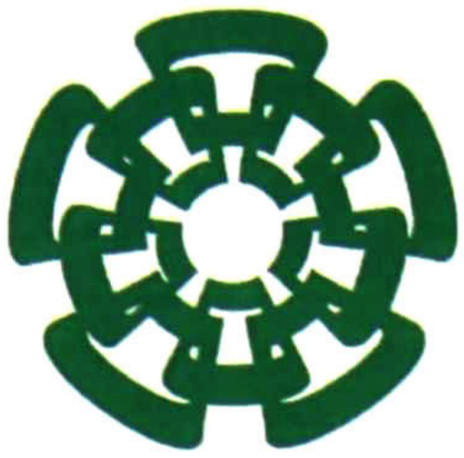
BC-665
Don. 2012

xx(182696.1)



Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional
Unidad Guadalajara

Control Adaptativo de Tráfico Urbano Mediante Programación Cuadrática

Tesis que presenta:

Sandra Mercado Pérez

para obtener el grado de:

Maestro en Ciencias

en la especialidad de:

Ingeniería Eléctrica

Directores de Tesis

Dr. Luis Ernesto López Mellado
Dr. Antonio Ramírez Treviño

CINVESTAV del IPN Unidad Guadalajara, Guadalajara, Jalisco, Junio de 2011.

JD-181203-1001

Control Adaptativo de Tráfico Urbano Mediante Programación Cuadrática

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:

Sandra Mercado Pérez

Ingeniero en Sistemas Computacionales

Instituto Tecnológico de Zacatecas 1997-2002

Directores de Tesis

Dr. Luis Ernesto López Mellado

Dr. Antonio Ramírez Treviño

CINVESTAV del IPN Unidad Guadalajara, Junio de 2011

Agradecimientos:

A Dios, por regalarme la vida, la salud y el tiempo para concluir este trabajo.

A mi familia, por todos los buenos deseos que han tenido para mí y por su apoyo incondicional a lo largo de toda mi vida.

A mi esposo, por su comprensión y apoyo.

A mis profesores, por compartir conmigo sus conocimientos y en especial a mis asesores, Dr. Ernesto López y Dr. Antonio Ramírez por su guía y apoyo.

A mis compañeros del CINVESTAV, porque además de compartir conmigo sus conocimientos y experiencias, cada uno de ustedes a través de su ejemplo me motivó a tratar de ser una mejor persona. Gracias por dejar en mi mente tan gratos y divertidos recuerdos.

Al CONACYT por su apoyo.

Control Adaptativo de Tráfico Urbano Mediante Programación Cuadrática

Resumen

En esta tesis se presenta un método de control adaptativo de tráfico urbano basado en programación cuadrática. Para cada intersección se resuelve un problema de optimización con la finalidad de obtener las variaciones de tiempo de fases de los semáforos que permitan nivelar la densidad de tráfico en las calles de entrada; dicha optimización se basa en medidas arrojadas por sensores establecidos en sus segmentos de calle de entrada y salida y está sujeta a restricciones donde se establece que cada intersección no debe afectar a las intersecciones vecinas, disminuyendo así la necesidad de negociación.

El método propuesto fue probado en el simulador de tráfico urbano SUMO, donde se implementó el control de tráfico en una red de caminos con seis intersecciones bajo distintos escenarios de prueba.

Adaptative Control of Urban Traffic Using Cuadratic Programming

Abstract

In this thesis an adaptative urban traffic control method via quadratic programming is presented. For every intersection a optimization problem is solved with the purpose of getting the time variations in phases of traffic lights that allows to level the traffic density in the input streets; such optimization is based in measurements generated by sensors installed on the input and output segment streets and is limited by constraints in which is settled that every intersection should not affect to the adjacent intersections, decreasing the need to negotiate.

The method proposed was probed in the urban traffic simulator SUMO, in which a traffic control of a net of six intersections is implemented under different tests scenarios.

Índice

Introducción

En los pasados 40 años el mundo ha presenciado la adopción del automóvil como medio primario de transporte en las principales ciudades de todos los países. Si se asocia este factor al aumento de la densidad poblacional que ha ocurrido durante las últimas décadas, se observa que las redes de tráfico del presente no son suficientes para manejar eficientemente la necesidad diaria de movimiento de tráfico.

Las adecuaciones que se realizan en los caminos urbanos generalmente están restringidas por las estructuras urbanas existentes; por lo que realizar cambios en estos caminos implica la inversión de grandes cantidades de recursos económicos. En consecuencia, estas adecuaciones sólo se realizan en los casos de mayor necesidad y con muy poca frecuencia, como son los puentes y pasos a desnivel vehiculares.

La mejora del flujo vehicular puede lograrse de varias maneras: incrementando la infraestructura vial, mejorando el servicio de transporte público e incentivando el uso de éste, o bien implementado nuevas estrategias de control de semáforos. Esta última solución parece la menos costosa pues la inversión requerida se enfoca a complementar la infraestructura de señalización ya existente [Hewage, 2004].

Aunque un sistema de semáforos programado adecuadamente es la solución más barata posible en cuanto al descongestionamiento de tráfico en las ciudades, se debe tener en cuenta que en algunos casos las modificaciones a la estructura de caminos son absolutamente necesarias para mejorar cl tráfico y evadir situaciones de crisis causadas por los embotellamientos.[Swida, 2006]

En el presente trabajo, el objetivo planteado ha sido proponer un esquema de optimización de tiempos en semáforos, donde los tiempos de verde se adapten dinámicamente de acuerdo a la situación de tráfico actual con el objetivo de mejorar la circulación de vehículos en áreas urbanas.

La optimización de tiempos para los semáforos es un problema complejo, ya que el estado de un semáforo afecta el flujo de tráfico hacia muchas otras intersecciones. Si el flujo vehicular enviado por una intersección (crucero) es mayor al flujo vehicular que pueden soportar las calles de las intersecciones vecinas, entonces se generan problemas de saturación. Por esta

razón es importante encontrar un esquema de coordinación efectivo donde se logre que los cambios en los tiempos de verde en una intersección no causen problemas en las intersecciones adyacentes [Herena, 2007].

En esta tesis se propone un esquema de control de tiempos en semáforos basado en la resolución de un sistema de ecuaciones mediante programación cuadrática, en donde las restricciones tomadas en cuenta por cada intersección impliquen no afectar a las intersecciones vecinas, las cuales a su vez respetan restricciones similares, evitando así la necesidad de negociación entre intersecciones o el uso de un esquema centralizado. Esto que implica costos aún menores de implementación, ya que se elimina la necesidad de comunicación entre intersecciones.

La presentación del trabajo desarrollado en esta tesis está organizado como sigue:

En el capítulo 1 se presentan las nociones y definiciones acerca del problema de control de tráfico urbano, así como las técnicas de control que han sido desarrolladas y los principales simuladores de tráfico existentes.

En el capítulo 2 se propone una estrategia de control de tiempos en semáforos en donde se explican las restricciones y la función objetivo a optimizar.

En el capítulo 3 se describe como fue implementado el sistema de control propuesto en el simulador SUMO y se analizan algunas ventajas y desventajas observadas durante la implementación de control en semáforos.

