Universidad Autónoma de Sinaloa

Facultad de Informática Culiacán
Facultad de Ciencias de la Tierra y el Espacio

Maestría en Ciencias de la Información

Optimización colectiva en rutas usando best response dynamics

TESIS

COMO REQUISITO PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA INFORMACIÓN

PRESENTA:

MARÍA DE LOURDES ANGULO DOMÍNGUEZ

DIRECTORES DE TESIS:

DR. ARTURO YEE RENDÓN

DR. ROLANDO MENCHACA MÉNDEZ

CULIACÁN, SINALOA, MÉXICO FEBRERO DEL 2019

Dedicatoria

Dedico mi presente tesis, a mis padres Juan Cecilio Angulo Armenta y Lourdes Domínguez Castro por todo su apoyo, amor, comprensión y confianza, por motivarme en mi preparación académica y ser un ejemplo y motivación en mi vida. Así también a mi hermosa hermana Cecilia Angulo Domínguez, por hacer divertida nuestra convivencia y confiar en mi . A mi esposo José Luis García Malacara, al ser mi compañero de vida que siempre me oriento cuando no encontraba una respuesta, así también al brindarme su amor y ser una motivación en mi vida. A mi tia Azucena Angulo por brindarme su casa mientras estudiaba la maestría. A Doña Silvia Mascareño mi mejor amiga, que siempre ha estado al pendiente de mi. Agradezco sinceramente a Ramoncita por ser mi compañera y amiga durante mis estudios de maestría y salidas a comer, aunque no podrá leer esto, se que estarás en mi presentación por que aunque pareciera estar muy lejos su recuerdo permanece en mi corazón, y no olvidare lo que un día me dijo. Por último me queda agradecer a toda mi gran familia. Muchas gracias.

Agradecimientos

A la Universidad Autónoma de Sinaloa por ser mi casa de estudios desde la licenciatura y por ayudarme a crecer de manera académica y personal.

Al CONACyT por otorgarme la beca con la cual pude dedicarme por completo a mis estudios de maestría.

Agradezco ampliamente a mis asesores Dr. Arturo Yee Rendón y Dr. Rolando Menchaca Méndez, por todo su tiempo, dedicación y comprensión invertidos en formarme en el ámbito de la investigación y lograr terminar mis estudios de maestría con la presente tesis. Los admiro como los buenos investigadores que son y sus amplios conocimientos.

Muchas gracias a nuestra coordinadora la Dra. Xiomara Penelope Zaldivar Colado por apoyarme en todas las actividades académicas que realicé, por su interés en mi bienestar y apoyo incondicional.

A nuestro director de la Facultad de Informática Culiacán, Mc. Gerardo Gálvez por que su apoyo fue parte importarte en el desarrollo de mi investigación.

A mis profesores el Mc. Gerardo Beltrán, Dr. Jorge Adalberto Navarro Castillo, Dr. Jesús Roberto Millán Almaraz, Dr. Inés Fernando Vega López, Dr. René Rodríguez Zamora y Mc. Zeus del Valle, por trasmitir sus conocimientos día a día en las materias que cursé, así también por ser algunos de ellos sinodales en mis presentaciones de avances de tesis y ayudarme en mi desarrollo como estudiante de maestría.

Así también agradezco a mis compañeros de generación Jesús Mojardín y Jessica

Urrea, y en especial a Mauro Montenegro por que independientemente de ser asesorados por el mismo Doctor, aprendimos a ser buenos amigos y durante nuestra estancia en la ciudad de México, siempre conté con su apoyo y nunca me dejó sola. A Dario Osuna y Graciela Rodríguez por su sincera amistad y por transmitir sus conocimientos y experiencias en muchas ocasiones. A Fernando Arce, Lucero Castañeda, Ricardo Gallego , Angel Chi Poot por coincidir en la estancia de investigación y llegar a ser buenos amigos. Como también, al resto de mis compañeros por su amistad y compañerismo.

Me queda agradecer al Centro de Investigación en Computación del IPN por brindarme todas atenciones para que realizara una estancia de investigación de cinco meses.

Abstract

Since 1980 in Mexico, cities with a population that ranges from 500,000 to one million inhabitants (average city) are characterized by being new urban centers with high population growth [1].

The city of Culiacán, Sinaloa with an estimated population of 905,265 [34] with a growth rate of 1.03 % is considered the typical average city. As a consequence of its population and its growth rate, the number of vehicles on the streets is increasing, generating traffic jams at certain times of the day [31]

In this thesis, we propose a new collective optimization scheme based on game theory to optimize the travel through the city, where users are treated in a collectively way and with the aim to prevent traffic jams.

The optimization scheme consists of representing a part of the road structure of the Culiacan city by means of a weighted directed graph, where its edges represent the streets and the vertices represent the intersections of the streets. The graph edges are weighted by a proposed mathematical congestion function, which considers characteristics of the streets, such as length, a number of lanes, maximum speed allowed and the number of vehicles that transit on. The proposed optimization scheme combines a game theory strategy: Best Response Dynamics for the collective route analysis, and Dijkstra for finding the shortest routes.

In the literature, there exist algorithms to optimize travel time of users routes, such as Google Maps, Waze and UPS, but they lack a proper route analysis, i.e., they do not consider a collective analysis of the different users that transit on the same

streets, so they do not prevent traffic jams.

Resumen

Los intensos cambios en la dinámica de crecimiento urbano en México desde 1980 muestran que las ciudades medias (de 500 mil a un millón de habitantes) se caracterizan por ser los nuevos centros urbanos de mayor crecimiento demográfico. Este patrón de urbanización se ha mantenido hasta principios del siglo XXI [1].

La ciudad de Culiacán, Sinaloa es una ciudad media en constante crecimiento demográfico, por lo que, se ha incrementado de manera considerable el número de autos que transitan por las vialidades, generando en ciertas horas, un aumento en el tráfico vehicular [31] [34]. En la presente tesis, se plantea una metodología para minimizar el tiempo de trayectos de forma colectiva de un conjunto de rutas y evitar que se produzca el fenómeno del congestionamiento vial. El caso de estudio es una sección de la estructura vial de la ciudad de Culiacán, Sinaloa. Ésta será representada mediante un grafo dirigido, donde sus aristas representan las calles de la ciudad y los nodos sus intersecciones. La dirección de sus aristas dependerá del sentido de las calles, y su peso, será asignado mediante una función matemática que modelará la congestión, dicha función se definirá en la parte metodológica del actual trabajo.

La causa fundamental de la congestión es la fricción entre los vehículos en el flujo de tránsito. Hasta un cierto nivel de tránsito, los vehículos pueden circular a una velocidad relativamente libre, determinada por los límites de velocidad, la frecuencia de las intersecciones, etc. Sin embargo, a volúmenes mayores, cada vehículo adicional estorba el desplazamiento de los demás, es decir, comienza el fenómeno de la congestión [27].

En la literatura, diferentes modelos y algoritmos computacionales se han desarrollado

para el diseño y modelado de transporte en redes de: Internet, eléctricas, agua y tráfico vehicular, en donde el esquema de funcionamiento busca minimizar el costo de operación de manera individualizada. La teoría de juegos proporciona un marco para modelar y analizar tales esquemas individualizados de forma colectiva [26]. En nuestra propuesta, utilizamos un esquema para optimizar de forma colectiva el tiempo de trayectos en una estructura vehicular a través de Best Response Dynamics.

Índice general

Dedicatoria I

Agradecimientos III

Abstract V

Resumen VII

Lista de Figuras XI

Lista de Tablas XIII

Lista de Ecuaciones XV

1 Introducción 1

1.1 Planteamiento del problema 1
1.2 Objetivo General y Específicos 5
1.3 Hipótesis 6
1.4 Organización de la tesis 6

2 Antecedentes 9

2.1 Teoría de juegos 9
2.2 Juegos cooperativos y no cooperativos 10
2.3 Equilibrio de Nash 12
2.4 Best Response Dynamics 13
2.5 Teoría de grafos 15
2.5.1 Representaciones de grafos 17

2.6 El problema de la ruta más corta 19
2.6.1 Algoritmos para encontrar la ruta más corta 19

3 Estado del arte **25**
3.1 Google Maps 25
3.2 United Parcel Service (UPS) 26
3.3 WAZE 26
3.4 Trabajos relacionados 29

4 Metodología **31**
4.1 Modelado formal 31
4.1.1 Grafo vial 32
4.1.2 Viajes 32
4.1.3 Función Objetivo 33
4.1.4 Función de congestión 33
4.1.5 Análisis de parámetros 37
4.1.6 Esquema de análisis de optimización colectiva 42
4.1.7 Análisis estratégico de rutas 43
4.2 Software Cartográfico OpenStreeMap 46
4.3 Simulador de rutas urbanas 49

5 Experimentación y resultados **53**
5.1 Resultados de las simulación en diferentes escenarios 53

6 Conclusiones y trabajos futuros **75**

A Python **79**

B Instalación y configuración de un ambiente en SUMO en sistema operativo Ubuntu **83**

C Canal de comunicación Python - SUMO **85**

D Resultados de las simulaciones en los diferentes escenarios. **86**

Bibliografía **99**

Lista de Figuras

1.1 Distribución de la población en el parque vehicular. 4

2.1 Ejemplo: Aplicando Best Response Dynamics a grafo dirigido ponderado 13
2.2 Grafo 16
2.3 (a) Grafo dirigido (b) Grafo no dirigido 17
2.4 Matriz de adyacencia Nodo a Nodo. 19
2.5 Operación triangular: Es la ruta (punteada) usando el vértice j como vértice intermedio más corto que la ruta (solida) sin usar el nodo j . 22

4.1 Descripción gráfica de la metodología propuesta. 31
4.2 Comportamiento de la función sigmoide. 34
4.3 Comportamiento de la función de congestión propuesta. 35
4.4 Puntos conocidos en la función de congestión. 38
4.5 Procedimiento. 45
4.6 Propiedades del mapa de forma gráfica. 49

5.1 Red vial de vehículos de la ciudad de Culiacán, Sinaloa. 54
5.2 Representación de la Figura 5.1, por medio de un Grafo y los vértices origen y destino de los viajes que se realizarán en las simulaciones. . . 54
5.3 Resultados del escenario 1 en FCP 57
5.4 Resultados del escenario 1 en SUMO 58
5.5 Resultados del escenario 2 en FCP 59
5.6 Resultados del escenario 2 en SUMO 60
5.7 Resultados del escenario 3 en FCP 61
5.8 Resultados del escenario 3 en SUMO 62
5.9 Resultados del escenario 4 en FCP 64
5.10 Resultados del escenario 4 en SUMO 65

5.11 Resultados del escenario 5 en FCP 66
5.12 Resultados del escenario 5 en SUMO 67
5.13 Carga vehicular en los escenarios 68
5.14 Resultados promedio de los cinco escenarios en SUMO 69
5.15 Resultados promedio de los cuatro escenarios en FCP 70
5.16 Comportamiento de los vehículos con Dijkstra en SUMO 71
5.17 Comportamiento de los vehículos con EOC en SUMO 72
5.18 EOC en SUMO 73
5.19 Comportamiento de los vehículos con Dijkstra en SUMO 73

6.1 Relación: EOC en SUMO Y FCP del escenario 1 76
6.2 Relación: EOC en SUMO Y FCP del escenario 2 77
6.3 Relación: EOC en SUMO Y FCP del escenario 3 77
6.4 Relación: EOC en SUMO Y FCP del escenario 4 78
6.5 Relación: EOC en SUMO Y FCP del escenario 5 78

Lista de Tablas

1.1 Crecimiento del parque vehicular respecto al número de habitantes a lo largo de los años . . . 3

2.1 Algoritmo de Dijkstra . . . 21
2.2 Algoritmo Floyd-Warshall . . . 23

4.1 Algoritmo de optimización colectiva (AOC) . . . 42
4.2 Representación del grafo en Python. . . . 49

5.1 Escenario 1: Sin carga vehicular en las calles . . . 56
5.2 Resultados del escenario 1 en FCP. . . . 56
5.3 Resultados del escenario 1 en SUMO. . . . 57
5.4 Escenario 2: carga vehicular del 0 % al 25 %. . . . 58
5.5 Resultados del escenario 2 en FCP. . . . 59
5.6 Resultados del escenario 2 en SUMO. . . . 60
5.7 Escenario 3: carga vehicular del 25 % al 50 % en FCP. . . . 60
5.8 Resultados 3: carga vehicular del 25 % al 50 % en FCP. . . . 61
5.9 Resultados 3: carga vehicular del 25 % al 50 % en SUMO. . . . 62
5.10 Escenario 4: carga vehicular del 50 % al 75 %. . . . 63
5.11 Resultados 4: carga vehicular del 50 % al 75 % en FCP. . . . 63
5.12 Resultados 4: carga vehicular del 50 % al 75 % en SUMO. . . . 64
5.13 Escenario 4: carga vehicular del 75 % al 100 %. . . . 65
5.14 Resultados 5: carga vehicular del 75 % al 100 % en FCP. . . . 66
5.15 Resultados 4: carga vehicular del 75 % al 100 % en SUMO. . . . 66

Lista de Ecuaciones

2.1 Estrategias del equilibrio de Nash y función de utilidad 12
4.1 Conjunto de viajes 32
4.2 Función objetivo 33
4.3 Función sigmoide 33
4.4 Función de congestión 34
4.5 Carga vehicular máxima de una arista 36
4.6 Tiempo mínimo 37
4.7 Tiempo máximo 37
4.11 Valor de la constante K4 39
4.14 Valor de la constante K1 40
4.19 Valor de la constante K2 40
4.25 Valor de la constante K3 41
B.1 Conjunto de rutas 84

Capítulo 1

Introducción

1.1. Planteamiento del problema

La movilidad urbana mantiene una relación entre la expansión espacial de la ciudad, las necesidades de traslado de la población, la oferta de infraestructura vial, la utilización de los espacios públicos, y los medios de trasporte utilizados para realizar viajes. Por lo tanto el funcionamiento adecuado de la movilidad urbana es indispensable para una ciudad.

Sin embargo, cuando la ciudad no refleja de forma positiva los tiempos de desplazamiento de los medios de trasporte, al contrario existe un constante incremento del tiempo, se dice que existe un problema de movilidad. Lo cual conlleva a que se presenten otros problemas tales como: la dificultad para acceder a los espacios de trabajo, educación, salud, vulnerabilidad ante riesgos, contaminación ambiental y auditiva por mencionar algunas, y un constante deterioro de la vida en ciudad [31]

La causa fundamental de la congestión es la fricción entre los vehículos en el flujo de tránsito. Hasta un cierto nivel de tránsito, los vehículos pueden circular a una velocidad relativamente libre, determinada por los límites de velocidad, la frecuencia de las intersecciones, etc. Sin embargo, a volúmenes mayores, cada vehículo adicional estorba el desplazamiento de los demás, es decir, comienza el fenómeno de la

congestión [27].

Relación vehículos y habitantes

Los intensos cambios en la dinámica de crecimiento urbano en México desde 1980 muestran que las ciudades medias (de 500 mil a un millón de habitantes) se caracterizan por ser los nuevos centros urbanos de mayor crecimiento demográfico. Este patrón de urbanización se ha mantenido hasta principios del siglo XXI [1].

El incremento del número de habitantes en las ciudades medias en el periodo de las últimas tres décadas (de 1980 a 2010) ha sido de un 144 % en su conjunto. Es el grupo de ciudades que muestran mayor crecimiento poblacional y lo componen veintidós urbes que están distribuidas en toda la extensión de la república mexicana [21].

La ciudad de Culiacán es una de estas veintidós ciudades medias [21], donde su población actual según INEGI [34] es de 905,265 hasta el 2015. Este rápido crecimiento, además de las demandas de movilidad de los ciudadanos y el parque vehicular, han ocasionado un incrementado considerable en el tráfico vehicular de la ciudad de Culiacán.

La Tabla 1.1, muestra los resultados del estudio realizado por el Plan Parcial de Movilidad para el Desarrollo Urbano de la Ciudad de Culiacán [31]. Donde se observa que la necesidad de tener un vehículo en Culiacán ha aumentado a la largo de los años. El número de habitantes respecto al número de vehículos en 2015 es casi 2 habitantes por cada vehículo, lo cual refleja que mientras Culiacán sea considerada una ciudad en crecimiento, al aumentar el número de habitantes el tráfico vial será mayor que en la actualidad.

Hasta el 2015 se reportan 470,016 vehículos registrados en la ciudad [34], lo cual está generando congestión vial en varios puntos de la ciudad. El presente trabajo de investigación pretende proponer un modelo de optimización de rutas de vehículos de manera colectiva tal que minimice el tiempo total de los trayectos, y así disminuir la congestión vial y los problemas que ella causa.

Año	Habitantes	Vehículos	Habitantes por vehículo
1970	168,000	18,064	9.3
2000	542,820	126,237	4.3
2004	605,300	242,120	2.5
2015	905,265	470,016	1.926

Tabla 1.1: Crecimiento del parque vehicular respecto al número de habitantes a lo largo de los años

Efectos negativos de la congestión vehicular

La congestión vehicular se produce cuando la distribución normal del transporte genera una demanda de espacio mayor que el disponible en las vialidades [32]. Hay una serie de consecuencias tales como:

- Contaminación del aire y aumento de las emisiones de dióxido de carbono, debido al aumento de ralentización, aceleración y frenado.
- Contaminación auditiva.
- Pérdida de tiempo para los automovilistas y pasajeros.
- Incapacidad para predecir con exactitud el tiempo de viaje.
- Aumento del uso de combustibles, en teoría, también puede causar un aumento de los costes de combustible.
- Automovilistas frustrados, el fomento de la ira de carretera y la reducción de la salud de los automovilistas.
- Respecto a emergencias, si se bloquea el tráfico esto podría interferir con el paso de los vehículos de emergencia para viajar a sus destinos.

En la Figura 1.1, se muestra que el 20 % de población de Culiacán se moviliza en vehículos particulares o servicios de taxi. Solamente estos dos tipos de transporte representan el 70 % de la actividad vehicular en las calles. El 30 % restante se atribuye al transporte de carga, autobuses u otros. Entonces, si la población está en constante crecimiento y con los años las personas adquieren más automóviles, el problema

del congestionamiento vial será más grave y tendrá mayores repercusiones en el medio ambiente, ya que actualmente se estima que el 80 % del total de emisiones contaminantes en la ciudad de Culiacán, son producidas por la congestión vehicular [33].

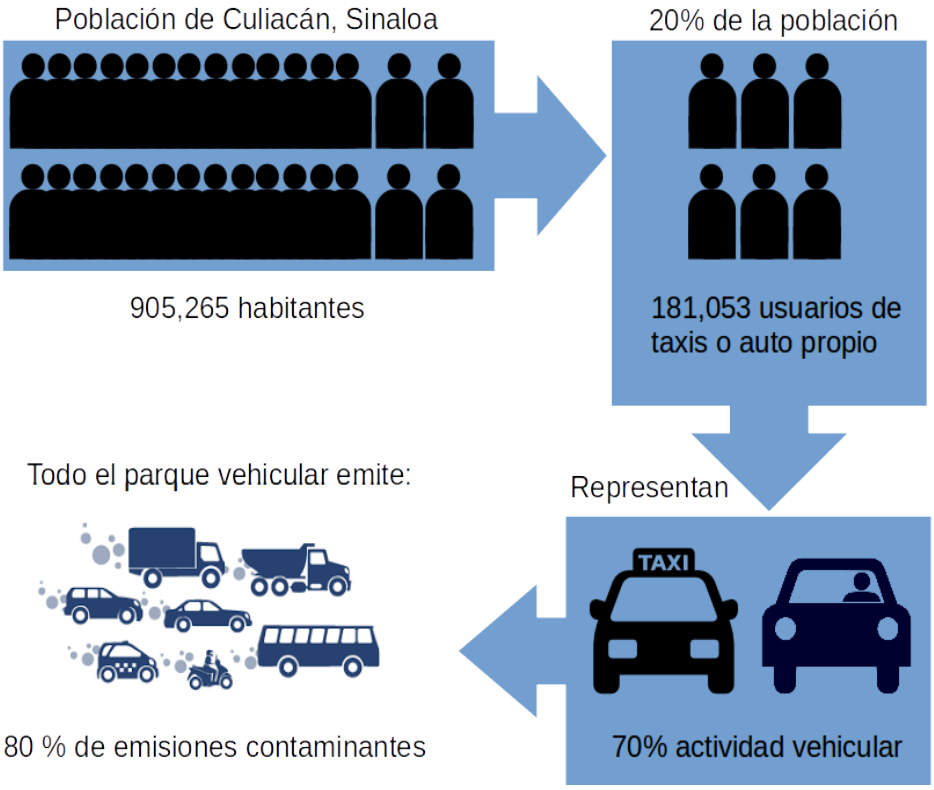


Figura 1.1: Distribución de la población en el parque vehicular.

La Organización Mundial de la Salud, realizó un estudio donde pone en claro que la mala calidad del medio ambiente causa 12.6 millones de muertes al año, es decir el 23 % de los fallecimientos mundiales [17].

Descripción formal del problema

En la actualidad, existen diversos modelos y aplicaciones que proporcionan rutas vehiculares, tales como Google Maps, Waze, UPS, sin embargo estos modelos carecen

de un análisis de rutas colectivas, generando rutas individuales sin considerar que pueden generar tráfico vehicular.

Se cree que existe una oportunidad de disminuir los problemas que ocasiona el tráfico vehicular aplicando teoría de juegos para minimizar el tiempo total de los trayectos con base en un análisis de rutas colectivas, en donde cada jugador seleccionará su mejor ruta considerando las rutas de los demás jugadores, aplicando Best Response Dynamics [24].

En términos formales, nuestro problema de optimización de rutas colectivas se puede especificar de la siguiente manera:

Sea $G = (V, E)$ un grafo dirigido, donde V representa un conjunto de vértices y E uno de aristas que definen la estructura vial de una parte de la ciudad de Culiacán. Dados v viajes definidos por su vértice inicial ($o \in V$) y su vértice final ($d \in V$), encontrar un conjunto de rutas tal que se minimice la suma del tiempo que toma realizar cada uno de los viajes.

1.2. Objetivo General y Específicos

Desarrollar un esquema estratégico para optimizar de manera colectiva, la duración de un conjunto de rutas viales (viajes) utilizando Best Response Dynamics, tal que minimice la suma total de la duración de los viajes.

Objetivos Específicos

Para lograr el objetivo general se proponen los siguientes objetivos específicos:

- Desarrollar un modelo computacional para representar la estructura vial de la ciudad por medio de grafos.
- Definir una función que asigne pesos a las aristas del grafo vial, en términos de las características físicas de las calles y de la carga vehicular.

- Diseñar, implementar y caracterizar un algoritmo de optimización colectiva de rutas basado en Best Response Dynamics.
- Desarrollar un conjunto de experimentos para caracterizar el desempeño del esquema estratégico propuesto.
- Realizar un estudio comparativo del desempeño del esquema estratégico por medio de un simulador de movilidad urbana.

1.3. Hipótesis

H1. El comportamiento del tiempo de trayecto en una calle (representada por una arista en el grafo vial) puede ser estimado por una función de tipo sigmoide parametrizada por el número de automóviles y carriles, así como la velocidad máxima en la calle y su longitud.

H2. El método de Best Response Dynamic permitirá optimizar de manera colectiva el tiempo de los trayectos de un conjunto de rutas.

1.4. Organización de la tesis

A continuación se presenta la organización de este documento de tesis.

En el capítulo II son los Antecedentes y en ellos se describen los conceptos a utilizar en la metodología tales como: Teoría de juegos, Best Response Dynamics, Teoría de grafos, el problema de la ruta más corta, algoritmos para encontrar la ruta más corta, entre otros.

El capítulo III describe el estado del arte en donde se realiza una comparativa con metodologías ya existentes que dan solución al mismo problema que se aborda en la presente tesis, así también se puntualizan los trabajos relacionados desarrollados por investigadores y publicados en diferentes revistas donde hacen uso de conceptos que se utilizan en la presente tesis para resolver problemas similares.

Continuando con el capítulo IV que corresponde a la Metodología, se describe el procedimiento para la obtención de la función de congestión que representa el comportamiento del tráfico vial y se expone la integración de la estrategia de Best Response Dynamics y Dijkstra que dan origen al algoritmo de optimización colectiva.

En la sección del capítulo V se muestran los resultados obtenidos con el objetivo de evaluar el desempeño del esquema de optimización propuesto en la presente tesis. Se realizaron pruebas en diferentes escenarios, cada escenario tiene un porcentaje de congestión vehicular en la red vial. También se realiza una comparativa entre los resultados de la estrategia de optimización colectiva propuesta con la metodología utilizada por la aplicación ya existente de Google Maps (Dijkstra) solamente para la obtención y distribución de rutas viales.

Finalmente, en el capítulo VI, se presentan las conclusiones de este trabajo de investigación así como algunas posibles prolongaciones a futuro que darán continuidad al presente trabajo.

