en su primera configuración, logrando

Figura 7. Tiempos de espera total. Cada línea representa un método de control de semáforos siendo: — semáforo de ciclos fijos, — Q-Learning primera configuración, — Q-Learning segunda configuración, — SARSA primera configuración, — SARSA segunda configuración.

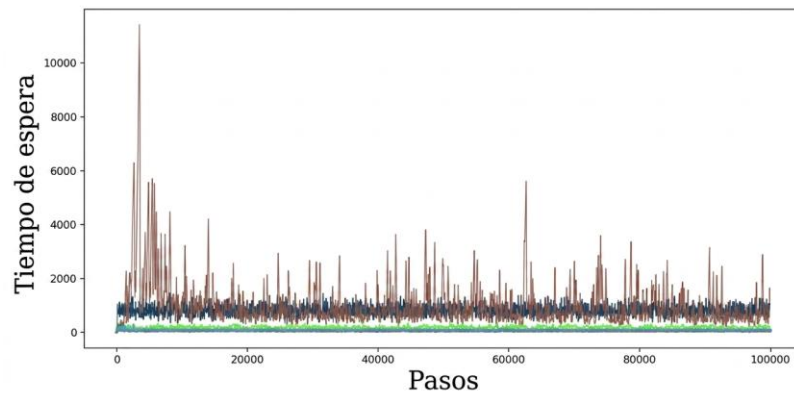
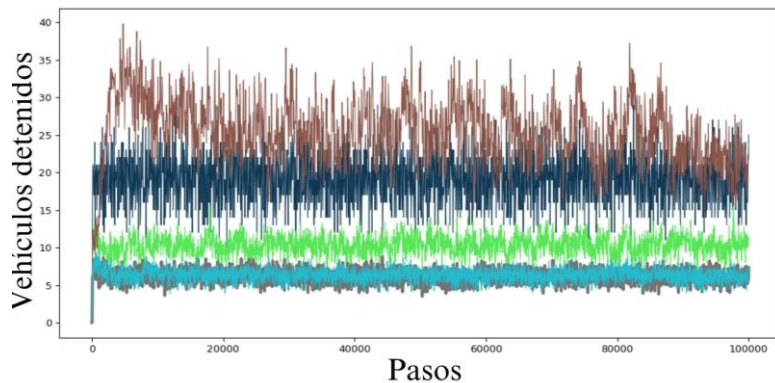


Figura 8. Número total de vehículos detenidos (VD). Cada línea representa un método de control de semáforos siendo: — semáforo de ciclos fijos, — Q-learning primera configuración, — Q-learning segunda configuración, — SARSA primera configuración, — SARSA segunda configuración.



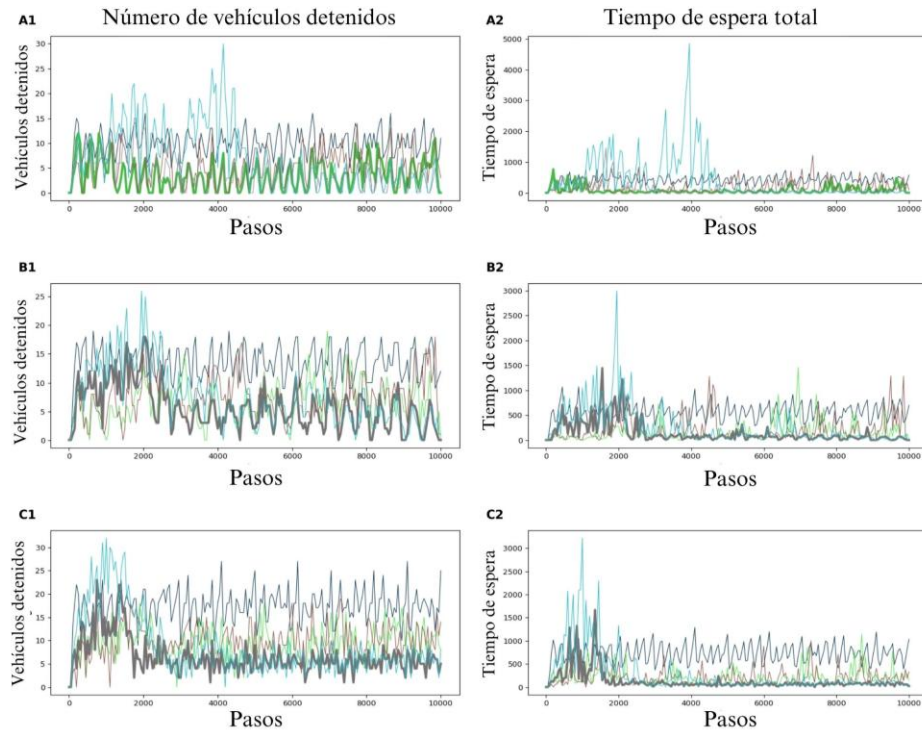
una mejora del 79,49% en el tiempo de espera promedio (TE) y del 63,53% en el número de vehículos promedio detenidos (VD) respecto a los semáforos de ciclos fijos, como se muestra en la Tabla 5. También se observa que SARSA en su segunda configuración presentó picos altos en algunos momentos, pero logró adaptarse conforme avanzaron los pasos de la simulación; sin embargo, esta inestabilidad inicial provocó que su tiempo de espera promedio empeorara un 4,11% en comparación con los ciclos fijos.