En el capítulo 4 se presentan los casos de prueba realizados, en donde se muestra que el sistema de control propuesto es capaz de adaptarse a las diferentes demandas de tráfico con el objetivo de mejorarlo.

Al final se tiene un apéndice en donde se incluye el código de algunas aplicaciones desarrolladas en lenguaje C como complemento del simulador de tráfico, con el objetivo de lograr capturar los parámetros deseados e implementar el cambio de tiempos en semáforos.

Capítulo 1

Sistemas de control de tráfico urbano

Resumen: En este capítulo se presentan los conceptos básicos y definiciones acerca del problema de control de tráfico urbano; se presentan también algunas de las técnicas de control que han sido desarrolladas y los principales simuladores de tráfico existentes.

1.1. Necesidad del control de tráfico

El rápido aumento de la población en las zonas urbanas y la adopción del automóvil como medio primario de transporte tienen como consecuencia que las redes de tráfico existentes sean insuficientes para satisfacer los movimientos diarios de tráfico. La consecuencia de esto son tiempos de viaje más largos, contaminación, estrés en conductores e incremento de costos en transporte de bienes.

Se puede pensar que la solución a este problema es agregar más carriles a los caminos urbanos ya existentes, pero esta es una solución poco factible ya que debido a la urbanización no existirá el espacio libre para agregar uno o más carriles a la derecha o izquierda de los ya existentes. Una solución alterna que lleve al mismo resultado es incrementar el número de carriles en niveles inferiores o superiores de los caminos más transitados, pero por factores económicos esto no se puede hacer con todas las calles que presente una congestión de tráfico.

Esto nos motiva a manejar la capacidad existente de las calles de una manera más inteligente.

1.2. Estudio del control de tráfico urbano

Desde hace varios años se investiga cómo mejorar el flujo de tráfico urbano sin intervenir la estructura física de los caminos de manera radical; muchas son las estrategias que se han propuesto, en todas se asume la existencia de uno o varios factores como son sensores de tráfico, comunicación entre semáforos, paneles con avisos del estado de tráfico y comunicación por radiofrecuencia con otros vehículos, entre otros.

A continuación se menciona la definición de algunas de las áreas de estudio que investigan sobre cómo mejorar el flujo de tráfico urbano.

Los **Sistemas de Transporte Inteligente** (ITS) es un área de estudio que incluye la aplicación de tecnologías de información para modelar, analizar y controlar sistemas de transporte y tráfico. [Di Febbraro, 2006]

En [Slinn, 2005] se cita la definición de **Ingeniería de Tráfico** como:

"Esa parte de la ingeniería que trata con la planeación de tráfico y el diseño de caminos, de desarrollo de andadores y de lugares de estacionamiento y con el control de tráfico para proveer movimiento de vehículos y peatones seguro, conveniente y económico."

Esta definición permanece válida al presente, pero claramente ha habido un cambio en el rol de la ingeniería de tráfico desde el tiempo que este libro fue producido (donde se menciona originalmente la definición). A los ingenieros de tráfico se les asignó la tarea de incrementar la capacidad del sistema de caminos para abastecer lo que parecía un crecimiento sin fin de tráfico de motores, a expensas de otros usuarios del camino [Slinn, 2005].

La base de la mayoría de los **sistemas de control de tráfico urbano** es la tarea de distribuir tanto el tiempo como el espacio físico disponible; para optimizar el movimiento de vehículos de una manera coordinada en una intersección se extiende de forma natural a la inclusión de intersecciones adyacentes [García, 2000].

Los caminos de tráfico urbano cuentan con semáforos en sus intersecciones con el propósito de mejorar la seguridad y la eficiencia en el flujo de vehículos y facilitar el paso a los peatones. La implementación de control de tráfico mediante semáforos nos ofrece la posibilidad de agilizar el tráfico sin tener que modificar la infraestructura existente; esto mediante la modificación de duraciones de ciclo de un semáforo y la duración de verde relativa de cada fase, así como el offset entre fases respectivas.

Optimizar los tiempos de luces en los semáforos nos lleva a reducir factores negativos en el tráfico urbano, como son los tiempos de espera de los conductores y la contaminación en el ambiente debido a una reducción en el tiempo de viaje de los vehículos.

Un sistema eficiente de control de tráfico debe ser adaptable; la dificultad está dada por los factores no predecibles que intervienen en el sistema, tales como la inteligencia humana, el clima, accidentes, reparación de calles, etc. Se puede considerar que un sistema de control de tráfico es más eficiente si para una densidad dada de autos se logra incrementar la velocidad promedio de los vehículos [Roozemond, 2000].

Los sistemas de control de tráfico operan para alcanzar uno o varios de los siguientes objetivos:

- Reducir los cuellos de botella
- Distribuir la densidad de vehículos en la zona controlada
- Minimizar la ocupación proporcional de los segmentos de calle
- Minimizar colas de espera
- Minimizar el tiempo de viaje de los vehículos
- Evadir congestiones
- Maximizar la tasa de flujo
- Minimizar tiempos de espera en colas
- Reducir número de paradas

1.3. **Conceptos básicos**

Para un mejor entendimiento, a continuación se describen los términos utilizados comúnmente en el tema de control de tráfico adoptando el enfoque de modificación de tiempos en semáforos:

Intersección: Área de Cruce común entre un conjunto de calles.

Fase: Combinación de luces en la cual algunos vehículos tienen permiso de cruzar la intersección mientras otros vehículos esperan.

Tiempo de Verde: Lapso de tiempo asignado a cada fase.

Ciclo: Lapso de tiempo en el cual todas las calles que cruzan la intersección obtienen un tiempo de verde dado por una fase.

Offset: Diferencia de tiempo en que inicia el ciclo de una intersección con relación al inicio del ciclo de la intersección de la cual recibe el flujo de vehículos. El offset puede ser ajustado para permitir que varios semáforos cooperen y crear así grupos de vehículos que crucen varias intersecciones sin necesidad de detenerse.

Pelotón: Grupo de autos que viajan a gran velocidad, con poca distancia entre ellos.

Sobresaturación: Situación que ocurre cuando vehículos llenan completamente la capacidad de la calle.

1.4. **Técnicas recientes de control de tráfico**

Se puede agrupar las técnicas actuales de control de tráfico mediante modificación de tiempos de fases en dos categorías generales, estrategias de tiempo fijo y estrategias de respuesta al tráfico.

1.4.1. **Estrategias de tiempo fijo**

En este tipo de estrategias las fases de cada semáforo tienen una duración de tiempo previamente asignada y estos tiempos son aplicados siempre. Este tipo de estrategia se caracteriza por un uso ineficiente de la capacidad de la calle, debido a que no se puede ajustar a las variaciones del flujo de tráfico [Herena, 2007].