Capítulo 2

Antecedentes

2.1. Teoría de juegos

La teoría de juegos es un área de la Matemática aplicada que se construyó bajo el impulso de John Von Neumann (1903-1957) y Oskar Morgenstern (1909-1977) [26]. Se puede definir como el estudio de modelos matemáticos de conflicto y cooperación entre jugadores [14]. Su principal objetivo es analizar el comportamiento estratégico de los jugadores modelando las interacciones entre las elecciones de los mismos [26]. Los jugadores son entes racionales, no necesariamente humanos. La competencia en muchas actividades sociales ha hecho de la teoría de juegos un enfoque fundamental para modelar diversas situaciones, como en economía, ciencia política, investigación de operaciones, planeación militar, gestión de recursos y redes [25].

La teoría de juegos se ha desarrollado gradualmente y, a grandes rasgos, se pueden distinguir dos grandes áreas: la teoría de juegos no cooperativa y la teoría de juegos cooperativa.

2.2. Juegos cooperativos y no cooperativos

Los juegos cooperativos son en donde los jugadores pueden acordar el reparto de utilidades, los jugadores forman una coalición la cual debe ser estable y se deben repartir las utilidades entre los miembros para que ninguno de ellos esté interesado en romper la coalición [26].

En cambio los juegos no cooperativos son un modelo en el que los jugadores no pueden hacer acuerdos previos. Los juegos no cooperativos que reciben más atención suelen ser bipersonales, es decir, con sólo dos jugadores. Pueden ser de suma cero, cuando el aumento en las utilidades de un jugador implica una disminución del otro, o de suma no nula en caso contrario, es decir, cuando la suma de las utilidades de los jugadores puede aumentar o disminuir en función de sus decisiones. El problema del dilema del prisionero es un ejemplo de juego no cooperativo de suma no nula [26].

Componentes de un juego

Un juego en forma normal o estratégica, que se denotará por: $G = S_1, ..., S_n; u_1, ..., u_n$ ó $G[S, U]$, consta de cuatro elementos esenciales:

- Los n jugadores que participan en el juego.
- Las estrategias disponibles para cada jugador, que son $S = S_1, ..., S_n$. En donde $S_i = s_{i_1}, ..., s_{i_q}$, es el conjunto de estrategias con que cuenta el jugador i. La notación suele simplificarse designando por s_i a un elemento arbitrario de S_i, donde $s_i \in S_i$
- El espacio de estrategias $S = s_1 \times s_2 \times s_3, ..., \times s_n$ donde $(s_1, ..., s_n \in S)$ conforman un perfil estrategias.
- El conjunto de funciones de utilidad $U = u_1, ..., u_n$, donde la función de utilidad de cada jugador $u_1 = u_1(s_1, ..., s_n), s_i \in S_i$ representa la utilidad obtenida por el *i-ésimo* jugador, que está en función de las estrategias elegidas por él y sus rivales en el juego.

En un juego en forma normal los jugadores eligen sus estrategias de forma simultánea, es decir, que cada jugador elige su jugada sin conocer las decisiones de los demás. Cada jugador recibe una utilidad $x = u_i(s_1, ..., s_n)$, dependiendo de las estrategias elegidas por los demás.

Utilizando la notación anterior, el problema del dilema del prisionero sería el siguiente:

1.- Jugadores
$J = \{j_1, j_2\}$

2.- Estrategias
$S = \{S_{j_1} \times S_{j_2}\}$
$S_{j_1} = s_{1(confesar)}, s_{2(no\ confesar)}$
$S_{j_2} = s_{1(confesar)}, s_{2(no\ confesar)}$

3.- Perfiles
$P = \{\ (s_{1_{(j_1)}}, s_{1_{(j_2)}}), (s_{1_{(j_1)}}, s_{2_{(j_2)}}),$
$(s_{2_{(j_1)}}, s_{1_{(j_2)}}), (s_{2_{(j_1)}}, s_{2_{(j_2)}})\}$

4.- Utilidades
$U = \{U_{j_1}, U_{j_2}\}$,
Preferencias por el jugador $j1$
$U_{j_1}(s_{1_{(j_1)}}, s_{2_{(j_2)}}) = 10 > U_{j_1}(s_{2_{(j_1)}}, s_{2_{(j_2)}}) = 5 > U_{j_1}(s_{1_{(j_1)}},$
$s_{1_{(j_2)}}) = 2 > U_{j_1}(s_{2_{(j_1)}}, s_{1_{(j_2)}}) = 1$

Preferencias por el jugador $j2$
$U_{j_2}(s_{2_{(j_1)}}, s_{1_{(j_2)}}) = 10 > U_{j_2}(s_{2_{(j_1)}}, s_{2_{(j_2)}}) = 5 > U_{j_2}(s_{1_{(j_1)}},$
$s_{1_{(j_2)}}) = 2 > U_{j_2}(s_{2_{(j_1)}}, s_{2_{(j_2)}}) = 1$

2.3. Equilibrio de Nash

Durante un juego, un perfil está formado por la estrategia seleccionada por cada jugador. Un perfil de estrategia está dominado si existe un jugador que puede mejorar su utilidad al desviarse a otra de sus estrategias en un perfil diferente. Una función de utilidad se utiliza para calcular el beneficio obtenido para cada perfil de estrategia posible en el juego [15] [29].

El equilibrio de Nash [15] es un concepto matemático ampliamente utilizado en la teoría de juegos, especialmente en juegos no cooperativos. Un equilibrio de Nash es una situación estable en la cual todos los jugadores han puesto en práctica una estrategia que maximiza sus ganancias dadas las estrategias de los otros. Consecuentemente, ningún jugador tiene ningún incentivo para modificar individualmente su estrategia. La definición matemática se da a continuación.

Las estrategias del equilibrio de Nash son denotadas por $s_1*, ..., s_n*$ donde s_i* es la mejor respuesta del jugador i de las otras estrategias de los $n-1$ jugadores, $s_1*, ..., s_{i-1}*, s_{i+1}*, ..., s_n*; (s_1*, ..., s_i*, ..., s_n*)$ es la n-tupla de estrategias para la ecuación máxima de la función de utilidad 2.1

$$u_i(s_1*, ..., s_{i-1}*, s_i*, ..., s_{i+1}*, ..., s_n*) \geq u_i(s_1*, ..., s_{i-1}*, s_i, s_{i+1}*, ..., s_n*) \forall i \in P, s_i \in S_i \tag{2.1}$$

Cada perfil de estrategia en cada función de utilidad es valorada y comparada con todas las demás para determinar si está dominada o no. Los perfiles dominados se descartan y los perfiles no dominados se ajustan al Equilibrio de Nash. Cualquier juego en forma normal tiene al menos un perfil de estrategia que se ajusta al Equilibrio de Nash [15] [29].

2.4. Best Response Dynamics

El concepto de Best Response Dynamics es fundamental para la contribución más conocida de John Nash, el equilibrio de Nash, el punto en el que cada jugador en un juego ha seleccionado su mejor estrategia (o una de las mejores) a partir de las estrategias de otros jugadores [16].

En un juego, las decisiones no sólo dependerá de la estructura del juego, sino que a menudo también de las decisiones que hacen los demás jugadores [26]

El ejemplo siguiente Figura 2.1 , se muestra el comportamiento de la estrategia Best Response Dynamics:
Se consideran un par jugadores y cada uno tiene dos opciones para construir una ruta: la ruta media a través de v, y la ruta externa usando una única arista. Suponiendo que cada jugador comienza con una ruta inicial, pero que continuamente evalúa la situación actual para decidir si es posible cambiar a una ruta mejor [11].

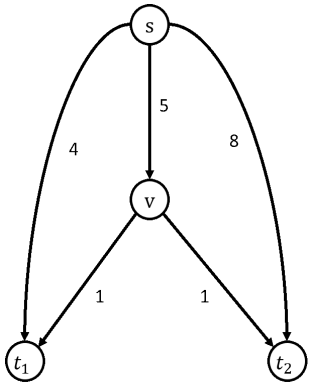


Figura 2.1: Ejemplo: Aplicando Best Response Dynamics a grafo dirigido ponderado

Si se representa el problema a un juego en forma normal, daría como resultado lo siguiente:

1.- Jugadores
$J = \{J_1, J_2\}$

2.- Estrategias

$S = \{s_1 = s, t_1 \; s_2 = s, v, t_1 \; s_3 = s, t_2 \; s_4 = s, v, t_2\}$

3.- Perfiles

$S = \{ \; (J_{1_{s_1}} = s, t_1 \times J_{2_{s_3}} = s, t_2),$
$(J_{1_{s_1}} = s, t_1 \times J_{2_{s_4}} = s, v, t_2),$
$(J_{1_{s_2}} = s, v, t_1 \times J_{2_{s_3}} = s, t_2),$
$(J_{1_{s_2}} = s, v, t_1 \times J_{2_{s_4}} = s, v, t_2)\}$

4.- Utilidades

$U = \{ \; u_1 = \{(J_1 = 4, \; J_2 = 8), \; (S = J_{1_{s_1}} \text{ y } J_{2_{s_3}})\},$
$u_2 = \{(J_1 = 4, \; J_2 = 6), \; (S = J_{1_{s_1}} \text{ y } J_{2_{s_4}})\},$
$u_3 = \{(J_1 = 6, \; J_2 = 8), \; (S = J_{1_{s_2}} \text{ y } J_{2_{s_3}})\},$
$u_4 = \{(J_1 = 6, \; J_2 = 7), \; (S = J_{1_{s_2}} \text{ y } J_{2_{s_4}})\}$

En el ejemplo Figura 2.1, suponiendo que los dos jugadores comienzan utilizando sus trayectorias externas. Entonces t_1 no ve ninguna ventaja en cambiar la ruta actual $(s - t_1)$ con costo de 4 a otra $(s - v - t_1)$ de costo 6, pero t_2 sí, ya que su ruta actual $(s - t_2)$ cuesta 8 y su alternativa $(s - v - t_2)$ 6 , por lo que t_2 actualiza su ruta moviéndose al centro.

Pero que sucede si ambos jugadores deciden tomar el camino del medio, es decir el J_1 toma la ruta $s_2 = s, v, t_1$ y el jugador J_2 toma la ruta $s_4 = s, v, t_2$, al pasar J_1 por su ruta el valor de cada una de las aristas aumenta en uno, por lo que a J_2 podría afectarle en su utilidad al coincidir ambas rutas en una arista de s a v. Entonces a J_2 no le convendría tomar esa ruta y optaría por cambiarla a otra donde la utilidad que reciba sea menor, en caso de que existiera. Este proceso de cambios de rutas respecto a las utilidades que reciben los jugadores es de lo que trata la estrategia de Best Response Dynamics. De acuerdo a la utilidad que recibe cada jugador en cada ronda,se decide por cambiar o no de rutas un número determinado de rondas o hasta encontrar un equilibrio de Nash.

En el problema de enrutamiento anterior Figura 2.1, cada uno de los jugadores tiene un objetivo individual pero interactúan para producir una solución colectiva. Y sucede de la siguiente manera:

Cada jugador está continuamente preparado para mejorar su utilidad en respuesta a los cambios realizados por el otro jugador, es decir actualiza su ruta en función de su mejor respuesta a la situación actual.

Best Response Dynamic es la estrategia que produce el resultado más favorable para un jugador, tomando en cuenta las estrategias de otros jugadores. Cuando cada jugador selecciona su mejor estrategia respecto a la de los demás, se dice que existe un equilibrio de Nash [7].

2.5. Teoría de grafos

La Teoría de grafos fue desarrollada por el matemático Leonhard Euler en el Siglo XVIII como una rama de la Topología Algebraica, e introducida en la Geografía en los años sesenta por W.L. Garrison y F.D. Marble, generalmente para estudiar la expresión de las redes de transporte sobre el espacio geográfico [28]. Su objetivo de estudio son las propiedades de los grafos [8].

El grafo es un término matemático utilizado para designar a un conjunto de puntos unidos entre sí por segmentos, que pueden representar un proceso o relación funcional de cualquier tipo, pero centra su atención en las relaciones topológicas entre sus elementos. Un grafo $G = (V, A)$ está formado por un par de conjuntos finitos (V, A). Sus elementos son V es el conjunto de vértices y A el conjunto de aristas. Un grafo es no dirigido cuando sus aristas sólo unen sus vértices y cuando las aristas tienen una dirección es un grafo dirigido [28].

En la Figura 2.2, se muestran términos asociadas a un grafo y a continuación su significado.

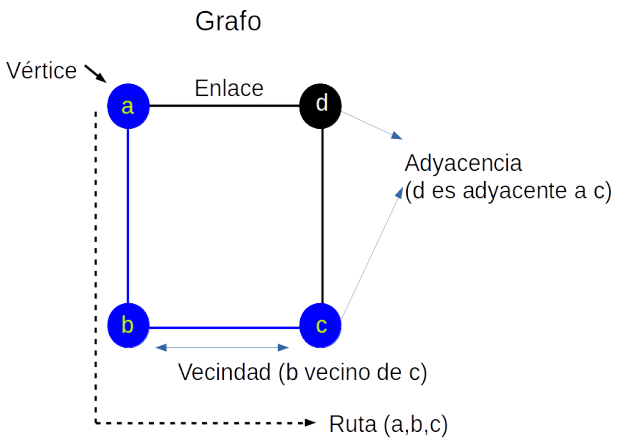


Figura 2.2: Grafo

Terminología de grafos.

- *Vértice*: nodo.
- *Arista*: conexión entre dos vértices.
- *Adyacencia*: Se dice que dos vértices son adyacentes si entre ellos hay una arista directa.
- *Vecindad*: Conjunto de vértices adyacentes a otros.
- *Ruta*: Conjunto de vértices que hay que recorrer para llegar desde un nodo origen hasta un nodo destino.
- *Grafo conectado*: Aquel que tiene camino directo entre todos los nodos.
- *Grafo dirigido*: Aquél cuyas aristas son unidireccionales e indican hacia donde están dirigidos.
- *Grafo con pesos*: Aquel cuyas aristas tienen asociado un valor.

Muchos problemas se presentan naturalmente con una formulación teórica de grafos. Por ejemplo, cuando se trata de problemas de tráfico, es necesario conocer que calles de la red son de un sólo sentido y que dirección tienen, para este caso suele utilizarse un grafo dirigido [2].

Formalmente un grafo dirigido D es una tripleta $(V(D), A(D), \psi D)$ [2] que consiste en un conjunto no vació $V(D)$ de vértices, un conjunto $A(D)$ de aristas, y una función de incidencia ψD que asocia cada arista de D a un par ordenado de vértices (no necesariamente distintos) de D [2].

Los grafos dirigidos tienen una representación gráfica simple y son representados por un diagrama subyacente junto con flechas en sus aristas, cada flecha apuntando hacia el vértice correspondiente, mientras que en los grafos no dirigidos, sólo existe una linea que une cada vértice. Ejemplos de ellos se muestran a continuación Figura 2.3 :

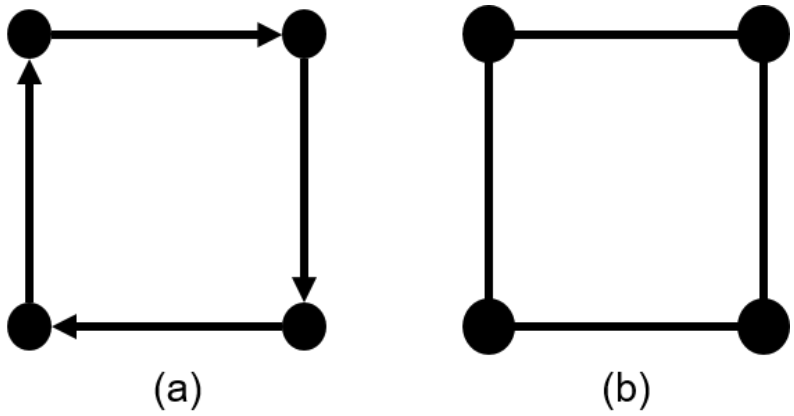


Figura 2.3: (a) Grafo dirigido (b) Grafo no dirigido

2.5.1. Representaciones de grafos

Existen diferentes maneras de almacenar un grafo en una computadora, por ejemplo mediante una una matriz de adyacencia o listas enlazadas, pero, es importante entender cómo cada representación puede afectar o mejorar su análisis con algún algoritmo de búsqueda (Sección 2.6.1). Las diferencias entre cada representación se

basan en la forma en que se almacena la información, por ejemplo, una matriz de adyacencia nos permite almacenar un grafo dirigido ponderado donde sus pesos se asigna a la unión del punto x y y en la matriz, si se requiere conocer el valor de la coordenada x y y solo se accede a ella y obtenemos el valor.

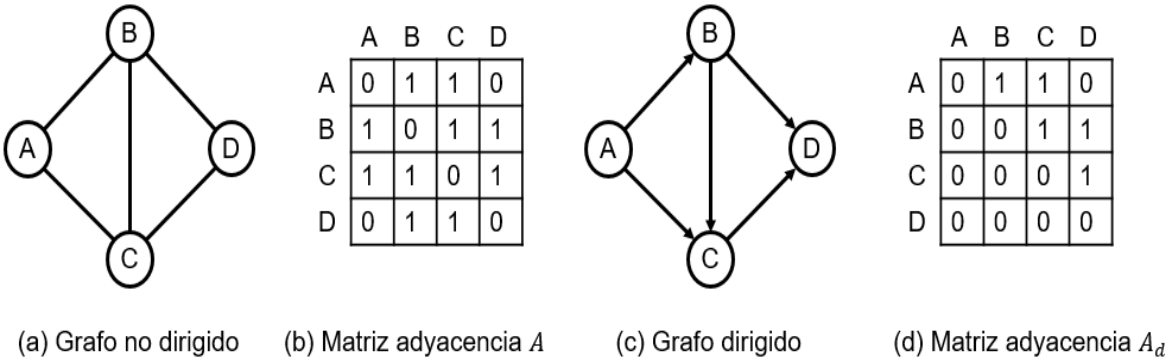
Matriz de adyacencia nodo - nodo

Esta representación consiste en una matriz $n \times n$, de modo que cada una de sus filas y columnas corresponde a un vértice y las intersecciones de fila - columna corresponden a un arista. La entrada (i, j) de la matriz de adyacencia nodo - nodo es : 1, si hay una arista desde el nodo i al nodo j-ésimo y 0 en caso contrario. En caso de grafos no dirigidos, es fácil ver que la adyacencia de nodo - nodo da origen a una matriz simétrica.

Por ejemplo una matriz de adyacencia nodo - nodo A de un grafo G es una matriz cuyas filas y columnas se definen en la Ec.2.2:

$$A_{ij} = \begin{cases} 1 & \textit{if nodo } i \textit{ y nodo } j \textit{ son adyacentes} \\ 0 & \textit{otra cosa} \end{cases} \tag{2.2}$$

En la Figura 2.4 (a) muestra un ejemplo de un grafo no dirigido y su representación en una matriz en Figura 2.4 (b) y un grafo dirigido en Figura 2.4 (c) y su representación en Figura 2.4 (d).



(a) Grafo no dirigido (b) Matriz adyacencia A (c) Grafo dirigido (d) Matriz adyacencia A_d

Figura 2.4: Matriz de adyacencia Nodo a Nodo.

2.6. El problema de la ruta más corta

El problema de la ruta más corta se define mediante un grafo dirigido y ponderado, donde esta ponderación suele ser llamada: peso de la arista, el cual puede ser dado por una función. El objetivo es encontrar una ruta desde un nodo origen, al nodo destino, tal que minimice la suma de los pesos a lo largo de la ruta [9].

Este problema a menudo se resuelve usando una variedad de algoritmos, dependiendo de la estructura del grafo.

2.6.1. Algoritmos para encontrar la ruta más corta

Existen una diversidad de algoritmos que encuentran la ruta más corta en un grafo. Sin embargo existen factores determinantes que hacen escoger unos de otros. El problema de la presente tesis se basa en un grafo dirigido que se pondera con el valor de una función de congestión parametrizada con las características de cada una de las calles seleccionadas de la ciudad de Culiacán. En tal grafo se debe encontrar la ruta mas corta de cada uno de los usuarios que quieran realizar un viaje.

A continuación se describen dos algoritmos que dan solución al problema de la ruta más corta.

Algoritmo Dijkstra

El algoritmo de Dijkstra fue formulado en 1959 por su creador E. W. Dijkstra [6]. Es un algoritmo voraz muy simple para resolver el problema de la ruta más corta. Tiene una complejidad de $O(n^2)$ [11].

El algoritmo de Dijkstra resuelve el problema de las rutas más cortas en un grafo dirigido y ponderado $G = (V, E)$ con pesos no negativos. Por lo tanto se supone que el peso de una arista es igual o mayor a cero. [3].

Teniendo un grafo dirigido ponderado de N nodos, sea x el nodo inicial. Un vector D de tamaño N guardará al final del algoritmo las distancias desde x hasta el resto de los nodos. El procedimiento seria el siguiente:

1.- Inicializar todas las distancias en D con un valor infinito relativo, ya que son desconocidas en un inicio, exceptuando la de x que se debe colocar 0, debido a que la distancia de x a x sería 0.

2.- Sea $a = x$ (se toma a como nodo actual)

3.- Se recorren todos los nodos adyacentes de a, excepto los nodos marcados. Se les llamará nodos no marcados v_i.

4.- Para el nodo actual, se calcula la distancia tentativa desde dicho nodo hasta sus vecinos. Es decir, la distancia tentativa del nodo $'v_i'$ es la distancia que actualmente tiene el nodo en el vector D más la distancia desde dicho nodo $'a'$ (el actual) hasta el nodo v_i. Si la distancia es menor que la distancia almacenada en el vector, se actualiza entonces el vector con esa distancia tentativa.

5.- Se marca como nodo marcado a a.

6.- Se toma como próximo nodo actual el de menor valor en D y se regresa al paso 3, mientras existan nodos no marcados.

En la Tabla 2.1 se muestra el algoritmo de Dijkstra, y lleva a cabo el siguiete procedimiento:

- Línea 1 inicializa la d y π valores de la manera habitual
- Línea 2 inicializa el conjunto S al conjunto vacío. El algoritmo mantiene que $Q = V - S$ al comienzo de cada iteración del ciclo *while* líneas 4-8.
- Línea 3 inicializa la cola de prioridad mínima Q para contener todos los vértices en V: desde $S = \emptyset$ en ese momento. Cada vez que pasa el ciclo *while* de las líneas 4-8,
- Línea 5 extrae un vértice u desde $Q = V - S$
- Línea 6 lo agrega al conjunto S. El vértice u, por lo tanto, tiene la menor estimación de camino más corto de cualquier vértice en $V - S$.
- Las líneas 7-8 relajan cada borde (u, v) dejando u, actualizando así la estimación $v.d$ y el predecesor $v.\pi$ si podemos mejorar el camino más corto hacia v encontrado hasta ahora a través de u.

Observe que el algoritmo nunca inserta vértices en Q después de la línea 3 y que cada vértice se extrae de Q y añadido a S exactamente una vez, de modo que el ciclo *while* de las líneas 4-8 itera exactamente $|V|$ veces [3].

Dijkstra (G, w, s)
1:INICIALIZAR - INDIVIDUAL - FUENTE (G, S)
2: $S = \emptyset$
3: $Q = G.V$
4: Mientras $Q \neq \emptyset$
5: u= EXTRAER- MIN (Q)
6: $S = S \cup u$
7: Para cada vértice $u \in G.Adj[u]$
8: Relax (u, v, w)

Tabla 2.1: Algoritmo de Dijkstra

Algoritmo Floyd-Warshall

El algoritmo de Floyd - Warshall, descrito por Bernard Roy en 1959, es un algoritmo de análisis sobre grafos para encontrar el camino mínimo en grafos dirigidos ponderados con pesos de aristas positivas y negativas [4].

El algoritmo mantiene una matriz (d_{ij}) de modo que en la iteración k, d_{ij} es el camino más corto desde i a j vértices usando $1, 2, ...k$ como vértices intermedios. Después de que el algoritmo termina, asumiendo que no hay un ciclo de costo negativo, la ruta más corta desde los nodos i hasta j es d_{ij}.

La operación principal en el algoritmo es: $j : d_{ik} = min(d_{ik}, d_{ij} + d_{jk})$. Esta operación a veces se denomina operación triangular Figura 2.5.

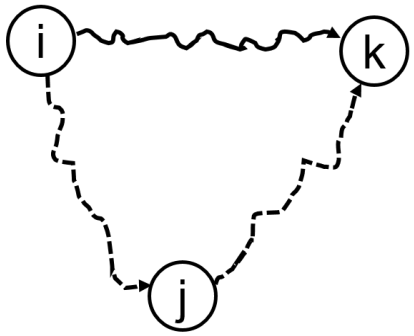


Figura 2.5: Operación triangular: Es la ruta (punteada) usando el vértice j como vértice intermedio más corto que la ruta (solida) sin usar el nodo j

La complejidad del algoritmo de Floyd-Warshall (Tabla 2.2) es $O(n_3)$ [10].

