Simulación de un agente enrutador de vehículos a través de un sistema basado en tiempos de viaje. Ing. Marco Luis Dorado Pineda 2014

Universidad Autónoma de Querétaro
Facultad de Ingeniería.

Simulación de un agente enrutador de vehículos a través de un sistema basado en tiempos de viaje.

Tesis
Que como parte de los requisitos para obtener el grado de:

Maestro en Ingeniería de Vías Terrestres

Presenta
Marco Luis Dorado Pineda.

Dirigido por:
Dr. Saúl Antonio Obregón Biosca.

C.U. Querétaro, Qro. enero de 2014.

Universidad Autónoma de Querétarc
Facultad de Ingeniería.
Maestría en Ingeniería de Vías Terrestres.

Simulación de un agente enrutador de vehículos a través de un sistema basado en tiempos de viaje.

TESIS

Que como parte de los requisitos para obtener grado de

Maestro en ingeniería de Vías Terrestres.

Presenta:
Marco Luis Dorado Pineda

Dirigido por:
Dr. Saúl Antonio Obregón Biosca.

SINODALES

Dr. Saúl Obregón Biosca
Presidente — Firma

Dr. Roberto de la Llata Gómez
Secretario — Firma

Dr. Guillermo Torres Vargas
Vocal — Firma

Dr. Alejandro Castañeda Miranda
Suplente — Firma

Dr. Eduardo Betanzo Quezada
Suplente — Firma

Dr. Aurelio Domínguez González
Director de la Facultad

Dr. Irineo Torres Pacheco
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Enero 2014
México

RESUMEN

El congestionamiento de las vialidades es un problema crónico que afecta a las ciudades, debido al incremento de automóviles y la migración de personas a estas zonas, ya que son los centros de mayor actividad económica del mundo. El problema no sólo incrementa las demoras en cruceros y vialidades, sino que también impacta a la calidad del aire de la zona urbana. Al no contar con medidas efectivas para mitigar el congestionamiento del tráfico, estas pueden tornarse caóticas, ocasionando el incremento de vehículos y por consiguiente los tiempos de viaje en las avenidas de mayor circulación. El desarrollo de sistemas inteligentes de transporte en las vialidades contribuirá grandemente a evitar que los caminos principales se saturen, informar de algún percance que obstaculice el flujo de automóviles e incluso a interactuar con semáforos para poder optimizar el flujo en los cruces. Se propone la implementación sobre el estado del arte de los sistemas inteligentes de transporte, de un sistema de información de tráfico, que interactúe con las localizaciones físicas de los vehículos, siendo estas enviadas a través de dispositivos dentro de los autos hacia un sistema de redes *Ad-Hoc* vehiculares localizadas en las avenidas principales y secundarias de una ciudad; permitiendo conocer las zonas de saturación, de manera que se pueda orientar a los usuarios que cuenten con la información proporcionada por el sistema sobre las rutas alternas y en la elección del menor tiempo de viaje de las calles por las cuales puedan transitar.

(**Palabras clave**: Simulación ITS, tiempos de viaje, congestionamiento, ATIS.)

SUMMARY

Traffic congestion is a chronic problem affecting cities because of the increase of cars and people migrating to these areas, as they are the centers of world economic activity. The problem not only increases delays on cruises and roads, but also affects the air quality of the urban area. If no effective measures are taken to minimize traffic congestion, it can become chaotic, causing the increase of vehicles and therefore travel times in main avenues. The development of intelligent transport systems in the roads will help fight the saturation of the main roads by reporting any problem that can cause congestion and interacting with traffic lights to optimize the flow of cars. Our proposal is to implement the state of the art of intelligent transportation systems, a traffic information system interacting with the physical locations of vehicles, using a network system connecting devices installed in cars to an Ad-Hoc network system Vehicular located in the main and secondary streets of the city. This system will allow its users to know which zones are saturated and provide them alternative routes, thus reducing travel times.

(**Key words**: Intelligent transport systems, travel time, jams, ATIS.)

Porque de él, y por él,
y para él, son todas las cosas.
A él sea la gloria por los siglos.
Amén.
Romanos 11:36.

AGRADECIMIENTOS

En el arduo esfuerzo que se ve ahora consolidado como trabajo de tesis, quiero agradecer a mi Dios, quien suple en todo momento de pan y sabiduría. A mi director de tesis, el Dr. Saúl Obregón Biosca de la Universidad Autónoma de Querétaro, por haberme brindado su apoyo. A CONACYT por el sustento económico brindado a lo largo de mis estudios. Le agradezco a la familia Pineda Fernández, por su abnegada hospitalidad. A mi madre, hermanas y abuela, mi familia, por su apoyo incondicional en todo momento. A mis compañeros de maestría por su amistad y ayuda. Y finalmente y muy especialmente a Arely, mi esposa, gracias por emprender esta aventura conmigo.

Í N D I C E

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

ÍNDICE DE ABREVIATURAS

AISMUN	Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks.
DRGS	Dynamic Route Guidance Systems.
EDU	Equilibrio Dinámico de Usuario.
FTVFL	Fijo con Tiempos de Viaje a Flujo Libre.
IBM	International Business Machines.
SEV	Sistema Enrutador de Vehículos.
WLAN	Wireless Local Area Network.

1. INTRODUCCIÓN

El tema de congestionamiento del tráfico es un problema que se agrava día con día. Los efectos negativos que se presentan en las grandes ciudades son cada vez más notorios. Si bien no existe un remedio que pueda revertir los efectos generados por los embotellamientos que se presentan de manera frecuente, es posible encontrar soluciones que mejoren las condiciones actuales de algunos puntos de conflicto vehicular.

En particular, se tratarán los objetivos que plantean los Sistemas Inteligentes de Transporte hoy en día, como son los de proveer una política que pueda desatascar los grandes ejes viales por los que circulan a diario miles de personas.

El asunto no es tan fácil, ya que los problemas no solamente son el aumento en la tenencia vehicular, que incrementa el uso del medio y cargando así la capacidad de la infraestructura disponible con el consiguiente conflicto en las vialidades, la sobrepoblación y la mala urbanización de las crecientes ciudades como Querétaro.

Para poder empezar a desentramar los problemas que presenta el congestionamiento de tráfico, comenzaremos por definir y analizar las teorías que han surgido a raíz de los estudios realizados por diferentes autores que se han dado a la tarea de analizar miles de datos generados por el tránsito de vehículos cada día y en diferentes países. Dichos análisis conllevan en la identificación de los agentes desencadenadores de los embotellamientos y los cuales se utilizarán para dar un sustento al presente trabajo.

Se presentarán tecnologías que tienen por objeto el establecer canales de comunicación entre dispositivos incorporados a los vehículos y redes vehiculares, con la finalidad de recabar información en tiempo real del estado de las vialidades.

De igual manera, se utilizarán los más recientes avances que existen en el *software* de simulación de tráfico. Si bien existen diferentes programas computacionales que integran modelos de comportamiento vehicular tanto macroscópico y microscópico, se utilizará el programa AIMSUN versión 7.0 (TSS - *Transport Simulation Systems*, 2011), creado por la Universidad de Catalunya.

Se espera que el presente trabajo sirva de base para futuros proyectos viales que contemplen medidas de mitigación de tráfico, ya que el constante desarrollo de vialidades necesita de un apoyo para su correcta funcionalidad.

1.1 Descripción del problema.

En la mayoría de los países del mundo, las carreteras a menudo se congestionan en sobremanera durante varias horas en el día. Los elementos comunes del tráfico, especialmente los que ocasionan el desajuste y los que propagan los embotellamientos dentro de las redes, se tienen que tomar en cuenta en los modelos de simulación de tráfico y sus aplicaciones (Rehborn *et al*; 2011).

Además de mermar la calidad de vida de los automovilistas que experimentan esto a diario, es común ver los efectos a flor de piel del congestionamiento vial: usuarios que incrementan su agresividad por la desesperación que se va haciendo presente con cada minuto que pasan detrás del volante, incrementando la posibilidad de un percance con algún otro vehículo.

Otro efecto negativo de estas condiciones prevalecientes en las ciudades es el incremento de contaminantes que afectan de manera directa la salud de quienes interactúan con el entorno en conflicto, en este sentido, según Laumbach y Kipen, (2011) es creciente la evidencia que apunta a que la contaminación del aire contribuye a la gran lista de muertes por enfermedades respiratorias y alérgicas que incluyen asma, neumonía y tuberculosis, entre otras. En el caso de México, alrededor de mil personas mueren cada año por culpa de las altas

concentraciones de partículas contaminantes, según el Instituto Nacional de Salud Pública (Riojas *et al*; 2012).

Los costos empleados para tratar dichas enfermedades se pueden estimar como pérdidas, ya que es el gobierno quien tiene que invertir más en los tratamientos utilizados para este tipo de enfermedades en las instituciones sanitarias, los usuarios incrementan su gasto en combustible, los trabajos tienen que pagar por las incapacidades de sus trabajadores y por si fuera poco, el desempeño de los trabajadores después de las horas que pasan en el tráfico es deficiente.

Si bien las soluciones más comunes que se le han dado al problema del congestionamiento radican en construir más caminos y con mayor longitud. Sin embargo, debido a los grandes costos financieros, sociales y medio ambientales de tales proyectos, no es una opción viable. Muchos proyectos e innovaciones deben de emplear el uso eficiente de la infraestructura vial existente, tales como: semáforos, paneles de mensajes variables, cambio de sentido en avenidas en horas pico del tráfico y centros de monitoreo de tráfico que recaban información estadística de las condiciones de flujo de los vehículos (Martin *et al*; 1990).

Por ello, uno de los objetivos primarios de la política de transporte es la reducción del congestionamiento, y tal como Noland y Quddus (2004) exponen, anhelado debido a los costos económicos asociados con la congestión del tráfico.

1.2 Objetivos.

Desarrollar un algoritmo el cual recabará y analizará las posiciones espaciales virtuales de usuarios de transporte privado que cuenten con un dispositivo especial, el cual les indique a los conductores en tiempo real la ruta con menores tiempos de recorrido desde su origen a su destino. Para la experimentación, dicho algoritmo será implementado en un programa de

microsimulación de tráfico (AIMSUN). Así, al poseer la información, el usuario podrá cambiar de opinión al momento de tomar la decisión de la ruta más conveniente, dando pie a elegir otro medio de transporte.

Como objetivos particulares se plantean:

- Implementar las bases teóricas para la implementación física en las vialidades de un sistema inteligente de transporte.
- Analizar la eficiencia de este sistema enrutador si fuese puesto en marcha en la red vial de una ciudad.
- Contar con un sistema de base de datos que permita el análisis de los comportamientos de movilidad en la ciudad.

1.3 Hipótesis.

Una red de antenas viales conectadas a un sistema que le indique al usuario a través de un dispositivo móvil las condiciones en tiempo real de las zonas congestionadas, lo guiará (si así lo decide), por vías alternas mejorando los tiempos de viaje en zonas urbanas.

Algunas otras hipótesis particulares planteadas en la investigación son:

- Los tiempos de viaje serán menores en las vialidades en las que los usuarios que cuenten con la información suministrada por el sistema hipotético.

- Las demás vías serán directamente impactadas por la disminución de vehículos, optimizando de igual manera los tiempos de viaje en las vialidades que fueron desocupadas por los vehículos instrumentados.

2. MARCO TEÓRICO

2.1 El tráfico

2.1.1 El fenómeno del tráfico.

El tráfico vehicular es un proceso dinámico extremadamente complejo, debido a que cada vehículo se comporta de manera diferente al resto de las demás, siendo este sistema parcialmente predecible, el cual también es asociado con el comportamiento espacio-temporal de un sistema de multipartículas (Kerner, 2009).

La Conferencia Europea de Ministros de Transporte (ECMT por sus siglas en inglés) ha descrito la congestión como un "estado del tráfico en los que los vehículos están en constante parada y arranque y en el que la concentración del vehículo es alta mientras que las velocidades de flujo son bajos (ECMT,2007).

De acuerdo con Black (2010), la definición anterior se refiere a un tipo recurrente de congestión con la que se encuentran los conductores regularmente. Explica además la existencia de 4 tipos de eventos que generan la congestión no periódica.

El primer tipo de evento es un incidente de tráfico, ya sea un incidente aislado en el cual se afecten uno o más carriles de la vialidad o un accidente grave en el que se involucre la pérdida de vidas.

La segunda razón involucra los tramos en reparación o mantenimiento de la vía, en la cual se reduce la velocidad debido al cierre de carriles o señalamiento preventivo para la reducción de la velocidad normal de operación de la vialidad.

El tercer tipo de evento que genera congestión está asociado al mal clima y las condiciones adversas que este genera.

Por último, el cuarto caso se asocia a los eventos especiales como sucesos deportivos y conciertos.

Para Downs (2004) la congestión se agrava por el modo en que la economía de las metrópolis modernas y las sociedades están organizadas. Se genera congestión ya que se persigue muchos objetivos en común, como lo es el minimizar el tiempo de traslados entre orígenes y destinos; el contar con la mayoría de la gente trabajando y estudiando en horarios similares, además de la preferencia generalizada por el automóvil.

Para el autor, la congestión del tráfico es un proceso inherente de las grandes metrópolis de hoy en día, que se ve agravado por la carencia de conocimiento del flujo en calles y avenidas, aunado a la decisión de los usuarios al preferir una ruta cotidianamente; lo cual se ve reflejado en el sistema vial.

Una encuesta realizada recientemente por *International Business Machines* (IBM,2011) en 20 ciudades del mundo como Beijing, Buenos Aires, Chicago, Madrid, y México entre otros, en la que se midió la percepción del conductor en el tráfico en cuanto a estrés, ira, salud y desempeño en su camino al trabajo o la escuela. El resultado fue que la ciudad de México es la más problemática para desplazarse por automóvil, mientras que las ciudades con mejor desempeño fueron Montreal, Londres y Chicago.

IBM resalta el hecho de que la mejor esperanza para mejorar el desempeño dentro de las calles radica primordialmente en el uso de la tecnología.

2.1.2 Elementos del tráfico.

El fenómeno del tráfico de vehículos está asociado con un proceso dinámico de variables que se mueven dentro del espacio-tiempo. Así, solamente a

través de un estudio de tipo espacial-temporal, será posible analizar y entender los patrones que se presentan en el (Kerner, 2009).

Para distinguir los patrones del flujo de tráfico, se dividen en 2 partes: variables macroscópicas y variables microscópicas.

En el enfoque macroscópico, no se visualiza a los vehículos como entidades separadas. Las variables macroscópicas pueden ser calculadas para cualquier lugar, en cualquier punto en el tiempo para cada intervalo de medición. Se pueden utilizar detectores de tráfico convencionales para medir este tipo de variables. Se hablará de los dispositivos de detección de tráfico en la sección 2.1.4.

Ejemplos de variables macroscópicas son: tasa de flujo, densidad de vehículos, ocupación y velocidad media de viaje (Barceló, 2010). En las variables macroscópicas del tráfico podemos asumir que el comportamiento de los vehículos está asociado con un promedio de la muestra dentro de un intervalo de tiempo.

Es decir, las variables microscópicas examinan a cada vehículo por separado y los modelos derivados de estas variables describen la interacción entre los vehículos.

Immers y Logghe (2002) explican que debido a que es imposible conocer el comportamiento exacto de cada conductor, se han implementado modelos estocásticos para este propósito.

Ejemplos de las variables microscópicas son: coordenadas del vehículo y su dependencia en el tiempo, un tiempo de viaje autónomo y un distanciamiento entre vehículos, una longitud y velocidad independientes. Estas variables se pueden denotar como datos únicos de vehículo (Kerner, 2009)

2.1.3 La medición del tráfico.

El tiempo de viaje y las demoras son las medidas base para la medición de la congestión, sin embargo, otras mediciones son necesarias dependiendo de la necesidad. Existen algunas variables capaces de medir la congestión de una ciudad o sistema vial. Lomax *et al.* (1997) identifica dichas variables, entre las cuales se encuentran:

- Duración.
- Nivel
- Intensidad
- Confiabilidad.

La duración se define como el total del tiempo en que la congestión afecta los viajes en el sistema.

El nivel o extensión se calcula estimando el número de personas o vehículos afectados por la congestión y por la distribución geográfica de la congestión.

La intensidad es definida como la severidad de la congestión que afectan los desplazamientos. Este componente es regularmente utilizado para diferenciar entre niveles de congestión en sistemas de transporte y para definir la cantidad total de congestión.

El rubro de confiabilidad este componente clave de la estimación de la congestión es descrito como la variación en los otros tres elementos descritos. La confiabilidad es el impacto de la congestión no recurrente en el sistema vial.

Las siguientes variables son derivadas de los rubros anteriores, concibiéndose como medidas que pueden ser utilizadas para análisis multimodales y análisis aislados de un tipo en particular de transporte,

1. Tasa de viajes (minutos necesarios para viajar un kilómetro)
2. Tasa de demoras (minutos por kilómetro menos los minutos aceptables por kilómetro).
3. Demora total de la sección (tasa de demoras de dos veces el volumen vehicular en la sección).
4. Índice de movilidad de la vialidad (el volumen de pasajeros por la velocidad promedio) dividido por un factor normalizador (25,000 para calles y 125,000 para autopistas).
5. Tasa relativa de demoras (la tasa de demora dividida por la tasa aceptable de viaje).
6. Relación de demoras (la tasa de demoras dividida por la tasa de viaje real)
7. Viaje congestionado (en kilómetros de vehículos, calculado con la longitud del segmento congestionado por el volumen de tráfico sumado para todos los segmentos).
8. Tramos congestionados (en kilómetros, la suma de todos los tramos congestionados).

2.1.4 Dispositivos de captación de las variables del tráfico.

Existen técnicas modernas para obtener dichos parámetros en campo. Barceló (2010) menciona de manera general los métodos y dispositivos más ampliamente utilizados para la determinación de las variables de tráfico. Dichos dispositivos se mencionan a continuación:

Detectores infrarrojos, los cuales detectan los vehículos que interrumpen el haz de luz infrarroja al pasar por dichos detectores. La información obtenida es: flujos agregados y velocidades agregadas en intervalos cortos de tiempo.

Detectores de radar, los cuales registran la velocidad de los vehículos usando el efecto Doppler. La información obtenida es: flujos agregados y velocidades agregadas.

Detectores de ciclos de inducción: Detectan la entrada de vehículos al atravesar el campo electromagnético formado por estos aparatos. La información que usualmente se obtiene de los detectores son: flujos agregados y velocidades agregadas en intervalos cortos de tiempo. Los tipos de vehículos pueden ser obtenidos a través de patrones de inducción y cuya tecnología se encuentra en etapa experimental.

Detectores ultrasónicos: Transmiten ondas ultrasónicas en lugar de ondas electromagnéticas. La información recabada es: flujos agregados en intervalos cortos de tiempo además de un registro del tipo de vehículos los cuales se distinguen por su longitud y altura. Si es posible colocar 2 detectores ultrasónicos es posible medir la velocidad de los vehículos.

Video cámaras: Detectan vehículos entrando y saliendo a la red vial. Con este tipo de dispositivos es posible obtener flujos agregados y velocidades agregadas en intervalos cortos de tiempo, además de poder obtener el tiempo individual de viaje si se utiliza reconocimiento de placas.

Sondas vehiculares: Transmiten mensajes que contienen la locación, velocidad y otros datos en un tiempo predefinido o en intervalos de tiempo.

Datos móviles, provenientes de teléfonos celulares a través de tecnología GSM/GPRS, los cuales se empiezan a usar para obtener el tiempo de viaje de los vehículos, relaciones dinámicas de origen y destino y modo de elección de la red de transporte.

Dispositivos electrónicos de peaje, cuyos datos muestran la localización y el número de veces que atraviesan dichas infraestructuras. Es posible también determinar el origen y destino de los vehículos y el tiempo de viaje.

Sin embargo, los dispositivos antes enlistados no poseen las características de detectar y transmitir de la información en tiempo real de las variables que se presentan según condiciones del tráfico que sirvan para detectar la presencia de congestionamientos, tales como tiempo de viaje de cada vehículo, colas en intersecciones, entre otras.

Para la simulación que se realizará se tomará en cuenta las posibilidades de transmisión de dispositivos móviles hacia elementos receptores que se localizarán en la red vial por la cual circularán los automóviles.

2.1.5 Teorías del flujo del tráfico.

Las teorías del flujo del tráfico describen el movimiento del tráfico y la interacción que los vehículos tienen entre sí.

Lighthill y Whitman (1955) fundaron las bases de la teoría macroscópica del flujo del tráfico, describiendo al tráfico como un fluido moviéndose a lo largo de una tubería.

Dicha teoría es la base para los modelos de comportamiento vehicular hoy en día utilizados. Existen autores que se han dado a la tarea de analizar y de crear teorías de lo que sucede en el tráfico, y entre los ejemplos que consideramos de importancia mencionaremos la teoría de Kerner, quien establece la existencia de fases dentro de la red vial. Menciona que una fase del tráfico es un estado del tráfico considerado en el tiempo y en el espacio y que posee características empíricas únicas. Su estudio consistió en identificar patrones de los automóviles dentro del tráfico y delimitar dichas fases; así su estudio mostró que dentro del tráfico existen tres fases.

Las tres fases del tráfico que identificó son:

Flujo Libre: Se observa usualmente en las vías que presentan poca densidad vehicular. La densidad al ser poca las interacciones entre vehículos son casi nulas. Además los vehículos tienen la oportunidad de moverse con las velocidades máximas a las que la vía por la que circulan lo permita.

Embotellamiento en movimiento: Un embotellamiento en movimiento es un estado dentro la congestión del tráfico, que mantiene la velocidad media del frente corriente abajo del embotellamiento, a la vez que este se propaga. Los vehículos aceleran corriente abajo del embotellamiento desde bajas velocidades, a veces desde cero kilómetros por hora a mayores velocidades. Este estado mantiene la velocidad media del atasco corriente abajo, aun cuando se desplace hacia otras fases de tráfico.

Flujo sincronizado: En esta fase, se divide el flujo en corriente abajo y corriente arriba. Corriente abajo indica hacia donde los vehículos siguen su rumbo definido. Corriente arriba son los autos que se encuentran más atrás de estos. En esta fase la corriente abajo no mantiene la velocidad media de la parte frontal de la corriente abajo.

Estas definiciones de tráfico durante la congestión son realizadas a través de los patrones macroscópicos y empíricos analizados de los datos de diferentes situaciones de congestionamiento, las cuales, siendo identificadas dentro del sistema vial de una ciudad, se pueden implementar medidas de mitigación de tráfico eficaces.

2.1.6 Efectos del tráfico.

El congestionamiento del tráfico se ha convertido en una constante en la vida de muchos de los automovilistas. Cada mañana, millones de conductores alrededor del mundo se sientan casi inmóviles en sus vehículos por largos

periodos de tiempo en el trayecto a sus lugares de labor, y repiten la misma experiencia en su trayecto de vuelta a casa (Kerner, 2004).

Otro efecto negativo de estas condiciones prevalecientes en las ciudades es el incremento de contaminantes que afectan de manera directa la salud de quienes interactúan con el entorno en conflicto, lo que ocasiona el incremento de padecimientos malignos por estas causas y por consiguiente un costo adicional mayor a la vida cotidiana.

A partir de lo expuesto, se observa el porqué la reducción del congestionamiento es declarado como objetivo primario de la política de transporte, debido a los costos que generan los efectos negativos de la congestión.

Estos costos se pueden estimar como pérdidas, ya que es el gobierno quien tiene que invertir más en los tratamientos utilizados para este tipo de enfermedades, los usuarios incrementan su gasto en combustible, los trabajos tienen que pagar por las incapacidades de sus trabajadores y por si fuera poco, el desempeño de los trabajadores después de las horas que pasan en el tráfico es deficiente.

2.1.7 Medidas de mitigación de tráfico.

Ya que el problema del tráfico no es nuevo, se han implementado algunas soluciones para reducir la congestión y estas se enlistan a partir de lo expuesto por Dunphy (1991):

- Mejoramiento en las señales de tráfico.
- Expansión del sistema vial.
- Implementación de brigadas para el control de accidentes.

- Vehículos compartidos para trabajadores y flexibilidad en horarios laborales.
- Carriles para vehículos de alta ocupación.
- Aprovisionamiento de un tren ligero urbano.
- Estrategias de uso de suelo.
- Trabajo a distancia.
- Vialidades urbanas de cuota.
- La alternativa de no hacer nada.

2.1.7.1 Mejoramiento de las señales de tráfico.

Uno de los problemas del tráfico son las intersecciones en calles urbanas. Existen dispositivos y señalamientos que permiten el flujo en diferentes direcciones de la intersección por un periodo de tiempo, llamadas fases. Los tiempos de estas fases, comúnmente prefijados, son los que en ocasiones provocan los congestionamientos debido a la nula interacción que presentan con los volúmenes de tráfico vehicular. La incursión de semáforos actuados en intersecciones (considerados como parte de los sistemas inteligentes de transporte), es un ejemplo de una medida de mitigación del congestionamiento vehicular que ha demostrado tener efectividad en dichos cruces (Yarger, 2010). Un ejemplo de una medida de mitigación del congestionamiento poco efectiva es la del señalamiento que permite la vuelta a la derecha en la intersección con luz roja si no existen vehículos aproximándose a la intersección. Dicha medida se encontró más tarde peligrosa tanto para otros vehículos así como para ciclistas, motociclistas y principalmente peatones, por lo que su uso se ha descontinuado.

2.1.7.2 Expansión del sistema vial.

Una de las primeras medidas en respuesta al problema del tráfico en países desarrollados fue el de expandir el sistema vial construyendo más calles y con más carriles. Esta medida tuvo su fundamento en que los urbanistas de épocas

anteriores no habían contado con la expansión en el número de vehículos que circularían por dichas calles. En algún punto de esta expansión tanto urbanistas como ambientalistas se preguntaron la efectividad de la expansión vial, llegando al punto de constatar que dicha expansión no contribuiría a la resolución de la congestión del tráfico, aunque esto no significa que no es necesario el dotar a una ciudad con mejores vialidades.

2.1.7.3 Implementación de brigadas para el control de accidentes.

En ocasiones, se presentan congestionamientos del tráfico debido a percances entre vehículos, provocando afectaciones en los carriles adyacentes al siniestro. Lo anterior deriva en la disminución de la velocidad de la vialidad por el embotellamiento que se presenta en el lugar del percance y la curiosidad de los automovilistas que disminuyen la velocidad para observar. Todo lo anterior deriva en la demora de los servicios especializados para la atención a los vehículos involucrados y el congestionamiento del tráfico prevalece hasta que se despeja la vía. Con la implementación de brigadas especializadas en atención de siniestros vehiculares que permanezcan circulando en horas pico en vialidades con alto índice de accidentalidad, la vía es despejada en poco tiempo y se reducen los efectos del embotellamiento.

2.1.7.4 Vehículos compartidos y flexibilidad en horarios laborales.

Actualmente el modelo de "*car-sharing*" cada vez está teniendo más aceptación como medida de mitigación del congestionamiento del tráfico y de contaminantes. Las estadísticas reportan alrededor de 60,000 usuarios en los Estados Unidos y 11,000 en Canadá en el año de 2004. Dicho modelo consiste en rentar un automóvil por un periodo de tiempo y pagar únicamente por dicho periodo. La creciente demanda es debida a que los costos fijos generados por la posesión de uno o más vehículos se convierten en insumos tarifados que generan

un ahorro a corto plazo. Este modelo también permite la optimización del espacio de estacionamientos en calles y edificios, provocando una dinámica de movilidad mayor dentro de las ciudades. Aunado a lo anterior, la incorporación de horarios flexibles de llegada y salida en instituciones gubernamentales y privadas permite una reducción de automóviles en las horas pico de la ciudad, distribuyendo el flujo total en tiempos más prolongados.

2.1.7.5 Carriles exclusivos para vehículos de alta ocupación.

La utilización exclusiva de carriles para vehículos de alta ocupación es una medida que ha logrado reducir el flujo de vehículos que circulan por avenidas altamente demandadas, reemplazando los viajes realizados en vehículos con un porcentaje bajo en ocupación por vehículos con mayor capacidad y mayor porcentaje de ocupación (VAO, BUS, entre otros). Al ser carriles exclusivos las velocidades alcanzadas son mayores a las adyacentes no exclusivas, por lo que los viajes se realizan de manera más rápida.

2.1.7.6 Aprovisionamiento de tren ligero urbano.

El tren ligero es un descendiente de los tranvías de principios del siglo XX. Es visto como una medida para reducir el congestionamiento del tráfico en las ciudades, aunque se ha demostrado que no es una disposición tan efectiva. La razón por la cual países desarrollados como Estados Unidos o Canadá han adoptado el tren ligero dentro de las urbes es por que dichas ciudades poseen diversos medios de transporte no contaminantes, por lo que el tren ligero es una clara alternativa al automóvil.

2.1.7.7 Estrategias de uso de suelo.

El uso de estas estrategias se delega a urbanistas o planificadores urbanos con el fin integrar diversos servicios en conjunto con el objetivo de minimizar el

uso del transporte. Estas medidas exigen el establecimiento cercano de zonas habitacionales, centros de trabajo, zonas de esparcimiento y supermercados con el fin de que el transporte no sea tan demandado.

2.1.7.8 Trabajo a distancia.

Sin lugar a dudas, la congestión del tráfico sería menor si existiera una reducción en la cantidad de automóviles que circulan en las calles. Si se generan problemas de aglomeramiento en las vialidades de usuarios tratando de llegar a sus centros de trabajo, ¿por qué no mejor optar por que trabajen desde su hogar?

Existen tecnologías de telepresencia que pueden ser utilizadas para este medio, logrando una interacción con sus colegas y desarrollar sus actividades normales de trabajo.

2.1.7.9 Vialidades urbanas de cuota.

Dentro de las ciudades se maneja el concepto de vialidades tarifadas para circular por vías no congestionadas. Este concepto se deriva de la necesidad de atribuirle un costo a la congestión, o más específicamente, a atribuirle un costo a una vialidad libre de congestionamiento. Este tipo de vialidades le permite decidir al usuario entre pagar un coste adicional para ahorrar en tiempo de recorrido o continuar dentro de las vialidades congestionadas (De Palma y Lindsey, 1998). Lo anterior es un ejemplo del "*User-pays principle*" el cual establece que se paga proporcionalmente a la cantidad del servicio que se utiliza.

2.1.7.10 La alternativa de no hacer nada.

Ciertas opiniones de los investigadores del transporte se han enfocado a los problemas que han incrementado después de aplicar alguna medida de mitigación del congestionamiento del tráfico.

Como resultado de lo anterior, se han incrementado el número de investigadores que congenian en la idea de que es más sensato no tratar de resolver el problema del congestionamiento del tráfico. El argumento utilizado es que si no se aplican medidas para tratar de resolver el problema de la congestión, el problema se resolverá "sólo", ya que esto llevará a las personas a no circular por vías saturadas o a no utilizar cierto transporte altamente demandado.

Estas medidas deben aplicarse de acuerdo a las necesidades de cada escenario en particular, y se deben realizar análisis de las condiciones de la red vial como el flujo, anchos de carriles, tipo de vehículos, entre otros; para así poder realizar un estudio cuyos resultados determinen si cierta medida de mitigación es la adecuada a implantar en la red vial analizada.

Kraus y Lee (2000) realizaron una simulación con base en un estudio de las características de un sistema vial cuyas condiciones de saturación terminarían por encaminar el sistema a una situación de colapso vial, por lo que evaluaron las alternativas de mitigación en una herramienta de microsimulación, lo que permitió ajustar las medidas de mitigación correspondientes.

Existe otra medida de mitigación que involucra avances tecnológicos en el ámbito electrónico y de telecomunicaciones, el cual por su relativa facilidad de instalación y operación dentro de las vialidades, engrandece las cualidades de dichos sistemas. A continuación se exponen las características de los Sistemas Inteligentes de Transporte.

2.2 Sistemas Inteligentes de Transporte (ITS)

Existen esfuerzos por parte de los gobiernos como los antes ya mencionados en el apartado 2.1.7 cuyo fin es el de unificar sistemas que ayuden a la reducción de la accidentalidad y de los problemas que ha traído consigo el

incremento de tráfico en ciertas partes del mundo. Dichos esfuerzos se han concentrado en sistemas que puedan proveer al usuario automovilista de seguridad, información del estado del tráfico y confort. El nombre que se le ha dado a dichos esfuerzos es Sistemas Inteligentes de Transportes (*Intelligent Transport Systems*, por sus siglas en inglés ITS) , como resultado se han creado sistemas complejos (que en algunas ocasiones se encuentran en el estado del arte), que se utilizan para mejorar la experiencia de conducción del automovilista. Estos sistemas involucran vehículos, conductores, pasajeros, operadores y administradores de carreteras, interactuando todos ellos entre sí (Williams, 2008).

Los ITS son una integración de *software*, *hardware*, conceptos de ingeniería de tránsito y sistemas de comunicación para mejorar la eficiencia y seguridad al conducir (Chowdhury y Sadek, 2003).

El tema de los sistemas inteligentes de transporte toma diferentes nombres en diferentes partes del mundo industrializado. En los Estados Unidos toma el nombre de ITS, en Europa se le nombró Prometheus y en Japón es conocido como AMTICS y RACS. Independientemente del nombre que se le haya dado, estos programas comparten objetivos similares, y la tecnología que ayudará a cumplir dichos objetivos se encuentra íntimamente relacionada (García *et al*. 1995).

De acuerdo con Black (2010), existen seis áreas reconocidas en el área de los ITS, de las cuales existen tres que son orientadas tecnológicamente, mientras que las demás son orientadas a aplicaciones. A continuación se describen únicamente las áreas tecnológicas.

Advanced Traffic Management Systems (ATMS)

Estos sistemas incluyen tecnologías designadas para monitorear, controlar y manejar el flujo del tráfico en calles y avenidas. Ejemplos de estos sistemas incluyen un centro de control y monitoreo de tráfico en sitios críticos en la zona

urbana con el fin de controlar los diversos elementos de infraestructura vial tales como semáforos, paneles de mensajes variables que indiquen las desviaciones por algún percance, y el despacho de vehículos de emergencia en situaciones de siniestros (Tang y Xi, 2010).

Advanced Traveler Information Systems (ATIS)

Esta área incluye tecnologías que son de ayuda para los viajeros para una correcta planeación de su ruta, así como proporcionar eficiencia en el desarrollo del viaje. Ejemplos de esta tecnología acoplada al vehículo puede incluir el desplegado de mapas digitales que muestren la información dirigida al conductor proveniente de las señales de la carretera, sistemas que interpreten datos digitales del tráfico y sistemas de manejo que reporten las condiciones de la carretera sobre el trayecto. Las no acopladas al vehículo incluyen: servicios de planeación de viaje y servicios de rutas de transporte público son las funciones más comunes provistas por esta tecnología (Martin *et al*. 2005).

Advanced Vehicle Control Systems (AVCS)

Los AVCS están diseñados para asistir a los usuarios en controlar sus vehículos, relevándolos de ciertas tareas a través de uso de dispositivos electrónicos, mecánicos o de comunicación. Incluyen los sistemas de control automático de la velocidad, el cual reduce la velocidad del vehículo si se detecta que el vehículo está demasiado cerca de otros; sistemas para incrementar la visión de los usuarios en la noche o durante eventos de tormenta; sistemas que adviertan al conductor que ha invadido el carril contrario (Shladover, 2010).

Al enfocarnos más a los sistemas ATIS, encontramos un subgrupo enfocado al ruteo de vehículos por medio de dispositivos incorporados en los automóviles, los Sistemas Dinámicos de Ruteo.

2.2.1 *Dynamic Route Guidance Systems*

Derivado de los ITS, los Sistemas Dinámicos de Ruteo (DRGS) están recibiendo bastante atención. De acuerdo con Dong (2011), estos sistemas pueden proveer sugerencias a los usuarios de las rutas óptimas por las cuales pueden circular de acuerdo a las condiciones del tráfico. La información del tráfico es recabada a partir de la posición de los automóviles, y se puede considerar como una solución de última generación para el problema del tráfico.

Dichos sistemas se dividen en 2:

- Sistemas Dinámicos.
- Sistemas Estáticos.

2.2.1.1 Sistemas de Ruteo Estáticos.

Los sistemas estáticos funcionan a partir de los datos históricos de los tiempos de viaje que se presentan en las vialidades. De esta manera, no es necesario realizar grandes cálculos computacionales, ya que dichos datos se encuentran dentro del dispositivo.

2.2.1.2 Sistemas de Ruteo Dinámicos.

Estos sistemas reciben información en tiempo real de las condiciones del tráfico, por lo que el cálculo computacional es grande y es necesario contar con información fidedigna para realizar dichos cálculos.

Como se mencionó estos sistemas recaban la información a partir de la posición del vehículo, por lo tanto es indispensable contar con mapas digitales detallados de la zona en donde se requiera

2.2.2.3 Mapas Digitales

Un punto fundamental para poder empezar a desarrollar un ITS son los mapas digitales. Para poder empezar a planear y reconocer los problemas que se encuentran dentro de una red vial, es preciso contar con la mayor y más precisa información cartográfica. Para el caso de los sistemas ATIS dichos mapas son fundamentales.

Por otra parte, los mapas digitales solo son una base para la elaboración de un sistema inteligente de transporte.

En el caso de un sistema vial, los elementos que son necesarios ubicar como parte de la infraestructura son las calles, avenidas, intersecciones, semáforos entre otros. Estos elementos se pueden considerar como fijos y de fácil dimensionamiento, ya que dichos elementos no son susceptibles de sufrir grandes variaciones a lo largo del tiempo. Otro punto importante es la localización de los elementos dinámicos dentro de esta red vial. Dichos elementos dinámicos son las partículas que se mueven dentro del sistema vial, es decir los automóviles. A continuación se presentan las maneras para realizar el posicionamiento de estos elementos dinámicos.

2.2.2.4 Sistemas Posicionadores de Vehículos.

Un aspecto importante para poder implementar un sistema DRGS, es necesario conocer la ubicación física de los vehículos, por lo que se presentan los métodos más recientes para la localización de vehículos.

2.2.2.4.1 Sistemas Posicionadores Globales Satelitales.

Los sistemas posicionadores satelitales o GPS, es el sistema totalmente funcional más utilizado hoy día. Es utilizado ampliamente en diversos campos alrededor del mundo, ya sea en trabajos específicos que requieren de precisión milimétrica o como actividad recreativa de campismo o excursión al únicamente

referir la posición con algunos metros de incertidumbre. El proceso de posicionamiento se basa en los datos enviados por un conjunto de satélites hacia el dispositivo receptor, de los cuales es necesario contar con la información de al menos 3 satélites para posicionar coordenadas latitud y longitud y 4 satélites para determinar la altitud. Una de las desventajas de este sistema radica en que el posicionamiento se dificulta debido a las condiciones climáticas adversas prevalecientes (neblina, lluvia, nubarrones, entre otros), lo cual dificulta su utilización en estos ambientes. Dentro de las ciudades, su utilización se ha incrementado como método de localización de lugares turísticos o centros de interés, al calcular la ruta deseada de acuerdo a la posición del vehículo. Otra aplicación es la de rastreo de vehículos por medio de dispositivos dentro del automóvil que envían la información de la localización del automóvil a centros dedicados a la recuperación de automóviles, también basadas en sistemas GPS.