Dentro de las estrategias de tiempo fijo podemos encontrar una variante, la cual consiste en aplicar tiempos fijos durante ciertas horas del día decididos previamente, de acuerdo a datos históricos de flujo de vehículos en horarios específicos. Existen algunos sistemas ya

desarrollados de este tipo. Los cuales son alimentados por datos históricos del tráfico, a continuación se mencionan algunos:

- TRANSYT es un programa que se ejecuta fuera de línea para determinar y estudiar los tiempos fijos óptimos para las señales de tráfico urbano, en cualquier red de tráfico donde los flujos de tráfico promedio son conocidos; su uso alrededor del mundo refleja lo apropiado y flexible del producto como una herramienta de optimización de red y de diseño [TRANSYT, 2010].

  MAXBAND es un programa que se ejecuta fuera de línea con el objetivo de maximizar el ancho de banda, el programa produce tiempos de ciclo, offsets, velocidades, división en tiempos de verde y fases con vuelta a la izquierda para maximizar la combinación de anchos de banda, la optimización usa el algoritmo de ramificación acotado MPCODE [Little, 1981]. Posteriormente este programa tuvo algunas extensiones que se aplicaron en el programa MAXBAND-86, además de hacer uso concurrente con el sistema TRANSYT.

- MULTIBAND usa programación lineal entera mixta para la optimización. Este programa proporciona la capacidad de adaptar el esquema de progresión a un patrón de tráfico específico en cada liga de la arteria [Gartner, 1990].

1.4.2. **Estrategias de respuesta al tráfico**

Este tipo de técnicas operan de acuerdo a la demanda de tráfico registrada a través de detectores de vehículos. Los tiempos de ciclo y de fases se ajustan de acuerdo al estado actual de tráfico y en ocasiones también se toma en cuenta otros parámetros como el día, la hora o el clima.

Algunos sistemas de control de tráfico de este tipo se utilizan ya en algunas ciudades del mundo, en donde se han obtenido mejoras en el tráfico urbano. Los enfoques que se han utilizado para este tipo de estrategias son: sistemas multiagentes, algoritmos genéticos, redes neuronales, lógica difusa, auto-organización, predicción y redes de Petri entre otros.

En [Mizuno, 2008] se propone un sistema multi-agente y un problema de satisfacción de restricciones (CSP) se extiende para hacerlo distribuido, en donde cada agente asignado a una intersección determina dinámicamente los parámetros de sus propias señales mientras recibe información de congestión de los agentes conectados a él. Las restricciones de cada agente se definen sobre sus variables locales y las de otros agentes. El CSP se define para fijar la proporción de tiempos de verde, tiempo de ciclo y offset.

En [Zand, 2007] se ha propuesto un enfoque de tres niveles, en donde se tiene un agente para el control de una intersección, agentes para el control de una zona y agentes móviles, cuya meta es mantener el nivel de tráfico de ciertas intersecciones; estos agentes son los que decidirán entre las diferentes estrategias de control de acuerdo con sus observaciones.

1.5. **Sistemas de control de tráfico en operación**

A continuación se mencionan algunos de los sistemas de control de respuesta al tráfico desarrollados actualmente:

SCATS (Sydney Coordinated Adaptive Traffic System). Sus objetivos fundamentales son la reducción del número de paradas debido a una mala regularización de las señales de tráfico de los semáforos en las intersecciones, los retrasos provocados por el flujo de tráfico denso y la optimización del tiempo necesario para realizar una serie de rutas origen-destino previamente establecidas.

SCATS es un sistema distribuido jerárquico en tres niveles. La arquitectura del sistema se compone de una computadora central para realizar la monitorización del sistema global, computadoras regionales, remotas o locales y controladores locales de señales de tráfico. Cada computadora regional puede gestionar hasta 128 controladores locales y su principal función consiste en controlar de forma autónoma las intersecciones de su área mediante el análisis de la información de los datos de los detectores por medio de los microprocesadores locales [García, 2000].

A la fecha, el sistema SCATS ha sido distribuido a 144 ciudades para el control de más de 33,000 intersecciones, entre las más relevantes están Sydney, Melbourne y Hong Kong [SCATS, 2010].

SCOOT (Split, Cycle & Offset Optimization Technique, Técnica de Optimización de los valores de Desplazamiento, Ciclo y Giro) es una herramienta para administrar y controlar las señales de tráfico en áreas urbanas. Se fundamenta en la experiencia aportada en la definición, desarrollo e implementación de TRANSYT. SCOOT es un sistema adaptable que responde automáticamente a los cambios en el flujo de tráfico a través del uso de detectores instalados en la calle. El envío de las instrucciones de control para los dispositivos lo hace a través de líneas telefónicas dedicadas. Ha sido implementado en más de 200 ciudades, incluyendo Pekin, Londres, Chicago, Sao Paulo, Toronto, entre otras. SCOOT opera en cuatro áreas clave: comunicaciones, control de congestión, prioridad a autobuses y servicios a peatones [Breherton, 2004].

TUC (Traffic Resposive Urban Control) El objetivo de control de TUC es la minimización y balance del número de vehículos en las calles de la red a controlar. Se asumen

tiempos de ciclo y offsets, se manipulan los tiempos de verde. Para esta estrategia se ha desarrollado una ley de control a través de la aplicación de la metodología cuadrática-lineal (LQ) al problema de control óptimo formulado. La aplicación de esta ley requiere disponibilidad de valores nominales de tiempos de verde. En caso de que estos valores no estén disponibles, se puede aplicar la metodología lineal cuadrática-integral (LQI).

Esta estrategia se ha probado bajo simulación de condiciones de tráfico real, la primera aplicación consistió de 13 intersecciones en la red de tráfico de la ciudad de Glasgow, y la segunda aplicación consistió de 17 intersecciones de la ciudad de Chania [Dinopoulou, 2002].

A principios de los 80's se crearon algunas estrategias llamadas de tercera generación, como OPAC (Optimization Policies for Adaptive Control) y PRODYN. Estas estrategias son operativas acíclicamente, es decir, no consideran offsets o ciclos dados. La optimización se basa en un criterio de retraso determinado por modelos simples. Para el desempeño de la optimización global OPAC emplea enumeración completa, mientas PRODYN usa programación dinámica. Debido a la complejidad exponencial de la solución para más de una intersección las implementaciones en tiempo real solo son factibles para intersecciones aisladas [Friedrich, 2002].

OPAC se ha mantenido en desarrollo, generando nuevas versiones de control adaptativo; se ha implementado en intersecciones aisladas en Arlington, Virginia y Tucson, Arizona, además de implementarse en arterias y redes en New Jersey [Gartner, 2001].

1.6. **Conclusiones**

Los sistemas de control de tráfico urbano constituyen un campo científico con actividades de investigación y desarrollo extensas. De los trabajos revisados se observa que para poder implementar el control a lo largo de alguna avenida o un área extensa de la red, las soluciones propuestas utilizan un sistema de jerarquías; se observó también que algunos de los trabajos utilizan optimización para calcular los tiempos de verde a aplicar.

El presente trabajo se relaciona con las tres áreas de estudio mencionadas en la sección 1.2, ya que se enfocará en optimizar el movimiento de vehículos modificando la distribución de tiempo durante el cual un grupo de vehículos tiene derecho a cruzar una calle. Se busca también distribuir la densidad de vehículos y minimizar la ocupación proporcional de de los segmentos ligados a una intersección.