Entrada: Una n x n matriz $[c_{ij}]$
Salida: Una n x n matriz $[d_{ij}]$ es la distancia más corta
1:Desde i a j debajo $[c_{i,j}]$
2:Una n x n matriz $[c_{ij}]$ es el nodo en la ruta desde i a j
3:Iniciar
4:For todo $i \neq j$
5: $d_{ij} := c_{ij}$;
6:For $j = 1, ..., n$
7: $d_{ij} := \infty$;
8:For $j = 1, ..., n$
9: For $i = 1, ..., n, i \neq j$,
10: For $k = 1, ..., n, k \neq j$
11: $d_{ik} := min\{d_{ik}, d_{ij} + d_{ij}\}$
12: $e_{ik} =$
13: j si $d_{ik} > d_{ik} + d_{jk}$
14: e_{ik} $otra\ cosa$

Tabla 2.2: Algoritmo Floyd-Warshall

Capítulo 3

Estado del arte

3.1. Google Maps

Google Maps es la aplicación de navegación más popular utilizada entre la población. Éste es un servidor de aplicaciones gratuitas de mapas en la web que pertenece a Alpthabet Inc. Ofrece imágenes de mapas desplázables, así como fotografías por satélite del mundo. Los usuarios acceden a su aplicación para obtener las rutas a lugares donde quieren ir. Es rápido y eficiente con las direcciones. El algoritmo básico que utiliza es el algoritmo de Dijkstra (Sección 2.6.1) [12].

Las búsquedas con Google Maps son de punto a punto, el usuario solicita ir a un lugar y este proporciona la mejor ruta hasta el momento, estas rutas son calculadas por cada individuo, no existe información en donde se diga que la ruta que proporciono considero al resto de usuarios que utilizan la aplicación [12], a diferencia de nuestra propuesta que si se está considerando los viajes que realizarán el conjunto de usuarios para proporcionarles una ruta.

3.2. United Parcel Service (UPS)

Otra empresa que utiliza un sistema para generar rutas es UPS (United Parcel Service, Inc.), fue fundada en 1907 como una empresa de mensajería de los Estados Unidos [22]. Actualmente UPS es una empresa multimillonaria internacional que tiene como meta hacer posible el comercio en todo el mundo. Maneja el flujo de mercancías, fondos e información en más de 200 países y territorios de todo el mundo.

UPS distribuye paquetes en diferentes camiones los cuales recorren rutas que incluyen varios destinos, utiliza un algoritmo llamado Orion, el cual es una heurística. La heurística realiza una búsqueda de la mejor respuesta que uno pueda encontrar, los resultados que genera continuamente se van refinando con el tiempo. Además, Orion consta de muchos componentes, incluido un algoritmo para el problema del agente viajero, con el cual se calcula la ruta más eficiente entre una variedad de puntos y mapeo geográfico. Lo que hace único a Orion es la forma en que reúne estos elementos, buscando un equilibrio entre un resultado óptimo y la consistencia, según menciono el Sr. Levis en una entrevista publicada en una revista estadounidense "The Wall Street Journal" en 2015 [22].

3.3. WAZE

Waze es una aplicación social de tránsito en tiempo real y navegación asistida por GPS desarrollada por Waze Mobile en Raánana, Israel en 2008 [30]. Al utilizar la aplicación de Waze, los servidores Waze usan algoritmos de enrutamiento para determinar la mejor ruta en ese momento en particular. La operación del servidor de enrutamiento se considera privada y una ventaja competitiva para Waze [36].

Cuando se solicita un cálculo de ruta, la solicitud se envía al servidor de Waze. Esa ruta luego se transmite de vuelta a su dispositivo cliente y se muestra por medio de una aplicación. Las solicitudes de enrutamiento varían según la configuración que se haya elegido en el dispositivo del cliente. En navegación, se puede elegir:

- La ruta más rápida o más corta.

- Permitir o evitar las carreteras con peaje.
- Permitir caminos de tierra, nunca permitir caminos de tierra, o evitar caminos de tierra largos.
- Evitar las principales autopistas.

La ruta más corta se refiere a la distancia física. La mayoría de las personas generalmente prefieren la opción de ruta más rápida; 100 km de autopista son mejores que 90 km de caminos rurales.

Se sabe que Waze utiliza informes en tiempo real de las velocidades actuales de la carretera por preferencia sobre las velocidades medias históricas de la carretera. Waze también usa informes de congestión de tráfico para redirigir el tráfico lento. Se supone que la velocidad de cualquier Wazer (usuario) en un tramo de carretera será la velocidad utilizada cuando se calcule la ruta de un Wazer posterior.

A medida que crece el número y la densidad de Wazers, estos datos en tiempo real adquieren una mayor importancia. Este énfasis es en parte un reflejo del objetivo original de Waze de crear viajes diarios óptimos. A medida que Waze se ha ido convirtiendo en un dispositivo de navegación GPS de uso más general en carreteras menos transitadas, la velocidad promedio histórica de las carreteras se vuelve más importante.

Waze conoce la velocidad promedio de cada carretera confirmada por el usuario y su destino. EL servidor Waze pude calcular qué lista de carreteras a tomar para minimizar el tiempo total de viaje.

Las siguientes son razones por las cuales el motor de enrutamiento puede no usar la velocidad promedio de la carretera.

1.- Variaciones de la hora del día:
Considere un camino que la mayoría de los Wazers manejan a las 5 p.m. cuando la velocidad promedio es de 12 mph (millas por hora). Se elige la ruta a las 10 a. m. Cuando el camino está despejado y la velocidad promedio es de 60 mph. Es el mejor camino a seguir, pero Waze elige una ruta diferente porque no tiene información

sobre la velocidad a las 10am y asume que la velocidad es de 12 mph todo el día.

Puede, por supuesto, conducir la carretera, y Waze finalmente aprende la velocidad para ese momento. No se sabe por cuánto tiempo Waze tiene los datos más antiguos al calcular la velocidad promedio. Sin embargo, si Waze descarta información antigua, es posible que no acumule tiempo nuevo más rápido de lo que se descarta. También es posible que sus tiempos sean ignorados por ser anormales.

2.- Semáforos y señales de pare.
Waze no registra la ubicación de los semáforos. Si bien algunos navegadores GPS ofrecen indicaciones como "gire a la derecha en el siguiente semáforo", la información suele ser incompleta, incorrecta o desactualizada. La opinión de consenso es que Waze no debe registrar la ubicación de los semáforos.

Waze sí toma semáforos y detiene las señales al notar el efecto que tienen en la velocidad del tráfico. Considere un semáforo con largos tiempos de espera. El segmento de carretera que conduce a ese semáforo tendrá una velocidad promedio baja. Si la velocidad promedio (basada en el tiempo de espera promedio) se vuelve lo suficientemente baja, una ruta más larga que evita la luz se convierte en la ruta preferida. Esto se ha observado en la práctica y es un ejemplo de comportamiento emergente. Waze no está programado para evitar los semáforos, pero evita las carreteras lentas; si el semáforo hace que la carretera sea lenta, Waze los evita.

Algunos conductores suelen tomar rutas más largas, incluso sinuosas por calles laterales, para evitar paradas o semáforos. Se sabe que Waze sugiere esto, y también se sabe que vuelve a esperar en las luces cuando se recoge mejor información de velocidad promedio de las calles laterales.

Pero se tiene que tener en cuenta que esto puede ser menos óptimo debido a los retrasos de giro discutidos anteriormente [36].

Las rutas que proporcionan Waze consideran la información almacenada en un historial que alimentan los usuarios, de esta forma existe una colectividad, pero solo al recabar la información, más no esta generando una ruta a cada usuarios de forma colectiva. Las rutas siguen siendo individuales. Lo importante a destacar es la forma

en como WAZE está recabando información en tiempo real, que sus propios usuarios suben a su plataforma y de esa manera alimenta de información sus algoritmos de optimización.

3.4. Trabajos relacionados

1.- En el articulo "Modelo origen destino para estimar el flujo de tráfico usando algoritmos genéticos", se ha desarrollado un nuevo método basado en Inteligencia Artificial para resolver un problema del matriz origen-destino (O-D) aplicado al caso de una red de tráfico vehicular. El método implementado, basado en algoritmos genéticos (AG), resuelve el problema de minimización asociado al problema de matriz O-D. Para validar la técnica, se ha utilizado una red vial con alta congestión vehicular.[13]

2.- EL articulo "Sistema multi-tipos de colonias de hormigas para la solución del problema de múltiples viajeros" de Yasel José Costa Salas, René Abreu Ledón, Norge Isaías Coello Machado y Ann Nowé propone una metodología para resolver el problema de los múltiples viajeros, como un problema combinatorio discreto y consiste en encontrar una ruta para cada uno de los m viajeros, los cuales comienzan y retornan a un nodo inicial.

   La formulación del problema de los múltiples viajeros, se ajusta a los problemas de la vida real, además puede ser extendida a una amplia variedad de los problemas de enrutamiento de vehículos, mediante la incorporación de algunas restricciones, tales como: limitar la capacidad del vehículo y establecer demanda cuantificable en cada cliente o ciudad.

   En este sentido, la presente investigación propuso un nuevo algoritmo inspirado en el comportamiento de las hormigas para la solución del problema de múltiples viajeros, específicamente un Sistema Multi-tipos de Colonias de Hormigas (M-ACS), donde cada colonia representa una posible solución global del problema. Las colonias cooperan mediante intercambios de feromona en busca de una solución eficiente para el problema de los múltiples viajeros.

El M-ACS presenta rasgos significativos de inteligencia colectiva. En el, un conjunto de colonias cooperan con la finalidad de encontrar mejores soluciones. El proceso de cooperación, consiste en los "frecuentes" intercambios del rastro de feromona alzado por las hormigas que pertenecen a cada colonia. Cada colonia tiene asociado dos matrices de rastros de feromona: la primera de éstas contiene los rastros de feromona de las hormigas que pertenecen a la propia colonia y la segunda matriz contiene los rastros de feromona alcanzado por las hormigas que pertenecen a las colonias restantes. Los "frecuentes" intercambios de feromona son realizados después de una cantidad de iteraciones predefinidas [5].

De esta forma obtienen rutas colaborativas para atacar el problema de multiples viajeros.

3.- En el articulo "Diseño y optimización de redes de distribución de agua utilizando algoritmos genéticos" de Pereyra, G. Pandolfi, D. y Villagra, A., desarrollaron un diseño de una red de distribución de agua utilizando los siguientes Algoritmos: Crossover elitism population, Half uniform crossover combination, Cataclysm mutation (CHC) y el Algoritmo Genético canónico, aplicados a redes clásicas de distribución de aguas como la red de Alperovits y Shamir y la red de Nueva York [18].

4.- La tesis "Optimización multiobjetivo para la planificación de transporte público aplicando técnicas metaheurísticas", desarrollada por David Peña Morales realiza una metodología para disminuir el costo de operación de la red de transporte público ya que una asignación adecuada de la flota para cierta demanda de pasajeros reduce la circulación de vehículos semivacíos haciendo más rentable la operación, asimismo favorece con la descongestión de las vías y el impacto ambiental, al disminuir el consumo innecesario de combustible. Para ello realizo una comparación entre los resultados obtenidos y aquellos generados por el algoritmo evolutivo multiobjetivo NSGA-II [19].

Capítulo 4

Metodología

4.1. Modelado formal

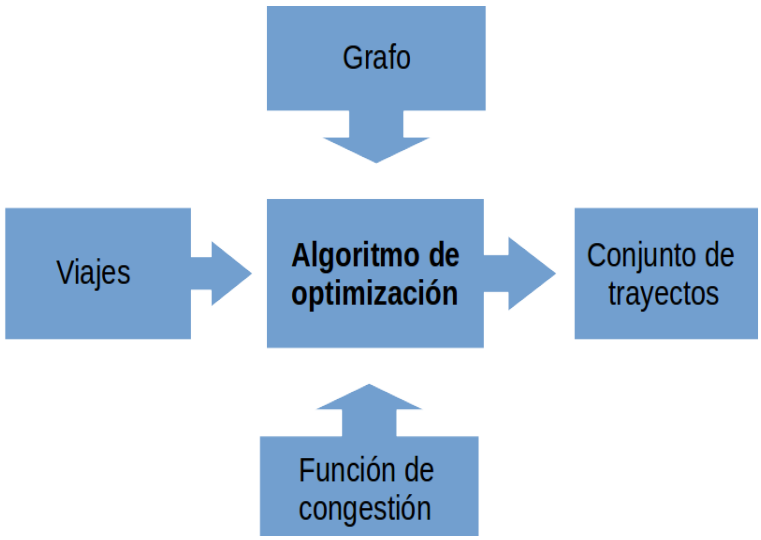


Figura 4.1: Descripción gráfica de la metodología propuesta.

El esquema estratégico propuesto en ésta tesis se bosqueja en la Figura 4.1. El esquema recibe como datos de entrada:

1.- Un grafo vial dirigido el cual representa una parte de la red vial de vehículos, en nuestro caso de estudio de la Ciudad de Culiacán.

2.- Un conjunto de viajes definidos en el grafo vial como parejas ordenadas de vértices que representan el conjunto de origen y destino de los viajes que realizará cada usuario.

3.- Y una función de congestión, la cual pondera las aristas (calles de la red vial) con base en las características físicas de las calles que ellas representan.

Como resultado se espera obtener un conjunto de viajes tal que la suma del tiempo que tardaría cada viaje sea menor que los viajes que se obtengan de forma individual.

El conjunto de viajes obtenidos se visualizarán en el simulador de movilidad urbana llamado SUMO, el cual permite observar el comportamiento de los usuarios en un ambiente de movilidad urbana.

4.1.1. Grafo vial

La representación de una sección de la red vial de vehículos de la ciudad de Culiacán, será por medio de un grafo dirigido, en donde el conjunto de aristas representan las calles y el conjunto de vértices las intersecciones de ellas, el sentido de las aristas está dado por el sentido de la circulación de las calles.

4.1.2. Viajes

Un viaje es una pareja de nodos en donde cada usuario especifica su origen y destino, es decir, determina el vértice origen y el vértice destino en el grafo vial. El conjunto de los n viajes V se describe en la Ec.4.1.2.

$$V = v_1, v_2, ..., v_N \tag{4.1}$$

Donde:
N= es el número total de viajes, uno para cada usuario
v_i= (o_i, d_i) donde $o_i, d_i \in V$ son los nodos origen y destino del i-ésimo viaje.

4.1.3. Función Objetivo

La función objetivo minimiza la suma total de la duración de los viajes, Ver Ec. 4.2.

$$min \sum_{i=1}^{N} \sum_{j=1}^{l_i} c(e_{ij}, L_{ij}) \tag{4.2}$$

Donde:
N= número de viajes definidos por los usuarios.
l_i= número de aristas que componen el viaje i.
e_{ij}= identificador de la arista j del viaje i
L_{ij}= valor de congestión de la arista j del viaje i.

4.1.4. Función de congestión

La función de congestión caracteriza el comportamiento del tráfico vehicular para definir la congestión que existe en un arista. Esta función es representada utilizando una función sigmoide ver Ec. 4.3

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \tag{4.3}$$

Donde:
e= es la base del logaritmo natural (conocido como el número de Euler).
x_o= es el valor de x en el punto medio del sigmoide.

L= es el parámetro para el rango dinámico de la función (valor máximo de la curva).
k= es el parámetro para la pendiente de la función.

Los valores de x están dentro del dominio de los números reales desde $-\infty$ a ∞, En la Figura 4.2, se observa que tiene un comportamiento asintótico creciente hacia L cuando x se acerca a ∞; y se acerca a cero, a medida que x se aproxima a $-\infty$.

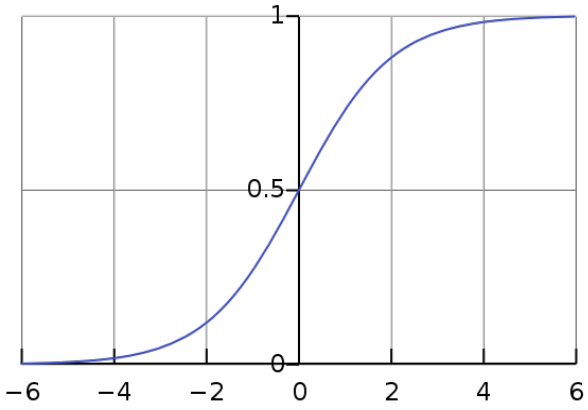


Figura 4.2: Comportamiento de la función sigmoide.

La función de congestión que se propone para modelar el tráfico vehicular se describe en la Ec.4.4

$$f(x) = \frac{k_1}{1 + e^{-k_2(x-k_3)}} + k_4 \tag{4.4}$$

Donde:
k_1: determina el tiempo máximo que tardaría un vehículo en pasar por n arista.
k_2: determina la pendiente de la curva.
k_3: el valor x en el punto medio del sigmoide.
k_4: determina el parámetro para la simetría (o sesgo) de la función, i.e., el tiempo mínimo que tardaría un vehículo en pasar por la arista x.

Gráficamente la función de congestión se comporta como se muestra en la Figura 4.3 en el caso de que las variables de entrada fueran las siguientes:
Longitud de la arista x= 100 m
Velocidad máxima en la arista x= 11.11 m/s
Carriles de la arista x= 1
Carros que pretenden transitar por la arista x = 21
Carga máxima de la arista x = 20
Tiempo mínimo de la arista x = 9 s
Tiempo máximo de la arista x = 100 s

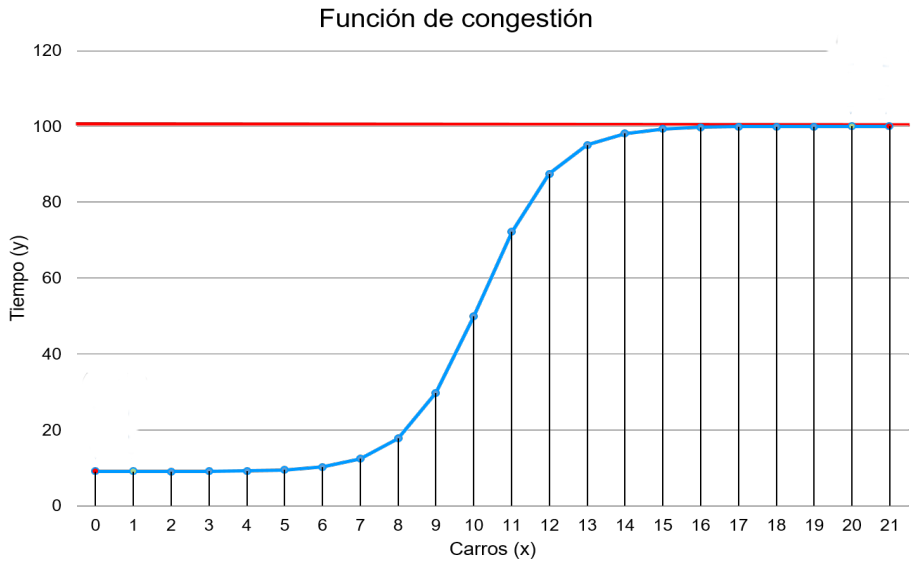


Figura 4.3: Comportamiento de la función de congestión propuesta.

El comportamiento de la función de congestión propuesta en la Ec.4.4 se describe a continuación:

1.- El tiempo que tarde en pasar un vehículo por una arista va a depender de la velocidad a la que se desplaza y la longitud de la arista, mientras más autos existan en la arista más tiempo tardará. Cuando el número de vehículos tiende a cero el tiempo tendrá a ser el mínimo, y cuando el número de vehículos tiende a

su carga máxima el tiempo tendrá a ser el máximo que puede tardar un vehículo en pasar por tal arista.

2.- Una arista puede contener de 0 a un número de vehículos definido, dependiendo de la longitud que tenga y el espacio que ocupe cada vehículo, lo cual define su carga máxima.

3.- Un vehículo se desplaza a un velocidad por la arista, mientras más autos existan en la arista la velocidad de los vehículos va a disminuir, es decir, cuando el número de vehículos se acerque a la carga máxima de la arista su velocidad tenderá a ser cero y cuando el número de vehículos tiende a cero, los vehículos se desplazan cerca o igual a la velocidad máxima permitida.

Carga vehicular máxima de una arista

Una arista representa una sección de la calle y está definido por un par de vértices. La longitud de la arista es un dato importante para determinar el número de vehículos que pudieran estar en ella. Por lo que es necesario conocer la carga vehicular máxima permitida en cada arista, ver Ec.4.5

$$Carga_{max} = \frac{l}{TamV} \times numCarriles \tag{4.5}$$

Donde:
l = longitud de la calle.
$TamV$= espacio promedio que ocuparía un automóvil en la arista.
$numCarriles$= número de carriles que tiene la arista.

Tiempo mínimo de una arista

El tiempo mínimo denotado como t_{min} determina el tiempo que se tarda en transitar por la arista x a la velocidad máxima permitida definida como v_{max}, ver Ec.4.6

$$t_{min}[x] = \frac{longitud[x]}{v_{max}[x]} \tag{4.6}$$

Tiempo máximo de una arista

El tiempo máximo denotado como t_{max} determina el tiempo que se tomaría en transitar por la arista x a la velocidad mínima permitida definida como v_{min}, ver Ec.4.7

$$t_{min}[x] = \frac{longitud[x]}{v_{min}[x]} \tag{4.7}$$

4.1.5. Análisis de parámetros

Para obtener los parámetros de la función de congestión de tal forma que se aproxime adecuadamente al comportamiento real del tráfico vial, se hará uso de algunos puntos conocidos de la función, tales como son el t_{min} y t_{max}. Estos puntos conocidos se ilustran en la Figura 4.4. Observe que debido a que la función sigmoide tienen un comportamiento asintótico en los extremos, los valores t_{min} y t_{max} se alcanzan sólo en los valores limite ($x = -\infty$ y $x = +\infty$). Esto es una limitante del uso de la función sigmoide para la aproximación.

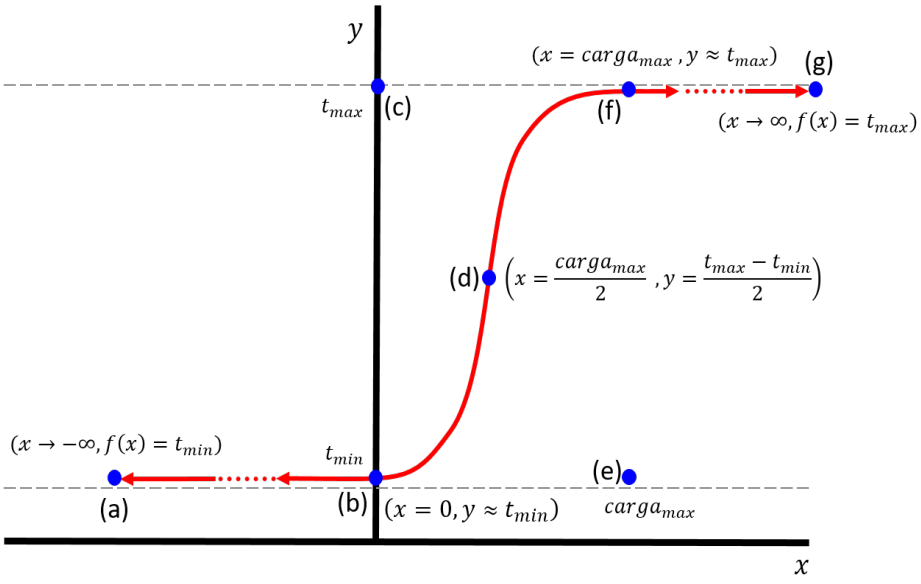


Figura 4.4: Puntos conocidos en la función de congestión.

Cuando se refiere a y se hace referencia a el eje y de la Figura 4.4 e igualmente para x, que corresponde al eje x de la misma figura. Recordando que el eje y hace referencia al tiempo y x al número de vehículos.

Cuando x vale cero, hay cero vehículos en la arista (la calle está totalmente libre) y los vehículos entrantes a la arista pueden circular libremente a su máxima velocidad. La función de congestión $f(x)$ recibe cero vehículos como entrada y su respuesta será (aproximadamente) el tiempo mínimo (inciso (b) Figura 4.4), es decir el tiempo de traslado a velocidad máxima. Donde tiempo mínimo t_{min} se definió en la Ec. 4.6.

$$y(0) = f(0) \approx t_{min} \tag{4.8}$$

Sustituyendo $x = 0$ y $y = f(0) \approx t_{min}$

$$t_{min} \approx y(x = 0) = F(x = 0) = \frac{k_1}{1 + e^{-k_2(x - k_3)}} + k_4 \tag{4.9}$$

En las ecuaciones 4.8 y 4.9 se utiliza "aproximadamente" porque en la ecuación

sigmoide el valor mínimo de y se alcanza solamente cuando $x = -\infty$, debido a su forma asintótica (inciso (a) Figura 4.4).

Evaluando la ecuación en $x = -\infty$ se obtiene:

$$t_{min} = y(x \to -\infty) = f(x \to -\infty) = \frac{k_1}{1 + e^{-k_2(x-k_3)}} + k_4 \tag{4.10}$$

$$t_{min} = \frac{k_1}{1 + e^{-(k_2*(-\infty)-k_3)}} + k_4 = \frac{k_1}{1 + e^{\infty}} + k_4 = \frac{k_1}{\infty} + k_4 = k_4 \tag{4.11}$$

Por lo tanto se determina que el valor de la constante k_4 es igual al tiempo mínimo.