Por otra parte, en los escenarios con flujos de 960 y 1200 vehículos por hora, ver Figuras 9B1 y B2; C1 y C2, todos los modelos iniciaron con valores elevados debido al acoplamiento del tráfico. No obstante, SARSA 1 mostró nuevamente una destacada capacidad de adaptación, ver Tabla 5, alcanzando los valores más bajos de la comparativa y superando el 75% de mejora en el tiempo de espera para ambos experimentos. En cuanto a los demás agentes inteligentes, aunque mostraron curvas de aprendizaje más estables en el tiempo, sus promedios finales se situaron por debajo de SARSA 1, registrando de igual forma mejoras significativas en los tiempos de espera por encima del 50%.

Tabla 5: Comparativa del tráfico homogéneo con tres tasas de flujo vehicular: 720; 960; 1200 veh/h.

	720 vehículos por hora				960 vehículos por hora				1200 vehículos por hora			
	VD	Mejora (VD)	TE	Mejora (TE)	VD	Mejora (VD)	TE	Mejora (TE)	VD	Mejora (VD)	TE	Mejora (TE)
Ciclos Fijos	9,87	0,00 %	415,30	0,00 %	13,39	0,00 %	560,5	0,00 %	17,09	0,00 %	717,54	0,00 %
QL 1	3,60	63,53 % ▲	85,19	79,49 % ▲	6,32	52,80 % ▲	152,6	72,77 % ▲	9,31	45,52 % ▲	197,08	72,53 % ▲
QL 2	5,38	45,49 % ▲	188,43	54,63 % ▲	7,53	43,76 % ▲	204,16	63,58 % ▲	9,84	42,42 % ▲	213,8	70,20 % ▲
SARSA 1	3,89	60,59 % ▲	116,05	72,06 % ▲	5,38	59,82 % ▲	135,84	75,76 % ▲	6,57	61,56 % ▲	137,62	80,82 % ▲
SARSA 2	7,05	28,57 % ▲	432,38	-4,11 % ▼	7,56	43,54 % ▲	273,32	51,24 % ▲	8,84	48,27 % ▲	268,98	62,51 % ▲

Figura 9. Comparativa del tráfico homogéneo con tres tasas de flujo vehicular: A) 720; B) 960; C) 1200 vehículos por hora. Cada fila corresponde a una de estas tasas, donde la columna 1 muestra el número de vehículos estacionados y la columna 2 el tiempo total de espera. — semáforo de ciclos fijos, — Q-learning primera configuración, — Q-learning segunda configuración, — SARSA primera configuración, — SARSA segunda configuración.



En las Figuras 10 A1 y A2, correspondientes al flujo de 1440 vehículos por hora, de nuevo se puede observar que todos los agentes iniciaron con valores muy similares. Sin embargo, SARSA en su configuración uno mostró una mejor capacidad para adaptarse, alcanzando los valores más bajos con una media de tiempo de espera de 192,38, ver Tabla 6, lo que representa una mejora del 93,21% respecto a los ciclos fijos. Por otro lado, en la gráfica se aprecia que SARSA en su segunda configuración presentó picos que superaron los valores máximos de Q-Learning en las etapas tempranas; aunque a medida que la simulación avanzó logró estabilizarse, esto no fue suficiente para reducir sus promedios generales y obtener mejores resultados que las opciones de Q-Learning.

En el escenario con un flujo de 1800 vehículos por hora, ver Figuras 10 B1 y B2, SARSA configuración dos alcanzó los mejores resultados en comparación con los demás agentes, mejorando un 57,72% en los tiempos de espera. Además, en este escenario, ambas configuraciones de Q-Learning presentaron valores extremadamente altos, siendo superadas incluso por el control basado en semáforos de ciclos fijos y registrando una caída del rendimiento cercana al 100% en ambos agentes. Por último, en el escenario con 2400 vehículos por hora, ver Figuras 10 C1 y C2, todos los agentes se vieron superados por el semáforo de ciclos fijos. En este entorno de alta saturación, Q-Learning mantuvo un comportamiento más estable pero elevado, mientras que SARSA 1 se comportó mejor al inicio; aunque cerca de la mitad de la simulación alcanzó valores similares a los de Q-Learning, esto permitió que su desempeño final no fuera tan grave, empeorando un 70,23% en tiempo de espera como se ve en la Tabla 6, mientras que el de los demás agentes superó por lo menos en 20 puntos porcentuales el 100% de pérdida.