Capítulo 2

Estrategia de control basada en programación cuadrática

Resumen: En este capítulo se propone una estrategia para determinar los tiempos de verde en semáforos basada en programación cuadrática. Se presentan las restricciones a establecer y la función objetivo a optimizar.

2.1. **Enfoque**

A continuación se mencionan algunas de las causas del problema de congestionamiento en el tráfico urbano:

- Se presenta sobrecarga de densidad de tráfico de manera desequilibrada en las calles, por lo que durante el tiempo de rojo de cada fase, en algunas ocasiones se forma cola hasta del 100% de la capacidad de la calle, mientras que en otras fases la cola no satura la capacidad de la calle, lo que a veces implica tiempo de verde desperdiciado.

- La sobrecarga mencionada cambia a lo largo del día, de acuerdo a horas pico, tipos de edificios y diversos eventos.

- Los semáforos permiten el paso de autos aún cuando en los segmentos de calle siguientes no hay espacios libres.

En este trabajo se obtienen parámetros como la densidad vehicular en calles, el flujo de vehículos y la cantidad de espacios libres. En base a estos parámetros se busca obtener tiempos de verde que contribuyan a nivelar la densidad vehicular y el tamaño de colas de espera de vehículos en las calles de entrada de una intersección. Por ello se diseñó una estrategia de control de tráfico, que mediante programación cuadrática, obtenga los tiempos de verde óptimos para cada intersección y se tomen en cuenta restricciones orientadas a reducir el efecto de las acciones de control en las intersecciones adyacentes, evitando así la negociación entre intersecciones vecinas.

Para el sistema de control propuesto se consideró un tiempo de ciclo fijo de manera predeterminada, de manera que lo único que cambie dinámicamente sea la proporción de tiempos en que se divide el ciclo para cada fase. En la figura 2.1 se muestra un ejemplo del cambio en la proporción de tiempos asignados a cada fase respecto al tiempo de ciclo.

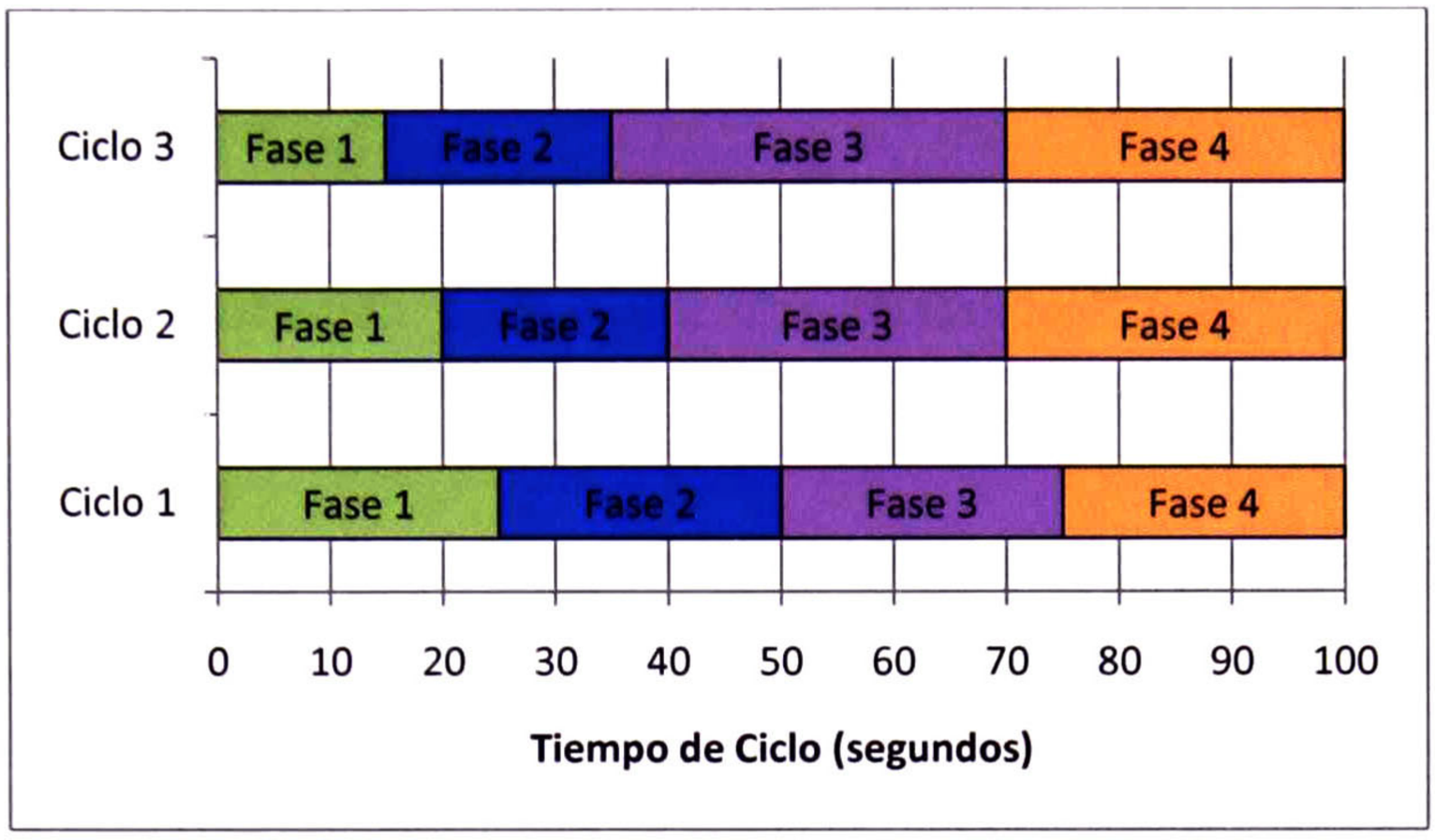


Figura 2. 1 Representación de diferentes tiempos de fase para distintos ciclos

2.2. **Solución propuesta**

2.2.1. **Función Objetivo**

Como se mencionó anteriormente, se busca obtener tiempos de verde que contribuyan a nivelar la densidad vehicular en las calles de entrada de una intersección, por lo que se establece una función objetivo que proporcione aumento de tiempo de verde para las fases en donde las calles tienen mayor densidad vehicular y disminuya tiempo de verde a las fases cuya densidad vehicular sea menor respecto a las otras fases de la intersección.

Los parámetros utilizados en la función objetivo son los siguientes:

F	Conjunto de fases. $\{f_1,f_2,f_3,\ldots.f_n\}$
e_j	Número de coches que pueden dejar la intersección durante la fase j en un segundo.
d_sup_j	Diferencia entre la densidad en la calle correspondiente a la fase j y el promedio de las densidades de todas las fases del ciclo. En caso de que esta diferencia sea negativa se asigna cero. Se toma la densidad media de la calle correspondiente a cada fase durante el ciclo.
d_inf_j	Diferencia entre el promedio de las densidades de todas las fases del ciclo y la densidad en la calle correspondiente a la fase j. En caso de que esta diferencia sea negativa de asigna cero. Se toma la densidad media de la calle correspondiente a cada fase durante la duración del ciclo.

