# Universidad Nacional de Loja

## Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

### Carrera de Ingeniería en Electrónica y Telecomunicaciones

# Técnicas Inteligentes para la Generación de Mapas de Ubicación de Señales Semafóricas que Minimicen el Tráfico en la Ciudad de Loja

**Trabajo de Titulación previo, a obtención del Título de Ingeniero en Electrónica y Telecomunicaciones**


**AUTOR:**

Mario Alexander Moreno González

**DIRECTOR:**

Ing. Marcelo Fernando Valdiviezo Condolo, Mg. Sc.


*Loja – Ecuador*

*2023*

**Certificación**

Loja, 18 de julio de 2023.

Ingeniero.

Marcelo Fernando Valdiviezo Condolo, Mg. Sc.
**DIRECTOR DE TRABAJO DE TITULACIÓN**

**CERTIFICO:**

Que he revisado y orientado todo proceso de la elaboración del Trabajo de Titulación denominado: **Técnicas Inteligentes para la Generación de Mapas de Ubicación de Señales Semafóricas que Minimicen el Tráfico en la Ciudad de Loja,** previo a la obtención del título de **Ingeniero en Electrónica y Telecomunicaciones,** de la autoría del estudiante **Mario Alexander Moreno González**, con **cédula de identidad Nro. 1105757304**, una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja para el efecto, autorizo la presentación para la respectiva sustentación y defensa.

Firmado electrónicamente por:
MARCELO FERNANDO
VALDIVIEZO CONDOLO

Ing. Marcelo Fernando Valdiviezo Condolo, Mg. Sc.
**DIRECTOR DE TRABAJO DE TITULACIÓN**

## Autoría

Yo, **Mario Alexander Moreno González**, declaro ser autor del presente Trabajo de Titulación y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja la publicación de mi Trabajo de Titulación en el Repositorio Digital Institucional – Biblioteca Virtual.

**Firma:** 


**Cédula de identidad:** 110575730-4

**Fecha:** 9 de noviembre de 2023

**Correo electrónico:** mario.a.moreno@unl.edu.ec

**Teléfono:** 2689-485 **Celular:** 0982875940

**Carta de autorización por parte del autor, para la consulta de reproducción parcial o total, y/o publicación electrónica del texto completo del Trabajo de Titulación**

Yo, **Mario Alexander Moreno González** declaro ser autor del Trabajo de Titulación **denominado: Técnicas Inteligentes para la Generación de Mapas de Ubicación de Señales Semafóricas que Minimicen el Tráfico en la Ciudad de Loja**; como requisito para optar por el Título de **Ingeniero en Electrónica y Telecomunicaciones**, autorizo al Sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos, muestre al mundo la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Digital Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Digital Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Titulación que realice un tercero.

Para constancia de esta autorización, suscribo en la ciudad de Loja, a los nueve días del mes de noviembre del dos mil veintitrés.

**Firma:** 


**Autor:** Mario Alexander Moreno González

**Cédula:** 110575730-4

**Dirección:** Amable María (Isla Genovesa e Isla Fernandina)

**Correo electrónico:** mario.a.moreno@unl.edu.ec

**Teléfono:** 2689-485 **Celular:** 0982875940

**DATOS COMPLEMENTARIOS:**

**Director del Trabajo de Titulación:** Ing. Marcelo Fernando Valdiviezo Condolo, Mg. Sc.

## Dedicatoria

Quiero expresar mi gratitud y dedicar este Trabajo de Titulación a mi querida Familia, a mis padres María y Pedro, a mis hermanas y hermanos: Rosa, Ana, María, Germania, Milton, Gerardo, Jonathan, Edisson y José, que me han enseñado que, con esfuerzo y dedicación se puede lograr grandes cosas, y ser una mejor persona cada día.

***Mario Alexander Moreno González***

**Agradecimiento**

Agradezco de manera sincera a Dios por su amor incondicional, por las bendiciones y oportunidades que me ha concedido diariamente, como también, por la sabiduría y la claridad mental que me ha otorgado para superar diversos obstáculos y para llevar a cabo este Trabajo de Titulación.

Quiero expresar mi profunda gratitud, a mis queridos padres María y Pedro, a mis hermanas y hermanos: Rosa, Ana, María, Germania, Milton, Gerardo, Jonathan, Edisson y José, por su apoyo y comprensión incondicional desde el inicio y culminación de mis estudios universitarios.

También quiero agradecer al director del presente Trabajo de Titulación, el Ing. Marcelo Fernando Valdiviezo Condolo, por su orientación y compromiso en la supervisión de todas las etapas llevadas a cabo para su cumplimiento. Su conocimiento y experiencia fueron fundamentales para desarrollar habilidades importantes y obtener un resultado exitoso.

¡Qué Dios los Bendiga!

***Mario Alexander Moreno González***

**Índice de contenidos**

**Índice de tablas:**

**Índice de figuras:**

**Índice de anexos:**

## 1. Título

# Técnicas Inteligentes para la Generación de Mapas de Ubicación de Señales Semafóricas que Minimicen el Tráfico en la Ciudad de Loja

## 2. Resumen

La movilidad urbana e inteligente se presenta como una alternativa, que apunta a reducir las emisiones contaminantes, optimizar la gestión de tránsito en las urbes a nivel mundial y mejorar la calidad de vida de todos los ciudadanos.

El presente Trabajo de Titulación, tomando en cuenta las consideraciones mencionadas anteriormente, tiene como propósito la creación de una herramienta, que permita realizar la optimización de la ubicación de las señales semafóricas en las intersecciones de las redes viales, para ello, se basa principalmente en la aplicación de técnicas metaheurísticas, como lo es, el modelamiento de un algoritmo genético, la implementación de una función para evaluar el coste de las soluciones y el uso de un simulador de microtráfico para simular los mapas de red viales.

La función de evaluación es aplicable al conjunto de soluciones que se obtienen mediante la ejecución del algoritmo genético, la finalidad es poder seleccionar la solución con mejor coste entre las demás. Dicha solución se utiliza para modificar el fichero de la red vial, que se simula mediante la herramienta de microtráfico SUMO (Simulation of Urban MObility). Cada una de las soluciones están dirigidas a minimizar el consumo de combustible, emisiones de gases contaminantes, duración de tiempo de viaje de los vehículos, y los tiempos de espera que los automotores deben permanecer frente a los semáforos en las redes viales de las ciudades.

***Palabras clave:*** *movilidad urbana, , SUMO, algoritmo genético, metaheurísticas, intersecciones, señales semafóricas.*

Abstract.

The concept of urban and intelligent mobility is presented as an alternative that reduces pollution, improves traffic management in cities worldwide, and improves citizens' quality of life.

The following research was conducted considering the considerations mentioned above. The purpose of the tool is to optimize traffic signal locations at intersections of road networks. To accomplish this goal, metaheuristic techniques are applied, such as modeling a genetic algorithm. It also includes the implementation of a function to evaluate the cost of solutions. It also includes the use of a microtraffic simulator to simulate road network maps.

Evaluation functions are applied to the set of solutions obtained by executing genetic algorithms. The purpose is to select the solution with the highest cost from the others. This solution modifies the road network file, which is simulated using the SUMO (Simulation of Urban MObility) micro-traffic tool. Each of the solutions aims to minimize fuel consumption, pollutant gas emissions, vehicle travel time, and motorist waiting times in front of traffic lights. This is on city roads.

**Keywords:** urban mobility, SUMO, genetic algorithm, metaheuristics, intersections, traffic signals.

## 3. Introducción

El crecimiento de las áreas urbanas se relaciona con la actividad económica que se concentra y desarrolla en estas localidades, esto provoca que la población rural se desplace hacia las ciudades en busca de nuevas oportunidades, tales como: empleo, salud y educación. Sin embargo, el aumento de la población en las ciudades ha generado un problema complicado de solucionar, como lo es, la movilidad vehicular. La falta de una adecuada planificación urbana, provoca que las ciudades presenten cada día mayor índice de congestión vehicular, lo que dificulta la fluidez del transporte, aumenta los tiempos de desplazamiento de los habitantes y desencadena un aumento de escenarios incómodos y contaminados para las personas que circulan por las urbes.

En décadas anteriores, las redes viales que se construían carecían de estudios previos para su desarrollo, lo que actualmente produce una limitante para la movilidad vehicular. Las señales semafóricas que se instalaban anteriormente, se basaban en los hábitos de los conductores, y la instalación de las señales semafóricas con este enfoque, permitía gestionar el pequeño parque automotor que circulaba por las ciudades en aquella época. Sin embargo, en la actualidad, el aumento del tráfico vehicular limita estas configuraciones. Por tal razón, es indispensable el estudio y desarrollo de herramientas que permitan reducir la congestión vehicular en los diferentes escenarios de las ciudades, especialmente en las horas de mayor concentración vehicular, de la mañana, tarde y noche.

La investigación sobre la adecuada gestión del tráfico vehicular a nivel mundial y en Latinoamérica es una preocupación constante, ya que es un problema que aumenta diariamente debido al crecimiento del parque automotor y la mala planificación en el desarrollo de las redes viales de las ciudades. En Ecuador, el Instituto Nacional de Estadística y Censos (2021) reporta un crecimiento anual promedio del 9,3% en el parque automotor, con un registro de 134 vehículos por cada mil habitantes. En la ciudad de Loja-Ecuador, se observa un constante aumento del flujo de vehículos y tráfico en las intersecciones de la urbe, proyectando un crecimiento vehicular para el año 2025 de 97.534 vehículos, como lo señalan Ogoño et al., (2020).

El objetivo principal de este proyecto es simular la distribución de los semáforos en la red vial haciendo uso de un software de simulación de tráfico que permita comparar la distribución semafórica actual de la ciudad de Loja-Ecuador con la distribución propuesta mediante la utilización de técnicas inteligentes.

Realizando un adecuado tratamiento de toda esta información, se pueden generar soluciones viables en un entorno controlado, lo que permite tomar decisiones fundamentadas y eficientes para la correcta gestión vehicular. Este proyecto busca mejorar la movilidad urbana y reducir los impactos ambientales asociados al transporte en la ciudad de Loja-Ecuador.

Para el desarrollo de este proyecto investigativo, se ha propuesto los siguientes objetivos, los cuales se mencionan a continuación:

**Objetivo General**

- Desarrollar una herramienta que permita la generación de mapas de ubicación de señales semafóricas mediante técnicas inteligentes para el estudio, aplicación y adaptación en los diferentes escenarios de vialidad en la ciudad de Loja.

**Objetivos Específicos**

- Investigar acerca del uso del simulador de microtráfico SUMO, su middleware, arquitectura, funcionamiento, y como puede ser utilizado para desarrollar herramientas que permitan disminuir el tráfico vehicular en las ciudades.
- Emplear técnicas inteligentes para diseñar modelos mediante situaciones de tráfico vehicular de la ciudad de Loja, utilizando el simulador de microtráfico SUMO con diferentes ubicaciones de las señales semafóricas.
- Analizar los resultados tanto del simulador de microtráfico como de la herramienta desarrollada para compararlos con base en los parámetros utilizados para su evaluación.

# 4. Marco teórico

La generación de herramientas para la gestión de la movilidad urbana forma parte de una creciente área de interés en la actualidad, como lo es la Inteligencia Artificial (IA), esto gracias a los beneficios que presenta para resolver problemas complejos de distintas ramas o enfoques. En este apartado trataremos ciertas propiedades de los semáforos, y cómo la Inteligencia Artificial está siendo utilizada para mejorar la movilidad en las grandes urbes, estudiando principalmente las técnicas inteligentes. Además, se realiza una caracterización acerca del simulador de tráfico SUMO, donde expondremos sus características primordiales y los motivos por los cuales ha sido elegido para esta investigación acerca de la generación de mapas de ubicación de señales semafóricas utilizando técnicas inteligentes.

## 4.1. Semáforos

Los semáforos son dispositivos de señalización luminosa que se pueden complementar mediante la utilización de señales y demarcaciones, estas se posicionan en las intersecciones de una red vial para mostrar indicaciones y controlar la circulación tanto de vehículos y peatones, esto con la finalidad de que puedan cruzar las intersecciones con un mínimo de dificultades, peligros y retrasos, como lo señala Valencia (2000).

Los semáforos más comunes que se instalan son de tres secciones, que pueden ser verticales u horizontales, situándose su conjunto de luces de arriba abajo o de izquierda a derecha respectivamente, y manteniendo siempre el siguiente orden: roja, amarilla y verde, como lo indica Moreno et al., (2021).

Según Toronto (2021) las señales que intervienen en la gestión de tráfico son dispositivos inteligentes que cumplen con la misión de permitir el paso a los peatones y conductores en las diversas intersecciones, además de informar a los usuarios sobre las normas que se deben de cumplir en determinada zona, advertir sobre posibles riesgos y precautelar en todo momento la seguridad de las personas y conductores que se desplazan por dicha área.

## 4.2. Tipos de Semáforos

Existen varios tipos de semáforos que se implementan en diferentes intersecciones y que cumplen determinados roles para realizar la gestión de la circulación vehicular, dependiendo del escenario en el cual sean instalados, entre ellos tenemos los siguientes:

### *4.2.1. Semáforos Peatonales*

Los semáforos peatonales son dispositivos dotados con señales luminosas y tienen la finalidad de informar y controlar el tránsito de las personas en las intersecciones. Las señales

tienen leyendas que indican que se puede pasar (PASE) o que hay que detenerse (ALTO) como lo indica Valencia (2000).

#### *4.2.2. Semáforos Vehiculares*

Los semáforos vehiculares están conformados por tres secciones que forman una sola unidad o semáforo, y dependiendo de la complejidad de la intersección se puede instalar módulos adicionales para realizar virajes. Este tipo de semáforos tienen tres colores de luces circulares, las cuales se encuentran instalados verticalmente y siguen el siguiente orden descendente: rojo, amarillo y verde, como lo señala El Reglamento Técnico Ecuatoriano INEN 004 PARTE 5 SEMAFORIZACIÓN.

#### *4.2.3. Semáforos de Prevención*

Este tipo de semáforos tiene la finalidad de advertir a los conductores acerca de posibles eventos o riesgos que se pueden presentar en las carreteras, son utilizados primordialmente en zonas escolares y en entradas o salidas de vehículos pesados. Además, permiten que los conductores puedan tomar decisiones anticipadas de seguridad en las vías, como lo indica Construcción y Señalización Vial en Quito (2023).

### 4.3. Sistema Semafórico

Según Hernández (2021), el sistema semafórico es un conjunto de dispositivos luminosos que trabajan de manera interconectada y coordinada para regular el tráfico vehicular y peatonal en las vías de las ciudades. Además, cada semáforo tiene diferentes características y propiedades, como el ciclo semafórico, que es el tiempo que transcurre al repetirse una misma fase en un grupo de semáforos después de haber cumplido una sucesión de acciones, este ciclo se repite continuamente y puede ser controlado de forma proporcional o mediante datos recolectados del entorno.

En cambio, la fase se refiere a cada una de las divisiones del ciclo en donde la configuración de colores de los semáforos permanece constante, así las fases se delimitan en la vía cuando existe un cambio de derecho de paso, es decir, cuando un movimiento peatonal o vehicular es detenido y otro inicia, como lo indican (Ruiz, 2015; Valencia, 2000).

### 4.4. Grupos de Señales Semafóricas

Actualmente para realizar una gestión eficiente del tráfico vehicular que cursa por las diferentes intersecciones de las ciudades, se hace uso de dos grupos de señales semafóricas, las cuales son:

### *4.4.1. Semáforos de Tiempo Fijo*

Los semáforos de control de tiempo fijo tienen un programa preestablecido para cambiar la duración del ciclo, intervalo, secuencia y desfasamiento del semáforo, esto es una limitante puesto que pueden llevar a tiempos de espera innecesarios para los vehículos en la intersección, especialmente durante los lapsos de baja concurrencia de automotores. Este tipo de semáforos son principalmente implementados en zonas que controlan una baja tasa de tráfico, volumen mínimo de peatones y donde se tiene experiencia sobre accidentes, como lo señala Franco (2014).

### *4.4.2. Semáforos de Tiempo Actualizado*

Este tipo de semáforos implementan sensores de tráfico, software avanzado de monitoreo e infrarrojos para realizar el conteo de vehículos y ajustar la duración de cada fase de los ciclos semafóricos de manera automática y en tiempo real, todo esto en función de la carga de tráfico vehicular, como lo indica Valencia (2000).

## 4.5. Inteligencia Artificial en el Transporte

Según Rouhiainen (2018) la inteligencia artificial tiene la capacidad de automatizar un gran número de procesos en diversos sectores, dado que es la capacidad que poseen las máquinas para ejecutar algoritmos y aprender de los datos suministrados para la toma de decisiones, de manera idéntica a como lo realizan los seres humanos.

En gran parte la Inteligencia Artificial (IA) está siendo implementada en el sector del transporte, ya que se utiliza para la modelación de diferentes rutas, análisis de patrones de tráfico y, al utilizar la Inteligencia Artificial junto con sensores, cámaras y dispositivos electrónicos se puede crear sistemas de transporte inteligente que contribuyan a detectar y prevenir situaciones de riesgo para los peatones y conductores, como lo indica Pabon et al., (2023).

### *4.5.1. Técnicas Metaheurísticas*

Las técnicas metaheurísticas han demostrado ser una herramienta efectiva para encontrar soluciones a problemas complejos en poco tiempo, esto gracias a la exploración de un conjunto de soluciones mediante un proceso inteligente, como lo señala Guananga et al., (2023).

Según Gutiérrez et al., (2016), a diferencia de otras técnicas de optimización, las técnicas metaheurísticas no necesitan de un modelo matemático riguroso del problema, en su lugar se trabaja con reglas generales para guiar el proceso de exploración para descubrir las soluciones.

Las técnicas metaheurísticas tales como la inteligencia de enjambre y los algoritmos evolutivos, trabajan como alternativas eficientes a los métodos de optimización habituales para resolver problemas de diferentes sectores, que pueden ser tanto en: salud, transporte, educación, producción agrícola, militar y muchos más, como lo manifiesta Jamal et al., (2021).

Entre algunos de los ejemplos de técnicas metaheurísticas tenemos los siguientes: algoritmos genéticos, recocido simulado, optimización por enjambre de partículas, optimización por colonia de hormigas, búsqueda tabú y evolución diferencial, los cuales se basan principalmente en procesos naturales y comportamientos sociales, como lo indica Abarca (2018).

## 4.6. Algoritmos Genéticos

Los Algoritmos Genéticos son una técnica de optimización inspirada en la teoría de la evolución de Charles Darwin, los cuales se modelan para resolver diferentes problemas en función de su complejidad, la finalidad de este tipo de algoritmos es descubrir la mejor solución dentro de un conjunto de posibles soluciones para optimizar un problema, como lo señalan (Yarasca et al., 2021; Morán, 2021).

Para realizar el proceso de optimización los Algoritmos Genéticos operan con una población aleatoria de soluciones, donde cada solución representa un punto de búsqueda en el espacio de las soluciones posibles a un problema. El desempeño de una solución se evalúa según una función de coste. Esta función permite ordenar de mejor a peor las soluciones de la población en un continuo proceso de adaptación, como lo indica Valencia (1997). Además, el proceso de búsqueda de soluciones de mejor calidad para el problema se realiza mediante un proceso iterativo, es decir, en cada iteración el Algoritmo Genético aplica probabilísticamente operadores genéticos de selección, cruce y mutación a la población actual, y el criterio de parada para el proceso iterativo del Algoritmo Genético usualmente se establece mediante un número determinado de iteraciones, tiempo límite de ejecución, una cota de calidad en los valores de coste y la detección de una condición de convergencia, como lo manifiesta Abarca (2018).

### *4.6.1. Términos de un Algoritmo Genético*

A continuación, se detalla algunos de los términos comúnmente utilizados cuando se trabaja con Algoritmos Genéticos, que son importantes de conocer en el desarrollo de esta investigación:

- Población: la población hace referencia a un conjunto de soluciones generadas de manera aleatoria, que constituyen las posibles soluciones al problema, está población deber ser lo suficientemente grande para que garantice una diversidad en las

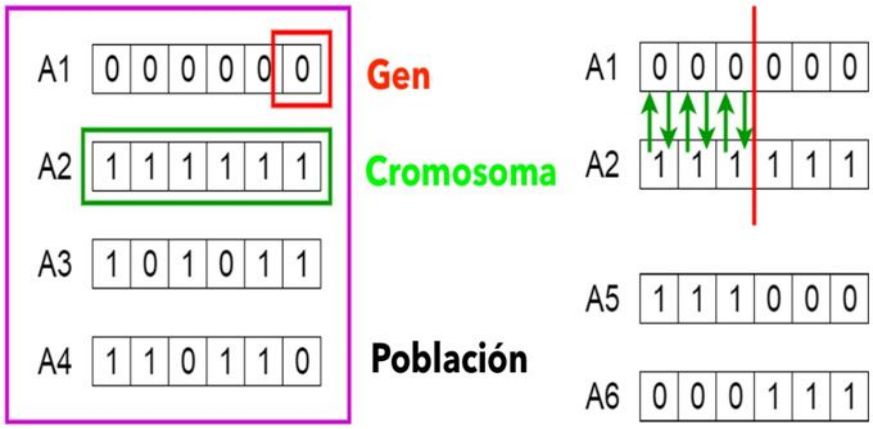
soluciones.

- Cromosoma: una solución codificada en forma binaria se denomina cromosoma, a lo largo de este trabajo se conoce a los cromosomas como soluciones.
- Genotipo: se denomina genotipo al conjunto completo de soluciones que permiten su tratamiento mediante el uso de un computador.
- Gen: es la posición de cada parámetro dentro de una solución.
- Alelo: se conoce como alelo, al valor que toma un parámetro dentro de una solución.
- Operador genético: se encarga de alterar las propiedades genéticas de la población de soluciones y de asegurar soluciones con nuevas características.
- Iteración: las soluciones de la población experimentan un proceso repetitivo de evolución, lo que permite obtener una nueva población de soluciones mediante la utilización de operadores genéticos.
- Coste: es una función que permite evaluar cada una de las soluciones al problema y brindar un valor de coste de dichas soluciones.

En la Figura 1 se presenta algunos de los términos empleados en la representación de un algoritmo genético.

**Figura 1.**

*Términos comunes en un Algoritmo Genético*




*Nota*. Los términos básicos de los algoritmos genéticos son: Gen, cromosoma y población. imagen tomada de Algoritmos genéticos: Funcionamiento, Pasos y Aplicaciones (2018).

### *4.6.2. Operadores Genéticos*

Un Algoritmo Genético utiliza operadores genéticos de selección, cruce y mutación para generar una nueva población a partir de una población existente. A continuación, se describen los operadores que intervienen en cada una de las etapas de evolución de los Algoritmos Genéticos.

### *4.6.3. Selección*

El proceso de selección escoge las soluciones más aptas para que existan en la siguiente población. La selección compara el coste de una solución en relación con otras soluciones y decide qué solución pasa a la siguiente población. A través de la selección, las "buenas soluciones" son beneficiadas para avanzar con alta probabilidad, mientras que las "malas soluciones" avanzan a la siguiente generación con baja probabilidad. La presión de selección es el grado en que se favorece a las mejores soluciones: cuanto mayor es la presión de selección, más se favorece a las mejores soluciones, como lo señala Jamshidi et al., (2017).

Un algoritmo genético puede utilizar diversos métodos para seleccionar aquellas soluciones que deben conservarse en la siguiente población. A continuación, se describe algunos de los métodos que son más utilizados:

- La selección por torneo elige n soluciones de manera aleatoria para que compitan entre sí, aquella solución que presente un mejor coste en la evaluación pasa a la siguiente etapa del algoritmo, dicho proceso se repite hasta haber alcanzado la cantidad de soluciones deseadas, como lo señala González (2020).
- La selección proporcional normaliza los valores de coste de todas las soluciones de la población y asigna dichos valores como probabilidades para que sus respectivas soluciones sean seleccionadas, como indica Kicinger et al., (2005).
- La selección por rueda de ruleta es un método en el que los individuos se eligen en función de su probabilidad proporcional a su valor de coste, y funciona idénticamente como una ruleta, como lo manifiesta Rawat et al., (2022).
- La selección de elitismo permite mejorar el rendimiento de la selección por rueda de ruleta, y basa su funcionamiento en asegurar que una solución elitista de una población siempre se propague a la próxima población automáticamente, como lo señala Katoch et al., (2021).
- El método de selección por rango ordena primero todas las soluciones de la población según su coste y, luego asigna el valor de coste a cada solución de acuerdo con aquella clasificación, como lo indica Joshi, (2021).