Figura 10. Comparativa del tráfico homogéneo con tres tasas de flujo vehicular: A) 1440; B) 1800; C) 2400 vehículos por hora. Cada fila corresponde a una de estas tasas, donde la columna 1 muestra el número de vehículos estacionados y la columna 2 el tiempo total de espera. — semáforo de ciclos fijos, — Q-learning primera configuración, — Q-learning segunda configuración, — SARSA primera configuración, — SARSA segunda configuración.

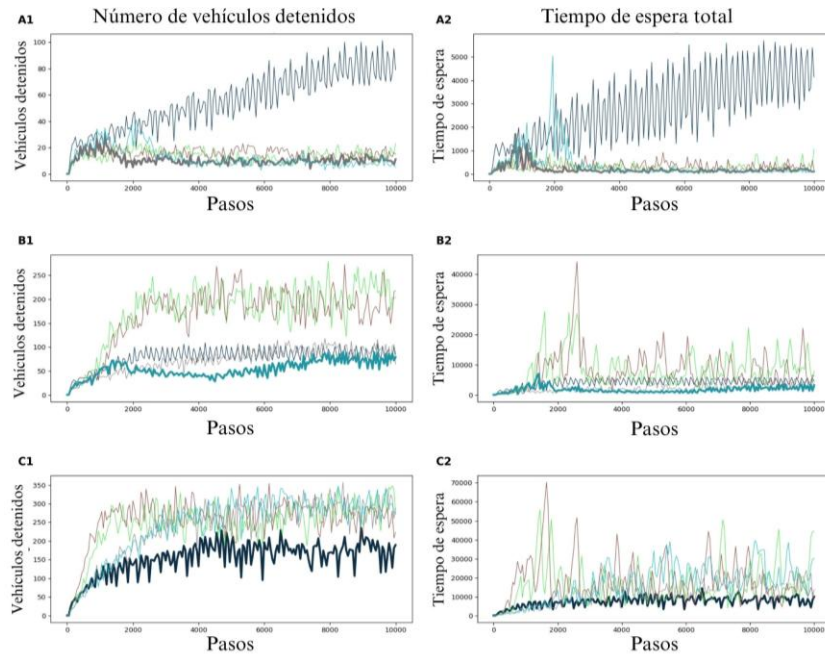


Tabla 6: Comparativa del tráfico homogéneo con tres tasas de flujo vehicular: 1440; 1800; 2400.

	1440 vehículos por hora				1800 vehículos por hora				2400 vehículos por hora			
	VD	Mejora (VD)	TE	Mejora (TE)	VD	Mejora (VD)	TE	Mejora (TE)	VD	Mejora (VD)	TE	Mejora (TE)
Ciclos Fijos	55,49	0,00 %	2832,33	0,00 %	78,35	0,00 %	4020,56	0,00 %	154,31	0,00 %	7460,41	0,00 %
QL 1	14,09	74,61 % ▲	273,36	90,35 % ▲	177,36	-55,82 % ▼	8508,55	-111,63 % ▼	246,42	-59,69 % ▼	16645,21	-123,11 % ▼
QL 2	15,54	71,99 % ▲	319,22	88,73 % ▲	167,28	-53,16 % ▼	7987,87	-98,68 % ▼	257,26	-66,72 % ▼	18111,65	-142,77 % ▼
SARSA 1	10,82	80,50 % ▲	192,38	93,21 % ▲	73,56	6,51 % ▲	2222,98	44,71 % ▲	236,12	-53,02 % ▼	12699,77	-70,23 % ▼
SARSA 2	12,7	77,11 % ▲	381,03	86,55 % ▲	54,21	44,53 % ▲	1700,03	57,72 % ▲	244,01	-58,13 % ▼	16453,08	-120,54 % ▼

3.2.2. Tráfico no homogéneo

En esta sección se realizaron pruebas con un tráfico diferente en el carril sur, ya que es el más angosto y puede concentrar un menor número de vehículos, simulando un escenario más realista debido a que es el menos concurrido. Las pruebas se realizaron con un flujo igual en los restantes tres carriles, este, norte y oeste, de la siguiente forma: El primer número indica los flujos originados en los carriles Norte, Este y Oeste, el segundo número indica el flujo originado en el carril sur. Las pruebas se realizaron con: (70,40); (100,50); (120,80); (140,70); (140,100); (190,120) vehículos por hora.

Al ver el comportamiento en la Figura 11 en conjunto con los valores promedio de la Tabla 7, se destaca el desempeño de SARSA 1 como el método más eficiente en todas las tasas de flujo vehicular, alcanzando las mayores reducciones tanto en vehículos detenidos como en tiempo de espera total, obteniendo mejoras de hasta un 81,80% en el escenario más denso.

Figura 11. Comparativa del tráfico no homogéneo con tres tasas de flujo vehicular: A) 70 y 40; B) 100 y 50; C) 120 y 80 vehículos por hora. Cada fila corresponde a una de estas tasas, donde la columna 1 muestra el número de vehículos estacionados y la columna 2 el tiempo total de espera. — semáforo de ciclos fijos, — Q-learning primera configuración, — Q-learning segunda configuración, — SARSA primera configuración, — SARSA segunda configuración.