En el caso de las pruebas realizadas en este trabajo, el simulador proporciona directamente la medida de densidad vehicular en cada segmento de calle, pero no proporciona

d_sup y *d_inf*, por lo que estos parámetros se calculan de manera externa con los datos proporcionados por el simulador.

El número de coches que pueden dejar la intersección en un segundo (e_j) se obtuvo midiendo el parámetro en simulación, en el caso de las pruebas realizadas en este trabajo, se tiene un valor de *0.96 veh/seg*. para una calle con dos carriles.

Las variables a calcular son:

t_j	Cantidad de tiempo que se agrega a la fase j. (en segundos)
u_j	Cantidad de tiempo que se disminuye de la fase j. (en segundos)

La función objetivo es la siguiente:

$$z = \sum_{j \in F} (d\_sup_j - t_j e_j)^2 + \sum_{j \in F} (d\_inf_j - u_j e_j)^2$$

Como se observa, es una función objetivo cuadrática, en donde se toma en cuenta dos elementos:

- La diferencia entre el exceso de densidad y los vehículos que pueden ser desalojados debido al aumento en el tiempo de cada fase.
- La diferencia entre la densidad sobrante y los vehículos que dejarían de pasar debido a la disminución del tiempo de verde de cada fase.

La minimización de esta función objetivo tiende a distribuir las densidades, es decir, aumentar tiempo a las fases más densas y quitar tiempo a las fases menos densas, de manera que la sumatoria de estas diferencias sea lo menor posible.

2.2.2. **Restricciones**

Para las restricciones tenemos los siguientes parámetros:

F	Conjunto de fases. $\{f_1, f_2, f_3, \ldots f_n\}$
TC	Tiempo de ciclo.
ta_j	Tiempo actual de verde de la fase j.
e_j	Número de coches que pueden dejar la intersección durante la fase j en un segundo.

esp_sig_j	Espacios disponibles en la(s) calle(s) hacia donde se dirige el flujo de la fase j. Se mide el número medio de espacios libres durante la duración de la fase j.
t_max	Máxima cantidad de tiempo que puede aumentar o disminuir una fase de un ciclo a otro. Expresada en número entero.
lim_max_fase	Porcentaje máximo del tiempo de ciclo permitido para una fase, expresado en reales positivos, [0-1].
lim_min_fase	Porcentaje mínimo del tiempo de ciclo permitido para una fase, expresado en reales positivos, [0-1].
d_sup_j	Diferencia entre la densidad en la calle correspondiente a la fase j y el promedio de las densidades de todas las fases del ciclo. En caso de que esta diferencia sea negativa se asigna cero. Se toma la densidad media de la calle correspondiente a cada fase durante el ciclo.
d_inf_j	Diferencia entre el promedio de las densidades de todas las fases del ciclo y la densidad en la calle correspondiente a la fase j. En caso de que esta diferencia sea negativa de asigna cero. Se toma la densidad media de la calle correspondiente a cada fase durante la duración del ciclo.
d_max	El valor mayor del conjunto $\{ d\_sup_j : j \in F \}$
d_min	El valor mayor del conjunto $\{ d\_inf_j : j \in F \}$

El tiempo de ciclo (TC) es un valor constante preestablecido. Se reserva para investigación futura el cálculo del valor ideal de este parámetro. Para las pruebas realizadas en este trabajo se consideró un tiempo de ciclo de 100 segundos.

El tiempo actual de verde para cada una de las fases del ciclo n lo proporciona el simulador al sistema de control para que calcule los tiempos de fase a aplicar en el ciclo n+1.

Como se mencionó anteriormente, el número de coches que pueden dejar la intersección en un segundo (e_j) se obtuvo midiendo el parámetro en simulación, en el caso de las pruebas realizadas en este trabajo, se tiene un valor de 0.96 veh/seg para una calle con dos carriles.

La cantidad de espacios disponibles en las calles hacia donde se dirige el flujo de una fase no la proporciona el simulador directamente, sino se tiene que calcular mediante una aplicación externa en base a otras medidas obtenidas del simulador.

La máxima cantidad de tiempo que puede aumentar o disminuir una fase de un ciclo a otro es un valor preestablecido para evitar cambios drásticos en el tiempo de alguna fase de un ciclo a otro.

El porcentaje de tiempo de ciclo máximo se establece para evitar que una de las fases monopolice el tiempo de ciclo aún cuando las otras fases posean densidades de tráfico mínimas.

El porcentaje de tiempo de ciclo mínimo se establece para evitar que fases que no posean flujo de vehículos lleguen a tener un tiempo de ciclo nulo.

Los valores d_sup y d_inf se calculan mediante una aplicación externa en base a otras medidas obtenidas del simulador.

- **Tiempo de ciclo fijo**

Como se mencionó, en este trabajo se considera el tiempo de ciclo fijo, por lo que la primera restricción puede expresarse:

$$\sum_{j \in F} (ta_j + t_j - u_j) = TC$$

Lo cual significa que la suma de tiempos de las fases aun con el incremento o decremento de tiempo calculado tiene que ser igual al valor constante del tiempo de ciclo.

Como ejemplo, en la figura 2.1 se observa que el tiempo para los ciclos 1, 2 y 3 es de 100 segundos, mientras que los tiempos de las fases 1, 2, 3 y 4 son diferentes en cada uno de los ciclos.

- **Suavidad en modificación de tiempos de verde**

En situaciones donde se dé un cambio drástico en las densidades de tráfico que origine una carga excesiva en pocas calles de una intersección, como se observa en la figura 2.2; el controlador tenderá a asignar un aumento de tiempo grande en la fase con carga excesiva. Por ejemplo, tomando en cuenta un tiempo de ciclo de 100 segundos, si la duración de la fase *x* en el ciclo *n* es de 25 seg. y se presenta un cambio drástico, entonces la duración de la fase *x* en el ciclo *n+1* podría ser hasta de 95 segs.

Figura 2. 2 Ejemplo de carga excesiva en una calle de entrada

Si el cambio drástico mencionado fuera ocasionado por un evento momentáneo y al siguiente ciclo las densidades en calles volvieran al estado anterior, entonces estarían ocurriendo cambios significativos en tiempos de fase de manera innecesaria.

Con el fin de evitar los cambios drásticos de tiempos de verde a las fases se propone acotar de manera preestablecida el tiempo máximo que una fase pueda aumentar de un ciclo a otro. Por ejemplo, si a este tiempo le damos un valor de 30 seg. y se retoma el caso antes mencionado, la duración de la fase *x* en el ciclo *n+1* sería de 75 segundos.

Las siguientes dos restricciones indican que el tiempo que se aumente o disminuya de alguna fase no debe ser mayor que un límite establecido.

$$t_j \leq t\_max \qquad u_j \leq t\_max \qquad \forall j \in F$$

En simulación se observó que para valores de *t_max* muy grandes se obtiene una mejor situación de tráfico en menor tiempo, aunque con oscilaciones en la desviación estándar de las densidades de tráfico; esto en contraste con valores de *t_max* menores, en donde se logra un resultado similar que se va alcanzando de manera gradual.