El valor máximo de la función de congestión es igual al tiempo máximo que necesita un vehículo para atravesar la arista Ec.4.7. A este valor tiende la función sigmoide cuando el número de vehículos es igual a la carga máxima Ec. 4.5 (inciso (f) Figura 4.4), pero se alcanza solamente cuando el valor de x tiende a infinito (inciso (g) Figura 4.4).

$$t_{max} = y(x \to \infty) = f(x \to \infty) = \frac{k_1}{1 + e^{-k_2(x-k_3)}} + k_4 \tag{4.12}$$

Sustituyendo:

$$t_{max} = \frac{k_1}{1 + e^{-(k_2*(\infty)-k_3)}} + k_4 = \frac{k_1}{1 + e^{-\infty}} + k_4 = \frac{k_1}{1 + 0} + k_4 = k_1 + k_4 \tag{4.13}$$

Por lo tanto se determina que el valor de la constante k_1 es igual al tiempo máximo menos k_4, despejando k_1 del resultado de la Ec.4.13 y sustituyendo k_4 por lo que representa que es el t_{min} Ec.4.11 .

$$k_1 = t_{max} - k_4 = t_{max} - t_{min} \tag{4.14}$$

La función alcanza el tiempo máximo cuando $x = \infty$ (inciso (g) Figura 4.4), pero el valor de la función se aproxima a este valor desde que $x = Carga_{max}$ (inciso (f) Figura 4.4). Esto se puede escribir como:

$$y \approx t_{max} = y(x = Carga_{max}) = f(x = Carga_{max}) = \frac{k_1}{1 + e^{-k_2(x-k_3)}} + k_4 \tag{4.15}$$

$$y \approx t_{max} = \frac{k_1}{1 + e^{-k_2(Carga_{max}-k_3)}} + k_4 \tag{4.16}$$

Aquí el valor de la exponencial tiende a cero, ya que $k_1 + k_4 = t_{max}$. Entonces,

$$e^{-(k_2 * Carga_{max}-k_3)} \approx 0 \tag{4.17}$$

La exponencial de un número negativo que se acerca a cero cuando este número es suficientemente grande. Suponga un número M (por ejemplo $M > 10$). Sustituyendo M en la Ec. 4.1.5,

$$k_2 * Carga_{max} - k_3 = M \tag{4.18}$$

Entonces k_2 se puede escribir en función de k_3 y M.

$$k_2 = \frac{M + k_3}{Carga_{max}} \tag{4.19}$$

Para resolver las otras incógnitas de la ecuación de congestión, necesitamos otro punto conocido. Suponga que la mitad del valor de congestión se alcanza cuando el número de vehículos en la arista es igual a la mitad de su carga máxima (inciso (d) Figura 4.4); esto es:

$$\frac{t_{max}}{2} = f\left(x = \frac{Carga_{max}}{2}\right) = \frac{k_1}{1 + e^{-k_2(x-k_3)}} + k_4 \tag{4.20}$$

Sustituyendo:

$$\frac{t_{max}}{2} = \frac{k_1}{1 + e^{-(k_2(\frac{Carga_{max}}{2})-k_3)}} + k_4 \tag{4.21}$$

Despejando la exponencial:

$$1 + e^{-(k_2(\frac{Carga_{max}}{2})-k_3)} = \frac{k_1}{\frac{t_{max}}{2} - k_4} \tag{4.22}$$

$$-k_2\frac{Carga_{max}}{2} + k_3 = ln\left(\frac{k_1}{\frac{t_{max}}{2} - k_4} - 1\right) \tag{4.23}$$

Ahora sustituimos k_2 de la Ec. (4.19):

$$k_3 = ln\left(\frac{k_1}{\frac{t_{max}}{2} - k_4} - 1\right) + \frac{M + k_3}{Carga_{max}}\frac{Carga_{max}}{2} \tag{4.24}$$

Despejando k_3

$$k_3 = 2ln\left(\frac{k_1}{\frac{t_{max}}{2} - k_4} - 1\right) + M \tag{4.25}$$

Ahora ya se puede escribir la función de congestión usando sólo parámetros de la calle, donde el parámetro M nos dice que tan pronunciada es la pendiente de la función sigmoide. Se ha visto que un valor de $M = 10$ genera una función de congestión adecuada para todos los escenarios.

4.1.6. Esquema de análisis de optimización colectiva

Esquema de optimización colectiva (EOC)

Nuestra propuesta descrita en la Tabla 4.1., la cual utiliza una estrategia de teoría de juegos conocida como Best Response Dynamics y el algoritmo Dijkstra para encontrar de forma colectiva un conjunto de rutas en donde la suma del tiempo total del conjunto de rutas sea menor que a un conjunto de rutas obtenidas individualmente.

Parámetros de entrada:
$G = (V, E)$ Grafo vial.
$V = (o, d) : o, d \in V$ Conjunto de viajes.
$c = ExN \rightarrow R^+$
1: para $i = 1$ hasta $i = \|V\|$
2: {
3: v_i= Dijkstra (G, c_0, o_i, d_i)
4: $R = R \cup v_i$
5: }
6: c_i= calcula congestión(G, R)
7: para $j = 1$ hasta I
8: {
9: a= Generar aleatorio $(1, \|V\|)$
10: $r'a$= Dijkstra (G, c_j, o_a, d_a)
11: $R = R - r_a \cup r'_a$
12: c_{j+1}=calcula congestión (G, R)
13:}
14: Devuelve R

Tabla 4.1: Algoritmo de optimización colectiva (AOC)

Desarrollo del algoritmo por etapas:

1.- Se inicializa el grafo con las características físicas de las calles representadas por aristas (longitud, velocidad, número de carriles), número de vehículos y su valor

de congestión.

2.- Se calculan las rutas más cortas (menor tiempo) para cada usuario utilizando el algoritmo de la ruta más corta: Dijkstra.

3.- Se actualizan los pesos de las aristas utilizando la función de congestión propuesta en la sección 4.1.4.

4.- Inicia la estrategia de Best Response Dynamics, con un número definido de iteraciones.

5.- Se selecciona un usuario al azar y se busca una nueva ruta con Dijkstra.

6.- Se calcula el tiempo que tardaría el usuario en realizar su nueva ruta.

7.- Se compara el tiempo de la anterior ruta con la nueva; si el tiempo del la nueva ruta es menor a la anterior, se asigna la nueva ruta al usuario y se descarta la anterior.

8.- Se calcula la congestión de las aristas afectadas por los cambios de ruta.

9.- Los pasos 9,10,11 y 12 se repiten el número de iteraciones que realiza Best Response Dynamics.

10.- Por último,se obtiene un conjunto de rutas de todos los usuarios, donde su colaboración indirecta permite minimizar la suma del tiempo del conjuntos de rutas.

Las rutas resultantes se pueden visualizar en el simulador de rutas urbanas (SUMO), donde se podrán observar las rutas de cada uno de los usuarios en un ambiente parecido a la realidad.

4.1.7. Análisis estratégico de rutas

Un juego esta definido por un conjunto de jugadores, de estrategias, los perfiles y funciones de utilidad. El problema que se platea en la presente tesis, es un problema de minimización, los jugadores son los usuarios, las estrategias se conforman de las diferentes rutas que los usuarios pueden tomar, los perfiles son el conjunto de rutas

y la utilidad será mayor, mientras el tiempo que tarda cada usuario en realizar su viaje sea menor.

Best Response Dinamics (BRD) (sección 2.4) es una estrategia de teoría de juegos, la cual tiene dos condiciones de parada, la primera consiste en un número determinado de iteraciones y la segunda es encontrar un equilibrio de Nash (sección 2.3). En el desarrollo del nuestro esquema EOC se integran las técnicas de BRD y Dijkstra (sección 2.6.1).

En cada iteración de BRD, genera un conjunto de perfiles con los viajes de cada usuario, estos perfiles pueden o no ser modificados. A continuación se representa la estrategia de BRD con el algoritmo de Dijkstra desde el punto de vista de la teoría de juegos.

1.- Jugadores: Usuarios.

2.- Estrategias: Todas las diversas rutas que pueden tomar los jugadores.

3.- Perfiles: En cada iteración de BRD, Dijkstra genera la ruta del viaje correspondiente a un usuario seleccionado y después se calcula su utilidad, si el nuevo viaje que encontró Dijkstra genera menor tiempo que el que se tiene actualmente, se cambia la ruta anterior por la nueva y se genera un nuevo perfil.

4.- Utilidad: Cada viaje tiene una duración, los cambios de los viajes van a depender de la utilidad que generen, dado que es un problema de minimización, se busca una menor utilidad (menor tiempo de viaje).

En la Figura 4.5, se muestra un ejemplo del funcionamiento del esquema de optimización colectiva propuesto desde el punto de vista de teoría de juegos. El cual contiene un grafo ponderado y una tabla con seis jugadores que pretenden ir del vértice a al c, el algoritmo de Dijkstra proporciona la ruta más corta a cada jugador, y con BRD se realizar un determinado número de iteraciones en donde las rutas de los jugadores son recalculadas por medio de Dijkstra, se realiza una comparación entre la utilidad que genera una nueva ruta encontrada por Dijkstra con la ya asignada al jugador, si la nueva ruta genera menos utilidad que la ya asignada, se realiza un cambio de ruta, en caso contrario no se realiza ningún cambio. A continuación se describe más

detalladamente:

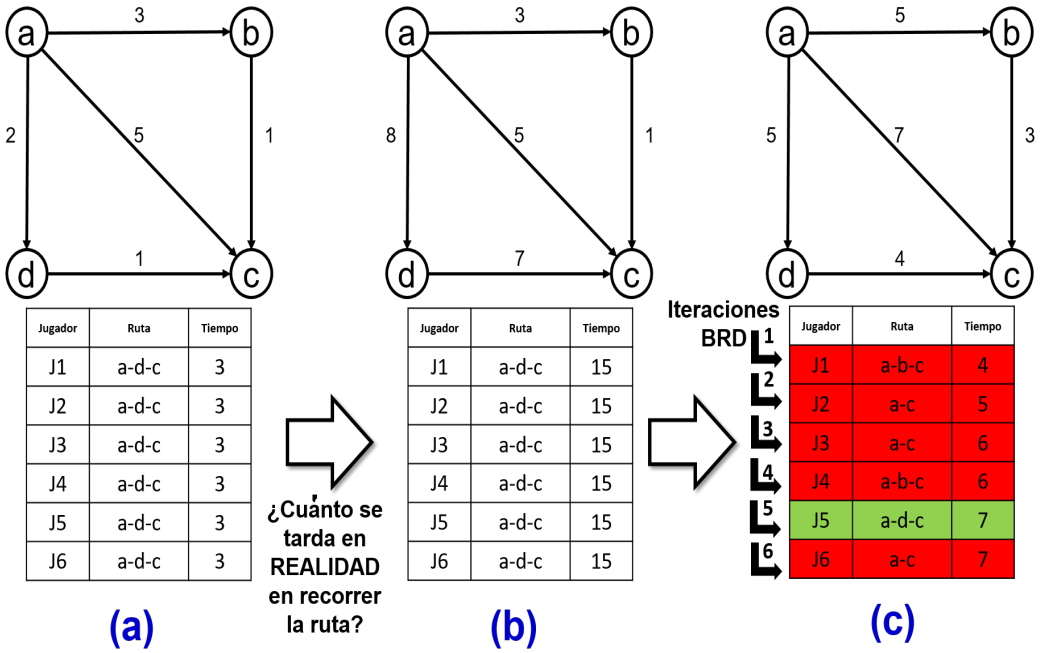


Figura 4.5: Procedimiento.

1.- En la Figura 4.5 inciso (a), seis jugadores pretender ir del punto a al c, Dijkstra calcula la ruta para cada jugador, sin considerar la ruta del resto de los jugadores que también quieren realizar el mismo viaje. A todos los jugadores les indico las misma ruta.

2.- La Figura 4.5 inciso (b), se calcula el tiempo que tardaría cada jugador en realizar su ruta tomando en cuenta que el peso de las aristas han cambiado, ya que se toma en cuenta que todos los jugadores pretenden pasar por las aristas que indican la misma ruta. Es decir, si se toma en cuenta que todos los jugadores van a utilizar la misma ruta el tiempo que se mostró en el inciso (a) para cada jugador en realidad será mayor.

3.- En el inciso (c) de la Figura 4.5 la estrategia de Best Response Dynamics realiza varias iteraciones donde se calcula nuevamente las ruta para los jugadores por medio de Dijkstra, el costo de las nuevas rutas son comparadas con las que tenian

anteriormente, en el caso de que un jugador no obtenga una ruta donde su tiempo sea mejor a la que ya tenia, no se realiza ningún cambio de ruta, pero, si el tiempo de la nueva ruta es menor, el jugador realiza un cambio de ruta. Al realizar este cambio se realizan modificaciones al grafo ya que las aristas que estaban afectadas por la ruta anterior, se les disminuye en uno el número de vehículos y las aristas de la nueva ruta se les suma uno, modificando el peso de las arista con la función de congestión. En caso de obtener la misma ruta y el mismo tiempo, no se realiza un cambio de ruta.

4.- Existe el caso en que se obtiene la misma ruta pero el tiempo es menor, tal como se muestra en la tabla de la Figura 4.5 inciso (c) en la fila coloreada de verde, lo que sucede en éstos casos es que los jugadores que tenían esa ruta, cambiaron a otra y el peso de las aristas que conforman esa ruta disminuyen y llega el momento en donde es buena opción para un jugador. En este caso el jugador conserva la misma ruta pero se actualiza su tiempo.

5.- Las filas de color rojo en la tabla de la Figura 4.5 inciso c son los jugadores que cambiaron su ruta, ya que el tiempo que durarán en recorrer su ruta será menor a la que tenían anteriormente.

4.2. Software Cartográfico OpenStreeMap

OpenSteetMap (OSM) es un proyecto colaborativo para crear mapas editables y libres, permite descargar mapas de diversos lugares alrededor del mundo, se descarga un script donde se especifican las características de las calles: su sentido, longitud, velocidad máxima, semáforos, prioridad, etc. Los mapas se crean utilizando información geográfica capturada con dispositivos GPS móviles, ortofotografías y otras fuentes libres. La cartografía y las imágenes creadas como los datos vectoriales almacenados en su base de datos, se distribuyen bajo licencia abierta, en si es una iniciativa abierta que busca construir de manera colaborativa mapas de acceso libre de todo el mundo [35].

Los datos o elementos básicos de la cartografía de OSM son:

- Los nodos (nodes): Son puntos que recogen una posición geográfica dada.
- Las vías (ways): Son una lista ordenada de nodos que representa una polilínea o un polígono.
- Las relaciones (relations): Grupos de nodos, vías u otras relaciones a las que se pueden asignar determinadas propiedades comunes.
- Las etiquetas (tags): Se pueden asignar a nodos, caminos o relaciones y constan de una clave (key) y un valor (value). Por ejemplo: *highway=trunk* define una vía como carretera troncal.

OpenSteeetMap facilita los datos en bruto para su descarga desde su propia página web y pueden ser modificados para ajustar un mapa con fines específicos.

A través de la herramienta OpenStreetMap se obtienen todas las propiedades que conforman la cartografía de una sección seleccionada. Para esto se ingresa a la pagina oficial de OpenSteetMap: *www.openstreetmap.org*, se selecciona una ubicación y área en el mapa para ser exportada: descargando un archivo en formato OSM. Este archivo se convierte a XML para su manipulación y lectura del simulador de rutas que se hablara más delante.

El archivo generado Tabla 4.1., contiene un identificador para cada uno de los vértices (*junction*) y para cada una de las aristas (*edge*). Una arista está delimitada por un par de vértices del mapa, otra característica que tienen es la longitud en metros y la velocidad máxima permitida en metros sobre segundo.

Tabla 4.1: Ejemplo de archivo XML

```
1    <edge id="187998872#2" from="1986252477"
2        to="1986252506" priority="4"
3        type="highway.residential"
4        index="0"
5        disallow="tram rail_urban rail rail_electric ship"
6        speed= "11.11"
7        length="34.41"
8        shape="191.39.267.22157.78,274.64"
```

```
9       </edge>
10      <junction id="1986252532"
11      type="priority"
12      x="241.21"
13      y="254.52"
14      incLanes="-183266091#3_0 187998872#0_0
15      183266091#2_0 -187998872#1_0"
16      />
```

Obteniendo estos datos se construye el grafo que representara el mapa de la ciudad, es importante recordar que un grafo está compuesto por un conjunto de vértices y aristas [26], los vértices representan los nodos (*junction*) y las aristas las uniones de éstos (*edge*), y cada uno de ellos tienen un "id" para su manipulación.

Entonces, el Tabla 4.1, gráficamente se vería como a continuación:

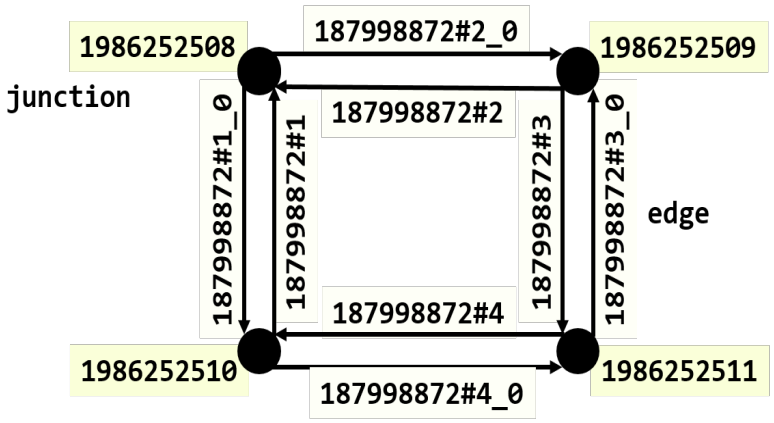


Figura 4.6: Propiedades del mapa de forma gráfica.

Para la presente investigación, se realizó un script para ser leído por Python el cual contiene la información necesaria para conformar el diccionario (Ver Sección A) que representa el grafo vial de la ciudad de Culiacán. Las claves que se muestran en el Tabla 4.2 (el número con 10 cifras) son los identificadores de los vértices (*junction*) en el archivo Tabla 4.1 y cada clave del diccionario contiene otras con los cuales tiene una arista, e implícitamente se forma el sentido de la calle. Dentro del arreglo Tabla 4.2 se guardan las características de cada una de las aristas como lo son: id, longitud, velocidad máxima permitida, número de carriles, número de carros y el valor de la congestión.

1986252508:1986252509=[A,L,V,CR,CA,CON], 1986252510=[A,L,V,CR,CA,CON]
1986252509:1986252508=[A,L,V,CR,CA,CON], 1986252511=[A,L,V,CR,CA,CON]
1986252510:1986252508=[A,L,V,CR,CA,CON], 1986252511=[A,L,V,CR,CA,CON]
1986252511:1986252510=[A,L,V,CR,CA,CON], 1986252509=[A,L,V,CR,CA,CON]

Tabla 4.2: Representación del grafo en Python.

4.3. Simulador de rutas urbanas

SUMO (Simulation of Urban MObility) fue fundado en 2001 por el Instituto de Sistemas de Transporte del Centro Aeroespacial Alemán. Es un simulador microscópico

de tráfico multimodal. Permite simular una determinada demanda de tráfico que consta de vehículos individuales que se mueven a través de una red de carreteras dada. La simulación permite abordar un gran conjunto de temas de gestión de tráfico. Cada vehículo es modelado explícitamente, tiene una ruta propia, y se mueve individualmente a través de la red. Las simulaciones son deterministas por defecto pero hay varias opciones para introducir aleatoriedad.

Incluye todas las aplicaciones necesarias para preparar y realizar una simulación de tráfico, tales como:

1.- Simulación.

2.- Movimiento del espacio continuo y tiempo discreto.

3.- Diferentes tipos de vehículos.

4.- Las calles de carril múltiple con cambio de carril.

5.- Una interfaz gráfica de usuario rápida de openGL.

6.- Gestiona redes con varios bordes (*calles*).

7.- Rápida velocidad de ejecución.

8.- Interoperabilidad con otras aplicaciones en tiempo de ejecución.

9.- Salidas basadas en la red, basadas en vehículos y basadas en detectores.

10.- Importaciones VISUM, Vissin, Shapefiles, OSM, RoboCup, MATsim, OpenDrive y XML.

11.- Los valores faltantes se determinan a través de la heurística.

12.- Enrutamiento.

13.- Rutas microscópicas.

14.- Diferentes algoritmos de asignación dinámica de usuarios.

15.- Alta portabilidad.

16.- Solo se utilizan bibliotecas estándar C++ y portátiles.

17.- Existen paquetes para distribuciones Linux, Windows.

18.- Alta interoperabilidad mediante el uso de datos XML, únicamente código abierto.

SUMO es un simulador que se sustenta en una amplia documentación, la cual se basa en modelos de tráfico que reflejan la realidad. De ahí la importancia de seleccionar este simulador para ver reflejados los resultados que proporcione el esquema EOC (Sección 4.1.6). La función de congestión que proporciona el tiempo se basa en algunas características del ámbito real, pero al introducir los resultados que se obtengan en cada simulación del esquema EOC a SUMO, se están incluyendo otros factores que reflejan el comportamiento real de los vehículos y su entorno, tales como: la agresividad de los conductores al conducir, los semáforos, señalamientos, el rango de prioridad de las calles, factores que van a repercutir en el tiempo. El cual estaría más apegado a la realidad.

El procedimiento de la instalación y configuración del ambiente en SUMO se puede consultar en el Apéndice B y la comunicación entre Python y SUMO se detalla en la Apéndice C.

Capítulo 5

Experimentación y resultados

En la presente sección se muestran los resultados computacionales con el objetivo de evaluar el desempeño del esquema propuesto en la sección 4.1.6. La ejecución del esquema de optimización colectiva ha sido desarrollado en un ordenador personal. La implementación computacional de los algoritmos han sido desarrolladas en Python 3.6.0

5.1. Resultados de las simulación en diferentes escenarios

Los resultados que se obtuvieron dependen esencialmente de dos factores: el primero corresponde en inicializar las calles con un porcentaje de vehículos no mayor a su carga máxima. El segundo lo determinan el número de viajes que se van a introducir en cada simulación. Estos factores son de suma importancia para la función de congestión, ya que mientras más sean los vehículos que estén presenten en las calles, la congestión va a ser mayor y se ve reflejado en los resultados que se exponen a continuación. Así también, el comportamiento del esquema que se propone es diferente en cada escenario, los cuales se diferencian en el porcentaje de vehículos que contienen sus calles.

Para las pruebas se utilizó el grafo de la Figura 5.1 el cual representa una sección del mapa de la ciudad de Culiacán, Sinaloa. En la Figura 5.2 , el vértice sombreado de verde es considerado el origen de los viajes de los usuarios y el amarillo el destino.

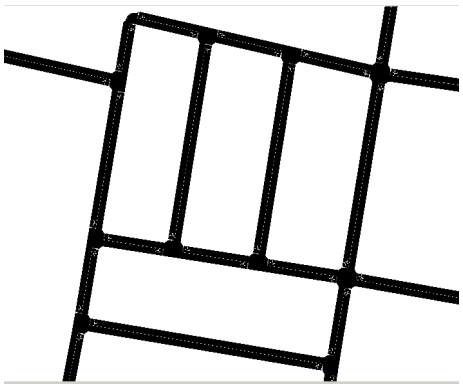


Figura 5.1: Red vial de vehículos de la ciudad de Culiacán, Sinaloa.

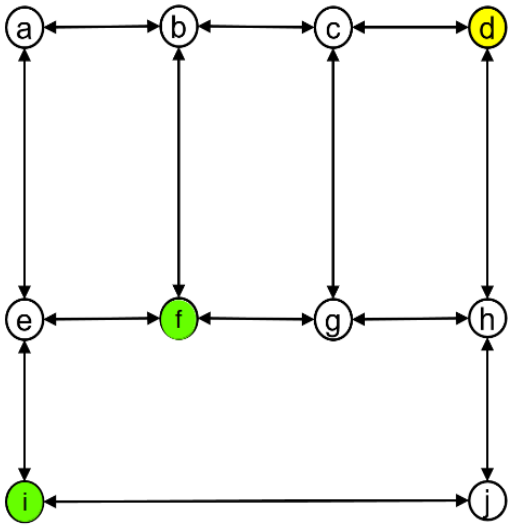


Figura 5.2: Representación de la Figura 5.1, por medio de un Grafo y los vértices origen y destino de los viajes que se realizarán en las simulaciones.

Todas las simulaciones fueron comparadas con el tiempo que arrojo solamente el algoritmo de Dijkstra, debido a que en el estado del arte, Google Maps (el cual da solución al mismo problema que se ataca en la presente tesis) utiliza el algoritmo de

Dijkstra [12] (Sección 3.1) para proporcionar a los usuarios sus viajes pero lo hace de forma individual, sin considerar a los demás usuarios. A diferencia con nuestra propuesta, que integra las decisiones de los demás usuarios para tomar una mejor decisión, con el objetivo de minimizar la suma del tiempo total del conjunto de viajes.

Nuestra función de congestión propuesta(FCP) muestra un tiempo total de las rutas para cada viaje, tomando en cuenta las características de las calles, así también en el Simulador SUMO se corrieron las rutas resultantes tanto por el algoritmo de Dijkstra y nuestra Estrategia de Optimización Colectiva (EOC) y se obtuvo un tiempo total de las rutas. Estos resultados se mostrarán a continuación en diferentes escenarios.