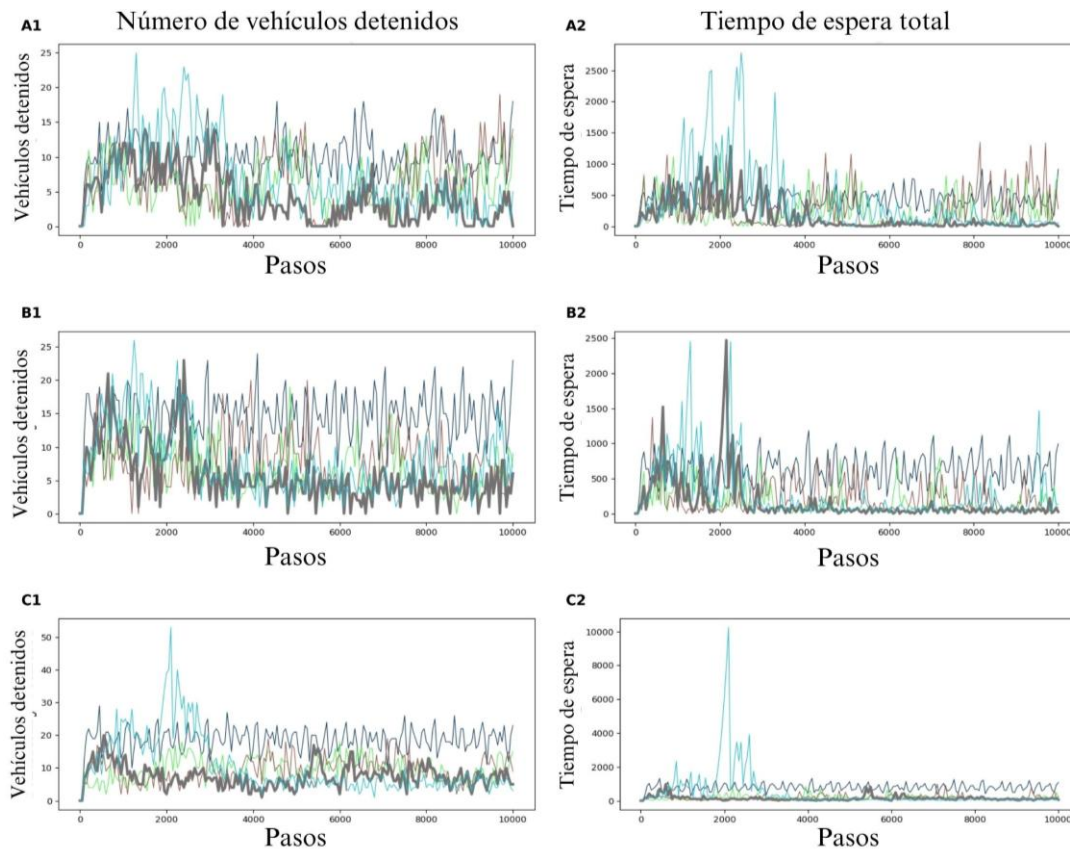


Tabla 7: Comparativa del tráfico no homogéneo con tasas de flujo vehicular: 70 y 40; 100 y 50; 120 y 80

	70 y 40 vehículos por hora				100 y 50 vehículos por hora				120 y 80 vehículos por hora			
	VD	Mejora (VD)	TE	Mejora (TE)	VD	Mejora (VD)	TE	Mejora (TE)	VD	Mejora (VD)	TE	Mejora (TE)
Ciclos Fijos	10,1	0,00 %	419,8	0,00 %	14,67	0,00 %	616,63	0,00 %	18,56	0,00 %	775,22	0,00 %
QL 1	5,98	40,79 % ▲	240,86	42,63 % ▲	6,72	54,19 % ▲	167,39	72,85 % ▲	9,74	47,52 % ▲	186,52	75,94 % ▲
QL 2	6,61	34,55 % ▲	276,61	34,11 % ▲	7,74	47,24 % ▲	217,59	64,71 % ▲	9,22	50,32 % ▲	178,59	76,96 % ▲
SARSA 1	4,44	56,04 % ▲	159,81	61,93 % ▲	5,45	62,85 % ▲	155,04	74,86 % ▲	7,62	58,94 % ▲	141,07	81,80 % ▲
SARSA 2	7,12	29,50 % ▲	396,6	5,53 % ▲	7,79	46,90 % ▲	291,97	52,65 % ▲	10,38	44,07 % ▲	480,52	38,02 % ▲