### *4.6.4. Cruce*

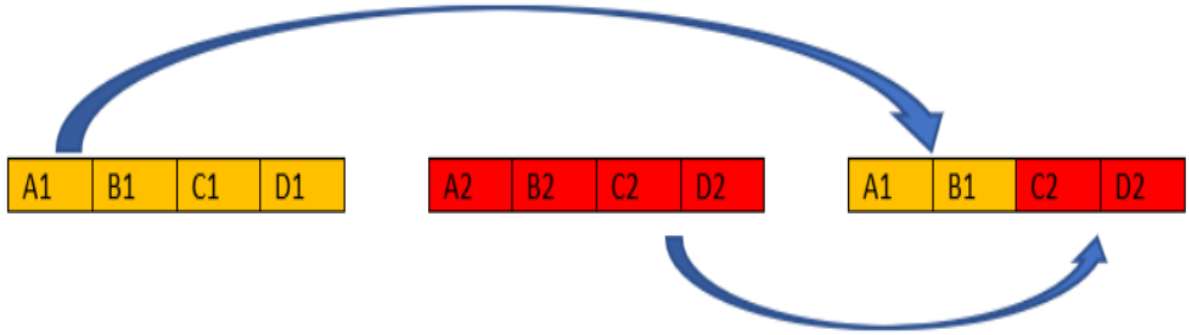
En un Algoritmo Genético, el operador de cruce, también llamado operador de recombinación o en inglés crossover, es uno de los métodos para generar nuevas soluciones en una población. Este operador trabaja seleccionando dos soluciones de la población y mezclando su material genético para crear nuevos descendientes, como lo señala Canales (2021).

A continuación, se describen los tipos de operadores de cruce que se suelen utilizar con frecuencia, estos son:

- Cruce en un punto: este operador selecciona un punto de cruce en una posición aleatoria de la cadena de genes de las dos soluciones a cruzar, una de las soluciones transfiere toda su cadena de genes anterior hasta ese punto, mientras que la otra solución contribuye toda su cadena de genes a partir de ese punto, con esto se logra producir una nueva descendencia a partir de la cadena de genes de las soluciones seleccionadas, como lo señala Aladdin et al., (2023). El proceso de cruce en un punto se puede apreciar en la Figura 2.

**Figura 2.**

*Operador de cruce en un punto*

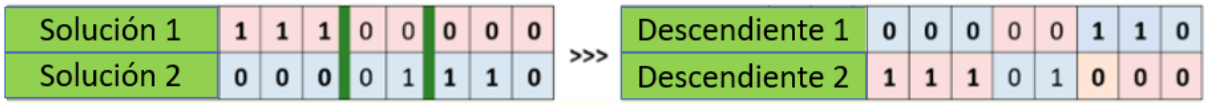



*Nota*. El operador de cruce en un punto, es el responsable de realizar la operación de cruce en un lugar específico para crear una nueva solución, imagen tomada de González (2020).

- Cruce en dos puntos: En este tipo de cruce, existen dos puntos de cruce en donde las secciones de las soluciones entre dichos puntos se intercambian, como lo indica Zainuddin et al., (2020). El proceso del operador de cruce en dos puntos se muestra en la Figura 3.

**Figura 3.**

*Operador de cruce en dos puntos*




*Nota*. En la imagen anterior se puede visualizar el proceso que realiza el operador de cruce en dos puntos para crear descendientes, imagen tomada de Aladdin et al., (2023).

- Cruce multipunto: este tipo de operador utiliza n puntos de corte en las soluciones y luego intercambia una parte de cada cadena de genes de las soluciones seleccionadas para crear descendencia y finalmente forma la cadena de genes de los nuevos descendientes, como lo indica Kumari et al., (2022). El proceso del operador de cruce multipunto punto se muestra en la Figura 4.

**Figura 4.**

*Operador de cruce multipunto*


*Nota.* El operador de cruce multipunto realiza una selección aleatoria de varios puntos para realizar la combinación del material genético, imagen tomada de Fonseca (2018).

- Cruce uniforme: en el operador de cruce uniforme la solución no puede dividirse en segmentos, es decir, se debe tratar cada gen por separado de la solución, por ende, cada gen del descendiente se forma de manera aleatoria de los genes de las dos soluciones seleccionadas para crear descendencia, como lo señala Katoch et al., (2021). El proceso que realiza el operador de cruce uniforme se muestra en la Figura 5.

**Figura 5.**

*Operador de cruce uniforme*


*Nota.* El operador de cruce uniforme permite modificar los genes de las soluciones de manera en que la descendencia tiene la misma probabilidad de pertenecer a uno u otro antecesor, imagen tomada de Crossover in Genetic Algorithm (2019).

### *4.6.5. Mutación*

El proceso de mutación se realiza con la finalidad de mantener la diversidad de la población de soluciones, evitar la convergencia prematura y encontrar la mejor solución. La

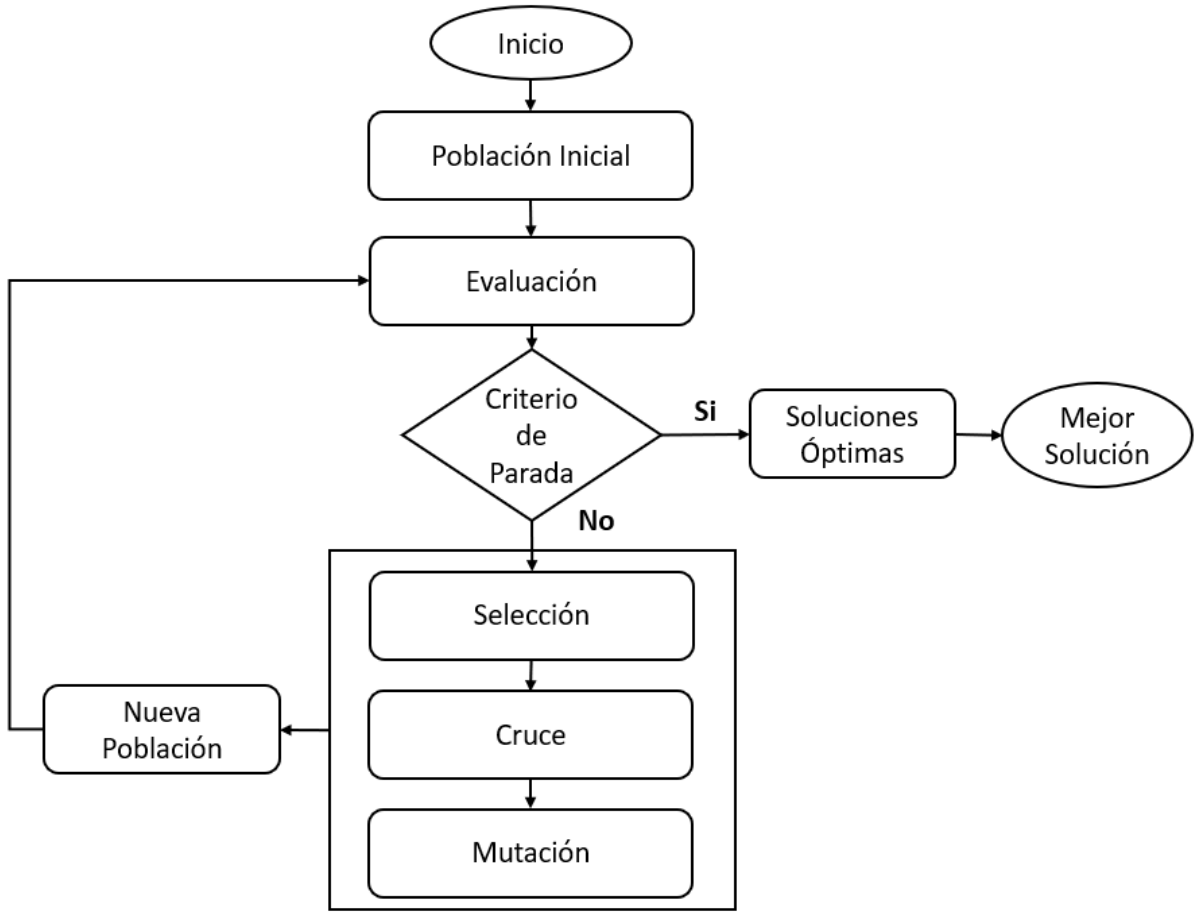
mutación dentro del algoritmo genético produce una variación aleatoria en el gen de una solución, dicho proceso se ejecuta con una probabilidad pm (probabilidad de mutación) que se fija antes del proceso de optimización. Para cada solución se calcula un valor aleatorio entre 0 y 1, el cual se compara con la probabilidad de mutación, y si este valor aleatorio es menor o igual que la probabilidad de mutación el gen de la solución mutará, con lo señala (Fonseca, 2018; Jamshidi et al., 2017).

#### *4.6.6. Estructura de un Algoritmo Genético*

Inicialmente un Algoritmo Genético parte generando una población de soluciones de manera aleatoria, luego las soluciones de la población son evaluadas mediante una función con la finalidad de obtener el valor de coste que poseen las soluciones para resolver el problema, posteriormente si el criterio de parada establecido para el algoritmo genético se cumple, se detiene el algoritmo y se presenta la mejor solución de la población actual, en caso de no cumplir el criterio de parada, se crea una nueva población aplicando los operadores genéticos de selección, cruce y mutación, la nueva población se volverá a evaluar, generando así, un proceso iterativo que finalizara cuando se cumpla con el criterio de parada, como lo indica Guerra et al., (2013). La estructura de un Algoritmo Genético se puede apreciar en el siguiente diagrama de flujo de la Figura 6.

**Figura 6.**

*Diagrama de flujo para un algoritmo genético*




*Nota*. La imagen presenta un diagrama de flujo que ilustra la secuencia de operaciones necesarias para alcanzar una solución óptima mediante la ejecución del algoritmo genético. Este diagrama es útil para entender el proceso y las etapas implicadas en la ejecución de un algoritmo genético, lo cual es fundamental tener en mente para su correcta implementación.

### 4.7. Introducción a SUMO (Simulation of Urban MObility)

El entorno de simulación que se utiliza en este trabajo de investigación, se conoce como Simulation of Urban MObility (SUMO), el cuál es un simulador de tráfico microscópico en donde cada vehículo se describe explícitamente mediante un identificador, esto permite tener un registro de todos los atributos relevantes del vehículo, tales como tamaño, peso, diámetro y propiedades de salida y llegada, tal como lo indica Srivstava et al., (2020).

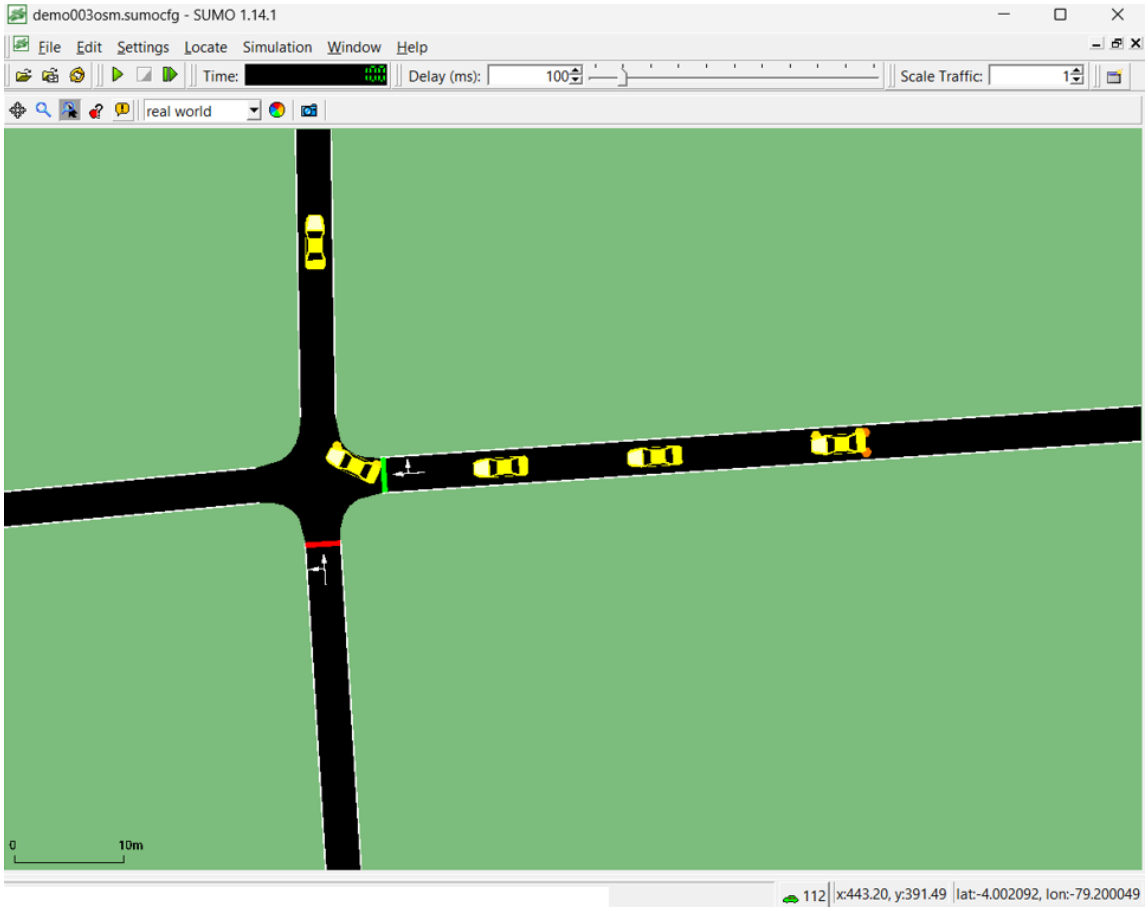
SUMO también incluye una amplia variedad de herramientas de soporte para tareas, tales como la visualización, la búsqueda de rutas, la importación de redes y el cálculo de emisiones contaminantes, además, es una herramienta de simulación de tráfico gratuito y de código abierto que permite simular sistemas de tráfico intermodal con vehículos, transporte

público y peatones, de la misma manera proporciona diversas interfaces de programación de aplicaciones (APIs), que permiten controlar las simulaciones en tiempo real, tal como se señala en Eclipse SUMO - Simulation of Urban Mobility (2022).

Una de las principales características que hacen que el simulador SUMO sea ampliamente utilizado en diferentes investigaciones, es su interfaz gráfica TraCI (Traffic Control Interface) que facilita la comunicación externa y la visualización de la simulación en tiempo real. Esta interfaz utiliza una arquitectura cliente-servidor en donde el cliente se puede programar en el lenguaje de programación Python, permitiendo consultar el peso de las variables de los objetos simulados y modificar el comportamiento de la simulación, como lo indica Behrisch et al., (2011). La interfaz gráfica del entorno del Simulador SUMO se muestra en la Figura 7.

**Figura 7.**

*Interfaz gráfica del simulador SUMO*




*Nota*. En la Figura 7 puede observar la interfaz gráfica del entorno del Simulador SUMO y el mapa de red vial.

Los archivos de configuración de las simulaciones en SUMO, la topología de las redes viales y las rutas de los vehículos están en formato XML, este formato permite leer y modificar fácilmente los datos de estos archivos, ya sea mediante software o de manera manual, esto facilita la aplicación de nuevas soluciones para la gestión del tráfico vehicular, como lo señalan (Coloma, 2019; Hernández, 2021).

## 4.8. Topología de Red Vial en SUMO

El simulador de microtráfico SUMO, necesita un archivo de red con extensión XML (net.xml) que debe estar previamente cargado y establecido en el archivo de configuración, este archivo de red, contiene la información y describe la parte que tiene relación con el tráfico de un mapa de red obtenido mediante la web de OpenStreetMap, así como de las carreteras e intersecciones por donde circulan los vehículos que forman parte de la simulación. El archivo de red dentro del entorno de SUMO está formado por nodos que también se denominan cruces, mismos que representan las intersecciones, y bordes que denotan a las carreteras.

La mayoría de las intersecciones que se encuentran modeladas en un archivo de red, o solo algunas de estas pueden contar y estar controladas por un semáforo o sistema semafórico dependiendo de la complejidad del diseño de la intersección. Dado el caso de que los cruces se encuentren controlados, el conjunto o sistema semafórico contará con un programa establecido de funcionamiento, el cual se encarga de indicarle las instrucciones que debe de seguir durante la simulación.

### *4.8.1. Semáforos en SUMO*

Los semáforos que se establecen en el archivo de red con extensión net.xml para ejecutar la simulación en SUMO, se generan inicialmente con la herramienta netconvert y netgenerate, al extraer el mapa de red desde OpenStreetMap, los programas calculados difieren de los que se encuentran en la realidad y se generan con ciclos fijos, para solucionar este problema, existe la opción de cargar a la configuración de SUMO archivos con definiciones de programas adicionales.

El archivo que almacena toda la información acerca de la lógica de los semáforos se conoce como tlLogic, en la Figura 8 se muestra la estructura y los atributos que se utilizan dentro del elemento tlLogic.

**Figura 8.**

*Estructura del archivo tlLogic con extensión XML*


*Nota.* El archivo tlLogic está conformado por dos etiquetas, en la primera etiqueta llamada tlLogic se establece los valores de identificación, el tipo de semáforo que controla la intersección, es decir, si el semáforo es estático o actuado en función de la carga de tráfico vehicular. La etiqueta secundaria llamada phase establece la duración, los estados de los ciclos y fases del semáforo.

A continuación, se describen los atributos adicionales con los que cuenta el fichero tlLogic en el archivo de red.

### *4.8.2. Atributos TlLogic*

- id: es el identificativo con el que cuenta el semáforo, la identificación de un semáforo se encuentra establecido dentro de un archivo net.xml y es idéntico al id que identifica a la intersección.
- type: este parámetro establece el tipo de semáforo, si es de duraciones de fase fija o en brechas de tiempo activadas por el aumento o disminución del tráfico vehicular.
- programID: aquí se establece la identificación del programa de los semáforos.
- offset: es el desplazamiento de tiempo inicial del programa.

### *4.8.3. Atributos Phase*

- duration: este parámetro indica la duración de la fase.
- state: son los estados de la señal de fase.
- minDur: valor de duración mínima de la fase cuando el semáforo es de tipo accionado.
- maxDur: valor de duración máxima de la fase cuando el semáforo es de tipo accionado.
- name: es una descripción que es de tipo opcional para la fase.
- next: indica la siguiente fase del ciclo después de la procesada anteriormente.

## 4.9. Intersecciones en SUMO

La distribución de la ubicación de los semáforos que controlan las intersecciones se genera directamente de los datos de la red extraída, esto mediante programas que se encuentran almacenados como archivos XML dentro del archivo que contiene toda la información de la

red del simulador SUMO. El archivo que lleva por nombre junction, contiene toda la información de las intersecciones, la estructura de este archivo se muestra en la Figura 9.

**Figura 9.**

*Estructura del archivo junction*

```
<junction id="1283413291" type="dead_end" x="520.70" y="550.23" incLanes="" intLanes="" shape="519.12,549.96 522.28,550.49"/>
<junction id="1440011859" type="priority" x="1669.06" y="1852.77" incLanes="37420707#0_0 -148619297#0_0 -37420707#1_0" intLanes=":1440011859_0_0
:1440011859_9_0 :1440011859_10_0 :1440011859_3_0 :1440011859_4_0 :1440011859_5_0 :1440011859_6_0 :1440011859_7_0 :1440011859_11_0" shape="
1678.70,1853.06 1676.76,1846.97 1674.70,1847.20 1674.04,1846.89 1673.62,1846.29 1673.44,1845.40 1673.51,1844.22 1667.17,1843.32 1666.77,1845.37
1666.21,1847.08 1665.48,1848.45 1664.58,1849.49 1663.51,1850.19 1662.27,1850.55 1663.22,1856.88 1666.57,1856.36 1669.05,1855.91 1671.07,1855.44
1673.06,1854.87 1675.46,1854.11">
    <request index="0" response="000000000" foes="100010000" cont="0"/>
    <request index="1" response="011000000" foes="011110000" cont="1"/>
    <request index="2" response="010001000" foes="010001000" cont="1"/>
    <request index="3" response="010000000" foes="010000100" cont="0"/>
    <request index="4" response="010000011" foes="110000011" cont="0"/>
    <request index="5" response="001000010" foes="001000010" cont="0"/>
    <request index="6" response="000000000" foes="000100010" cont="0"/>
    <request index="7" response="000000000" foes="000011110" cont="0"/>
    <request index="8" response="000010001" foes="000010001" cont="1"/>
</junction>
```

*Nota.* La estructura del programa que se encarga de la ubicación de los semáforos dentro de la red, está conformada por un elemento principal (junction), que es el área donde se cruzan diferentes carriles, y un conjunto de elementos secundarios (request), que describen, para cada uno de los enlaces, cuál de los flujos consta de una prioridad más alta, obligando al vehículo en el índice indicado a detenerse, lo que permite conocer que flujos están en conflicto.

El archivo junction consta de varios atributos adicionales, los cuales se describen a continuación.

### *4.9.1. Atributos de Junction*

- id: es el identificador de la unión a la que está asociado el programa.
- type: es el tipo de intersección que se genera en la red vial, donde dependiendo del tipo de la intersección, puede ser: traffic_light, priority, dead_end o internal.
- x: es la posición o coordenada x real de la intersección.
- y: es la posición o coordenada y real de la intersección.
- z: es la posición o coordenada z real de la intersección, esta puede ser opcional.
- incLanes: esto nos muestra los identificadores de los carriles que terminan en la intersección, se encuentran ordenados por dirección en el sentido de las agujas del reloj.
- intLanes: indica los ID de los carriles dentro de la intersección.
- shape: es un polígono el cual describe los limites viales de la intersección.

### *4.9.2. Atributos de Request*

- index: es el índice de la conexión descrita en la matriz de derecho de paso.
- response: es una cadena de bits que describe para cada una de las conexiones, la prohibición del paso sin desaceleración en la intersección para los vehículos en aquellas conexiones. El bit más a la derecha corresponde al índice 0.

- foes: es una cadena de bits que describe para cada conexión el conflicto que se puede producir con esta conexión.
- cont: describe para cada vehículo la autorización de que puede pasar la primera línea de parada, y esperar dentro de la conexión hasta que se carezca de vehículos con mayor prioridad.