Escenario 1:

Se realizaron 50 simulaciones del primer escenario con las características descritas en la Tabla 5.1.

Viajes	100
Iteraciones BRD	1000
Carga vehicular	0 %
Carga vehicular máxima en las calles	887 vehiculos

Tabla 5.1: Escenario 1: Sin carga vehicular en las calles

Cuando la carga vehicular es 0, las aristas del grafo se inicializarón con cero vehículos, y por lo tanto no se presenta congestión vehicular inicial.

Los resultados obtenidos con nuestra función de congestión propuesta (FCP), se muestran en la Figura 5.2.

Dijkstra	EOC	Porcentaje de mejora	Vehiculos
441 min.	271 min.	38.55 %	100

Tabla 5.2: Resultados del escenario 1 en FCP.

Los 100 vehiculos de las Tabla 5.2, corresponden a los 100 viajes realizados en este mismo escenario.

El comportamiento de los resultados de las 50 simulaciones se representan en la Figura 5.3, la linea naranja corresponde a los resultados de Dijkstra y la azul la de EOC. El tiempo de la linea azul se mantuvo inferior a la roja, la estrategia EOC distribuyo los vehículos por una ruta diferente a la de Dijkstra con la cual el tiempo total del conjunto de rutas fue menor. En promedio Dijkstra obtuvo un tiempo de 441 min, mientras que nuestra propuesta EOC 271 Min, una ventaja del 38.55 %.

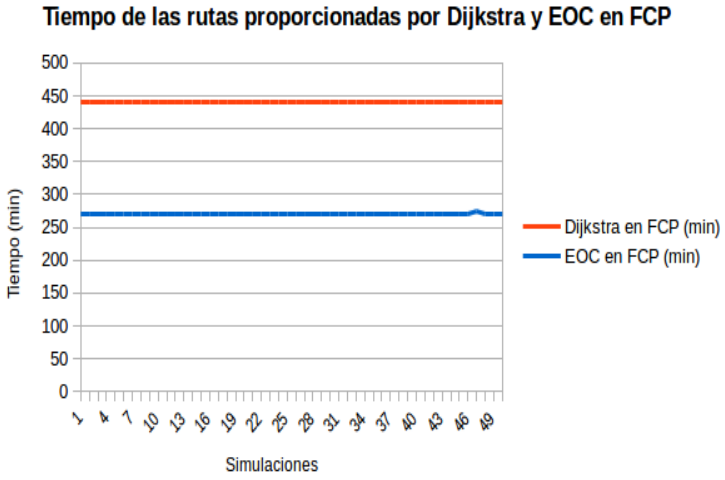


Figura 5.3: Resultados del escenario 1 en FCP

Los resultados de Dijkstra (linea naranja en la Figura 5.3) es una linea que no cambia, ya que en todas las simulaciones dio como resultado las misma ruta a cada viaje, mientras que los resultados de EOC (linea azul en la Figura 5.3) no es completamente recta, ya que hubo al menos una ruta diferente en alguna simulación.

Las rutas proporcionadas por Dijkstra y EOC fuerón simuladas en SUMO y el tiempo promedio obtenido del conjunto de rutas se muestra en la Tabla 5.3 y el comportamiento de las 50 simulaciones en la Tabla 5.4. Es importante recordar que el tiempo que dio como resultado SUMO es muy significativo,puesto que la simulación está basada en comportamientos y patrones de la vida real. Lo cual nos permite poner en práctica los resultados obtenidos como si los estuviéramos aplicando en las calles de la ciudad de Culiacán.

Dijkstra	EOC	Porcentaje de mejora	Vehiculos
351.7 min.	186.8 min.	46.88 %	100

Tabla 5.3: Resultados del escenario 1 en SUMO.

En la simulación las rutas proporcionadas por el algoritmo de Dijkstra tuvieron una duración de 351.7 min. y con nuestra propuesta EOC 186.8 min, lo cual se traduce

a un 46.88 % a favor de nuestra propuesta.

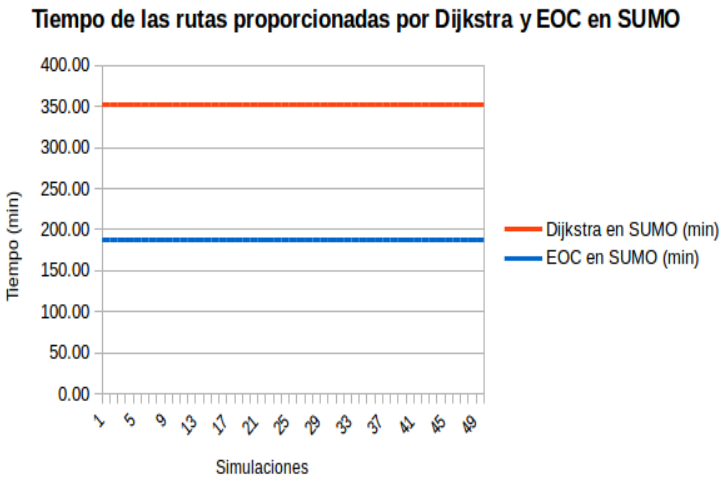


Figura 5.4: Resultados del escenario 1 en SUMO

El comportamiento de los resultados en las 50 simulación es parecida a la obtenida con la FCP, la misma ruta para Dijkstra y al menos dos diferentes para EOC.

Escenario 2:

Se realizaron 50 simulaciones con las siguientes características ver Tabla 5.4

Usuarios	100
Iteraciones BRD	1000
Carga vehicular	Del 0 % al 25 %
Carga vehicular máxima en las calles	887 vehiculos

Tabla 5.4: Escenario 2: carga vehicular del 0 % al 25 %.

En este escenario la carga vehicular es del 0 % al 25 % de la carga máxima permitida en cada calle, es decir, las calles se inicializaron en un margen de 0 a 221 vehículos en total. Los resultados obtenidos se describe en la Tabla 5.5

Dijkstra	EOC	Porcentaje de mejora	Vehiculos
441 min.	271.22 min.	38.50 %	199

Tabla 5.5: Resultados del escenario 2 en FCP.

Los resultados de las 50 simulaciones se muestran en la Figura 5.5: la linea azul corresponde a los resultados de Dijkstra y la naranja de EOC. Se puede observar que la forma de las lineas son similares al escenario anterior. La congestión en las calles no es tan significativa para que los algoritmos puedan distribuir los vehículos por varias rutas diferentes. En este escenario nuestra propuesta sigue manteniendo el tiempo inferior en comparación con los resultados de Dijkstra, obteniendo una ventaja del 38.5 %.

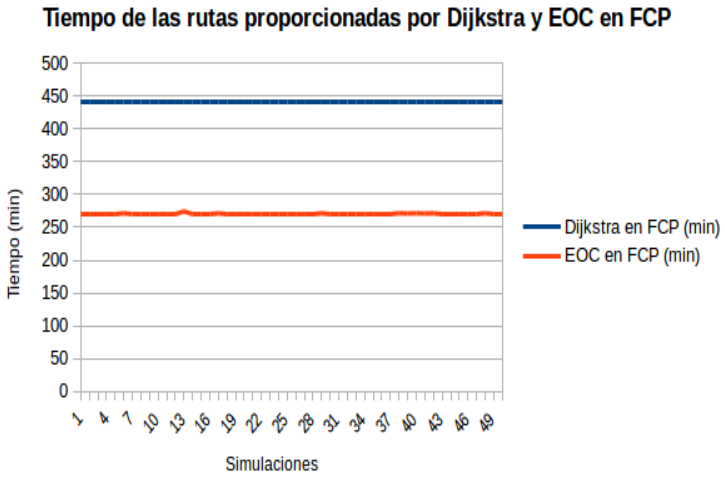


Figura 5.5: Resultados del escenario 2 en FCP

En el simulador SUMO, las rutas resultantes del algoritmo de Dijkstra tuvierón una duración de 351.7 min, mientras que con EOC 186.8 min, obteniendo una ventaja del 46.88 % respecto a Dijkstra. La Tabla 5.6 muestra los resultados obtenidos y la Tabla 5.6 el comportamiento de las 50 simulaciones en ambos algoritmos.

Dijkstra	EOC	Porcentaje de mejora	Vehiculos
351.7 min.	186.8 min.	46.89 %	199

Tabla 5.6: Resultados del escenario 2 en SUMO.

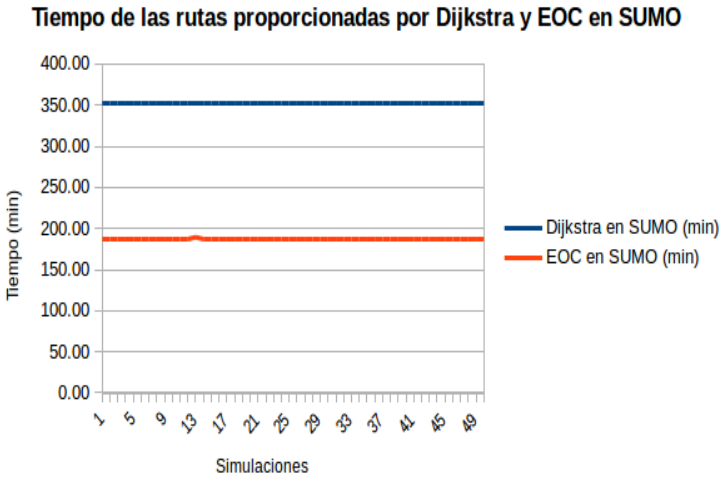


Figura 5.6: Resultados del escenario 2 en SUMO

Escenario 3:

Se realizaron 50 simulaciones con las siguientes características ver Tabla 5.7

Usuarios	100
Iteraciones BRD	1000
Carga vehicular	Del 25 % al 50 %
Carga vehicular máxima en las calles	887 vehiculos

Tabla 5.7: Escenario 3: carga vehicular del 25 % al 50 % en FCP.

Los resultados obtenidos en este escenario se presentan en la Tabla 5.8, el porcentaje de mejora es menor que los dos casos anteriores, esto es debido a que el grafo está siendo congestionado a tal grado que las alternativas de cambios de rutas por los

vehículos son mayores. En esta ocasión el mapa de ejemplo de la ciudad de Culiacán tiene de 221 a 443 vehículos. Para nuestra propuesta existe una ventaja del 32.74 % respecto al tiempo de Dijkstra.

Dijkstra	EOC	Porcentaje de mejora	Vehículos
446.94 min.	300.62 min.	32.74 %	413

Tabla 5.8: Resultados 3: carga vehicular del 25 % al 50 % en FCP.

El comportamiento de las 50 simulaciones se pueden observar en la siguiente Figura 5.7, nuestra propuesta (linea naranja) tiene mejores resultados en comparación con los obtenidos por Dijkstra (linea azul). En este escenario las lineas no son rectas lo cual indica que las rutas proporcionadas por Dijkstra y EOC estuvieron cambiando, lo cual origino que se obtuvieran tiempos diferentes en cada simulación.

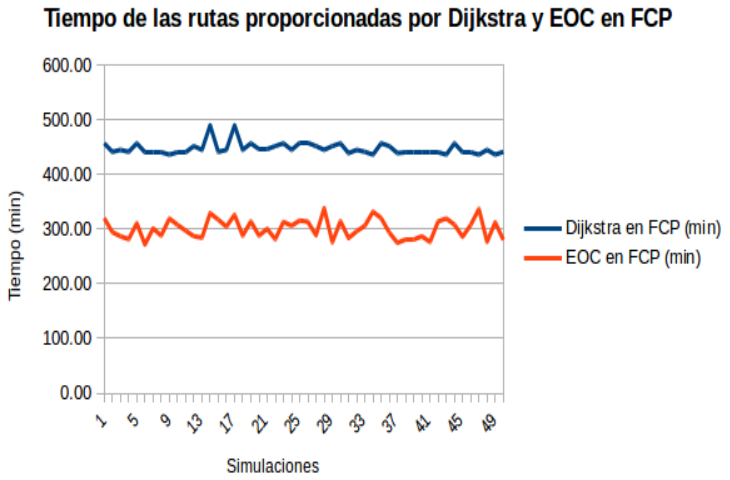


Figura 5.7: Resultados del escenario 3 en FCP

En el simulador de rutas SUMO, los resultados obtenidos se muestran en la Tabla 5.9.

Dijkstra	EOC	Porcentaje de mejora	Vehiculos
284.13 min.	236.02 min	16.94 %	413

Tabla 5.9: Resultados 3: carga vehicular del 25 % al 50 % en SUMO.

El porcentaje de mejora es significativamente menor al escenario anterior, esto es debido al aumento de vehículos en las calles, EOC ha obtenido un ventaja del 16.94 % respecto a Dijkstra , la distribución de las rutas en el mapa seleccionado ha sido mayor, ya que EOC a cambiado de rutas en varias ocasiones en cada simulación, al igual que Dijkstra que también a buscado otras alternativas de las rutas resultantes en los anteriores escenarios. Las rutas proporcionadas por ambos algoritmos tienden a ser las mismas.

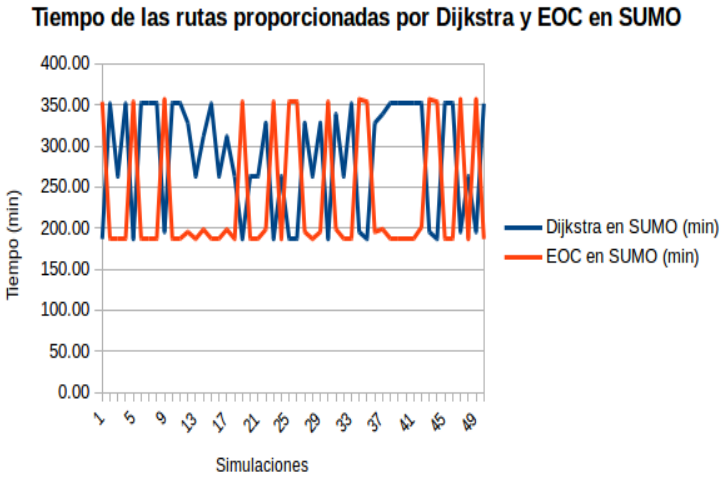


Figura 5.8: Resultados del escenario 3 en SUMO

Escenario 4:

Se realizaron 50 simulaciones con las siguientes características descritas en la Tabla 5.10

Usuarios	100
Iteraciones BRD	1000
Carga vehicular	Del 50 % al 75 %
Carga vehicular máxima en las calles	887 vehiculos

Tabla 5.10: Escenario 4: carga vehicular del 50 % al 75 %.

En el actual escenario, la congestión asignada es del 50 % al 75 %. Los resultados obtenidos se muestran en la Tabla 5.11. En donde se puede observar que existe una diferencia del 15.97 % a favor de nuestra propuesta EOC. El rango de vehículos en las calles va de 443 a 665.

Dijkstra	EOC	Porcentaje de mejora	Vehiculos
442.66 min.	371.97 min.	15.97 %	635

Tabla 5.11: Resultados 4: carga vehicular del 50 % al 75 % en FCP.

El comportamiento de los resultados en las 50 simulaciones se representan en la Figura 5.9.

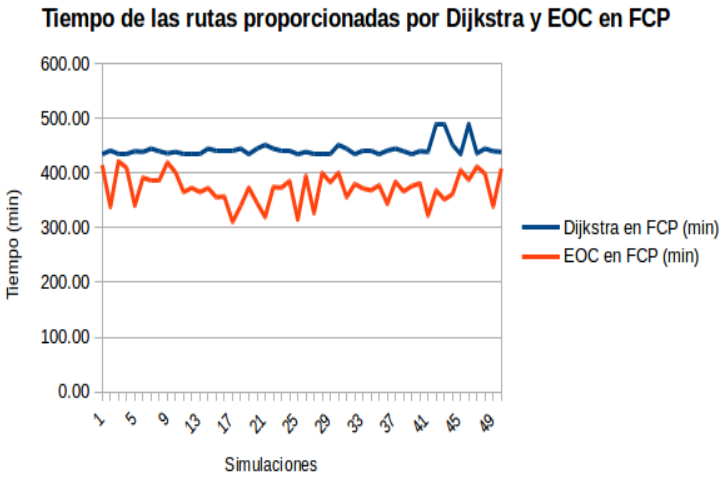


Figura 5.9: Resultados del escenario 4 en FCP

En SUMO la diferencia de tiempo de ambos algoritmos fue del 16.93 % a favor de nuestra propuesta EOC, en este escenario la congestión de las calles es considerada y nuestra propuesta distribuye los vehículos por rutas alternas a las proporcionadas por Dijkstra, aquí la distancia que pudieran tener las rutas no es tan importante como en los escenarios anteriores ya que en este punto todas las rutas posibles aun cuando unas estén mas largas que otras el tiempo será similar. La Tabla 5.12, muestra los resultados obtenidos y la Tabla 5.10, el comportamiento de los resultados en las 50 simulaciones.

Dijkstra	EOC	Porcentaje de mejora	Vehiculos
296.7 min.	282.1 min.	16.93 %	635

Tabla 5.12: Resultados 4: carga vehicular del 50 % al 75 % en SUMO.

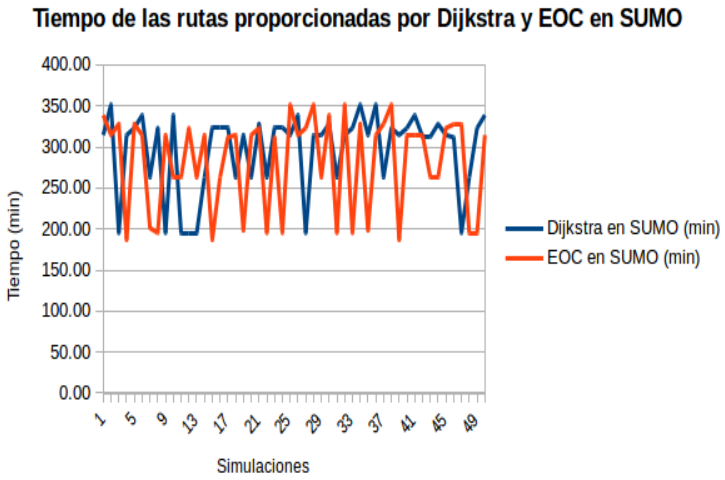


Figura 5.10: Resultados del escenario 4 en SUMO

Escenario 5:

Se realizaron 50 simulaciones con las siguientes características descritas en la Tabla 5.13

Usuarios	100
Iteraciones BRD	1000
Carga vehicular	Del 75 % al 100 %
Carga vehicular máxima en las calles	887 vehiculos

Tabla 5.13: Escenario 4: carga vehicular del 75 % al 100 %.

Los resutlados obtenidos en el programa en Python (PP) cuando la congestión va del 75 % al 100 % se muestran en la Tabla 5.14. Las rutas proporcionadas por Dijkstra tuvieron una duración de 435.07 min. y EOC 433.97 min, es decir una diferencia del 0.25 % a favor de EOC.

Dijkstra	EOC	Porcentaje de mejora	Vehiculos
435.07 min.	433.97 min.	0.25 %	854

Tabla 5.14: Resultados 5: carga vehicular del 75 % al 100 % en FCP.

El comportamiento de los resultados de las 50 simulaciones se muestra en la Tabla 5.11, las rutas de ambos algoritmos tienden a ser las mismas, ya que EOC no encuentra rutas alternativas por las cuales evitar el tráfico, todas las calles se encuentran saturadas.

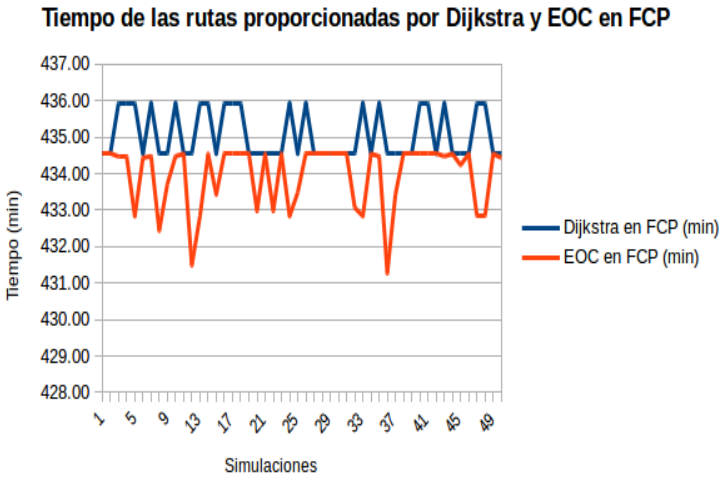


Figura 5.11: Resultados del escenario 5 en FCP

En un escenario donde las calles están saturadas de vehículos, un usuario tarda el mismo tiempo en recorrer una ruta de menor distancia con muchos vehículos que una ruta de mayor distancia con pocos vehículos.

La simulación en SUMO, EOC presenta el 4.84 % de mejora respecto a Dijkstra, los resultados se muestran en la Tabla 5.15 y su comportamiento en la Tabla 5.12.

Dijkstra	EOC	Porcentaje de mejora	Vehiculos
295.41 min.	281.11	4.84 %	854

Tabla 5.15: Resultados 4: carga vehicular del 75 % al 100 % en SUMO.

Las rutas resultantes de EOC y Dijkstra son prácticamente las mismas, de ahí que los tiempos sean similares.

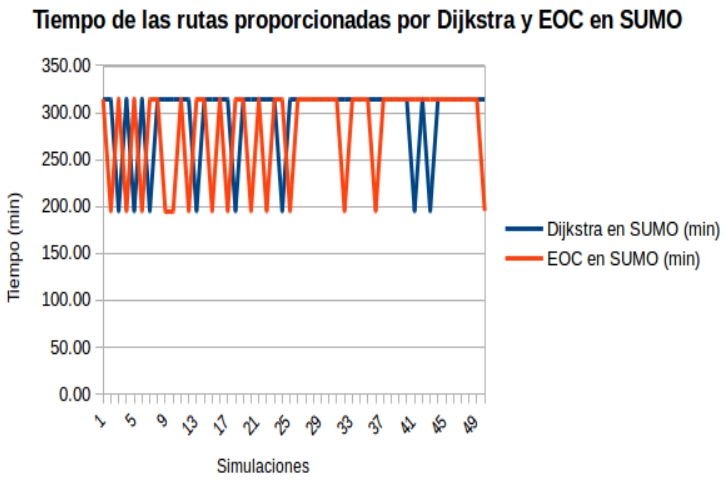


Figura 5.12: Resultados del escenario 5 en SUMO

En los cuatro escenario presentados, mientras más congestión tiene la red vial, los resultados de nuestra propuesta EOC y Dijkstra tienden a ser los mismos.

Tal como se observa en cada uno de los escenarios la carga vehicular en las calles va aumentando del 0 % al 100 %,es decir de 0 a 887 vehículos. En Figura 5.13 se visualiza el aumento de vehículos conforme a cada uno de los escenarios.

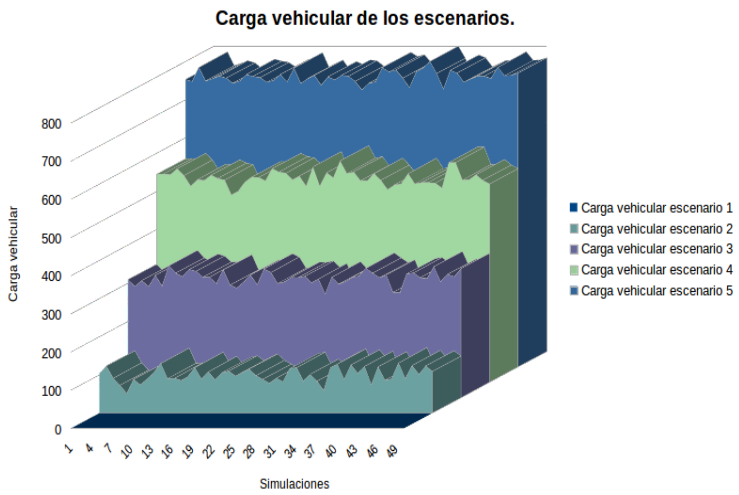


Figura 5.13: Carga vehicular en los escenarios

El desglose de datos resultantes de los diferentes escenarios se pueden consultar en el Apéndice D.

Tiempo promedio de los cinco escenarios en SUMO y FCP

¿Cuál es el comportamiento del tiempo de los cinco escenarios en Dijkstra y EOC en el Simulador SUMO?.
En la Figura 5.14, se observa la relación de la carga vehicular de cada uno de los escenarios con el tiempo promedio obtenido en cada uno.