Por simulación se eligió 20 segundos como valor para t_max; se reserva como trabajo futuro el cálculo del valor ideal para este parámetro.

- **Cotas en el tiempo de verde**

Como complemento de la restricción anterior, se define una cota máxima para los tiempos de verde, con el fin de evitar que una fase monopolize el tiempo de ciclo; esto se establece mediante un valor predeterminado que indique el porcentaje máximo de tiempo de ciclo que una fase puede llegar a poseer.

Con la siguiente restricción se establece que cualquier fase, aún con el tiempo que aumente o disminuya, no deberá exceder de un porcentaje máximo establecido del tiempo de ciclo.

$$ta_j + t_j - u_j \leq TC(lim\_\max\ \_fase) \qquad \forall j \in F$$

En la estrategia de control propuesta no se mencionan de manera explícita los tiempos de amarillo y de cambio de fase, ya que estos tiempos se consideran incluídos en el tiempo de verde de la fase. Por lo tanto en los tiempos calculados mediante este sistema de control no debe obtenerse un tiempo de verde igual a cero, por lo cual se establece un valor mínimo para la duración de una fase en base a porcentaje de ciclo.

Con la siguiente restricción se establece que el tiempo de cada una de las fases debe ser mayor que un porcentaje mínimo establecido del tiempo de ciclo.

$$ta_j + t_j - u_j \geq TC(lim\_\min\ \_fase) \qquad \forall j \in F$$

- **Interacción con intersecciones vecinas**

Con el sistema de control propuesto se pretende que cada intersección no afecte a sus intersecciones vecinas; por lo cual la cantidad autos extra que son liberados de una intersección a causa del incremento de tiempo de alguna fase, no debe ser mayor que los espacios libres existentes en las calles hacia donde se dirige el flujo en esa fase. En el ejemplo de la figura 2.3 se muestran los espacios libres representados con un rectángulo. Suponiendo que la fase 1 necesita un incremento de 10 segundos y que pueden salir dos vehículos de la intersección por segundo, se tiene la liberación de 20 vehículos, sin embargo la fase solo recibiría un incremento de 9 segundos, ya que solo existen 18 espacios libres en las calles siguientes.

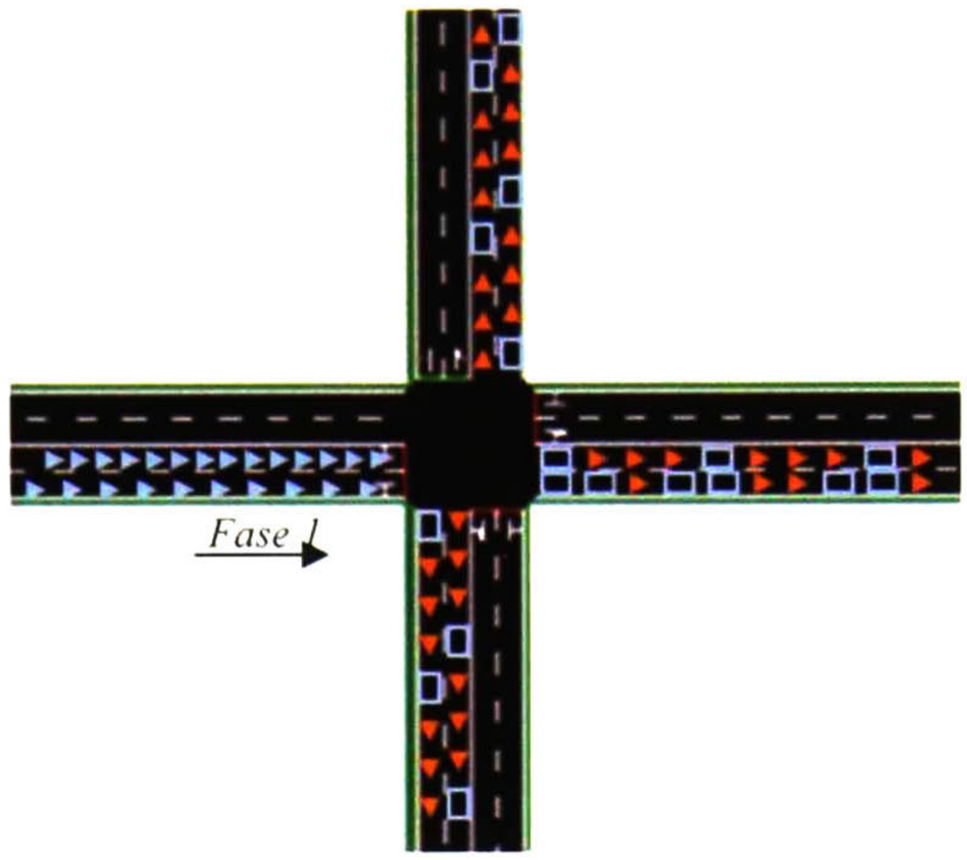


Figura 2. 3 Ejemplo de espacios disponibles para una fase

La siguiente restricción especifica que el número de autos que dejan la intersección no debe exceder el número de espacios libres de la calle(s) siguientes(s).

$$t_j e_j \leq esp\_sig_j \qquad \forall j \in F$$

- **Balanceo de densidades**

Con la estrategia propuesta se intenta además reducir la diferencia de las densidades de tráfico en las calles, por lo que se calcula la variación de la densidad de cada calle con respecto al promedio de las densidades de todas las calles del área; si la densidad de la calle correspondiente a la fase j está por encima del promedio, entonces la cantidad extra de vehículos que esta calle va a liberar a causa del incremento de tiempo en la fase, debería ser como máximo el excedente que se tiene en densidad con respecto al promedio.

Con la siguiente restricción se evita que el número de coches extra que pueden dejar la intersección debido al aumento de tiempo de verde, sea mayor que los vehículos extra que tiene la intersección, de acuerdo al exceso de densidad.

$$t_j e_j \leq d\_sup_j \qquad \forall j \in F$$

Si la densidad de la calle correspondiente a la fase j está por abajo del promedio, entonces la cantidad de vehículos que dejan de salir de ésta a causa del decremento de tiempo en la fase, debería ser como máximo el excedente que se tiene en densidad con respecto al promedio.

Con la siguiente restricción se evita que el número de coches que dejan de salir de la intersección debido a la disminución de tiempo de verde sea mayor que la densidad faltante para alcanzar el promedio.

$$u_j e_j \leq d\_inf_j \qquad \forall j \in F$$

- **Proporcionalidad del incremento de tiempos**

La siguiente restricción permite establecer el tiempo a aumentar de cada fase de manera proporcional de acuerdo a los valores de d_sup_j, es decir, la fase cuya d_sup_j sea mayor podrá disponer del 100% del tiempo del tiempo de aumento permitido (t_max) y las otras fases cuya d_sup_j sea positiva tomarán solo un porcentaje de t_max equivalente al porcentaje que d_sup_j representa respecto a la d_sup_j mayor, que es el 100%.

$$t_j \leq \frac{d\_sup_j \times t\_max}{d\_max} \qquad \forall j \in F$$

Lo anterior es con la finalidad de evitar, por ejemplo, que si la fase 1 requiere de un aumento de 30 segundos y la fase 2 requiere de un aumento de 60 segundos, a ambas se les asigne un aumento de 30 como consecuencia de la acotación del tiempo máximo que una fase puede aumentar de un ciclo a otro. Con la restricción mencionada a la fase 2 se le asignaría el aumento máximo permitido que en este caso es de 30 y a la fase 1 se le asignaría un aumento de 15 segundos.