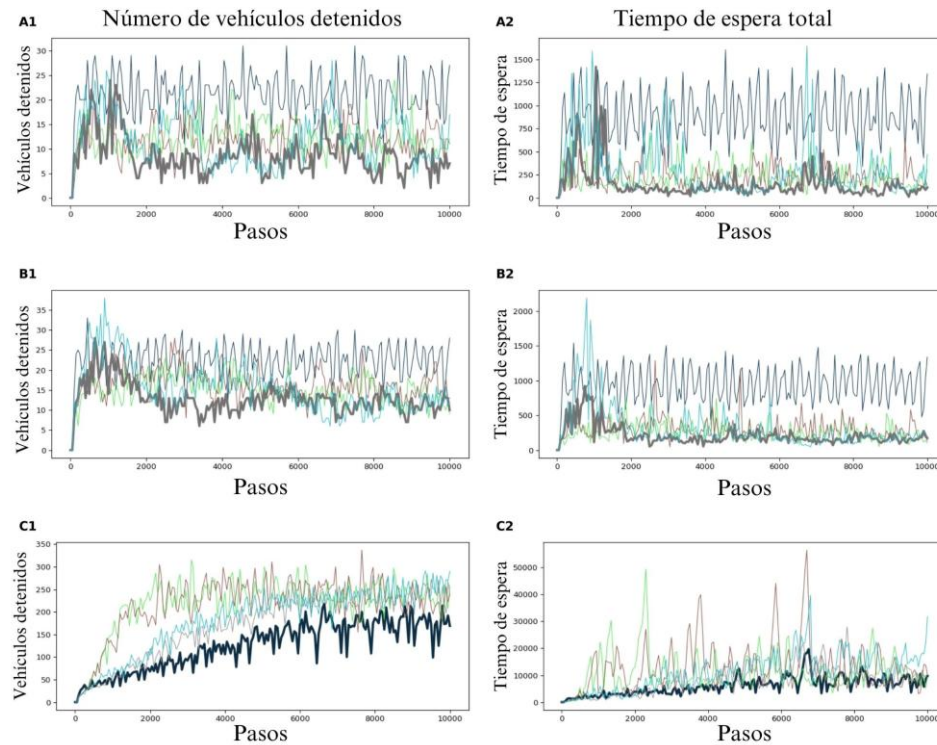
En las últimas tres configuraciones de flujo vehicular se observa un comportamiento similar en todos los agentes. Siguiendo la tendencia de los escenarios previos, SARSA 1 presenta una adaptación superior en los escenarios de 140 y 70; 140 y 100 veh/h; como se ve en la Tabla 8, este agente demuestra ser el más eficiente al lograr disminuir el tiempo de espera total con optimizaciones del 82,42% y 77,89%, respectivamente, logrando estabilizar los valores elevados registrados al comienzo de la simulación Figura 12 A1, A2, B1 y B2.

Por el contrario, en el escenario de máxima saturación de 190 y 120 veh/h, se genera un alto crecimiento en el rendimiento: todos los métodos basados en aprendizaje por refuerzo son ampliamente superados por el control de ciclos fijos, el cual mantiene métricas sustancialmente menores, Figuras 12 C1 y C2. En estas condiciones críticas, la ineficiencia se manifiesta de forma severa en todos los agentes; mientras SARSA 2 reporta el mayor volumen promedio de vehículos detenidos, con 181,72, como se aprecia en la Tabla 8.

Tabla 8: Comparativa del tráfico no homogéneo con tasas de flujo vehicular: 140 y 70; 140 y 100; 190 y 120

	140 y 70 vehículos por hora				140 y 100 vehículos por hora				190 y 120 vehículos por hora			
	VD	Mejora (VD)	TE	Mejora (TE)	VD	Mejora (VD)	TE	Mejora (TE)	VD	Mejora (VD)	TE	Mejora (TE)
Ciclos Fijos	21,03	0,00 %	881,22	0,00 %	22,24	0,00 %	925,87	0,00 %	125,7	0,00 %	6116,91	0,00 %
QL 1	12,54	40,37 % ▲	239,91	72,78 % ▲	15,03	32,42 % ▲	263,04	71,59 % ▲	217,66	-73,16 % ▼	11204,23	-83,17 % ▼
QL 2	11,77	44,03 % ▲	222	74,81 % ▲	16,52	25,72 % ▲	314,93	65,99 % ▲	216,21	-72,00 % ▼	12768,06	-108,73 % ▼
SARSA 1	8,7	58,63 % ▲	154,89	82,42 % ▲	12,51	43,75 % ▲	204,75	77,89 % ▲	171,55	-36,48 % ▼	8434,84	-37,89 % ▼
SARSA 2	12,18	42,08 % ▲	296,32	66,37 % ▲	15,57	29,99 % ▲	324,14	64,99 % ▲	181,72	-44,57 % ▼	10946,74	-78,96 % ▼

Figura 12. Comparativa del tráfico no homogéneo con tres tasas de flujo vehicular: A) 140 y 70; B) 140 y 100; C) 190 y 120 vehículos por hora. Cada fila corresponde a una de estas tasas, donde la columna 1 muestra el número de vehículos estacionados y la columna 2 el tiempo total de espera. — semáforo de ciclos fijos, — Q-learning primera configuración, — Q-learning segunda configuración, — SARSA primera configuración, — SARSA segunda configuración.