## 4.10. Conexiones en SUMO

El archivo de conexiones de una red vial utilizada para realizar la simulación en SUMO, describe cómo y cuáles son los bordes entrantes y salientes de un nodo o cruce conectado. Estas conexiones también pueden ser especificadas por el usuario, esto siempre que se detalle cual o cuales son los carriles entrantes que se conectaran a los carriles salientes en una intersección. En la Figura 10 se puede observar la estructura del archivo para las conexiones que llevan el nombre de connection:

**Figura 10.**

*Estructura del archivo connection XML*

```
<connection from="53473392#4" to="32698246#6" fromLane="0" toLane="0" via=":3365902859_0_0" tl="GS_3365902859" linkIndex="0" dir="r" state="o"/>
<connection from="53473392#4" to="53473392#6" fromLane="0" toLane="0" via=":3365902859_1_0" tl="GS_3365902859" linkIndex="1" dir="s" state="o"/>
<connection from="53473392#6" to="776033273#1" fromLane="0" toLane="0" via=":367910406_0_0" tl="GS_367910406" linkIndex="0" dir="s" state="o"/>
<connection from="53473392#6" to="32695273#3" fromLane="0" toLane="0" via=":367910406_1_0" tl="GS_367910406" linkIndex="1" dir="l" state="o"/>
<connection from="641374799" to="1124893648" fromLane="0" toLane="0" via=":367909898_0_0" dir="r" state="M"/>
<connection from="641374799" to="705010412" fromLane="0" toLane="0" via=":367909898_1_0" dir="s" state="M"/>
<connection from="641374800" to="1124893004" fromLane="0" toLane="0" via=":10252456242_0_0" dir="s" state="m"/>
<connection from="641374800" to="69231954#6" fromLane="0" toLane="0" via=":10252456242_1_0" dir="l" state="m"/>
```

### *4.10.1. Atributos de Connection*

Los atributos que se establecen para la etiqueta connection, hacen referencia a las conexiones existentes entre los diferentes componentes de la red vial, como los nodos, enlaces y giros. Los atributos más comunes se detallan a continuación:

- from: este atributo nos indica el identificador del enlace de origen de la conexión.
- to: este atributo es el identificador del enlace donde termina la conexión.
- fromLane: indica el carril del borde entrante en el que inicia la conexión.
- toLane: es el carril del enlace saliente en donde termina la conexión.
- via: este atributo es el identificador de un enlace intermedio opcional que se utiliza para conectar el enlace de origen y el enlace de destino
- tl: este atributo es el identificador del semáforo que controla la conexión, este atributo no será asignado en el caso que la intersección carezca de un semáforo.
- linkIndex: este atribute establece el índice de la señal responsable de la conexión dentro del semáforo, el atributo no se asigna en el caso de que una conexión carezca de un semáforo.

- dir: es el atributo que establece la dirección de la conexión, que puede ser "s" (straight) para una conexión recta o "l" (left) o "r" (right) para una conexión de giro, "L" = parcialmente izquierda, R = parcialmente derecha, "inválido" = sin dirección.
- state: indica el estado de la conexión, que puede ser "-" = callejón sin salida, "=" = igual, "m" = enlace menor, "M" = enlace principal, semáforo solamente: "O" = controlador apagado, "o" = parpadeo amarillo, "y" = enlace menor amarillo, "Y" = enlace principal amarillo, "r" = rojo, "g" = menor verde, "G" mayor verde.

# 5. Metodología

En este apartado iniciaremos describiendo la manera en que se trabajará para solucionar el problema planteado al inicio de este proyecto de investigación, mismo que trata acerca de la optimización de la ubicación de los semáforos en las intersecciones de la red vial de la ciudad de Loja, para ello es importante iniciar describiendo como están codificados los semáforos dentro del archivo de red que nos proporciona el simulador de microtráfico SUMO y como se realiza la distribución de los semáforos en las distintas intersecciones, seguidamente trataremos la manera en que se han obtenido los datos de entrada para alimentar nuestro algoritmo genético, los mismos que comprenden desde los mapas de red de carreteras de la ciudad, las rutas que deberán tomar los vehículos y los archivos de configuración inicial necesarios para realizar las simulaciones.

Finalizando este apartado se explicará cómo se ha trabajado la propuesta de solución mediante la ejecución del algoritmo genético, la representación de la población que se ha proporcionado para su análisis y los operadores genéticos que han sido implementados con la finalidad de que brinden un óptimo desempeño en la obtención de la mejor solución.

## 5.1. Área de Estudio

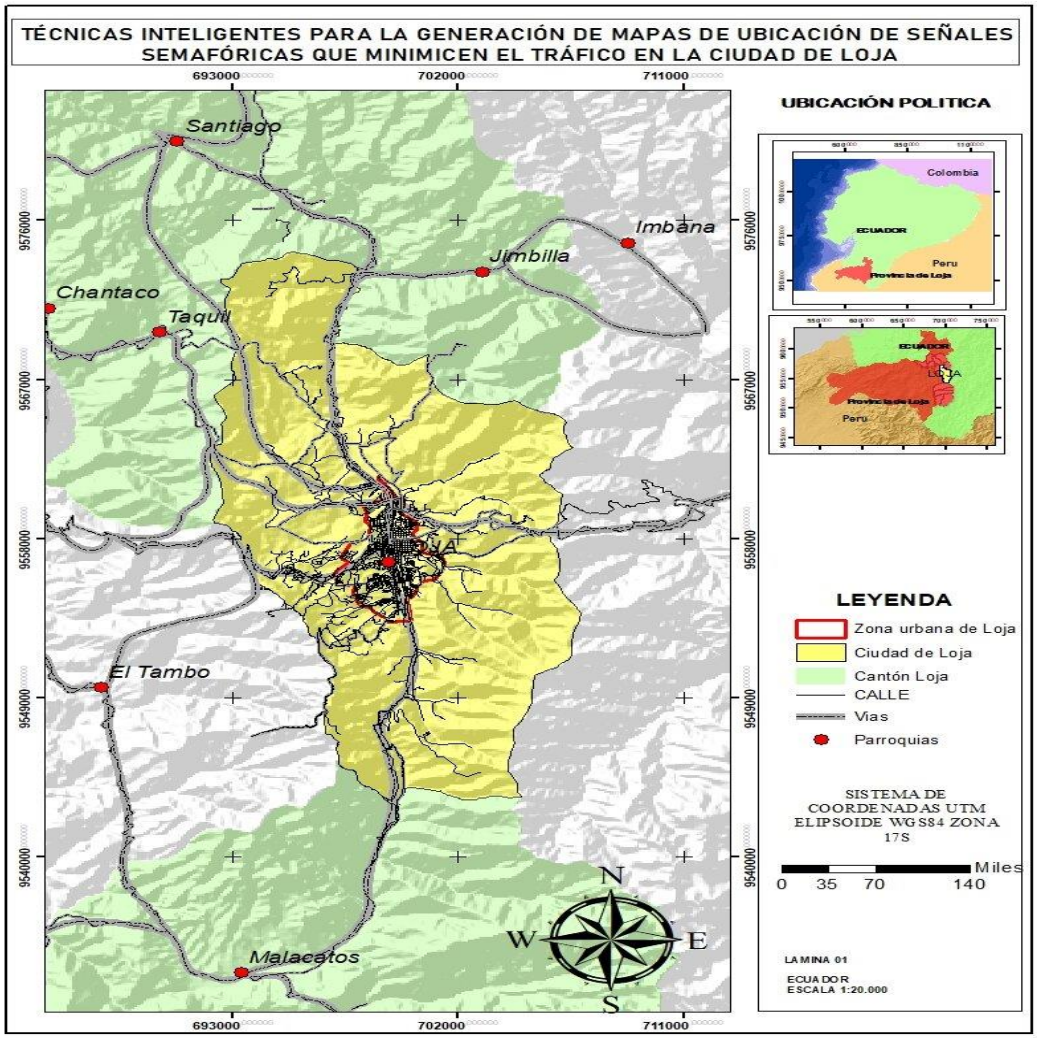
El escenario escogido para el desarrollo de nuestro trabajo de investigación, corresponde a la zona urbana de la ciudad de Loja – Ecuador, tomando en cuenta las áreas más conflictivas, esto con la finalidad de contribuir a minimizar el tráfico vehicular en las intersecciones y reducir los índices de contaminación generados por el parque automotor que circula por las vías de la urbe durante todos los días del año.

La ciudad de Loja está ubicada en la parte sur de Ecuador, a una altitud de aproximadamente 2100 metros sobre el nivel del mar. En términos geográficos, se encuentra entre las latitudes sur de 03º 39' 55'' y 04º 30' 38'' y las longitudes oeste de 79º 05' 58'' y 79º 05' 58''. Una característica que la distingue de otras ciudades del país es su relieve variado, con altitudes que van desde los 700 metros hasta más de 3700 metros sobre el nivel del mar. Esto le da a Loja una gran diversidad de ecosistemas y lugares turísticos interesantes, como lo señala Instituto Geográfico Militar-Ecuador (2022).

A continuación, en la Figura 11 se muestra la ubicación geográfica de la ciudad de Loja, misma que perteneciente a la provincia de Loja de la región sur del país:

**Figura 11.**

*Área de estudio, ubicación geográfica de la ciudad de Loja-Ecuador*




*Nota*. La Figura 11 permite visualizar la ubicación geográfica de la ciudad de Loja-Ecuador e identificar la zona urbana, Elaborado por el autor mediante el Software ArcGIS.

## 5.2. Implementación de la Solución

A continuación, se explica los pasos a seguir para obtener la solución al problema, iniciando con la descripción acerca de las características que presenta la Topología de Red de SUMO, la cual está conformada por intersecciones, lógicas de semáforos, bordes, distritos, descriptores de rotondas y la manera en que los semáforos son distribuidos en el mapa de red que hemos previamente obtenido con la herramienta OpenStreetMap. Luego hablaremos de la técnica con la cual han sido obtenidos los datos de entrada para alimentar nuestro algoritmo genético. Posteriormente detallaremos los archivos con extensión XML que hemos generado, tanto de mapas de carreteras, rutas que deberán tomar los vehículos, y de configuración inicial, indispensables para poder realizar las simulaciones en SUMO.

Finalizando este apartado, abordaremos las técnicas que hemos utilizado para representar la población, la función de evaluación considerada para el análisis de las soluciones y los operadores genéticos que se han utilizado en el algoritmo genético.

### *5.2.1. Obtención de los Datos de Entrada*

El simulador de microtráfico SUMO puede realizar la simulación de diferentes escenarios de tráfico vehicular de distintas ciudades del mundo, para ello debe contener un conjunto de archivos para su ejecución, el primer archivo que se le asigne deberá contener el mapa de la red por donde van a desplazarse los vehículos, un segundo archivo acerca de las rutas por donde deberán circular los vehículos dentro de la simulación y un tercer archivo de configuración en donde se indique las instrucciones para realizar la simulación. Al simulador SUMO se le podrá cargar archivos adicionales a los tres antes mencionados, pero esto ya dependerá de los escenarios que se quieran analizar y de los propósitos para lo cual se haga uso de esta herramienta.

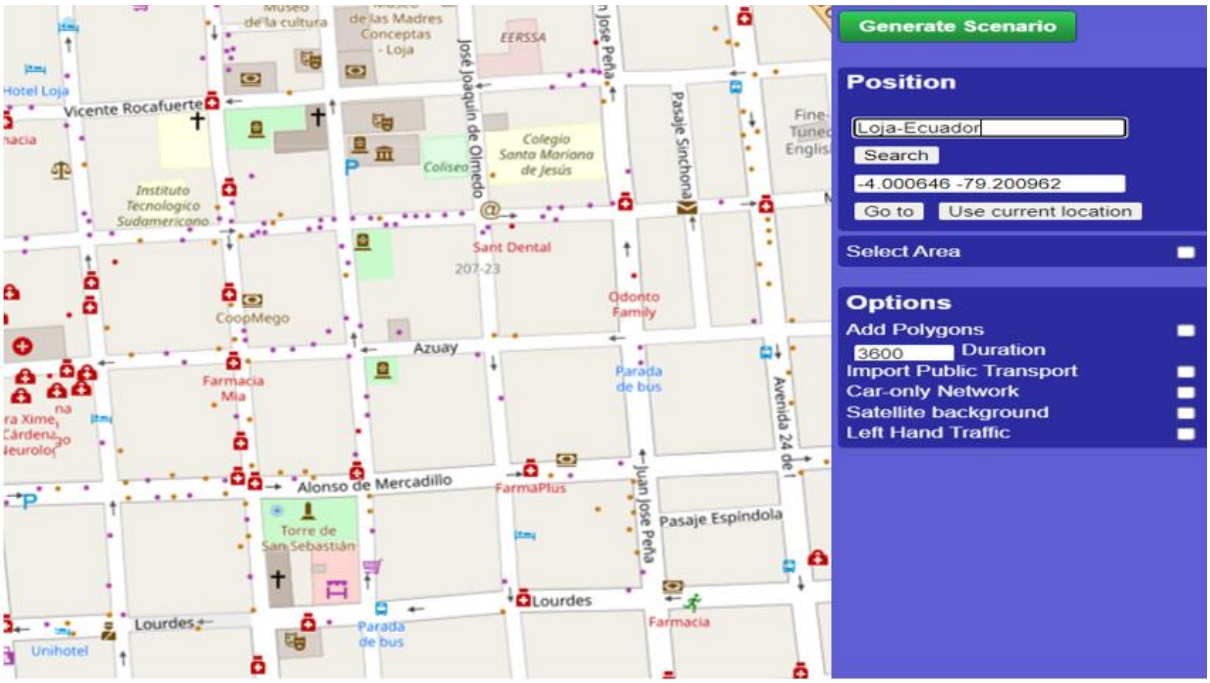
### *5.2.2. Extracción de Red Vehicular Mediante OpenStreetMap*

La extracción del archivo de red vehicular que se utiliza para realizar la simulación dentro del entorno de SUMO, se obtiene mediante la utilización de OpenStreetMap que es una plataforma que cuenta con datos a nivel global acerca de las redes viales, haciendo uso de esta herramienta se puede extraer una red de carreteras para realizar la experimentación y las simulaciones respectivas para poder abarcar cada una de las etapas de la investigación.

A continuación, en la Figura 12 se muestra la zona delimitada que se puede extraer haciendo uso de OpenStreetMap.

**Figura 12.**

*Zona seleccionada y delimitada de la ciudad de Loja*



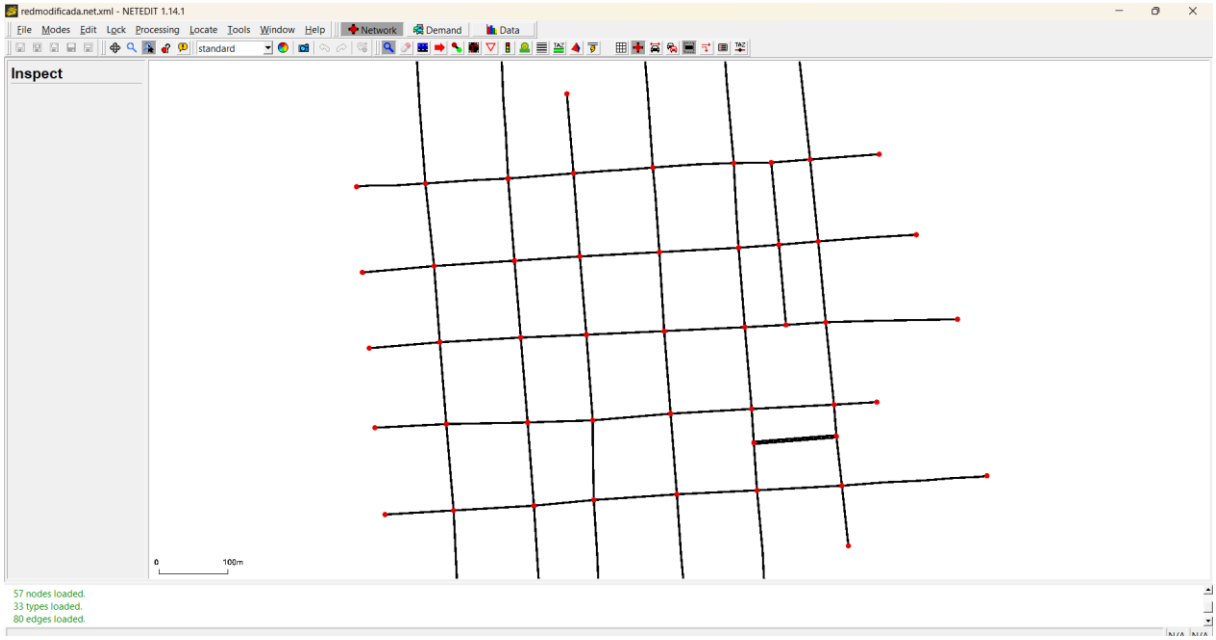

*Nota*. La herramienta OpenStreetMap nos permite seleccionar el tamaño y los componentes que se desea formen parte del mapa de red vial que se genera para realizar las simulaciones.

El asistente de SUMO (OSMWebWizard) permite acceder y exportar determinadas áreas de una manera sencilla desde la página web de OpenStreetMap, para ello simplemente generamos un escenario desde la interfaz gráfica de OpenStreetMap de la ciudad que deseamos analizar, el mapa que se genera ya contiene el formato requerido por el simulador, por ello no es necesario realizar conversiones de formato.

Una de las funciones con las que cuenta SUMO, es NETEDIT que nos permite editar el mapa de red vial, y corregir algunos de los errores que se pueden presentar al momento de generar y exportar el mapa de red vial. Además, esta función nos permite agregar, eliminar aristas o elementos innecesarios de acuerdo a los parámetros requeridos para la simulación. En la Figura 13 se muestra el aspecto final del mapa de red vial.

**Figura 13.**

*Red vial de la ciudad de Loja, mapa editado mediante la función NETEDIT del simulador*




*Nota.* La herramienta NETEDIT nos permite modificar manualmente varias estructuras del archivo de red mediante la interfaz gráfica de SUMO, ya que, en muchos casos, las redes que se extraen mediante OpenStreetMap pueden diferir de las redes reales.

### *5.2.3. Generación de Rutas para los Vehículos*

Luego de la selección y creación de la red vial que vamos a simular, se debe proceder a generar un archivo de rutas para la red vial. Existen varias formas en las que se puede crear estos archivos, una de ellas, es la elección del archivo de rutas que genera automáticamente el asistente de SUMO, pero en este caso por motivos de análisis y pruebas, hacemos uso de la función RandomTrips.py para generar dicho archivo de rutas. Esta función nos sirve para generar y almacenar de forma rápida las rutas en un archivo XML, las rutas obtenidas son las que deberán de seguir cada uno de los vehículos dentro de la simulación.

Adicionalmente, a los aspectos antes mencionados, podemos ajustar la frecuencia con la cual aparecen los vehículos, brindándonos la posibilidad de generar escenarios con una alta o baja tasa de tráfico. Es importante recalcar que RandomTrips.py genera el conjunto de rutas de manera aleatoria para una red vial determinada. La estructura del archivo de rutas obtenido lo podemos observar en la Figura 14.

**Figura 14.**

*Estructura del archivo de rutas*

```
<routes
    <vehicle id="0" depart="0.00">
        <route edges="52229689#2 52229689#3 52229689#4 32698374#4 32695037#1 32695037#2"/>
    </vehicle>
    <vehicle id="1" depart="1.00">
        <route edges="32695273#5 641374799 705010411#0 705010411#1 705010411#2 705010411#3 705010411#4 32698234#0"/>
    </vehicle>
```

*Nota.* El archivo de rutas con extensión rou.xml que se presenta en la Figura 14, describe el conjunto de rutas para los vehículos en la red vial, también contiene el identificador para cada uno de los vehículos, la hora de salida y los bordes que atraviesa a lo largo de la ruta hacia su destino en la red vial.

SUMO permite simular emisiones contaminantes de vehículos en base a datos del Manual de Factores de Emisión HBEFA3, la clase HandBook Emission FActors (HBEFA), y realizar la simulación principalmente de los siguientes contaminantes: Dióxido de Carbono (CO2), Monóxido de Carbono (CO), Hidrocarburos No Quemados (HC), Óxidos de Nitrógeno (NOx), Material Particulado (POx) y consumo de combustible.

Los vehículos que cursan las rutas generadas por RandomTrips.py para la red vial configurada, utilizan la definición eclass= "HBEFA3/PC_G_EU4", la cual es una clase de emisión de contaminantes que se utiliza para el tipo de vehículos de pasajeros con motor de gasolina basados en la norma euro 4, dicha normativa establece los límites de emisiones contaminantes que los automóviles deben de cumplir una vez que salgan a circular en la Unión Europea, como lo señala Box Repsol (2021).

### *5.2.4. Archivo de Configuración*

El archivo de configuración con extensión osm.sumocfg deberá contener las instancias necesarias para permitir ejecutar la simulación tanto de manera gráfica o mediante consola, para ello, el archivo de configuración deberá apuntar a los archivos de red y rutas en la carpeta que contiene estos registros. La estructura del archivo de configuración la podemos apreciar en la Figura 15.

**Figura 15.**

*Estructura de archivo de configuración SUMO para realizar la simulación*

```
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instar

    <input>
        <net-file value="demo003osm.net.xml"/>
        <route-files value="alexander.routes.rou.xml"/>
    </input>

    <output>
        <tripinfo-output value = "demo003output-tripinfos.xml"/>
        <emission-output value = "emisiones.xml"/>
    </output>


    <processing>
        <ignore-route-errors value="true"/>
    </processing>

    <time>
        <begin value ="0"/>
        <end value="500"/>
        <route-steps value="-1"/>
    </time>

    <routing>
        <device.rerouting.adaptation-steps value="18"/>
        <device.rerouting.adaptation-interval value="10"/>
    </routing>

    <report>
        <verbose value="true"/>
        <duration-log.statistics value="true"/>
        <no-step-log value="true"/>
    </report>

    <gui_only>
        <gui-settings-file value="osm.view.xml"/>
    </gui_only>
```

En el archivo de configuración como parámetros de entrada se debe agregar en la etiqueta net-file el nombre del archivo de mapa de red vial que vamos a simular y en la etiqueta route-files se asigna el archivo de rutas generados para el mapa de red vial. El parámetro ignore-route-errors con valor igual a true permite ignorar las rutas que por motivos de la generación de mapas se pueden crear con algunos errores o tramos de vías incompletas, que de ser necesario deberán ser corregidas para poder realizar la simulación.

Finalmente, el indicador time-to-teleport con un valor de -1 se configura con la finalidad de evitar que los automóviles se teletransporten dado que han permanecido mucho tiempo parados frente a una intersección. Los demás atributos que contine el archivo de configuración son opcionales para ejecutar la simulación y se configuran por defecto una vez que ha sido descargado el archivo de mapa de red vial.

## 5.3. Desarrollo del Algoritmo Genético

Luego de haber obtenido todos los archivos necesarios para realizar las simulaciones, tenemos que empezar a desarrollar el Algoritmo Genético (AG) que se encargará de optimizar la ubicación de las señales semafóricas en el mapa de red vial, y junto con la herramienta SUMO realizar la simulación de las mejores soluciones, donde cada una de las simulaciones que se

ejecuten le permitirán al AG evaluar que tan buena es una solución en comparación a otra de la población.

El algoritmo genético ha sido modelado mediante el entorno Anaconda Navigator que nos proporciona diferentes aplicaciones de programación, y para este trabajo se ha hecho uso de la aplicación Jupyter Notebook que funciona bajo el Software Python 3. Para complementar al software se hace uso de las librerías de TraCI (Interfaz de control de Tráfico) de SUMO, ya que permiten el acceso a una simulación de tráfico rodado en curso, permitiendo recuperar valores de objetos simulados y manipular su comportamiento mediante consola.

El algoritmo genético que estaremos ejecutando seguirá la estructura que se muestra en la Figura 6, en el cual se establecen las operaciones y la secuencia que deberá de seguir el algoritmo para su correcto funcionamiento.

### *5.3.1. Codificación de la solución inicial*

En nuestro trabajo de investigación se realiza la codificación de la solución inicial a partir del archivo de mapa de red vial con extensión osm.net.xml esto para extraer los datos acerca de la ubicación de las señales semafóricas en las intersecciones, la información obtenida permitirá realizar la codificación para cada gen de la solución.

En el simulador de microtráfico SUMO las etiquetas de tipo type = traffic_light indican que la intersección está controlada por un semáforo, en cambio las etiquetas del tipo type = priority señalan que la intersección no está controlada por un semáforo, las etiquetas de tipo type = dead_end indican puntos terminales de la red vial y las etiquetas de tipo type = internal indican que en dicha intersección se utiliza una lógica interna para evitar que los vehículos invadan la intersección en una situación de atasco.

La solución codificada en base al archivo de mapa de red vial obtiene una codificación de números enteros, en principio se analiza y se manipula la etiqueta *junction* contenida en el archivo de mapa de red vial , aquellas etiquetas de tipo type = traffic_light serán representadas mediante un gen que tomara un valor de 1, las etiquetas de tipo type = priority serán representadas mediante un gen que tomara un valor de 0, las etiquetas de tipo type = dead_end serán representadas mediante un gen que tomara el valor de 2 y finalmente las etiquetas de tipo type = internal serán representadas mediante un gen que tomara un valor de 3, de esta manera todos los genes se agruparán en una cadena de información para codificar la solución inicial.