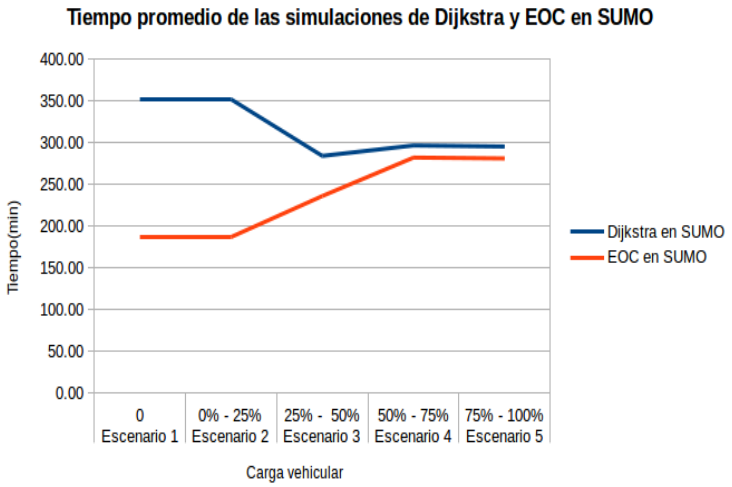


Figura 5.14: Resultados promedio de los cinco escenarios en SUMO

El comportamiento de los resultados de EOC en cada escenario es como una sigmoide tal como se planteo en nuestras hipótesis. El tiempo de los escenarios de Dijkstra y SUMO tienden a ser los mismo cuando se presenta un alto grado de congestión en las calles.

Por otra parte los resultados promedios con FCP, se puede ver en la Figura 5.15, y también cumple el comportamiento esperado, ya que al aumentar la congestión vehicular en las calles el tiempo del conjunto de rutas será mayor.

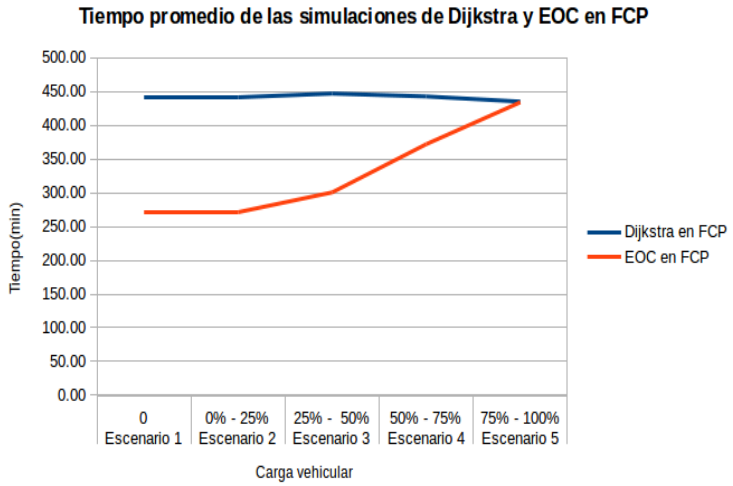


Figura 5.15: Resultados promedio de los cuatro escenarios en FCP

Comportamiento de las rutas de Dijkstra y EOC en SUMO

Comportamiento de los resultados de Dijkstra Figura 5.16 vs nuestra propuesta EOC Figura 5.17 vistos en el simulador de SUMO del Escenario 2.

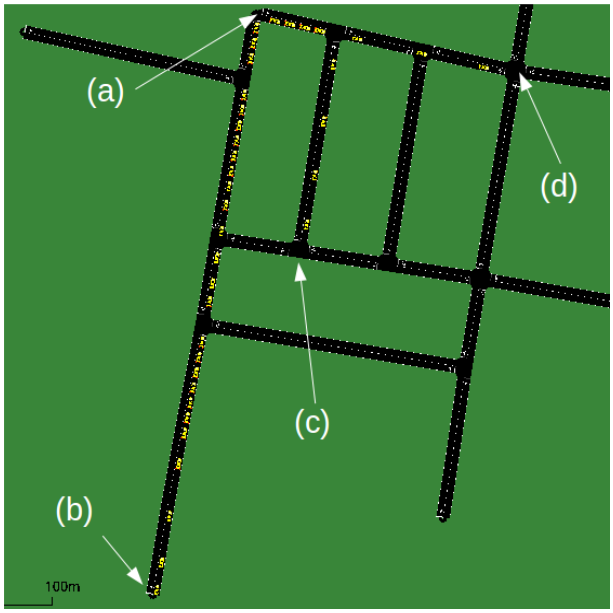


Figura 5.16: Comportamiento de los vehículos con Dijkstra en SUMO

Con Dijkstra los usuarios solo toman una sola ruta (Figura 5.16) para realizar su viaje. En el inciso (a) de la Figura 5.16, se origino un punto de conflicto con las rutas proporcionadas por Dijkstra, en los vehículos que toman la ruta del inciso (b) a (d) (Figura 5.16) que se detienen por los vehículos que se desplazan del punto (c) a (d) (Figura 5.16), ya que las calles que conforman la ruta de (c) a (d) tiene preferencia. Es común ver este fenómeno denominado cuello de botella lo cual genera un retraso en los vehículos, aumentando el tiempo que tardaría en recorrer la ruta.

Mientras que nuestra propuesta EOC, las rutas de los usuarios se distribuyen en el grafo vial, tal como se muestra en la Figura 5.17. Los vehículos se desplazan del punto (a) a (c) (Figura 5.17) y del punto (b) a (c) (Figura 5.17) por rutas diferentes sin crear conflictos.

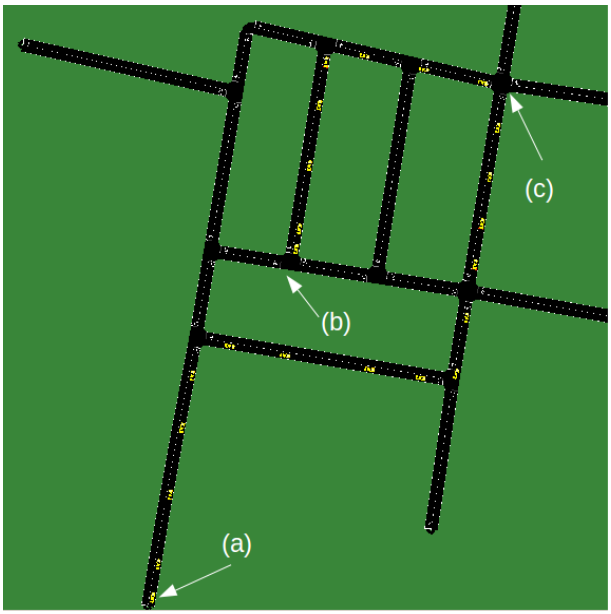


Figura 5.17: Comportamiento de los vehículos con EOC en SUMO

En este caso el algoritmo de Dijkstra utilizó la ruta de longitud más corta originando un conflicto entre sus dos rutas, y el algoritmo de EOC utilizó una ruta que a pesar de ser más larga su longitud obtuvo un tiempo menor que el de Dijkstra.

En otro escenario EOC dio como resultado una distribución de los vehículos por cuatro rutas diferentes, tal como se muestra en la Figura 5.18.

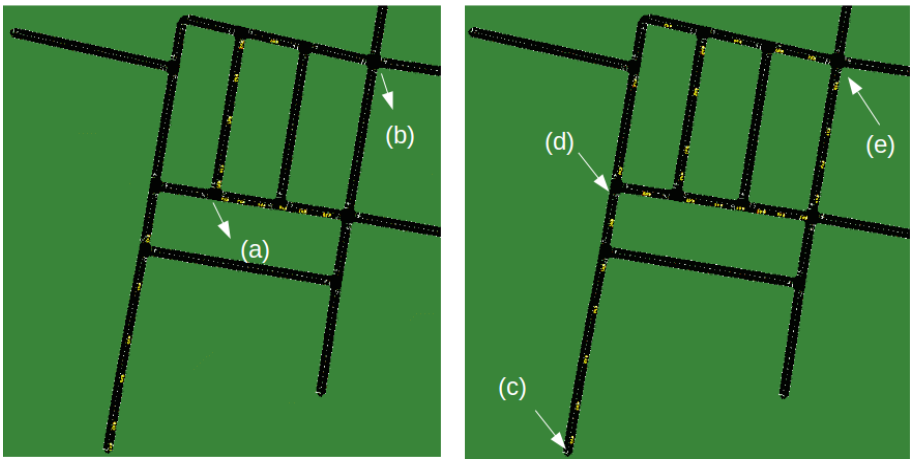


Figura 5.18: EOC en SUMO

En el inciso (a) de la Figura 5.18, se indica el inicio de la ruta que va de (a) a (b) y se divide en dos rutas. El punto (c), es el origen de la ruta de (c) a (e), esta ruta se divide también en dos en el punto (d), los vehículos de esta ruta toman rutas diferentes. De esta forma EOC obtiene una ventaja de tiempo respecto a Dijkstra que solo realiza una ruta para cada viaje de (a) a (c) y de (b) a (c) en la Figura 5.19.

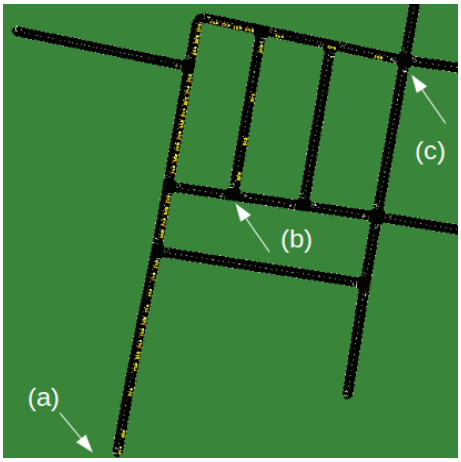


Figura 5.19: Comportamiento de los vehículos con Dijkstra en SUMO

Capítulo 6

Conclusiones y trabajos futuros

La principal aportación en este trabajo es un esquema de análisis de rutas colectivas utilizando Best Response Dynamics y Dijkstra.

Los resultados obtenidos con la metodología propuesta son favorables para minimizar tiempos de trayectos en una red vial. Los vehículos se distribuyen sobre la red vial de tal manera que minimizan su tiempo de trayecto de forma colectiva.

Cuando la carga vehicular en las calles no supera su carga máxima, el EOC en SUMO y FCP es capaz de minimizar el tiempo considerablemente. Por ejemplo, cuando la red vial no presenta carga vehicular, el tiempo disminuye un 46.89 % respecto a utilizar solamente Dijkstra que genera rutas individuales y cuando la carga vehicular está del 75 % al 100 % el tiempo disminuye un 4.84 %, tal como se mostró en la simulación del Escenario 5, en donde se distribuyeron 100 vehículos y se ahorraron en promedio 14.3 minutos.
Sin embargo cuando la carga vehicular de las calles supera la carga máxima es difícil encontrar rutas alternativas porque todas las calles están saturadas. El punto importante es mantener un flujo de vehículos por todas las vías sin que se lleguen a saturar las calles.

Como trabajo futuro se han considerado los puntos siguientes:

1.- Utilizar el EOC en casos reales, y obtener datos de movimiento vehicular en

tiempo real.

2.- Diseñar una metodología para particionar el mapa de la ciudad y calcular distancias muy largas.

3.- Implementar esta metodología en problemas similares al que se atendió en la presente tesis. Por ejemplo en una red eléctrica, de agua potable, de Internet, entre otras. Redes que se puedan representar por medio de un grafo dirigido y ponderado.

Es importante realizar una comparación del comportamiento de los resultados en SUMO y la FCP de los cinco escenarios con nuestra EOC, se puede observar el corto margen de diferencia entre ambas lineas, lo cual nos permite ver el acercamiento que se tuvo con nuestra función de congestión propuesta y un simulador de movilidad urbana (SUMO) que contiene un amplio número de comportamientos del la vida real.

En la Figura 6.1, se presenta los resultados del escenario 1.

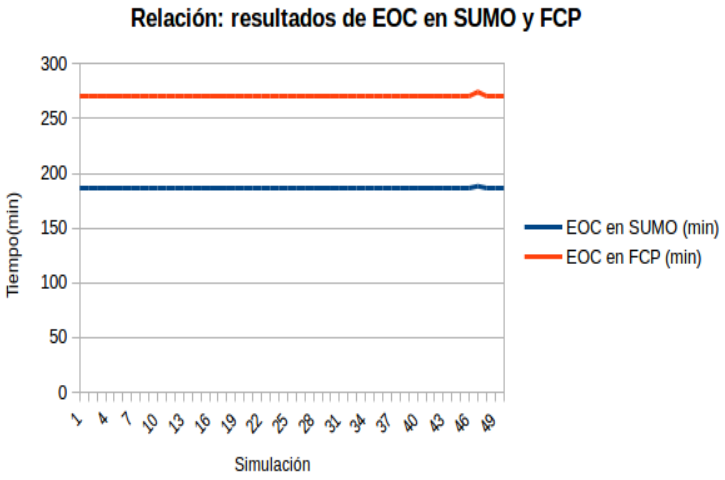


Figura 6.1: Relación: EOC en SUMO Y FCP del escenario 1

En la Figura 6.2, se presenta los resultados del escenario 2.

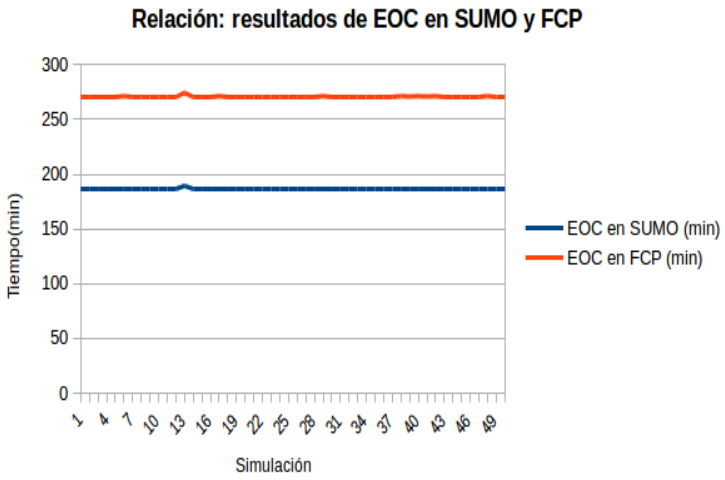


Figura 6.2: Relación: EOC en SUMO Y FCP del escenario 2

En la Figura 6.3, se presenta los resultados del escenario 3.

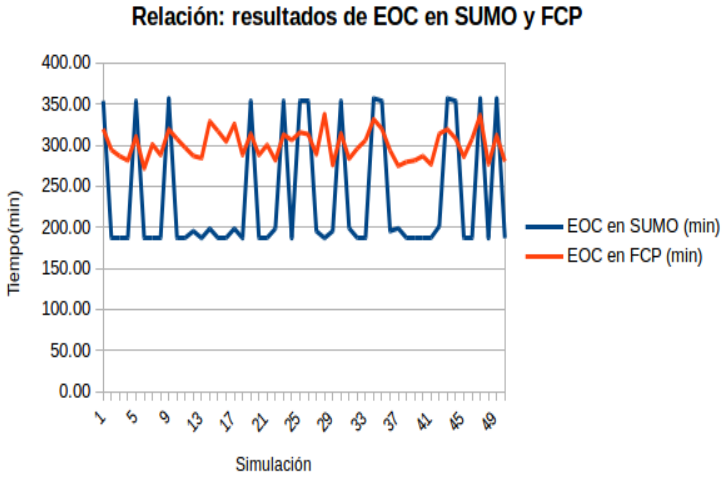


Figura 6.3: Relación: EOC en SUMO Y FCP del escenario 3

En la Figura 6.4, se presenta los resultados del escenario 4.

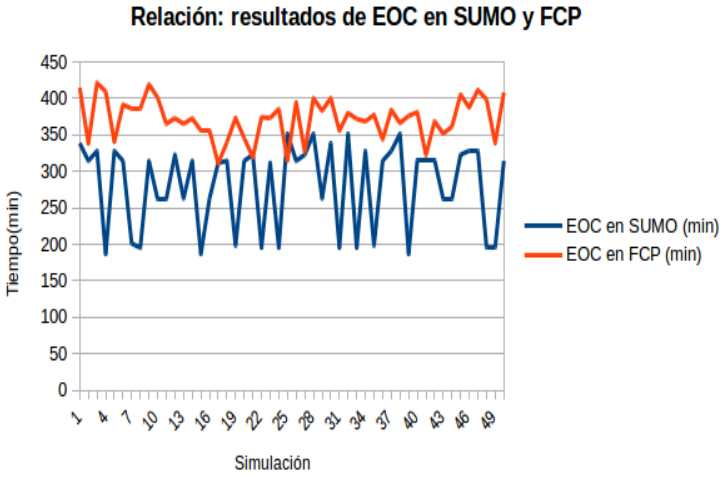


Figura 6.4: Relación: EOC en SUMO Y FCP del escenario 4

En la Figura 6.5, se presenta los resultados del escenario 5.

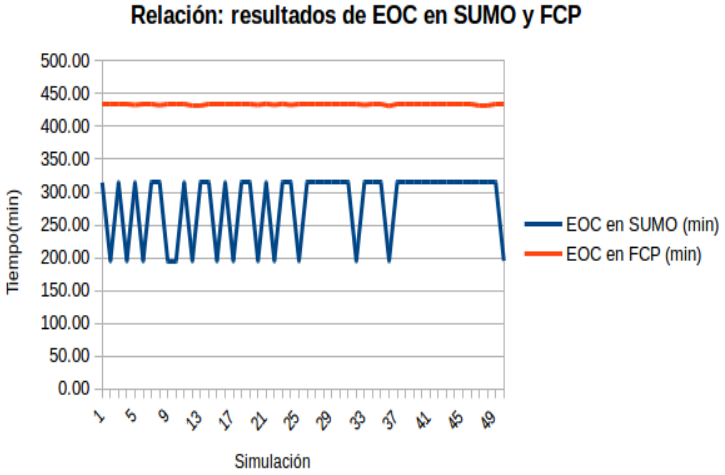


Figura 6.5: Relación: EOC en SUMO Y FCP del escenario 5

En un futuro se podrían agregar más variables a la función de congestión que nos permita acercarnos a los resultados del simulador de rutas (SUMO), esto con el fin de obtener resultados más apegados a la realidad.

Apéndice A

Python

Python cuenta con estructuras de datos eficientes, de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para el desarrollo rápido de aplicaciones en diversas áreas [23].

El intérprete de Python y la extensa biblioteca estándar están a libre disposición en forma binaria y de código fuente para las principales plataformas desde el sitio web de Python. El intérprete de Python puede extenderse fácilmente con nuevas funcionalidades y tipos de datos implementados en C o C++ (u otros lenguajes accesibles desde C). Python también puede usarse como un lenguaje de extensiones para aplicaciones personalizables [20].

Python ofrece muchas opciones para el desarrollo web:

- Frameworks como Django y Pyramid.
- Micro-frameworks como Frasco y Botella.
- Sistemas avanzados de administración de contenido como Plone y django CMS.

La biblioteca estándar de Python admite muchos protocolos de Internet:

- HTML, XML.

- JSON.
- Procesamiento de correo electrónico.
- Soporte para FTP, IMAP y otros protocolos de Internet.
- Interfaz de socket fácil de usar.

Y el índice de paquetes tiene aún más bibliotecas.

- Solicitudes, una poderosa biblioteca de cliente HTTP.
- BeautifulSoup, un analizador de HTML que puede manejar todo tipo de HTML extraños.
- Feedparser, para analizar feeds RSS / Atom.
- Paramiko, implementando el protocolo SSH2.
- Twisted Python, un marco para la programación de redes asíncronas. [23]

La parte de programación se decidió realizar en el lenguaje Python porque, admite el protocolo HTML que fue utilizado para la creación de scripts que contienen toda la información referente de la cartografía, la cual es útil para la creación del grafo y poder representarlo en un diccionario. Python además nos permite tener una comunicación con el simulador de rutas de interés (SUMO) y observar el comportamiento que tienen los resultados que se obtengan con el algoritmo que se propone en el actúa trabajo de tesis.

Diccionario en Python

Un diccionario es un tipo de dato en Python con características especiales que nos permiten almacenar cualquier tipo de valor como enteros, cadenas, listas e incluso otras funciones. Estos diccionarios permiten identificar cada elemento por una clave (*Key*). Pasos para la creación y manipulación de un diccionario.

1. Para definir un diccionario, se encierra el listado de valores entre llaves. Las parejas

de clave y valor se separan con comas, y la clave y el valor se separan con dos puntos.

Tabla A.1: Definición de diccionario en Python.

```
1    diccionario={'nombre':'Sofia',edad':24,
2    'domicilio':'Conocido'}
```

2. Se puede acceder al elemento de un Diccionario mediante la clave de este elemento.

Tabla A.2: Acceder a diccionario en Python.

```
1    print diccionario['nombre'] #Sofia
2    print diccionario['edad'] #24
3    print diccionario['domicilio'] #Conocido
```

3. Para recorrer todo el contenido del Diccionario, se pueden utilizar una estructura for.

Tabla A.3: Recorrer diccionario.

```
1    for key in diccionario:
2        print(key,":",diccionario[key])
```

La creación y manipulación del grafo que representa la red vial de vehículos de la ciudad de Culiacán, será modelado utilizando un diccionario en Python, el cual contendrá la información de la propia red y se podrá accesar de una forma rápida y sencilla cuando se busque la información de alguno de sus vértices. A continuación se describe la información contenida en él.

1.- Las claves del diccionario son los vértices del grafo.

2.- Cada una de estas claves contienen otras con las cuales se tiene una aristas.

3.- Cada arista contendrá un arreglo con: el identificador de la arista que une los dos vértices, la longitud de la arista, su velocidad máxima, número de carriles, número de vehículos, valor de congestión de la arista (proporcionada por la función de congestión Sección 4.1.4).

Los motivos para utilizar un diccionario es la facilidad que se tiene al práctico acceso a la información por medio de claves, y su fácil administración. El uso dinámico de la memoria para agregar o eliminar registros son transparentes y se realizan de manera eficiente.

Apéndice B

Instalación y configuración de un ambiente en SUMO en sistema operativo Ubuntu

La instalación y configuración de un ambiente en el simulador de rutas SUMO, se puntualizan a continuación:

1. Instalar SUMO.
    - *sudo apt-get install sumo sumo-doc sumo-tools*
2. Configuración y creación de un ambiente.
    - *sumo (versión de sumo).*

    2.1 Descargar el archivo sumo-src-0.30.0.tar.gz desde la página oficial de SUMO *(www.sumo.dlr.de/index.html) (versión puede variar).*

    2.2 Crear carpeta de trabajo para el mapa a trabajar.

    2.3 Exportar mapa desde OpenStreetMap *(www.openstreemap.org).*

    2.4 Convertir el archivo del mapa exportado en formato OSM a XML. *netconvert –osm-files nombre.osm -o nombre.map.xml.*

2.5 Crear archivo con rutas y vehículos.

2.6 Crear archivo de polygonos.

2.7 Crear archivo de configuración para SUMO. Ver B.1.

3. Ejecutar aplicación SUMO con el archivo de configuración.

   - *sumo-gui nombre.sumo.cfg*

Tabla B.1: Descripción del archivo de configuración para SUMO.

```
1   <configuration>
2   <input>
3       <net-file value="map.net.xml"/>
4       <route-files value="map.rou.xml"/>
5       <additional-files value="map.poly.xml"/>
6   </input>
7   <time>
8   <begin value="0"/>
9   <end value="100"/>
10  <step-length value="0.1"/>
11  </time>
12  </configuration>
```

Se espera obtener un conjunto de rutas Ec.B.1, una para cada jugador. Las cuáles serán la entrada al simulador de rutas SUMO.

$$R = r_1, r_2, ..., r_N \tag{B.1}$$

Donde:
R es un conjunto de rutas.
r_i es la i-esima ruta.

Apéndice C

Canal de comunicación Python - SUMO

Se requiere crear un canal de comunicación entre Python y SUMO, y será posible a través de una librería llamada TraCi que permite la comunicación entre ambas aplicaciones. En Python se realizará lo que corresponde a la optimización y en SUMO la simulación.

El código para crear el canal de comunicación con SUMO desde Python es la Tabla C.1:

Tabla C.1: Conexión Python y SUMO.

```
1 traci.start([sumoBinary, "-c","map.sumo.cfg",
2 "--tripinfo-output","tripinfo.xml"])
3 step=0
4 while traci.simulation.getMinExpectedNumber()>0;
5       nume=traci.simulation.getMinExpectedNumber()
6       traci.simulationStep()
7       step+=1
8 traci.close(False)
```

Apéndice D

Resultados de las simulaciones en los diferentes escenarios.

El desglose del conjunto de resultados obtenidos en los diferentes escenarios se muestran en este apartado. Se realizaron 50 simulaciones en cada escenario.

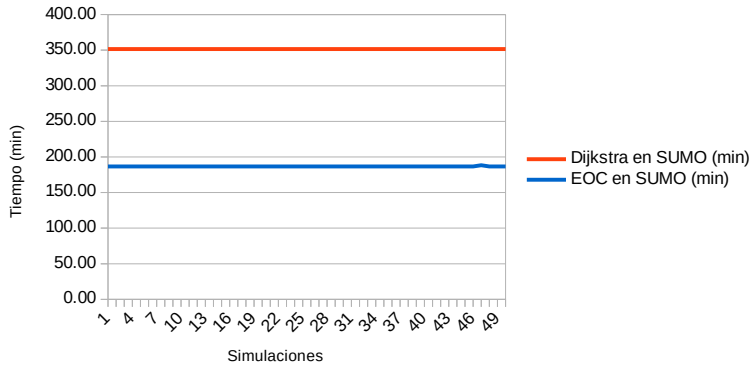
Escenario 1: 0% de carga vehicular.