3.2.3. Carril sin flujo

En este escenario uno de los carriles está vacío, con el fin de simular horas pico donde todo el flujo viene principalmente de algunos carriles, mientras los otros se encuentran vacíos. El carril sin flujo es el que se origina en el oeste y se probaron 3 episodios diferentes en los que se tiene un flujo bajo, un flujo medio y un flujo alto.

De las Figuras 13 A1 y A2, correspondientes al escenario de tráfico bajo, se observa un comportamiento particular: el mejor desempeño en tiempo de espera total fue obtenido por Q-Learning 2, registrando una media de 103,18 y una mejora del 60,4% respecto a los semáforos de ciclos fijos. En este experimento, el agente con menores tiempos de retraso no coincide directamente con el que logra el menor número de vehículos detenidos, el cual fue SARSA 1 con un promedio de 2,1 a partir de los datos de la Tabla 9. Inicialmente se pensaba que ambas variables serían proporcionales; sin embargo, este fue el único escenario donde dicha relación lineal no se cumplió.

Aun así, consideramos que Q-Learning 2 tuvo el desempeño más acertado, ya que el factor utilizado para la recompensa del aprendizaje fue el *diff-waiting-time*, el cual busca específicamente reducir los tiempos de espera en la intersección.

Figura 13. Comparativa con uno de los carriles sin flujo con tres tasas de flujo vehicular: A) Bajo; B) Medio; C) Alto vehículos por hora. Cada fila corresponde a una de estas tasas, donde la columna 1 muestra el número de vehículos estacionados y la columna 2 el tiempo total de espera. Cada línea representa un método de control de semáforos siendo: — semáforo de ciclos fijos, — Q-learning primera configuración, — Q-learning segunda configuración, — SARSA primera configuración, — SARSA segunda configuración.

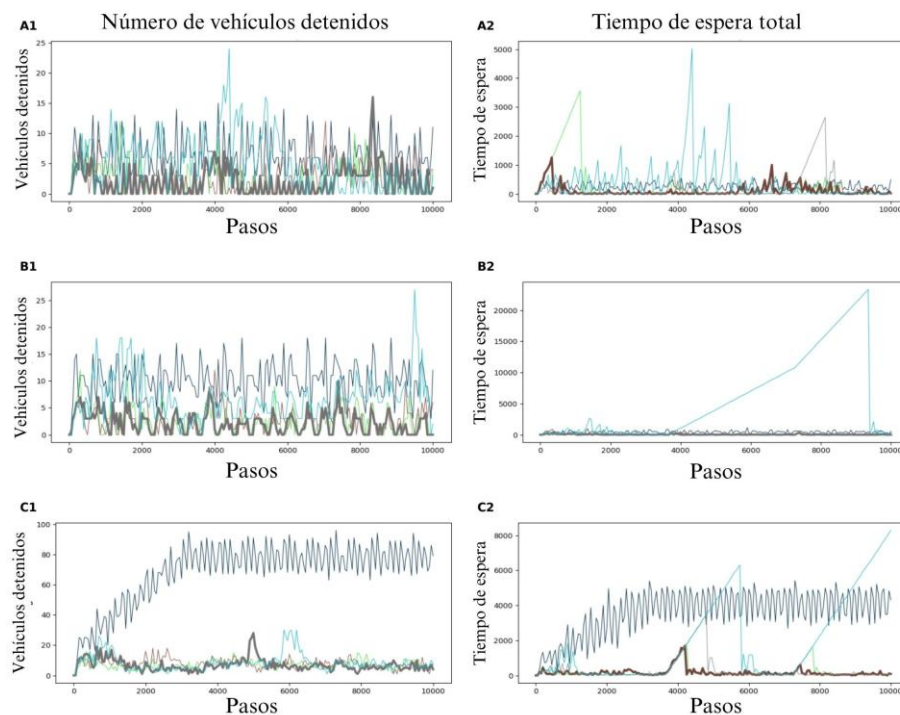


Tabla 9: Comparativa con uno de los carriles sin flujo con tres tasas de flujo vehicular: Bajo; Medio; Alto

	tráfico bajo				tráfico medio				tráfico alto			
	VD	Mejora (VD)	TE	Mejora (TE)	VD	Mejora (VD)	TE	Mejora (TE)	VD	Mejora (VD)	TE	Mejora (TE)
Ciclos Fijos	6,2	0,00 %	260,78	0,00 %	10,21	0,00 %	428,4	0,00 %	68,79	0,00 %	3595,93	0,00 %
QL 1	2,34	62,26 % ▲	271,82	-4,23 % ▼	2,58	74,73 % ▲	49,55	88,43 % ▲	6,89	89,98 % ▲	189,15	94,74 % ▲
QL 2	2,5	59,68 % ▲	103,18	60,43 % ▲	3	70,62 % ▲	76,1	82,24 % ▲	7,76	88,72 % ▲	163,78	95,45 % ▲