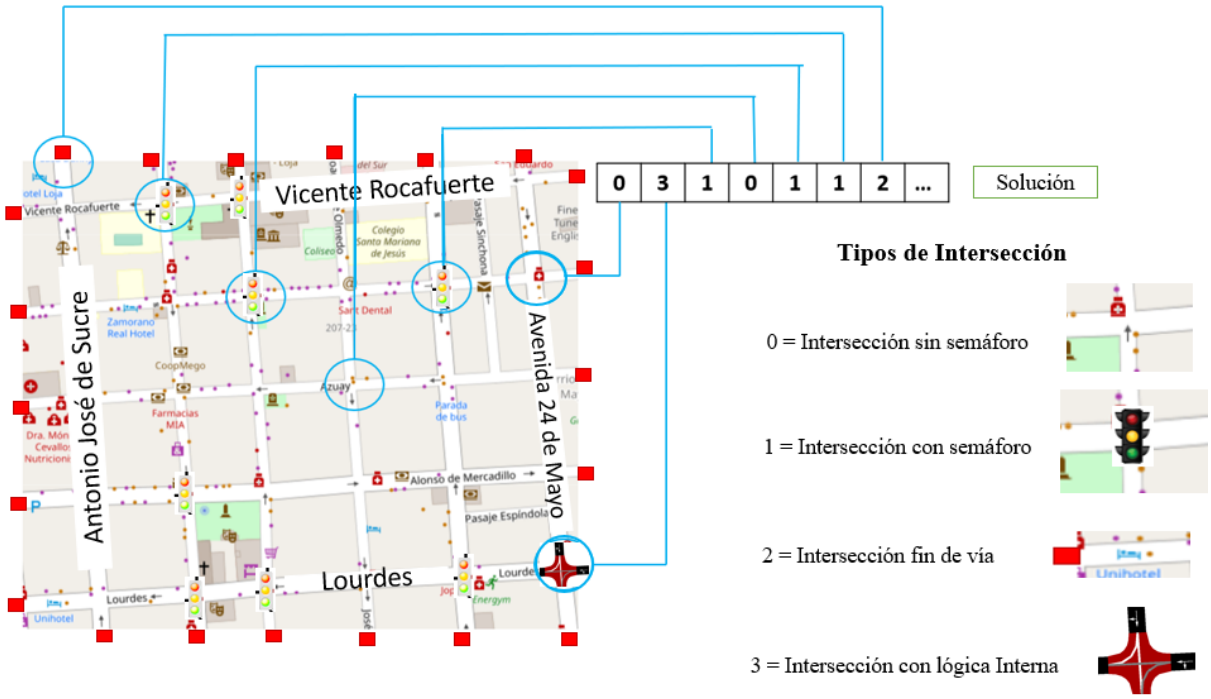
Las etiquetas de tipo type = dead_end y tipo type =internal al ser tomadas en cuenta para codificar a la solución inicial, permiten trabajar y simular redes viales mucho más cercanas

a la realidad, ya que permiten agregar tramos de fin de vía y enlaces internos en las intersecciones de las redes viales de las ciudades.

En la Figura 16 se muestra cómo se encuentran codificadas y decodificadas cada una de las intersecciones, en este caso práctico se trabaja con el parámetro type para conocer el tipo de la intersección y poder asignar los valores de cero, uno, dos y tres respectivamente a cada uno de los genes de la solución.

**Figura 16.**

*Codificación de la solución en base a lo datos de la red vial*



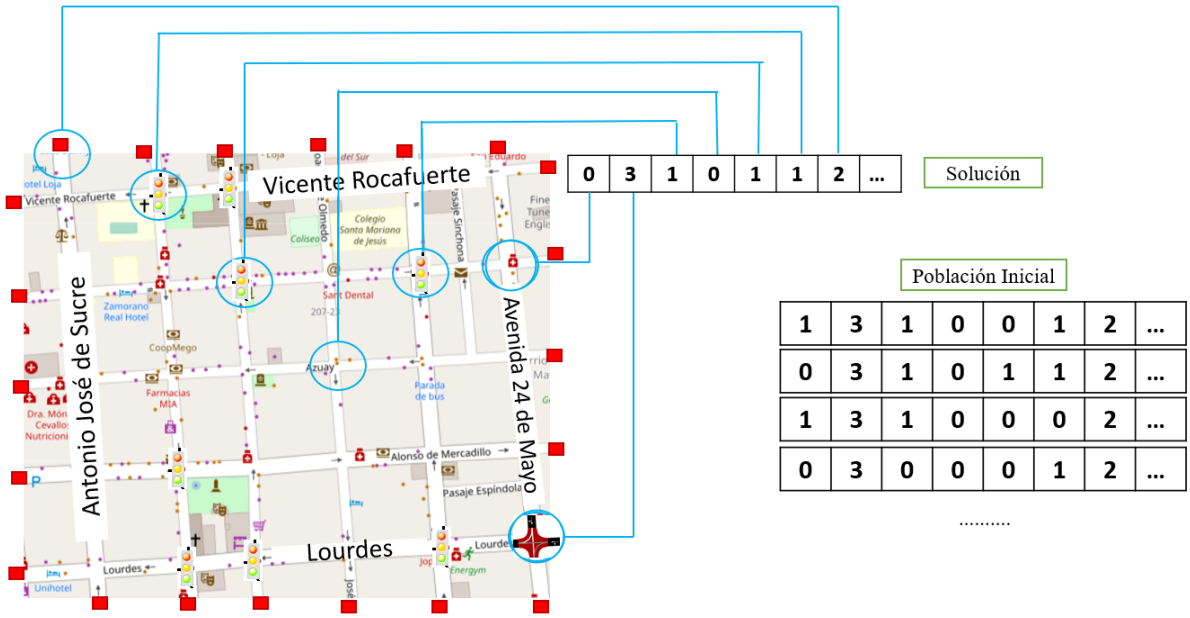

*Nota.* La Figura 16 nos permite apreciar cómo están codificados cada uno de los genes para formar la solución inicial, esto dependiendo del tipo de intersección que se encuentre establecida en el archivo de mapa de red, imagen elaborada por el autor.

### *5.3.2. Población Inicial*

A partir de la solución inicial que se obtiene del proceso anterior, se genera una población de soluciones de manera aleatoria para alimentar a nuestro Algoritmo Genético, con esto se logra que las nuevas soluciones tengan las mismas dimensiones que la solución que se obtiene inicialmente del archivo de mapa de red, dicho proceso se puede observar en la Figura 17.

**Figura 17.**

*Generación de Población inicial aleatoria*




### *5.3.3. Función de Evaluación Utilizada*

A continuación, se describe la función de evaluación utilizada para conocer el nivel de coste que tiene cada solución al problema, para ello las soluciones se evalúan utilizando SUMO, y luego de cada simulación se extrae un conjunto de parámetros para determinar el coste de las soluciones.

Los parámetros que se extraen en cada una de las simulaciones son los siguientes:

- Emisiones de Dióxido de Carbono (CO2): cantidad total de CO2 generado por todos los vehículos en la simulación.
- Duración (D): sumatorio total del tiempo que necesita cada vehículo para completar el viaje en la red.
- Consumo de combustible (FC): cantidad total de combustible que consumen todos los vehículos en la red.
- Tiempo perdido (TL): sumatoria del tiempo perdido de todos los vehículos de la red.
- Vehículos en destino (Vd): números total de vehículos que llegan a su destino.

La siguiente función matemática será utilizada para evaluar la población de soluciones, esto para obtener los valores de coste de cada solución, dicha función ha sido modelada en base a la representación utilizada en Olivera et al., (2015), donde las variables consideradas de tráfico

para conformar la función de evaluación, tienen como finalidad, minimizar el tiempo de espera (Wt), el consumo de combustible (FC) y reducir las emisiones de gases contaminantes.

$$Coste = \frac{(\alpha FC + \beta D + \gamma TL)}{(\delta Vd)}$$

Los parámetros utilizados en la función, necesitan ser normalizados, esto se realiza con la finalidad de igualar la escala de los diferentes parámetros y que todos tengan el mismo impacto en la función. Esto se realiza ya que algunos valores pueden ser muy grandes en comparación con otros, lo que puede llegar a un sesgo en la selección de las soluciones. Con este proceso logramos que todas las variables tengan una igualdad matemática en la función, asegurando así, una búsqueda más eficiente y precisa de la mejor solución.

Las variables adicionales $\alpha, \beta, \gamma,$ y $\delta$ que están presentes en la función, permiten asignar pesos a cada variable y se establecen en función de los valores máximos que pueden obtener cada uno de los parámetros dentro de la simulación, con esto se puede establecer la importancia de cada uno de estos en la función.

Los parámetros que se mencionan anteriormente, están contenidos en diferentes archivos, para obtener cada uno de ellos, se puede utilizar la consola o también mediante instrucciones de código, los archivos que contienen los datos que alimentan la función, son los siguientes:

**Tripinfo-output.** Con el parámetro tripinfo-output obtenemos un archivo output-tripinfos.xml que contiene información sobre la hora de salida de cada vehículo, la hora a la que quería partir el vehículo y la hora de llegada del vehículo.

**Fcd-output fcd_out.xml.** Mediante este parámetro se puede generar un archivo que contiene la ubicación y velocidad para cada uno de los vehículos en la simulación.

**Amitran-output armitran_out.xml.** Haciendo uso de este parámetro podemos obtener un archivo que contenga información sobre el tipo, la velocidad actual y la aceleración que realiza cada vehículo durante el tiempo que dura la simulación.

**queue-output queue_out.xml.** El parámetro queue-output permite obtener un archivo con información de la cola de vehículos que se puede presentar frente a una intersección que puede tener un controlador semafórico o carecer de uno.

**Device.emissions.probability 1.** Este parámetro se puede configurar para aumentar o reducir el conjunto de vehículos que pueden generar emisiones en la simulación.

**emission-output value = "emisiones.xml".** Este parámetro se lo debe establecer en el archivo de configuración de SUMO, para construir un archivo de salida sobre las emisiones generadas en la simulación.

## 5.4. Operadores Genéticos Aplicados

Los operadores genéticos son técnicas utilizadas en los algoritmos genéticos para simular el proceso de evolución que se da en la naturaleza, estos operadores permiten la creación de nuevas soluciones con mejor valor de coste a partir de los existentes y fomentar la diversidad en la población. En este trabajo incluimos los siguientes operadores genéticos para permitir que nuestra población de soluciones mejore con el tiempo.

### *5.4.1. Selección*

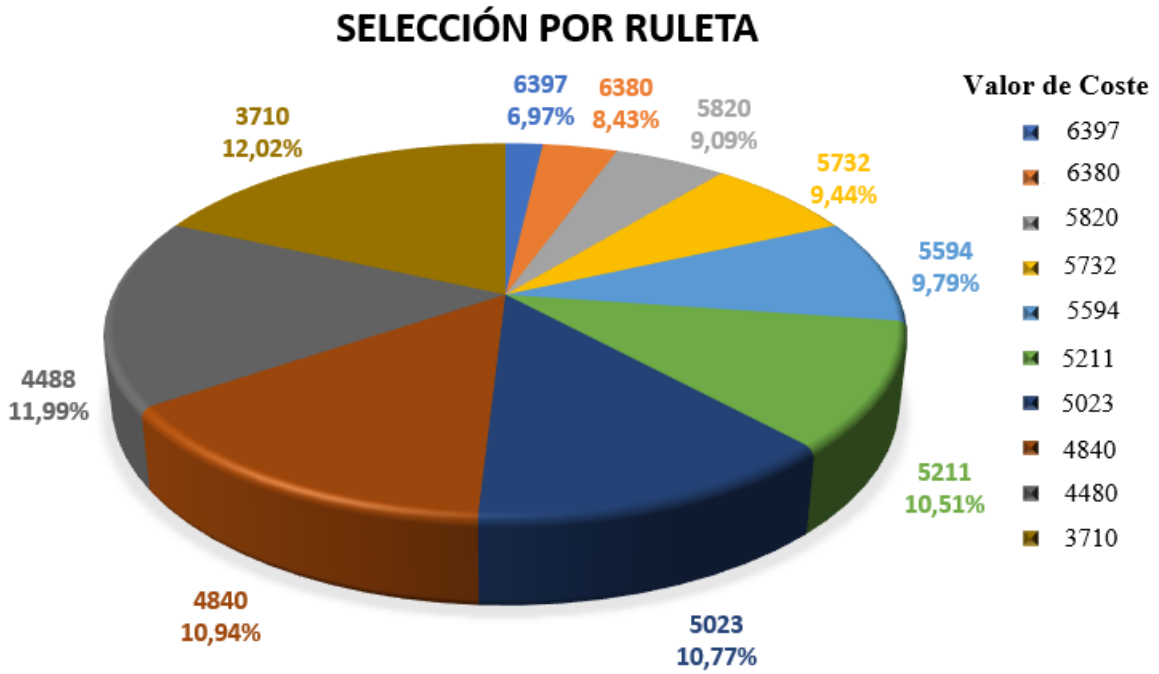
El proceso de selección aplicado para escoger las soluciones de la población, es la técnica de selección por regla de la ruleta, a través del siguiente proceso:

- Calcular el valor de coste: se calcula el valor de coste para cada solución en la población.
- Suma Total: se calcula la suma total del valor de coste de todas las soluciones de la población.
- Valores de costes relativos: Se calcula los valores de costes relativos de las soluciones dividiendo sus valores de coste entre la Suma Total.
- Generar número aleatorio: Generamos un número entre 0 y la Suma Total.
- Selección de soluciones: Seleccionar la solución cuyo segmento cubre el número aleatorio.
- Número de Soluciones: Repetir el proceso de selección hasta obtener el número deseado de soluciones.

En la Figura 18 se puede observar el método de selección por ruleta utilizado en el algoritmo genético.

**Figura 18.**

*Gráfica de operador de selección por rueda de ruleta*




*Nota*. Cada color representado en la rueda de ruleta significa la porción de ruleta que se asigna a cada solución dependiendo de su valor de coste, se puede observar que las soluciones con menor valor de coste tienen mayor probabilidad de ser seleccionadas por que estamos minimizando la función.

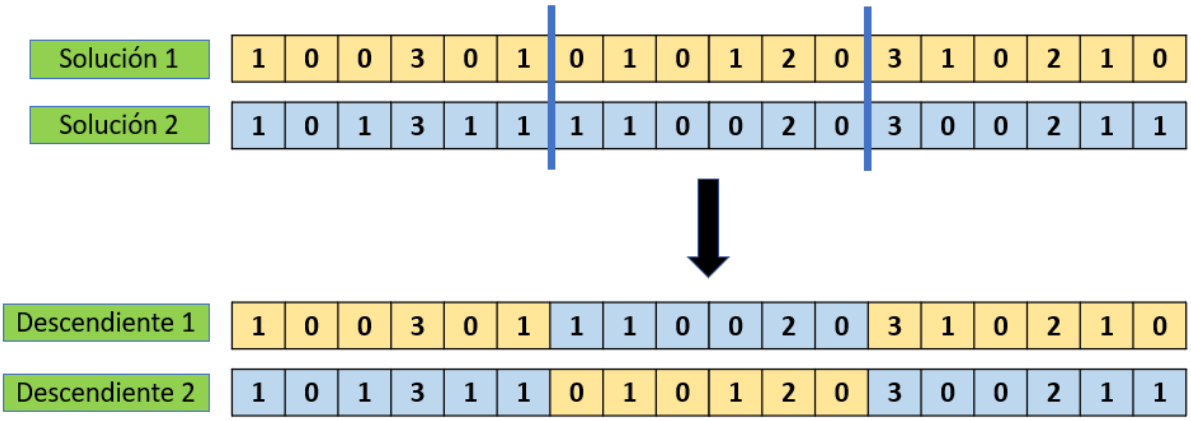
### *5.4.2. Cruce*

Una vez realizado el proceso de selección de las soluciones, se debe proceder a realizar la recombinación de los genes que conforman las soluciones, de esta manera se puede producir nuevas soluciones las cuales estarán presentes en la nueva población.

El tipo de operador de cruce que se ha tomado en cuenta para realizar la reproducción, es el cruce en dos puntos, esta técnica inicialmente selecciona dos soluciones, para luego proceder a cortar dichas soluciones en dos puntos seleccionados al azar y realizar el intercambio de las colas para obtener las nuevas soluciones. En la Figura 19 se muestra el método de cruce en dos puntos utilizado en el algoritmo genético.

**Figura 19.**

*Gráfica de operador de cruce en dos puntos*



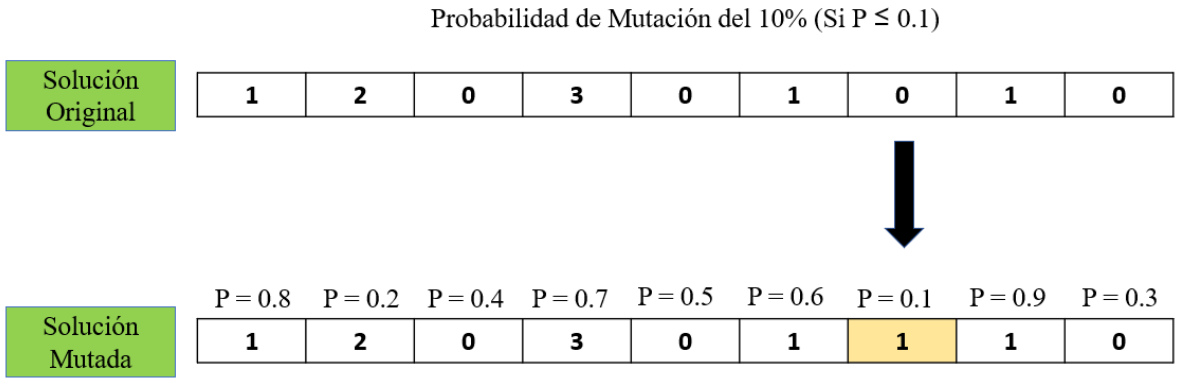

*Nota:* Se ha elegido esta técnica de cruce, dado que cada solución proporciona una distribución de los semáforos en las intersecciones de la red vial, mismos que se encuentran codificados en una lista de números enteros de (0, 1, 2, 3), logrando de esta manera, generar soluciones que tengan una mayor diversidad en la población.

### *5.4.3. Mutación*

El operador de mutación en el algoritmo genético permite introducir cambios aleatorios en las soluciones de la población y sirve para evitar la pérdida de diversidad en las soluciones. El proceso de mutación que se aplica es el siguiente, inicialmente se selecciona de manera aleatoria los n posibles genes a modificar de la solución. Aquellos genes seleccionados contaran con una probabilidad mínima de mutar, el gen que mutara es elegido aleatoriamente y el valor aleatorio se le es asignado, tanto sumando y restando valores dentro de un rango determinado, y como nuestras soluciones están conformadas por valores de 0 y 1, manteniendo fijo los valores 2 y 3, al realizar el proceso de mutación los valores de 0 cambiaran a 1 y viceversa. En la Figura 20 se puede observar el proceso que realiza el operador de mutación utilizado en el algoritmo genético.

**Figura 20.**

*Gráfica de operador de mutación*




*Nota:* En la Figura 20 se muestra el proceso que realiza el operador de mutación, en donde la probabilidad de mutación que se establece está directamente relacionada con la naturaleza del problema a solucionar. La finalidad del operador de mutación es evitar la pérdida de diversidad en las soluciones de la población.

## 5.5. Algoritmo Genético

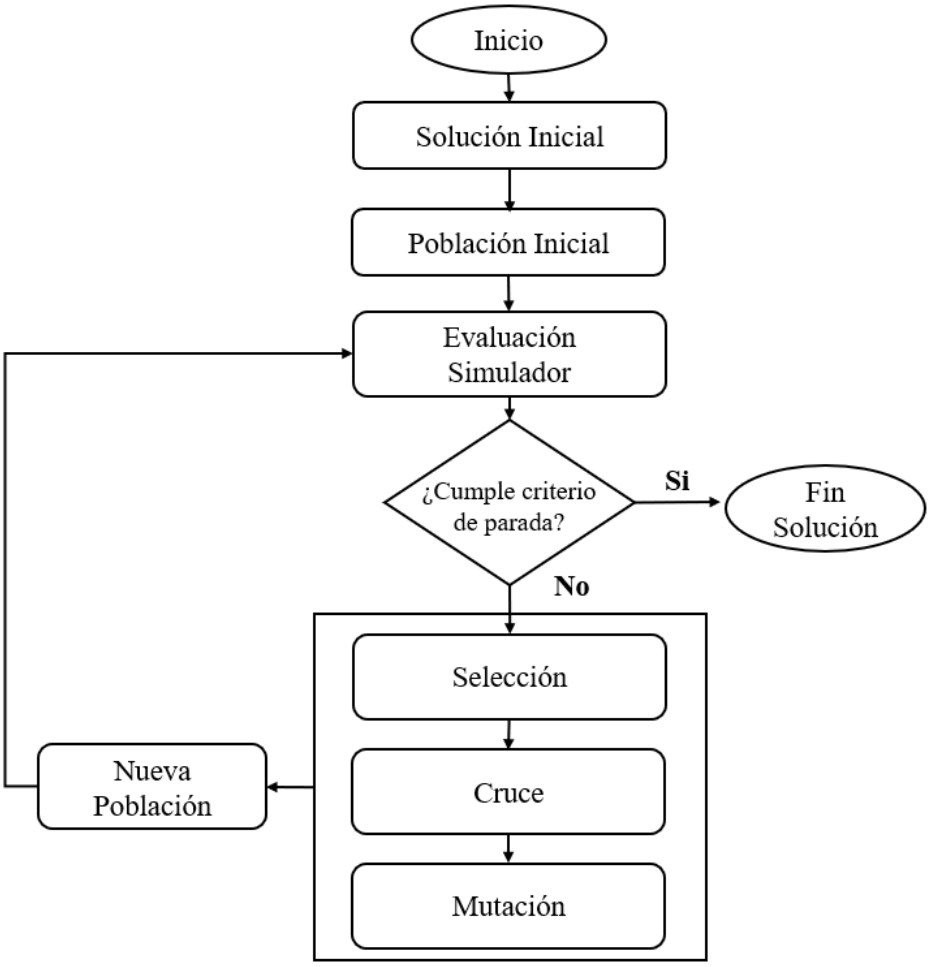
Luego de haber definido las características del mapa de red vial a simular, el archivo de rutas, así como el número de vehículos que circularán por la red vial y establecer los operadores genéticos, el algoritmo genético comienza su proceso con la inicialización de la población. Para ello, se genera una población inicial de posibles soluciones al problema. La primera solución que sirve como base para crear la población inicial se obtiene a partir de la solución con la que se extrae el mapa de red vial que se simula en SUMO.

Luego se evalúan todas las soluciones que abarca la población mediante el simulador SUMO y con la función diseñada para medir el coste de las soluciones. Posteriormente establecemos el criterio de parada mediante el número de iteraciones que nos permita llegar a la convergencia de las soluciones, de no cumplirse con el criterio de parada se aplica los operadores genéticos para obtener nuevas soluciones.

Finalmente, luego de cumplirse con el criterio de parada se presenta la mejor solución encontrada por el algoritmo genético, la cual contiene la nueva distribución de las señales semafóricas de la red vial para ser simuladas utilizando SUMO. En la Figura 21 se presenta la secuencia que sigue el algoritmo genético.

**Figura 21.**

*Diagrama de flujo del Algoritmo Genético*




*Nota.* El diagrama de flujo de un algoritmo genético, nos permite comprender la secuencia de las operaciones que se realizan una y otra vez en la búsqueda de soluciones al problema, mientras no se cumpla con el criterio de parada establecido.

El código fuente utilizado para el desarrollo del Algoritmo Genético del presente proyecto de investigación, se puede encontrar en el Anexo 1.