2.2.2.4.2 Sistemas Posicionadores basados en Redes Inalámbricas.

El uso de dispositivos para conectar inalámbricamente redes de computadoras se ha extendido dentro de diversos ámbitos de la sociedad. Hoy en día no es difícil encontrar escuelas con conexión a internet de manera inalámbrica, al igual que en instituciones gubernamentales, privadas, oficinas particulares y locales de comida, entro otros. Las características de una red de área local inalámbrica (WLAN) vienen definidas por la especificación del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, 2012). Wi-Fi es el nombre de la certificación otorgada a los dispositivos que cumplen con los requisitos para operar en una red inalámbrica que cumple con el estándar 802.11. Los dispositivos certificados encargados de transmitir y recibir datos de manera inalámbrica hacia las terminales como Laptops, PDA’s o se denominan puntos de acceso (Access Points, por sus siglas en inglés AP’s). El modelo definido por el estándar 802.11 no contempla ningúna característica dentro de sus especificaciones que permita posicionar una terminal dentro del rango de acción del AP.

El posicionamiento en interiores empleando la tecnología basada en el estándar 802.11 se fundamenta en el análisis de los datos proporcionados tanto por la terminal como por el AP denominados *beacons*. A través del procesamiento de estos beacons es posible realizar 3 métodos básicos de posicionamiento:

- Sensor de proximidad.
- Multilateración.
- Huella digital (*Fingerprinting*)

Para el método de sensor de proximidad, que es el más sencillo de implementar, pero el que más variación presenta. Multilateración presenta un enfoque más estructurado, al incorporarle coordenadas o puntos de referencia a cada AP, teniendo mejor desempeño al momento de calcular la posición de alguna terminal.

De estos 3 métodos de posicionamiento para interiores el más efectivo ha resultado ser el de huella digital y multilateración.

El posicionamiento para exteriores basado en los métodos igualmente utilizados para interiores aún se encuentra en estado del arte, pero con resultados prometedores.

Li *et al.* (2008), llevaron a cabo un experimento en la ciudad de Sydney en donde utilizaron el método de huella digital y trilateración para probarlo con fines de posicionamiento en exteriores, el cual puede ser utilizado para determinar la posición de vehículos, personas, camiones de carga, entre otros. En dicho experimento también se encontraron ventajas en utilizar el posicionamiento a través de redes de área local inalámbricas contra el posicionamiento por GPS, el cual resultó tener contraveniencias al tener dificultades para derterminar la posición en exteriores en la ciudad debido a la interferencia delos edificios con la señal transmitida por los satélites, en la cual resultó con mejores resultados la señal de los AP’s.

2.2.2.4.3 Sistemas Posicionadores basados en Redes Inalámbricas y GPS.

La inclusión de ambos sistemas de posicionamiento tanto de GPS como de WLAN incrementa las posibilidades de contrarrestar las deficiencias que se puedan encontrar en ambos sistemas. El posicionamiento a través de WLAN puede ser mejor en situaciones donde los edificios bloqueen la señal satelital, y el GPS puede ser mejor en lugares donde no se encuentren los suficientes AP's para realizar una correcta localización. Las técnicas para fusionar ambos sistemas se basan en algoritmos de filtrado, como el *Kalman Filter*, entre otros.

2.2.2 Análisis del empleo de ITS.

Una vez que se han cubierto las cuestionas básicas para la implementación de un ITS, es necesario justificar su implementación en alguna zona de conflicto vial. Si bien los ITS son soluciones cuyo único objetivo es el de mejorar las condiciones viales existentes, es necesario realizar un análisis a profundidad en donde se pueda validar su futuro funcionamiento. Es aquí donde el desarrollo computacional de herramientas que puedan simular el comportamiento de los vehículos desplazándose por las vialidades resulta un método efectivo para el análisis de los ITS. Al emplear estas herramientas de simulación vial es posible adaptar virtualmente el ITS y verificar si los resultados del análisis son satisfactorios o no (Kraus y Lee, 2000).

Es conveniente, dicho lo anterior, centrarse en el manejo de dichas herramientas y las rutinas que utilizan para llevar a cabo la simulación de vehículos. Comenzaremos por definir las partes que integran a la simulación de tráfico.

2.3 Simulación de Tráfico.

En años recientes, la simulación de tráfico, y más específicamente la microsimulación de tráfico, ha migrado del ámbito académico al mundo profesional. La microsimulación de tráfico se basa en la emulación de los flujos de los vehículos, tomando a cada vehículo como un elemento independiente y dinámico que se mueve dentro del espacio y el tiempo, y que a su vez interactúa con otros vehículos. Dicho enfoque ha resultado atractivo para los planificadores de transporte e instituciones dedicadas a proveer de soluciones para el fenómeno del tráfico y sus efectos dentro del sistema vial.

Para comprender mejor el objetivo de los programas de cómputo de microsimulación de tráfico, comenzaremos con definir los principales modelos que los componen.

2.3.1 Fundamentos de la Simulación de Tráfico.

Actualmente se han creado programas de simulación de tráfico cuyo objetivo ha sido el de incluir modelos matemáticos que tienen similitud con el comportamiento de los conductores dentro de las redes viales, permitiendo así simular el comportamiento de los vehículos dentro de una red vial, proveyendo a los científicos de una herramienta para el análisis del tráfico vehicular.

Algunos de estos programas son: AVENUE (Kuwahara, Horiguchi, Hanabusa, 2010), Paramics (Skyes, 2010), Aimsun (Ferrer y Barceló, 1993; Barceló *et al*; 1994, 1998), entre otros.

Dichos programas son capaces de simular el tráfico de manera macroscópica y microscópica.

Los modelos que utilizan dichos programas para poder simular de manera macroscópica el tráfico, se basan en modelos hidrodinámicos, los cuales hacen

analogía con los flujos vehiculares y las variables macroscópicas tomadas como promedios tales como: densidad, volumen y velocidad (Barceló, 2010).

Para poder simular las variables microscópicas de una vialidad, es necesario emplear modelos más especializados que puedan imitar de forma más exacta, las interacciones de vehículos y los eventos que se presentan más a menudo en las calles.

Los modelos matemáticos comúnmente utilizados en la microsimulación de tráfico son:

- Modelo de seguimiento de vehículos.
- Modelo de cambio de carril.
- Modelo de aceptación de espacio.

2.3.1.1 Modelo de seguimiento de vehículos.

Existen varios modelos de seguimiento de vehículos, el modelo que más se asemeja al comportamiento de seguimiento de vehículos es el de Gipps (1981), el cual está implementado en AIMSUN. Se considera un modelo *Ad hoc* en el cual los parámetros del modelo no son globales, son determinados por la influencia de parámetros de su alrededor que y que dependen del “tipo de conductor” (velocidad límite), la geometría de la sección, la influencia de vehículos en las líneas adyacentes, entre otros (TSS, 2010).

Básicamente el modelo consiste en 2 componentes, aceleración y desaceleración. El primero representa la intención del vehículo de adquirir cierta velocidad de deseo, mientras que el segundo punto reproduce la limitación impuesta por el vehículo precedente al tratar de conducir a la velocidad deseada.

El modelo establece que la máxima velocidad que un vehículo puede acelerar durante un periodo de tiempo (t, t+T) está dado por:

$$Va(n,t+T) = V(n,t) + 2.5a(n)T(1 - \frac{V(n,t)}{V^*(n)})\sqrt{0.025 + \frac{V(n,t)}{V^*(n)}}\ldots \text{ (Eq. 1)}$$

Donde:

$V(n,t)$: velocidad del vehículo *n* al tiempo *t*;

$V^*(n)$: velocidad deseada del vehículo (*n*) para la sección dada;

$a(n)$: máxima aceleración para el vehículo *n*;

T: tiempo de reacción.

Respecto a la máxima velocidad que el mismo vehículo puede alcanzar durante el mismo intervalo (*t*, *t*+*T*), de acuerdo a las características propias y las limitaciones impuestas por la presencia del vehículo líder (vehículo *n*-1) es:

$$V_b(n,t+T) = d(n)T + \sqrt{d(n)^2T^2 - d(n)[2\{x(n-1,t) - s(n-1) - x(n,t)\} - V(n,t)T - \frac{V(n-1,t)^2}{d'(n-1)}]} \text{ (Eq. 2)}$$

Donde:

d(n) (<0) es la máxima desaceleración deseada para el vehículo n;

x(n,t) es la posición del vehículo n en el tiempo t;

x(n-1,t) es la posición del vehículo precedente (n-1) al tiempo t;

s(n-1) es la longitud efectiva del vehículo (n-1);

d'(n-1) es una estimación de la desaceleración deseada del vehículo(n-1).

En cualquier caso, la velocidad definitiva para el vehículo n durante el intervalo de tiempo (t ,t+T)es la mínima de las que anteriormente definidas:

$$V(n,t+T) = \min\{Va(n,t+T), Vb(n,t+T) \quad \text{(Eq. 3)}$$

Entonces, la posición del vehículo n dentro del carril en cuestión es actualizada tomando esta velocidad dentro de la ecuación de movimiento:

$$x(n, t+T) = x(n,t) + V(n, t+T)T \qquad \text{(Eq. 4)}$$

2.3.1.2 Modelo de cambio de carril.

El modelo de cambio de carril se puede considerar como un desarrollo del modelo de cambio de carril de Gipps (Gipps, 1986a y 1986b). El cambio de carril es modelado como un proceso de decisión, analizando la necesidad de cambiar de carril (por alguna maniobra de giro determinada por la ruta), el deseo de cambiar de carril (si el vehículo de enfrente es lento y se desea alcanzar mayor velocidad) y la factibilidad de poder realizar el cambio, dependiendo de la localización del vehículo dentro de la red vial. Dicho modelo se aproxima al comportamiento del usuario de la siguiente manera:

Cada vez que un vehículo tiene que ser actualizado, se realiza la siguiente pregunta: ¿Es necesario cambiar de carril? La respuesta a la pregunta depende de varios factores: la factibilidad de girar en el carril, la distancia al siguiente giro y las condiciones del tráfico en el carril en cuestión. Las condiciones del tráfico son medidas en términos de velocidad y distancias de cola. Cuando un conductor va a una velocidad menor de la deseada, tratará de rebasar al vehículo frente a él. Por otro lado, cuando está conduciendo demasiado rápido, tenderá a regresar a la línea de menor velocidad.

Si la respuesta a la pregunta anterior es afirmativa, para cambiar con éxito de carril, se deben considerar otras 2 preguntas.

¿Es deseable cambiar el carril? Si la velocidad del carril a tomar es lo suficientemente rápida comparada con la actual, o si la cola es corta, entonces es deseable el cambio de carril.

Y la consecuente pregunta: ¿Es posible cambiar carril? Se debe verificar que existe un espacio intervehicular para realizar el cambio de carril con seguridad. Para este propósito, se calcula el frenado impuesto por el inmediato flujo vehicular al vehículo intercambiador y el frenado impuesto por el vehículo intercambiador al futuro flujo ascendente.

2.3.1.3 Modelo de aceptación de espacio.

El modelo de aceptación de espacio (*gap-acceptance*), es usado para modelar la maniobra de ceder el paso a vehículos. Este modelo determina cuando un vehículo de baja prioridad en la vía acercándose a una intersección puede o no cruzar dependiendo de la posición y velocidad de los vehículos de más alta prioridad en la vía. Este modelo toma en consideración la distancia a el punto hipotético de colisión en la intersección, por medio de los valores de aceleración, velocidad de los vehículos aproximándose al punto de encuentro. Después determina el tiempo necesario para que el vehículo cruce o se integre a la intersección.

Otros parámetro pero de igual importancia tomados en consideración en este modelo son la distancia de visibilidad hacia la intersección y la velocidad de giro, los cuales están relacionados con la geometría de la intersección.

No sólo se trata de explicar los comportamientos viales por medio de modelos matemáticos, sino que estos son apenas una parte del motor del programa. Existen también rutinas destinadas a la distribución del tráfico por las avenidas virtuales que se introducen en el programa. Básicamente estamos hablando de un algoritmo que elige la ruta que tomará cada vehículo dentro del sistema.

Los datos que emplean los programas de microsimulación para asignar el punto de salida y de llegada a cada vehículo los determina la matriz de origen-

destino. Esta tiene la particularidad de ser una matriz cuadrada, es decir, con el mismo número de filas y de columnas; donde cada celda representa la cantidad de viajes que se realizan entre el origen y el destino registrados.

Con base en estos datos, el programa genera las directrices necesarias para trasladar un porcentaje de vehículos de un punto de origen a otro de destino, pero para que esto pueda ser posible, existe también otra parte del programa que se encarga de “dictar” la ruta por la que los vehículos deberán circular.

Esta parte del programa es llamada*: Dynamic Traffic Assignment,* y es este algoritmo el encargado de dirigir el tráfico dentro del sistema (TSS, 2011).

Para poder saber qué camino tomarán los automóviles, la rutina de asignación dinámica de tráfico calcula el camino mínimo entre el origen y el destino, eligiendo el que mejor se ajuste al criterio.

La forma elegida para representar las vialidades por las que los automóviles circularán, semejará una red de arcos y nodos, elegidos por la teoría de grafos para representar interconexiones de elementos relacionados entre sí.

2.3.2 Modelos de Asignación de Tráfico.

Dentro de los programas utilizados para la modelación microscópica del tráfico es necesario contar con modelos de asignación que puedan reproducir lo más fielmente posible lo ocurrido en las vialidades que se pretenden representar en dichos programas. El proceso de asignación dados un conjunto de viajes a un sistema de transporte es comúnmente referido como asignación de tráfico. Los propósitos de los modelos de asignación de tráfico son:

a) Estimar el volumen de tráfico en los arcos de la red y poder obtener mediciones agregadas de la red.
b) Estimar el costo de viaje intrazona.

c) Analizar el patrón de viaje de cada par Origen-Destino (O-D).
d) Identificar los arcos congestionados y registrar datos del tráfico para el diseño de futuras intersecciones.

De acuerdo con Mathew y Rao (2007), dentro de los modelos de asignación de tráfico más destacados que se pueden mencionar se encuentran:

a) Modelo de asignación Todo o Nada.
b) Modelo de asignación Equilibrio de Usuario (UE).
c) Modelo de asignación de Sistema Óptimo (SO).

Cabe mencionar que el modelo de Equilibrio de Usuario es el más adecuado a utilizar en un enfoque microscópico de simulación, ya que se basa en el primer principio de Wardrop, que afirma: “Los tiempos de viaje en todas las rutas usadas son menores o iguales que los que necesitaría un usuario en cualquier ruta no utilizada”. El modelo asume que cada automovilista busca minimizar el tiempo de viaje empleado para llegar a su destino y asigna a usuarios a sus rutas hasta encontrar un flujo de equilibrio, el cual se logra cuando la asignación de flujos a las vialidades es tal que cada vehículo no puede cambiar su ruta sin incrementar su costo de viaje, lo cual se puede observar en situaciones del mundo real cuando las vías se encuentran congestionadas.

Con base en estos modelos antes descritos, es posible comenzar a recrear un escenario dentro de un sistema vial, pero también es necesario conocer la representación que debe poseer el sistema vial para poder hacer uso de los modelos antes mencionados. A continuación se menciona la teoría de grafos o teoría de redes con la cual podemos representar el sistema vial.

2.4 Teoría de Redes.

Una red o grafo es una representación de un sistema interconectado entre sí, con el fin de plasmar de manera más práctica las interrelaciones que existen en él.

Las entidades del sistema que tienen relaciones entre si se designan como arcos del grafo y las uniones entre estas relaciones se identifican como nodos en el grafo.

Hernández (2005) define a la teoría de redes como una clase de modelos matemáticos que involucra la representación gráfica de ciertos problemas de optimización.

Dicha optimización se refiere a la utilización eficaz de los recursos que se desean usar en la red, es decir, encontrar un método para dirigir los recursos a un costo óptimo.

El origen de dicha teoría se le atribuye a Leonhard Euler (1736), quien por primera vez involucró un problema de red con un enfoque matemático.

Biggs *et al.* (1998) explican cómo Euler se propuso resolver un acertijo que era considerado por los algunos como imposible de solucionar y otros tantos tenían duda de si podría tener respuesta. Se trataba de una isla dentro de una ciudad, las cuales tenían interconexión entre sí a través de 7 puentes. El desarrollo de este problema llevó a Euler a 2 grandes pasos: reemplazó el mapa de la localización de dichos puentes y sus conexiones con la ciudad y lo reemplazó con un diagrama y por otro lado, formuló los aspectos más relevantes del problema de modo que el diagrama resultara innecesario. Así mismo, aplicando su propio criterio y sentido común, ideó una serie de reglas que le permitieran conocer si el arreglo propuesto en el diagrama cumplía con la hipótesis realizada sobre él (p.ej. ¿Es posible trazar una ruta que pase por todos los nodos una sola vez?). Estas reglas se convirtieron en lo que hoy conocemos como algoritmos para la solución de problemas de redes.

Existen problemas básicos de redes como lo son: ruta más corta, flujo máximo, flujo a costo mínimo, entre otros.

La ruta más corta tiene por objeto encontrar el camino más corto entre los vértices que componen la gráfica, de acuerdo a las distancias previamente establecidas en el planteamiento del problema. Para resolver el problema de la ruta más corta existen varios algoritmos dependiendo del tipo de datos con los que se cuenten. Por ejemplo el algoritmo de Dijkstra (Dijkstra, 1959), es ideal para solucionar grafos dirigidos o no dirigidos y con pesos de arco no negativos, es decir, cuando no existen datos negativos en la red. En este algoritmo se definen un vértice de origen y uno de destino y es posible calcular el camino mínimo a cualquiera de los vértices.

Muchas situaciones del mundo real pueden describirse convenientemente por medio de un diagrama consistente en una serie de puntos uniendo con líneas ciertos puntos de esta gráfica. Estos puntos pueden representar personas, intersecciones, destinos, orígenes.

Conceptualmente, una gráfica está compuesta por vértices y arcos conectando dichos vértices.

Una gráfica sin vértices es llamado un grafo nulo.

Es posible representar este conjunto como G=(V,E), donde V es la cantidad de nodos que componen la gráfica y E son los vértices que los unen (Figura 2.1).

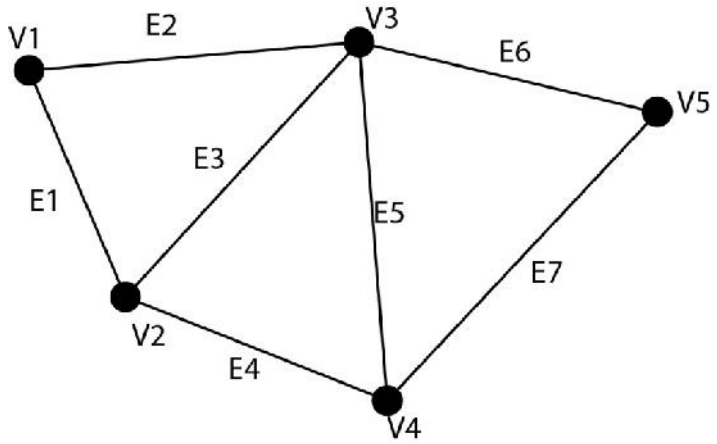

Figura 2.1. Representación del conjunto (V,E).

Fuente: Elaboración propia.

Podemos representar este diagrama de la siguiente manera:

V:={V1,V2,V3,V4,V5,V6}
E:={E1,E2,E3,E4,E5,E6,E7}

Tal que

E1:={V1,V2}
E2:={V1,V3}
E3:={V2,V3}
E4:={V2,V4}
E5:={V4,V3}
E6:={V3,V5}
E7:={V4,V5}

En este ejemplo, la red posee nodos interconectados por vértices de diferentes maneras, pero se considera un grafo no dirigido, ya que no existen flechas de dirección en los extremos de los arcos.

Cabe señalar que el ejemplo de la Figura 2.1 es la forma más básica de representación de una red, pero es necesario añadir más elementos tales como pesos de arcos y direcciones de flujo de los dependiendo del tipo de red que se quiera representar.

Existen sistemas que requieren se les dote de direcciones de flujo en sus vértices, estableciendo así una conexión en un sólo sentido entre nodos, por ejemplo, si quisiéramos representar el siguiente conjunto de vialidades de la Figura 2.2.

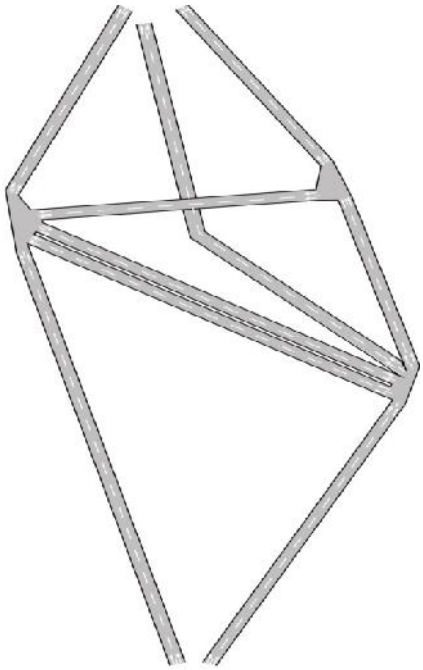

Figura 2.2. Vialidades a representar en un diagrama de grafos.

Fuente: Elaboración propia.

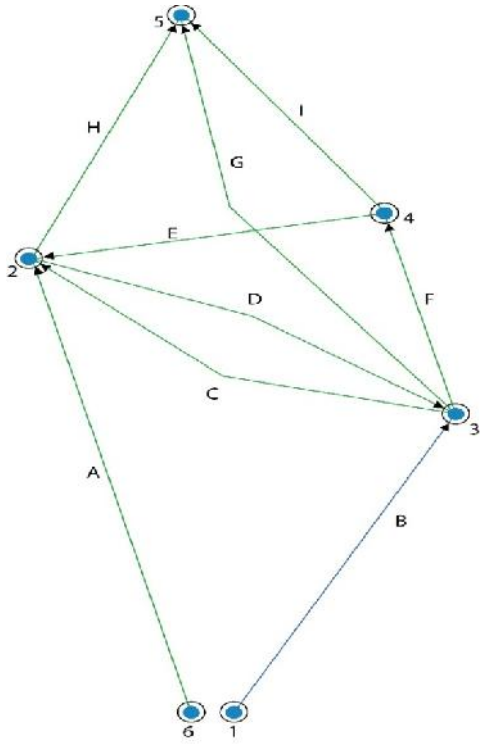

Figura 2.3. Vialidades representadas en un diagrama de grafos.

Fuente: Elaboración propia.

Podríamos representarlo en un diagrama conceptual de las vialidades mostradas en la Figura 2.3. Además por ser un sistema que posee conexiones únicas en sus nodos (sentido de circulación) se deberá representar como un grafo dirigido.

En este ejemplo, tenemos:

V:={1,2,3,4,5,6}

E:={A,B,C,D,E,F,G,H,I}

Cuyos vértices tienen la siguiente conexión con los nodos:

A:={1,2}

B:={1,3}

C:={3,2}

D:={2,3}

E:={4,2}

F:={3,4}

G:={3,5}

H:={2,5}

I:={4,5}

En el análisis de este sistema, se puede incluir una variable más en cada vértice del diagrama, que represente la distancia entre cada nodo. Cabe mencionar que para este ejemplo, se podrían incluir otras variables que existan en el sistema, como tiempo de viaje, capacidad, flujo de automóviles, entre otros.

Estas variables son conocidas como pesos de los vértices, que dependiendo de las variables elegidas (distancia, tiempo de viaje) se puede realizar un análisis diferente para cada escenario elegido.

2.4.1 Problemas de Redes.

Existen diversos problemas en la teoría de redes. Algunos de los más comunes los podemos mencionar a continuación:

- Problema del flujo a costo mínimo.
- Problema del camino más corto.
- Problema de flujo máximo.

Según Ahuja *et al*. (1993) el problema del flujo a costo mínimo se plantea la siguiente pregunta: Si se incurre en un costo por unidad de flujo en una red y necesitamos enviar unidades que residen en uno o más puntos de la red hacia uno o más destinos, ¿Cómo podemos enviar los bienes a costo mínimo?

Para Thulasiraman y Swamy (1992) encontrar los caminos más cortos especificando un nodo hacia cualquier otro nodo de la red, es un problema común en varios problemas de optimización.

West (2002), se plantea una pregunta similar, en el caso del problema de flujo máximo: ¿Cuál es el máximo flujo en una red por unidad de tiempo que puede circular entre 2 puntos?

Si bien desde el punto de vista matemático cada uno de estos problemas puede tener infinidad de soluciones, no todas estas soluciones son adecuadas en el aspecto del tiempo de análisis y procesamiento de la información. Lo anterior es referente al método que se pretenda usar para resolver el problema de red elegido. Es por eso que se han adoptado soluciones a estos problemas que contribuyan a gastar el menor tiempo posible en su solución. Estas soluciones se han nombrado como algoritmos de optimización.

2.4.2 Algoritmos para la Solución de Redes.

Como se mencionó anteriormente, dentro de la conceptualización de un problema de redes es necesario definir el problema que se quiere resolver. En este capítulo se tratará únicamente el tema del problema del camino más corto en redes.

Ahuja *et al*. (1993) mencionan que los problemas del camino más corto se encuentran dentro de las funciones principales del flujo de redes.

Como ya se había comentado, el problema del camino más corto nos permite resolver en la práctica problemas como el envío de algún material (p.ej. un archivo a través de internet, una llamada telefónica entre otros) a través de una red, de manera que sea lo más barato y rápido posible.

En este tipo de problemas es necesario realizar ciertas consideraciones en el grafo que estemos analizando.

Como primer punto es necesario establecer qué tipo de grafo analizaremos y haremos algunas consideraciones al respecto. La primera consideración es considerar una gráfica dirigida G = (N,A,d) , donde N será el número de nodos o vértices, A será el número de arcos de la red y d el peso asociado a cada arco. La red contará con un vértice s, el cual será el vértice de inicio. El análisis al problema del camino más corto determinará para cada nodo excepto el de inicio la distancia menor del vértice inicial s a los demás nodos de la red.

Suponiendo que todos los pesos de los arcos son enteros y no negativos, es posible discriminar entre el tipo de algoritmo a utilizar. Como se mencionó anteriormente, el algoritmo ideal para este tipo de problemas es el Dijkstra (Dijkstra, 1959).

2.4.2.1 El algoritmo de Dijkstra

Dentro de los algoritmos para la resolución del problema del camino más corto, es necesario mencionar al algoritmo Dijkstra como uno de los más sobresalientes, para el caso de que los pesos del grafo sean no negativos.

El algoritmo Dijkstra encuentra los caminos mínimos dado un nodo inicial *S* a los demás nodos del grafo con pesos no negativos. En los pasos de relajación, el algoritmo divide los nodos en 2 grupos: aquellos que se designan como permanentemente marcados o los temporalmente marcados. La distancia marcada hacia cualquier nodo marcado permanentemente representa la distancia más corta del nodo inicial a dicho nodo. Para cualquier nodo marcado temporalmente, la distancia marcada es el límite superior en la distancia más corta a ese nodo. La idea básica del algoritmo es analizar los pesos desde el nodo de origen *S* y marcar permanentemente los nodos restantes en orden a sus

distancias desde el nodo *S*. Inicialmente, le damos al nodo *S* una marca permanente de 0m y a todos los demás nodos *j* una marca temporal igual a ∞. En cada iteración, la marca del nodo *i* es la distancia más corta desde el nodo inicial a lo largo del recorrido cuyos nodos internos están permanentemente marcados. El algoritmo selecciona un nodo con la distancia mínima marcada temporalmente, la convierte en permantente, y posteriormente analiza los arcos cercanos al nodo para actualizar la distancia de los nodos adyacentes, y el algoritmo termina cuando todos los nodos han sido marcados permanentemente.

A continuación se presenta un diagrama en pseudocódigo del funcionamiento antes descrito del algoritmo Dijkstra (Ahuja *et al*; 1993).

algoritmo Dijkstra;
inicio
S : = ∅; $\bar{S}$ *: =* ***N;***
d(i) : = ∞ para cada nodo i ∈ *N;*
d(s) : = 0 y pred(s) : = 0;
mientras |S| < n **hacer**
inicio
dejar que sea i ∈ $\bar{S}$ un nodo para cada cual d(i) = min{d(j) : j ∈ $\bar{S}$};
S : = S ∪ {i};
$\bar{S}$: = $\bar{S}$ – {i};
para cada (i,j) ∈ A(i) **hacer**
si d(j) > d(i) + c_{ij} **entonces** d(j): = d(i) + c_{ij} y pred(j) : = i;
fin;
fin;

El resultado de implementar el algoritmo Dijkstra en un grafo, nos arroja los nodos cuyos arcos contienen los menores pesos y por consecuencia, dichos nodos son los que definen la ruta hacia el nodo de destino, con base en el nodo de origen.

Para el diseño del agente enrutador, este algoritmo de optimización será esencial para la determinación de los tiempos de viaje más cortos por los cuales deberán de circular los automóviles. Se incorporará parte del código fuente en lenguaje C++ del algoritmo Dijkstra escrito por Arias (2010) al código del algoritmo enrutador.

3. METODOLOGÍA

En este apartado se definirán las características del sistema ITS a implementar y las herramientas y procedimientos a emplear en la construcción de una rutina que asemeje las características del ITS que se desea simular.

Primeramente se dará una breve explicación del sistema que se desea conceptualizar, y posteriormente se detallarán los pasos seguidos para la realización de la simulación de dicho sistema.

3.1 Descripción del sistema a simular.

Como se mencionó en apartados anteriores, el sistema a simular es un agente enrutador de vehículos, el cual está basado en los fundamentos de los Sistemas Inteligentes de Transporte y la principal variable que tomará en cuenta será el tiempo de viaje de los vehículos en las vialidades.

Para poder simular un sistema que pueda funcionar como un agente enrutador de vehículos, el sistema debe de tener las siguientes características:

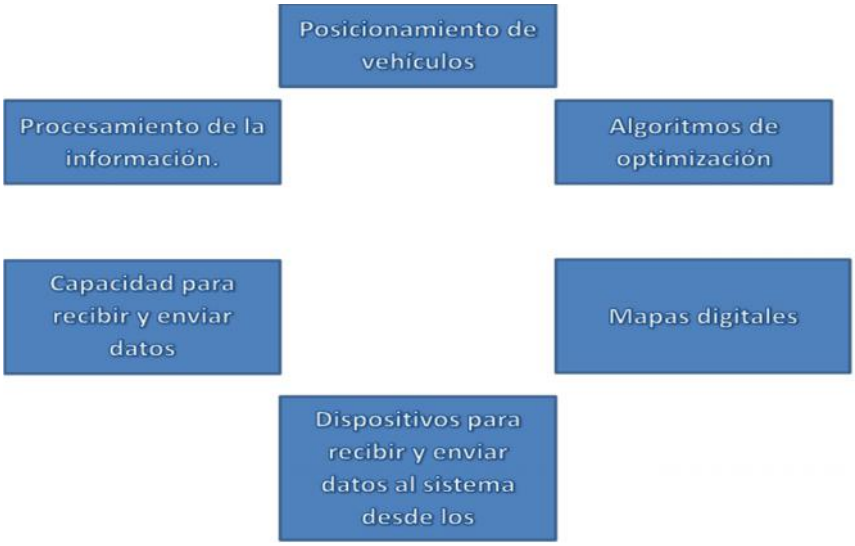

Figura 3.1. Elementos del sistema enrutador de vehículos.

Fuente: Elaboración propia.

Los elementos de la Figura 3.1 describen los elementos más importantes con los que debe de contar el sistema, los cuales estarán interactuando continuamente con el objetivo de proveer a los usuarios la información de las vialidades con el menor tiempo de viaje de acuerdo al punto de inicio de su ruta.

Cabe mencionar que los elementos que se describen, formarán parte del esquema hipotético que se desea simular, en ningún apartado de esta investigación se pretende realizar experimentos que involucren su instalación en campo.

3.1.1 El sistema enrutador

A continuación se muestra la forma en la que deberán interactuar dichos elementos en el sistema enrutador.

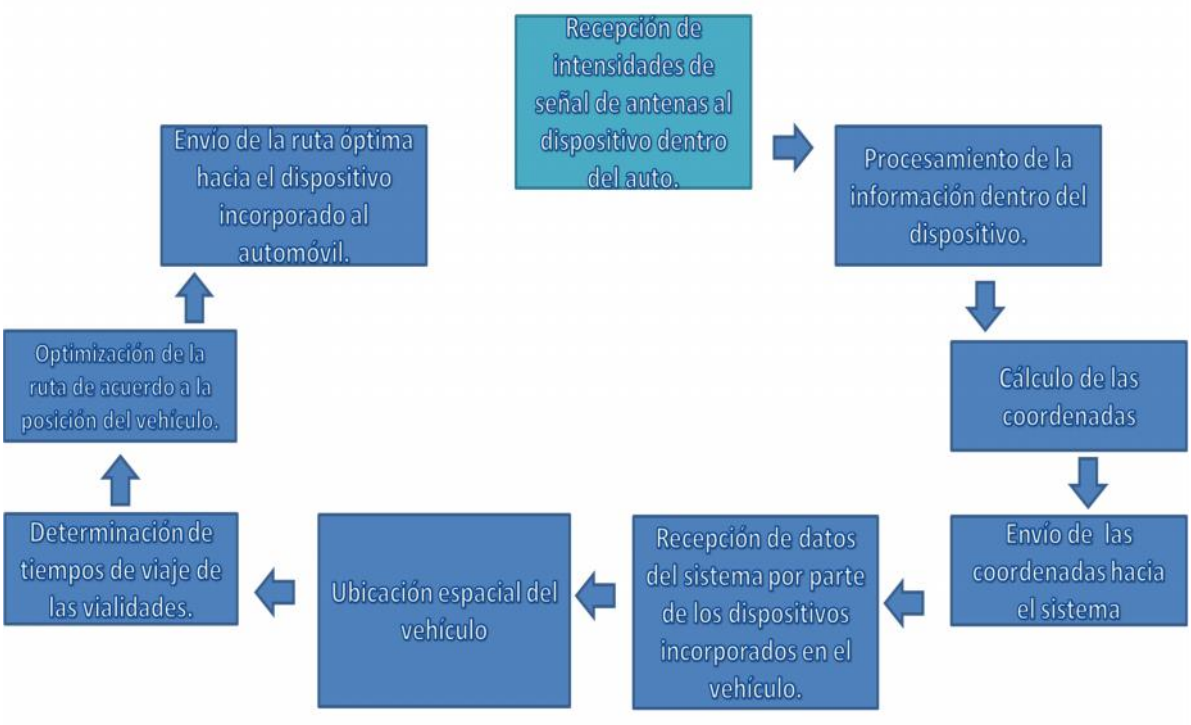

Figura 3.2. Interacción de los elementos del sistema enrutador.

Fuente: Elaboración propia.

En el esquema de la Figura 5 se puede apreciar que el sistema comienza con la recepción de intensidades de señal de antenas Wi-Fi al dispositivo dentro del automóvil, dado que este es el primer paso que se deberá completar en orden de inicializar el sistema enrutador.

En el apartado 2.2.2.4 se hablo acerca de los sistemas posicionadores de vehículos, los cuales son capaces de determinar la posición en tiempo real de dichos automóviles con base en los resultados que arrojó la experimentación de posicionamiento a través de redes inalámbricas (WLAN). En dicho experimento, se utilizó la trilateración de intensidad de señales por parte de los puntos de acceso (AP) para obtener la posición relativa a dichas antenas. Es por eso que es necesario que se cuente con un dispositivo que sea capaz de procesar dicha información de intensidades de señal (computadora portátil, *smartphones*, entre otros), y que además cuente con capacidades de procesamiento de dicha información para que esta sea enviada a través del mismo hacia una infraestructura vial hacia una central de información.

En el siguiente apartado, se describirá brevemente las características con las que debe de contar el dispositivo que deberá ser incorporado al automóvil, de manera temporal o permanente.

3.1.2 El dispositivo electrónico.

Actualmente, existe una gran variedad de dispositivos electrónicos que integran una gran cantidad de funcionalidades: toma de fotografía y video, procesamiento de datos, posicionamiento global satelital, entre otros.

Esta última funcionalidad le da la capacidad al usuario de alguno de los dispositivos mencionados anteriormente de conocer su posición espacial por medio del análisis de los datos de un conjunto de los satélites de posición global que orbitan la tierra, pudiendo así determinar la localización espacial en coordenadas geográficas.

Para su mejor interpretación, se hace referencia a las coordenadas proyectadas en el sistema UTM, el cual maneja las coordenadas de manera decimal, por lo que se requieren 2 datos para conocer la posición del aparato: La coordenada X (longitud) y la coordenada Y (latitud).

Dicho lo anterior, al encenderse dicho dispositivo mientras el usuario hace uso del automóvil, los datos que se pueden recabar son la posición geográfica del vehículo y el tiempo en que se obtuvieron dichas coordenadas, como se muestra en la Figura 3.3.

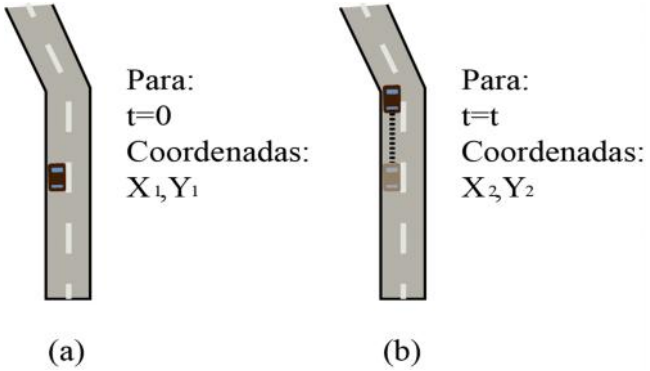

Figura 3.3. Representación del cambio de posición del vehículo y los datos recabados.
Fuente: Elaboración propia.

En el esquema (a) de la figura anterior, el dispositivo obtiene las coordenadas para ese punto para el tiempo *t*=0, y sucesivamente para los siguientes intervalos de tiempo (Figura 6 b).

De igual manera, cambiando el sistema de posicionamiento de GPS (basado en el posicionamiento a través de señales de satélite) por el de WLAN, es posible obtener las coordenadas a través de la trilateración de intensidades de señal de AP's (*Access points*).

En el sistema de enrutamiento, se tomará como base el método de posicionamiento a través de redes inalámbricas, por lo que es necesario que el dispositivo cuente con las siguientes características.

-Recepción de señales provenientes de los AP (para el cálculo de coordenadas).