SARSA 1	2,1	66,13 % ▲	178,13	31,69 % ▲	2,2	78,45 % ▲	36,48	91,48 % ▲	6,84	90,06 % ▲	303,94	91,55 % ▲
SARSA 2	5,02	19,03 % ▲	403,87	-54,87 % ▼	6,79	33,50 % ▲	5701,06	-1230,78 % ▼	8,28	87,96 % ▲	1939,6	46,06 % ▲

Por lo tanto, este agente fue el que mejor cumplió con el objetivo principal planteado para el sistema inteligente. En el escenario de tráfico medio, el mejor rendimiento lo obtuvo SARSA en su primera configuración, ver Figuras 13 B1 y B2, consolidando un tiempo promedio de apenas 36,48 que equivale a un 91,48% de mejora. Por el contrario, SARSA en su segunda configuración quedó críticamente por detrás del resto; como se aprecia en la gráfica B2, su curva presentó una elevación alta que lo llevó a promediar 5701,06, lo que equivale a una pérdida drástica del -1230,78% frente al control clásico.

Finalmente, en el escenario de tráfico alto, ver Figuras 13 C1 y C2, la tendencia se invierte favorablemente para la mayoría de los agentes, logrando superar con amplios márgenes al semáforo de ciclos fijos. En estas condiciones de alta demanda, ambas configuraciones de Q-Learning demostraron una excelente robustez operativa, destacando QL 2 con la máxima optimización del tiempo de espera al alcanzar un 95,45% de mejora, seguido de cerca por QL 1 con 94,74% y SARSA 1 con 91,55%, mientras que SARSA 2 volvió a mostrar un comportamiento inestable con picos crecientes hacia el final de la simulación.

4. CONCLUSIONES

En este estudio se realizó una evaluación sistemática entre un sistema de control semafórico tradicional por ciclos fijos y uno basado en aprendizaje por refuerzo, utilizando los algoritmos Q-Learning y SARSA. A partir de los resultados experimentales, se encontró que las técnicas de aprendizaje reforzado superan de forma notable al control tradicional en la mayoría de los escenarios analizados, alcanzando reducciones en los tiempos promedio de espera de los vehículos que oscilan entre el 20% y más del 50%.

El análisis del comportamiento dinámico revela que el algoritmo SARSA consolida la mayor adaptabilidad y estabilidad operativa en entornos de complejidad intermedia. Específicamente, en los escenarios de tráfico moderado, SARSA alcanzó los niveles más altos de eficiencia, obteniendo 93,21% de mejora en el flujo uniforme de 1440 veh/h y hasta 81,80% en demandas no homogéneas, validando la ventaja de actualizar las políticas basándose en acciones efectivamente ejecutadas en el entorno. Por su parte, Q-Learning demostró una robustez superior y un comportamiento significativamente más estable frente a flujos vehiculares reducidos. En contextos de bajo flujo, Q-Learning minimizó eficazmente la aparición de picos de inestabilidad que afectaron a las configuraciones de SARSA durante las fases iniciales de la simulación.

No obstante, el rendimiento de las políticas inteligentes expone una alta dependencia del nivel de saturación de la infraestructura. Ante demandas extremas uniformes de 2400 veh/h, la intersección alcanza un punto crítico en el que la capacidad operativa del entorno se desborda. Bajo estas condiciones extremas de congestión, las colas en los accesos impiden que la optimización dinámica basada en la recompensa *diff-waiting-time* surta efecto, permitiendo que el control por ciclos fijos logre una mayor estabilidad en la gestión del tráfico, con lo cual un sistema híbrido sería de interés.

Como investigaciones futuras, se propone modelar dos o más intersecciones, implementar otros algoritmos de aprendizaje por refuerzo como Dyna-Q, DQN, A2C o TD3, comparar el desempeño de un solo agente versus un control basado en agentes con aprendizaje por refuerzo distribuido.

Limitaciones del Trabajo

Este estudio presenta algunas limitaciones que deben tenerse en cuenta para su aplicación práctica o trabajos futuros. En primer lugar, el análisis se redujo a una única intersección aislada, es decir, se omitió el análisis de las intersecciones vecinas y la dinámica de propagación de la congestión. Por otro lado, el modelo se entrenó utilizando datos de tráfico simulados y constantes. Tampoco se logró calibrar el sistema con datos históricos y reales de Valledupar, y se excluyeron factores cotidianos de la dinámica urbana, como la incidencia de conductas disruptivas (giros indebidos o cruces imprudentes), las decisiones de los conductores, el paso de peatones o los accidentes viales. Finalmente, la validez estadística del estudio es limitada debido a que solo se ejecutó una prueba por cada escenario.