## 5.6. Diseño de Pruebas

La etapa de diseño de pruebas es importante en un algoritmo genético porque permite identificar problemas, ajustar parámetros y evaluar el rendimiento del algoritmo antes de su implementación. Lo que se busca con esta etapa es garantizar que el algoritmo produzca resultados precisos y confiables en un tiempo razonable de ejecución.

## 5.7. Zona Escogida

Para este trabajo de investigación la zona elegida corresponde a una parte céntrica de la ciudad de Loja, conformada por distintos tipos de intersecciones, que pueden ser de tipo traffic_logic, priority, internal y dead_end. El mapa vial extraído de OpenStreetMap cuenta solamente con 8 intersecciones que se encuentran controladas por un semáforo, lo que indica que la información contenida en la Web de OpenStreetMap no se encuentra actualizada y se debe utilizar la herramienta NETEDIT del simulador SUMO para modificar y añadir los semáforos faltantes en las intersecciones correspondientes, esto con la finalidad de configurar el mapa de red vial en todo lo posible al escenario actual. La zona escogida se encuentra delimitada de la siguiente manera, desde la parte Norte (N) con la calle Vicente Rocafuerte, en la parte Sur (S) la calle Lourdes, en la parte Este (E) con la Avenida 24 de mayo y finalmente al Oeste (W) con la calle Antonio José de Sucre. La longitud de la solución corresponde al número de intersecciones de la red vial, en este caso tenemos 68 intersecciones que forman el mapa de red vial ya que estamos tomando en cuenta todos los tipos de intersecciones que se han generado en el mapa de red vial. La solución que se menciona anteriormente sirve para la generación de la población inicial para el algoritmo genético. En la Figura 22 se puede observar el mapa de red vial que será utilizado para la presente etapa de pruebas.

**Figura 22.**

*Mapa vial de la zona céntrica de la ciudad de Loja extraída de OpenStreetMap*


*Nota*. En la Figura 22 se puede visualizar una parte de la zona céntrica de la ciudad de Loja que ha sido seleccionada para realizar el trabajo de investigación, esto debido a que se busca brindar una solución dirigida a disminuir la congestión vehicular en la ciudad, imagen obtenida de OpenStreetMap.

En lo que concierne a la generación de rutas para los vehículos utilizaremos la función ramdomTrips.py como se mencionó en el apartado de Generación de Rutas para los Vehículos y, el tiempo de simulación se ha configurado en 500 segundos (s), por qué será el tiempo máximo que puede tardar un vehículo desde su punto de partido hacia su punto de llegada. Este valor de tiempo ha sido determinado por que se utiliza el simulador SUMO para evaluar las soluciones y el cálculo del valor de coste debe ser lo suficientemente rápido ya que debe realizarse una y otra vez, como lo indica Rawat et al., (2022).

Los parámetros que se configuran para cada simulación se presentan en la Tabla 1.

**Tabla 1.**

*Parámetros de simulación de la red vial*

| Parámetros | Valor |
|---|---|
| Tiempo de Simulación | 500 s |
| Número de vehículos | 100 |
| Velocidad Máxima de Vehículo | 50 km/h |
| Número de Intersecciones | 68 |

*Nota*. El número de vehículos indica la cantidad de automóviles que circulan por la red vial durante el tiempo que dura la simulación. Además, se establece la velocidad máxima permitida en vías urbanas en 50 km/h.

## 5.8. Parámetros del Algoritmo Genético

Para determinar la configuración óptima de los parámetros del algoritmo genético, se debe realizar varias simulaciones con valores previamente definidos. El primer parámetro considerado en la evaluación del Algoritmo Genético es el tamaño de la población de soluciones, ya que este parámetro puede afectar su rendimiento y la calidad de la solución que se obtiene.

### *5.8.1. Tamaño de la Población*

Iniciamos variando el tamaño de la población inicial desde 10 hasta 200 soluciones, con los demás parámetros que se muestran en la Tabla 2.

**Tabla 2.**

*Parámetros de simulación para evaluar el tamaño de la población*

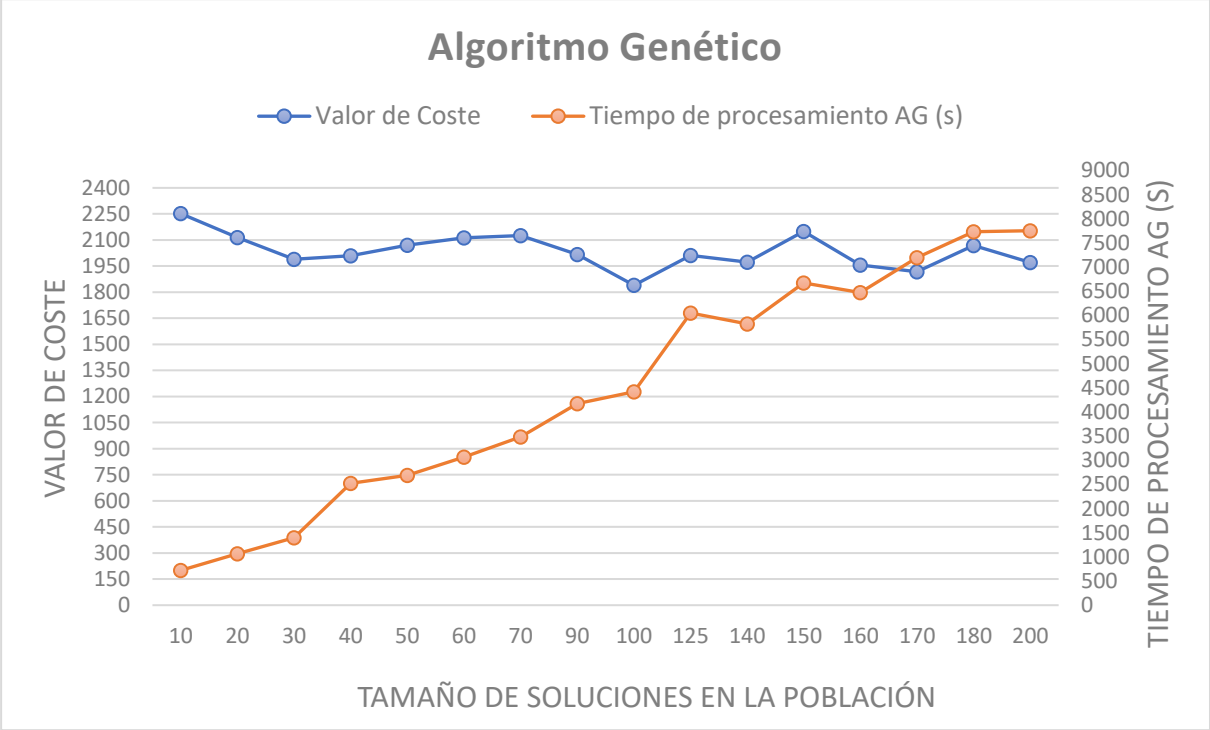
| Parámetros | Valor |
|---|---|
| Número de Iteraciones | 100 |
| Probabilidad de Mutación | 0,06 % |
| Tamaño de Población | 10 – 200 Soluciones |
| Operador de Selección | Ruleta |

*Nota*. La probabilidad de mutación se establece en un valor intermedio en el rango recomendado entre 0,005% y 0,10%, esto se realiza por el motivo que al configurar un valor muy bajo en el operador de mutación las nuevas soluciones serán las mismas que existían tras la reproducción, en cambio al configurar un valor alto en el operador de mutación las nuevas soluciones cambiaran la mayoría de sus genes provocando soluciones aleatorias en la población limitando la diversidad de la población de soluciones. Lo que se busca con el operador de mutación es realizar pequeños cambios en las soluciones para conservar la diversidad en la población.

Luego de ejecutar el algoritmo, en la Figura 23 se observa, que el incremento del tamaño de la población de soluciones tiene un impacto directo en el valor de coste que se obtiene de cada solución, y se puede evidenciar que, al aumentar el número de soluciones en la población, también aumenta el tiempo de procesamiento del algoritmo genético. Por esta razón, se elige un tamaño de población de 100 soluciones, que permite tener una diversidad aceptable en la población sin incrementar excesivamente el tiempo de procesamiento del algoritmo genético.

**Figura 23.**

*Algoritmo genético evaluando el tamaño de la población*




*Nota.* En la Figura 23 se muestra en color azul la variación del valor de coste de las soluciones cuando se modifica el tamaño de la población, además se puede evidenciar en color naranja como el aumento del tamaño de la población de soluciones también incrementa el tiempo de procesamiento del algoritmo genético.

### *5.8.2. Probabilidad de Mutación*

Con los resultados anteriores, se fija los parámetros según se muestran en la Tabla 3 y se procede a realizar las pruebas variando la probabilidad de mutación en un intervalo comprendido entre 0,005 % y 0,10 %.

**Tabla 3.**

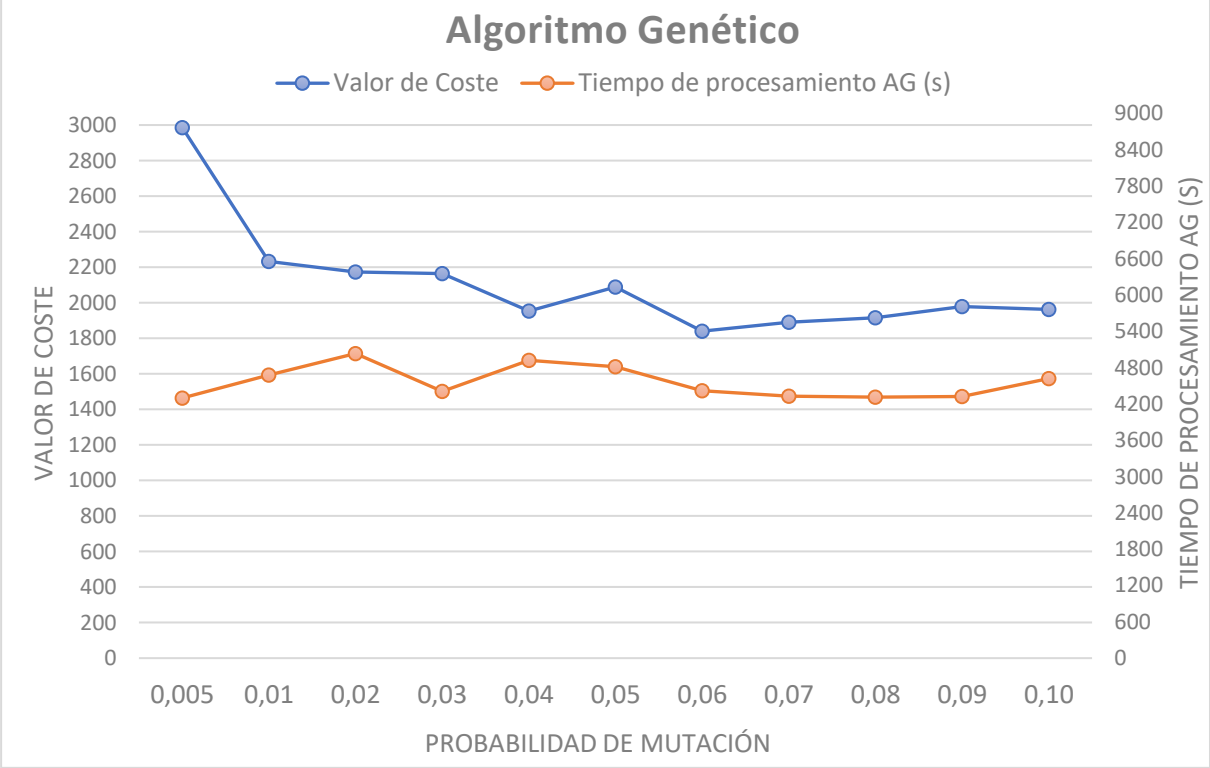
*Parámetros de simulación para evaluar la probabilidad de mutación*

| Parámetros | Valor |
|---|---|
| Cruce | 2 puntos |
| Probabilidad de Mutación | 0,005 – 0,10 % |
| Tamaño de la Población | 100 |
| Operador de Selección | Ruleta |

Luego de realizar las pruebas con respecto a la probabilidad de mutación, en la Figura 24 se puede apreciar los valores de coste que se obtienen al aplicar distintas tasas de probabilidad de mutación, también se evidencia como aumenta y disminuye el tiempo de procesamiento del algoritmo genético para cada valor de probabilidad de mutación que se ha configurado.

**Figura 24.**

*Algoritmo genético variando la probabilidad de mutación*




### *5.8.3. Número de iteraciones*

Para realizar las pruebas con respecto al número de iteraciones, se fija el tamaño de la población de soluciones en función de los resultados anteriores y se modifica entre un rango de 10 a 150 el número de iteraciones. Los parámetros que se configuran en el algoritmo genético para determinar un número de iteraciones adecuado se muestran en la Tabla 4.

**Tabla 4.**

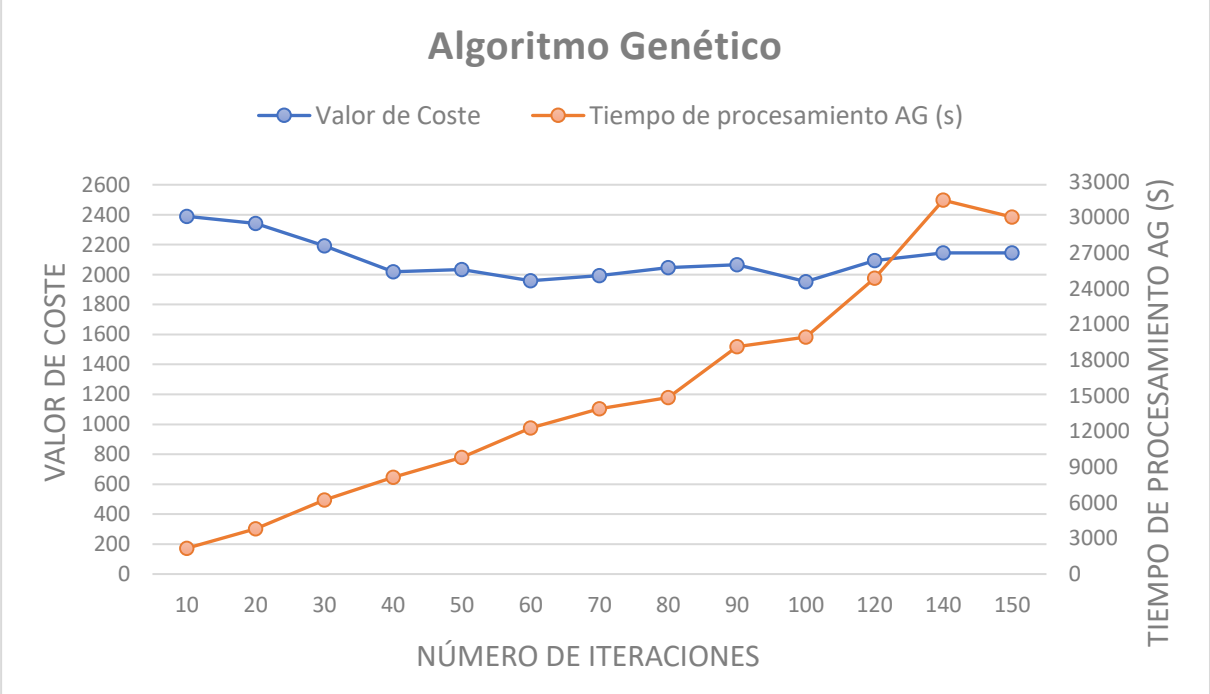
*Parámetros de simulación para evaluar el número de iteraciones*

| Parámetros | Valor |
|---|---|
| Cruce | 2 puntos |
| Tamaño de la Población | 100 |
| Probabilidad de Mutación | 0,06 % |
| Número de iteraciones | 10 -150 |
| Operador de Selección | Ruleta |

En la Figura 25 se puede apreciar que a medida que se incrementa el número de iteraciones, los valores de coste de las soluciones de la población empiezan a mantener cierta estabilidad, en cambio, el tiempo de procesamiento del algoritmo genético sigue incrementado conforme aumenta el número de iteraciones, con esto se puede establecer un valor intermedio de número de iteraciones basándonos en el valor de coste que se obtiene de las soluciones y del tiempo de procesamiento del algoritmo genético.

**Figura 25.**

*Algoritmo genético variando el número de iteraciones*




## 5.9. Parámetros Finales de Evaluación para el AG

Los valores finales que se utilizan para evaluar el desempeño del algoritmo genético se presentan en la Tabla 5, estos valores se configuran en base a los resultados de las pruebas realizadas anteriormente.

**Tabla 5.**

*Parámetros establecidos para el algoritmo genético*

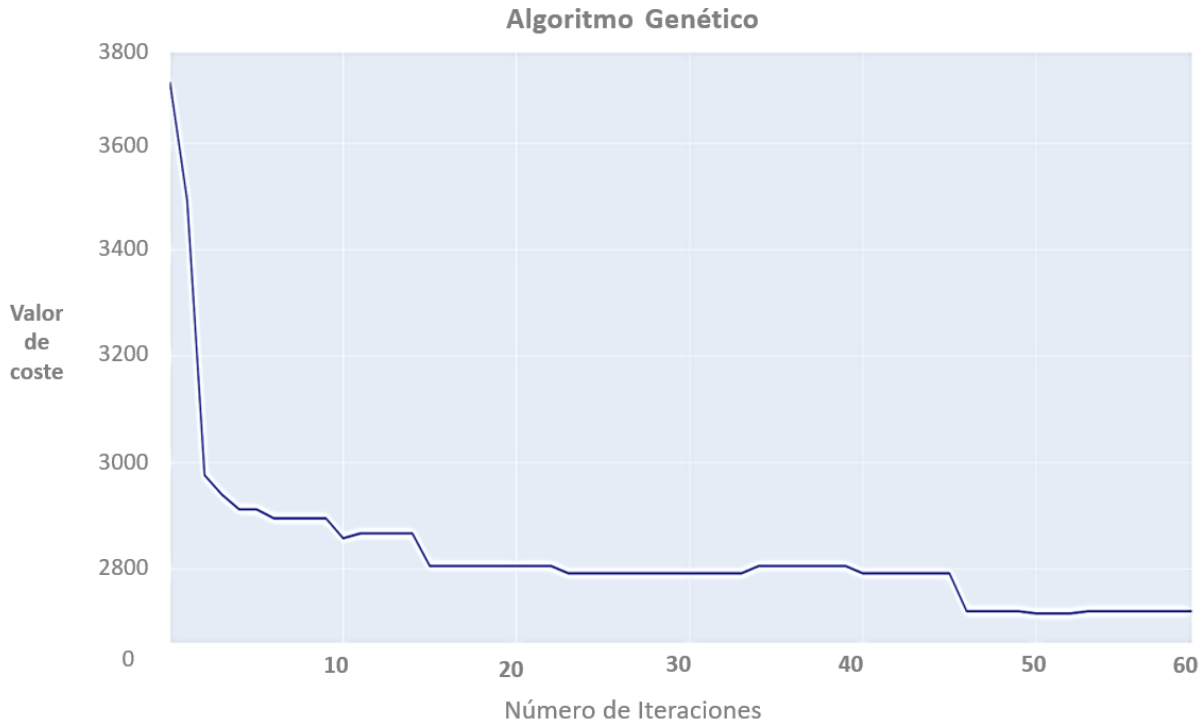
| Parámetros | Valor |
|---|---|
| Cruce | 2 puntos |
| Número de Iteraciones | 60 |
| Operador de Selección | Ruleta |
| Probabilidad de Mutación | 0,06 % |
| Tamaño de la Población de Soluciones | 100 |

*Nota*. Los parámetros que se presentan en la Tabla 5 permiten configurar el algoritmo genético con la finalidad de encontrar la mejor solución en el menor tiempo posible.

Luego de realizar diez ejecuciones del algoritmo genético con los parámetros presentados en la Tabla 5, en la Figura 26 se muestra el proceso de convergencia, en donde se puede analizar que antes de la iteración número 50 las soluciones empiezan a converger y se mantienen constantes hasta que se cumple con el número de iteraciones establecido.

**Figura 26.**

*Proceso de evolución del algoritmo genético*




*Nota.* El número de iteraciones es el criterio de parada que se establece para terminar el algoritmo genético, y el valor de coste en cambio indica como se va encontrando mejores soluciones a medida que avanza el proceso iterativo que realiza el algoritmo genético.

A continuación, en la Tabla 6 se presenta un promedio de los resultados tanto de las variables de valor de coste de las soluciones, tiempo de espera, vehículos en destino, y de las emisiones de Dióxido de Carbono (CO2), Monóxido de Carbono (CO), Hidrocarburos No Quemados (HC), Óxidos de Nitrógeno (NOx), Material Particulado (PMx), y Consumo de Combustible, que se generan de las diez ejecuciones del algoritmo genético realizadas con los parámetros presentados en la Tabla 5.

**Tabla 6.**

*Emisiones registradas de evaluación de soluciones del Algoritmo Genético*

| **Parámetros** | **Valor** |
|---|---|
| Valor de Coste | 2129,90 |
| Tiempo Espera | 48,34 (s) |
| Duración del viaje | 126,51 (s) |
| Vehículos en Destino | 87/100 |
| Dióxido de Carbono (CO2) | 18440,63 (mg/s) |
| Monóxido de Carbono (CO) | 4253,09 (mg/s) |
| Hidrocarburos No Quemados (HC) | 2149,29 (mg/s) |
| Óxidos de Nitrógeno (NOx) | 4084,73 (mg/s) |
| Material Particulado (PMx) | 2188,65 (mg/s) |
| Consumo de Combustible | 14704,89 (mg/s) |

*Nota*. Se debe tener en cuenta que la unidad de salida relacionada a las emisiones de combustible cambió en SUMO desde la versión 1.14.0 de litros a miligramos por segundo (mg/s).

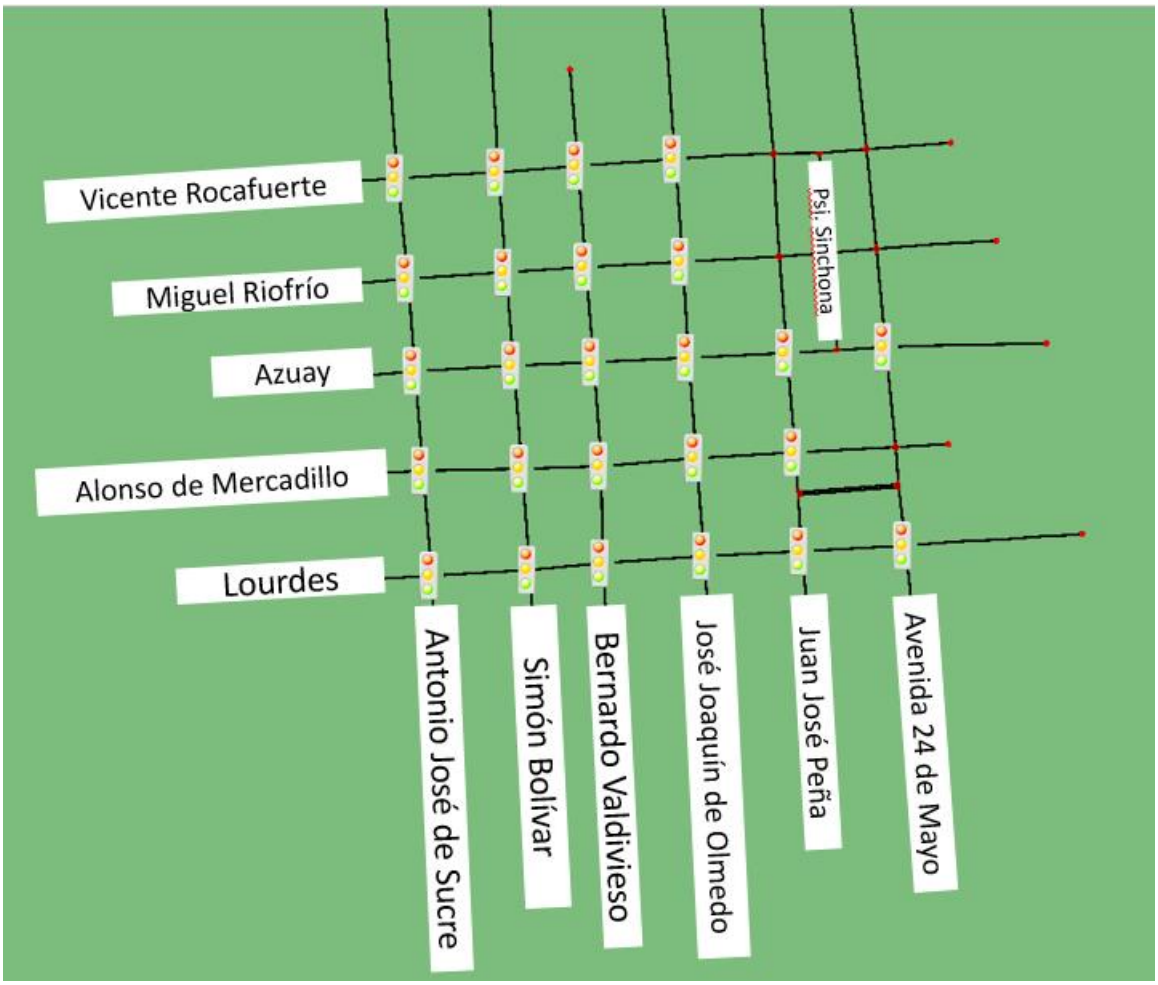
## 6. Resultados

En este apartado se realiza una comparación entre los resultados que se generan utilizando el Simulador de Microtráfico SUMO con la distribución de los semáforos con la que cuenta el escenario actual, y de utilizar el Simulador SUMO con la solución propuesta mediante el algoritmo genético.