De manera similar se establece el tiempo a disminuir de cada fase, esto de manera proporcional de acuerdo a los valores de d_inf_j.

$$u_j \leq \frac{d\_inf_j \times t\_max}{d\_min} \qquad \forall j \in F$$

Resumiendo, el problema de programación cuadrática propuesto queda de la siguiente manera:

$$Min\ z = \sum_{j \in F} (d\_sup_j - t_j e_j)^2 + \sum_{j \in F} (d\_inf_j - u_j e_j)^2$$

s.a

$$\sum_{j \in F} (ta_j + t_j - u_j) = TC$$

$$t_j \leq t\_max$$

$$u_j \leq t\_max$$

$$ta_j + t_j - u_j \leq TC(lim\_\max\ \_fase)$$

$$ta_j + t_j - u_j \geq TC(lim\_\min\ \_fase)$$

$$t_j e_j \leq esp\_sig_j$$

$$t_j e_j \leq d\_sup_j$$

$$u_j e_j \leq d\_inf_j$$

$$t_j \leq \frac{d\_sup_j \times t\_max}{d\_max}$$

$$u_j \leq \frac{d\_inf_j \times t\_max}{d\_min} \qquad \forall j \in F$$

[Problema 2.1]

2.3. **Ejemplo**

A continuación se muestra el planteamiento del problema para una intersección, la cual forma parte de un área de red de tráfico urbano de 6 intersecciones como la mostrada en la figura 2.4, en donde en cada intersección los tiempos de fase se han calculado resolviendo el problema 2.1. Los datos ilustrados son los utilizados en la simulación de área del escenario 1 en la intersección 1 en el ciclo ocurrido en el intervalo t=2000 seg. y t=2100 seg.

Cada una de las intersecciones de la red mostrada en la figura 2.4 corresponde a la intersección descrita con la figura 2.5.

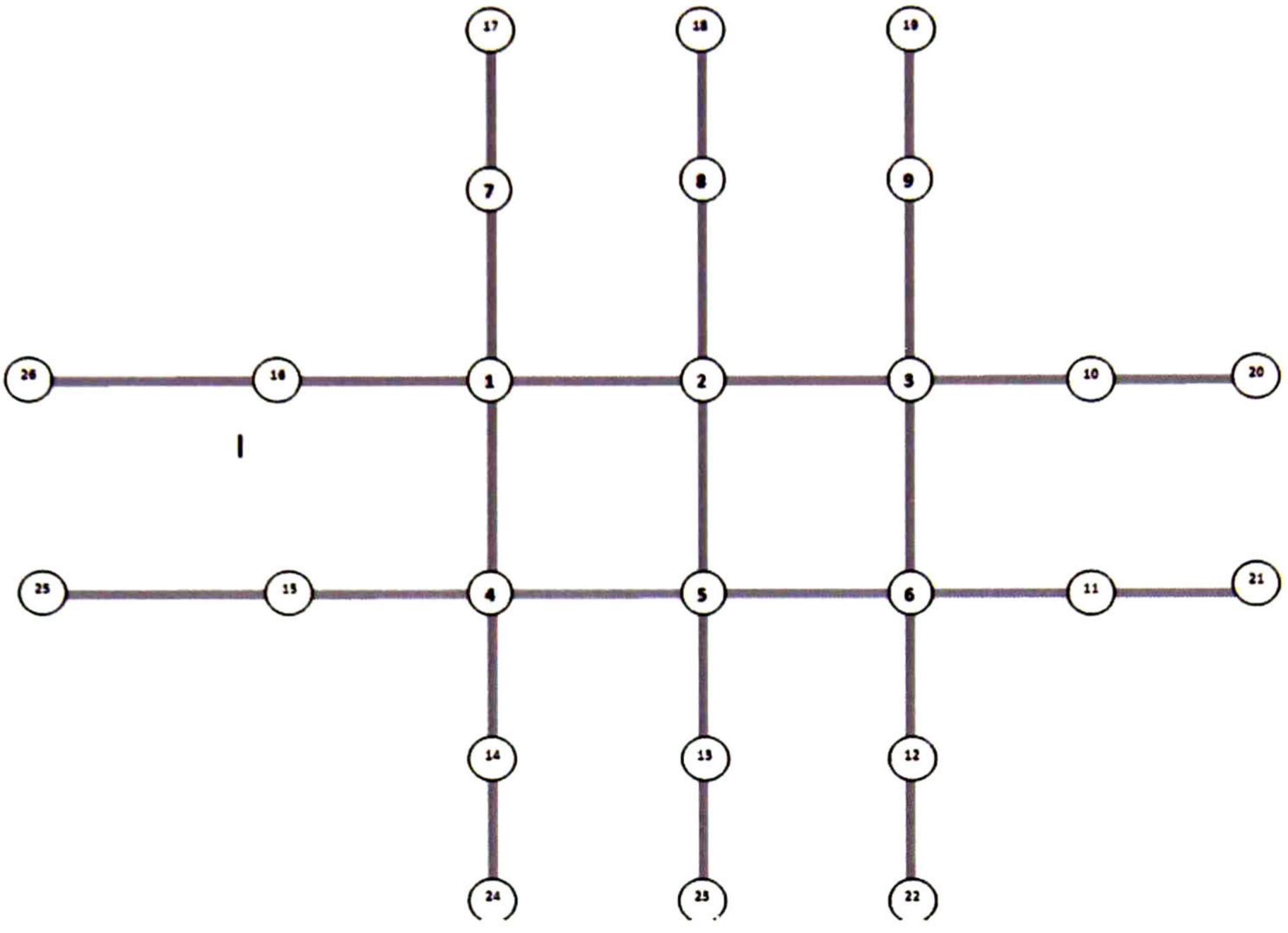


Figura 2. 4 Red de caminos con 6 intersecciones

La figura 2.5 representa la intersección que será controlada mediante el esquema de programación cuadrática propuesto. Esta intersección consta de 4 fases, una para cada calle de entrada, en cada fase se permite transitar hacia todos los sentidos.