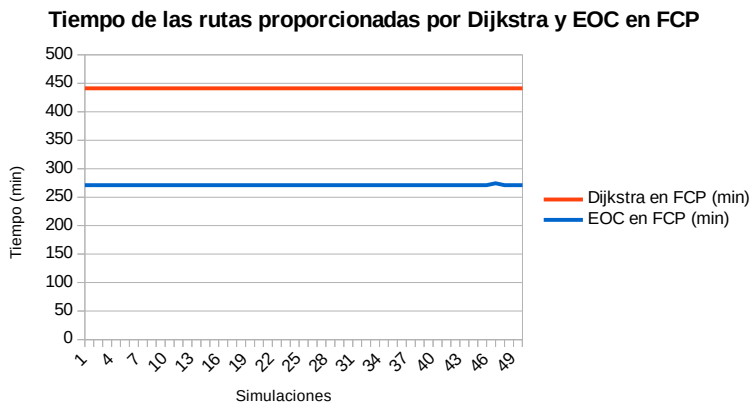
Viajes	100
Ciclos BRD	1000
Congestión en calles (%)	0
Cap. Max. Veh. calles	887
Tiempo de ejecución	20m46.557s

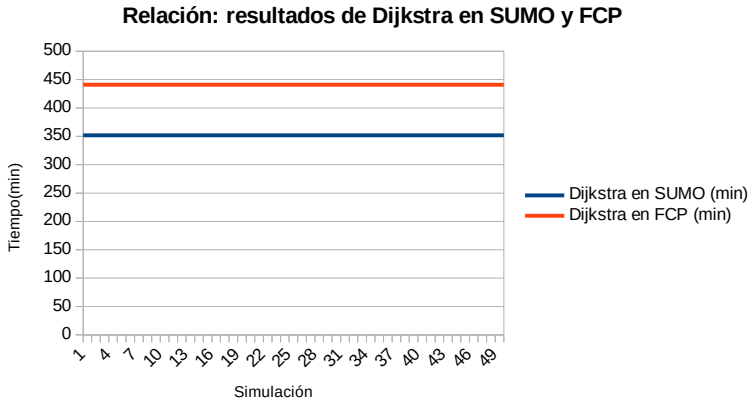
Abreviaciones				
BRD	Best Response Dynamics			
EOC	Estrategia de optimización colectiva			
FCP	Función de congestion propuesta			
SUMO	Simulation of	Urban Mobility		

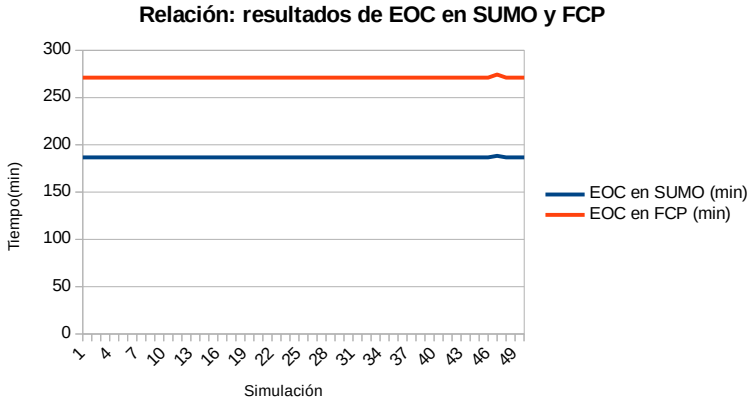
	Vehiculos circulando	Dijkstra en SUMO (min)	EOC en SUMO (min)	Dijkstra en FCP (min)	EOC en FCP (min)	Diferencia (%) en SUMO	Diferencia (%) en FCP
Corrida 1	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 2	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 3	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 4	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 5	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 6	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 7	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 8	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 9	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 10	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 11	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 12	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 13	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 14	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 15	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 16	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 17	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 18	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 19	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 20	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 21	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 22	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 23	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 24	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 25	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 26	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 27	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 28	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 29	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 30	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 31	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 32	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 33	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 34	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 35	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 36	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 37	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 38	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 39	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 40	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 41	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 42	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 43	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 44	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 45	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 46	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 47	0	351.68	188.31	441	274.41	46.45	37.78
Corrida 48	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 49	0	351.68	186.76	441	271.01	46.89	38.55
Corrida 50	0	351.68	186.76	441	271.01	46.89	38.55
PROMEDIO	**0**	**351.7**	**186.8**	**441.0**	**271.1**	**46.89**	**38.53**

Tiempo de las rutas proporcionadas por Dijkstra y EOC en SUMO




Tiempo de las rutas proporcionadas por Dijkstra y EOC en FCP
Tiempo (min)
500
450
400
350
300
250
200
150
100
50
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49
Simulaciones
Dijkstra en FCP (min)
EOC en FCP (min)


Relación: resultados de Dijkstra en SUMO y FCP
Tiempo(min)
500
450
400
350
300
250
200
150
100
50
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49
Simulación
Dijkstra en SUMO (min)
Dijkstra en FCP (min)


Relación: resultados de EOC en SUMO y FCP
Tiempo(min)
300
250
200
150
100
50
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49
Simulación
EOC en SUMO (min)
EOC en FCP (min)

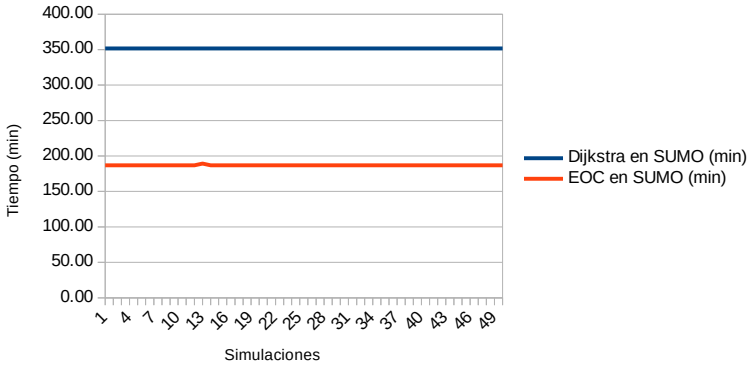
Escenario 2: Del 0% al 25% de carga vehicular.

Viajes	100
Ciclos BRD	1000
Congestión en calles (%)	Del 0 al 25
Cap. Max. Veh. calles	887
Tiempo de ejecución	20m46.607s

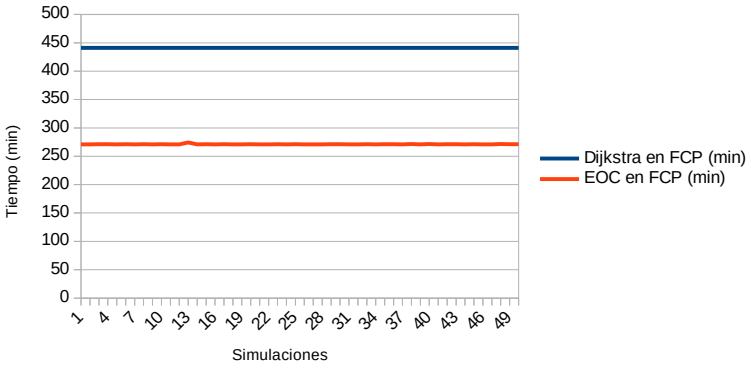
Abreviaciones	
BRD	Best Response Dynamics
EOC	Estrategia de optimización colectiva
FCP	Función de congestión propuesta
SUMO	Simulation of Urban Mobility

	Vehiculos circulando	Dijkstra en SUMO (min)	EOC en SUMO (min)	Dijkstra en FCP (min)	EOC en FCP (min)	Diferencia (%) en SUMO	Diferencia (%) en FCP
Corrida 1	104	351.68	186.76	441	271.02	46.89	38.54
Corrida 2	123	351.68	186.76	441	271.03	46.89	38.54
Corrida 3	91	351.68	186.76	441	271.25	46.89	38.49
Corrida 4	72	351.68	186.76	441	271.17	46.89	38.51
Corrida 5	51	351.68	186.76	441	271.03	46.89	38.54
Corrida 6	88	351.68	186.76	441	271.47	46.89	38.44
Corrida 7	74	351.68	186.76	441	271.01	46.89	38.55
Corrida 8	89	351.68	186.76	441	271.17	46.89	38.51
Corrida 9	106	351.68	186.76	441	271.02	46.89	38.54
Corrida 10	130	351.68	186.76	441	271.23	46.89	38.50
Corrida 11	90	351.68	186.76	441	271.03	46.89	38.54
Corrida 12	91	351.68	186.76	441	271.04	46.89	38.54
Corrida 13	85	351.68	189.44	441	274.45	46.13	37.77
Corrida 14	95	351.68	186.76	441	271.08	46.89	38.53
Corrida 15	119	351.68	186.76	441	271.34	46.89	38.47
Corrida 16	89	351.68	186.76	441	271.02	46.89	38.54
Corrida 17	108	351.68	186.76	441	271.47	46.89	38.44
Corrida 18	87	351.68	186.76	441	271.10	46.89	38.53
Corrida 19	105	351.68	186.76	441	271.02	46.89	38.54
Corrida 20	110	351.68	186.76	441	271.18	46.89	38.51
Corrida 21	97	351.68	186.76	441	271.03	46.89	38.54
Corrida 22	106	351.68	186.76	441	271.10	46.89	38.53
Corrida 23	115	351.68	186.76	441	271.17	46.89	38.51
Corrida 24	98	351.68	186.76	441	271.03	46.89	38.54
Corrida 25	88	351.68	186.76	441	271.24	46.89	38.49
Corrida 26	78	351.68	186.76	441	271.09	46.89	38.53
Corrida 27	91	351.68	186.76	441	271.03	46.89	38.54
Corrida 28	82	351.68	186.76	441	271.03	46.89	38.54
Corrida 29	116	351.68	186.76	441	271.45	46.89	38.45
Corrida 30	119	351.68	186.76	441	271.34	46.89	38.47
Corrida 31	84	351.68	186.76	441	271.03	46.89	38.54
Corrida 32	101	351.68	186.76	441	271.02	46.89	38.54
Corrida 33	83	351.68	186.76	441	271.16	46.89	38.51
Corrida 34	59	351.68	186.76	441	271.04	46.89	38.54
Corrida 35	119	351.68	186.76	441	271.20	46.89	38.50
Corrida 36	128	351.68	186.76	441	271.15	46.89	38.51
Corrida 37	88	351.68	186.76	441	271.03	46.89	38.54
Corrida 38	126	351.68	186.76	441	271.50	46.89	38.44
Corrida 39	104	351.68	186.76	441	271.06	46.89	38.54
Corrida 40	122	351.68	186.76	441	271.50	46.89	38.44
Corrida 41	74	351.68	186.76	441	271.05	46.89	38.54
Corrida 42	121	351.68	186.76	441	271.43	46.89	38.45
Corrida 43	86	351.68	186.76	441	271.21	46.89	38.50
Corrida 44	89	351.68	186.76	441	271.06	46.89	38.54
Corrida 45	131	351.68	186.76	441	271.23	46.89	38.50
Corrida 46	90	351.68	186.76	441	271.04	46.89	38.54
Corrida 47	124	351.68	186.76	441	271.07	46.89	38.53
Corrida 48	101	351.68	186.76	441	271.50	46.89	38.44
Corrida 49	123	351.68	186.76	441	271.18	46.89	38.51
Corrida 50	109	351.68	186.76	441	271.20	46.89	38.50
PROMEDIO	**99**	**351.7**	**186.8**	**441.0**	**271.22**	**46.88**	**38.50**

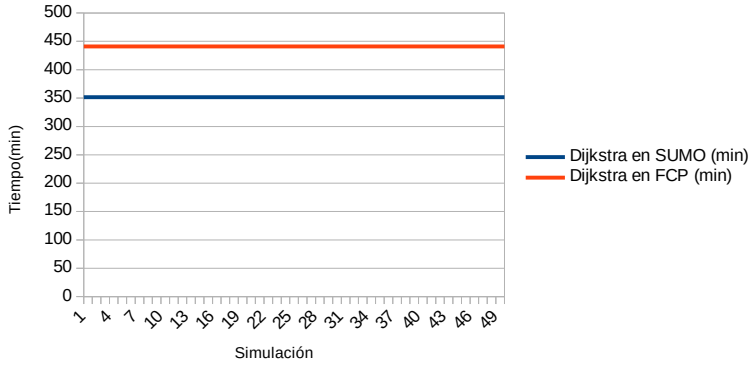
Tiempo de las rutas proporcionadas por Dijkstra y EOC en SUMO



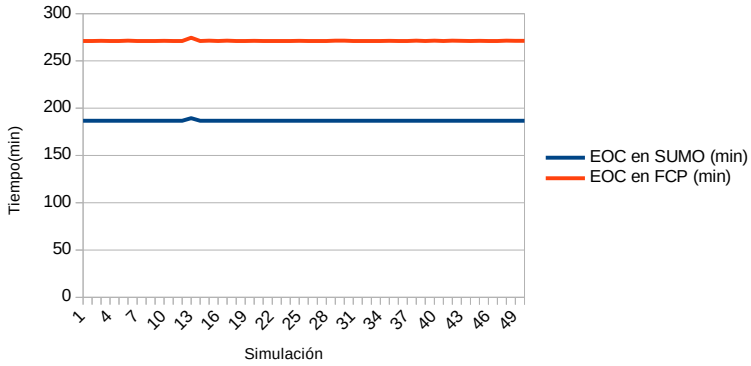
Tiempo de las rutas proporcionadas por Dijkstra y EOC en FCP



Relación: resultados de Dijkstra en SUMO y FCP



Relación: resultados de EOC en SUMO y FCP



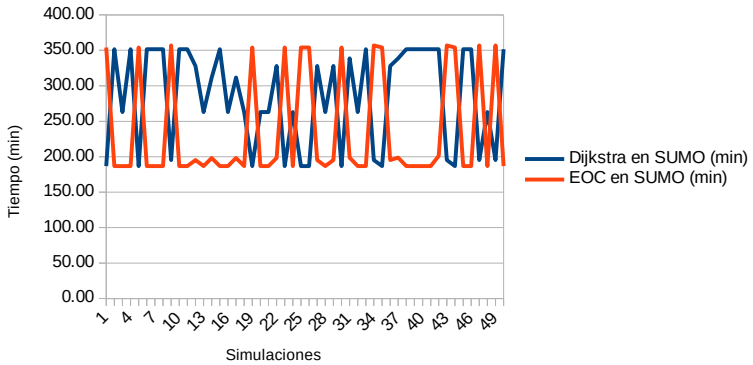
Escenario 3: Del 25% al 50% de carga vehicular.

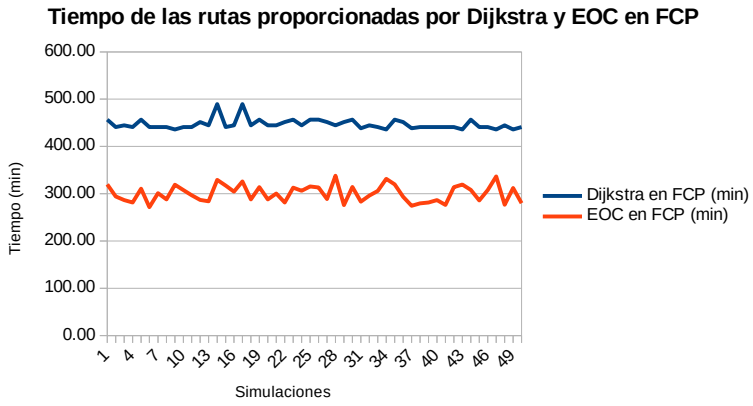
Viajes	100
Ciclos BRD	1000
Congestión en calles (%)	Del 25 al 50
Cap. Max. Veh. calles	887
Tiempo de ejecución	22m33.228s

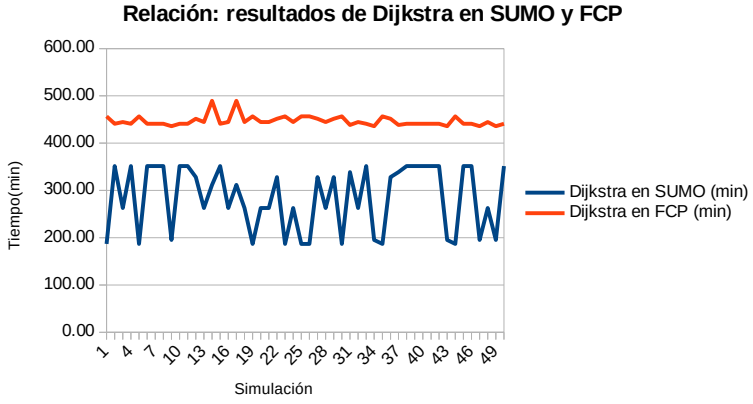
Abreviaciones	
BRD	Best Response Dynamics
EOC	Estrategia de optimización colectiva
FCP	Función de congestión propuesta
SUMO	Simulation of Urban Mobility

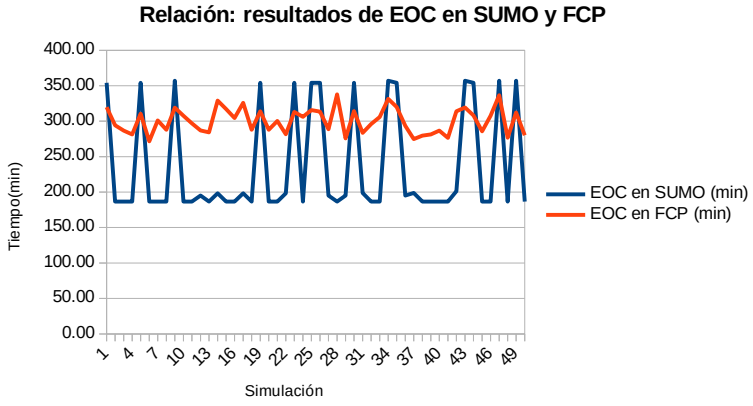
	Vehiculos circulando	Dijkstra en SUMO (min)	EOC en SUMO (min)	Dijkstra en FCP (min)	EOC en FCP (min)	Diferencia (%) en SUMO	Diferencia (%) en FCP
Corrida 1	309	186.76	353.99	456.60	319.79	-89.54	29.96
Corrida 2	291	351.68	186.76	441.00	294.23	46.89	33.28
Corrida 3	307	262.93	186.76	444.66	286.73	28.97	35.52
Corrida 4	291	351.68	186.76	441.00	281.42	46.89	36.19
Corrida 5	320	186.76	353.99	456.60	310.58	-89.54	31.98
Corrida 6	293	351.68	186.76	441.00	271.59	46.89	38.41
Corrida 7	346	351.68	186.76	441.00	301.04	46.89	31.74
Corrida 8	326	351.68	186.76	441.00	287.98	46.89	34.70
Corrida 9	317	195.31	356.89	435.94	319.18	-82.73	26.78
Corrida 10	336	351.68	186.76	441.00	307.60	46.89	30.25
Corrida 11	331	351.68	186.76	441.00	296.98	46.89	32.66
Corrida 12	315	328.06	195.31	451.54	286.91	40.47	36.46
Corrida 13	316	262.93	186.76	444.66	284.23	28.97	36.08
Corrida 14	299	311.71	198.38	489.56	329.09	36.36	32.78
Corrida 15	335	351.68	186.76	441.00	317.13	46.89	28.09
Corrida 16	295	262.93	186.76	444.66	304.38	28.97	31.55
Corrida 17	287	311.71	198.38	489.56	325.96	36.36	33.42
Corrida 18	303	262.93	186.76	444.66	287.98	28.97	35.24
Corrida 19	321	186.76	353.99	456.60	313.93	-89.54	31.25
Corrida 20	296	262.93	186.76	444.66	287.98	28.97	35.24
Corrida 21	337	262.93	186.76	444.66	300.32	28.97	32.46
Corrida 22	328	328.06	198.30	451.54	281.53	39.55	37.65
Corrida 23	299	186.76	353.99	456.60	312.95	-89.54	31.46
Corrida 24	303	262.93	186.76	444.66	306.18	28.97	31.14
Corrida 25	315	186.76	353.99	456.60	315.57	-89.54	30.89
Corrida 26	314	186.76	353.99	456.60	313.36	-89.54	31.37
Corrida 27	318	328.06	195.31	451.54	288.68	40.47	36.07
Corrida 28	301	262.93	186.76	444.66	337.91	28.97	24.01
Corrida 29	311	328.06	195.31	451.54	275.82	40.47	38.92
Corrida 30	271	186.76	353.99	456.60	314.27	-89.54	31.17
Corrida 31	316	338.77	198.79	438.63	283.42	41.32	35.39
Corrida 32	297	262.93	186.76	444.66	295.99	28.97	33.43
Corrida 33	304	351.68	186.76	441.00	305.93	46.89	30.63
Corrida 34	315	195.31	356.89	435.94	331.60	-82.73	23.93
Corrida 35	319	186.76	353.99	456.60	319.92	-89.54	29.94
Corrida 36	339	328.06	195.31	451.54	293.53	40.47	34.99
Corrida 37	326	338.77	198.79	438.63	274.74	41.32	37.36
Corrida 38	314	351.68	186.76	441.00	279.59	46.89	36.60
Corrida 39	321	351.68	186.76	441.00	281.42	46.89	36.19
Corrida 40	276	351.68	186.76	441.00	286.71	46.89	34.99
Corrida 41	275	351.68	186.76	441.00	276.48	46.89	37.31
Corrida 42	324	351.68	201.19	441.00	313.90	42.79	28.82
Corrida 43	327	195.31	356.89	435.94	319.31	-82.73	26.75
Corrida 44	314	186.76	353.99	456.60	308.01	-89.54	32.54
Corrida 45	312	351.68	186.76	441.00	285.78	46.89	35.20
Corrida 46	343	351.68	186.76	441.00	307.60	46.89	30.25
Corrida 47	304	195.31	356.89	435.94	336.56	-82.73	22.80
Corrida 48	323	262.93	186.76	444.66	276.76	28.97	37.76
Corrida 49	317	195.31	356.89	435.94	312.22	-82.73	28.38
Corrida 50	338	351.68	186.76	441.00	280.17	46.89	36.47
PROMEDIO	**313**	**284.13**	**236.02**	**446.94**	**300.62**	**16.94**	**32.73**

Tiempo de las rutas proporcionadas por Dijkstra y EOC en SUMO




Tiempo de las rutas proporcionadas por Dijkstra y EOC en FCP
600.00
500.00
400.00
300.00
200.00
100.00
0.00
Tiempo (min)
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49
Simulaciones
Dijkstra en FCP (min)
EOC en FCP (min)


Relación: resultados de Dijkstra en SUMO y FCP
600.00
500.00
400.00
300.00
200.00
100.00
0.00
Tiempo(min)
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49
Simulación
Dijkstra en SUMO (min)
Dijkstra en FCP (min)


Relación: resultados de EOC en SUMO y FCP
400.00
350.00
300.00
250.00
200.00
150.00
100.00
50.00
0.00
Tiempo(min)
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49
Simulación
EOC en SUMO (min)
EOC en FCP (min)

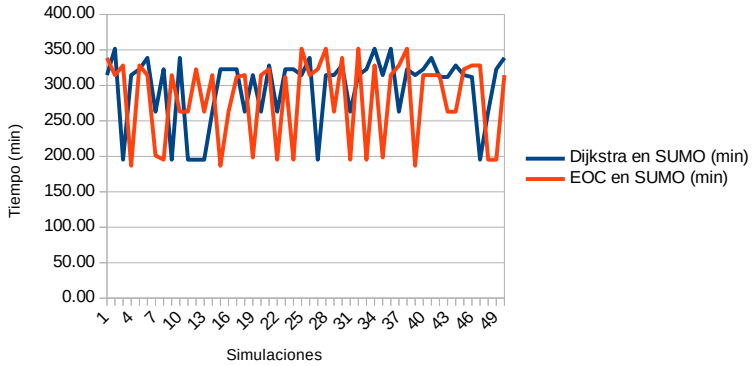
Escenario 4: Del 50% al 75% de carga vehicular.