En la Figura 27 se muestra el escenario actual a simular, en este se puede observar la distribución de los semáforos con los que actualmente cuenta la zona seleccionada de la ciudad de Loja.

**Figura 27.**

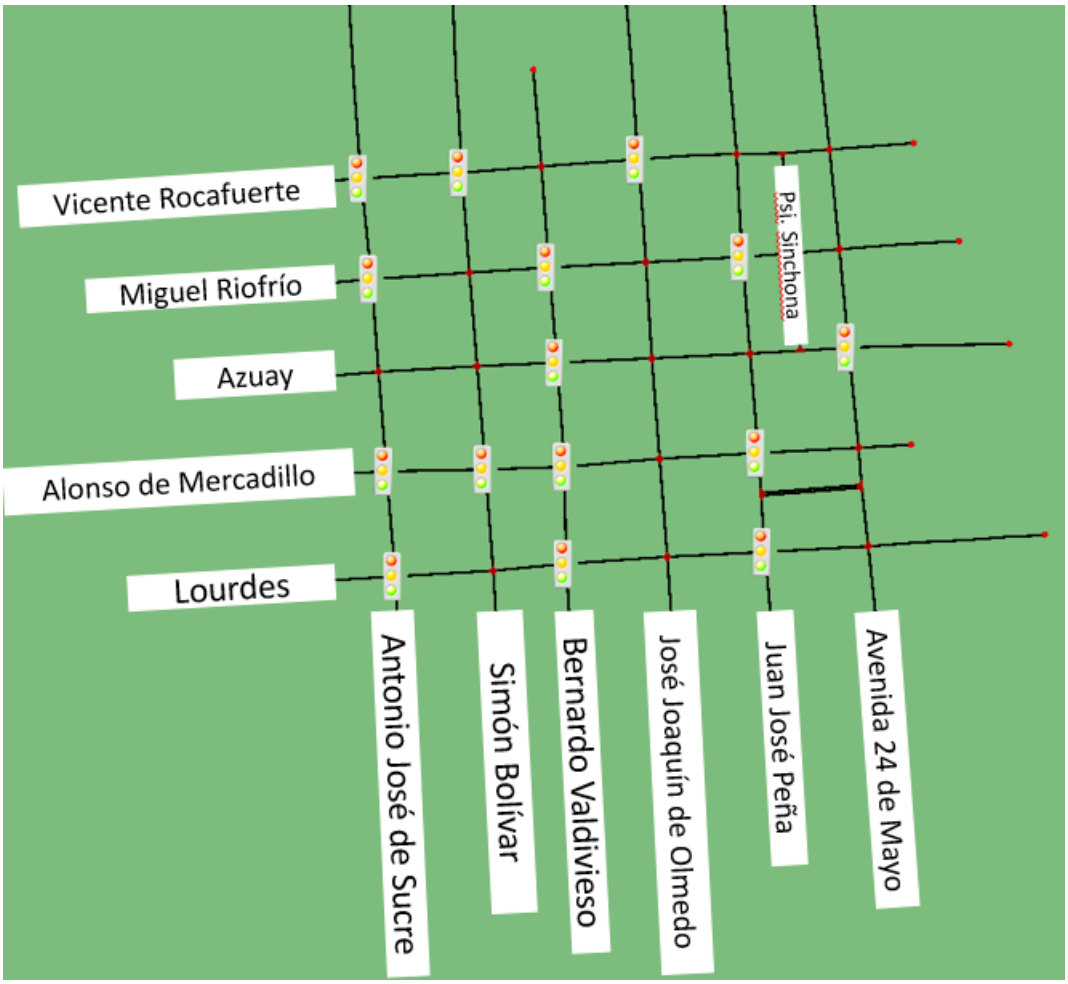
*Escenario con distribución actual de señales semafóricas*




En la Figura 28 en cambio tenemos el segundo escenario, en el cual se presenta la distribución de los semáforos utilizando la solución propuesta por el algoritmo genético.

**Figura 28.**

*Escenario con distribución de señales semafóricas con solución propuesta*

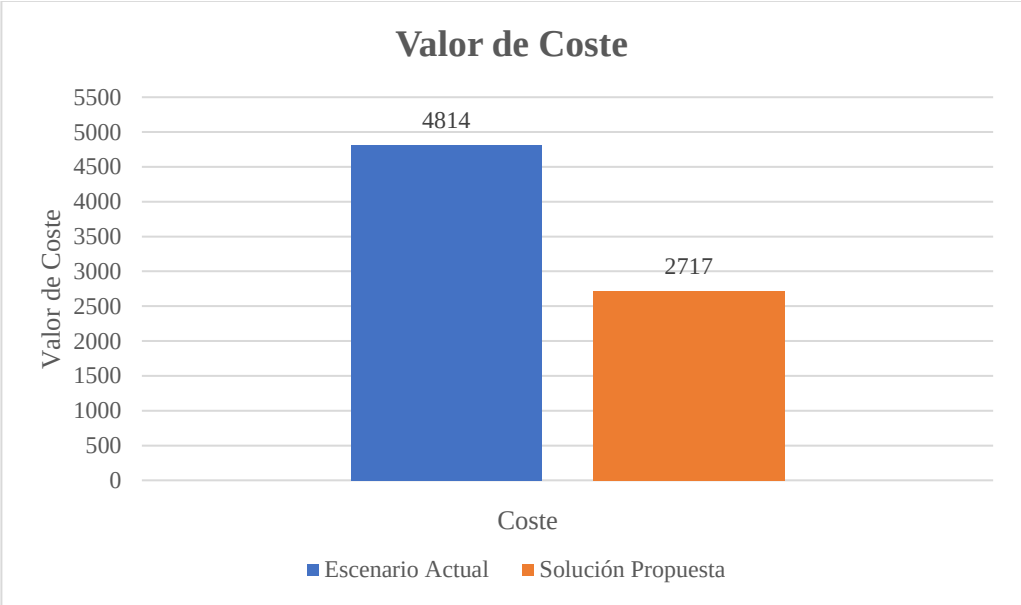



## 6.1. Valor de Coste

El valor de coste es el primer parámetro que se analiza, en la Figura 29 se muestra una comparación entre el valor de coste del escenario actual y del escenario con la solución propuesta. Como se puede observar, el primer escenario tiene un valor de coste alto con la distribución de semáforos que actualmente cuentan las intersecciones de la zona elegida, en cambio el segundo escenario minimiza el valor de coste con la solución propuesta para la distribución de las señales semafóricas en la red vial.

**Figura 29.**

*Comparación del valor de coste de las soluciones del escenario actual y escenario con la solución propuesta*



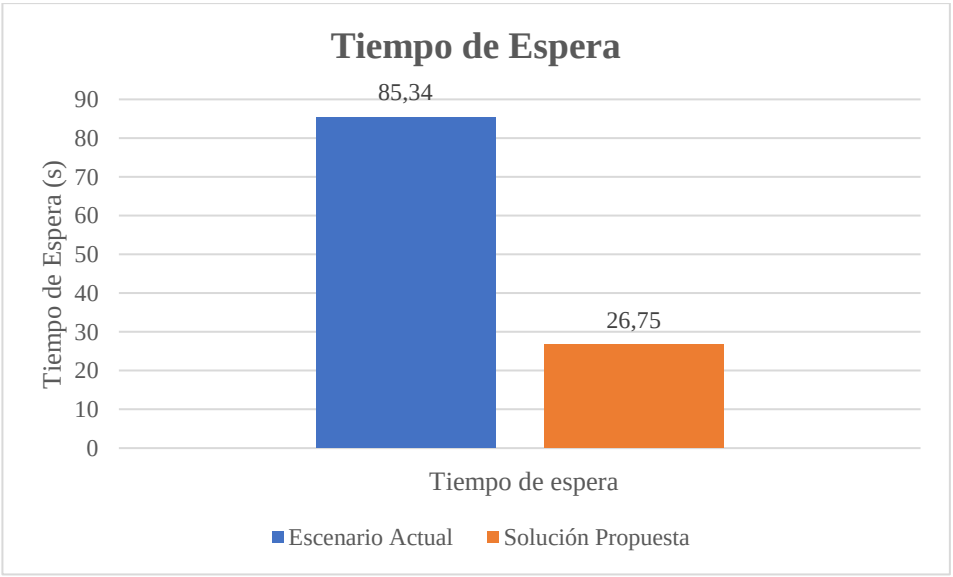

*Nota*. Se debe considerar que el simulador de microtráfico SUMO, se utiliza tanto para evaluar y simular las soluciones generadas mediante el algoritmo genético, por tal razón, el tiempo de procesamiento del algoritmo genético puede aumentar.

## 6.2. Tiempo de Espera

En este apartado, se realiza una comparación del tiempo en que la velocidad de los vehículos fue inferior o igual a 0,1 m/s, es decir, el tiempo de espera que los vehículos deben permanecer detenidos frente a los semáforos antes de que puedan volver a circular. La comparación realizada entre los dos escenarios se muestra en la Figura 30.

**Figura 30.**

*Comparación del tiempo de espera de los automóviles*

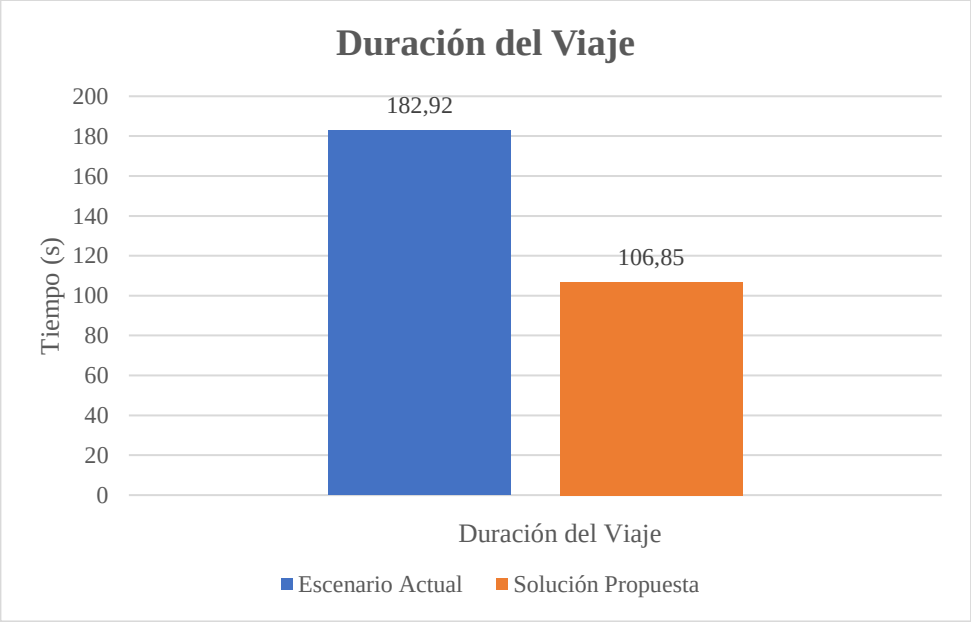


## 6.3. Duración del Viaje

La duración del viaje indica el tiempo que el automóvil necesita para completar la ruta, en la Figura 31 se muestra la comparación de la duración del viaje, tanto para el escenario actual y escenario con la solución propuesta, tomando en cuenta que se utilizan las mismas rutas generadas aleatoriamente mediante RamdomTrips.py para ambos escenarios.

**Figura 31.**

*Comparación de la duración del viaje de los vehículos*

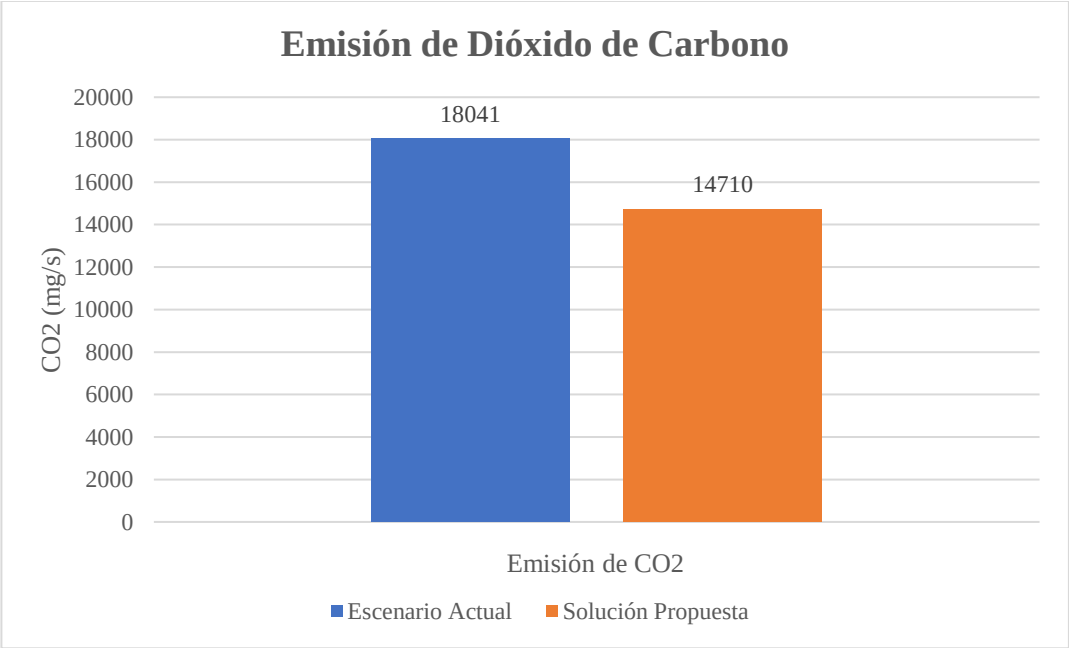



## 6.4. Emisión de Dióxido de Carbono (CO2)

En esta etapa se muestra una comparativa acerca de las emisiones de Dióxido de Carbono (CO2), en la Figura 32 se puede apreciar que se logra minimizar la cantidad de Dióxido de Carbono generado por los vehículos en el escenario con la solución propuesta, a diferencia de lo que ocurre en el escenario actual.

**Figura 32.**

*Comparación emisiones de dióxido de carbono*

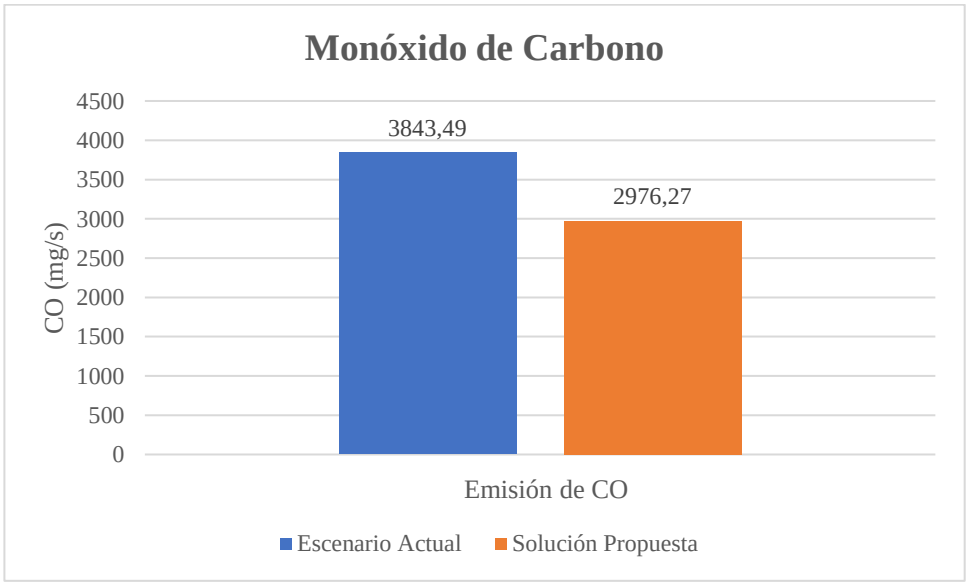



## 6.5. Emisión de Monóxido de Carbono (CO)

En esta fase se presenta una comparación sobre las emisiones de Monóxido de Carbono (CO), el cual es un gas tóxico que en grandes concentraciones puede perjudicar la salud de las personas, en la Figura 33 se puede visualizar que el escenario actual presenta un alto valor de emisiones de monóxido de carbono y en el segundo escenario con la solución propuesta se logra disminuir las emisiones de este tipo de gas.

**Figura 33.**

*Comparación de monóxido de carbono*

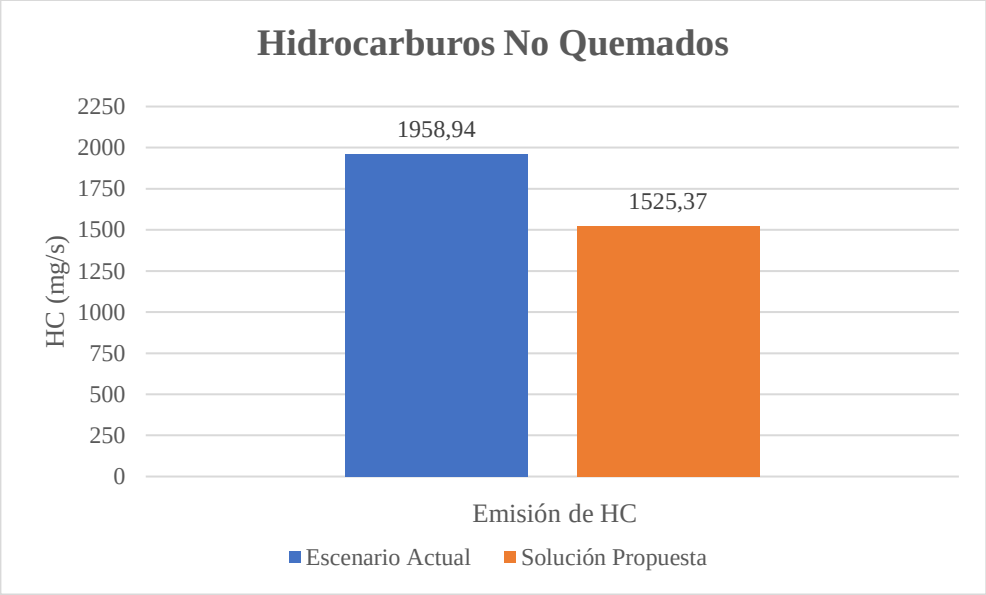


## 6.6. Emisión de Hidrocarburos No Quemados (HC)

La presente etapa incluye una evaluación comparativa acerca de la Emisión de Hidrocarburos No Quemados, el escenario actual genera niveles altos de HC y el segundo escenario con la solución propuesta logra minimizar las emisiones de HC, esta comparativa se puede apreciar en la Figura 34.

**Figura 34.**

*Comparativa de emisión de hidrocarburos no quemados*

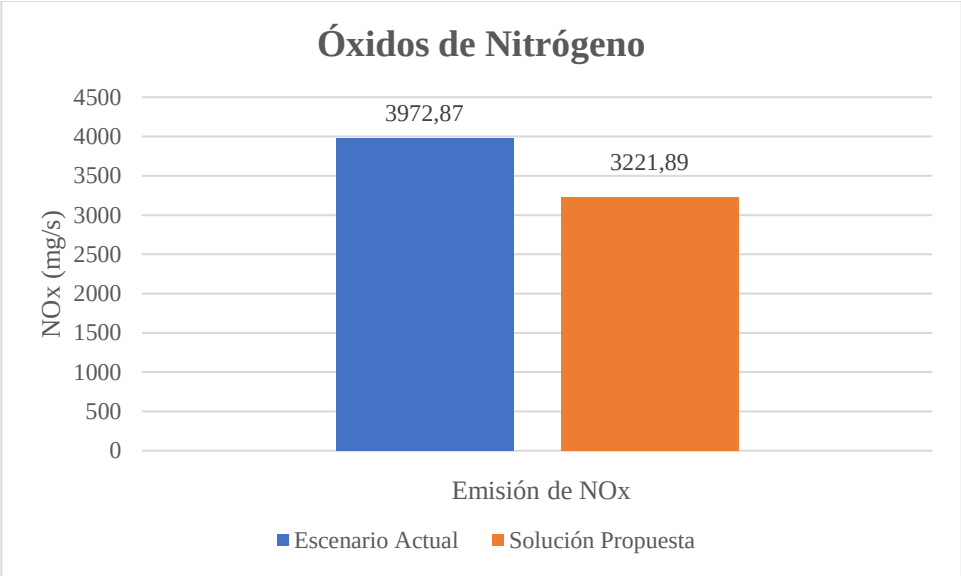



## 6.7. Emisión de Óxidos de Nitrógeno (NOx)

A continuación, la Figura 35 presenta una comparación acerca de las emisiones de Óxido de Nitrógeno de los dos escenarios, el escenario actual da origen a niveles altos de óxido de nitrógeno, en cambio el escenario con la solución propuesta permite la reducir este tipo de emisión, la razón por la cual se busca minimizar este tipo de gas contamínate, es porque contribuye a la formación ozono, y al reaccionar con otro tipo de compuestos en la atmósfera puede formar ácido nítrico, lo que aporta a la generación de lluvia ácida.

**Figura 35.**

*Comparación de emisiones de óxido de nitrógeno*

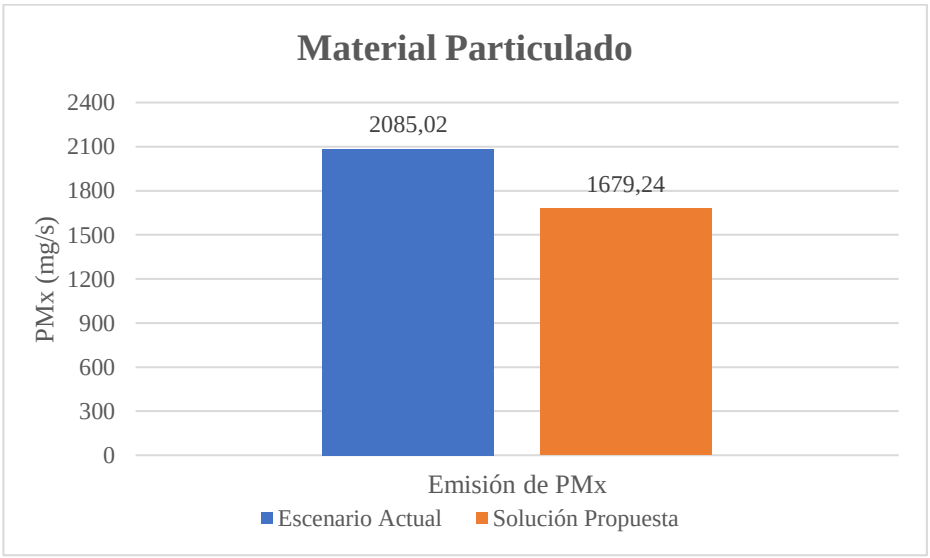



## 6.8. Emisión de Material Particulado (PMx)

La etapa actual presenta una comparativa acerca de los niveles de Material Particulado, la Figura 36 muestra la cantidad de material particulado que se genera tanto del escenario actual y del escenario con la solución propuesta. Se puede visualizar que, en el escenario actual genera una alta cantidad de material particulado, y en cambio en el escenario con la solución propuesta se evidencia una disminución en los niveles de material particulado.