-Envío de señales hacia los AP (para el envío de las coordenadas calculadas)

-Despliegue de mapas y de rutas (las rutas óptimas por las cuales los tiempos de viaje son menores).

-Procesamiento de datos (para calcular las coordenadas con base en la trilateración de señales).

De acuerdo con estas características, el dispositivo podrá interacturar con una infraestructura basada en redes inalámbricas (WLAN). Dicha infraestructura se describe a continuación.

3.1.3 Infraestructura vial de redes inalámbricas.

Si bien previamente se explicó que el sistema se basará en el posicionamiento a través de rédes inalámbricas, es necesario establecer ciertas características de la infraestructura que se colocaría en las calles y avenidas para la difusión de intensidades de señal hacia los dispositivos previamente definidos.

La infraestructura que es necesaria para el funcionamiento del sistema enrutador, constará primordialmente de AP’s, como previamente se explicó en apartados anteriores.

Estos AP’s, deberán de contar con un enlace físico para intercambio de datos hacia un centro de recopilación de información, de manera que los datos que provengan de los dispositivos (coordenadas y tiempo) se concentren en dicha central. De igual manera, se le deberán de asignar coordenadas previamente

establecidas a cada AP, de manera que se transmitan dichas coordenadas y sirvan de referencia al dispositivo dentro del automóvil al momento de calcular las coordenadas del vehículo.

Una vez implementado un sistema que pueda recabar la información de la ubicación espacial con la precisión requerida de un número de automóviles instrumentados con que circulan por alguna vialidad, primeramente se necesitaría un algoritmo que discrimine la ubicación de estos y descarte las señales que pudieran provenir de la zona de la acera o pasos peatonales, por lo que se requieren de mapas digitales a detalle de la zona. De igual manera se tendría que ordenar los datos recabados de los vehículos para una fácil lectura del sistema.

En la Figura 3.4, se muestra gráficamente como podría darse el intercambio de información entre los AP's y el dispositivo.

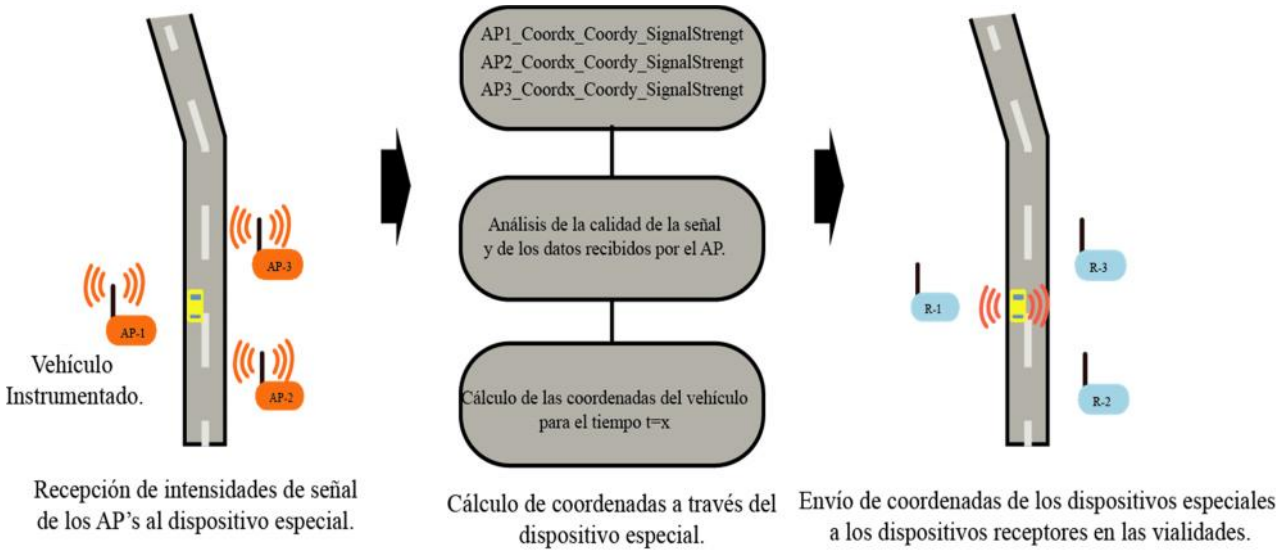

Figura 3.4. Esquema del funcionamiento hipotético del sistema.
Fuente: Elaboración propia.

Como se puede apreciar en la Figura 3.4, se aprecia la manera en que los datos pudieran ser ordenados. Los datos recabados es posible ordenarlos de diversas formas, únicamente se tiene que realizar de manera uniforme para todos ellos. Para el caso de envío de datos de los AP's a los dispositivos se propone realizarlo de la siguiente manera:

AP1_CoordinateX1_CoordinateY1_SignalStrenght
AP2_CoordinateX1_CoordinateY1_SignalStrenght
AP3_CoordinateX1_CoordinateY1_SignalStrenght

Así, sería posible implementar una función que ubique más rápidamente el arco en el que se encuentra el vehículo a través de una base de datos que contenga el identificador del AP y el arco al que pertenece.

De manera similar, para el caso de envío de datos del dispositivo especial a los dispositivos receptores se propone:

Signal1_CoordinateX1_CoordinateY1_timeVeh1
Signal2_CoordinateX1_CoordinateY1_timeVeh1
Signal5_CoordinateX1_CoordinateY1_timeVeh1
Signal1_CoordinateX2_CoordinateY2_timeVeh2

De esta forma, se discrimina entre las señales recibidas y sus coordenadas.Estos últimos datos son los que serán enviados a al sistema central a través del enlace físico de los AP’s con el sistema central, de manera que con estos datos se podrán calcular los tiempos de viaje de las calles y avenidas.

Enseguida se explicará el propósito por cual se han determinado usar estas conveniencias al recabar los datos.

3.1.4 Sistema enrutador.

En secciones anteriores definimos las características del dispositivo y de la red vial a implementar. En este apartado se describirá a grandes rasgos el rol que desempeñará el sistema central.

El sistema enrutador será el encargado de recopilar todos los datos que poseen las coordenadas y el tiempo en que se envían dichas coordenadas. Esto servirá para interpretar gráficamente la posición del vehículo respecto a sus coordenadas.

En la siguiente imagen (Figura 3.5) se presenta un esquema entre los datos obtenidos y su interpretación gráfica.

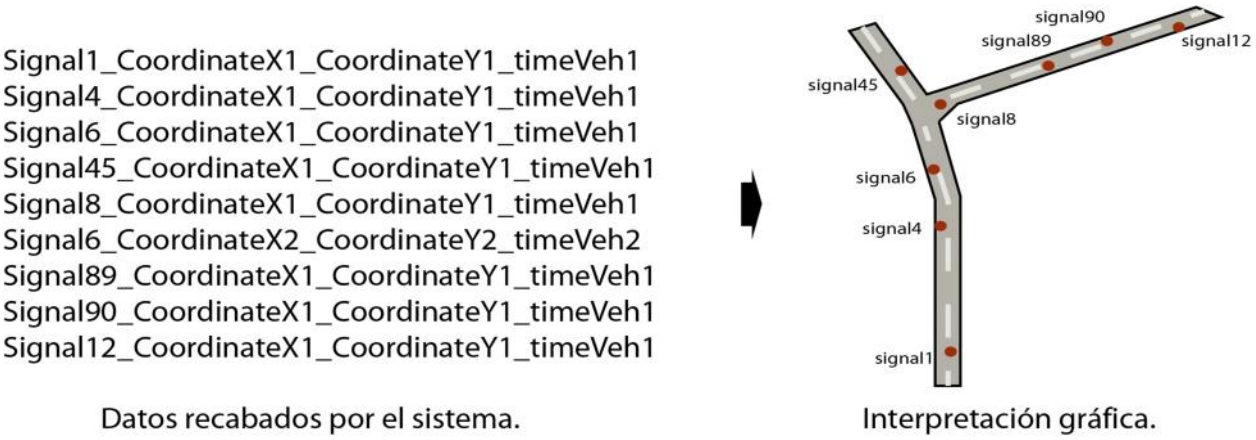

Figura 3.5. Esquema representativo entre los datos obtenidos y su interpretación.
Fuente: Elaboración propia.

Al tener la certeza de que el sistema está trabajando únicamente con datos "validos", es decir, con datos provenientes únicamente de los automóviles; se necesitará obtener tiempos de viaje de automóviles y avenidas principales.

Esto se puede realizar con una simple diferencia en los tiempos de entrada y salida de una sección cualquiera, o en otras palabras, un arco. La Figura 3.6 esquematiza lo mencionado.

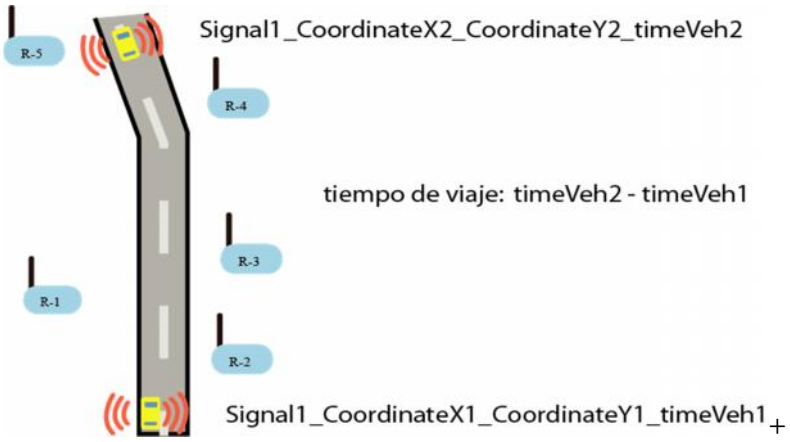

Figura 3.6. Cálculo del tiempo de viaje.
Fuente: Elaboración propia.

De acuerdo a lo anterior, se puede inferir que no es posible obtener los tiempos de viaje de todas las calles, ya que los automóviles instrumentados pudieran no estar circulando siempre por las calles a las que se desea obtener dichos tiempos de viaje, por lo que sería necesaria una base de datos de tiempos promedios de viaje obtenidos mediante mediciones en campo a diferentes horas. Con esto se asegura que se tendrán los tiempos de viaje de todas la vialidades del sistema vial.

Una vez que se obtuvieron los datos de las calles de interés, es necesario representarlos gráficamente, de manera que sea posible ubicarlos fácilmente a través de la implementación de un grafo consistente en nodos y arcos en los cuales los nodos serán las intersecciones o los centroides de las vialidades, los arcos las calles y el peso de cada arco serán los tiempos de viaje, tal como se muestra en la Figura 3.7.

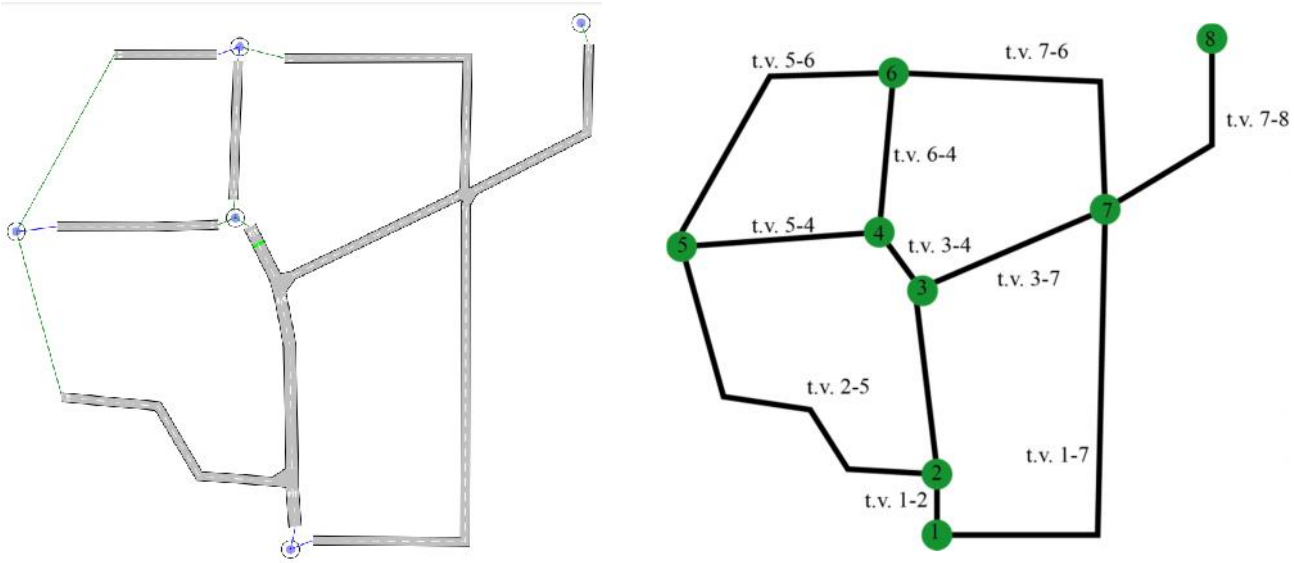

Figura 3.7. Representación de un conjunto de vialidades en un grafo.
Fuente: Elaboración propia.

Al haber realizado lo anterior, es necesario implementar un algoritmo que calcule el tiempo de viaje mínimo, dado el punto de inicio y final por parte del usuario, para lo cual existen diversas maneras de realizarlo, y como se había explicado anteriormente, se implementará el algoritmo de Dijkstra para resolver el grafo dado un vértice de inicio y uno final, que en este caso representará el origen y el destino del automovilista sobre la vialidad. Una vez obtenida la ruta óptima para cada vehículo, le será enviada a través de los AP's hacia el dispositivo, el cual le mostrará la ruta calculada según los tiempos de viaje.

3.2 Construcción del Algoritmo del Sistema Enrutador de Vehículos.

Una vez que sabemos que elementos se tomarán del sistema que se pretende simular, procederemos a establecer los pasos para realizar la simulación.

Primeramente se realizará un modelo simple de vialidades en donde se simularán las condiciones normales en las que podría operar este modelo y las condiciones en las que los automóviles cuenten con un sistema de ruteo

integrado, el cual será el modelo de pruebas. Para el modelo de pruebas se desarrollarán los pasos mostrados en la Figura 3.8.

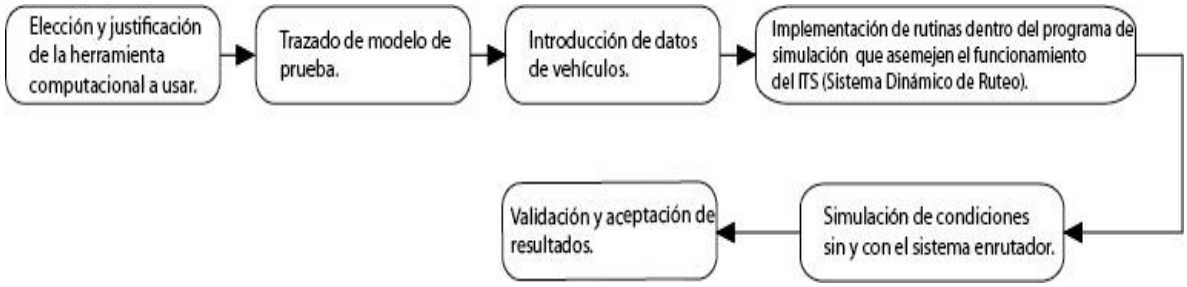

Figura 3.8. Vialidades representadas en un diagrama de grafos.

Fuente: Elaboración propia.

3.2.1 Elección y justificación de la herramienta computacional a usar.

Como se mencionó anteriormente, existen diversos programas enfocados a la simulación de tráfico. Para la elección de la herramienta tendremos que tomar en cuenta que sea un *software* capaz de aceptar controles externos desarrollados por el usuario y se puedan realizar operaciones con las funciones de ruteo de vehículo a gusto del usuario.

Uno de los programas computacionales que se ha desarrollado con el objeto de emular el proceso del flujo de tráfico vial implementando modelos matemáticos que explican el comportamiento observado en la realidad, es el programa llamado AIMSUN (*Advanced Interactive Microscopic Simulator for Urban and non-urban Net-works*), un *software* desarrollado en la Universidad Politécnica de Cataluña.

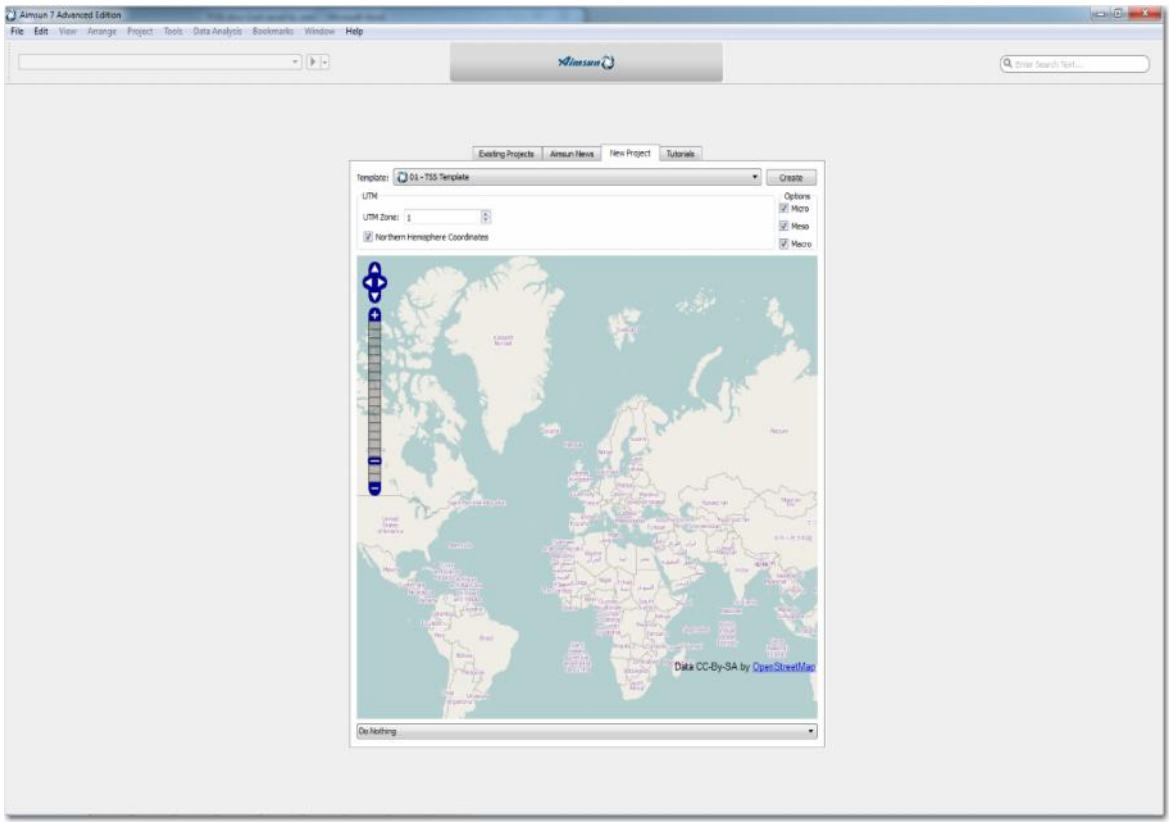

Figura 3.9. Pantalla de inicio del programa AIMSUN.

Fuente: TSS-Transport Simulation Systems.

Este *software* posee los modelos necesarios para poder simular las condiciones de tráfico de un sistema vial, y poder modelar virtualmente alguna adecuación física, como la adición de carriles a alguna vialidad o el cambio de tiempos en las fases de semáforo de alguna intersección. En la Figura 3.9 se aprecia una captura de la pantalla de inicio de dicho programa.

Otra función importante del *software* es la capacidad de poder interactuar con rutinas programadas por el usuario, desarrolladas en lenguaje C++ o Phyton, capaz de interactuar con las variables de entorno con las que AIMSUN trabaja, de manera que es posible crear una rutina que trabaje en conjunto con los datos que la simulación esté trabajando.

Esta capacidad se basa en las Aplicaciones Telemáticas Avanzadas (ATA), las cuales pueden ser adaptadas en tiempo real en una vía en específico, para poder medir elementos tales como procesadores de imágenes, contadores de autos en un estacionamiento, entre otros. Dichos dispositivos generan una respuesta que puede ser interpretada por el sistema de manera que se genere una contramedida, por ejemplo, si existe una cantidad importante de vehículos en una intersección, el sistema podría modificar el ciclo del semáforo y cambiar el tiempo de fase verde.

El Módulo que permite este desarrollo se llama AIMSUN API, el cual trabaja con rutinas hechas dentro del lenguaje de programación Phyton o Visual C++. Mediante la utilización de las funciones de AIMSUN se pueden modelar los atributos de elementos de control de tráfico como se muestra en la Figura 3.10.

Figura 3.10. Esquema del funcionamiento de un control externo en AIMSUN.
Fuente: TSS-Transport Simulation Systems.

El modelo en AIMSUN de la red vial emula el proceso de detección. Después, a través de una serie de funciones provee a la aplicación externa con las variables de la Simulación (flujo, ocupación, entre otros). La aplicación externa (la aplicación que se desea desarrollar) usa estos datos para alimentar alguna regla de control y decide que acción se tiene que aplicar en el modelo. Finalmente, la APLICACIÓN EXTERNA envía, las correspondientes acciones (por ejemplo: cambiar los tiempos de semáforo, duraciones de ciclo, desplegar mensajes en un Panel de Mensajes Variables, etc.) dentro del modelo, que emula la operación a través de los correspondientes componentes del modelo como señales de tránsito, sistemas de paneles con mensajes variables, etc.

Con base en las características antes mencionadas, Aimsun es el *software* de microsimulación adecuado para emplear en el experimento

3.2.2 Modelo de Pruebas.

Según el diagrama de flujo de la figura anteriormente citado, es necesario elaborar un modelo esquemático de experimentación para poder observar y analizar detalladamente los resultados de los experimentos.

Dicho modelo de vialidades se construirá con base en un grafo dirigido con pesos de arco no negativos compuesto de cinco nodos y nueve arcos, como se muestra en la Figura 3.11.

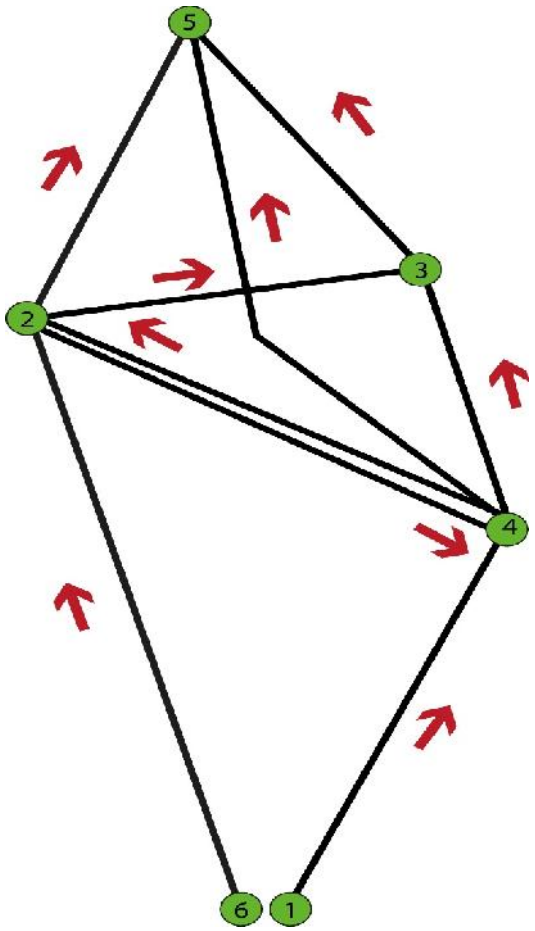

Figura 3.11. Grafo que se utilizará como modelo experimental.

Fuente: Elaboración propia.

En este grafo que posee la forma convencional de representación según la teoría de redes y el cual posee la siguiente matriz de adyacencia:

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

El propósito de este grafo será el de hacer la similitud con una red vial, el cual los vehículos tendrán como origen el nodo1 y como destino el nodo 5. Dicho grafo tendrá que ser representado mediante vialidades en el caso de los arcos y de intersecciones, en el caso de los nodos internos.

Así, el grafo propuesto anteriormente se trasladará a su equivalente dentro del programa, como lo ilustra la Figura 3.12:

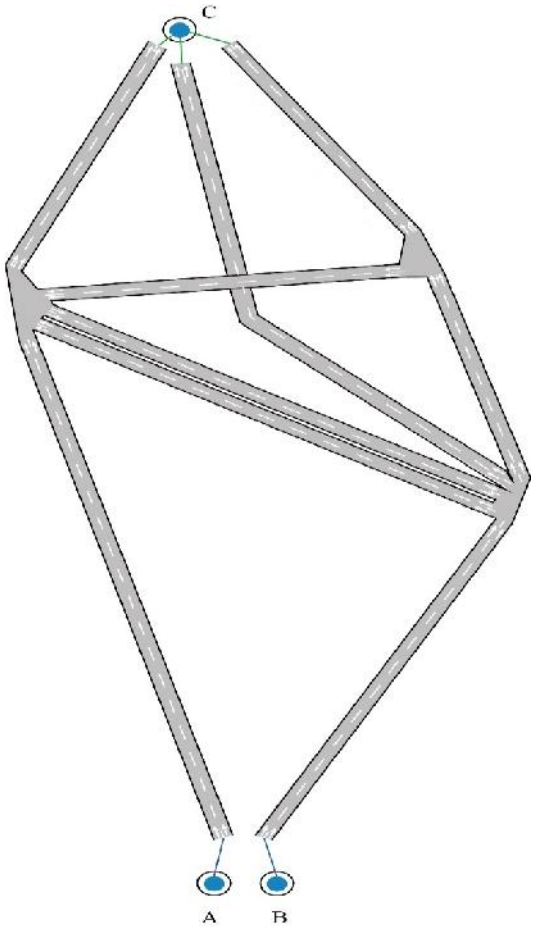

Figura 3.12. Vialidades dibujadas dentro del programa AIMSUN, siguiendo el patrón del grafo base.
Fuente: TSS-Transport Simulation Systems.

Una vez trazadas las vialidades dentro del programa, es necesario establecer las adyacencias de las vialidades en las intersecciones, es decir, tenemos que decirle al programa que vialidades están conectadas entre si.

En la siguiente figura se muestra el detalle del nodo 2, en el cual se definen las adyacencias con las demás vialidades, a través de una intersección.

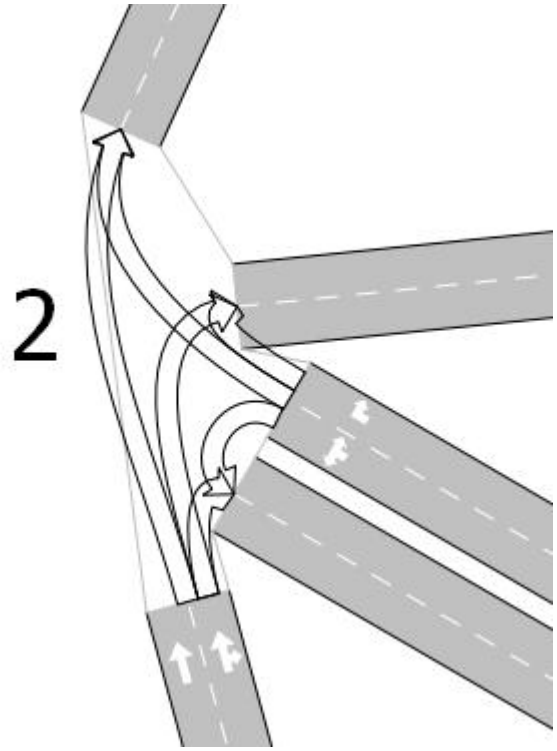

Figura 3.13. Aplicación de adyacencias del nodo 2, en el modelo experimental.
Fuente: TSS-Transport Simulation Systems.

Lo anterior se realizará para los demás nodos del modelo, de manera que sea posible que los vehículos que serán simulados, tengan la posibilidad de girar en las vialidades según la información de la matriz de adyacencias. Una vez terminadas las adyacencias en los nodos, se introducirán los datos de los vehículos. Se necesitarán datos de vehículos para poder asignar características de velocidad, ancho, largo, aceleración y desaceleración a dichos vehículos.

Se simularán únicamente vehículos compactos, únicamente con fines de análisis de los tiempos de viaje de los vehículos, por lo que un solo tipo de vehículos es suficiente.

Tomaremos una tabla realizada de dichos datos de vehículos en el estado de Querétaro (Ramirez, 2012), en la que se muestran los datos de los vehículos

tipo que con mayor frecuencia circulan en las vialidades de la ciudad. Los datos se muestran en la Tabla 3.1.

Auto (A2)	Dimensiones				Potencia		Torque		Aceleración
	Largo (m)	Ancho (m)	Alto (m)	ejes (m)	hp	rpm	lb-ft	rpm	0-100 km/h (s)
Audi A3	4.238	1.765	1.421		150	5700			
Audi S3	4.23	1.765	1.399		261	6000		2500	5.7
Aveo 2009	3.92	1.68	1.505	2.48	94	6200	130	3400	
Aveo Sedan	4.31	1.71	1.505	2.48	94	6200	130	3400	
Chevrolet Agile	3.996	1.939	1.539		91	6000	120	3200	
Chevrolet Astra	4.2	1.989	1.431	2.614	115		125		
Chevrolet Aveo 2012	4.315	1.709	1.506		103	5800	107	3600	
Chevrolet Celta	3.799	1.857	1.408	2.443	91	6000	121	2800	11.6
Chevrolet Corsa	2.491	1.646	1.43	3.822	100	5200	122	2800	
Chevrolet Meriva	4.042	1.944	1.573	2.63	120		125	2800	12.2
Chevrolet Spark	3.64	1.91	1.522	2.375	80	6400	111	4800	
Chevrolet Vectra	4.618	2.017	1.458	2.703	148	5200			
Chevrolet Zafira	4.317	1.742	1.689	2.694	114	5200	125	2400	
Classic Wagon	4.056	1.768	1.448	2.443	91	5600			
Clio Grand Tour	4.228	1.719	1.513						
Cruze	4.597	1.788	1.477	2.685	140	6200	130	3800	
Ford Fiesta 2011	3.95	1.722	1.481		119	6350	112	5000	
Ford Focus 2012	4.358	1.823	1.484		150	6500	146	4450	8.5
Ford Fusion 2010	4.013	1.724	1.543		263	6250	249	4500	7.5
Ford Mustang 2010	4.765	1.877	1.382	2.718	300				5.7
Ford Shelby Cobra	4.292	1.943	1.194	2.405	540	6500	510	6500	4
Ford Taurus	5.154	1.935	1.542	2.867	263				7.2
Golf Plus	4.206	1.759	1.58		105	4000			11.9
Ibiza ST	4.227	1.693	1.445		105	5000	175	4100	
jetta 2010	4.402	1.735	1.438	2.513	115	5400	122	2800	8.9
Sail	4.249	1.69	1.495	2.48	102	6000	131	4200	
Smart	3.495	1.495	1.5	2.34	52	6000	72	4400	
Sonic Sedan	4.399	1.735	1.517	2.525	115	6000	155	4600	
Toyota Auris	4.245	1.76	1.515						
Tsuru 2012	4.325	1.65	1.381		105	6000	102	4000	

Tabla 3.1. Ficha técnica de los modelos de vehículos que transitan con mayor frecuencia en el estado de Querétaro.

Fuente: Ramirez (2012).

Analizaremos los datos de manera que podamos obtener el promedio de los datos y sus desviaciones estándar.

Introduciremos los datos en el programa Minitab, el cual nos dará las estadísticas de los datos requeridos, los resultados se muestran en la Tabla 3.2:

Largo (m)				Ancho (m)				Alto (m)			
Media	Desv. Est.	Max	Min	Media	Desv. Est.	Max	Min	Media	Desv. Est.	Max	Min
4.2415	0.4501	5.15	2.49	1.7595	0.1182	2.01	1.49	1.4895	0.0845	1.68	1.19

Tabla 3.2. Parámetros estadísticos de los vehículos que transitan con mayor frecuencia en el estado de Querétaro.

Fuente: Elaboración propia.

Ahora se introducen dichos datos al programa en el apartado de vehículos, dentro de los datos de demanda de la simulación.

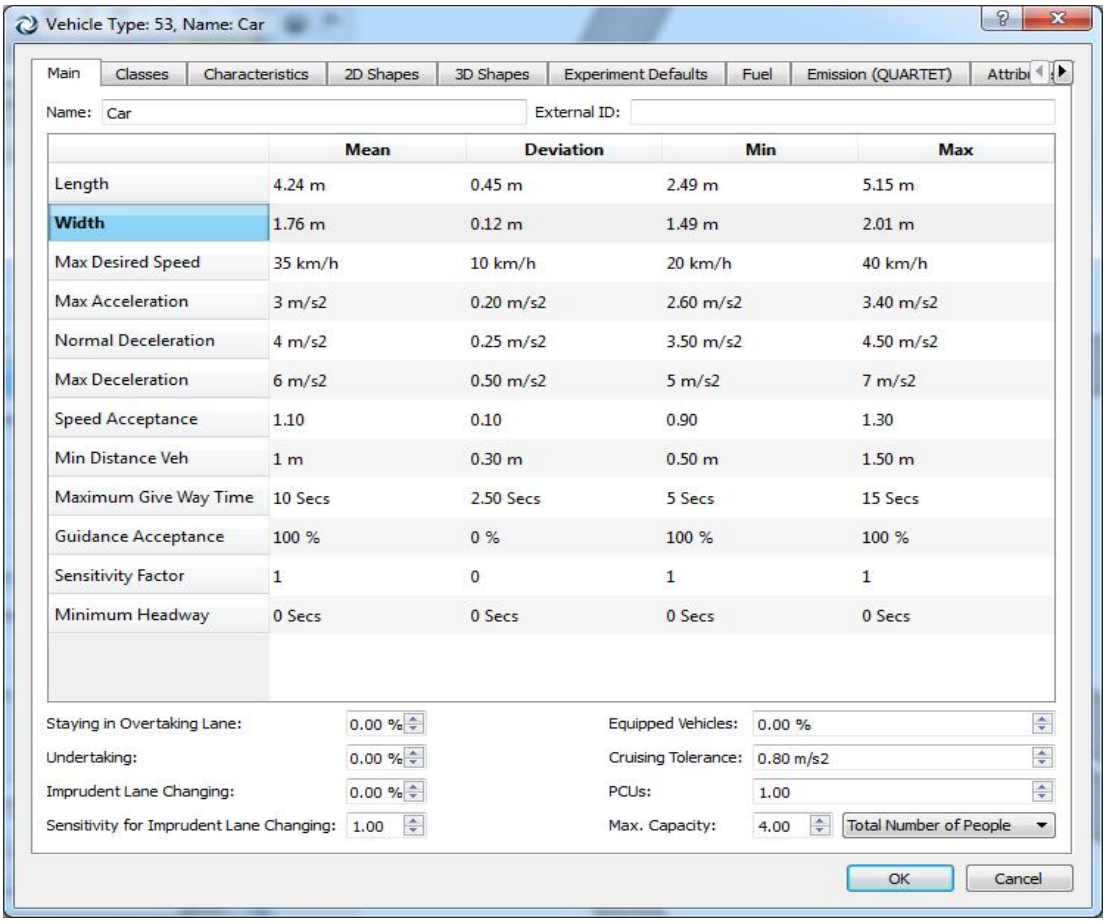

Figura 3.14. Introducción de datos de vehículos en el programa AIMSUN.

Fuente: TSS-Transport Simulation Systems.

Terminado este paso, procederemos a incluir en el programa los datos de la demanda de tráfico, en la cual se simulará durante una hora, con datos de origen destino como se muestra a continuación en la Figura 3.15:

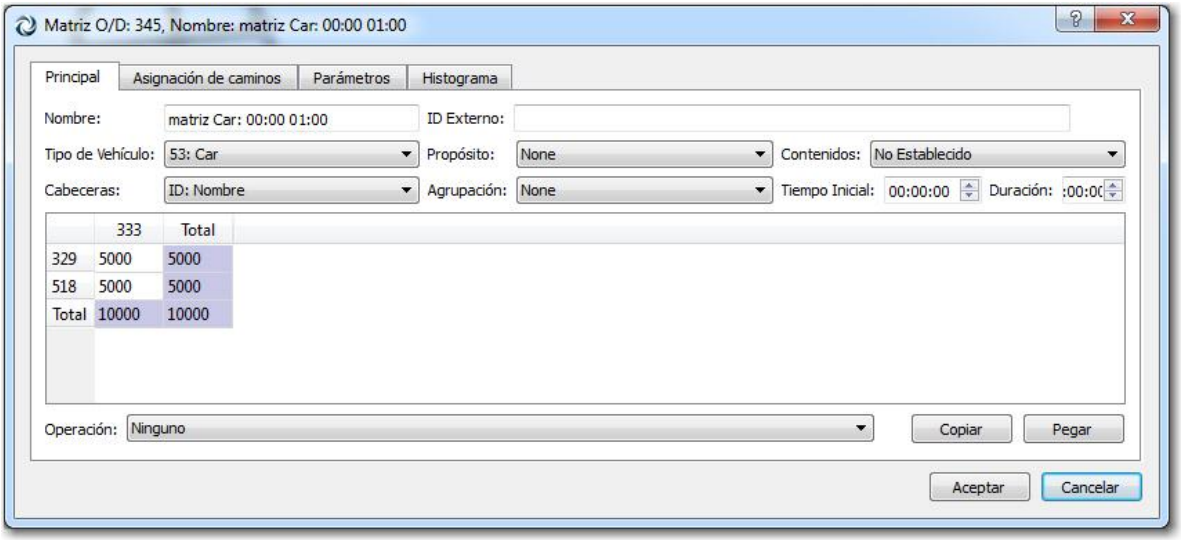

Figura 3.15. Introducción de los datos de demanda de vehículos en el programa AIMSUN.

Fuente: TSS-Transport Simulation Systems.

Se simularán entonces 10,000 vehículos que irán desde el centroide A y B, (con identificadores 329 y 518, respectivamente) al centroide C (con identificador 333), tal como se observa en la Figura 3.12. Se instrumentará un cierto porcentaje de vehículos para analizar los resultados una vez que se implemente el algoritmo para optimizar el tiempo de viaje. La justificación de estos datos es únicamente para que no haya una interrupción de flujo por parte del centroide, de esta manera si al modificar el patrón de flujo de los automóviles se incrementa el flujo de una avenida, existan autos que sigan circulando por dicha vialidad.

Ya que se tienen los datos anteriores dentro del programa, es posible realizar una simulación de prueba, para corroborar que los datos se han metido correctamente.