Viajes	100
Ciclos BRD	1000
Congestión en calles (%)	Del 50 al 75
Cap. Max. Veh. calles	887
Tiempo de ejecución	26m34.305s

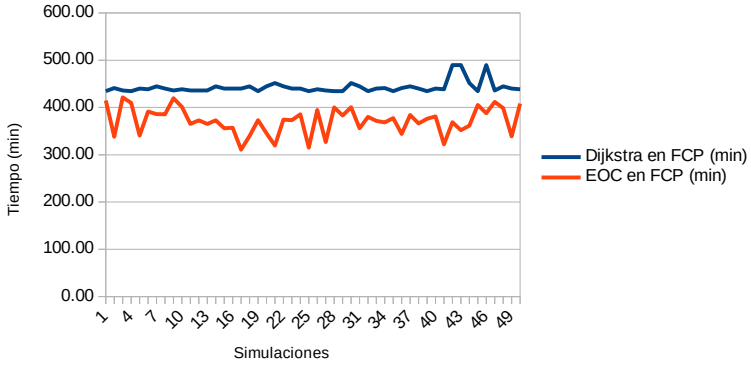
Abreviaciones	
BRD	Best Response Dynamics
EOC	Estrategia de optimización colectiva
FCP	Función objetivo propuesta
SUMO	Simulation of Urban Mobility

	Vehiculos circulando	Dijkstra en SUMO (min)	EOC en SUMO (min)	Dijkstra en FCP (min)	EOC en FCP (min)	Diferencia (%) en SUMO	Diferencia (%) en FCP
Corrida 1	545	314.47	338.77	434.54	414.68	-7.72	4.57
Corrida 2	546	351.68	314.47	441.00	338.25	10.58	23.30
Corrida 3	544	195.31	328.06	435.94	421.39	-67.97	3.34
Corrida 4	560	314.47	186.76	434.54	409.65	40.61	5.73
Corrida 5	542	322.93	328.06	439.93	340.53	-1.59	22.59
Corrida 6	514	338.77	314.47	438.63	391.52	7.17	10.74
Corrida 7	531	262.93	201.19	444.66	385.86	23.48	13.22
Corrida 8	529	322.93	195.31	439.93	385.38	39.52	12.40
Corrida 9	543	195.31	314.47	435.94	419.37	-61.02	3.80
Corrida 10	534	338.77	262.93	438.63	401.38	22.38	8.49
Corrida 11	530	195.31	262.93	435.94	365.23	-34.63	16.22
Corrida 12	491	195.31	322.93	435.94	372.87	-65.35	14.47
Corrida 13	501	195.31	262.93	435.94	365.23	-34.63	16.22
Corrida 14	525	262.93	314.47	444.66	372.71	-19.60	16.18
Corrida 15	538	322.93	186.76	439.93	355.84	42.17	19.11
Corrida 16	537	322.93	262.93	439.93	357.09	18.58	18.83
Corrida 17	528	322.93	311.71	439.93	310.68	3.48	29.38
Corrida 18	560	262.93	314.47	444.66	339.96	-19.60	23.55
Corrida 19	551	314.47	198.30	434.54	373.31	36.94	14.09
Corrida 20	548	262.93	314.47	444.66	345.59	-19.60	22.28
Corrida 21	531	328.06	322.93	451.54	319.37	1.56	29.27
Corrida 22	540	262.93	195.31	444.66	374.26	25.72	15.83
Corrida 23	514	322.93	311.71	439.93	373.23	3.48	15.16
Corrida 24	564	322.93	195.31	439.93	385.38	39.52	12.40
Corrida 25	514	314.47	351.68	434.54	314.94	-11.83	27.52
Corrida 26	549	338.77	314.47	438.63	394.75	7.17	10.00
Corrida 27	536	195.31	322.93	435.94	326.83	-65.35	25.03
Corrida 28	580	314.47	351.68	434.54	400.23	-11.83	7.90
Corrida 29	547	314.47	262.93	434.54	383.13	16.39	11.83
Corrida 30	551	328.06	338.77	451.54	400.45	-3.26	11.31
Corrida 31	528	262.93	195.31	444.66	355.92	25.72	19.96
Corrida 32	528	314.47	351.68	434.54	380.15	-11.83	12.52
Corrida 33	547	322.93	195.31	439.93	371.92	39.52	15.46
Corrida 34	529	351.68	328.06	441.00	368.47	6.72	16.45
Corrida 35	505	314.47	198.30	434.54	377.47	36.94	13.13
Corrida 36	517	351.68	314.47	441.00	343.88	10.58	22.02
Corrida 37	520	262.93	328.06	444.66	384.19	-24.77	13.60
Corrida 38	547	322.93	351.68	439.93	366.33	-8.90	16.73
Corrida 39	520	314.47	186.76	434.54	376.13	40.61	13.44
Corrida 40	523	322.93	314.47	439.93	381.14	2.62	13.36
Corrida 41	524	338.77	314.47	438.63	322.13	7.17	26.56
Corrida 42	522	311.71	314.47	489.56	368.62	-0.89	24.70
Corrida 43	509	311.71	262.93	489.56	351.77	15.65	28.15
Corrida 44	576	328.06	262.93	451.54	361.28	19.85	19.99
Corrida 45	576	314.47	322.93	434.54	405.28	-2.69	6.74
Corrida 46	530	311.71	328.06	489.56	387.56	-5.25	20.84
Corrida 47	531	195.31	328.06	435.94	411.70	-67.97	5.56
Corrida 48	544	262.93	195.31	444.66	398.77	25.72	10.32
Corrida 49	529	322.93	195.31	439.93	338.76	39.52	23.00
Corrida 50	520	338.77	314.47	438.63	408.22	7.17	6.93
PROMEDIO	**535**	**296.7**	**282.1**	**442.66**	**371.97**	**4.92**	**15.97**

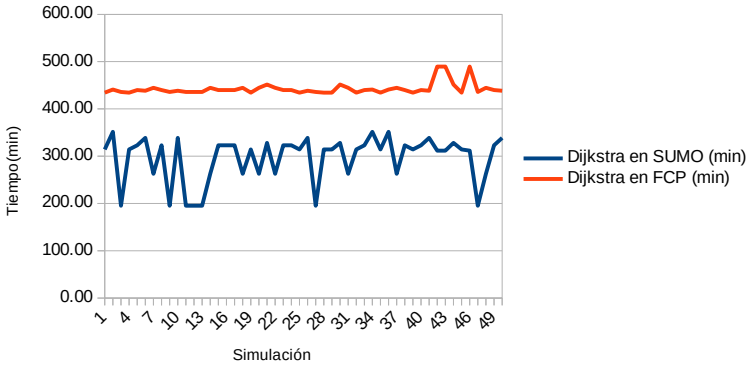
Tiempo de las rutas proporcionadas por Dijkstra y EOC en SUMO



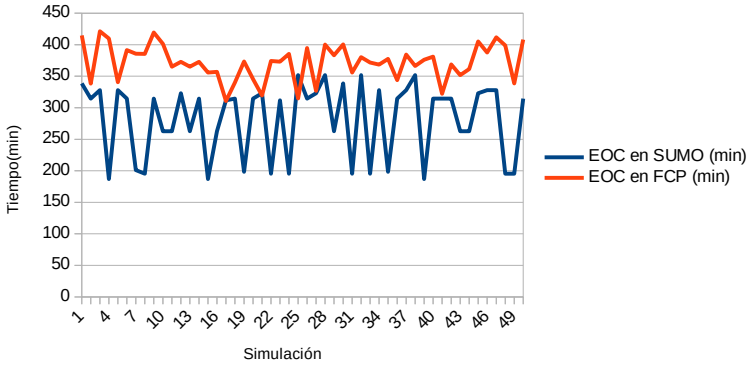
Tiempo de las rutas proporcionadas por Dijkstra y EOC en FCP



Relación: resultados de Dijkstra en SUMO y FCP



Relación: resultados de EOC en SUMO y FCP



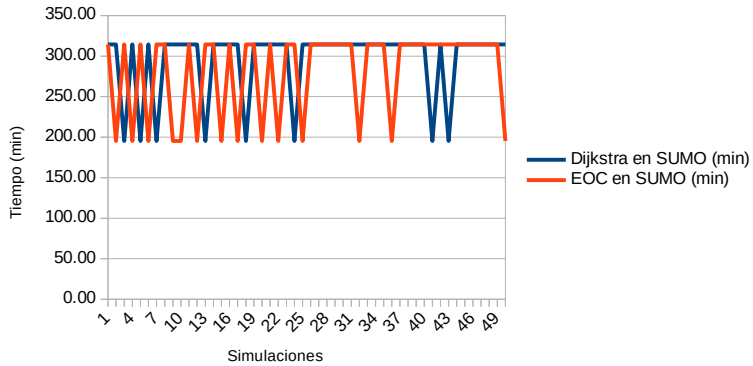
Escenario 5: Del 75% al 100% de carga vehicular.

Viajes	100
Ciclos BRD	1000
Congestión en calles (%)	Del 75 al 100
Cap. Max. Veh. calles	887
Tiempo de ejecución	28m34.578s

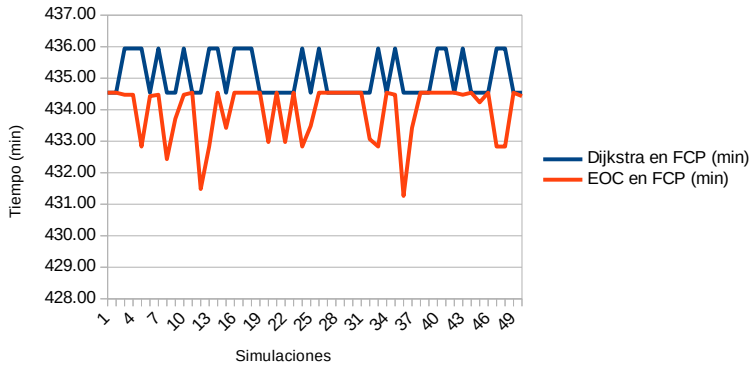
Abreviaciones	
BRD	Best Response Dynamics
EOC	Estrategia de optimización colectiva
FCP	Función de congestión propuesta
SUMO	Simulation of Urban Mobility

	Vehiculos circulando	Dijkstra en SUMO (min)	EOC en SUMO (min)	Dijkstra en FCP (min)	EOC en FCP (min)	Diferencia (%) en SUMO	Diferencia (%) en FCP
Corrida 1	752	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 2	765	314.47	195.31	434.54	434.54	37.89	0.00
Corrida 3	716	195.31	314.47	435.94	434.48	-61.02	0.34
Corrida 4	752	314.47	195.31	435.94	434.48	37.89	0.34
Corrida 5	746	195.31	314.47	435.94	432.83	-61.02	0.71
Corrida 6	766	314.47	195.31	434.54	434.43	37.89	0.02
Corrida 7	744	195.31	314.47	435.94	434.48	-61.02	0.34
Corrida 8	735	314.47	314.47	434.54	432.43	0.00	0.49
Corrida 9	771	314.47	195.31	434.54	433.72	37.89	0.19
Corrida 10	743	314.47	195.31	435.94	434.48	37.89	0.34
Corrida 11	742	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 12	772	314.47	195.31	434.54	431.48	37.89	0.70
Corrida 13	761	195.31	314.47	435.94	432.83	-61.02	0.71
Corrida 14	761	314.47	314.47	435.94	434.54	0.00	0.32
Corrida 15	742	314.47	195.31	434.54	433.42	37.89	0.26
Corrida 16	743	314.47	314.47	435.94	434.54	0.00	0.32
Corrida 17	712	314.47	195.31	435.94	434.54	37.89	0.32
Corrida 18	783	195.31	314.47	435.94	434.54	-61.02	0.32
Corrida 19	745	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 20	768	314.47	195.31	434.54	432.98	37.89	0.36
Corrida 21	744	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 22	777	314.47	195.31	434.54	432.98	37.89	0.36
Corrida 23	779	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 24	736	195.31	314.47	435.94	432.83	-61.02	0.71
Corrida 25	755	314.47	195.31	434.54	433.48	37.89	0.24
Corrida 26	733	314.47	314.47	435.94	434.54	0.00	0.32
Corrida 27	758	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 28	732	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 29	729	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 30	785	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 31	770	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 32	762	314.47	195.31	434.54	433.07	37.89	0.34
Corrida 33	757	314.47	314.47	435.94	432.83	0.00	0.71
Corrida 34	749	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 35	740	314.47	314.47	435.94	434.48	0.00	0.34
Corrida 36	745	314.47	195.31	434.54	431.27	37.89	0.75
Corrida 37	756	314.47	314.47	434.54	433.42	0.00	0.26
Corrida 38	749	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 39	750	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 40	776	314.47	314.47	435.94	434.54	0.00	0.32
Corrida 41	768	195.31	314.47	435.94	434.54	-61.02	0.32
Corrida 42	748	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 43	791	195.31	314.47	435.94	434.48	-61.02	0.34
Corrida 44	754	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 45	724	314.47	314.47	434.54	434.23	0.00	0.07
Corrida 46	777	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 47	770	314.47	314.47	435.94	432.83	0.00	0.71
Corrida 48	773	314.47	314.47	435.94	432.83	0.00	0.71
Corrida 49	768	314.47	314.47	434.54	434.54	0.00	0.00
Corrida 50	745	314.47	195.31	434.54	434.43	37.89	0.02
PROMEDIO	**754**	**295.41**	**281.11**	**435.07**	**433.97**	**4.84**	**0.25**

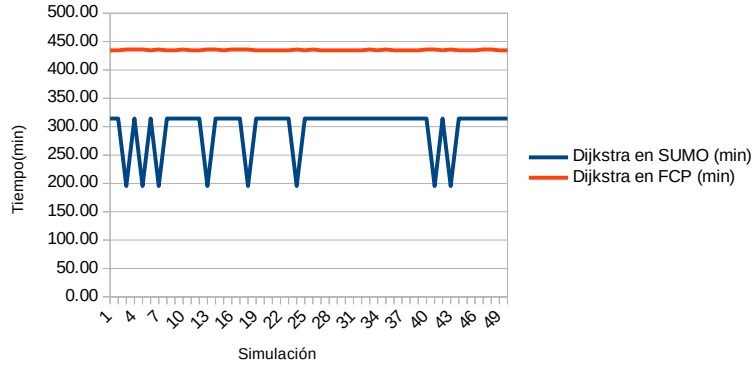
Tiempo de las rutas proporcionadas por Dijkstra y EOC en SUMO



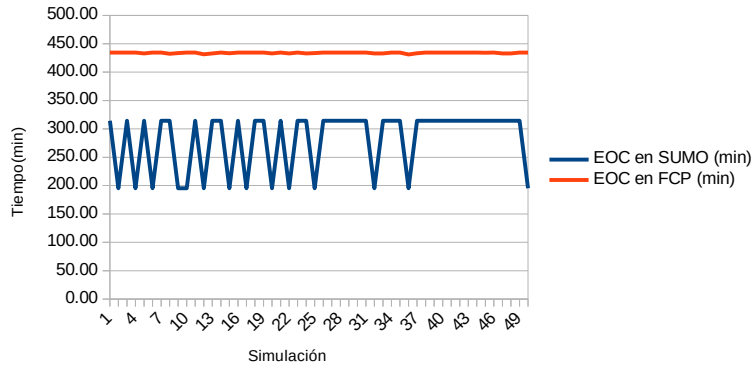
Tiempo de las rutas proporcionadas por Dijkstra y EOC en FCP



Relación: resultados de Dijkstra en SUMO y FCP



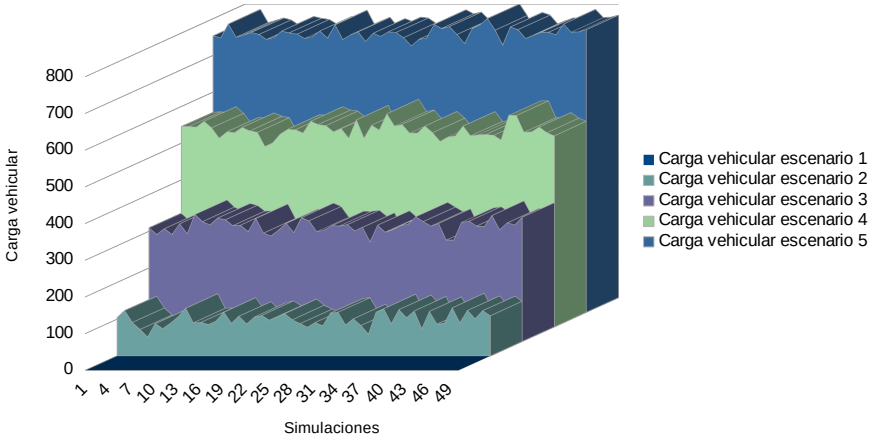
Relación: resultados de EOC en SUMO y FCP



Carga vehicular de los diferentes escenarios

Carga vehicular escenario 1	Carga vehicular escenario 2	Carga vehicular escenario 3	Carga vehicular escenario 4	Carga vehicular escenario 5
0	104	309	545	753
0	123	291	546	748
0	91	307	544	785
0	72	291	560	749
0	51	320	542	755
0	88	293	514	762
0	74	346	531	757
0	89	326	529	743
0	106	317	543	749
0	130	336	534	766
0	90	331	530	761
0	91	315	491	758
0	85	316	501	746
0	95	299	525	751
0	119	335	538	766
0	89	295	537	747
0	108	287	528	783
0	87	303	560	743
0	105	321	551	755
0	110	296	548	764
0	97	337	531	738
0	106	328	540	760
0	115	299	514	752
0	98	303	564	764
0	88	315	514	762
0	78	314	549	748
0	91	318	536	726
0	82	301	580	743
0	116	311	547	747
0	119	271	551	784
0	84	316	528	773
0	101	297	528	778
0	83	304	547	757
0	59	315	529	732
0	119	319	505	773
0	128	339	517	783
0	88	326	520	800
0	126	314	547	769
0	104	321	520	728
0	122	276	523	777
0	74	275	524	770
0	121	324	522	746
0	86	327	509	754
0	89	314	576	762
0	131	312	576	762
0	90	343	530	756
0	124	304	531	785
0	101	323	544	763
0	123	317	529	764
0	109	338	520	770

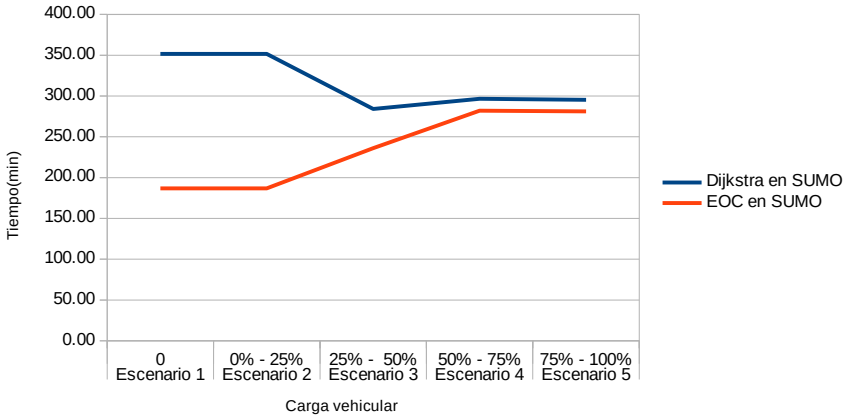
Carga vehicular de los escenarios.



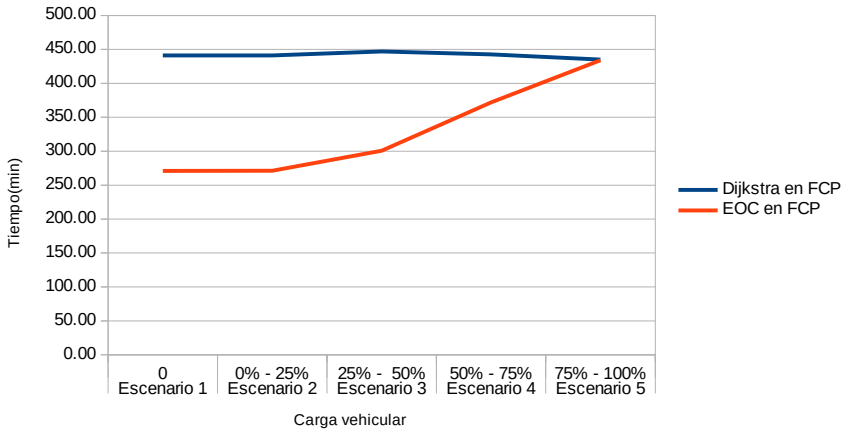
Resultados de tiempo promedio de los escenarios con relación a su carga vehicular

	Carga vehicular	Dijkstra en SUMO	Dijkstra en FCP	EOC en SUMO	EOC en FCP
Escenario 1	0%	351.70	441.00	186.80	271.10
Escenario 2	0% - 25%	351.70	441.00	186.80	271.22
Escenario 3	25% - 50%	284.13	446.94	236.02	300.62
Escenario 4	50% - 75%	296.70	442.66	282.10	371.97
Escenario 5	75% - 100%	295.41	435.07	281.11	433.97

Tiempo promedio de las simulaciones de Dijkstra y EOC en SUMO



Tiempo promedio de las simulaciones de Dijkstra y EOC en FCP



Bibliografía

[1] Álvares de la Torre . Estructura urbana de las ciudades intermedias en méxico. frontera norte. *scielo*, págs. 91–124, 2011.

[2] Bondy , J.A y Murty U.S.R. *Graph Theory with Applications*. North-Holland, 1976.

[3] Cormen , Tomas H., Leiserson Charles E., Rivest Ronald L., y Stein Clifford. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2da. ed$^{\underline{\text{ón}}}$., 2001. Section 24., "Single-Source Shortest Paths".

[4] Cormen , Tomas H., Leiserson Charles E., Rivest Ronald L., y Stein Clifford. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2da. ed$^{\underline{\text{ón}}}$., 2001. Section 25., "All-Pairs Shortest Paths".

[5] Costa Salas , José, Abreu Ledón René, Machado Isaías, Coello, y Nowé Ann. Multi-type ant colony system for solving the multiple traveling salesman problem. *Rev. Téc. Ing. Univ. Zulia.*, 2012.

[6] Dijkstra , Edsger W. A note on two problems in connexion with graphs. *Numerische mathematik 1*, págs. 269–271, 1959.

[7] Gibbons , R. *A primer in game theory*. Harvester Wheatsheaf, 1992.

[8] Godsil , C. y Royle G.F. *Algebraic graph theory*. Springer Science & Business Media, 2001.

[9] Hochbaum , Dorit. Lectura notes for ieor 266: Graph algorithms and network flows. 2014. Section 3., "Minimun cost network flow problem", pages: 8-9.

[10] Hochbaum , Dorit. Lectura notes for ieor 266: Graph algorithms and network flows. 2014. Section 8., "Shortest paths", pages: 41-42.

[11] Kleinberg , Jon y Tardos Éva. *Algorithm Design.* PERSON, 2006.

[12] Lanning , Daniel R. Dijkstra's algorithm and google maps. 2014.

[13] Milton , R, Aldás S, Marco J, y Flores C. Modelo origen destino para estimar el flujo de tráfico usando algoritmos genéticos. *MASKANA, I+D+ingeniería*, 2014.

[14] Myerson , Roger.B. *Game Theory: Analysis of Conflict.* Harvard University press, 1991.

[15] Nash , J. Non-cooperative games. *Annals of Mathematics*, págs. 286–295, 1951.

[16] Nash , John F. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, págs. 48–49, 1950.

[17] Organización Mundial de la Salud , OMS. Mala calidad del medio ambiente. *OMS*, 2016.

[18] Pereyra , G, Pandolfi D, y Villagra A. Design and optimization of water distribution network using genetic alghoritm. *UNPA*, 2016.

[19] Peña Morales , David. *Optimización multiobjetivo para la planificación de transporte público aplicando técnicas metaheurísticas.* Proyecto Fin de Carrera, CICESE, 2017.

[20] Python Software Foundation . Herramientas python. Inf. téc., Python Foundation, 2018.

[21] Rojo , Juan C. *El deterioro del espacio público y su impacto en las áreas destinadas a la socialización y al desarrollo de la accesibilidad en las ciudades medias mexicanas. Caso Culiacán, Sinaloa.* Tesis Doctoral, Barcelona University, 2017.

[22] Rosenbush , Steven. At ups, the algorithm is the driver. *The wall street journal*, 2015.

[23] Rossum , Van Guido. El tutorial de python. *The wall street journal*, 2017.

[24] Roughgarden , Tim. Algoritmic game theory lecture 16; best responce dynamics. *Stanford University*, 2013.

[25] Sánchez , Joss. Juegos cooperativos y sus aplicaciones económicas. *Perspectivas. Revista de Análisis de Economía, Comercio y Negocios Internacionales*, págs. 59–60, 2011.

[26] Tadelis , S et al. *Game Theory; an introduction*. Pricenton University Press, 2013.

[27] Thomson , Ian y Bull Alberto. La congestión del tránsito urbano: causas y consecuencias económicas y sociales. *Revista de la Cepal 76*, pág. 110, 2002.

[28] Trudeau , R.J. *Introduction to Graph Theory*. doverpublications, 1993.

[29] Yee Rendón , Arturo. *Selection of Strategies in Complex Games: Baseball, American Football and Go*. Tesis Doctoral, CINVESTAV, 2015.

[30] Bloomberg. Company overview of waze mobile limited, 2018. URL: http://www.bloomberg.com/research/stocks/private/snapshot.asp?privcapld=58203778.

[31] Instituto Municipal de Planeación Urbana de Culiacán. Plan de movilidad diagnóstico. *Plan de desarrollo municipal de Culiacán*, págs. 1–9, 2007.

[32] FHWA. Congestion: A national issue, 2017. Institutción: U.S. Department of Transportation Federal Highway Administration URL: http://ops.fhwa.dot.gov/aboutus/opstory.htm.

[33] IMICO. Índices de competitividad urbana. *Instituto Mexicano de la Competitividad*, 2016.

[34] INEGI. Censo de población, culiacán, sinaloa. Inf. téc., INEGI, 2018.

[35] SLGPS. Construye tus mapas: Openstreetmap y waze. Inf. téc., LBSPro, 2017. Sercicios de Localización, Navegación GPS y Mapas.

[36] WAZE. Waze, routing server, 2016. URL: http://wiki.waze.com/wiki/Routing_server.