**Figura 36.**

*Comparación de emisiones de material particulado*



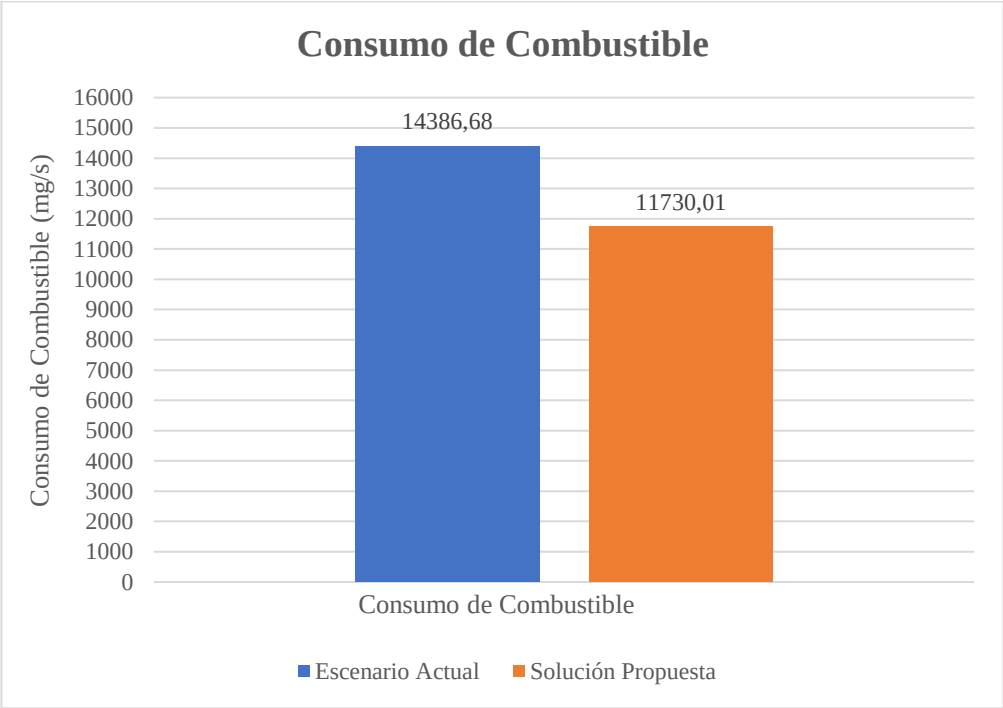

## 6.9. Consumo de Combustible

En la Figura 37 se muestra los niveles de consumo de combustible tanto para el escenario actual y el escenario con la solución propuesta, el escenario actual genera un alto

consumo de combustible y, por otro lado, el escenario con la solución propuesta logra disminuir el consumo de combustible.

**Figura 37.**

*Comparación consumo de combustible*




## 6.10. Número de vehículos

En esta etapa se presenta una comparativa de los niveles de emisiones y del número de vehículos que llegan a su destino, tanto para el escenario actual y el escenario con la solución propuesta. Para realizar esta prueba se aumenta a 201 el número de vehículos que se simulan en la red vial. En la Tabla 7 se presenta los niveles de emisiones registrados para el escenario actual y el número de vehículos que llegan a su destino para el escenario actual.

**Tabla 7.**

*Emisiones registradas del escenario actual*

| Parámetros | Valor |
|---|---|
| Vehículos en Destino | 120/201 |
| Dióxido de Carbono (CO2) | 30950,41 (mg/s) |
| Monóxido de Carbono (CO) | 7408,24 (mg/s) |
| Hidrocarburos No Quemados (HC) | 3730,30 (mg/s) |
| Óxidos de Nitrógeno (NOx) | 6885,08 (mg/s) |
| Material Particulado (PMx) | 3709,82 (mg/s) |
| Consumo de Combustible | 24680,48 (mg/s) |

En la Tabla 8 se presenta los niveles de emisiones registrados y el número de vehículos que llegan a su destino para el escenario con la solución propuesta.

**Tabla 8.**

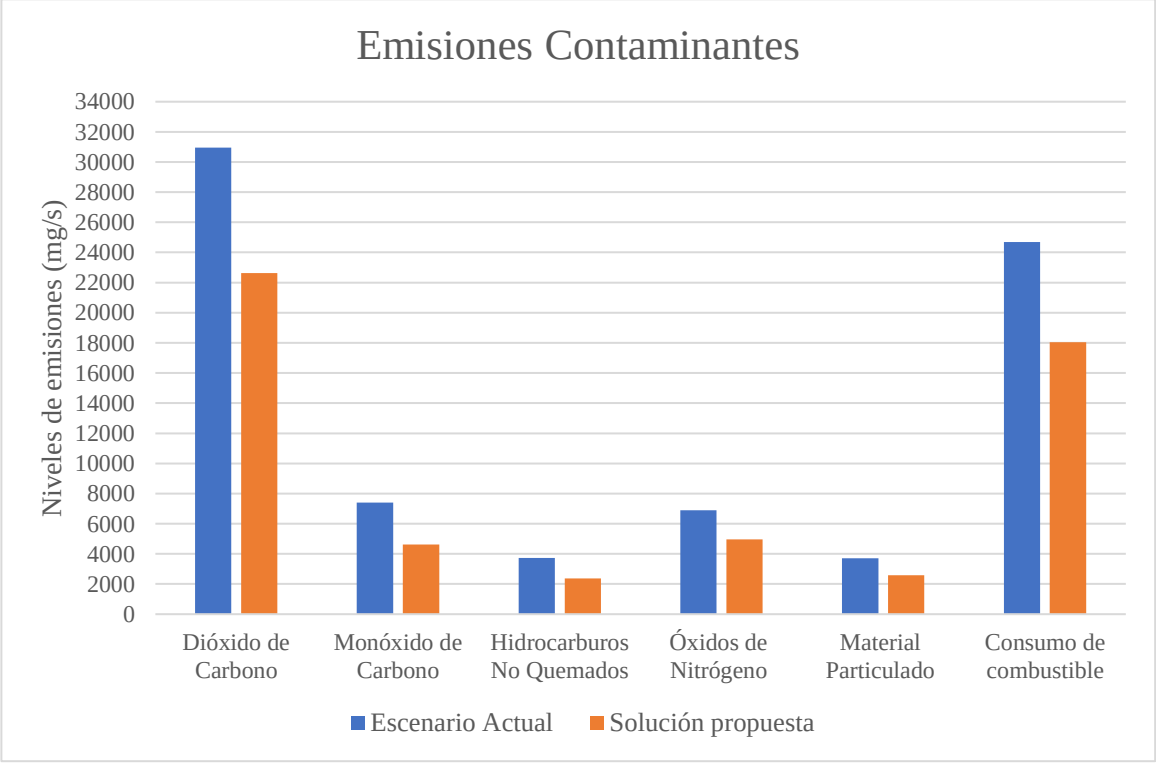
*Emisiones registradas del escenario con la solución propuesta*

| Parámetros | Valor |
|---|---|
| Vehículos en Destino | 158/201 |
| Dióxido de Carbono (CO2) | 22637,73 (mg/s) |
| Monóxido de Carbono (CO) | 4629,63 (mg/s) |
| Hidrocarburos No Quemados (HC) | 2369,86 (mg/s) |
| Óxidos de Nitrógeno (NOx) | 4962, (mg/s) |
| Material Particulado (PMx) | 2585,53 (mg/s) |
| Consumo de Combustible | 18051,60 (mg/s) |

A continuación, en la Figura 38 se presenta una comparación acerca de las emisiones contaminantes de los dos escenarios, se puede observar que el escenario actual presenta niveles de emisiones contaminantes altos y el escenario con la solución propuesta ayuda a reducir las emisiones contaminantes.

**Figura 38.**

*Comparación de emisiones entre el escenario actual y escenario con solución propuesta*

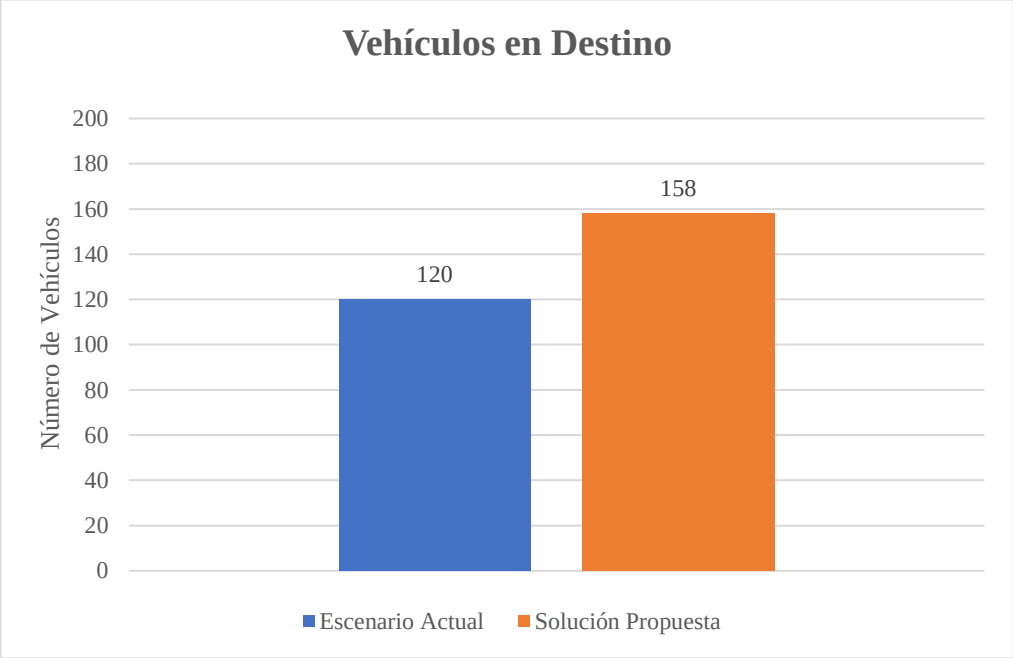


*Nota.* En la Figura 38 se puede apreciar los niveles de emisiones contaminantes del escenario actual y escenario con solución propuesta, la unidad de medida que se utiliza para medir los niveles de emisiones contaminantes es miligramos por segundo (mg/s).

En la Figura 39 se muestra una comparación del número de vehículos que llegan a su destino durante el tiempo que dura la simulación, tanto para el escenario actual y el escenario con la solución propuesta, se puede apreciar que el escenario con la solución propuesta maximiza el número de vehículos que llegan a su destino y en cambio en el escenario actual el número de vehículos que llegan a su destino es menor.

**Figura 39.**

*Vehículos en destino, escenario actual y escenario con solución propuesta*



## 7. Discusión

El escenario seleccionado de la red vial de la ciudad de Loja consta de 68 intersecciones o cruces, el principal objetivo de este trabajo de investigación, se enfoca en encontrar la mejor solución para la distribución de las señales semafóricas para las intersecciones del área delimitada.

Los resultados generados en el apartado de pruebas son prometedores y permiten asegurar que las técnicas metaheurísticas con el adecuado tratamiento y modelamiento de los datos, pueden ayudar en el diseño y desarrollo de herramientas para la optimización del tráfico vehicular de las redes urbanas, como lo respalda, Oliviera et al. (2015) que propuso un enfoque de Inteligencia de Enjambre, para la programación óptima de los programas de sincronización de semáforos para urbes de tamaño metropolitano y que obtuvo reducciones significativas en las tasas de emisiones y consumo total de combustible.

Sumado a lo expuesto anteriormente, Abu-Shawish et al. (2020) demostró que la integración de técnicas de optimización en el campo de control de señales semafóricas tiene un gran impacto en el rendimiento del control del tráfico vehicular en muchas ciudades del mundo, en este trabajo su principal objetivo fue maximizar el número de vehículos que llegaban a su destino o abandonaban la red en un periodo de tiempo determinado, con la finalidad de disminuir la congestión vehicular en las carreteras.

Esto demuestra que las técnicas Metaheurísticas permiten solucionar problemas que presentan un alto valor de complejidad y que utilizando otras técnicas serían muy difíciles de solucionar.

# 8. Conclusiones

El presente trabajo de investigación basado en la aplicación de técnicas metaheurísticas, utilizando el enfoque de un algoritmo genético, para desarrollar una herramienta que permita generar mapas de ubicación de señales semafóricas, permite obtener las siguientes conclusiones:

- En base a los objetivos planteados para dar cumplimiento a este trabajo de investigación, se ha desarrollado una herramienta que permita realizar la ubicación de las señales semafóricas en las intersecciones de una red vial de manera óptima para la ciudad de Loja, además, se ha logrado analizar y estudiar la arquitectura y lógica en la cual se basa el simulador de microtráfico SUMO, y el análisis de los resultados en base a los parámetros utilizados para su evaluación fueron favorables, ya que permiten reducir de manera satisfactoria los niveles de emisión de gases contaminantes en un 18,46% el Dióxido de Carbono CO2, Monóxido de Carbono CO un 22,56%, Hidrocarburos No Quemados HC un 22,13%, Óxidos de Nitrógeno NOx un 18,90%, Material Particulado PMx un 19,46 y el Consumo de Combustible en un 18,46%.
- La utilización de Simuladores de microtráfico para emular escenarios de tráfico vehicular, son una herramienta muy importante para analizar múltiples parámetros de contaminación de manera sencilla, pero se debe tener presente que el tiempo de procesamiento de los datos mediante el algoritmo genético puede incrementar, debido a la cantidad de parámetros que se analizan y del tamaño de las redes viales que se vayan a simular.
- El algoritmo genético desarrollado implementa una función multi-objetivo, que, junto al simulador de microtráfico SUMO, permite relacionar varios parámetros de emisiones contaminantes, lo que facilita el tratamiento de los datos que se desea minimizar.
- El estudio de las variables utilizadas en el modelamiento de la función de coste, sirve para evaluar las soluciones generadas por el algoritmo genético, lo que permite indagar que, el consumo de combustible es uno de los parámetros más importantes, en el análisis de las emisiones de gases contaminantes que se generan del parque automotor en una ciudad, esto en vista de que una disminución en el consumo de combustible por parte de los automotores que circulan por la red vial, tiene relación directa con la reducción de las emisiones de gases nocivos que afectan a la salud de las personas y cuidado del medio ambiente.

# 9. Recomendaciones

Las siguientes sugerencias que se detallan a continuación, se encaminan hacia futuras investigaciones para ayudar a la optimización del tráfico vehicular.

- Realizar múltiples pruebas con diferentes cantidades de vehículos y haciendo variar la probabilidad de aparición de los vehículos en la red vial, para analizar el funcionamiento y desempeño de la herramienta.
- El tamaño de los escenarios viales a extraer dependerá de la zona y complejidad del área que se desea analizar, es preferible que las redes viales que se vayan a simular, limiten el número de polígonos o adornos que se extraen de la red inicial, ya que estos, pueden aumentar el tiempo de procesamiento y ejecución del algoritmo genético.
- La exportación de mapas viales mediante la herramienta OpenStreetMap es una manera sencilla en que se puede extraer la red vial para ser simulada mediante SUMO, es importante considerar que por motivo de falta de investigaciones o datos concernientes a movilidad urbana en muchas partes del mundo, las redes viales extraídas pueden presentar incongruencias con las redes originales, por tal razón, es aconsejable asegurarse que cumplan con las características reales, y realizar las correcciones adecuadas antes de realizar la simulación.
- El trabajo desarrollado para la optimización de la ubicación de las señales semafóricas, puede ser utilizado en estudios futuros con investigaciones relacionadas a los programas de sincronización de semáforos de áreas metropolitanas, una solución híbrida de dichos enfoques podría mejorar la gestión del tráfico vehicular en las ciudades.

## 10. Bibliografía


Abarca, J. D. C. P. (2018). Metaheurísticas. Inventio, la génesis de la cultura universitaria en Morelos, 14(34), 25-32.

Abu-Shawish, I., Ghunaim, S., Azzeh, M., & Nassif, A. (2020) Metaheuristic Techniques in Optimizing Traffic Control Lights: A Systematic Review. International Journal of Systems Applications.

Aladdin, A., y Rashid, T. (2023). A New Lagrangian Problem Crossover—A Systematic Review and Meta-Analysis of Crossover Standards. Systems, 11(3), 144.

Algoritmos genéticos: *Funcionamiento, Pasos y Aplicaciones.* (2018). Tecnologias-informacion.com. [Citado el: 08 de febrero de 2022]. https://www.tecnologias-informacion.com/algoritmosgeneticos.html

Behrisch, M., Bieker, L., Erdmann, J., y Krajzewicz, D. (2011). SUMO–simulation of urban mobility: an overview. In Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation. ThinkMind.

Box Repsol. (2021). Diferencias entre la normativa euro 4 y la normativa Euro 5: sostenibilidad del motor. Box Repsol. https://www.boxrepsol.com/es/vive-tu-moto/diferencias-entre-la-normativa-euro-4-y-la-normativa-euro-5-sostenibilidad-del-motor/

Canales, F. (2021). Optimización de parámetros de la máquina virtual de Java con algoritmo genético. Disponible en https://repositorio.uchile.cl/handle/2250/183384

Caparrini, F., y Windmill, W. (2019) *Algoritmos genéticos*. Cs.Us.Es. http://www.cs.us.es/~fsancho/?e=65

Coloma, W. (2019) *Análisis de las herramientas de generación de demanda de tráfico en SUMO. Caso de estudio: Vías de acceso a Quito*. [Tesis de fin de Grado, Universidad Politécnica Nacional]. https://bibdigital.epn.edu.ec/bitstream/15000/20260/1/CD%209719.pdf

Construcción y Señalización Vial en Quito. (2023). https://www.dakmatraffic.com.ec/productos-de-alquiler

Crossover in genetic algorithm. (2019) GeeksforGeeks. [Citado el: 08 de febrero de 2022]. https://www.geeksforgeeks.org/crossover-in-genetic-algorithm/

Eclipse SUMO - simulation of urban MObility. (2022) *Eclipse SUMO - Simulation of Urban MObility*. Recuperado el 9 de febrero de 2022, de https://www.eclipse.org/sumo/

Fonseca, D. (2018) *Adaptación de una red neuronal para la negociación en el mercado de divisas*. Docplayer. [Tesis de Maestría, Universidad de Rosario].

https://docplayer.es/115893648-Adaptacion-de-una-red-neuronal-para-la-negociacion-en-el- mercado-de-divisas.html

Franco, L. (2014). Aplicación de Simulación en el Control de Tráfico, una Propuesta para Ciudad del Este. FPUNE Scientific, (4). http://servicios.fpune.edu.py:83/fpunescientific/index.php/fpunescientific/article/view/35/34

Garduño, R. (2018) *Algoritmos genéticos.* Universidad Nacional Autónoma de México. https://conogasi.org/articulos/algoritmos-geneticos/

González, N. (2020). Algoritmos evolutivos para el diseño estructural: estado del arte y caso estudio. Disponible en https://repositorio.uchile.cl/handle/2250/178216

Guananga, B., y Naranjo, E. (2023). Optimización de fresado de alta precisión con técnicas metaheurísticas e Inteligencia Artificial: revisión sistemática. Polo del Conocimiento, 8(3), 908-929. doi: http://dx.doi.org/10.23857/pc.v8i3.5342

Guerra, M., Pardo, E., y Salas, R. (2013). Problema del school timetabling y algoritmos genéticos: una revisión. Revista vínculos, 10(2), 259-276.

Gutiérrez, D., López, J., y Villa, W. (2016). Metaheuristic techniques applied to the optimal reactive power dispatch: a review. IEEE Latin America Transactions, 14(5), 2253-2263.

Hernández, M. (2021) *Diseño y aplicación de técnicas metaheurísticas para el control de tráfico*. [Tesis Doctoral, Universitat Politècnica de València]. RiuNet Repositorio Institucional UPV. http://hdl.handle.net/10251/174107

Instituto Geográfico Militar-Ecuador. (2022) *Ubicación Geográfica*. Obtenido de http://www.geograficomilitar.gob.ec/

Instituto Nacional de Estadística y Censos. (2021). *Estudios e Investigaciones*. Instituto Nacional de Estadísticas y Censos. Recuperado el 25 de abril de 2022, de https://www.ecuadorencifras.gob.ec/estudios-e-investigaciones/

Jamal, A., M. Al-Ahmadi, H., Muhammad Butt, F., Iqbal, M., Almoshaogeh, M. y Ali, S. (2021). *Metaheurística para el control y la optimización del tráfico: desafíos y perspectivas actuales.* IntechOpen. doi: 10.5772 / interchopen.99395

Jamshidi, M., Krohling, R., Coelho, L., y Fleming, P. (2003) *Robust Control Systems with Genetic Algorithms* (1st ed.). CRC Press. https://doi.org/10.1201/9781315219219

Joshi, D. (2021). Genetic Algorithm and its Applications-A Brief Study. Asian Journal For Convergence In Technology (AJCT) ISSN-2350-1146, 7(3), 8-12.

Katoch, S., Chauhan, S. S., y Kumar, V. (2021). A review on genetic algorithm: past, present, and future. Multimedia tools and applications, 80, 8091-8126.

Kicinger, R., Arciszewski, T., y De Jong, K (2005). Evolutionary computation and structural design: A survey of the state-of-the-art. Computers & Structures, 83(23-24), 1943–1978. doi: 10.1016/j.compstruc.2005.03.002

Kumari, P., y Singh, M. (2022). Different Challenges in Energy-Efficient Cloud Security: A Brief Review. Research Developments in Science and Technology Vol. 6, 112-122.

Mariñelarena, D., Quinatana, J., Carrión, F., Crespo, S., Guzmán, A., y López, J. (2013). Diseño de algoritmos genéticos para la detección de daños estructurales. [Publicación Técnica No. 386]. ResearchGate. Recuperado de https://www.researchgate.net/publication/315074828_Diseno_de_algoritmos_geneticos_para_la_ deteccion_de_danos_en_estructuras

Mendoza, P., y Villacis, C. (2014) *Análisis y solución al congestionamiento vehicular en horas pico utilizando una aplicación móvil con GPS*. [Tesis de Grado, Universidad Politécnica Salesiana Sede Guayaquil]. Recuperado de https://dspace.ups.edu.ec/bitstream/123456789/6505/1/UPS-GT000596.pdf

Mirjalili, S. (2019) Genetic Algorithm. In: Evolutionary Algorithms and Neural Networks. Studies in Computational Intelligence, vol 780. Springer, Cham. https://doi.org/10.1007/978-3-319-93025-1_4

Mirjalili, S., Song Dong, J., Sadiq, AS., y Faris, H. (2020) *Algoritmo genético: teoría, revisión de literatura y aplicación en la reconstrucción de imágenes.* En: Mirjalili S., Song Dong J., Lewis A. (eds) Nature-Inspired Optimizers. Estudios en Inteligencia Computacional, vol 811. Springer, Cham. https://doi.org/10.1007/978-3-030-12127-3_5

Morán, R. (2021). Técnicas de inteligencia artificial y big data en la gestión óptima de parques fotovoltaicos: Estado del arte. (Trabajo Fin de Máster Inédito). Universidad de Sevilla, Sevilla.