Para poder comenzar con la simulación, es necesario especificar en las opciones del programa, que se simulará microscópicamente y con un modelo de elección de ruta fijo. Existen diferentes tipos de elección de ruta. Los disponibles en el microsimulador de AISMUN son: proporcionales, logit multimodal y c-logit,

pero el usuario puede definir su propio modelo de elección de ruta, en el editor de funciones. Al correr la simulación con parámetros de elección de ruta fija, únicamente se calcularán los caminos mínimos una sola vez al principio de la simulación. Con estos parámetros, simularemos una vialidad de manera que su comportamiento sea el más parecido a la realidad.

3.2.3 Construcción de las subrutinas del algoritmo enrutador de vehículos.

Como se puede apreciar, se tienen datos de tiempos promedio de viaje de cada arco, esto es posible gracias a los contadores virtuales que incorpora el software AIMSUN. Estos contadores se incorporan en el mismo arco, permitiendo así conocer los tiempos de viaje generalizados de todos los arcos y para todos los vehículos.

El propósito del algoritmo enrutador, será el de obtener los datos de las coordenadas de los vehículos que se enviarán a través de los dispositivos móviles, analizar los datos y calcular la sección en que se encuentran, obtener los tiempos de viaje de las secciones del sistema vial, y a través de un algoritmo de caminos mínimos, obtener los menores tiempos de viaje hacia el destino del vehículo.

Se simularán los procesos críticos del escenario hipotético, por lo que primeramente se tendrán que discriminar entre automóviles instrumentados y automóviles sin instrumentación. Para lograr lo anterior, es necesario desarrollar el programa del control externo que interactuará con los datos de los automóviles simulados por el programa AIMSUN. Dicho de otra manera, se controlará externamente el comportamiento de los vehículos instrumentados, de acuerdo a los datos arrojados por el programa AIMSUN.

A continuación en la Figura 3.16 se presenta un diagrama de flujo del funcionamiento de algoritmo enrutador.

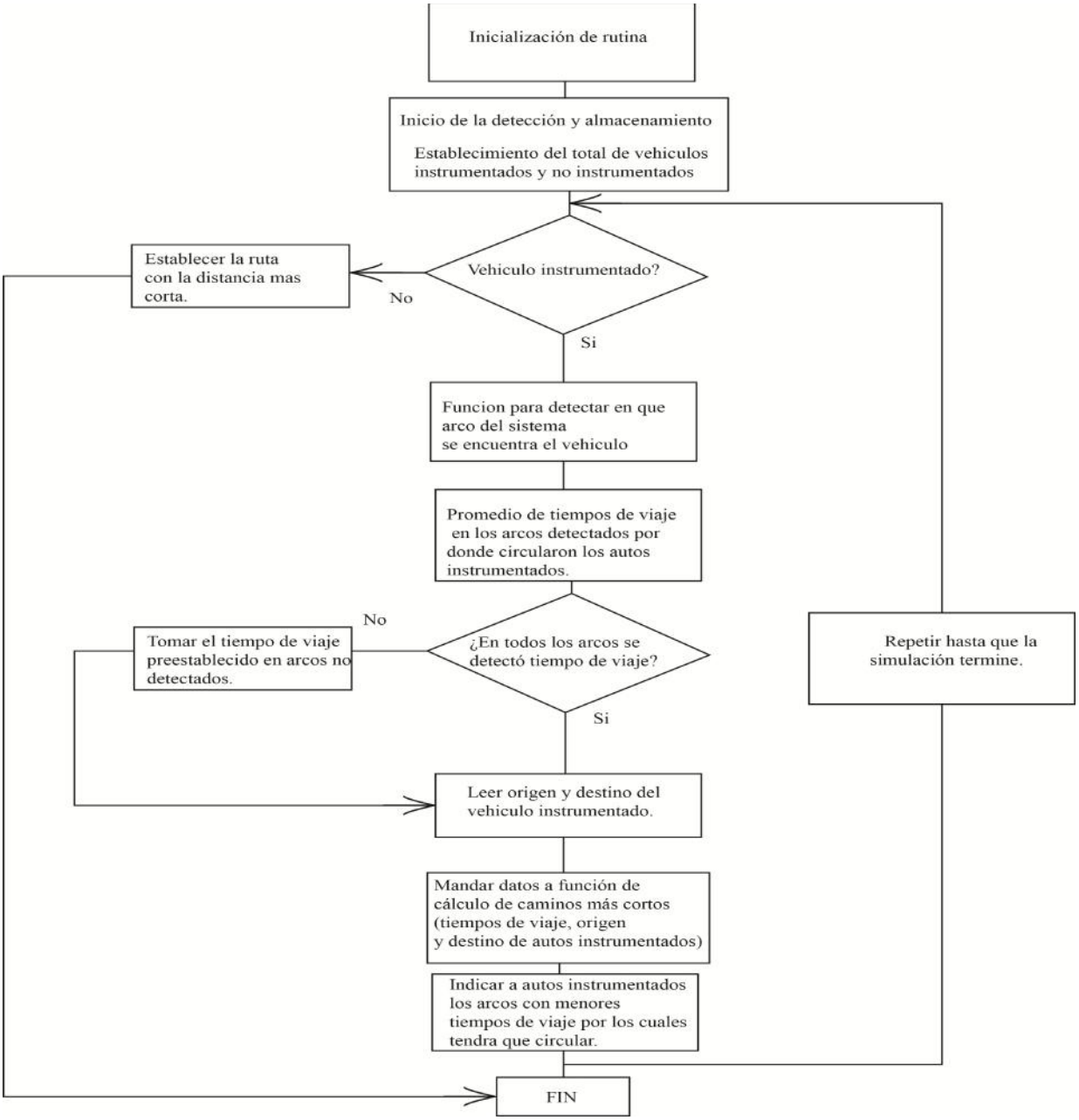

Figura 3.16. Diagrama de flujo del algoritmo enrutador.

Fuente: Elaboración propia.

Para poder comenzar a recopilar los datos que servirán para determinar la ruta idónea de los vehículos instrumentados, es necesario determinar ciertos porcentajes de estos vehículos con instrumentación, explicado en el siguiente apartado.

3.2.3.1 Rutina de asignación de los vehículos instrumentados.

El primer paso será instrumentar un porcentaje del total de los vehículos instrumentados. La asignación propuesta se basará en un proceso iterativo en el que se instrumentará a los vehículos de manera secuencial, como se explica en la Figura 3.17.

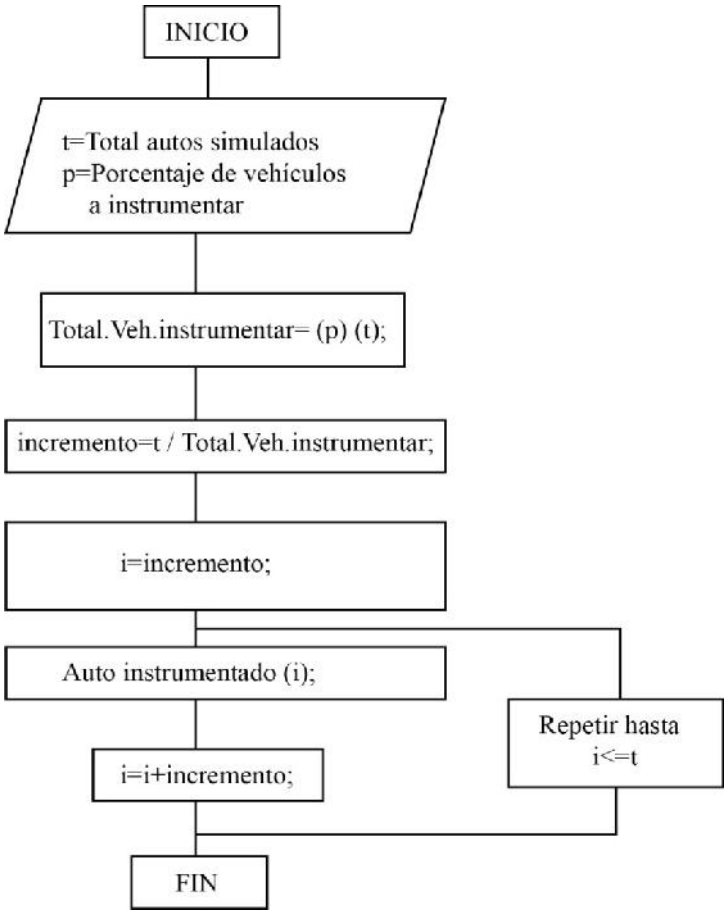

Figura 3.17. Diagrama de flujo que ilustra el proceso de instrumentación de vehículos dentro del programa AIMSUN.

Fuente: Elaboración propia.

La anterior rutina es válida para la variable *i* únicamente para datos de tipo entero.

Una vez que se tiene el porcentaje de autos instrumentados, es necesario ubicarlos en el sistema a través de las coordenadas locales del sistema. Este paso

nos ayudará a ubicar a los vehículos dentro de las vialidades, dando pie a la posibilidad de calcular su entrada y salida de alguna vialidad determinada.

3.2.3.2 Rutina de ubicación espacial de vehículos.

Dentro de las funciones internas del programa de microsimulación, es posible obtener las coordenadas de los vehículos instrumentados, de acuerdo a la posición del grafo en el espacio. La anterior funcionalidad es necesaria para dotar al control externo que programaremos dentro del programa para hacer una similitud con el sistema hipotético planteado.

Para poder ubicar la posición del vehículo dentro del sistema a través de sus coordenadas, necesitaremos identificar las secciones existentes en nuestro modelo de prueba y subsecuentemente añadir un identificador único a dicha sección.

Una sección entonces, quedará definida como el subtramo de 1 o más carriles de un tramo de la vialidad, acotado por la unión de este con 2 o más subtramos de la vialidad; siendo el inicio y el final de esta delimitada por la dirección del tráfico establecida.

De esta manera, se facilita la localización de los vehículos por secciones y es posible dar un mejor seguimiento al vehículo.

En la Figura 3.19 se muestran los identificadores que se manejarán en el programa para cada sección.

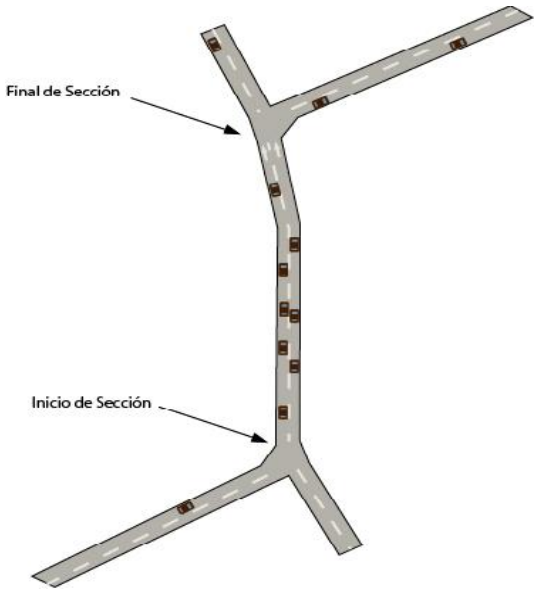

Figura 3.18. Esquema de la delimitación de una sección vial.

Fuente: Elaboración propia.

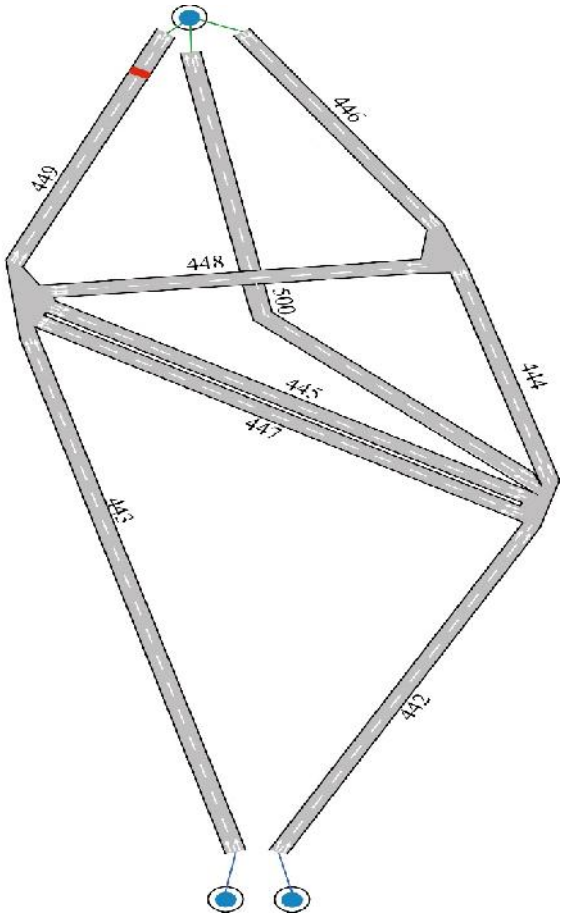

Figura 3.19. Claves de secciones del modelo de pruebas.

Fuente: Elaboración propia.

Ya que se han identificado las secciones (Figura 3.18), es necesario crear variables que puedan servir para identificar la sección en que se encuentra el vehículo con base en sus coordenadas, para lo que es necesario crear una retícula sobre estas secciones y crear las variables mencionadas. A continuación en la Figura 3.20 se muestra la retícula elaborada para construir las variables que servirán para determinar la sección en la que se encuentran los vehículos a partir de sus coordenadas X y Y.

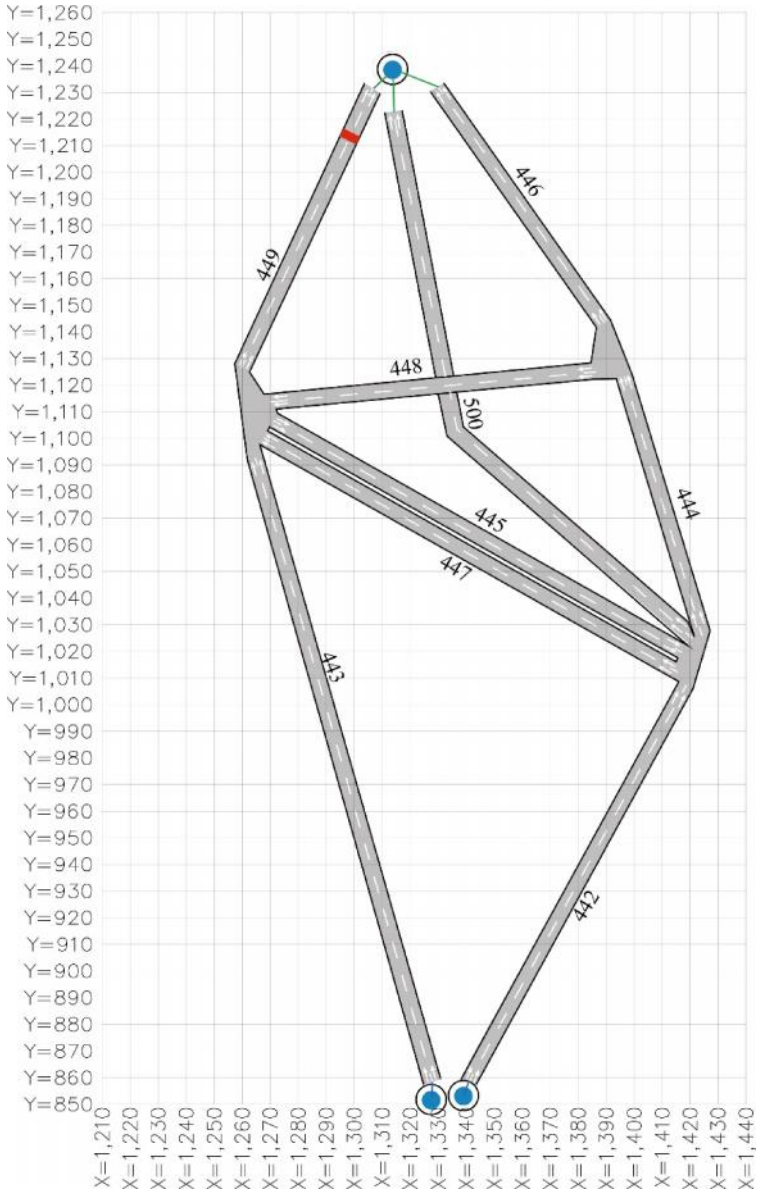

Figura 3.20. Grafo dentro de una cuadrícula para localización de secciones mediante coordenadas.

Fuente: Elaboración propia.

Con esta cuadricula realizada sobre el modelo de prueba de la vialidad, se construyen las siguientes variables:

coordenadax[1250]=449;
coordenadax[1260]=443,449,448;
coordenadax[1270]=443,447,445,448,449;
coordenadax[1280]=443,447,445,448,449;
coordenadax[1290]=443,447,445,448,449;
coordenadax[1300]=443,447,445,448,449;
coordenadax[1310]=443,447,445,448,500,514;
coordenadax[1320]=443,482,445,447,448,500,450;
coordenadax[1330]=443,482,445,447,448,500,450;
coordenadax[1340]=485,447,445,500,448,450,446;
coordenadax[1350]=442,447,445,500,448,450,446;
coordenadax[1360]=442,447,445,500,448,450,446;
coordenadax[1370]=442,447,445,500,448,450,446;
coordenadax[1380]=442,447,445,448,450,446;
coordenadax[1390]=442,447,445,500,444;
coordenadax[1400]=442,447,445,444;
coordenadax[1410]=442,447,445,444;
coordenadax[1420]=444;
coordenaday[860]=482,485;
coordenaday[870]=482,485;
coordenaday[880]=443,442;
coordenaday[890]=443,442;
coordenaday[900]=443,442;
coordenaday[910]=443,442;
coordenaday[920]=443,442;
coordenaday[930]=443,442;
coordenaday[940]=443,442;

coordenaday[960]=443,442;
coordenaday[980]=443,442;
coordenaday[1000]=443,442;
coordenaday[1010]=443,442;
coordenaday[1020]=443,447,445;
coordenaday[1030]=443,447,445,500,444;
coordenaday[1040]=443,447,445,500,444;
coordenaday[1050]=443,447,445,500,444;
coordenaday[1060]=443,447,445,500,444;
coordenaday[1070]=443,447,445,500,444;
coordenaday[1080]=443,447,445,500,444;
coordenaday[1090]=443,447,445,500,444;
coordenaday[1100]=443,449,445,500,444;
coordenaday[1110]=449,448,500,444;
coordenaday[1120]=449,442,448,500,444;
coordenaday[1130]=449,500,446;
coordenaday[1140]=449,500,446;
coordenaday[1150]=449,500,446;
coordenaday[1160]=449,500,446;
coordenaday[1170]=449,500,446;
coordenaday[1180]=449,500,446;
coordenaday[1190]=449,500,446;
coordenaday[1200]=449,500,446;
coordenaday[1210]=449,500,446;

Las coordenadas se marcaron en la cuadrícula a cada 10 metros, ya que los cambios entre secciones no son tan continuos.

Las variables coordenada “x” y coordenada “y” son variables de tipo *array*, las cuales pueden ser de una, 2 y hasta 3 dimensiones. En este caso los arreglos para cada ordenada X o Y son de una sola dimensión, y se le asigna a cada

dimensión el identificador o identificadores de la secciones que se encuentren en el rango de coordenadas.

Por ejemplo, en el rango de la coordenada y=1060, se encuentran las secciones: 443,447,445,500,444. De igual manera se realiza para el eje x. Una vez obtenidas las coordenadas se realizó una rutina para tomar los datos de las coordenadas de vehículos instrumentados y determinar la sección en que se encuentran.

El código de la rutina se encuentra en los anexos, y se presenta en diagrama de flujo en la Figura 3.21.

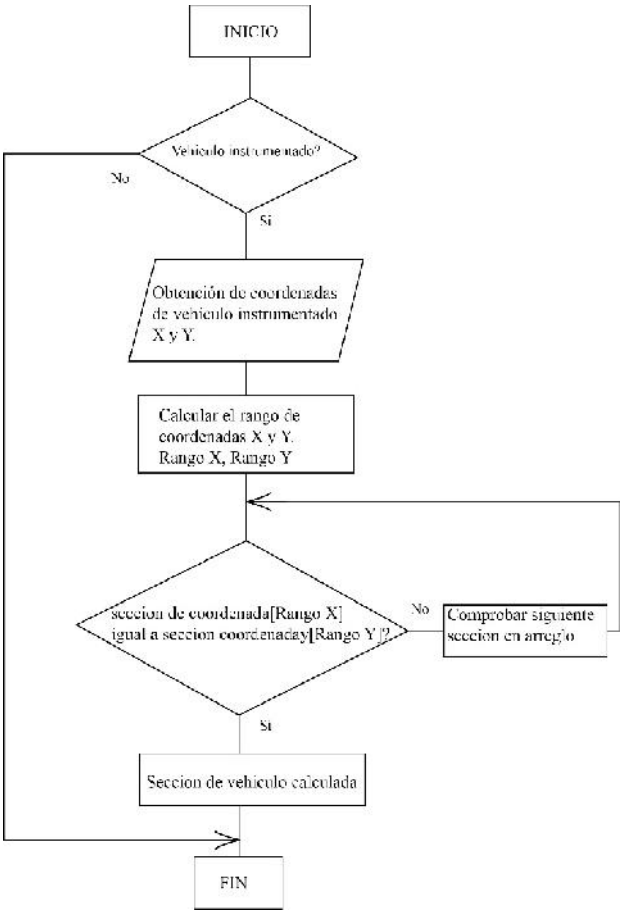

Figura 3.21. Diagrama de flujo para la obtención de la sección en la que se encuentra el automóvil mediante sus coordenadas.

Fuente: Elaboración propia.

3.2.3.3 Rutina de cálculo de tiempos de viaje.

Una vez que se conoce la sección en la que se encuentran los autos instrumentados para el tiempo de simulación, es posible determinar los tiempos de viaje de los automóviles haciendo la diferencia tomando el tiempo de salida menos el tiempo de entrada del vehículo a cada sección. Se asignará a cada sección una variable que contenga el promedio del tiempo de viaje todos los automóviles instrumentados que circulen por dicha sección. Se presenta un diagrama de flujo en la Figura 3.22.

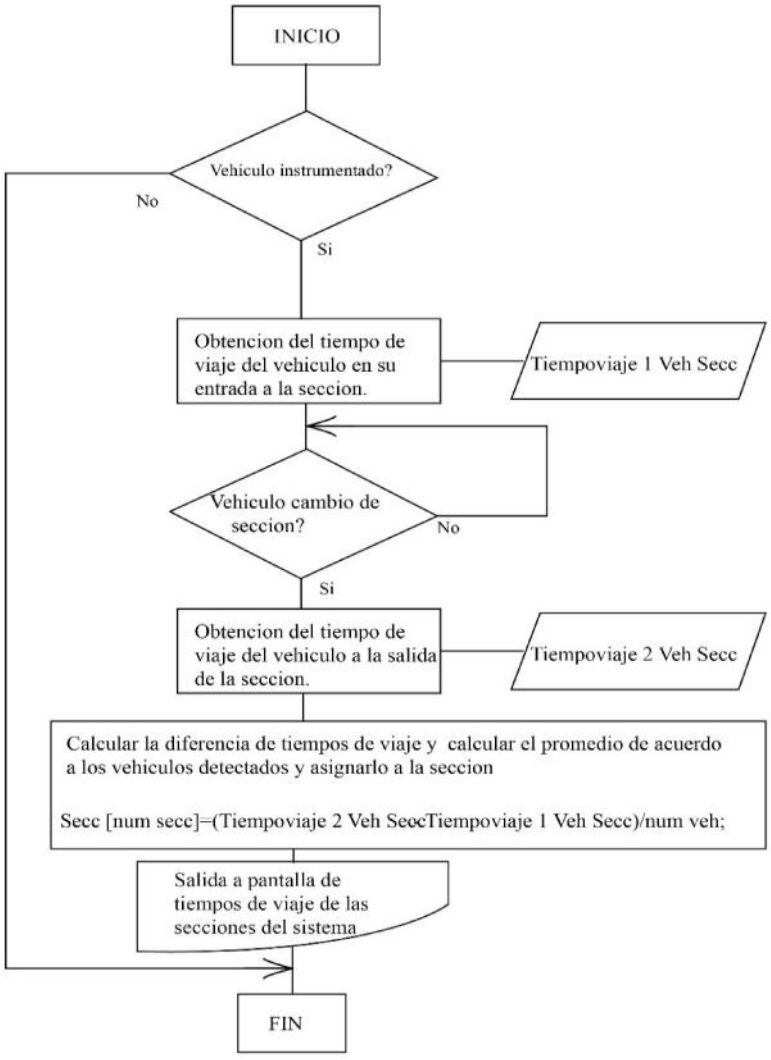

Figura 3.22. Diagrama de flujo del algoritmo enrutador.

Fuente: Elaboración propia.

3.2.3.4 Rutina de cálculo de rutas con menor tiempo de viaje.

Ya que se obtuvieron los tiempos de viajes para las vialidades, es necesario calcular la ruta óptima por la cual deberán circular los automóviles instrumentados. Se le deberá requerir a cada vehículo el nodo del cual procede y al cual se dirige, para poder realizar el cálculo de la ruta con menor tiempo de viaje, por medio del algoritmo Dijkstra. Se presenta un diagrama de flujo en la Figura 3.23, conteniendo la explicación anterior.

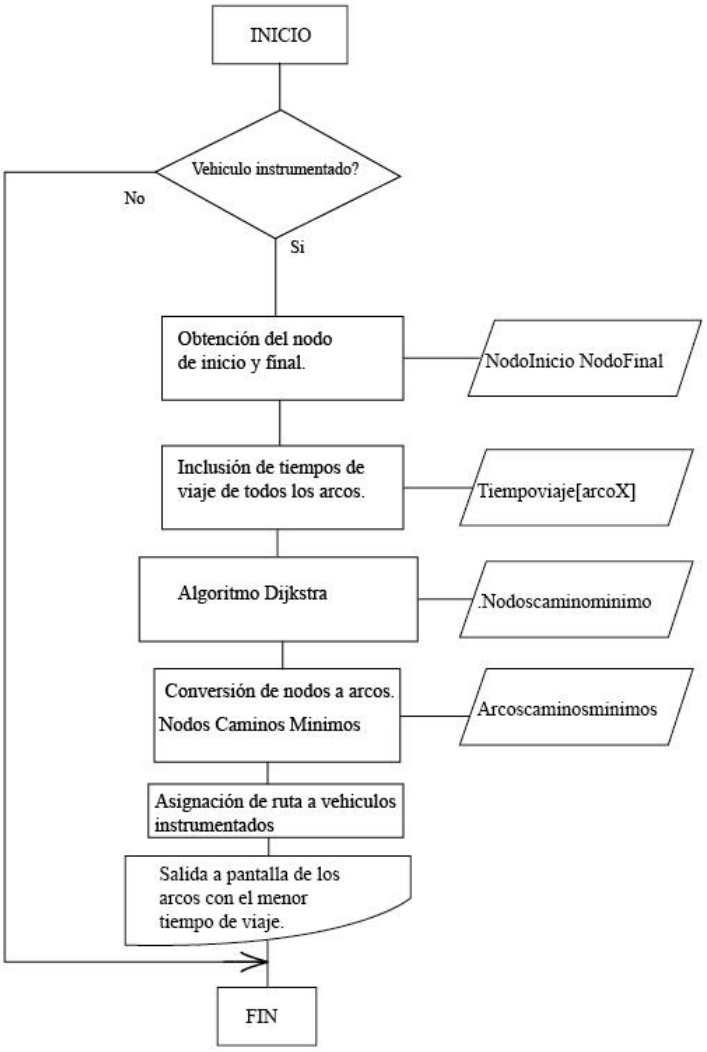

Figura 3.23. Diagrama de flujo del cálculo de ruta con menor tiempo de viaje.

Fuente: Elaboración propia.

En este paso, se hace una consulta a cada vehículo para conocer el nodo del que procede y al nodo que se dirige, con esta información es posible aplicar el algoritmo Dijkstra, con la información previa de los tiempos de viaje de todas las secciones del sistema vial. El resultado del algoritmo Dijkstra son los nodos cuyos arcos tienen el menor tiempo de viaje, por lo que para poder guiar al automóvil por la ruta con menor tiempo de viaje es necesario convertir los nodos a arcos, para poder asignar la ruta con el menor tiempo de viaje.

Finalmente se asigna a la ruta del automóvil por la cual deberá circular. Cabe destacar que es un proceso dinámico en el cual se realiza el cálculo de la ruta óptima en cada paso de la simulación, por lo que se utilizan los tiempos de viaje de las secciones viales según varíen, lo cual establece que el sistema trabaja en tiempo real.

Con base en las rutinas construidas, se ensambla una rutina en conjunto, la cual es el motor del algoritmo enrutador. Se presenta a continuación la simulación del sistema y la comparativa con distintos métodos de asignación de tráfico.

3.3 Simulación del Sistema Enrutador de Vehículos.

Para poder establecer que existe una ventaja en el sistema enrutador de vehículos desarrollado en este trabajo, es necesario compararlo con los modelos de asignación que usualmente se utilizan para asemejar el comportamiento de los vehículos dentro del sistema vial.

3.3.1 Simulación con el Modelo de Asignación Fijo con Tiempos de Viaje a Flujo Libre.

Primeramente, se simuló el modelo de pruebas a través del modelo de asignación fijo con tiempos de viaje a flujo libre (FTVFL). Este método de asignación es poco utilizado, ya que no es muy acertado en cuanto al comportamiento de los vehículos en el sistema vial, pero se utilizó ya que este

modelo genera que las vías con la menor distancia se congestionen, lo cual es reflejo de la realidad en los caminos de una ciudad. La simulación se realizó tres veces y los resultados se presentan en el apartado 4.1.

3.3.2 Simulación con el Modelo de Asignación Equilibrio Dinámico de Usuario.

Seguidamente, se simuló el modelo de pruebas con la asignación de tráfico Equilibrio Dinámico de Usuario (EDU). Como anteriormente se explicó, este método realiza iteraciones con base en los tiempos de viaje de los arcos utilizados por los vehículos, por lo que "equilibra" el número de vehículos que circula por cada sección. Se realizó la simulación tres veces y los resultados se presentan en el apartado 4.2.

3.3.3 Simulación con el Sistema Enrutador de Vehículos.

Finalmente, se simuló el modelo de pruebas con el algoritmo desarrollado, el Sistema Enrutador de Vehículos (SEV). La simulación se realizó tres veces y se realizó en el entorno del modelo de asignación fijo con tiempos de viaje a flujo libre. Se instrumentaron del 10 al 100 por ciento de vehículos con incrementos de 10%. Los resultados y comparativas con otros modelos de asignación se presentan en el apartado 4.3.

3.3 Simulación del Sistema Enrutador de Vehículos en Vialidades de la Ciudad de Querétaro.

Como parte de una mejor apreciación de los resultados que se pueden obtener por parte del SEV, se construyó un grafo de ciertas vialidades de la ciudad de Querétaro, con el objetivo de darle un enfoque más práctico al sistema antes construido.

En la siguiente figura (Figura 3.24) se muestran las vialidades a simularse con el SEV y los centroides de dichas vialidades.

Figura 3.23. Vialidades de la ciudad de Querétaro a simular.

Fuente: Elaboración propia.

La construcción de las vialidades se realizó con ayuda del programa Google Earth, para la determinación de anchos de carriles y sentidos de circulación.

Se simularon tres escenarios:

a) Simulación realizada con el Modelo de Asignación Fijo con Tiempos de Viaje a Flujo Libre.
b) Simulación realizada con el Modelo de Asignación de Equilibrio Dinámico de Usuario.
c) Simulación realizada con el Sistema Enrutador de Vehículos.

Para la simulación del Sistema Enrutador de Vehículos se instrumentaron el 100% de vehículos.

Los resultados se pueden observar en el apartado 4.2.

4. RESULTADOS

A continuación se presentan las tablas con los resultados de las simulaciones realizadas.

4.1 Resultados de la simulación en el Modelo de Pruebas.

4.1.1 De la simulación con el Modelo de Asignación Fijo con Tiempos de Viaje a Flujo Libre.

En la Tabla 4.1, se muestran los resultados de la simulación del modelo de pruebas con el modelo de asignación fijo con tiempos de viaje a flujo libre.

Serie Temporal	Valor	Desviación Estándar	Unidades
Cola Virtual Máxima Car	5676	ND	vehs
Cola Virtual Media Car	2793.86	ND	vehs
Densidad Car	344.5	ND	veh/km
Distancia total recorrida Car	1668.82	ND	km
Flujo Car	4146	ND	veh/h
Longitud media de cola Car	84.31	ND	vehs
Número de Paradas Car	4.82	ND	
Tiempo de Demora Car	2155.29	1665.39	seg/km
Tiempo de Parada Car	2117.28	1669.62	seg/km
Tiempo de Viaje Car	2272.75	1665.5	seg/km
Tiempo total de viaje Car	1037.95	ND	h
Vehículos Dentro Car	163	ND	vehs
Vehículos Errantes Dentro Car	0	ND	vehs
Vehículos Errantes Fuera Car	0	ND	vehs
Vehículos Esperando para Entrar Car	5676	ND	vehs
Vehículos Fuera Car	4146	ND	vehs
Velocidad Car	17.59	8.25	km/h
Velocidad Harmónica Car	10.01	8.71	km/h

Tabla 4.1. Resultados de la simulación con el modelo de asignación fijo con tiempos de viaje a flujo libre.

Fuente: Elaboración propia.

En esta tabla destacan las variables Flujo Car, la cual contabiliza el número de vehículos por hora que circularon por las vialidades del modelo de pruebas. En esta simulación el flujo es de 4,146 veh/h. La variable Tiempo de viaje Car

contiene el tiempo promedio en el que un vehículo transita por un kilómetro dentro de la red, para esta simulación el valor es de 2,272.25 seg/km. Este es el promedio de todos los tiempos de viaje, para cada vehículo que ha atravesado la red, convertido a tiempo/kilómetro. La variable Tiempo Total de Viaje es el total de la suma del tiempo de viaje que experimentado por todos los vehículos que han sido simulados en la red, que en esta simulación resultó ser de 1,037.95 horas.

4.1.2 De la simulación con el Modelo de Asignación Equilibrio Dinámico de Usuario.

Serie Temporal	Valor	Desviación Estándar	Unidades
Cola Virtual Máxima Car	3483	ND	vehs
Cola Virtual Media Car	2046.51	ND	vehs
Densidad Car	468.09	ND	veh/km
Distancia total recorrida Car	2732.65	ND	km
Flujo Car	6873	ND	veh/h
Longitud media de cola Car	2.22	ND	vehs
Número de Paradas Car	3.21	ND	
Tiempo de Demora Car	1860.1	895.61	seg/km
Tiempo de Parada Car	1818.56	895.44	seg/km
Tiempo de Viaje Car	1978.15	895.81	seg/km
Tiempo total de viaje Car	1499.14	ND	h
Vehículos Dentro Car	128	ND	vehs
Vehículos Errantes Dentro Car	0	ND	vehs
Vehículos Errantes Fuera Car	0	ND	vehs
Vehículos Esperando para Entrar Car	3483	ND	vehs
Vehículos Fuera Car	6873	ND	vehs
Velocidad Car	22.28	2.09	km/h
Velocidad Harmónica Car	22.09	2.06	km/h

Tabla 4.2. Resultados de la simulación con el modelo de asignación Equilibrio Dinámico de Usuario.

Fuente: Elaboración propia.

Para la simulación con el modelo de EDU se obtuvieron valores más bajos de tiempo de viaje y un flujo mayor. En la Tabla 4.2, se aprecia que para la variable Flujo Car, el total de vehículos que circularon por el sistema fueron 6,837, un 65% mayor que con el modelo de asignación FTVFL. Para la variable Tiempo de Viaje Car, el promedio fue de 1,978.15, un 13% menor que con el modelo de asignación FTVFL. La variable Tiempo de viaje la cual resultó ser de 1,499.14h para todos los vehículos es un 40% mayor que la de FTVFL, debido al incremento

del flujo presentado en el sistema. En este caso, la variable Tiempo de Viaje es de las más representativas, ya que es un indicador de un flujo óptimo de vehículos.

4.1.3 Resultados de la simulación con el Sistema Enrutador de Vehículos.

Al igual que las simulaciones anteriores, se realizó tres veces el proceso, pero en esta ocasión solamente se instrumentó un cierto porcentaje de vehículos, hasta alcanzar el 100% de vehículos instrumentados, los cuales tienen la información de los tiempos de viaje de todas las secciones y por consiguiente, sus rutas son las óptimas.

La Tabla 4.3 muestra los resultados para el 10% de autos instrumentados.

Serie Temporal	Valor	Desviación Estándar	Unidades
Cola Virtual Máxima Car	5601	N/A	vehs
Cola Virtual Media Car	2761.85	N/A	vehs
Densidad Car	356.68	N/A	veh/km
Distancia total recorrida Car	1713.86	N/A	km
Flujo Car	4225	N/A	veh/h
Longitud media de cola Car	74.26	N/A	vehs
Número de Paradas Car	6.29	N/A	
Tiempo de Demora Car	2197.57	1635.89	sec/km
Tiempo de Parada Car	2148.73	1624.95	sec/km
Tiempo de Viaje Car	2314.96	1635.74	sec/km
Tiempo total de viaje Car	1087.86	N/A	h
Vehículos Dentro Car	159	N/A	vehs
Vehículos Errantes Dentro Car	0	N/A	vehs
Vehículos Errantes Fuera Car	0	N/A	vehs
Vehículos Esperando para Entrar Car	5601	N/A	vehs
Vehículos Fuera Car	4225	N/A	vehs
Velocidad Car	17.59	8.41	km/h
Velocidad Harmónica Car	10.5	8.63	km/h

Tabla 4.3. Resultados de la simulación con el 10% de vehículos instrumentados.

Fuente: Elaboración propia.

La Tabla 4.4 muestra los resultados para el 20% de autos instrumentados.

Serie Temporal	Valor	Desviación Estándar	Unidades
Cola Virtual Máxima Car	5554	N/A	vehs
Cola Virtual Media Car	2745.56	N/A	vehs
Densidad Car	367.8	N/A	veh/km
Distancia total recorrida Car	1751.12	N/A	km
Flujo Car	4280	N/A	veh/h
Longitud media de cola Car	69.8	N/A	vehs
Número de Paradas Car	6.9	N/A	
Tiempo de Demora Car	2223.39	1582.04	sec/km
Tiempo de Parada Car	2174.7	1572.75	sec/km
Tiempo de Viaje Car	2340.65	1581.74	sec/km
Tiempo total de viaje Car	1127.32	N/A	h
Vehículos Dentro Car	151	N/A	vehs
Vehículos Errantes Dentro Car	0	N/A	vehs
Vehículos Errantes Fuera Car	0	N/A	vehs
Vehículos Esperando para Entrar Car	5554	N/A	vehs
Vehículos Fuera Car	4280	N/A	vehs
Velocidad Car	17.3	8.29	km/h
Velocidad Harmónica Car	10.99	8.32	km/h

Tabla 4.4. Resultados de la simulación con el 20% de vehículos instrumentados.

Fuente: Elaboración propia.

La Tabla 4.5 muestra los resultados para el 30% de autos instrumentados.