REFERENCIAS

- [1] Ghanadbashi, S., y Golpayegani, F. (2021). "An Ontology-Based Intelligent Traffic Signal Control Model". IEEE, 2554-2561. <https://doi.org/10.1109/itsc48978.2021.9564962>
- [2] Jayanto, N. D., Satwika, I. P., Bayuwindra, A., Cahyono, R. T., y Husni, E. M. (2023). "Traffic Light Optimization using SUMO at the Samsat Intersection". IEEE, 1095-1100. <https://doi.org/10.1109/smarttechcon57526.2023.10391623>
- [3] Kekuda, A., Anirudh, R., y Krishnan, M. (2021). "Reinforcement Learning based Intelligent Traffic Signal Control using n-step SARSA". IEEE, 379-384. <https://doi.org/10.1109/icaais50930.2021.9395942>
- [4] William, I., Kozhevnikov, S., Sontheimer, M., y Chou, S. (2024). "Enhancing Urban Traffic Management in Taipei: A Reinforcement Learning Approach". IEEE, 1-6. <https://doi.org/10.1109/scsp61506.2024.10549687>
- [5] R. Zhang, A. Ishikawa, W. Wang, B. Striner y O. K. Tonguz, "Using Reinforcement Learning With Partial Vehicle Detection for Intelligent Traffic Signal Control", IEEE Transactions on Intelligent Transportation Systems, vol. 22, núm. 1, pp. 404–415, enero de 2021. <https://doi.org/10.1109/TITS.2019.2958859>
- [6] Deepika, M. S., Mahajan, D., Vibhashree, S. H., Kiran, K., Shenoy, P. D., y Venugopal, K. R. (2024). "Analysing Mobility Patterns: A Comparative Study of SUMO RandomTrips and Duarouter". IEEE, 1-6. <https://doi.org/10.1109/indiscon62179.2024.10744313>

- [7] M. Noaeen, A. Naik, L. Goodman, J. Crebo, T. Abrar, Z. Shakeri Hossein Abad, A. L. C. Bazzan y B. Far, “Reinforcement learning in urban network traffic signal control: A systematic literature review”, *Expert Systems with Applications*, vol. 199, art. 116830, 2022. <https://doi.org/10.1016/j.eswa.2022.116830>
- [8] M. Miletić, E. Ivanjko, M. Gregurić y K. Kušić, “A review of reinforcement learning applications in adaptive traffic signal control”, *IET Intelligent Transport Systems*, vol. 16, págs. 1269–1285, 2022. <https://doi.org/10.1049/itr2.12208>
- [9] S. Park, E. Han, S. Park, H. Jeong y I. Yun, “Deep Q-network-based traffic signal control models”, *PLOS ONE*, vol. 16, núm. 9, art. e0256405, sep. 2021. <https://doi.org/10.1371/journal.pone.0256405>
- [10] H. Wei, G. Zheng, V. Gayah y Z. Li, “Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation”, *ACM SIGKDD Explorations Newsletter*, vol. 22, núm. 2, págs. 12–18, diciembre de 2020. <https://doi.org/10.1145/3447556.3447565>
- [11] A. R. M. Jamil, K. K. Ganguly y N. Nower, “Adaptive traffic signal control system using composite reward architecture based deep reinforcement learning”, *IET Intelligent Transport Systems*, vol. 14, págs. 2030–2041, 2020. <https://doi.org/10.1049/iet-its.2020.0443>
- [12] O. Mejia, R. Ceballos, R. Torres and J. Hoyos, "Adaptive Navigation System for an Autonomous Vehicle in a Goal-Oriented Environment," *IEEE Latin America Transactions*, vol. 23, no. 10, pp. 848-855, Oct. 2025, <https://doi.org/10.1109/TLA.2025.11150628>
- [13] A. Haydari and Y. Yılmaz, “Deep Reinforcement Learning for Intelligent Transportation Systems: A Survey”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 11–32, Jan. 2022. <https://doi.org/10.1109/TITS.2020.3008612>
- [14] J. Clifton and E. Laber, “Q-Learning: Theory and Applications,” *Annual Review of Statistics and Its Application*, vol. 7, pp. 279–301, 2020. <https://doi.org/10.1146/annurev-statistics-031219-041220>
- [15] M. Corazza and A. Sangalli, "Q-Learning and SARSA: A Comparison between Two Intelligent Stochastic Control Approaches for Financial Trading," *SSRN Electronic Journal*, 2015. <https://dx.doi.org/10.2139/ssrn.2617630>
- [16] H. van Seijen, A. R. Mahmood, P. M. Pilarski, M. C. Machado, and R. S. Sutton, “True Online Temporal-Difference Learning,” *J. Mach. Learn. Res.*, vol. 17, no. 145, pp. 1–40, 2016. <https://www.jmlr.org/papers/v17/15-599.html>
- [17] Lucas N. Alegre, “SUMO-RL,” GitHub repository, 2019. [Online]. Available: <https://github.com/LucasAlegre/sumo-rl>