Moreno, A. C., Bravo, D., y Moreno, M. (2021). Herramienta de simulación para evaluar configuraciones semafóricas. Revista Cubana De Transformación Digital, 2(1), 102–114. Recuperado de https://rctd.uic.cu/rctd/article/view/100

Moyano, C, y Ordoñez, A. (2016) *Diseño semafórico y señalización vial del centro urbano de Saraguro*. [Tesis de Fin de Grado, Universidad del Azuay]. Repositorio institucional de la Universidad Nacional del Azuay. https://dspace.uazuay.edu.ec/handle/datos/5852

Ogoño, J., y Orozco, L. (2020) Análisis del tránsito vehicular en las intersecciones viales en el centro histórico de la ciudad de Loja, determinando el nivel de servicio. [Tesis de Fin de Grado, Universidad Politécnica Salesiana]. Repositorio Institucional de la Universidad Politécnica Salesiana. https://dspace.ups.edu.ec/handle/123456789/19381

Olivera, A., García, J., y Alba, E. Reducing vehicle emissions and fuel consumption in the city by using particle swarm optimization. Appl Intell 42, 389–405 (2015). https://doi.org/10.1007/s10489-014-0604-3

Orozco, O., y Llano, G. (2014). Aplicaciones para redes VANET enfocada en la sostenibilidad ambiental, una revisión sistemática. Ciencia E Ingeniería Neogranadina, 24(2), 111–132. https://doi.org/10.18359/rcin.396

Pabon, J., Aizaga, M., Recalde, H., y Toasa, R. (2023). Revisión de literatura sobre impacto de la inteligencia artificial y su aplicación en el Ecuador. Revista Ibérica de Sistemas e Tecnologias de Informação, (E55), 100-113.

Rawat, B., Duwal, D., Phuyal, S., y Pant, A. (2022). A Comparative Review Between Various Selection Techniques In Genetic Algorithm For Finding Optimal Solutions.

Reglamento Técnico Ecuatoriano INEN 004 “Señalización Vial”. https://www.normalizacion.gob.ec/buzon/reglamentos/RTE-004-5.pdf

Rouhiainen, L. (2018). Inteligencia artificial 101 cosas que debes saber hoy sobre nuestro futuro. www.planetadelibros.com

Ruiz, K. (2015). Optimización del sistema semafórico en la ciudad de Barcelona (Bachelor's thesis, Universitat Politècnica de Catalunya).

Srivastava, R., y Kumar, V. (2020). Accident Avoidance Simulation using SUMO. 2020 9th International Conference System Modeling and Advancement in Research Trends (SMART), (), –. doi:10.1109/smart50582.2020.9337079

Toronto, C. (2021). Traffic Lights. City of Toronto. https://www.toronto.ca/services-payments/streets-parking-transportation/traffic-management/traffic-signals-street-signs/types-of-traffic-signals/traffic-lights/

Valencia, P. (1997). Optimización mediante algoritmos genéticos. In Anales del Instituto de Ingenieros de Chile (Vol. 109, No. 2, pp. 83-92).

Valencia, V. (2000). Principios sobre semáforos. Universidad Politécnica Salesiana]. Repositorio Institucional de la Universidad Nacional de Colombia. https://repositorio.unal.edu.co/handle/unal/21291

Yarasca, A., Campos, N., Sotomayor, J., Rabanal, E., y Perez, C. (2021). Optimization of urban transport in Lima applying the genetic algorithms Tabú and Colonia de Hormigas [Optimización del transporte urbano en Lima aplicando los algoritmos genéticos Tabú y Colonia de Hormigas].

Zainuddin, F., Abd Samad, F., y Tunggal, D. (2020). A review of crossover methods and problem representation of genetic algorithm in recent engineering applications. International Journal of Advanced Science and Technology, 29(6s), 759-769.

# 11. Anexos

**Anexo 1.** *Código fuente utilizado para el desarrollo del algoritmo genético*

```
# Importamos librerías a utilizar en el Algoritmo Genético #
############################################
import random
import numpy as np
import csv
import xml.dom.minidom
from xml.dom.minidom import parse, parseString
import os
from lxml import etree
import time
import xml.etree.ElementTree as ET
from xml.dom import minidom
from bs4 import BeautifulSoup as bs
import matplotlib.pyplot as plt
import psutil
import plotly.offline as offline
import plotly.graph_objs as go
import lxml.etree as ET
############################################
```

```
# Técnica utilizada para editar un archivo con extensión XML #
#############################################################################
global waitingTime # configuramos variable waitinTime como global
def editXML(filename):
    tree = ET.ElementTree(file=filename)
    root = tree.getroot()
    waitingTime=duration=timeLoss=CO2=fuel=routeLength=0 # inicialización
de variables utilizadas para la función de evaluación #
    Vehiculos_destino=1 # se iguala el valor a 1 para poder realizar la
evaluación, y para que los Veh_destino no sean 0 #
    for vehiculo in root.iter("tripinfo"):
        waitingTime+=float(vehiculo.get('waitingTime'))
        duration+=float(vehiculo.get('duration'))
        timeLoss+=float(vehiculo.get('timeLoss'))
        CO2+=float(vehiculo.find('emissions').get('CO2_abs'))
        fuel+=float(vehiculo.find('emissions').get('fuel_abs'))
        routeLength+=float(vehiculo.get('routeLength'))
        Vehiculos_destino+=1
    tree = ET.ElementTree(root)
    return waitingTime,duration,timeLoss,CO2,fuel,Vehiculos_destino
#############################################################################
```

```
# Obtención de ficheros para la extracción de datos #
#####################################################
```

```
f=os.popen("sumo -c demo003osm.sumocfg --device.emissions.probability 1 --
amitran-output armitran_out.xml --queue-output queue_out.xml --fcd-output
fcd_out.xml --lanechange-output change_carril.xml  --no-step-log true")
log_out = f.read()
############################################################
```

```
# Creación de ficheros para su análisis y extracción de datos necesarios
para el modelamiento #
############################################################
# archivo queue permite detectar la cola frente al cruce controlado/no
controlado #
fichero_1 = parse('queue_out.xml')
# red vial a simular, esta fichero reemplaza al fichero de red original en
cada simulación#
fichero_2 = parse('redmodificada.net.xml')
# fichero fcd_out.xml, contiene la ubicación y la velocidad junto con otra
información para cada vehículo en la red #
fichero_3 = parse('fcd_out.xml')
# fichero de rutas para los vehiculos #
fichero_4 = parse('alexander.routes.rou.xml')
# obtener elementos por nombre de etiqueta tlLogic #
tlLogic = fichero_2.getElementsByTagName("tlLogic")
# obtener elementos por nombre de etiqueta junction #
junction = fichero_2.getElementsByTagName("junction")
#Extraccion de individuo a partir de la red inicial #
# este fichero modifica la red vial#
tree = ET.parse("redmodificada.net.xml")
#Añadimos a la raiz el fichero de red #
root = tree.getroot()
# Codificación del individuo inicial que permite generar la población
inicial #
individual = []
# Recorremos todas las etiquetas junctions #
for junction in root.findall("junction"):
    # Codificamos en función del tipo de junction #
    junction_type = junction.get("type")
    # Junction controlada por semaforo #
    if junction_type == "traffic_light":
        individual.append(1)
    # Junction que carece de semaforo #
    elif junction_type == "priority":
        individual.append(0)
    # Junction de fin de via #
    elif junction_type == "dead_end":
        individual.append(2)
    # junction internas #
    elif junction_type == "internal": # junction internas #
        individual.append(3)
    # Se muestra el individuo codificado #
print("Individuo Inicial:")
print(individual) # Mostramos codificación del individuo #
print(len(individual)) # Muestra la ongitud del individuo #
```

```
#################################################################


#Analizamos el archivo tripinfos.xml#
#####################################
filename = "demo003output-tripinfos.xml"
tree = ET.ElementTree(file=filename)
root = tree.getroot()
timeLossInd=[]
for vehiculo in root.iter("tripinfo"):
    #print(type(vehiculo.get('duration')))
    #print(type(vehiculo.get('waitSteps')))
    #print(vehiculo.get('waitingTime'))
    timeLossInd.append(float(vehiculo.get('duration')) -
float(vehiculo.get('waitingTime')))
tree = ET.ElementTree(root)
duration_no_ws=(sum(timeLossInd))
print(sum(timeLossInd))
#####################################


#Elementos por nombre de etiqueta de los ficheros que se han generado#
#######################################################################
# fichero de redmodificada.net.xml, etiqueta connection
connections = fichero_2.getElementsByTagName("connection")
# fichero de redmodificada.net.xml, etiqueta vehicule
vehicle_routes = fichero_4.getElementsByTagName("vehicle")
#######################################################################


# Esta función permite modificar el archivo de red de SUMO y permite
calcular el valor de Fitness de las soluciones#
#############################################################################
def calcularFitnessSUMO(individuo):
    Fit=0
    linkIndex = 0
    tl = None
    tree = ET.parse("redmodificada.net.xml")
    root = tree.getroot()
    # list to store the intLanes values
    intLanes = []
    # Inicio de creacion de junction
    for i, junction in enumerate(root.findall("junction")):
        if individuo[i] == 1:
            junction.set("type", "traffic_light")
        elif individuo[i] == 0:
            junction.set("type", "priority")
        elif individuo[i] == 2:
            junction.set("type", "dead_end")
        elif individuo[i] == 3:
            junction.set("type", "internal")
```

```
    # Iteramos sobre todos los elementos conexión
    for connection in root.iter("connection"):
        # eliminar los atributos tl y linkIndex existentes
        if "tl" in connection.attrib:
            del connection.attrib["tl"]
        if "linkIndex" in connection.attrib:
            del connection.attrib["linkIndex"]

     # Iteramos sobre todos los elementos junction
    for junction in root.findall("junction"):
        if junction.get("type") == "traffic_light":
            tl_id = junction.get("id")
            intLanes = junction.get("intLanes")
            for connection in root.findall("connection"):
                via = connection.get("via")
                if via:
                    if via in intLanes:
                        connection.set("tl", tl_id)
                        connection.set("linkIndex", str(linkIndex))
                        linkIndex += 1
            linkIndex = 0
    #Intrucciones para insertar los tlLogics en la red
    # Número de fases
    num_phases = 4

    # Duración del verde, estado, duración mínima y duración máxima
    green_duration = 42
    state = 'GGrr'
    min_dur = 5
    max_dur = 50

    # Contar cuántos elementos junction con type="traffic_light" existen y
almacenar sus IDs
    tl_ids = [junction.get('id') for junction in
root.findall(".//junction[@type='traffic_light']")]
    junction_count = len(tl_ids)
    #print(tl_ids)
    #print(len(tl_ids))

    # Obtener todas las etiquetas tlLogic
    tl_logics = root.findall('.//tlLogic')

    # Eliminar todas las etiquetas tlLogic excepto la primera
    for tl_logic in tl_logics[1:]:
        root.remove(tl_logic)

    # Contar cuántas etiquetas tlLogic existen y modificar sus IDs
    for i, tl_logic in enumerate(root.findall('.//tlLogic')):
        tl_logic.set('id', tl_ids[i])
        #print(len(tl_logic))
    # Configuración de las fases
    phases = [{'duration': '40', 'state': 'GGrr', 'minDur': '10', 'maxDur':
'50'}, {'duration': '5', 'state': 'yyrr'}, {'duration': '40', 'state':
'rrGG', 'minDur': '5', 'maxDur': '50'},          {'duration': '5', 'state':
'rryy'}]
```

```
    # Generar etiquetas tlLogic adicionales, si es necesario
    if junction_count > len(root.findall('.//tlLogic')):
        # Encontrar la última etiqueta tlLogic para insertar las etiquetas
faltantes
        last_tl_logic = root.findall('.//tlLogic')[-1]
        for tl_id in tl_ids[len(root.findall('.//tlLogic')):]:
            tl_logic = ET.Element('tlLogic', attrib={'id': tl_id, 'type':
'actuated', 'programID': '0', 'offset': '0'})
            for phase in phases:
                phase_attribs = {'duration': phase['duration'], 'state':
phase['state']}
                if 'minDur' in phase:
                    phase_attribs['minDur'] = phase['minDur']
                if 'maxDur' in phase:
                    phase_attribs['maxDur'] = phase['maxDur']
                phase_element = ET.Element('phase', attrib=phase_attribs)
                tl_logic.append(phase_element)
            root.insert(root.index(last_tl_logic) + 1, tl_logic)

    # Escribimos los datos actualizados en el archivo de red que se carga a
SUMO para la simulación
    tree.write("redmodificada.net.xml", xml_declaration=True,
encoding='UTF-8', method="xml")

    # la siguiente función nos permite realizar la evaluación de las
soluciones mediante el simulador, con esto se puede obtener el valor de
fitness de cada solucion
    f=os.popen("sumo -c demo003osm.sumocfg --device.emissions.probability 1
--no-step-log true --tripinfo-output demo003output-tripinfos.xml --
duration-log.statistics")
    log_out = f.read().split('\n')[-15:-2]

    # Extraccion de parámetros de simulación para el cálculo de la aptitud,
mediante la función editXML definidad al inicio
    waitingTime,duration,timeLoss,CO2,fuel,Vehiculos_destino =
editXML("demo003output-tripinfos.xml")

    # Calculo de la aptitud de las soluciones como una combinación de las
variables de consumo de combustible, duration, tiempo perdido y vehiculos
en destino

Fit+=((duration/1000)+(fuel/5000)+(timeLoss/1000))*Vehiculos_destino/100
    return (Fit)
#############################################################################
```

```
# Algoritmo Evolutivo con criterio de minimización de fitness para las
soluciones #
#############################################################################
#Tamaño de la población
```

```
size = 100
# Número de individuos que serán seleccionados en la población
num = 100
def generate_population(size, individual):
    # Creamos una lista vacía para almacenar la población
    poblacion = []
    # Obtenemos los índices de las posiciones que deben variar
    indices = [i for i in range(len(individual)) if individual[i] not in
[2, 3]]

    # Generamos la población de tamaño size
    for i in range(size):
        # Creamos una copia del individuo original
        new_individual = individual.copy()
        # Intercambiamos aleatoriamente las posiciones de los genes en el
individuo
        random.shuffle(indices)
        # Actualizamos los valores en las posiciones que deben variar
        for j, index in enumerate(indices):
            new_individual[index] = individual[index] if j < len(indices)
// 2 else 1 - individual[index]
        # Añadimos el nuevo individuo a la población
        poblacion.append(new_individual)
    return poblacion

#Función de selección por ruleta utilizando el simulador SUMO #
def roulette_select_SUMO(poblacion, num):
    """ Roulette selection, implemented according to:
        <http://stackoverflow.com/questions/177271/roulette
        -selection-in-genetic-algorithms/177278#177278>
    """
    fitness=[10/calcularFitnessSUMO(i) for i in poblacion]
    total_fitness = float(sum(fitness))
    rel_fitness = [f/total_fitness for f in fitness]
    # Generate probability intervals for each individual
    probs = [sum(rel_fitness[:i+1]) for i in range(len(rel_fitness))]
    # Draw new population
    new_population = []
    for n in range(num):
        r = random.uniform(0, 1)
        for (i, individuales) in enumerate(poblacion):
            if r <= probs[i]:
                new_population.append(individuales)
                break
    return new_population

# Función de seleccion y reprodución con cruce en dos puntos
padres_selecc=2
fitnessFinal=[]
fitness_optimo = float('inf')


def selection_and_reproduction(poblacion, fitnessFinal):
    global fitness_optimo, individuo_optimo
    # Seleccionar subpoblacion utilizando la funcion roulette_select_SUMO #
    subpoblacion = roulette_select_SUMO(poblacion, 30)
```

```
    puntuados = np.array([(calcularFitnessSUMO(i), i) for i in
subpoblacion], dtype=object)
    puntuados = puntuados[np.argsort(puntuados[:,0])[::-1]]

    # Imprimir el mejor individuo y su valor de fitness
    optimo_individual = puntuados[-1][1]
    optimo_fitness = puntuados[-1][0]
    print("Valor de fitness:", optimo_fitness)
    print("Individuo:",optimo_individual )
    fitnessFinal.append(optimo_fitness)

    # Seleccionar los dos más aptos
    padres = (puntuados[-1][1], puntuados[-2][1])

    # Realizar cruce en dos puntos entre los padres
    punto_cruce_1 = random.randint(1, len(padres[0])-2)
    punto_cruce_2 = random.randint(punto_cruce_1+1, len(padres[0])-1)
    hijo1 = padres[0][:punto_cruce_1] +
padres[1][punto_cruce_1:punto_cruce_2] + padres[0][punto_cruce_2:]
    hijo2 = padres[1][:punto_cruce_1] +
padres[0][punto_cruce_1:punto_cruce_2] + padres[1][punto_cruce_2:]

    # Modificar los hijos para cumplir con la especificación de la
codificación
    hijo1 = [padres[0][i] if hijo1[i] == 2 else padres[1][i] if hijo1[i] ==
3 else hijo1[i] for i in range(len(hijo1))]
    hijo2 = [padres[0][i] if hijo2[i] == 2 else padres[1][i] if hijo2[i] ==
3 else hijo2[i] for i in range(len(hijo2))]

    # Reemplazar los dos últimos individuos por los hijos generados
    poblacion[-2:] = [hijo1, hijo2]

    # Generar hijos a partir de los padres seleccionados
    hijos = [hijo1, hijo2] + [None]*(len(poblacion)-4)
    for i in range(0, len(hijos), 2):
        punto_cruce_1, punto_cruce_2 = random.sample(range(1,
len(padres[0])-1), 2)
        punto_cruce_1, punto_cruce_2 = min(punto_cruce_1, punto_cruce_2),
max(punto_cruce_1, punto_cruce_2)

        # Realizar cruce en dos puntos entre los padres
        hijo1 = padres[0][:punto_cruce_1] +
padres[1][punto_cruce_1:punto_cruce_2] + padres[0][punto_cruce_2:]
        hijo2 = padres[1][:punto_cruce_1] +
padres[0][punto_cruce_1:punto_cruce_2] + padres[1][punto_cruce_2:]

        # Modificar los hijos para cumplir con la especificación de la
codificación
        hijo1 = [padres[0][j] if hijo1[j] == 2 else padres[1][j] if
hijo1[j] == 3 else hijo1[j] for j in range(len(hijo1))]
        hijo2 = [padres[0][j] if hijo2[j] == 2 else padres[1][j] if
hijo2[j] == 3 else hijo2[j] for j in range(len(hijo2))]
        hijos[i] = hijo1
        hijos[i+1] = hijo2
    # Actualizar la población con los hijos generados
    poblacion[:-4] = hijos
```

```
    poblacion[-4:] = padres
    # Devolver la población actualizada
    return poblacion, puntuados

#Función de mutación
prob_mutacion=0.06 # Valor de probabilidad de mutación
def mutation(poblacion, prob_mutacion):
    for i in range(len(poblacion)):
        for j in range(len(poblacion[i])):
            if random.random() < prob_mutacion:
                if poblacion[i][j] == 1:
                    poblacion[i][j] = 0
                elif poblacion[i][j] == 0:
                    poblacion[i][j] = 1
    return poblacion

start = time.time()
#Evolución del algoritmo
# Definimos el número de iteraciones para evolucionar el algoritmo genético
iteraciones=60
# Genera nueva población aleatoria
poblacion = generate_population(size, individual)
for i in range(generaciones):
    print('iteracion: ',i)
    # Aplicar selección y reproducción
    poblacion, puntuados = selection_and_reproduction(poblacion,
fitnessFinal)
    poblacion = mutation(poblacion, prob_mutacion)
    # Actualizar el valor de fitness mínimo si se encuentra un valor menor
    if puntuados[-1][0] < fitness_optimo:
        fitness_optimo = puntuados[-1][0]
        individuo_optimo = puntuados[-1][1]

# Imprimir el individuo con el mejor valor de fitness
print("Valor de fitness optimo:", fitness_optimo)
print("Individuo con el mejor valor de fitness:", puntuados[-1][1])
end = time.time()

#Graficar el valor del mejor fitness para cada generación
plt.plot(range(1, generaciones+1), fitnessFinal)
plt.xlabel("Generación")
plt.ylabel("Fitness")
plt.title("Evolución del fitness")
plt.show()

print('time')
print(end - start)
############################################################




# Realizamos la simulación con la solución del escenario 1
############################################################
```

```
mejor_individuo = [1, 2, 1, 1, 0, 1, 1, 1, 2, 2, 0, 0, 0, 2, 1, 1, 1, 1, 0,
0, 1, 1, 2, 1, 2, 1, 2, 2, 1, 2, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 2, 2,
2, 2, 2, 2, 2, 0, 2, 2, 2, 0, 1, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
#En este caso el mejor_individuo corresponde al individuo que se conforma
de la distribución de semáforos del escenario real, que se denomina
escenario 1
calcularFitnessSUMO(mejor_individuo)
############################################################



# Función para extraer los valores de CO2, CO, HC, NOx, PMx, fuel generados
de la simulación #
#########################################################################
# Cargamos el archivo XML de emisiones que se genera de la simulación
tree = ET.parse('emisiones.xml')
root = tree.getroot()

# Creamos variables para las sumas de CO2, CO, HC, NOx, PMx, fuel
total_co2 = 0
total_co = 0
total_hc = 0
total_nox = 0
total_pmx = 0
total_fuel = 0

# Recorremos los elementos "vehicle" en el archivo XML
for vehicle in root.findall('.//vehicle'):

    # Obtenemos los valores de CO2, CO, HC, NOx, PMx, fuel de cada elemento
"vehicle"
    co2 = float(vehicle.get('CO2'))
    co = float(vehicle.get('CO'))
    hc = float(vehicle.get('HC'))
    nox = float(vehicle.get('NOx'))
    pmx = float(vehicle.get('PMx'))
    fuel = float(vehicle.get('fuel'))

    # Sumamos los valores de CO2, CO, HC, NOx, PMx, fuel a las variables
correspondientes
     # Escalamos los valores de cada variable a una escala de 0 a 1
    total_co2 += co2/ 5000  # El valor de normalización de CO2 es 5000
    total_co += co/ 1000  # El valor de normalización de CO es 1000
    total_hc += hc/ 10  # El valor de normalización de HC es 10
    total_nox += nox/ 10  # El valor de normalización de NOx es 10
    total_pmx += pmx/ 1  # El valor de normalización de PMx es 1
    total_fuel += fuel/ 2000  # El valor de normalización de fuel es 2000

# Imprimimos las sumas de CO2, CO, HC, NOx, PMx, fuel por separado, junto a
sus unidades de medida
print("Total de emisiones de CO2: ", total_co2, "mg/s")
print("Total de emisiones de CO: ", total_co, "mg/s")
print("Total de emisiones de HC: ", total_hc, "mg/s")
print("Total de emisiones de NOx: ", total_nox, "mg/s")
print("Total de emisiones de PMx: ", total_pmx, "mg/s")
print("Total de consumo de combustible: ", total_fuel, "mg/s")
```

##########################################################################

**Anexo 2.** *Certificación de traducción del abstract*

*Mg. Yanina Quizhpe Espinoza*
*Licenciada en Ciencias de Educación mención Inglés*
*Magister en Traducción y mediación cultural*

Celular: 0989805087
Email: yaniqes@icloud.com
Loja, Ecuador 110104

Loja, 18 de julio de 2023

Yo, Lic. Yanina Quizhpe Espinoza, con cédula de identidad 1104337553, docente del Instituto de Idiomas de la Universidad Nacional de Loja, y certificada como traductora e interprete en la Senescyt y en el Ministerio de trabajo del Ecuador con registro **MDT-3104-CCL-252640,** certifico:

Que tengo el conocimiento y dominio de los idiomas español e inglés y que la traducción del resumen del Trabajo de Titulación **"Técnicas Inteligentes para la Generación de Mapas de Ubicación de Señales Semafóricas que Minimicen el Tráfico en la Ciudad de Loja",** de autoría del estudiante Mario Alexander Moreno González, con cédula 1105757304, es verdadero y correcto a mi mejor saber y entender.

Atentamente

YANINA BELEN QUIZHPE ESPINOZA
Firmado digitalmente por YANINA BELEN QUIZHPE ESPINOZA
Fecha: 2023.07.18 11:30:04 -05'00'

Yanina Quizhpe Espinoza.
**Traductora Freelance**

*Full text translator: servicios de traducción*