Serie Temporal	Valor	Desviación Estándar	Unidades
Cola Virtual Máxima Car	5389	ND	vehs
Cola Virtual Media Car	2655.48	ND	vehs
Densidad Car	379.5	ND	veh/km
Distancia total recorrida Car	1839.63	ND	km
Flujo Car	4446	ND	veh/h
Longitud media de cola Car	62.43	ND	vehs
Número de Paradas Car	7.01	ND	
Tiempo de Demora Car	2187.53	1523.07	seg/km
Tiempo de Parada Car	2140.09	1514.77	seg/km
Tiempo de Viaje Car	2304.94	1523.03	seg/km
Tiempo total de viaje Car	1167.65	ND	h
Vehículos Dentro Car	150	ND	vehs
Vehículos Errantes Dentro Car	0	ND	vehs
Vehículos Errantes Fuera Car	0	ND	vehs
ehículos Esperando para Entrar C	5389	ND	vehs
Vehículos Fuera Car	4446	ND	vehs
Velocidad Car	17.17	7.97	km/h
Velocidad Harmónica Car	11.89	7.93	km/h

Tabla 4.5. Resultados de la simulación con el 30% de vehículos instrumentados.

Fuente: Elaboración propia.

La Tabla 4.6 muestra los resultados para el 40% de autos instrumentados.

Serie Temporal	Valor	Desviación Estándar	Unidades
Cola Virtual Máxima Car	5228	ND	vehs
Cola Virtual Media Car	2575.48	ND	vehs
Densidad Car	391.71	ND	veh/km
Distancia total recorrida Car	1935.84	ND	km
Flujo Car	4619	ND	veh/h
Longitud media de cola Car	54.77	ND	vehs
Número de Paradas Car	7.33	ND	
Tiempo de Demora Car	2166.01	1439.33	seg/km
Tiempo de Parada Car	2118.65	1431.66	seg/km
Tiempo de Viaje Car	2283.32	1439.46	seg/km
Tiempo total de viaje Car	1218.7	ND	h
Vehículos Dentro Car	138	ND	vehs
Vehículos Errantes Dentro Car	0	ND	vehs
Vehículos Errantes Fuera Car	0	ND	vehs
ehículos Esperando para Entrar C	5228	ND	vehs
Vehículos Fuera Car	4619	ND	vehs
Velocidad Car	16.86	7.08	km/h
Velocidad Harmónica Car	12.86	7.17	km/h

Tabla 4.6. Resultados de la simulación con el 40% de vehículos instrumentados.

Fuente: Elaboración propia.

La Tabla 4.7 muestra los resultados para el 50% de autos instrumentados.

Serie Temporal	Valor	Desviación Estándar	Unidades
Cola Virtual Máxima Car	4909	ND	vehs
Cola Virtual Media Car	2455.16	ND	vehs
Densidad Car	409.51	ND	veh/km
Distancia total recorrida Car	2101.19	ND	km
Flujo Car	4945	ND	veh/h
Longitud media de cola Car	43.02	ND	vehs
Número de Paradas Car	6.8	ND	
Tiempo de Demora Car	2104.01	1287.72	seg/km
Tiempo de Parada Car	2058.59	1283.61	seg/km
Tiempo de Viaje Car	2221.38	1287.61	seg/km
Tiempo total de viaje Car	1287.45	ND	h
Vehículos Dentro Car	132	ND	vehs
Vehículos Errantes Dentro Car	0	ND	vehs
Vehículos Errantes Fuera Car	0	ND	vehs
ehículos Esperando para Entrar C	4908	ND	vehs
Vehículos Fuera Car	4945	ND	vehs
Velocidad Car	17.54	6.16	km/h
Velocidad Harmónica Car	14.6	6.55	km/h

Tabla 4.7. Resultados de la simulación con el 50% de vehículos instrumentados.

Fuente: Elaboración propia.

La Tabla 4.8 muestra los resultados para el 60% de autos instrumentados.

Serie Temporal	Valor	Desviación Estándar	Unidades
Cola Virtual Máxima Car	4390	ND	vehs
Cola Virtual Media Car	2186.44	ND	vehs
Densidad Car	403.44	ND	veh/km
Distancia total recorrida Car	2368.48	ND	km
Flujo Car	5478	ND	veh/h
Longitud media de cola Car	21.16	ND	vehs
Número de Paradas Car	4.97	ND	
Tiempo de Demora Car	1847.01	1106.29	seg/km
Tiempo de Parada Car	1804.16	1104.71	seg/km
Tiempo de Viaje Car	1964.44	1106.49	seg/km
Tiempo total de viaje Car	1281.88	ND	h
Vehículos Dentro Car	117	ND	vehs
Vehículos Errantes Dentro Car	0	ND	vehs
Vehículos Errantes Fuera Car	0	ND	vehs
ehículos Esperando para Entrar C	4390	ND	vehs
Vehículos Fuera Car	5478	ND	vehs
Velocidad Car	19.72	4.68	km/h
Velocidad Harmónica Car	18.19	5.27	km/h

Tabla 4.8. Resultados de la simulación con el 60% de vehículos instrumentados.

Fuente: Elaboración propia.

La Tabla 4.9 muestra los resultados para el 70% de autos instrumentados.

Serie Temporal	Valor	Desviación Estándar	Unidades
Cola Virtual Máxima Car	4314	ND	vehs
Cola Virtual Media Car	2151.15	ND	vehs
Densidad Car	401.37	ND	veh/km
Distancia total recorrida Car	2432.48	ND	km
Flujo Car	5542	ND	veh/h
Longitud media de cola Car	12.43	ND	vehs
Número de Paradas Car	4.42	ND	
Tiempo de Demora Car	1777.37	1073	seg/km
Tiempo de Parada Car	1735.45	1072.15	seg/km
Tiempo de Viaje Car	1894.97	1073.05	seg/km
Tiempo total de viaje Car	1270.53	ND	h
Vehículos Dentro Car	129	ND	vehs
Vehículos Errantes Dentro Car	0	ND	vehs
Vehículos Errantes Fuera Car	0	ND	vehs
ehículos Esperando para Entrar C	4314	ND	vehs
Vehículos Fuera Car	5542	ND	vehs
Velocidad Car	20.65	3.66	km/h
Velocidad Harmónica Car	19.84	4	km/h

Tabla 4.9. Resultados de la simulación con el 70% de vehículos instrumentados.

Fuente: Elaboración propia.

La Tabla 4.10 muestra los resultados para el 80% de autos instrumentados.

Serie Temporal	Valor	Desviación Estándar	Unidades
Cola Virtual Máxima Car	4168	ND	vehs
Cola Virtual Media Car	2093.55	ND	vehs
Densidad Car	401.72	ND	veh/km
Distancia total recorrida Car	2539.79	ND	km
Flujo Car	5700	ND	veh/h
Longitud media de cola Car	9.14	ND	vehs
Número de Paradas Car	3.86	ND	
Tiempo de Demora Car	1705.98	1033.32	seg/km
Tiempo de Parada Car	1665.95	1033.46	seg/km
Tiempo de Viaje Car	1823.53	1033.63	seg/km
Tiempo total de viaje Car	1276.91	ND	h
Vehículos Dentro Car	117	ND	vehs
Vehículos Errantes Dentro Car	0	ND	vehs
Vehículos Errantes Fuera Car	0	ND	vehs
ehículos Esperando para Entrar C	4168	ND	vehs
Vehículos Fuera Car	5700	ND	vehs
Velocidad Car	21.41	3.33	km/h
Velocidad Harmónica Car	20.76	3.68	km/h

Tabla 4.10. Resultados de la simulación con el 80% de vehículos instrumentados.

Fuente: Elaboración propia.

La Tabla 4.11 muestra los resultados para el 90% de autos instrumentados.

Serie Temporal	Valor	Desviación Estándar	Unidades
Cola Virtual Máxima Car	3971	ND	vehs
Cola Virtual Media Car	1981.62	ND	vehs
Densidad Car	393.44	ND	veh/km
Distancia total recorrida Car	2661.32	ND	km
Flujo Car	5894	ND	veh/h
Longitud media de cola Car	3.71	ND	vehs
Número de Paradas Car	2.99	ND	
Tiempo de Demora Car	1583.15	948.63	seg/km
Tiempo de Parada Car	1542.64	948.26	seg/km
Tiempo de Viaje Car	1700.71	949	seg/km
Tiempo total de viaje Car	1251.63	ND	h
Vehículos Dentro Car	120	ND	vehs
Vehículos Errantes Dentro Car	0	ND	vehs
Vehículos Errantes Fuera Car	0	ND	vehs
ehículos Esperando para Entrar C	3971	ND	vehs
Vehículos Fuera Car	5894	ND	vehs
Velocidad Car	22.28	2.61	km/h
Velocidad Harmónica Car	21.89	2.91	km/h

Tabla 4.11. Resultados de la simulación con el 90% de vehículos instrumentados.

Fuente: Elaboración propia.

La Tabla 4.12 muestra los resultados para el 100% de autos instrumentados.

Serie Temporal	Valor	Desviación Estándar	Unidades
Cola Virtual Máxima Car	3898	ND	vehs
Cola Virtual Media Car	1950.96	ND	vehs
Densidad Car	393.73	ND	veh/km
Distancia total recorrida Car	2731.74	ND	km
Flujo Car	5963	ND	veh/h
Longitud media de cola Car	1.24	ND	vehs
Número de Paradas Car	2.43	ND	
Tiempo de Demora Car	1535.58	921.34	seg/km
Tiempo de Parada Car	1493.32	920.94	seg/km
Tiempo de Viaje Car	1653.25	921.54	seg/km
Tiempo total de viaje Car	1250.95	ND	h
Vehículos Dentro Car	125	ND	vehs
Vehículos Errantes Dentro Car	0	ND	vehs
Vehículos Errantes Fuera Car	0	ND	vehs
ehículos Esperando para Entrar C	3897	ND	vehs
Vehículos Fuera Car	5963	ND	vehs
Velocidad Car	22.44	1.86	km/h
Velocidad Harmónica Car	22.25	2.07	km/h

Tabla 4.12. Resultados de la simulación con el 100% de vehículos instrumentados.

Fuente: Elaboración propia.

Se instrumentaron estos porcentajes de vehículos para poder apreciar el comportamiento del flujo general del sistema.

Para poder apreciar mejor el fenómeno los datos se graficaron en la Figura 4.1. Se puede apreciar el tiempo total de viaje de todos los autos del sistema y su comparación con base en el número de autos instrumentados.

Se visualiza en dicha figura que para el 10%, 20% y 30% de autos instrumentados, se obtuvieron mayores tiempos de viaje en todo el sistema. Lo anterior es provocado por el incremento de la interacción en las intersecciones de vehículos no instrumentados y de los que sí lo estaban, por lo que se presentaron demoras en dichos puntos. Posteriormente se presentaron mejoras en el tiempo de viaje general de todo el sistema con un mayor porcentaje de vehículos

instrumentados. La línea roja que se presenta en la tabla es el tiempo de viaje tomado como referencia del modelo FTVFL, el cual es de 2,272.75 seg/km.

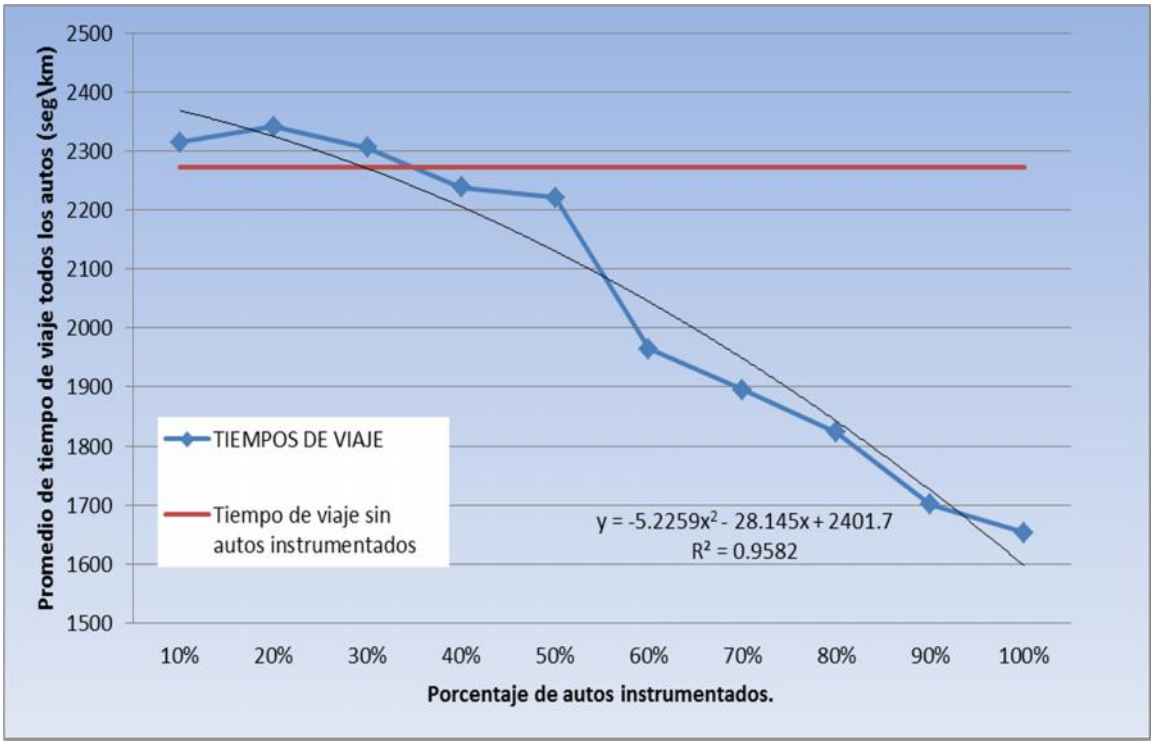

Figura 4.1. Resultados de la simulación con el 100% de vehículos instrumentados.
Fuente: Elaboración propia.

En comparación con el modelo EDU, la reducción es mayor, ya que el tiempo de viaje general del sistema presentado por el modelo EDU es de 1,978.15 seg/km y del SEV para el 100% de vehículos instrumentados es de 1,653.25 seg/km, lo que refleja una disminución en tiempo real del 16% en comparación con el modelo EDU. Lo anterior es debido a que el flujo de automóviles es mayor con EDU que con SEV.

4.2 Resultados de la simulación en el Modelo de Vialidades de la Ciudad de Querétaro.

A continuación se expone la asignación de vehiculos y la aplicación del algoritmo en una sección de vialidades en la ciudad de Santiago de Querétaro, considerando una temporalidad de 12:00pm a 1:00pm y con un precalentamiento de 5 minutos.

4.2.1 De la simulación con el Modelo de Asignación Fijo con Tiempos de Viaje a Flujo Libre.

La simulación realizada con el modelo de asignación FTVFL, nos presenta un flujo de 5652 veh/h con un tiempo total de viaje de 275.18h y un tiempo promedio de viaje de 75.99 seg/km de acuerdo a la Tabla 4.13.

Serie Temporal	Valor	Desviación Estándar	Unidades
Cola Virtual Máxima Auto-1999-A.	1	ND	vehs
Cola Virtual Media Auto-1999-A.	0.01	ND	vehs
Densidad Auto-1999-A.	1.89	ND	veh/km
Distancia total recorrida Auto-1999-A.	13950.32	ND	km
Flujo Auto-1999-A.	5652	ND	veh/h
Longitud media de cola Auto-1999-A.	12.72	ND	vehs
Número de Paradas Auto-1999-A.	0.33	ND	
Tiempo de Demora Auto-1999-A.	9.28	23.06	seg/km
Tiempo de Parada Auto-1999-A.	5.8	21.44	seg/km
Tiempo de Viaje Auto-1999-A.	75.99	25.66	seg/km
Tiempo total de viaje Auto-1999-A.	275.18	ND	h
Vehículos Dentro Auto-1999-A.	301	ND	vehs
Vehículos Errantes Dentro Auto-1999-A.	0	ND	vehs
Vehículos Errantes Fuera Auto-1999-A.	6	ND	vehs
Vehículos Esperando para Entrar Auto-1999-A.	0	ND	vehs
Vehículos Fuera Auto-1999-A.	5652	ND	vehs
Velocidad Auto-1999-A.	50.72	12.57	km/h
Velocidad Harmónica Auto-1999-A.	47.4	12.54	km/h

Tabla 4.13. Resultados de la simulación para el modelo de asignación FTVFL.

Fuente: Elaboración propia.

En la Figura 4.2 es posible observar en distintos tonos el flujo en los arcos más utilizados por los automóviles para circular. Siento el verde el color para los arcos con menor flujo y el rojo para los arcos con mayor flujo.

Figura 4.2. Flujo de automóviles en la simulación con el modelo de asignación FTVFL.

Fuente: Elaboración propia.

4.2.2 De la simulación con el Modelo de Asignación Equilibrio Dinámico de Usuario.

La simulación realizada con el modelo de asignación EDU, nos presenta un flujo de 5,791 veh/h con un tiempo total de viaje de 272.94h y un tiempo promedio de viaje de 73.7 seg/km, de acuerdo a la Tabla 4.14. Este resultado mejora en un porcentaje no muy grande (3%) el flujo y el tiempo de viaje de los usuarios dentro del sistema, con respecto al modelo de asignación FTVFL.

Serie Temporal	Valor	Desviación Estándar	Unidades
Cola Virtual Máxima Auto-1999-A.	2	ND	vehs
Cola Virtual Media Auto-1999-A.	0.01	ND	vehs
Densidad Auto-1999-A.	1.87	ND	veh/km
Distancia total recorrida Auto-1999-A.	14136.49	ND	km
Flujo Auto-1999-A.	5791	ND	veh/h
Longitud media de cola Auto-1999-A.	6	ND	vehs
Número de Paradas Auto-1999-A.	0.26	ND	
Tiempo de Demora Auto-1999-A.	6.7	16.08	seg/km
Tiempo de Parada Auto-1999-A.	3.4	14.27	seg/km
Tiempo de Viaje Auto-1999-A.	73.7	18.98	seg/km
Tiempo total de viaje Auto-1999-A.	272.94	ND	h
Vehículos Dentro Auto-1999-A.	301	ND	vehs
Vehículos Errantes Dentro Auto-1999-A.	1	ND	vehs
Vehículos Errantes Fuera Auto-1999-A.	3	ND	vehs
Vehículos Esperando para Entrar Auto-1999-A.	0	ND	vehs
Vehículos Fuera Auto-1999-A.	5791	ND	vehs
Velocidad Auto-1999-A.	51.17	11.11	km/h
Velocidad Harmónica Auto-1999-A.	48.87	10.61	km/h

Tabla 4.14. Resultados de la simulación para el modelo de asignación EDU.

Fuente: Elaboración propia.

En la Figura 4.3 es posible observar en distintos tonos el flujo en los arcos más utilizados por los automóviles para circular. Se muestra en color verde para los arcos con menor flujo y el rojo para los arcos con mayor flujo.

En la Figura 4.3 es posible observar el cambio en la saturación en los arcos con respecto de los resultados de la primera simulación, cuestión que ocasionó la reducción en los tiempos de viaje y el incremento en el flujo. Se observa además que fueron necesarias tomar distintas rutas por los automóviles que no llegaron a saturarse, pero se observa que se incrementó el flujo por dichas calles, estos arcos se muestran en amarillo.

Figura 4.3. Flujo de automóviles en la simulación con el modelo de asignación EDU.

Fuente: Elaboración propia.

4.2.3 Resultados de la simulación con el Sistema Enrutador de Vehículos.

La simulación realizada con el sistema enrutador de vehículos nos presenta un flujo de 5643 veh/h con un tiempo total de viaje de 293.2h y un tiempo promedio de viaje de 76.53 seg/km de acuerdo a la Tabla 4.15. Este resultado es el más desfavorable de las simulaciones, sin embargo se explica debido a que la actualización de rutas y caminos más cortos se realizó cada minuto, lo que disminuyó razonablemente las capacidades de ruteo del algoritmo.

Serie Temporal	Valor	Desviación Estándar	Unidades
Cola Virtual Máxima Auto-1999-A.	1	ND	vehs
Cola Virtual Media Auto-1999-A.	0.01	ND	vehs
Densidad Auto-1999-A.	2.01	ND	veh/km
Distancia total recorrida Auto-1999-A.	14542.66	ND	km
Flujo Auto-1999-A.	5643	ND	veh/h
Longitud media de cola Auto-1999-A.	9.7	ND	vehs
Número de Paradas Auto-1999-A.	0.34	ND	
Tiempo de Demora Auto-1999-A.	9	18.11	seg/km
Tiempo de Parada Auto-1999-A.	4.63	16.19	seg/km
Tiempo de Viaje Auto-1999-A.	76.53	20.78	seg/km
Tiempo total de viaje Auto-1999-A.	293.2	ND	h
Vehículos Dentro Auto-1999-A.	310	ND	vehs
Vehículos Errantes Dentro Auto-1999-A.	0	ND	vehs
Vehículos Errantes Fuera Auto-1999-A.	5	ND	vehs
Vehículos Esperando para Entrar Auto-1999-A.	0	ND	vehs
Vehículos Fuera Auto-1999-A.	5643	ND	vehs
Velocidad Auto-1999-A.	49.52	11.35	km/h
Velocidad Harmónica Auto-1999-A.	47.07	10.75	km/h

Tabla 4.15. Resultados de la simulación con el Sistema Enrutador de Vehículos.

Fuente: Elaboración propia.

En la Figura 4.4 se muestra cambio en el flujo de los arcos al utilizar el SEV. Los tiempos de viaje son similares a los de la simulación empleando el modelo de asignación FTVFL, debido a la restricción de calcular solamente cada minuto las rutas más cortas en tiempo y guiar a los automóviles por dichas rutas.

Figura 4.4. Flujo de automóviles en la simulación con el sistema enrutador de vehículos.

Fuente: Elaboración propia.

A continuación se muestran los resultados de los 3 métodos y su comparativa. En la Figura 4.5 se muestra la comparativa de los flujos, en veh/h.

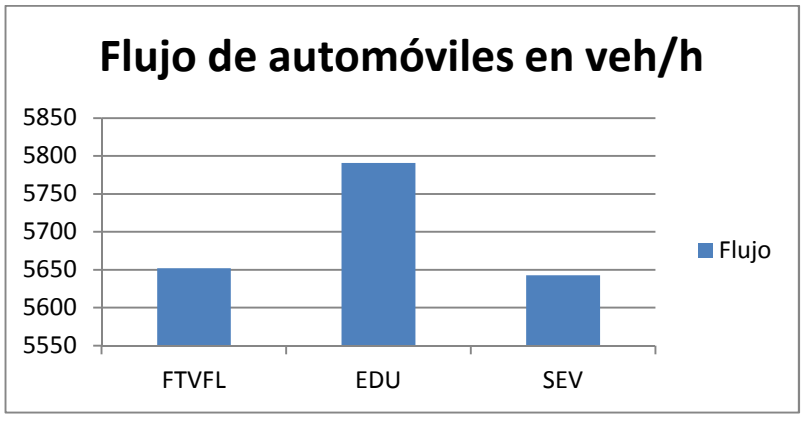

Figura 4.5. Flujo de automóviles de las 3 simulaciones realizadas.

Fuente: Elaboración propia.

En la Figura 4.6 se puede apreciar la comparativa de los tiempos de viaje acumulados de los 3 escenarios.

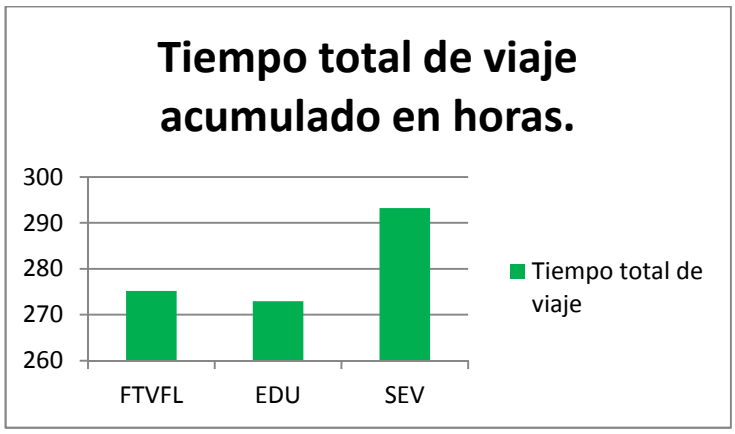

Figura 4.6. Tiempo acumulado de viaje de automóviles de las 3 simulaciones.

Fuente: Elaboración propia.

Finalmente, en la Figura 4.7 se muestra la comparativa entre los tiempos de viaje promedio de las 3 simulaciones realizadas.

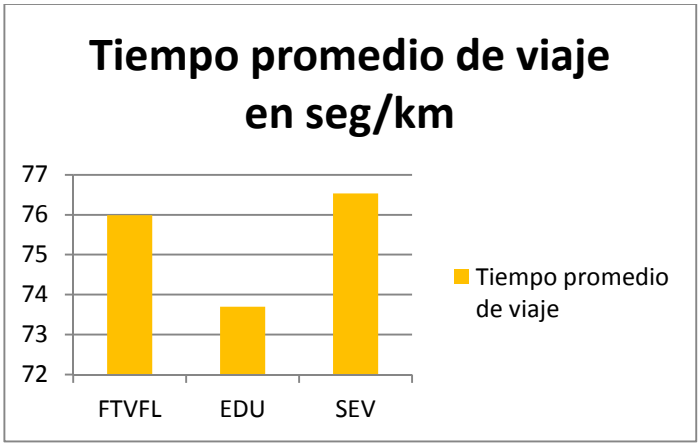

Figura 4.7. Tiempo promedio de viaje de automóviles de las 3 simulaciones.

Fuente: Elaboración propia.

Es posible apreciar el cambio en los tiempos de viaje del SEV, debido a la restricción de calcular cada minuto los caminos más cortos. Se puede entonces decir que es necesario contar con la información de las rutas más cortas en intervalos menores de un minuto, para poder alcanzar los tiempos de viaje similares al modelo de asignación EDU.

5. CONCLUSIONES

El algoritmo enrutador de vehículos es una opción viable para mejorar los tiempos de viaje en una ciudad dotada de un sistema enrutador dinámico, como el conceptualizado en este trabajo, siempre y cuando todos los automóviles siguieran las instrucciones del algoritmo enrutador desde el inicio del viaje, lo anterior valida la hipótesis de la presente investigación

Al mejorar los tiempos de viaje se incrementó el flujo de automóviles lo que significa que optimizaría los desplazamientos dentro de las vialidades evitando congestionamiento y la acumulación de contaminantes emitidos por los automóviles.

Una ciudad dotada de este sistema permitiría al usuario conocer el tiempo de viaje estimado a su destino, antes de salir de casa y decidir el medio de transporte más rápido por el cual pueda llegar a su destino.

Al conocer los desplazamientos de los automóviles dentro del sistema, se tendría la posibilidad de tener la información para construir una matriz origen-destino de la ciudad, la cual serviría para implementar otro tipo de estrategias de mitigación como la de uso de suelo, además de tener la información para una futura integración de otro medio de transporte a la ciudad.

Una de las desventajas mencionadas por la literatura en este tipo de sistemas es el aumento de flujo por calles en las que anteriormente no se tenía dicho flujo, por lo que se tendría que dotar a dichas calles de elementos de señalamiento vial preventivo ante el incremento vehicular.

El mejoramiento de los tiempos de viaje con un sistema de estas características, lo hacen una opción atractiva y barata para mitigar el congestionamiento vial en una ciudad. Dicho mejoramiento también depende

bastante de las capacidades de procesamiento del *hardware* con el que se pueda llegar a implementar el sistema enrutador de vehículos.

6. LITERATURA CITADA

Ahuja, R., Magnati, T. y Orlin, J. (1988). *Network Flows*. Sloan School of Management. MIT.

Arias, J. (2010) Algoritmo de Dijkstra escrito en C++.

Barceló, J. (2010). *Fundamentals of Traffic Simulation*. Ed. Springer. 440p.

Barceló, J. y Kuwahara, M. (2010). *Traffic Data Collection and its Standardization*. Ed. Springer. 194p.

Biggs, N., Lloyd, E. y Wilson, R. (1986). *Graph Theory 1736-1936*. Clarendon Press. 435pp.

Black, W. (2010). *Sustainable Transportation*. The Guilford Press. ISBN 978-1-60623-485-3.

Casas, J., Ferrer, J., Garcia, D.,Perarnau, J. y Torday, A. (2010) *Fundamentals of Traffic Simulation: Traffic Simulation with Aimsun*. International Series in Operations Research & Management Science. Springer.

Chowdhury M. y Sadek, A. (2003). *Fundamentals of Intelligent Transportation Systems Planning*. Artech House.

De Palma, A. y Lindsey, R. (1998). *Private toll roads: Competition under various ownership regimes*. The annals of Regional Science.

Dijkstra, E. (1959) A Note on Two Problems in Connexion with Graphs. *Numerishe Mathematik,* 1, 269 – 271.

Dong, W. (2011). An overview of in-vehicle route guidance system. Australasian Transport Research Forum.

Downs, A. (2004). *Still Stuck in Traffic, Coping with peak-hour traffic congestion*. The Brookins Institution. ISBN 0-8157-1929-9.

Dunphy, R. (1991). *12 Tools for Improving Mobility and Managing Congestion*. Urban Land Institute.

ECMT, OECD. (2007). *Managing Urban Traffic Congestion*. European Conference of Ministers of Transport. ISBN 978-92-821-0128-5.

García-Ortiz, A., Amin, S. y Wootton, J. (2000). *Intelligent transportation systems—Enabling technologies*. Elsevier.

Gipps, P. (1986) A Model for the Structure of Lane Changing Decisions. *Transportation Research Board,* 20. pp.107 -120.

Hernández, G. (2003) *Grafos: Teoría y Algoritmos*. Servicio de Publicaciones, Facultad de Informática, UPM.

IEEE. (2012). *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE Computer Society.

Immers, H. y Logghe, S., (2003). *Traffic Flow Theory*. Katholieke Universiteit Leuven.

International Bussines Machines (IBM). (2011). Frustration Rising: IBM 2011 Commuter Pain Survey.

Kerner, B. (2009). *The Physics of Traffic. Empirical Freeway Patter Features, Engineering Applications, and Theory*. Springer. 688pp.

Kraus, E. y Lee, W. (2000) Evaluation of Traffic Mitigation Strategies Using a Micro-Simulation Software, NETSIM and a Pilot Rhode Island Intelligent Road. *80th Annual Meeting of the Transportation Research Board*. Artículo Nº 01-2430.

Kuwahara,M., Horiguchi, R. y Hanabusa, H. (2010) *Fundamentals of Traffic Simulation: Traffic Simulation with AVENUE*. International Series in Operations Research & Management Science. Springer.

Laumbach, R. y Kipen,H. (2011). Respiratory health effects of air pollution: Update on biomass smoke and traffic pollution. *American Academy of Allergy, Asthma & Immunology*. No 1, Vol 129, pp. 1-9.

Li, B., Quader, I. y Dempster, A. (2008) On Outdoor positioning with Wi-Fi. *Journal of Global Positioning Systems*. Vol. 7, No. 1, pp. 18-26.

Lighthill, J. y Whitman,B. (1955). A Theory of traffic flow on long crowded roads. Proceedings of the Royal Society A.

Lomax, T. (1997) *Quantifying Congestion*. National Academy Press. 38pp.

Martin, A., Marini, H. y Tosunoglu, S. (1999). Intelligent Vehicle/Highway System: A Survey— Part 1. Conferencia en Florida sobre los recientes avances en robótica. Gainesville, Florida.

Martin, P., Lahon, D., Cook, K. y Stevanovic, A. (2005) *Traveler Information Systems, Evaluation of UDOT's ATIS Technologies*. Departament of Civil and Enviromental Engineering, University of Utah.

Mathew, T. y Rao, K. (2007) Introduction to Transportation Engineering. NPTEL Online.

Noland, R. y Quddus, M. (2005). Congestion and safety: a spatial analysis of London. *Transport. Res. A,* 39, pp. 737–754.

Ramírez, E. (2012). *Comparación de la interacción vehicular inducida por el empleo de carriles preferenciales para el transporte público utilizando microsimulación de tráfico*. Universidad Autónoma de Querétaro.

Rehborn, H., Klenov,S. y Palmer,J. (2010). Common Traffic Congestion Features studied in USA, UK, and Germany employing Kerner's Three-Phase Traffic Theory. *Proc. IEEE Intell. Veh. Symp*., pp.19 -24, 2011.

Riojas, H., Alamo,U., Texcalac, J. y Romieu, I. (2012*). Evaluación de impacto en salud por exposiciòn a ozono y material particulado (PM10) en la Zona Metropolitana del Valle de México*. Ed. CISP, ISBN 978-607-511-028-8.

Shladover, S. 2010. Review of the state of development of advanced vehicle control systems (AVCS). *International Journal of Vehicle Mechanics and Mobility*, vol. 24, pp.551 -595

Skyes,P. (2010) *Fundamentals of Traffic Simulation: Traffic Simulation with Paramics*. International Series in Operations Research & Management Science. Springer.

Tang, Y. y Xi, B. (2010). Dynamic forecasting of traffic volume based on quantificational dynamics: A nearness perspective. *Scientific Research and Essays*, 5(4), pp. 389–394.

Thulasiraman, K. y Swamy, S. (1992) *Graphs: Theory and Algorithms*. John Wiley & Sons, Incorporated.

TSS-Transport Simulation Systems. (2008). *Aimsun Scripting Manual*. 45p.

TSS-Transport Simulation Systems. (2010). *Aimsun MicroMeso User's Manual* v6.1 391p.

TSS-Transport Simulation Systems. (2010). *Aimsun Microsimulator API Manual*. 116p.

TSS-Transport Simulation Systems. (2011). *Aimsun User's Manual v7.0*. 348p.

West, D. (2002) *Introduction to Graph Theory*. Pearson Education. ISBN 81-7808-830-4.

Williams, B. (2008). *Intelligent Transport Systems Standards*. Artech House. 827 p.

Yarger, W. (2010). *Fully actuated vs. semi-actuated traffic signal systems*. Yarger Engineering, Inc.

7. ANEXOS

7.1 Código Fuente Algoritmo Enrutador de Vehículos para el Modelo de Pruebas.

```
#include "AKIProxie.h"
#include "CIProxie.h"
#include "ANGConProxie.h"
#include "AAPI.h"
#include <stdio.h>
#include <vector>
#include <queue>
using namespace std;
#define MAX 10005 //maximo numero de vértices
//definimos el nodo como un par( first , second ) donde first es el
vertice adyacente y second el peso de la arista
#define Node pair< int , int >
#define INF 1<<30
      int orig[100],dest[100],r,gg;
      int pes[100];
struct cmp {
    bool operator() ( const Node &a , const Node &b ) {
        return a.second > b.second;
    }
};
char astring[128];
int i,secc2nodo[50];
      vector<int> vecEjemplo;
int bandera[10000];
double promediotv[10000];
double tiempoviaje[10000][10000],vehiculo[10000][2],suma[10000] ;
int tiempo,repetid[10000],via,idatras,see,vehi;
int nodosec[10000][10000];
int E ,re, origen, destino, inicial, peso;
int vehnodcmc[10000][10000];
//Base de datos de nodos y secciones.

int bandijkstra=0;

      vector< Node > ady[ MAX ]; //lista de adyacencia
int distancia[ MAX ];      //distancia[ u ] distancia de vértice inicial
a vértice con ID = u
bool visitado[ MAX ];      //para vértices visitados
//priority queue propia del c++, usamos el comparador definido para que
el de menor valor este en el tope
 priority_queue< Node , vector<Node> , cmp > Q;
int V;                     //numero de vertices
int previo[ MAX ];

void init(){
    for( int i = 0 ; i <= V ; ++i ){
        distancia[ i ] = INF;  //inicializamos todas las distancias con
valor infinito
        visitado[ i ] = false; //inicializamos todos los vértices como no
visitados
```

```
        previo[ i ] = -1;      //inicializamos el previo del vertice i
con -1
    }
}
void relajacion( int actual , int adyacente , int peso ){
    //Si la distancia del origen al vertice actual + peso de su arista es
menor a la distancia del origen al vertice adyacente
    if( distancia[ actual ] + peso < distancia[ adyacente ] ){
        distancia[ adyacente ] = distancia[ actual ] + peso;  //relajamos
el vertice actualizando la distancia
        previo[ adyacente ] = actual;                          //a su vez
actualizamos el vertice previo
        Q.push( Node( adyacente , distancia[ adyacente ] ) ); //agregamos
adyacente a la cola de prioridad
    }
}

//Impresion del camino mas corto desde el vertice inicial y final
ingresados
void print( int destino ){

      int dest;
    if( previo[ destino ] != -1 )    //si aun poseo un vertice previo
        print( previo[ destino ] );  //recursivamente sigo explorando

      vecEjemplo.push_back(destino);            //terminada la recursion
imprimo los vertices recorridos
}

void dijkstra( int inicial ){
    init(); //inicializamos nuestros arreglos
    Q.push( Node( inicial , 0 ) ); //Insertamos el vértice inicial en la
Cola de Prioridad
    distancia[ inicial ] = 0;      //Este paso es importante,
inicializamos la distancia del inicial como 0
    int actual , adyacente , peso;
    while( !Q.empty() ){                  //Mientras cola no este vacia
        actual = Q.top().first;           //Obtengo de la cola el nodo
con menor peso, en un comienzo será el inicial
        Q.pop();                          //Sacamos el elemento de la
cola
        if( visitado[ actual ] ) continue; //Si el vértice actual ya fue
visitado entonces sigo sacando elementos de la cola
        visitado[ actual ] = true;        //Marco como visitado el
vértice actual

        for( int i = 0 ; i < ady[ actual ].size() ; ++i ){ //reviso sus
adyacentes del vertice actual
            adyacente = ady[ actual ][ i ].first;   //id del vertice
adyacente
            peso = ady[ actual ][ i ].second;        //peso de la arista
que une actual con adyacente ( actual , adyacente )
            if( !visitado[ adyacente ] ){        //si el vertice
adyacente no fue visitado
                relajacion( actual , adyacente , peso ); //realizamos el
paso de relajacion
            }
```

```
        }
    }




}


int AAPILoad()
{
 return 0;
}

int AAPIInit()
{

      for (int r = 0; r < 10000; r++){ //Asignamos el valor de uno para
la variable repetid
  repetid[r] = 1;
}
      for (int r = 0; r < 10000; r++){//Asignamos el valor de uno para la
variable repetid
  bandera[r] = 0;
}
 return 0;
}

int AAPIManage(double time, double timeSta, double timeTrans, double
acycle)
{




 return 0;
}

int AAPIPostManage(double time, double timeSta, double timeTrans, double
acycle)

{
      // En este apartado obtenemos todas las secciones del sistema,
      //y posteriormente leemos de cada sección el número de autos que se
encuentran en el sistema
      tiempo=time;
      InfVeh infVeh;
 int nba = AKIInfNetNbSectionsANG();
 for( i=0; i<nba;i++){
  int id = AKIInfNetGetSectionANGId(i);
  int nb = AKIVehStateGetNbVehiclesSection(id,true);
  for (int j=0; j<nb;j++){

   infVeh = AKIVehStateGetVehicleInfSection(id,j);
   InfVeh infveh = AKIVehTrackedGetInf(infVeh.idVeh);
```

```
   StaticInfVeh vehi= AKIVehTrackedGetStaticInf(infVeh.idVeh);
        if (infveh.report >= 0)//Imprimimos la información estática de
los automóviles instrumentados únicamente.
        {
            sprintf(astring,"Vehicle %d , Section %d , Lane %d,
CentroideDest %d, CurrentSpeed %f\n",
            infVeh.idVeh, infVeh.idSection, infVeh.numberLane,
vehi.centroidDest ,infVeh.CurrentSpeed);
   AKIPrintString(astring);
        }
//Detectamos la entrada y salida del automóvil en una sección
 if ((infveh.report >= 0)&&(bandera[infVeh.idVeh] == 0))
        {
    vehiculo[infVeh.idVeh][1]=infVeh.idSection;//asignamos a un arreglo
la sección de entrada del automóvil
    vehiculo[infVeh.idVeh][2]=time;//asignamos el tiempo de entrada al
sistema del automóvil a un arreglo
             bandera[infVeh.idVeh]=1;//Nos aseguramos que sólo se repita
una sola vez al entrar al sistema

        }
//Revisamos la condición si las secciones de inicio y final son
diferentes y si el automóvil está instrumentado
if((vehiculo[infVeh.idVeh][1] != infVeh.idSection) && (infveh.report >=
0) )
{
idatras=vehiculo[infVeh.idVeh][1];
tiempo=repetid[idatras];
tiempoviaje[idatras][tiempo]= time - vehiculo[infVeh.idVeh][2];
suma[idatras]=suma[idatras]+tiempoviaje[idatras][tiempo];
promediotv[idatras] = suma[idatras]/repetid[idatras];
repetid[idatras]=repetid[idatras]+1;
bandera[infVeh.idVeh]=0;


}

//Inicializamos los datos para el Algoritmo Dikjstra
      ////////////////////////////////////////////////////////////////
///
if ( ((time > 60)) && (infveh.report >= 0)){
           orig[1]=6;
    dest[1]=2;
    pes[1]=promediotv[443];

    orig[2]=1;
    dest[2]=4;
    pes[2]=promediotv[442];

    orig[3]=2;
    dest[3]=3;
    pes[3]=promediotv[448];

    orig[4]=2;
    dest[4]=4;
    pes[4]=promediotv[447];
```

```
    orig[5]=3;
    dest[5]=5;
    pes[5]=promediotv[446];


    orig[6]=4;
    dest[6]=2;
    pes[6]=promediotv[445];

    orig[7]=4;
    dest[7]=3;
    pes[7]=promediotv[444];

    orig[8]=4;
    dest[8]=5;
    pes[8]=promediotv[500];

      orig[9]=2;
    dest[9]=5;
    pes[9]=promediotv[449];


      secc2nodo[443]=6;
      secc2nodo[449]=2;
      secc2nodo[446]=3;
      secc2nodo[448]=2;
      secc2nodo[500]=4;
      secc2nodo[445]=4;
      secc2nodo[447]=2;
      secc2nodo[442]=1;
      secc2nodo[444]=4;



    V=6;
      E=9;

     for (r=1;r<=E;r++)
      {
        origen=orig[r];
        destino=dest[r];
        peso=pes[r];

        ady[ origen ].push_back( Node( destino , peso ) ); //consideremos
grafo dirigido
        //ady[ destino ].push_back( Node( origen , peso ) ); //grafo no
dirigido
    }
       re=infVeh.idSection;
    inicial=secc2nodo[re];

    dijkstra( inicial );

      //destino=vehi.centroidDest;//En dado caso que hubiera mas
destinos, utilizamos esta linea de codigo
    destino=5;//El nodo final para todos los casos
      //sprintf(astring,"DESTINO: %d ", destino);
```

```
      //AKIPrintString(astring);
    print( destino );
    for (int gg = 0; gg < vecEjemplo.size(); gg ++) {
vehnodcmc[infVeh.idVeh][gg]=vecEjemplo[gg];
//vecEjemplo me da los nodos que debo de cruzar para llegar a mi destino
mas corto
     sprintf(astring,"%d \t", vecEjemplo[gg]);
      AKIPrintString(astring);
     }

}
////////////////////////////////////////////////////////////////////////
///
//RUTINA PARA CONVERTIR DE NODOS A ARCOS

int ig=0;


if (( infveh.report >= 0)&&(bandijkstra==1)){
nodosec[6][2]=443;
nodosec[1][4]=442;
nodosec[4][2]=445;
nodosec[2][4]=447;
nodosec[4][3]=444;
nodosec[4][5]=500;
nodosec[2][3]=448;
nodosec[3][5]=446;
nodosec[2][5]=449;
int fff;
fff=vecEjemplo.size();
      for(int q=0; q < (fff-1); q++){
      int r1=vehnodcmc[infVeh.idVeh][q];
      int r2=vehnodcmc[infVeh.idVeh][q+1];
      int secc=nodosec[r1][r2];


      if (infveh.idSection == secc)
      {
            ig=1;
            continue;

      }


      if (ig == 1){
      AKIVehTrackedModifyNextSection (infVeh.idVeh, secc);
      ig=0;

      }
      }
}

//TERMINA RUTINA PARA CONVERTIR DE NODOS A ARCOS
////////////////////////////////////////////////////////////////////////
///
```

```
bandijkstra=1;

////////////////////////////////////////////////////////////////////////
/////
vecEjemplo.clear();
ady[origen].erase(ady[origen].begin(), ady[origen].end());
  }//END FOR 2

 }// END FOR 1

//IMPRESIÓN DE LOS TIEMPOS PROMEDIO DE VIAJE DE LAS SECCIONES
sprintf(astring,"PROMEDIO 443: %f ",promediotv[443]);
   AKIPrintString(astring);
    sprintf(astring,"PROMEDIO 444: %f ",promediotv[444]);
   AKIPrintString(astring);
   sprintf(astring,"PROMEDIO 442: %f ",promediotv[442]);
   AKIPrintString(astring);
   sprintf(astring,"PROMEDIO 449: %f ",promediotv[449]);
   AKIPrintString(astring);
    sprintf(astring,"PROMEDIO 447: %f ",promediotv[447]);
   AKIPrintString(astring);
    sprintf(astring,"PROMEDIO 445: %f ",promediotv[445]);
   AKIPrintString(astring);
    sprintf(astring,"PROMEDIO 448: %f ",promediotv[448]);
   AKIPrintString(astring);
     sprintf(astring,"PROMEDIO 446: %f ",promediotv[446]);
   AKIPrintString(astring);
    sprintf(astring,"PROMEDIO 500: %f ",promediotv[500]);
   AKIPrintString(astring);
//Asignamos tiempos de viaje
   if (promediotv[443]==0)
   {
         promediotv[443]=20;
   }
  if (promediotv[444]==0)
   {
         promediotv[444]=20;
   }
    if (promediotv[442]==0)
   {
         promediotv[442]=20;
   }
        if (promediotv[449]==0)
   {
         promediotv[449]=20;
   }
          if (promediotv[447]==0)
   {
         promediotv[447]=39.018904;
   }
              if (promediotv[445]==0)
   {
         promediotv[445]=23.541667;
   }
  if (promediotv[448]==0)
```

```
   {
         promediotv[448]=16.331126;
   }
  if (promediotv[500]==0)
   {
         promediotv[500]=16.331126;
   }
    if (promediotv[446]==0)
   {
         promediotv[446]=16.331126;
   }

   //TERMINA ALGORITMO DIJKSTRA

      for (i=1;i<=10000;i+=10)
      {

      AKIVehSetAsTracked (i);

      // INSTRUMENTAMOS EL 10% DE LOS VEHICULOS QUE CIRCULARAN EN EL
SISTEMA
      }


      tiempo=time;

 return 0;
}

int AAPIFinish()
{
 return 0;
}

int AAPIUnLoad()
{
 return 0;
}
```

7.2 Código Fuente Algoritmo Enrutador de Vehículos para grafo de la Ciudad de Querétaro.

```
#include "AKIProxie.h"
#include "CIProxie.h"
#include "ANGConProxie.h"
#include "AAPI.h"
#include <stdio.h>
#include <vector>
#include <queue>
#include <math.h>
using namespace std;
#define MAX 10005 //maximo numero de vértices
#define Node pair< int , int > //definimos el nodo como un par( first ,
second ) donde first es el vertice adyacente y second el peso de la
arista
#define INF 1<<30
int orig[1000],dest[1000],r,gg;
int pes[1000];
// Procedures could be modified by the user
struct cmp {
      bool operator() ( const Node &a , const Node &b ) {
            return a.second > b.second;
      }
};
char astring[128];
int i,secc2nodo[500];
vector<int> vecEjemplo;
int bandera[30000];//Depende del número total de vehículos rastreados a
simular
double promediotv[999999];
static float tiempoviaje[999999][9];
double vehiculo[20000][2];
double suma[999999] ;
int tiempo,repetid[999999],via,idatras,see,vehi;
int nodosec[10000][10000];
int E ,re, origen, destino, inicial, peso;
int vehnodcmc[10000][10000];
int centroidest[999999];
//Base de datos de nodos y secciones.

int bandijkstra=0;

vector< Node > ady[ MAX ]; //lista de adyacencia
int distancia[ MAX ];      //distancia[ u ] distancia de vértice inicial
a vértice con ID = u
bool visitado[ MAX ];      //para vértices visitados
priority_queue< Node , vector<Node> , cmp > Q;//priority queue propia del
c++, usamos el comparador definido para que el de menor valor este en el
tope
int V;                     //numero de vertices
int previo[ MAX ];

void init(){
      for( int i = 0 ; i <= V ; ++i ){
```

```
            distancia[ i ] = INF;  //inicializamos todas las distancias
con valor infinito
            visitado[ i ] = false; //inicializamos todos los vértices
como no visitados
            previo[ i ] = -1;      //inicializamos el previo del vertice
i con -1
      }
}
void relajacion( int actual , int adyacente , int peso ){
      //Si la distancia del origen al vertice actual + peso de su arista
es menor a la distancia del origen al vertice adyacente
      if( distancia[ actual ] + peso < distancia[ adyacente ] ){
            distancia[ adyacente ] = distancia[ actual ] + peso;
//relajamos el vertice actualizando la distancia
            previo[ adyacente ] = actual;                          //a su
vez actualizamos el vertice previo
            Q.push( Node( adyacente , distancia[ adyacente ] ) );
//agregamos adyacente a la cola de prioridad
      }
}

//Impresion del camino mas corto desde el vertice inicial y final
ingresados
void print( int destino ){

      int dest;
      if( previo[ destino ] != -1 )    //si aun poseo un vertice previo
            print( previo[ destino ] );  //recursivamente sigo explorando

      vecEjemplo.push_back(destino);            //terminada la recursion
imprimo los vertices recorridos
}

void dijkstra( int inicial ){
      init(); //inicializamos nuestros arreglos
      Q.push( Node( inicial , 0 ) ); //Insertamos el vértice inicial en
la Cola de Prioridad
      distancia[ inicial ] = 0;      //Este paso es importante,
inicializamos la distancia del inicial como 0
      int actual , adyacente , peso;
      while( !Q.empty() ){                  //Mientras cola no este
vacia
            actual = Q.top().first;           //Obtengo de la cola el
nodo con menor peso, en un comienzo será el inicial
            Q.pop();                          //Sacamos el elemento de
la cola
            if( visitado[ actual ] ) continue; //Si el vértice actual ya
fue visitado entonces sigo sacando elementos de la cola
            visitado[ actual ] = true;        //Marco como visitado el
vértice actual

            for( int i = 0 ; i < ady[ actual ].size() ; ++i ){ //reviso
sus adyacentes del vertice actual
                  adyacente = ady[ actual ][ i ].first;   //id del
vertice adyacente
                  peso = ady[ actual ][ i ].second;        //peso de la
arista que une actual con adyacente ( actual , adyacente )
```

```
                  if( !visitado[ adyacente ] ){        //si el vertice
adyacente no fue visitado
                        relajacion( actual , adyacente , peso );
//realizamos el paso de relajacion
                  }
            }
      }









}


int AAPILoad()
{
      return 0;
}

int AAPIInit()
{

      for (int r = 0; r < 100000; r++){
            repetid[r] = 1;
      }
      for (int r = 0; r < 100000; r++){
            bandera[r] = 0;
      }
      return 0;
}

int AAPIManage(double time, double timeSta, double timeTrans, double
acycle)
{




      return 0;
}

int AAPIPostManage(double time, double timeSta, double timeTrans, double
acycle)

{
      tiempo=time;
      InfVeh infVeh;
      int nba = AKIInfNetNbSectionsANG();
      for( i=0; i<nba;i++){
            int id = AKIInfNetGetSectionANGId(i);
            int nb = AKIVehStateGetNbVehiclesSection(id,true);
            for (int j=0; j<nb;j++){

                  infVeh = AKIVehStateGetVehicleInfSection(id,j);
                  InfVeh infveh = AKIVehTrackedGetInf(infVeh.idVeh);
```

```
                  StaticInfVeh vehi=
AKIVehTrackedGetStaticInf(infVeh.idVeh);


                  if ((infveh.report >= 0)&&(bandera[infVeh.idVeh] == 0))
                  {

      vehiculo[infVeh.idVeh][1]=infVeh.idSection;//asignamos a un arreglo
la sección de entrada del automóvil
                        vehiculo[infVeh.idVeh][2]=time;//asignamos el
tiempo de entrada al sistema del automóvil a un arreglo
                        bandera[infVeh.idVeh]=1;//Nos aseguramos que sólo
se repita una sola vez al entrar al sistema

                  }

                  if((vehiculo[infVeh.idVeh][1] != infVeh.idSection) &&
(infveh.report >= 0) )
                        //Revisamos la condición si las secciones de
inicio y final son diferentes y si el automóvil está instrumentado
                  {
                        idatras=vehiculo[infVeh.idVeh][1];
                        tiempo=repetid[idatras];
                        tiempoviaje[idatras][tiempo]= time -
vehiculo[infVeh.idVeh][2];

      suma[idatras]=suma[idatras]+tiempoviaje[idatras][tiempo];
                        promediotv[idatras] =
suma[idatras]/repetid[idatras];
                        repetid[idatras]=repetid[idatras]+1;
                        bandera[infVeh.idVeh]=0;


                  }//no tocar


      ////////////////////////////////////////////////////////////////
///
                  if ( (time > 60) && (infveh.report >= 0) &&
(fmod(time,60) == 0)){
                        orig[1]=108;
                        dest[1]=105;
                        pes[1]=promediotv[35939];

                        orig[2]=105;
                        dest[2]=81;
                        pes[2]=promediotv[35946];

                        orig[3]=81;
                        dest[3]=80;
                        pes[3]=promediotv[35949];

                        orig[4]=81;
                        dest[4]=75;
                        pes[4]=promediotv[35950];

                        orig[5]=76;
```

```
dest[5]=75;
pes[5]=promediotv[35954];

orig[6]=75;
dest[6]=17;
pes[6]=promediotv[35955];

orig[7]=17;
dest[7]=16;
pes[7]=promediotv[35963];

orig[8]=17;
dest[8]=12;
pes[8]=promediotv[35967];

orig[9]=14;
dest[9]=12;
pes[9]=promediotv[35969];

orig[10]=12;
dest[10]=5;
pes[10]=promediotv[35974];

orig[11]=5;
dest[11]=4;
pes[11]=promediotv[35977];

orig[12]=5;
dest[12]=3;
pes[12]=promediotv[35978];

orig[13]=108;
dest[13]=103;
pes[13]=promediotv[39290];

orig[14]=103;
dest[14]=104;
pes[14]=promediotv[39310];

orig[15]=103;
dest[15]=99;
pes[15]=promediotv[39311];

orig[16]=99;
dest[16]=100;
pes[16]=promediotv[39315];

orig[17]=99;
dest[17]=98;
pes[17]=promediotv[39316];

orig[18]=104;
dest[18]=96;
pes[18]=promediotv[39320];

orig[19]=98;
dest[19]=104;
```

```
pes[19]=promediotv[39321];

orig[20]=106;
dest[20]=111;
pes[20]=promediotv[39329];

orig[21]=106;
dest[21]=107;
pes[21]=promediotv[39332];

orig[22]=107;
dest[22]=111;
pes[22]=promediotv[39342];

orig[23]=112;
dest[23]=94;
pes[23]=promediotv[39375];

orig[24]=92;
dest[24]=91;
pes[24]=promediotv[39381];

orig[25]=92;
dest[25]=93;
pes[25]=promediotv[39382];

orig[26]=95;
dest[26]=107;
pes[26]=promediotv[39386];

orig[27]=101;
dest[27]=97;
pes[27]=promediotv[39387];

orig[28]=96;
dest[28]=98;
pes[28]=promediotv[39388];

orig[29]=98;
dest[29]=100;
pes[29]=promediotv[39389];

orig[30]=97;
dest[30]=80;
pes[30]=promediotv[39400];

orig[31]=80;
dest[31]=76;
pes[31]=promediotv[39401];

orig[32]=76;
dest[32]=36;
pes[32]=promediotv[39430];

orig[33]=16;
dest[33]=14;
pes[33]=promediotv[39441];
```

```
orig[34]=14;
dest[34]=7;
pes[34]=promediotv[39444];

orig[35]=9;
dest[35]=15;
pes[35]=promediotv[39451];

orig[36]=13;
dest[36]=74;
pes[36]=promediotv[39452];

orig[37]=13;
dest[37]=15;
pes[37]=promediotv[39458];

orig[38]=15;
dest[38]=18;
pes[38]=promediotv[39462];

orig[39]=36;
dest[39]=16;
pes[39]=promediotv[39469];

orig[40]=35;
dest[40]=37;
pes[40]=promediotv[39472];

orig[41]=74;
dest[41]=82;
pes[41]=promediotv[39476];

orig[42]=74;
dest[42]=73;
pes[42]=promediotv[39479];

orig[43]=73;
dest[43]=77;
pes[43]=promediotv[39483];

orig[44]=79;
dest[44]=83;
pes[44]=promediotv[39487];

orig[45]=79;
dest[45]=82;
pes[45]=promediotv[39497];

orig[46]=6;
dest[46]=4;
pes[46]=promediotv[39509];

orig[47]=4;
dest[47]=3;
pes[47]=promediotv[39510];
```

```
orig[48]=252;
dest[48]=8;
pes[48]=promediotv[39514];

orig[49]=252;
dest[49]=13;
pes[49]=promediotv[39536];

orig[50]=9;
dest[50]=7;
pes[50]=promediotv[39548];

orig[51]=8;
dest[51]=221;
pes[51]=promediotv[39565];

orig[52]=6;
dest[52]=8;
pes[52]=promediotv[39568];

orig[53]=35;
dest[53]=36;
pes[53]=promediotv[39649];

orig[54]=36;
dest[54]=35;
pes[54]=promediotv[39650];

orig[55]=82;
dest[55]=106;
pes[55]=promediotv[39681];

orig[56]=94;
dest[56]=96;
pes[56]=promediotv[39689];

orig[57]=97;
dest[57]=93;
pes[57]=promediotv[39690];

orig[58]=91;
dest[58]=113;
pes[58]=promediotv[39692];

orig[59]=222;
dest[59]=221;
pes[59]=promediotv[45328];

orig[60]=114;
dest[60]=86;
pes[60]=promediotv[45354];

orig[61]=114;
dest[61]=85;
pes[61]=promediotv[45355];

orig[62]=86;
```

```
dest[62]=87;
pes[62]=promediotv[45356];

orig[63]=112;
dest[63]=95;
pes[63]=promediotv[48127];

orig[64]=113;
dest[64]=90;
pes[64]=promediotv[48143];

orig[65]=84;
dest[65]=92;
pes[65]=promediotv[48150];

orig[66]=85;
dest[66]=86;
pes[66]=promediotv[48163];

orig[67]=113;
dest[67]=84;
pes[67]=promediotv[48169];

orig[68]=88;
dest[68]=89;
pes[68]=promediotv[48179];

orig[69]=87;
dest[69]=191;
pes[69]=promediotv[48188];

orig[70]=122;
dest[70]=121;
pes[70]=promediotv[48617];

orig[71]=122;
dest[71]=118;
pes[71]=promediotv[51600];

orig[72]=123;
dest[72]=125;
pes[72]=promediotv[51603];

orig[73]=138;
dest[73]=145;
pes[73]=promediotv[51609];

orig[74]=123;
dest[74]=122;
pes[74]=promediotv[51630];

orig[75]=119;
dest[75]=123;
pes[75]=promediotv[51632];

orig[76]=232;
dest[76]=233;
```

```
pes[76]=promediotv[54443];

orig[77]=87;
dest[77]=88;
pes[77]=promediotv[54446];

orig[78]=90;
dest[78]=114;
pes[78]=promediotv[54487];

orig[79]=89;
dest[79]=90;
pes[79]=promediotv[54499];

orig[80]=232;
dest[80]=58;
pes[80]=promediotv[54509];

orig[81]=77;
dest[81]=71;
pes[81]=promediotv[54511];

orig[82]=77;
dest[82]=78;
pes[82]=promediotv[54515];

orig[83]=78;
dest[83]=79;
pes[83]=promediotv[54525];

orig[84]=226;
dest[84]=207;
pes[84]=promediotv[54531];

orig[85]=232;
dest[85]=236;
pes[85]=promediotv[54532];

orig[86]=222;
dest[86]=223;
pes[86]=promediotv[54541];

orig[87]=225;
dest[87]=222;
pes[87]=promediotv[54548];

orig[88]=225;
dest[88]=223;
pes[88]=promediotv[54551];

orig[89]=224;
dest[89]=226;
pes[89]=promediotv[54565];

orig[90]=224;
dest[90]=222;
pes[90]=promediotv[54566];
```

```
orig[91]=222;
dest[91]=226;
pes[91]=promediotv[54575];

orig[92]=100;
dest[92]=102;
pes[92]=promediotv[54634];

orig[93]=137;
dest[93]=142;
pes[93]=promediotv[64131];

orig[94]=89;
dest[94]=112;
pes[94]=promediotv[75694];

orig[95]=138;
dest[95]=140;
pes[95]=promediotv[75762];

orig[96]=137;
dest[96]=138;
pes[96]=promediotv[75771];

orig[97]=138;
dest[97]=137;
pes[97]=promediotv[75782];

orig[98]=137;
dest[98]=136;
pes[98]=promediotv[75788];

orig[99]=119;
dest[99]=127;
pes[99]=promediotv[75794];

orig[100]=116;
dest[100]=182;
pes[100]=promediotv[75921];

orig[101]=250;
dest[101]=204;
pes[101]=promediotv[75925];

orig[102]=250;
dest[102]=160;
pes[102]=promediotv[75928];

orig[103]=250;
dest[103]=248;
pes[103]=promediotv[75934];

orig[104]=188;
dest[104]=187;
pes[104]=promediotv[75943];
```

```
orig[105]=188;
dest[105]=167;
pes[105]=promediotv[75947];

orig[106]=150;
dest[106]=147;
pes[106]=promediotv[78727];

orig[107]=116;
dest[107]=88;
pes[107]=promediotv[78774];

orig[108]=116;
dest[108]=190;
pes[108]=promediotv[78777];

orig[109]=118;
dest[109]=119;
pes[109]=promediotv[78813];

orig[110]=84;
dest[110]=85;
pes[110]=promediotv[87536];

orig[111]=126;
dest[111]=123;
pes[111]=promediotv[229224];

orig[112]=194;
dest[112]=195;
pes[112]=promediotv[258545];

orig[113]=228;
dest[113]=229;
pes[113]=promediotv[258546];

orig[114]=194;
dest[114]=176;
pes[114]=promediotv[258555];

orig[115]=146;
dest[115]=133;
pes[115]=promediotv[258589];

orig[116]=164;
dest[116]=158;
pes[116]=promediotv[258590];

orig[117]=144;
dest[117]=143;
pes[117]=promediotv[258591];

orig[118]=146;
dest[118]=144;
pes[118]=promediotv[258600];

orig[119]=200;
```

```
dest[119]=201;
pes[119]=promediotv[262165];

orig[120]=202;
dest[120]=203;
pes[120]=promediotv[262166];

orig[121]=201;
dest[121]=224;
pes[121]=promediotv[262167];

orig[122]=201;
dest[122]=250;
pes[122]=promediotv[262176];

orig[123]=202;
dest[123]=201;
pes[123]=promediotv[262187];

orig[124]=67;
dest[124]=68;
pes[124]=promediotv[333144];

orig[125]=67;
dest[125]=78;
pes[125]=promediotv[333146];

orig[126]=67;
dest[126]=66;
pes[126]=promediotv[333152];

orig[127]=66;
dest[127]=59;
pes[127]=promediotv[373556];

orig[128]=83;
dest[128]=84;
pes[128]=promediotv[373576];

orig[129]=191;
dest[129]=192;
pes[129]=promediotv[373590];

orig[130]=178;
dest[130]=179;
pes[130]=promediotv[373591];

orig[131]=191;
dest[131]=179;
pes[131]=promediotv[373592];

orig[132]=179;
dest[132]=180;
pes[132]=promediotv[373601];

orig[133]=118;
dest[133]=116;
```

```
pes[133]=promediotv[373615];

orig[134]=223;
dest[134]=201;
pes[134]=promediotv[374043];

orig[135]=253;
dest[135]=251;
pes[135]=promediotv[374079];

orig[136]=237;
dest[136]=230;
pes[136]=promediotv[374080];

orig[137]=236;
dest[137]=253;
pes[137]=promediotv[374081];

orig[138]=230;
dest[138]=231;
pes[138]=promediotv[374090];

orig[139]=253;
dest[139]=194;
pes[139]=promediotv[374101];

orig[140]=160;
dest[140]=161;
pes[140]=promediotv[374121];

orig[141]=160;
dest[141]=137;
pes[141]=promediotv[374125];

orig[142]=160;
dest[142]=162;
pes[142]=promediotv[374128];

orig[143]=197;
dest[143]=239;
pes[143]=promediotv[374143];

orig[144]=227;
dest[144]=240;
pes[144]=promediotv[374144];

orig[145]=227;
dest[145]=228;
pes[145]=promediotv[374145];

orig[146]=66;
dest[146]=61;
pes[146]=promediotv[374158];

orig[147]=27;
dest[147]=31;
pes[147]=promediotv[374159];
```

```
orig[148]=58;
dest[148]=57;
pes[148]=promediotv[374169];

orig[149]=52;
dest[149]=45;
pes[149]=promediotv[374171];

orig[150]=58;
dest[150]=67;
pes[150]=promediotv[374175];

orig[151]=58;
dest[151]=246;
pes[151]=promediotv[374178];

orig[152]=147;
dest[152]=132;
pes[152]=promediotv[385166];

orig[153]=147;
dest[153]=146;
pes[153]=promediotv[385167];

orig[154]=157;
dest[154]=165;
pes[154]=promediotv[385176];

orig[155]=158;
dest[155]=159;
pes[155]=promediotv[385186];

orig[156]=158;
dest[156]=134;
pes[156]=promediotv[385187];

orig[157]=151;
dest[157]=130;
pes[157]=promediotv[385200];

orig[158]=129;
dest[158]=127;
pes[158]=promediotv[385202];

orig[159]=127;
dest[159]=151;
pes[159]=promediotv[385203];

orig[160]=151;
dest[160]=150;
pes[160]=promediotv[385215];

orig[161]=18;
dest[161]=20;
pes[161]=promediotv[385223];
```

```
orig[162]=18;
dest[162]=19;
pes[162]=promediotv[385226];

orig[163]=23;
dest[163]=22;
pes[163]=promediotv[385235];

orig[164]=28;
dest[164]=29;
pes[164]=promediotv[385236];

orig[165]=28;
dest[165]=35;
pes[165]=promediotv[385237];

orig[166]=50;
dest[166]=255;
pes[166]=promediotv[396853];

orig[167]=35;
dest[167]=34;
pes[167]=promediotv[396867];

orig[168]=39;
dest[168]=38;
pes[168]=promediotv[396868];

orig[169]=37;
dest[169]=72;
pes[169]=promediotv[396869];

orig[170]=28;
dest[170]=34;
pes[170]=promediotv[396880];

orig[171]=34;
dest[171]=33;
pes[171]=promediotv[396881];

orig[172]=229;
dest[172]=230;
pes[172]=promediotv[396900];

orig[173]=221;
dest[173]=220;
pes[173]=promediotv[563780];

orig[174]=219;
dest[174]=220;
pes[174]=promediotv[563781];

orig[175]=219;
dest[175]=209;
pes[175]=promediotv[563782];

orig[176]=221;
```

```
dest[176]=9;
pes[176]=promediotv[563792];

orig[177]=221;
dest[177]=222;
pes[177]=promediotv[563798];

orig[178]=144;
dest[178]=145;
pes[178]=promediotv[774691];

orig[179]=145;
dest[179]=148;
pes[179]=promediotv[774692];

orig[180]=148;
dest[180]=147;
pes[180]=promediotv[774704];

orig[181]=148;
dest[181]=149;
pes[181]=promediotv[774707];

orig[182]=228;
dest[182]=210;
pes[182]=promediotv[912218];

orig[183]=243;
dest[183]=244;
pes[183]=promediotv[912225];

orig[184]=189;
dest[184]=177;
pes[184]=promediotv[912227];

orig[185]=177;
dest[185]=193;
pes[185]=promediotv[912229];

orig[186]=244;
dest[186]=245;
pes[186]=promediotv[912230];

orig[187]=245;
dest[187]=192;
pes[187]=promediotv[912231];

orig[188]=251;
dest[188]=237;
pes[188]=promediotv[912261];

orig[189]=237;
dest[189]=236;
pes[189]=promediotv[912262];

orig[190]=236;
dest[190]=235;
```

```
pes[190]=promediotv[912263];

orig[191]=247;
dest[191]=235;
pes[191]=promediotv[912264];

orig[192]=235;
dest[192]=234;
pes[192]=promediotv[912265];

orig[193]=234;
dest[193]=233;
pes[193]=promediotv[912267];

orig[194]=207;
dest[194]=208;
pes[194]=promediotv[912270];

orig[195]=208;
dest[195]=220;
pes[195]=promediotv[912271];

orig[196]=209;
dest[196]=208;
pes[196]=promediotv[912272];

orig[197]=209;
dest[197]=227;
pes[197]=promediotv[912273];

orig[198]=220;
dest[198]=219;
pes[198]=promediotv[912279];

orig[199]=207;
dest[199]=227;
pes[199]=promediotv[912282];

orig[200]=220;
dest[200]=221;
pes[200]=promediotv[912289];

orig[201]=210;
dest[201]=209;
pes[201]=promediotv[912298];

orig[202]=210;
dest[202]=217;
pes[202]=promediotv[912299];

orig[203]=217;
dest[203]=52;
pes[203]=promediotv[912300];

orig[204]=211;
dest[204]=229;
pes[204]=promediotv[912307];
```

```
orig[205]=213;
dest[205]=212;
pes[205]=promediotv[912310];

orig[206]=212;
dest[206]=211;
pes[206]=promediotv[912311];

orig[207]=211;
dest[207]=210;
pes[207]=promediotv[912312];

orig[208]=215;
dest[208]=214;
pes[208]=promediotv[912313];

orig[209]=45;
dest[209]=44;
pes[209]=promediotv[912314];

orig[210]=45;
dest[210]=46;
pes[210]=promediotv[912320];

orig[211]=47;
dest[211]=51;
pes[211]=promediotv[912326];

orig[212]=47;
dest[212]=45;
pes[212]=promediotv[912329];

orig[213]=33;
dest[213]=32;
pes[213]=promediotv[912330];

orig[214]=29;
dest[214]=30;
pes[214]=promediotv[912331];

orig[215]=27;
dest[215]=23;
pes[215]=promediotv[912332];

orig[216]=27;
dest[216]=26;
pes[216]=promediotv[912333];

orig[217]=45;
dest[217]=27;
pes[217]=promediotv[912334];

orig[218]=19;
dest[218]=21;
pes[218]=promediotv[912335];
```

```
orig[219]=21;
dest[219]=26;
pes[219]=promediotv[912336];

orig[220]=26;
dest[220]=46;
pes[220]=promediotv[912337];

orig[221]=71;
dest[221]=72;
pes[221]=promediotv[912338];

orig[222]=40;
dest[222]=39;
pes[222]=promediotv[912339];

orig[223]=31;
dest[223]=43;
pes[223]=promediotv[912340];

orig[224]=44;
dest[224]=53;
pes[224]=promediotv[912341];

orig[225]=246;
dest[225]=55;
pes[225]=promediotv[912342];

orig[226]=42;
dest[226]=54;
pes[226]=promediotv[912343];

orig[227]=54;
dest[227]=216;
pes[227]=promediotv[912344];

orig[228]=216;
dest[228]=211;
pes[228]=promediotv[912345];

orig[229]=216;
dest[229]=215;
pes[229]=promediotv[912346];

orig[230]=219;
dest[230]=218;
pes[230]=promediotv[912347];

orig[231]=254;
dest[231]=255;
pes[231]=promediotv[912348];

orig[232]=51;
dest[232]=254;
pes[232]=promediotv[912349];

orig[233]=254;
```

```
dest[233]=218;
pes[233]=promediotv[912350];

orig[234]=218;
dest[234]=217;
pes[234]=promediotv[912351];

orig[235]=217;
dest[235]=216;
pes[235]=promediotv[912352];

orig[236]=32;
dest[236]=41;
pes[236]=promediotv[912354];

orig[237]=34;
dest[237]=38;
pes[237]=promediotv[912355];

orig[238]=38;
dest[238]=63;
pes[238]=promediotv[912356];

orig[239]=63;
dest[239]=64;
pes[239]=promediotv[912357];

orig[240]=64;
dest[240]=69;
pes[240]=promediotv[912358];

orig[241]=63;
dest[241]=62;
pes[241]=promediotv[912359];

orig[242]=65;
dest[242]=64;
pes[242]=promediotv[912360];

orig[243]=69;
dest[243]=68;
pes[243]=promediotv[912361];

orig[244]=69;
dest[244]=70;
pes[244]=promediotv[912362];

orig[245]=39;
dest[245]=33;
pes[245]=promediotv[912363];

orig[246]=33;
dest[246]=29;
pes[246]=promediotv[912364];

orig[247]=29;
dest[247]=23;
```

```
pes[247]=promediotv[912365];

orig[248]=23;
dest[248]=21;
pes[248]=promediotv[912366];

orig[249]=21;
dest[249]=20;
pes[249]=promediotv[912367];

orig[250]=26;
dest[250]=24;
pes[250]=promediotv[912368];

orig[251]=62;
dest[251]=61;
pes[251]=promediotv[912369];

orig[252]=61;
dest[252]=60;
pes[252]=promediotv[912370];

orig[253]=60;
dest[253]=57;
pes[253]=promediotv[912371];

orig[254]=59;
dest[254]=60;
pes[254]=promediotv[912372];

orig[255]=55;
dest[255]=56;
pes[255]=promediotv[912375];

orig[256]=231;
dest[256]=212;
pes[256]=promediotv[912376];

orig[257]=212;
dest[257]=215;
pes[257]=promediotv[912377];

orig[258]=49;
dest[258]=47;
pes[258]=promediotv[912378];

orig[259]=46;
dest[259]=48;
pes[259]=promediotv[912379];

orig[260]=195;
dest[260]=251;
pes[260]=promediotv[912380];

orig[261]=175;
dest[261]=195;
pes[261]=promediotv[912381];
```

```
orig[262]=168;
dest[262]=175;
pes[262]=promediotv[912382];

orig[263]=174;
dest[263]=196;
pes[263]=promediotv[912384];

orig[264]=198;
dest[264]=173;
pes[264]=promediotv[912385];

orig[265]=205;
dest[265]=199;
pes[265]=promediotv[912386];

orig[266]=206;
dest[266]=207;
pes[266]=promediotv[912387];

orig[267]=171;
dest[267]=172;
pes[267]=promediotv[912388];

orig[268]=169;
dest[268]=174;
pes[268]=promediotv[912389];

orig[269]=200;
dest[269]=206;
pes[269]=promediotv[912390];

orig[270]=199;
dest[270]=200;
pes[270]=promediotv[912391];

orig[271]=203;
dest[271]=200;
pes[271]=promediotv[912392];

orig[272]=172;
dest[272]=203;
pes[272]=promediotv[912393];

orig[273]=204;
dest[273]=172;
pes[273]=promediotv[912394];

orig[274]=172;
dest[274]=173;
pes[274]=promediotv[912395];

orig[275]=198;
dest[275]=199;
pes[275]=promediotv[912396];
```

```
orig[276]=238;
dest[276]=237;
pes[276]=promediotv[912397];

orig[277]=196;
dest[277]=197;
pes[277]=promediotv[912398];

orig[278]=177;
dest[278]=178;
pes[278]=promediotv[912399];

orig[279]=192;
dest[279]=178;
pes[279]=promediotv[912400];

orig[280]=178;
dest[280]=181;
pes[280]=promediotv[912401];

orig[281]=181;
dest[281]=182;
pes[281]=promediotv[912402];

orig[282]=176;
dest[282]=177;
pes[282]=promediotv[912403];

orig[283]=192;
dest[283]=193;
pes[283]=promediotv[912404];

orig[284]=193;
dest[284]=247;
pes[284]=promediotv[912405];

orig[285]=245;
dest[285]=247;
pes[285]=promediotv[912406];

orig[286]=247;
dest[286]=234;
pes[286]=promediotv[912407];

orig[287]=234;
dest[287]=241;
pes[287]=promediotv[912408];

orig[288]=241;
dest[288]=244;
pes[288]=promediotv[912409];

orig[289]=193;
dest[289]=194;
pes[289]=promediotv[912410];

orig[290]=180;
```

```
dest[290]=181;
pes[290]=promediotv[912411];

orig[291]=182;
dest[291]=183;
pes[291]=promediotv[912413];

orig[292]=183;
dest[292]=190;
pes[292]=promediotv[912414];

orig[293]=183;
dest[293]=184;
pes[293]=promediotv[912415];

orig[294]=186;
dest[294]=189;
pes[294]=promediotv[912416];

orig[295]=130;
dest[295]=153;
pes[295]=promediotv[912417];

orig[296]=186;
dest[296]=183;
pes[296]=promediotv[912418];

orig[297]=129;
dest[297]=128;
pes[297]=promediotv[912419];

orig[298]=165;
dest[298]=169;
pes[298]=promediotv[912420];

orig[299]=156;
dest[299]=166;
pes[299]=promediotv[912421];

orig[300]=175;
dest[300]=176;
pes[300]=promediotv[912422];

orig[301]=174;
dest[301]=175;
pes[301]=promediotv[912423];

orig[302]=173;
dest[302]=174;
pes[302]=promediotv[912425];

orig[303]=173;
dest[303]=170;
pes[303]=promediotv[912426];

orig[304]=131;
dest[304]=130;
```

```
pes[304]=promediotv[912427];

orig[305]=132;
dest[305]=131;
pes[305]=promediotv[912428];

orig[306]=150;
dest[306]=149;
pes[306]=promediotv[912429];

orig[307]=152;
dest[307]=151;
pes[307]=promediotv[912430];

orig[308]=159;
dest[308]=163;
pes[308]=promediotv[912437];

orig[309]=163;
dest[309]=171;
pes[309]=promediotv[912438];

orig[310]=170;
dest[310]=164;
pes[310]=promediotv[912439];

orig[311]=185;
dest[311]=184;
pes[311]=promediotv[912440];

orig[312]=184;
dest[312]=153;
pes[312]=promediotv[912441];

orig[313]=187;
dest[313]=186;
pes[313]=promediotv[912442];

orig[314]=153;
dest[314]=154;
pes[314]=promediotv[912443];

orig[315]=155;
dest[315]=156;
pes[315]=promediotv[912444];

orig[316]=154;
dest[316]=155;
pes[316]=promediotv[912450];

orig[317]=156;
dest[317]=157;
pes[317]=promediotv[912451];

orig[318]=127;
dest[318]=126;
pes[318]=promediotv[912454];
```

```
orig[319]=162;
dest[319]=163;
pes[319]=promediotv[912455];

orig[320]=163;
dest[320]=164;
pes[320]=promediotv[912456];

orig[321]=164;
dest[321]=165;
pes[321]=promediotv[912457];

orig[322]=165;
dest[322]=166;
pes[322]=promediotv[912458];

orig[323]=143;
dest[323]=135;
pes[323]=promediotv[912459];

orig[324]=135;
dest[324]=159;
pes[324]=promediotv[912460];

orig[325]=136;
dest[325]=135;
pes[325]=promediotv[912461];

orig[326]=135;
dest[326]=134;
pes[326]=promediotv[912462];

orig[327]=134;
dest[327]=133;
pes[327]=promediotv[912463];

orig[328]=133;
dest[328]=132;
pes[328]=promediotv[912464];

orig[329]=62;
dest[329]=39;
pes[329]=promediotv[912465];

orig[330]=71;
dest[330]=70;
pes[330]=promediotv[912468];

orig[331]=72;
dest[331]=73;
pes[331]=promediotv[912473];

orig[332]=70;
dest[332]=67;
pes[332]=promediotv[912477];
```

```
orig[333]=38;
dest[333]=37;
pes[333]=promediotv[912482];

orig[334]=68;
dest[334]=65;
pes[334]=promediotv[912485];

orig[335]=65;
dest[335]=62;
pes[335]=promediotv[912488];

orig[336]=19;
dest[336]=22;
pes[336]=promediotv[912520];

orig[337]=20;
dest[337]=24;
pes[337]=promediotv[912539];

orig[338]=25;
dest[338]=26;
pes[338]=promediotv[912543];

orig[339]=26;
dest[339]=27;
pes[339]=promediotv[912544];

orig[340]=24;
dest[340]=25;
pes[340]=promediotv[912545];

orig[341]=25;
dest[341]=48;
pes[341]=promediotv[912548];

orig[342]=61;
dest[342]=40;
pes[342]=promediotv[912551];

orig[343]=61;
dest[343]=66;
pes[343]=promediotv[912554];

orig[344]=40;
dest[344]=32;
pes[344]=promediotv[912569];

orig[345]=40;
dest[345]=61;
pes[345]=promediotv[912572];

orig[346]=32;
dest[346]=40;
pes[346]=promediotv[912580];

orig[347]=32;
```

```
dest[347]=30;
pes[347]=promediotv[912583];

orig[348]=57;
dest[348]=41;
pes[348]=promediotv[912592];

orig[349]=41;
dest[349]=43;
pes[349]=promediotv[912596];

orig[350]=42;
dest[350]=56;
pes[350]=promediotv[912601];

orig[351]=30;
dest[351]=27;
pes[351]=promediotv[912607];

orig[352]=31;
dest[352]=32;
pes[352]=promediotv[912610];

orig[353]=43;
dest[353]=45;
pes[353]=promediotv[912635];

orig[354]=59;
dest[354]=58;
pes[354]=promediotv[912655];

orig[355]=56;
dest[355]=58;
pes[355]=promediotv[912659];

orig[356]=55;
dest[356]=54;
pes[356]=promediotv[912663];

orig[357]=54;
dest[357]=53;
pes[357]=promediotv[912664];

orig[358]=53;
dest[358]=52;
pes[358]=promediotv[912665];

orig[359]=52;
dest[359]=51;
pes[359]=promediotv[912666];

orig[360]=51;
dest[360]=50;
pes[360]=promediotv[912667];

orig[361]=44;
dest[361]=42;
```

```
pes[361]=promediotv[912668];

orig[362]=48;
dest[362]=49;
pes[362]=promediotv[912674];

orig[363]=49;
dest[363]=50;
pes[363]=promediotv[912677];

orig[364]=246;
dest[364]=214;
pes[364]=promediotv[912682];

orig[365]=214;
dest[365]=213;
pes[365]=promediotv[912708];

orig[366]=213;
dest[366]=232;
pes[366]=promediotv[912712];

orig[367]=255;
dest[367]=219;
pes[367]=promediotv[912715];

orig[368]=239;
dest[368]=238;
pes[368]=promediotv[912766];

orig[369]=240;
dest[369]=239;
pes[369]=promediotv[912767];

orig[370]=205;
dest[370]=240;
pes[370]=promediotv[912768];

orig[371]=206;
dest[371]=205;
pes[371]=promediotv[912769];

orig[372]=231;
dest[372]=232;
pes[372]=promediotv[912770];

orig[373]=195;
dest[373]=196;
pes[373]=promediotv[912798];

orig[374]=143;
dest[374]=137;
pes[374]=promediotv[912815];

orig[375]=134;
dest[375]=144;
pes[375]=promediotv[912819];
```

```
orig[376]=157;
dest[376]=158;
pes[376]=promediotv[912825];

orig[377]=240;
dest[377]=198;
pes[377]=promediotv[912833];

orig[378]=239;
dest[378]=227;
pes[378]=promediotv[912836];

orig[379]=197;
dest[379]=198;
pes[379]=promediotv[912845];

orig[380]=196;
dest[380]=238;
pes[380]=promediotv[912853];

orig[381]=238;
dest[381]=228;
pes[381]=promediotv[912854];

orig[382]=233;
dest[382]=242;
pes[382]=promediotv[912857];

orig[383]=242;
dest[383]=243;
pes[383]=promediotv[912861];

orig[384]=242;
dest[384]=241;
pes[384]=promediotv[912865];

orig[385]=243;
dest[385]=87;
pes[385]=promediotv[912868];

orig[386]=189;
dest[386]=188;
pes[386]=promediotv[912882];

orig[387]=176;
dest[387]=188;
pes[387]=promediotv[912890];

orig[388]=204;
dest[388]=202;
pes[388]=promediotv[912917];

orig[389]=171;
dest[389]=250;
pes[389]=promediotv[912921];
```

```
orig[390]=170;
dest[390]=171;
pes[390]=promediotv[912950];

orig[391]=168;
dest[391]=169;
pes[391]=promediotv[912956];

orig[392]=167;
dest[392]=168;
pes[392]=promediotv[912960];

orig[393]=169;
dest[393]=170;
pes[393]=promediotv[912972];

orig[394]=166;
dest[394]=167;
pes[394]=promediotv[912978];

orig[395]=154;
dest[395]=131;
pes[395]=promediotv[913000];

orig[396]=182;
dest[396]=189;
pes[396]=promediotv[913014];

orig[397]=8;
dest[397]=9;
pes[397]=promediotv[913028];

orig[398]=180;
dest[398]=116;
pes[398]=promediotv[913036];

orig[399]=190;
dest[399]=185;
pes[399]=promediotv[913041];

orig[400]=162;
dest[400]=250;
pes[400]=promediotv[913050];

orig[401]=159;
dest[401]=160;
pes[401]=promediotv[913054];

orig[402]=184;
dest[402]=129;
pes[402]=promediotv[913060];

orig[403]=128;
dest[403]=119;
pes[403]=promediotv[913070];

orig[404]=185;
```

```
dest[404]=128;
pes[404]=promediotv[913074];

orig[405]=130;
dest[405]=129;
pes[405]=promediotv[913081];

orig[406]=136;
dest[406]=160;
pes[406]=promediotv[913092];

orig[407]=133;
dest[407]=157;
pes[407]=promediotv[913101];

orig[408]=132;
dest[408]=155;
pes[408]=promediotv[913107];

orig[409]=149;
dest[409]=152;
pes[409]=promediotv[913115];

orig[410]=152;
dest[410]=126;
pes[410]=promediotv[913121];

orig[411]=131;
dest[411]=150;
pes[411]=promediotv[913125];

orig[412]=153;
dest[412]=186;
pes[412]=promediotv[913131];

orig[413]=187;
dest[413]=154;
pes[413]=promediotv[913134];

orig[414]=7;
dest[414]=6;
pes[414]=promediotv[913141];

orig[415]=7;
dest[415]=11;
pes[415]=promediotv[913145];

orig[416]=10;
dest[416]=6;
pes[416]=promediotv[913146];

orig[417]=3;
dest[417]=2;
pes[417]=promediotv[913157];

orig[418]=96;
dest[418]=97;
```

```
pes[418]=promediotv[913170];

orig[419]=93;
dest[419]=94;
pes[419]=promediotv[913181];

orig[420]=94;
dest[420]=95;
pes[420]=promediotv[913182];

orig[421]=22;
dest[421]=28;
pes[421]=promediotv[913219];

orig[422]=1;
dest[422]=252;
pes[422]=promediotv[913233];

orig[423]=115;
dest[423]=88;
pes[423]=promediotv[913238];

orig[424]=117;
dest[424]=116;
pes[424]=promediotv[913243];

orig[425]=120;
dest[425]=122;
pes[425]=promediotv[913271];

orig[426]=124;
dest[426]=122;
pes[426]=promediotv[913277];

orig[427]=109;
dest[427]=108;
pes[427]=promediotv[913283];

orig[428]=110;
dest[428]=105;
pes[428]=promediotv[913286];

orig[429]=139;
dest[429]=138;
pes[429]=promediotv[913297];

orig[430]=141;
dest[430]=138;
pes[430]=promediotv[913305];

orig[431]=249;
dest[431]=225;
pes[431]=promediotv[913321];

secc2nodo[35939]=108;
secc2nodo[35946]=105;
secc2nodo[35949]=81;
```

```
secc2nodo[35950]=81;
secc2nodo[35954]=76;
secc2nodo[35955]=75;
secc2nodo[35963]=17;
secc2nodo[35967]=17;
secc2nodo[35969]=14;
secc2nodo[35974]=12;
secc2nodo[35977]=5;
secc2nodo[35978]=5;
secc2nodo[39290]=108;
secc2nodo[39310]=103;
secc2nodo[39311]=103;
secc2nodo[39315]=99;
secc2nodo[39316]=99;
secc2nodo[39320]=104;
secc2nodo[39321]=98;
secc2nodo[39329]=106;
secc2nodo[39332]=106;
secc2nodo[39342]=107;
secc2nodo[39375]=112;
secc2nodo[39381]=92;
secc2nodo[39382]=92;
secc2nodo[39386]=95;
secc2nodo[39387]=101;
secc2nodo[39388]=96;
secc2nodo[39389]=98;
secc2nodo[39400]=97;
secc2nodo[39401]=80;
secc2nodo[39430]=76;
secc2nodo[39441]=16;
secc2nodo[39444]=14;
secc2nodo[39451]=9;
secc2nodo[39452]=13;
secc2nodo[39458]=13;
secc2nodo[39462]=15;
secc2nodo[39469]=36;
secc2nodo[39472]=35;
secc2nodo[39476]=74;
secc2nodo[39479]=74;
secc2nodo[39483]=73;
secc2nodo[39487]=79;
secc2nodo[39497]=79;
secc2nodo[39509]=6;
secc2nodo[39510]=4;
secc2nodo[39514]=252;
secc2nodo[39536]=252;
secc2nodo[39548]=9;
secc2nodo[39565]=8;
secc2nodo[39568]=6;
secc2nodo[39649]=35;
secc2nodo[39650]=36;
secc2nodo[39681]=82;
secc2nodo[39689]=94;
secc2nodo[39690]=97;
secc2nodo[39692]=91;
secc2nodo[45328]=222;
secc2nodo[45354]=114;
```

```
secc2nodo[45355]=114;
secc2nodo[45356]=86;
secc2nodo[48127]=112;
secc2nodo[48143]=113;
secc2nodo[48150]=84;
secc2nodo[48163]=85;
secc2nodo[48169]=113;
secc2nodo[48179]=88;
secc2nodo[48188]=87;
secc2nodo[48617]=122;
secc2nodo[51600]=122;
secc2nodo[51603]=123;
secc2nodo[51609]=138;
secc2nodo[51630]=123;
secc2nodo[51632]=119;
secc2nodo[54443]=232;
secc2nodo[54446]=87;
secc2nodo[54487]=90;
secc2nodo[54499]=89;
secc2nodo[54509]=232;
secc2nodo[54511]=77;
secc2nodo[54515]=77;
secc2nodo[54525]=78;
secc2nodo[54531]=226;
secc2nodo[54532]=232;
secc2nodo[54541]=222;
secc2nodo[54548]=225;
secc2nodo[54551]=225;
secc2nodo[54565]=224;
secc2nodo[54566]=224;
secc2nodo[54575]=222;
secc2nodo[54634]=100;
secc2nodo[64131]=137;
secc2nodo[75694]=89;
secc2nodo[75762]=138;
secc2nodo[75771]=137;
secc2nodo[75782]=138;
secc2nodo[75788]=137;
secc2nodo[75794]=119;
secc2nodo[75921]=116;
secc2nodo[75925]=250;
secc2nodo[75928]=250;
secc2nodo[75934]=250;
secc2nodo[75943]=188;
secc2nodo[75947]=188;
secc2nodo[78727]=150;
secc2nodo[78774]=116;
secc2nodo[78777]=116;
secc2nodo[78813]=118;
secc2nodo[87536]=84;
secc2nodo[229224]=126;
secc2nodo[258545]=194;
secc2nodo[258546]=228;
secc2nodo[258555]=194;
secc2nodo[258589]=146;
secc2nodo[258590]=164;
secc2nodo[258591]=144;
```

```
secc2nodo[258600]=146;
secc2nodo[262165]=200;
secc2nodo[262166]=202;
secc2nodo[262167]=201;
secc2nodo[262176]=201;
secc2nodo[262187]=202;
secc2nodo[333144]=67;
secc2nodo[333146]=67;
secc2nodo[333152]=67;
secc2nodo[373556]=66;
secc2nodo[373576]=83;
secc2nodo[373590]=191;
secc2nodo[373591]=178;
secc2nodo[373592]=191;
secc2nodo[373601]=179;
secc2nodo[373615]=118;
secc2nodo[374043]=223;
secc2nodo[374079]=253;
secc2nodo[374080]=237;
secc2nodo[374081]=236;
secc2nodo[374090]=230;
secc2nodo[374101]=253;
secc2nodo[374121]=160;
secc2nodo[374125]=160;
secc2nodo[374128]=160;
secc2nodo[374143]=197;
secc2nodo[374144]=227;
secc2nodo[374145]=227;
secc2nodo[374158]=66;
secc2nodo[374159]=27;
secc2nodo[374169]=58;
secc2nodo[374171]=52;
secc2nodo[374175]=58;
secc2nodo[374178]=58;
secc2nodo[385166]=147;
secc2nodo[385167]=147;
secc2nodo[385176]=157;
secc2nodo[385186]=158;
secc2nodo[385187]=158;
secc2nodo[385200]=151;
secc2nodo[385202]=129;
secc2nodo[385203]=127;
secc2nodo[385215]=151;
secc2nodo[385223]=18;
secc2nodo[385226]=18;
secc2nodo[385235]=23;
secc2nodo[385236]=28;
secc2nodo[385237]=28;
secc2nodo[396853]=50;
secc2nodo[396867]=35;
secc2nodo[396868]=39;
secc2nodo[396869]=37;
secc2nodo[396880]=28;
secc2nodo[396881]=34;
secc2nodo[396900]=229;
secc2nodo[563780]=221;
secc2nodo[563781]=219;
```

```
secc2nodo[563782]=219;
secc2nodo[563792]=221;
secc2nodo[563798]=221;
secc2nodo[774691]=144;
secc2nodo[774692]=145;
secc2nodo[774704]=148;
secc2nodo[774707]=148;
secc2nodo[912218]=228;
secc2nodo[912225]=243;
secc2nodo[912227]=189;
secc2nodo[912229]=177;
secc2nodo[912230]=244;
secc2nodo[912231]=245;
secc2nodo[912261]=251;
secc2nodo[912262]=237;
secc2nodo[912263]=236;
secc2nodo[912264]=247;
secc2nodo[912265]=235;
secc2nodo[912267]=234;
secc2nodo[912270]=207;
secc2nodo[912271]=208;
secc2nodo[912272]=209;
secc2nodo[912273]=209;
secc2nodo[912279]=220;
secc2nodo[912282]=207;
secc2nodo[912289]=220;
secc2nodo[912298]=210;
secc2nodo[912299]=210;
secc2nodo[912300]=217;
secc2nodo[912307]=211;
secc2nodo[912310]=213;
secc2nodo[912311]=212;
secc2nodo[912312]=211;
secc2nodo[912313]=215;
secc2nodo[912314]=45;
secc2nodo[912320]=45;
secc2nodo[912326]=47;
secc2nodo[912329]=47;
secc2nodo[912330]=33;
secc2nodo[912331]=29;
secc2nodo[912332]=27;
secc2nodo[912333]=27;
secc2nodo[912334]=45;
secc2nodo[912335]=19;
secc2nodo[912336]=21;
secc2nodo[912337]=26;
secc2nodo[912338]=71;
secc2nodo[912339]=40;
secc2nodo[912340]=31;
secc2nodo[912341]=44;
secc2nodo[912342]=246;
secc2nodo[912343]=42;
secc2nodo[912344]=54;
secc2nodo[912345]=216;
secc2nodo[912346]=216;
secc2nodo[912347]=219;
secc2nodo[912348]=254;
```

```
secc2nodo[912349]=51;
secc2nodo[912350]=254;
secc2nodo[912351]=218;
secc2nodo[912352]=217;
secc2nodo[912354]=32;
secc2nodo[912355]=34;
secc2nodo[912356]=38;
secc2nodo[912357]=63;
secc2nodo[912358]=64;
secc2nodo[912359]=63;
secc2nodo[912360]=65;
secc2nodo[912361]=69;
secc2nodo[912362]=69;
secc2nodo[912363]=39;
secc2nodo[912364]=33;
secc2nodo[912365]=29;
secc2nodo[912366]=23;
secc2nodo[912367]=21;
secc2nodo[912368]=26;
secc2nodo[912369]=62;
secc2nodo[912370]=61;
secc2nodo[912371]=60;
secc2nodo[912372]=59;
secc2nodo[912375]=55;
secc2nodo[912376]=231;
secc2nodo[912377]=212;
secc2nodo[912378]=49;
secc2nodo[912379]=46;
secc2nodo[912380]=195;
secc2nodo[912381]=175;
secc2nodo[912382]=168;
secc2nodo[912384]=174;
secc2nodo[912385]=198;
secc2nodo[912386]=205;
secc2nodo[912387]=206;
secc2nodo[912388]=171;
secc2nodo[912389]=169;
secc2nodo[912390]=200;
secc2nodo[912391]=199;
secc2nodo[912392]=203;
secc2nodo[912393]=172;
secc2nodo[912394]=204;
secc2nodo[912395]=172;
secc2nodo[912396]=198;
secc2nodo[912397]=238;
secc2nodo[912398]=196;
secc2nodo[912399]=177;
secc2nodo[912400]=192;
secc2nodo[912401]=178;
secc2nodo[912402]=181;
secc2nodo[912403]=176;
secc2nodo[912404]=192;
secc2nodo[912405]=193;
secc2nodo[912406]=245;
secc2nodo[912407]=247;
secc2nodo[912408]=234;
secc2nodo[912409]=241;
```

```
secc2nodo[912410]=193;
secc2nodo[912411]=180;
secc2nodo[912413]=182;
secc2nodo[912414]=183;
secc2nodo[912415]=183;
secc2nodo[912416]=186;
secc2nodo[912417]=130;
secc2nodo[912418]=186;
secc2nodo[912419]=129;
secc2nodo[912420]=165;
secc2nodo[912421]=156;
secc2nodo[912422]=175;
secc2nodo[912423]=174;
secc2nodo[912425]=173;
secc2nodo[912426]=173;
secc2nodo[912427]=131;
secc2nodo[912428]=132;
secc2nodo[912429]=150;
secc2nodo[912430]=152;
secc2nodo[912437]=159;
secc2nodo[912438]=163;
secc2nodo[912439]=170;
secc2nodo[912440]=185;
secc2nodo[912441]=184;
secc2nodo[912442]=187;
secc2nodo[912443]=153;
secc2nodo[912444]=155;
secc2nodo[912450]=154;
secc2nodo[912451]=156;
secc2nodo[912454]=127;
secc2nodo[912455]=162;
secc2nodo[912456]=163;
secc2nodo[912457]=164;
secc2nodo[912458]=165;
secc2nodo[912459]=143;
secc2nodo[912460]=135;
secc2nodo[912461]=136;
secc2nodo[912462]=135;
secc2nodo[912463]=134;
secc2nodo[912464]=133;
secc2nodo[912465]=62;
secc2nodo[912468]=71;
secc2nodo[912473]=72;
secc2nodo[912477]=70;
secc2nodo[912482]=38;
secc2nodo[912485]=68;
secc2nodo[912488]=65;
secc2nodo[912520]=19;
secc2nodo[912539]=20;
secc2nodo[912543]=25;
secc2nodo[912544]=26;
secc2nodo[912545]=24;
secc2nodo[912548]=25;
secc2nodo[912551]=61;
secc2nodo[912554]=61;
secc2nodo[912569]=40;
secc2nodo[912572]=40;
```

```
secc2nodo[912580]=32;
secc2nodo[912583]=32;
secc2nodo[912592]=57;
secc2nodo[912596]=41;
secc2nodo[912601]=42;
secc2nodo[912607]=30;
secc2nodo[912610]=31;
secc2nodo[912635]=43;
secc2nodo[912655]=59;
secc2nodo[912659]=56;
secc2nodo[912663]=55;
secc2nodo[912664]=54;
secc2nodo[912665]=53;
secc2nodo[912666]=52;
secc2nodo[912667]=51;
secc2nodo[912668]=44;
secc2nodo[912674]=48;
secc2nodo[912677]=49;
secc2nodo[912682]=246;
secc2nodo[912708]=214;
secc2nodo[912712]=213;
secc2nodo[912715]=255;
secc2nodo[912766]=239;
secc2nodo[912767]=240;
secc2nodo[912768]=205;
secc2nodo[912769]=206;
secc2nodo[912770]=231;
secc2nodo[912798]=195;
secc2nodo[912815]=143;
secc2nodo[912819]=134;
secc2nodo[912825]=157;
secc2nodo[912833]=240;
secc2nodo[912836]=239;
secc2nodo[912845]=197;
secc2nodo[912853]=196;
secc2nodo[912854]=238;
secc2nodo[912857]=233;
secc2nodo[912861]=242;
secc2nodo[912865]=242;
secc2nodo[912868]=243;
secc2nodo[912882]=189;
secc2nodo[912890]=176;
secc2nodo[912917]=204;
secc2nodo[912921]=171;
secc2nodo[912950]=170;
secc2nodo[912956]=168;
secc2nodo[912960]=167;
secc2nodo[912972]=169;
secc2nodo[912978]=166;
secc2nodo[913000]=154;
secc2nodo[913014]=182;
secc2nodo[913028]=8;
secc2nodo[913036]=180;
secc2nodo[913041]=190;
secc2nodo[913050]=162;
secc2nodo[913054]=159;
secc2nodo[913060]=184;
```

```
secc2nodo[913070]=128;
secc2nodo[913074]=185;
secc2nodo[913081]=130;
secc2nodo[913092]=136;
secc2nodo[913101]=133;
secc2nodo[913107]=132;
secc2nodo[913115]=149;
secc2nodo[913121]=152;
secc2nodo[913125]=131;
secc2nodo[913131]=153;
secc2nodo[913134]=187;
secc2nodo[913141]=7;
secc2nodo[913145]=7;
secc2nodo[913146]=10;
secc2nodo[913157]=3;
secc2nodo[913170]=96;
secc2nodo[913181]=93;
secc2nodo[913182]=94;
secc2nodo[913219]=22;
secc2nodo[913233]=1;
secc2nodo[913238]=115;
secc2nodo[913243]=117;
secc2nodo[913271]=120;
secc2nodo[913277]=124;
secc2nodo[913283]=109;
secc2nodo[913286]=110;
secc2nodo[913297]=139;
secc2nodo[913305]=141;
secc2nodo[913321]=249;


centroidest[913210]=2;
centroidest[913209]=1;
centroidest[913211]=10;
centroidest[913212]=11;
centroidest[913236]=101;
centroidest[913237]=102;
centroidest[913227]=109;
centroidest[913228]=110;
centroidest[913229]=111;
centroidest[913241]=115;
centroidest[913242]=117;
centroidest[913246]=120;
centroidest[913247]=121;
centroidest[913248]=124;
centroidest[913249]=125;
centroidest[913250]=139;
centroidest[913251]=140;
centroidest[913252]=141;
centroidest[913253]=142;
centroidest[913254]=161;
centroidest[913255]=248;
centroidest[913256]=249;
```

```
                        V=255;
                        E=431;

                        for (r=1;r<=E;r++)
                        {
                              origen=orig[r];
                              destino=dest[r];
                              peso=pes[r];

                              ady[ origen ].push_back( Node( destino ,
peso ) ); //consideremos grafo dirigido
                              //ady[ destino ].push_back( Node( origen ,
peso ) ); //grafo no dirigido
                        }
                        re=infVeh.idSection;
                        inicial=secc2nodo[re];
                        //sprintf(astring,"INICIAL: %d ", inicial);
                        //AKIPrintString(astring);
                        dijkstra( inicial );

                        //destino=vehi.centroidDest;Aquí se tiene que
convertir el nodo de destino, en este caso 333 a 5o nodo de destino.
                        destino=centroidest[vehi.centroidDest];
                        //sprintf(astring,"DESTINO: %d ", destino);
                        //AKIPrintString(astring);
                        print( destino );
                        for (int gg = 0; gg < vecEjemplo.size(); gg ++) {
                              vehnodcmc[infVeh.idVeh][gg]=vecEjemplo[gg];

                              sprintf(astring,"%d \t",
vecEjemplo[gg]);//vecEjemplo me da los nodos que debo de cruzar para
llegar a mi destino mas corto
                              AKIPrintString(astring);
                        }

                  }

      ////////////////////////////////////////////////////////////////
/////////
                  //RUTINA PARA CONVERTIR DE NODOS A ARCOS

                  int ig=0;


                  if (( infveh.report >= 0)&&(bandijkstra==1)){

                        nodosec[108][105]=35939;
                        nodosec[105][81]=35946;
                        nodosec[81][80]=35949;
                        nodosec[81][75]=35950;
                        nodosec[76][75]=35954;
                        nodosec[75][17]=35955;
                        nodosec[17][16]=35963;
                        nodosec[17][12]=35967;
                        nodosec[14][12]=35969;
                        nodosec[12][5]=35974;
```

```
nodosec[5][4]=35977;
nodosec[5][3]=35978;
nodosec[108][103]=39290;
nodosec[103][104]=39310;
nodosec[103][99]=39311;
nodosec[99][100]=39315;
nodosec[99][98]=39316;
nodosec[104][96]=39320;
nodosec[98][104]=39321;
nodosec[106][111]=39329;
nodosec[106][107]=39332;
nodosec[107][111]=39342;
nodosec[112][94]=39375;
nodosec[92][91]=39381;
nodosec[92][93]=39382;
nodosec[95][107]=39386;
nodosec[101][97]=39387;
nodosec[96][98]=39388;
nodosec[98][100]=39389;
nodosec[97][80]=39400;
nodosec[80][76]=39401;
nodosec[76][36]=39430;
nodosec[16][14]=39441;
nodosec[14][7]=39444;
nodosec[9][15]=39451;
nodosec[13][74]=39452;
nodosec[13][15]=39458;
nodosec[15][18]=39462;
nodosec[36][16]=39469;
nodosec[35][37]=39472;
nodosec[74][82]=39476;
nodosec[74][73]=39479;
nodosec[73][77]=39483;
nodosec[79][83]=39487;
nodosec[79][82]=39497;
nodosec[6][4]=39509;
nodosec[4][3]=39510;
nodosec[252][8]=39514;
nodosec[252][13]=39536;
nodosec[9][7]=39548;
nodosec[8][221]=39565;
nodosec[6][8]=39568;
nodosec[35][36]=39649;
nodosec[36][35]=39650;
nodosec[82][106]=39681;
nodosec[94][96]=39689;
nodosec[97][93]=39690;
nodosec[91][113]=39692;
nodosec[222][221]=45328;
nodosec[114][86]=45354;
nodosec[114][85]=45355;
nodosec[86][87]=45356;
nodosec[112][95]=48127;
nodosec[113][90]=48143;
nodosec[84][92]=48150;
nodosec[85][86]=48163;
nodosec[113][84]=48169;
```

```
nodosec[88][89]=48179;
nodosec[87][191]=48188;
nodosec[122][121]=48617;
nodosec[122][118]=51600;
nodosec[123][125]=51603;
nodosec[138][145]=51609;
nodosec[123][122]=51630;
nodosec[119][123]=51632;
nodosec[232][233]=54443;
nodosec[87][88]=54446;
nodosec[90][114]=54487;
nodosec[89][90]=54499;
nodosec[232][58]=54509;
nodosec[77][71]=54511;
nodosec[77][78]=54515;
nodosec[78][79]=54525;
nodosec[226][207]=54531;
nodosec[232][236]=54532;
nodosec[222][223]=54541;
nodosec[225][222]=54548;
nodosec[225][223]=54551;
nodosec[224][226]=54565;
nodosec[224][222]=54566;
nodosec[222][226]=54575;
nodosec[100][102]=54634;
nodosec[137][142]=64131;
nodosec[89][112]=75694;
nodosec[138][140]=75762;
nodosec[137][138]=75771;
nodosec[138][137]=75782;
nodosec[137][136]=75788;
nodosec[119][127]=75794;
nodosec[116][182]=75921;
nodosec[250][204]=75925;
nodosec[250][160]=75928;
nodosec[250][248]=75934;
nodosec[188][187]=75943;
nodosec[188][167]=75947;
nodosec[150][147]=78727;
nodosec[116][88]=78774;
nodosec[116][190]=78777;
nodosec[118][119]=78813;
nodosec[84][85]=87536;
nodosec[126][123]=229224;
nodosec[194][195]=258545;
nodosec[228][229]=258546;
nodosec[194][176]=258555;
nodosec[146][133]=258589;
nodosec[164][158]=258590;
nodosec[144][143]=258591;
nodosec[146][144]=258600;
nodosec[200][201]=262165;
nodosec[202][203]=262166;
nodosec[201][224]=262167;
nodosec[201][250]=262176;
nodosec[202][201]=262187;
nodosec[67][68]=333144;
```

```
nodosec[67][78]=333146;
nodosec[67][66]=333152;
nodosec[66][59]=373556;
nodosec[83][84]=373576;
nodosec[191][192]=373590;
nodosec[178][179]=373591;
nodosec[191][179]=373592;
nodosec[179][180]=373601;
nodosec[118][116]=373615;
nodosec[223][201]=374043;
nodosec[253][251]=374079;
nodosec[237][230]=374080;
nodosec[236][253]=374081;
nodosec[230][231]=374090;
nodosec[253][194]=374101;
nodosec[160][161]=374121;
nodosec[160][137]=374125;
nodosec[160][162]=374128;
nodosec[197][239]=374143;
nodosec[227][240]=374144;
nodosec[227][228]=374145;
nodosec[66][61]=374158;
nodosec[27][31]=374159;
nodosec[58][57]=374169;
nodosec[52][45]=374171;
nodosec[58][67]=374175;
nodosec[58][246]=374178;
nodosec[147][132]=385166;
nodosec[147][146]=385167;
nodosec[157][165]=385176;
nodosec[158][159]=385186;
nodosec[158][134]=385187;
nodosec[151][130]=385200;
nodosec[129][127]=385202;
nodosec[127][151]=385203;
nodosec[151][150]=385215;
nodosec[18][20]=385223;
nodosec[18][19]=385226;
nodosec[23][22]=385235;
nodosec[28][29]=385236;
nodosec[28][35]=385237;
nodosec[50][255]=396853;
nodosec[35][34]=396867;
nodosec[39][38]=396868;
nodosec[37][72]=396869;
nodosec[28][34]=396880;
nodosec[34][33]=396881;
nodosec[229][230]=396900;
nodosec[221][220]=563780;
nodosec[219][220]=563781;
nodosec[219][209]=563782;
nodosec[221][9]=563792;
nodosec[221][222]=563798;
nodosec[144][145]=774691;
nodosec[145][148]=774692;
nodosec[148][147]=774704;
nodosec[148][149]=774707;
```

```
nodosec[228][210]=912218;
nodosec[243][244]=912225;
nodosec[189][177]=912227;
nodosec[177][193]=912229;
nodosec[244][245]=912230;
nodosec[245][192]=912231;
nodosec[251][237]=912261;
nodosec[237][236]=912262;
nodosec[236][235]=912263;
nodosec[247][235]=912264;
nodosec[235][234]=912265;
nodosec[234][233]=912267;
nodosec[207][208]=912270;
nodosec[208][220]=912271;
nodosec[209][208]=912272;
nodosec[209][227]=912273;
nodosec[220][219]=912279;
nodosec[207][227]=912282;
nodosec[220][221]=912289;
nodosec[210][209]=912298;
nodosec[210][217]=912299;
nodosec[217][52]=912300;
nodosec[211][229]=912307;
nodosec[213][212]=912310;
nodosec[212][211]=912311;
nodosec[211][210]=912312;
nodosec[215][214]=912313;
nodosec[45][44]=912314;
nodosec[45][46]=912320;
nodosec[47][51]=912326;
nodosec[47][45]=912329;
nodosec[33][32]=912330;
nodosec[29][30]=912331;
nodosec[27][23]=912332;
nodosec[27][26]=912333;
nodosec[45][27]=912334;
nodosec[19][21]=912335;
nodosec[21][26]=912336;
nodosec[26][46]=912337;
nodosec[71][72]=912338;
nodosec[40][39]=912339;
nodosec[31][43]=912340;
nodosec[44][53]=912341;
nodosec[246][55]=912342;
nodosec[42][54]=912343;
nodosec[54][216]=912344;
nodosec[216][211]=912345;
nodosec[216][215]=912346;
nodosec[219][218]=912347;
nodosec[254][255]=912348;
nodosec[51][254]=912349;
nodosec[254][218]=912350;
nodosec[218][217]=912351;
nodosec[217][216]=912352;
nodosec[32][41]=912354;
nodosec[34][38]=912355;
nodosec[38][63]=912356;
```

```
nodosec[63][64]=912357;
nodosec[64][69]=912358;
nodosec[63][62]=912359;
nodosec[65][64]=912360;
nodosec[69][68]=912361;
nodosec[69][70]=912362;
nodosec[39][33]=912363;
nodosec[33][29]=912364;
nodosec[29][23]=912365;
nodosec[23][21]=912366;
nodosec[21][20]=912367;
nodosec[26][24]=912368;
nodosec[62][61]=912369;
nodosec[61][60]=912370;
nodosec[60][57]=912371;
nodosec[59][60]=912372;
nodosec[55][56]=912375;
nodosec[231][212]=912376;
nodosec[212][215]=912377;
nodosec[49][47]=912378;
nodosec[46][48]=912379;
nodosec[195][251]=912380;
nodosec[175][195]=912381;
nodosec[168][175]=912382;
nodosec[174][196]=912384;
nodosec[198][173]=912385;
nodosec[205][199]=912386;
nodosec[206][207]=912387;
nodosec[171][172]=912388;
nodosec[169][174]=912389;
nodosec[200][206]=912390;
nodosec[199][200]=912391;
nodosec[203][200]=912392;
nodosec[172][203]=912393;
nodosec[204][172]=912394;
nodosec[172][173]=912395;
nodosec[198][199]=912396;
nodosec[238][237]=912397;
nodosec[196][197]=912398;
nodosec[177][178]=912399;
nodosec[192][178]=912400;
nodosec[178][181]=912401;
nodosec[181][182]=912402;
nodosec[176][177]=912403;
nodosec[192][193]=912404;
nodosec[193][247]=912405;
nodosec[245][247]=912406;
nodosec[247][234]=912407;
nodosec[234][241]=912408;
nodosec[241][244]=912409;
nodosec[193][194]=912410;
nodosec[180][181]=912411;
nodosec[182][183]=912413;
nodosec[183][190]=912414;
nodosec[183][184]=912415;
nodosec[186][189]=912416;
nodosec[130][153]=912417;
```

```
nodosec[186][183]=912418;
nodosec[129][128]=912419;
nodosec[165][169]=912420;
nodosec[156][166]=912421;
nodosec[175][176]=912422;
nodosec[174][175]=912423;
nodosec[173][174]=912425;
nodosec[173][170]=912426;
nodosec[131][130]=912427;
nodosec[132][131]=912428;
nodosec[150][149]=912429;
nodosec[152][151]=912430;
nodosec[159][163]=912437;
nodosec[163][171]=912438;
nodosec[170][164]=912439;
nodosec[185][184]=912440;
nodosec[184][153]=912441;
nodosec[187][186]=912442;
nodosec[153][154]=912443;
nodosec[155][156]=912444;
nodosec[154][155]=912450;
nodosec[156][157]=912451;
nodosec[127][126]=912454;
nodosec[162][163]=912455;
nodosec[163][164]=912456;
nodosec[164][165]=912457;
nodosec[165][166]=912458;
nodosec[143][135]=912459;
nodosec[135][159]=912460;
nodosec[136][135]=912461;
nodosec[135][134]=912462;
nodosec[134][133]=912463;
nodosec[133][132]=912464;
nodosec[62][39]=912465;
nodosec[71][70]=912468;
nodosec[72][73]=912473;
nodosec[70][67]=912477;
nodosec[38][37]=912482;
nodosec[68][65]=912485;
nodosec[65][62]=912488;
nodosec[19][22]=912520;
nodosec[20][24]=912539;
nodosec[25][26]=912543;
nodosec[26][27]=912544;
nodosec[24][25]=912545;
nodosec[25][48]=912548;
nodosec[61][40]=912551;
nodosec[61][66]=912554;
nodosec[40][32]=912569;
nodosec[40][61]=912572;
nodosec[32][40]=912580;
nodosec[32][30]=912583;
nodosec[57][41]=912592;
nodosec[41][43]=912596;
nodosec[42][56]=912601;
nodosec[30][27]=912607;
nodosec[31][32]=912610;
```

```
nodosec[43][45]=912635;
nodosec[59][58]=912655;
nodosec[56][58]=912659;
nodosec[55][54]=912663;
nodosec[54][53]=912664;
nodosec[53][52]=912665;
nodosec[52][51]=912666;
nodosec[51][50]=912667;
nodosec[44][42]=912668;
nodosec[48][49]=912674;
nodosec[49][50]=912677;
nodosec[246][214]=912682;
nodosec[214][213]=912708;
nodosec[213][232]=912712;
nodosec[255][219]=912715;
nodosec[239][238]=912766;
nodosec[240][239]=912767;
nodosec[205][240]=912768;
nodosec[206][205]=912769;
nodosec[231][232]=912770;
nodosec[195][196]=912798;
nodosec[143][137]=912815;
nodosec[134][144]=912819;
nodosec[157][158]=912825;
nodosec[240][198]=912833;
nodosec[239][227]=912836;
nodosec[197][198]=912845;
nodosec[196][238]=912853;
nodosec[238][228]=912854;
nodosec[233][242]=912857;
nodosec[242][243]=912861;
nodosec[242][241]=912865;
nodosec[243][87]=912868;
nodosec[189][188]=912882;
nodosec[176][188]=912890;
nodosec[204][202]=912917;
nodosec[171][250]=912921;
nodosec[170][171]=912950;
nodosec[168][169]=912956;
nodosec[167][168]=912960;
nodosec[169][170]=912972;
nodosec[166][167]=912978;
nodosec[154][131]=913000;
nodosec[182][189]=913014;
nodosec[8][9]=913028;
nodosec[180][116]=913036;
nodosec[190][185]=913041;
nodosec[162][250]=913050;
nodosec[159][160]=913054;
nodosec[184][129]=913060;
nodosec[128][119]=913070;
nodosec[185][128]=913074;
nodosec[130][129]=913081;
nodosec[136][160]=913092;
nodosec[133][157]=913101;
nodosec[132][155]=913107;
nodosec[149][152]=913115;
```

```
                        nodosec[152][126]=913121;
                        nodosec[131][150]=913125;
                        nodosec[153][186]=913131;
                        nodosec[187][154]=913134;
                        nodosec[7][6]=913141;
                        nodosec[7][11]=913145;
                        nodosec[10][6]=913146;
                        nodosec[3][2]=913157;
                        nodosec[96][97]=913170;
                        nodosec[93][94]=913181;
                        nodosec[94][95]=913182;
                        nodosec[22][28]=913219;
                        nodosec[1][252]=913233;
                        nodosec[115][88]=913238;
                        nodosec[117][116]=913243;
                        nodosec[120][122]=913271;
                        nodosec[124][122]=913277;
                        nodosec[109][108]=913283;
                        nodosec[110][105]=913286;
                        nodosec[139][138]=913297;
                        nodosec[141][138]=913305;
                        nodosec[249][225]=913321;



                        int fff;
                        fff=vecEjemplo.size(); //igualamamos
                        for(int q=0; q < (fff-1); q++){
                              int r1=vehnodcmc[infVeh.idVeh][q];
                              int r2=vehnodcmc[infVeh.idVeh][q+1];
                              int secc=nodosec[r1][r2];


                              if (infveh.idSection == secc)
                              {
                                    ig=1;
                                    continue;

                              }


                              if (ig == 1){
                                    AKIVehTrackedModifyNextSection
(infVeh.idVeh, secc);
                                    ig=0;

                              }
                        }
                  }

                  //TERMINA RUTINA PARA CONVERTIR DE NODOS A ARCOS

      ////////////////////////////////////////////////////////////////
/////////
```

```
                bandijkstra=1;

      ////////////////////////////////////////////////////////////////////
//////////
                vecEjemplo.clear();
                ady[origen].erase(ady[origen].begin(),
ady[origen].end());
            }//END FOR 2

      }// END FOR 1



      if (promediotv[35939]==0)
      {
            promediotv[35939]=8.67;
      }
      if (promediotv[35946]==0)
      {
            promediotv[35946]=16.06;
      }
      if (promediotv[35949]==0)
      {
            promediotv[35949]=2.88;
      }
      if (promediotv[35950]==0)
      {
            promediotv[35950]=17.93;
      }
      if (promediotv[35954]==0)
      {
            promediotv[35954]=4.15;
      }
      if (promediotv[35955]==0)
      {
            promediotv[35955]=17.74;
      }
      if (promediotv[35963]==0)
      {
            promediotv[35963]=2.65;
      }
      if (promediotv[35967]==0)
      {
            promediotv[35967]=11.65;
      }
      if (promediotv[35969]==0)
      {
            promediotv[35969]=4;
      }
      if (promediotv[35974]==0)
      {
            promediotv[35974]=9.09;
      }
      if (promediotv[35977]==0)
```

```
{
      promediotv[35977]=7.88;
}
if (promediotv[35978]==0)
{
      promediotv[35978]=7.77;
}
if (promediotv[39290]==0)
{
      promediotv[39290]=10.57;
}
if (promediotv[39310]==0)
{
      promediotv[39310]=6.18;
}
if (promediotv[39311]==0)
{
      promediotv[39311]=4.95;
}
if (promediotv[39315]==0)
{
      promediotv[39315]=10.43;
}
if (promediotv[39316]==0)
{
      promediotv[39316]=3.50;
}
if (promediotv[39320]==0)
{
      promediotv[39320]=4.91;
}
if (promediotv[39321]==0)
{
      promediotv[39321]=7;
}
if (promediotv[39329]==0)
{
      promediotv[39329]=14.14;
}
if (promediotv[39332]==0)
{
      promediotv[39332]=4;
}
if (promediotv[39342]==0)
{
      promediotv[39342]=23.27;
}
if (promediotv[39375]==0)
{
      promediotv[39375]=2.56;
}
if (promediotv[39381]==0)
{
      promediotv[39381]=5;
}
if (promediotv[39382]==0)
{
```

```
      promediotv[39382]=2.67;
}
if (promediotv[39386]==0)
{
      promediotv[39386]=17.23;
}
if (promediotv[39387]==0)
{
      promediotv[39387]=42.06;
}
if (promediotv[39388]==0)
{
      promediotv[39388]=5.49;
}
if (promediotv[39389]==0)
{
      promediotv[39389]=3.07;
}
if (promediotv[39400]==0)
{
      promediotv[39400]=25.66;
}
if (promediotv[39401]==0)
{
      promediotv[39401]=28.15;
}
if (promediotv[39430]==0)
{
      promediotv[39430]=19.08;
}
if (promediotv[39441]==0)
{
      promediotv[39441]=13.12;
}
if (promediotv[39444]==0)
{
      promediotv[39444]=18.04;
}
if (promediotv[39451]==0)
{
      promediotv[39451]=17.40;
}
if (promediotv[39452]==0)
{
      promediotv[39452]=25.75;
}
if (promediotv[39458]==0)
{
      promediotv[39458]=3;
}
if (promediotv[39462]==0)
{
      promediotv[39462]=16.69;
}
if (promediotv[39469]==0)
{
      promediotv[39469]=18;
```

```
}
if (promediotv[39472]==0)
{
      promediotv[39472]=3.37;
}
if (promediotv[39476]==0)
{
      promediotv[39476]=17.43;
}
if (promediotv[39479]==0)
{
      promediotv[39479]=7.12;
}
if (promediotv[39483]==0)
{
      promediotv[39483]=6.43;
}
if (promediotv[39487]==0)
{
      promediotv[39487]=4.78;
}
if (promediotv[39497]==0)
{
      promediotv[39497]=6.89;
}
if (promediotv[39509]==0)
{
      promediotv[39509]=14.52;
}
if (promediotv[39510]==0)
{
      promediotv[39510]=8.69;
}
if (promediotv[39514]==0)
{
      promediotv[39514]=25.51;
}
if (promediotv[39536]==0)
{
      promediotv[39536]=20.29;
}
if (promediotv[39548]==0)
{
      promediotv[39548]=8.90;
}
if (promediotv[39565]==0)
{
      promediotv[39565]=28.49;
}
if (promediotv[39568]==0)
{
      promediotv[39568]=4.55;
}
if (promediotv[39649]==0)
{
      promediotv[39649]=1;
}
```

```
if (promediotv[39650]==0)
{
      promediotv[39650]=5.30;
}
if (promediotv[39681]==0)
{
      promediotv[39681]=15.67;
}
if (promediotv[39689]==0)
{
      promediotv[39689]=5.02;
}
if (promediotv[39690]==0)
{
      promediotv[39690]=4.22;
}
if (promediotv[39692]==0)
{
      promediotv[39692]=2.79;
}
if (promediotv[45328]==0)
{
      promediotv[45328]=16.72;
}
if (promediotv[45354]==0)
{
      promediotv[45354]=41;
}
if (promediotv[45355]==0)
{
      promediotv[45355]=5.70;
}
if (promediotv[45356]==0)
{
      promediotv[45356]=8.91;
}
if (promediotv[48127]==0)
{
      promediotv[48127]=1.40;
}
if (promediotv[48143]==0)
{
      promediotv[48143]=3.24;
}
if (promediotv[48150]==0)
{
      promediotv[48150]=10.34;
}
if (promediotv[48163]==0)
{
      promediotv[48163]=35.42;
}
if (promediotv[48169]==0)
{
      promediotv[48169]=21;
}
if (promediotv[48179]==0)
```

```
{
      promediotv[48179]=57.86;
}
if (promediotv[48188]==0)
{
      promediotv[48188]=28.24;
}
if (promediotv[48617]==0)
{
      promediotv[48617]=8.29;
}
if (promediotv[51600]==0)
{
      promediotv[51600]=33.49;
}
if (promediotv[51603]==0)
{
      promediotv[51603]=3.07;
}
if (promediotv[51609]==0)
{
      promediotv[51609]=20.54;
}
if (promediotv[51630]==0)
{
      promediotv[51630]=37.63;
}
if (promediotv[51632]==0)
{
      promediotv[51632]=14.29;
}
if (promediotv[54443]==0)
{
      promediotv[54443]=8.68;
}
if (promediotv[54446]==0)
{
      promediotv[54446]=15.36;
}
if (promediotv[54487]==0)
{
      promediotv[54487]=1.70;
}
if (promediotv[54499]==0)
{
      promediotv[54499]=9.91;
}
if (promediotv[54509]==0)
{
      promediotv[54509]=32.19;
}
if (promediotv[54511]==0)
{
      promediotv[54511]=3;
}
if (promediotv[54515]==0)
{
```

```
      promediotv[54515]=1.50;
}
if (promediotv[54525]==0)
{
      promediotv[54525]=15.38;
}
if (promediotv[54531]==0)
{
      promediotv[54531]=7.10;
}
if (promediotv[54532]==0)
{
      promediotv[54532]=14.13;
}
if (promediotv[54541]==0)
{
      promediotv[54541]=1.50;
}
if (promediotv[54548]==0)
{
      promediotv[54548]=60.16;
}
if (promediotv[54551]==0)
{
      promediotv[54551]=22.22;
}
if (promediotv[54565]==0)
{
      promediotv[54565]=2.86;
}
if (promediotv[54566]==0)
{
      promediotv[54566]=1;
}
if (promediotv[54575]==0)
{
      promediotv[54575]=2.80;
}
if (promediotv[54634]==0)
{
      promediotv[54634]=29.74;
}
if (promediotv[64131]==0)
{
      promediotv[64131]=12.64;
}
if (promediotv[75694]==0)
{
      promediotv[75694]=8.22;
}
if (promediotv[75762]==0)
{
      promediotv[75762]=7.81;
}
if (promediotv[75771]==0)
{
      promediotv[75771]=7.65;
```

```
}
if (promediotv[75782]==0)
{
      promediotv[75782]=6.44;
}
if (promediotv[75788]==0)
{
      promediotv[75788]=12.85;
}
if (promediotv[75794]==0)
{
      promediotv[75794]=12.07;
}
if (promediotv[75921]==0)
{
      promediotv[75921]=10.99;
}
if (promediotv[75925]==0)
{
      promediotv[75925]=11.69;
}
if (promediotv[75928]==0)
{
      promediotv[75928]=21.02;
}
if (promediotv[75934]==0)
{
      promediotv[75934]=8.63;
}
if (promediotv[75943]==0)
{
      promediotv[75943]=11.37;
}
if (promediotv[75947]==0)
{
      promediotv[75947]=5.64;
}
if (promediotv[78727]==0)
{
      promediotv[78727]=9.25;
}
if (promediotv[78774]==0)
{
      promediotv[78774]=56.16;
}
if (promediotv[78777]==0)
{
      promediotv[78777]=9.71;
}
if (promediotv[78813]==0)
{
      promediotv[78813]=25.95;
}
if (promediotv[87536]==0)
{
      promediotv[87536]=13.34;
}
```

```
if (promediotv[229224]==0)
{
      promediotv[229224]=20.39;
}
if (promediotv[258545]==0)
{
      promediotv[258545]=9.11;
}
if (promediotv[258546]==0)
{
      promediotv[258546]=6.49;
}
if (promediotv[258555]==0)
{
      promediotv[258555]=9.94;
}
if (promediotv[258589]==0)
{
      promediotv[258589]=1;
}
if (promediotv[258590]==0)
{
      promediotv[258590]=9.30;
}
if (promediotv[258591]==0)
{
      promediotv[258591]=9.04;
}
if (promediotv[258600]==0)
{
      promediotv[258600]=10.53;
}
if (promediotv[262165]==0)
{
      promediotv[262165]=45.55;
}
if (promediotv[262166]==0)
{
      promediotv[262166]=9;
}
if (promediotv[262167]==0)
{
      promediotv[262167]=17.52;
}
if (promediotv[262176]==0)
{
      promediotv[262176]=23.59;
}
if (promediotv[262187]==0)
{
      promediotv[262187]=6.12;
}
if (promediotv[333144]==0)
{
      promediotv[333144]=4;
}
if (promediotv[333146]==0)
```

```
{
      promediotv[333146]=13.84;
}
if (promediotv[333152]==0)
{
      promediotv[333152]=5;
}
if (promediotv[373556]==0)
{
      promediotv[373556]=3.05;
}
if (promediotv[373576]==0)
{
      promediotv[373576]=8.12;
}
if (promediotv[373590]==0)
{
      promediotv[373590]=17.54;
}
if (promediotv[373591]==0)
{
      promediotv[373591]=25.61;
}
if (promediotv[373592]==0)
{
      promediotv[373592]=10.73;
}
if (promediotv[373601]==0)
{
      promediotv[373601]=8.03;
}
if (promediotv[373615]==0)
{
      promediotv[373615]=43.46;
}
if (promediotv[374043]==0)
{
      promediotv[374043]=18.89;
}
if (promediotv[374079]==0)
{
      promediotv[374079]=9;
}
if (promediotv[374080]==0)
{
      promediotv[374080]=11;
}
if (promediotv[374081]==0)
{
      promediotv[374081]=5.95;
}
if (promediotv[374090]==0)
{
      promediotv[374090]=1;
}
if (promediotv[374101]==0)
{
```

```
      promediotv[374101]=5;
}
if (promediotv[374121]==0)
{
      promediotv[374121]=9.33;
}
if (promediotv[374125]==0)
{
      promediotv[374125]=27.98;
}
if (promediotv[374128]==0)
{
      promediotv[374128]=98.66;
}
if (promediotv[374143]==0)
{
      promediotv[374143]=12;
}
if (promediotv[374144]==0)
{
      promediotv[374144]=11.02;
}
if (promediotv[374145]==0)
{
      promediotv[374145]=9.20;
}
if (promediotv[374158]==0)
{
      promediotv[374158]=8;
}
if (promediotv[374159]==0)
{
      promediotv[374159]=3;
}
if (promediotv[374169]==0)
{
      promediotv[374169]=5;
}
if (promediotv[374171]==0)
{
      promediotv[374171]=11.40;
}
if (promediotv[374175]==0)
{
      promediotv[374175]=17.75;
}
if (promediotv[374178]==0)
{
      promediotv[374178]=6.78;
}
if (promediotv[385166]==0)
{
      promediotv[385166]=12.06;
}
if (promediotv[385167]==0)
{
      promediotv[385167]=6.86;
```

```
}
if (promediotv[385176]==0)
{
      promediotv[385176]=8;
}
if (promediotv[385186]==0)
{
      promediotv[385186]=40.74;
}
if (promediotv[385187]==0)
{
      promediotv[385187]=11.28;
}
if (promediotv[385200]==0)
{
      promediotv[385200]=12.07;
}
if (promediotv[385202]==0)
{
      promediotv[385202]=11;
}
if (promediotv[385203]==0)
{
      promediotv[385203]=14.50;
}
if (promediotv[385215]==0)
{
      promediotv[385215]=3.71;
}
if (promediotv[385223]==0)
{
      promediotv[385223]=2;
}
if (promediotv[385226]==0)
{
      promediotv[385226]=1.13;
}
if (promediotv[385235]==0)
{
      promediotv[385235]=7;
}
if (promediotv[385236]==0)
{
      promediotv[385236]=8;
}
if (promediotv[385237]==0)
{
      promediotv[385237]=3.58;
}
if (promediotv[396853]==0)
{
      promediotv[396853]=2;
}
if (promediotv[396867]==0)
{
      promediotv[396867]=1.79;
}
```

```
if (promediotv[396868]==0)
{
      promediotv[396868]=8;
}
if (promediotv[396869]==0)
{
      promediotv[396869]=5.54;
}
if (promediotv[396880]==0)
{
      promediotv[396880]=2;
}
if (promediotv[396881]==0)
{
      promediotv[396881]=8.76;
}
if (promediotv[396900]==0)
{
      promediotv[396900]=2.56;
}
if (promediotv[563780]==0)
{
      promediotv[563780]=9.06;
}
if (promediotv[563781]==0)
{
      promediotv[563781]=7;
}
if (promediotv[563782]==0)
{
      promediotv[563782]=6.05;
}
if (promediotv[563792]==0)
{
      promediotv[563792]=28.55;
}
if (promediotv[563798]==0)
{
      promediotv[563798]=17.34;
}
if (promediotv[774691]==0)
{
      promediotv[774691]=5.25;
}
if (promediotv[774692]==0)
{
      promediotv[774692]=17.48;
}
if (promediotv[774704]==0)
{
      promediotv[774704]=4;
}
if (promediotv[774707]==0)
{
      promediotv[774707]=7.09;
}
if (promediotv[912218]==0)
```

```
{
      promediotv[912218]=1;
}
if (promediotv[912225]==0)
{
      promediotv[912225]=14.12;
}
if (promediotv[912227]==0)
{
      promediotv[912227]=14.83;
}
if (promediotv[912229]==0)
{
      promediotv[912229]=10.09;
}
if (promediotv[912230]==0)
{
      promediotv[912230]=4.22;
}
if (promediotv[912231]==0)
{
      promediotv[912231]=8.26;
}
if (promediotv[912261]==0)
{
      promediotv[912261]=5;
}
if (promediotv[912262]==0)
{
      promediotv[912262]=8;
}
if (promediotv[912263]==0)
{
      promediotv[912263]=5.19;
}
if (promediotv[912264]==0)
{
      promediotv[912264]=7;
}
if (promediotv[912265]==0)
{
      promediotv[912265]=3.61;
}
if (promediotv[912267]==0)
{
      promediotv[912267]=15.24;
}
if (promediotv[912270]==0)
{
      promediotv[912270]=10.69;
}
if (promediotv[912271]==0)
{
      promediotv[912271]=10.50;
}
if (promediotv[912272]==0)
{
```

```
      promediotv[912272]=7;
}
if (promediotv[912273]==0)
{
      promediotv[912273]=11.35;
}
if (promediotv[912279]==0)
{
      promediotv[912279]=8.77;
}
if (promediotv[912282]==0)
{
      promediotv[912282]=8.89;
}
if (promediotv[912289]==0)
{
      promediotv[912289]=19.69;
}
if (promediotv[912298]==0)
{
      promediotv[912298]=1;
}
if (promediotv[912299]==0)
{
      promediotv[912299]=4;
}
if (promediotv[912300]==0)
{
      promediotv[912300]=7.88;
}
if (promediotv[912307]==0)
{
      promediotv[912307]=10.78;
}
if (promediotv[912310]==0)
{
      promediotv[912310]=4;
}
if (promediotv[912311]==0)
{
      promediotv[912311]=4;
}
if (promediotv[912312]==0)
{
      promediotv[912312]=6;
}
if (promediotv[912313]==0)
{
      promediotv[912313]=6.95;
}
if (promediotv[912314]==0)
{
      promediotv[912314]=5.29;
}
if (promediotv[912320]==0)
{
      promediotv[912320]=3;
```

```
}
if (promediotv[912326]==0)
{
      promediotv[912326]=9;
}
if (promediotv[912329]==0)
{
      promediotv[912329]=3;
}
if (promediotv[912330]==0)
{
      promediotv[912330]=10.71;
}
if (promediotv[912331]==0)
{
      promediotv[912331]=8;
}
if (promediotv[912332]==0)
{
      promediotv[912332]=8;
}
if (promediotv[912333]==0)
{
      promediotv[912333]=3;
}
if (promediotv[912334]==0)
{
      promediotv[912334]=8;
}
if (promediotv[912335]==0)
{
      promediotv[912335]=4;
}
if (promediotv[912336]==0)
{
      promediotv[912336]=8;
}
if (promediotv[912337]==0)
{
      promediotv[912337]=8;
}
if (promediotv[912338]==0)
{
      promediotv[912338]=14;
}
if (promediotv[912339]==0)
{
      promediotv[912339]=8;
}
if (promediotv[912340]==0)
{
      promediotv[912340]=8;
}
if (promediotv[912341]==0)
{
      promediotv[912341]=8;
}
```

```
if (promediotv[912342]==0)
{
      promediotv[912342]=5;
}
if (promediotv[912343]==0)
{
      promediotv[912343]=8;
}
if (promediotv[912344]==0)
{
      promediotv[912344]=6;
}
if (promediotv[912345]==0)
{
      promediotv[912345]=5.41;
}
if (promediotv[912346]==0)
{
      promediotv[912346]=4.07;
}
if (promediotv[912347]==0)
{
      promediotv[912347]=4.85;
}
if (promediotv[912348]==0)
{
      promediotv[912348]=4;
}
if (promediotv[912349]==0)
{
      promediotv[912349]=2;
}
if (promediotv[912350]==0)
{
      promediotv[912350]=2;
}
if (promediotv[912351]==0)
{
      promediotv[912351]=4.18;
}
if (promediotv[912352]==0)
{
      promediotv[912352]=7.76;
}
if (promediotv[912354]==0)
{
      promediotv[912354]=8;
}
if (promediotv[912355]==0)
{
      promediotv[912355]=2;
}
if (promediotv[912356]==0)
{
      promediotv[912356]=1;
}
if (promediotv[912357]==0)
```

```
{
      promediotv[912357]=4;
}
if (promediotv[912358]==0)
{
      promediotv[912358]=3;
}
if (promediotv[912359]==0)
{
      promediotv[912359]=8;
}
if (promediotv[912360]==0)
{
      promediotv[912360]=7;
}
if (promediotv[912361]==0)
{
      promediotv[912361]=5;
}
if (promediotv[912362]==0)
{
      promediotv[912362]=7;
}
if (promediotv[912363]==0)
{
      promediotv[912363]=2;
}
if (promediotv[912364]==0)
{
      promediotv[912364]=2;
}
if (promediotv[912365]==0)
{
      promediotv[912365]=3;
}
if (promediotv[912366]==0)
{
      promediotv[912366]=3;
}
if (promediotv[912367]==0)
{
      promediotv[912367]=2;
}
if (promediotv[912368]==0)
{
      promediotv[912368]=2;
}
if (promediotv[912369]==0)
{
      promediotv[912369]=8;
}
if (promediotv[912370]==0)
{
      promediotv[912370]=1;
}
if (promediotv[912371]==0)
{
```

```
      promediotv[912371]=5;
}
if (promediotv[912372]==0)
{
      promediotv[912372]=6;
}
if (promediotv[912375]==0)
{
      promediotv[912375]=8;
}
if (promediotv[912376]==0)
{
      promediotv[912376]=6;
}
if (promediotv[912377]==0)
{
      promediotv[912377]=3;
}
if (promediotv[912378]==0)
{
      promediotv[912378]=3;
}
if (promediotv[912379]==0)
{
      promediotv[912379]=3;
}
if (promediotv[912380]==0)
{
      promediotv[912380]=4;
}
if (promediotv[912381]==0)
{
      promediotv[912381]=10;
}
if (promediotv[912382]==0)
{
      promediotv[912382]=13;
}
if (promediotv[912384]==0)
{
      promediotv[912384]=10;
}
if (promediotv[912385]==0)
{
      promediotv[912385]=10.66;
}
if (promediotv[912386]==0)
{
      promediotv[912386]=9;
}
if (promediotv[912387]==0)
{
      promediotv[912387]=10;
}
if (promediotv[912388]==0)
{
      promediotv[912388]=12;
```

```
}
if (promediotv[912389]==0)
{
      promediotv[912389]=13;
}
if (promediotv[912390]==0)
{
      promediotv[912390]=24;
}
if (promediotv[912391]==0)
{
      promediotv[912391]=4.95;
}
if (promediotv[912392]==0)
{
      promediotv[912392]=4;
}
if (promediotv[912393]==0)
{
      promediotv[912393]=5;
}
if (promediotv[912394]==0)
{
      promediotv[912394]=9;
}
if (promediotv[912395]==0)
{
      promediotv[912395]=9;
}
if (promediotv[912396]==0)
{
      promediotv[912396]=4.04;
}
if (promediotv[912397]==0)
{
      promediotv[912397]=9;
}
if (promediotv[912398]==0)
{
      promediotv[912398]=1.32;
}
if (promediotv[912399]==0)
{
      promediotv[912399]=11;
}
if (promediotv[912400]==0)
{
      promediotv[912400]=11.32;
}
if (promediotv[912401]==0)
{
      promediotv[912401]=8;
}
if (promediotv[912402]==0)
{
      promediotv[912402]=9.00;
}
```

```
if (promediotv[912403]==0)
{
      promediotv[912403]=6;
}
if (promediotv[912404]==0)
{
      promediotv[912404]=8.96;
}
if (promediotv[912405]==0)
{
      promediotv[912405]=8.11;
}
if (promediotv[912406]==0)
{
      promediotv[912406]=7;
}
if (promediotv[912407]==0)
{
      promediotv[912407]=3.41;
}
if (promediotv[912408]==0)
{
      promediotv[912408]=2.82;
}
if (promediotv[912409]==0)
{
      promediotv[912409]=3.59;
}
if (promediotv[912410]==0)
{
      promediotv[912410]=6.70;
}
if (promediotv[912411]==0)
{
      promediotv[912411]=12.13;
}
if (promediotv[912413]==0)
{
      promediotv[912413]=10.35;
}
if (promediotv[912414]==0)
{
      promediotv[912414]=13;
}
if (promediotv[912415]==0)
{
      promediotv[912415]=12.16;
}
if (promediotv[912416]==0)
{
      promediotv[912416]=11.42;
}
if (promediotv[912417]==0)
{
      promediotv[912417]=8.22;
}
if (promediotv[912418]==0)
```

```
{
      promediotv[912418]=14;
}
if (promediotv[912419]==0)
{
      promediotv[912419]=11;
}
if (promediotv[912420]==0)
{
      promediotv[912420]=10;
}
if (promediotv[912421]==0)
{
      promediotv[912421]=8;
}
if (promediotv[912422]==0)
{
      promediotv[912422]=9;
}
if (promediotv[912423]==0)
{
      promediotv[912423]=7;
}
if (promediotv[912425]==0)
{
      promediotv[912425]=9;
}
if (promediotv[912426]==0)
{
      promediotv[912426]=14.56;
}
if (promediotv[912427]==0)
{
      promediotv[912427]=4;
}
if (promediotv[912428]==0)
{
      promediotv[912428]=8;
}
if (promediotv[912429]==0)
{
      promediotv[912429]=6.43;
}
if (promediotv[912430]==0)
{
      promediotv[912430]=8.15;
}
if (promediotv[912437]==0)
{
      promediotv[912437]=8;
}
if (promediotv[912438]==0)
{
      promediotv[912438]=15.64;
}
if (promediotv[912439]==0)
{
```

```
      promediotv[912439]=10.09;
}
if (promediotv[912440]==0)
{
      promediotv[912440]=20.75;
}
if (promediotv[912441]==0)
{
      promediotv[912441]=15.31;
}
if (promediotv[912442]==0)
{
      promediotv[912442]=5;
}
if (promediotv[912443]==0)
{
      promediotv[912443]=7.58;
}
if (promediotv[912444]==0)
{
      promediotv[912444]=2.49;
}
if (promediotv[912450]==0)
{
      promediotv[912450]=8;
}
if (promediotv[912451]==0)
{
      promediotv[912451]=5.43;
}
if (promediotv[912454]==0)
{
      promediotv[912454]=5;
}
if (promediotv[912455]==0)
{
      promediotv[912455]=14.36;
}
if (promediotv[912456]==0)
{
      promediotv[912456]=9;
}
if (promediotv[912457]==0)
{
      promediotv[912457]=9;
}
if (promediotv[912458]==0)
{
      promediotv[912458]=6;
}
if (promediotv[912459]==0)
{
      promediotv[912459]=12;
}
if (promediotv[912460]==0)
{
      promediotv[912460]=12;
```

```
}
if (promediotv[912461]==0)
{
      promediotv[912461]=9;
}
if (promediotv[912462]==0)
{
      promediotv[912462]=9;
}
if (promediotv[912463]==0)
{
      promediotv[912463]=9;
}
if (promediotv[912464]==0)
{
      promediotv[912464]=6;
}
if (promediotv[912465]==0)
{
      promediotv[912465]=2;
}
if (promediotv[912468]==0)
{
      promediotv[912468]=2;
}
if (promediotv[912473]==0)
{
      promediotv[912473]=7.43;
}
if (promediotv[912477]==0)
{
      promediotv[912477]=5;
}
if (promediotv[912482]==0)
{
      promediotv[912482]=2;
}
if (promediotv[912485]==0)
{
      promediotv[912485]=3;
}
if (promediotv[912488]==0)
{
      promediotv[912488]=3;
}
if (promediotv[912520]==0)
{
      promediotv[912520]=4.63;
}
if (promediotv[912539]==0)
{
      promediotv[912539]=9;
}
if (promediotv[912543]==0)
{
      promediotv[912543]=2;
}
```

```
if (promediotv[912544]==0)
{
      promediotv[912544]=3;
}
if (promediotv[912545]==0)
{
      promediotv[912545]=0;
}
if (promediotv[912548]==0)
{
      promediotv[912548]=8;
}
if (promediotv[912551]==0)
{
      promediotv[912551]=3;
}
if (promediotv[912554]==0)
{
      promediotv[912554]=10.28;
}
if (promediotv[912569]==0)
{
      promediotv[912569]=2;
}
if (promediotv[912572]==0)
{
      promediotv[912572]=3.19;
}
if (promediotv[912580]==0)
{
      promediotv[912580]=3.05;
}
if (promediotv[912583]==0)
{
      promediotv[912583]=2;
}
if (promediotv[912592]==0)
{
      promediotv[912592]=6;
}
if (promediotv[912596]==0)
{
      promediotv[912596]=2;
}
if (promediotv[912601]==0)
{
      promediotv[912601]=3.71;
}
if (promediotv[912607]==0)
{
      promediotv[912607]=3;
}
if (promediotv[912610]==0)
{
      promediotv[912610]=2;
}
if (promediotv[912635]==0)
```

```
{
      promediotv[912635]=3;
}
if (promediotv[912655]==0)
{
      promediotv[912655]=9.78;
}
if (promediotv[912659]==0)
{
      promediotv[912659]=13.99;
}
if (promediotv[912663]==0)
{
      promediotv[912663]=3;
}
if (promediotv[912664]==0)
{
      promediotv[912664]=2;
}
if (promediotv[912665]==0)
{
      promediotv[912665]=3;
}
if (promediotv[912666]==0)
{
      promediotv[912666]=3;
}
if (promediotv[912667]==0)
{
      promediotv[912667]=4;
}
if (promediotv[912668]==0)
{
      promediotv[912668]=2.66;
}
if (promediotv[912674]==0)
{
      promediotv[912674]=0;
}
if (promediotv[912677]==0)
{
      promediotv[912677]=9;
}
if (promediotv[912682]==0)
{
      promediotv[912682]=7.00;
}
if (promediotv[912708]==0)
{
      promediotv[912708]=3.31;
}
if (promediotv[912712]==0)
{
      promediotv[912712]=10.43;
}
if (promediotv[912715]==0)
{
```

```
      promediotv[912715]=2;
}
if (promediotv[912766]==0)
{
      promediotv[912766]=8;
}
if (promediotv[912767]==0)
{
      promediotv[912767]=0;
}
if (promediotv[912768]==0)
{
      promediotv[912768]=1;
}
if (promediotv[912769]==0)
{
      promediotv[912769]=12;
}
if (promediotv[912770]==0)
{
      promediotv[912770]=6.28;
}
if (promediotv[912798]==0)
{
      promediotv[912798]=9.18;
}
if (promediotv[912815]==0)
{
      promediotv[912815]=13.33;
}
if (promediotv[912819]==0)
{
      promediotv[912819]=13.23;
}
if (promediotv[912825]==0)
{
      promediotv[912825]=9.81;
}
if (promediotv[912833]==0)
{
      promediotv[912833]=10.58;
}
if (promediotv[912836]==0)
{
      promediotv[912836]=10;
}
if (promediotv[912845]==0)
{
      promediotv[912845]=6.44;
}
if (promediotv[912853]==0)
{
      promediotv[912853]=9;
}
if (promediotv[912854]==0)
{
      promediotv[912854]=11;
```

```
}
if (promediotv[912857]==0)
{
      promediotv[912857]=2.18;
}
if (promediotv[912861]==0)
{
      promediotv[912861]=2.88;
}
if (promediotv[912865]==0)
{
      promediotv[912865]=12;
}
if (promediotv[912868]==0)
{
      promediotv[912868]=42.58;
}
if (promediotv[912882]==0)
{
      promediotv[912882]=6.95;
}
if (promediotv[912890]==0)
{
      promediotv[912890]=28.04;
}
if (promediotv[912917]==0)
{
      promediotv[912917]=4.28;
}
if (promediotv[912921]==0)
{
      promediotv[912921]=12.60;
}
if (promediotv[912950]==0)
{
      promediotv[912950]=9.70;
}
if (promediotv[912956]==0)
{
      promediotv[912956]=6.34;
}
if (promediotv[912960]==0)
{
      promediotv[912960]=2.68;
}
if (promediotv[912972]==0)
{
      promediotv[912972]=9.64;
}
if (promediotv[912978]==0)
{
      promediotv[912978]=10;
}
if (promediotv[913000]==0)
{
      promediotv[913000]=9.76;
}
```

```
if (promediotv[913014]==0)
{
      promediotv[913014]=13.84;
}
if (promediotv[913028]==0)
{
      promediotv[913028]=2.74;
}
if (promediotv[913036]==0)
{
      promediotv[913036]=11.01;
}
if (promediotv[913041]==0)
{
      promediotv[913041]=23.98;
}
if (promediotv[913050]==0)
{
      promediotv[913050]=326.56;
}
if (promediotv[913054]==0)
{
      promediotv[913054]=95.94;
}
if (promediotv[913060]==0)
{
      promediotv[913060]=7;
}
if (promediotv[913070]==0)
{
      promediotv[913070]=92.26;
}
if (promediotv[913074]==0)
{
      promediotv[913074]=29.61;
}
if (promediotv[913081]==0)
{
      promediotv[913081]=14;
}
if (promediotv[913092]==0)
{
      promediotv[913092]=25.45;
}
if (promediotv[913101]==0)
{
      promediotv[913101]=1;
}
if (promediotv[913107]==0)
{
      promediotv[913107]=12.04;
}
if (promediotv[913115]==0)
{
      promediotv[913115]=4.19;
}
if (promediotv[913121]==0)
```

```
{
      promediotv[913121]=13.29;
}
if (promediotv[913125]==0)
{
      promediotv[913125]=17.64;
}
if (promediotv[913131]==0)
{
      promediotv[913131]=9.38;
}
if (promediotv[913134]==0)
{
      promediotv[913134]=9.30;
}
if (promediotv[913141]==0)
{
      promediotv[913141]=3.38;
}
if (promediotv[913145]==0)
{
      promediotv[913145]=3.22;
}
if (promediotv[913146]==0)
{
      promediotv[913146]=5.86;
}
if (promediotv[913157]==0)
{
      promediotv[913157]=2.36;
}
if (promediotv[913170]==0)
{
      promediotv[913170]=3.34;
}
if (promediotv[913181]==0)
{
      promediotv[913181]=2.95;
}
if (promediotv[913182]==0)
{
      promediotv[913182]=1.64;
}
if (promediotv[913219]==0)
{
      promediotv[913219]=3.91;
}
if (promediotv[913233]==0)
{
      promediotv[913233]=3;
}
if (promediotv[913238]==0)
{
      promediotv[913238]=13.76;
}
if (promediotv[913243]==0)
{
```

```
            promediotv[913243]=14.01;
      }
      if (promediotv[913271]==0)
      {
            promediotv[913271]=35.09;
      }
      if (promediotv[913277]==0)
      {
            promediotv[913277]=66.72;
      }
      if (promediotv[913283]==0)
      {
            promediotv[913283]=21.31;
      }
      if (promediotv[913286]==0)
      {
            promediotv[913286]=15.15;
      }
      if (promediotv[913297]==0)
      {
            promediotv[913297]=12.62;
      }
      if (promediotv[913305]==0)
      {
            promediotv[913305]=29.53;
      }
      if (promediotv[913321]==0)
      {
            promediotv[913321]=93.12;
      }



      //TERMINA ALGORITMO DIJKSTRA

      for (i=1;i<=12000;i++)
      {
            //if((i % 10) == 0){
            //continue;
            //}
            AKIVehSetAsTracked (i);

            //AKIVehSetAsTracked (i+1);
            //AKIVehSetAsTracked (i+2);
            //AKIVehSetAsTracked (i+3);// INSTRUMENTAMOS EL 10% DE LOS
VEHICULOS QUE CIRCULARAN EN EL SISTEMA
      }


      tiempo=time;

      return 0;
}

int AAPIFinish()
{
      return 0;
```

```
}

int AAPIUnLoad()
{
	return 0;
}
```