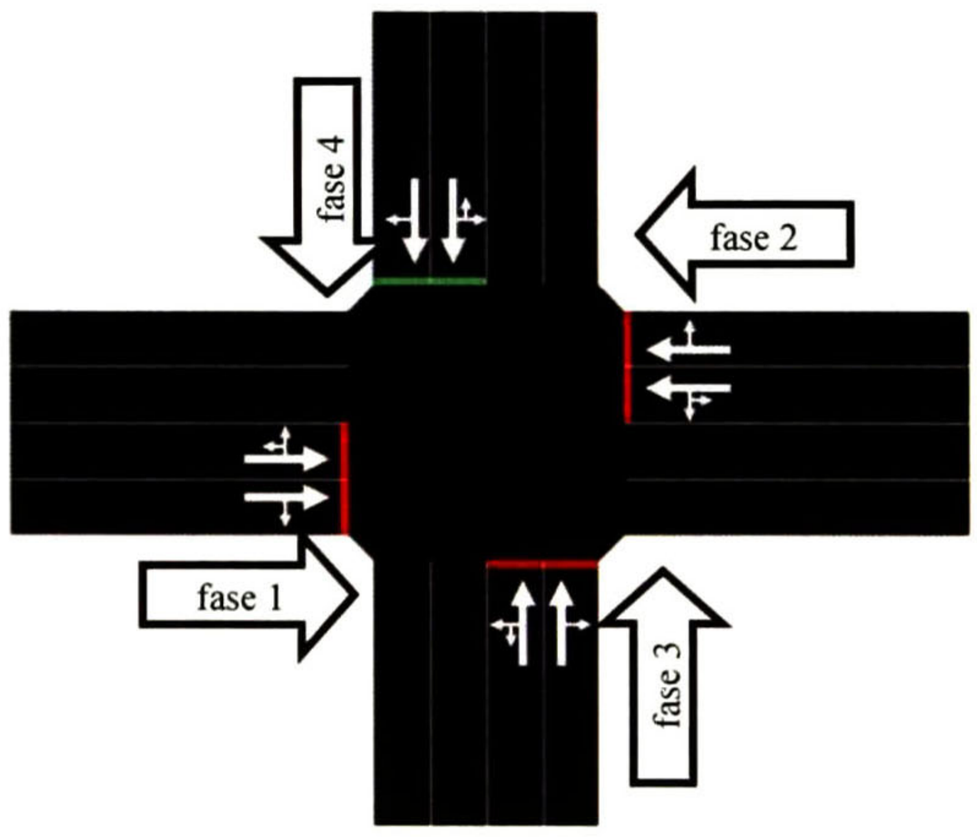


Figura 2. 5 Una intersección de la red

A continuación se muestran los datos que se obtendrían de los sensores de tráfico y el planteamiento de programación cuadrática para esta intersección.

Conjunto de parámetros:

F	{1, 2, 3, 4}
TC	100 seg.
ta_1	25 seg.
ta_2	25 seg.
ta_3	25 seg.
ta_4	25 seg.
e_1	0.96 veh/seg
e_2	0.96 veh/seg
e_3	0.96 veh/seg
e_4	0.96 veh/seg
t_max	20 seg.
lim_max_fase	0.80
lim_min_fase	0.05
n_km_1	0.5
n_km_2	0.5
n_km_3	0.5
n_km_4	0.5
Promedio de densidades	224.21+12.98+9.72+8.02=63.73
d_sup_1	160.48 veh/km
d_sup_2	0 veh/km
d_sup_3	0 veh/km
d_sup_4	0 veh/km
d_inf_1	0 veh/km
d_inf_2	50.75 veh/km
d_inf_3	54.01 veh/km
d_inf_4	55.71 veh/km
d_max	160.48 veh/km
d_min	55.71 veh/km

Estos parámetros se sustituyen en el problema 2.1 y se obtiene:

$Min\ z = (160.48 - t_1(0.96))^2 + (0 - t_2(0.96))^2 + (0 - t_3(0.96))^2 + (0 - t_4(0.96))^2 + (0 - u_1(0.96))^2 + (50.75 - u_2(0.96))^2 + (54.01 - u_3(0.96))^2 + (55.71 - u_4(0.96))^2$
s.a
$(25 + t_1 - u_1) + (25 + t_2 - u_2) + (25 + t_3 - u_3) + (25 + t_4 - u_4) = TC$
$t_1 \leq 20, \quad t_2 \leq 20, \quad t_3 \leq 20, \quad t_4 \leq 20$
$u_1 \leq 20, \quad u_2 \leq 20, \quad u_3 \leq 20, \quad u_4 \leq 20$
$25 + t_1 - u_1 \leq TC(0.80),\ 25 + t_2 - u_2 \leq TC(0.80),\ 25 + t_3 - u_3 \leq TC(0.80),\ 25 + t_4 - u_4 \leq TC(0.80)$
$25 + t_1 - u_1 \geq TC(0.05), 25 + t_2 - u_2 \geq TC(0.05), 25 + t_3 - u_3 \geq TC(0.05),\ 25 + t_4 - u_4 \geq TC(0.05)$
$t_1(0.96) \leq 345\ , t_2(0.96) \leq 344\ , t_3(0.96) \leq 343\ , t_4(0.96) \leq 339$
$t_1(0.96) \leq 160.48\ , t_2(0.96) \leq 0\ , t_3(0.96) \leq 0\ , t_4(0.96) \leq 0$
$u_1(0.96) \leq 0\ , u_2(0.96) \leq 50.75\ , u_3(0.96) \leq 54.01\ , u_4(0.96) \leq 55.71$
$t_1 \leq \frac{160.48\ (20)}{160.48}, \quad t_2 \leq \frac{0\ (20)}{160.48}, \quad t_3 \leq \frac{0\ (20)}{160.48}, \quad t_4 \leq \frac{0\ (20)}{160.48}$
$u_1 \leq \frac{0\ (20)}{55.71}. \quad u_2 \leq \frac{50.75\ (20)}{55.71}. \quad u_3 \leq \frac{54.01(20)}{55.71}. \quad u_4 \leq \frac{55.71\ (20)}{55.71}$

Resolviendo mediante el método de conjunto activo con la función QPsolve de la aplicación MAPLE se obtiene:

$$t_1 = 20\ , t_2 = 0, t_3 = 0, t_4 = 0, u_1 = 0, u_2 = 4, u_3 = 7, u_4 = 9$$

De acuerdo a las variables obtenidas y dado que los tiempos de fase en el ciclo anterior al instante 2100 (seg.) son de 25 segundos para cada fase, se determina que a partir del instante 2100 de la simulación los tiempos de fase a aplicar serán los siguientes:

Fase 1: 25 + 20 - 0 =45

Fase 2: $25 + 0 - 4 = 21$

Fase 3: $25 + 0 - 7 = 18$

Fase 4: $25 + 0 - 9 = 16$

Se observa que a la fase 1 se le asignó el mayor incremento de tiempo permitido, ya que su densidad estaba muy por arriba del promedio. A las fases 2, 3 y 4 se les redujo el tiempo,

siendo la fase 4 quien tuvo la mayor reducción debido a que era la fase que poseía la menor densidad vehicular.

Con la aplicación de los nuevos tiempos de fase, se logra que la calle de entrada a la intersección que tiene la mayor cantidad de vehículos goze de un mayor tiempo de verde, mientras que las calles que no estén en esta situación reducen su tiempo de fase. Consecuentemente las diferencias entre las longitudes de colas de espera serían menores y una mayor cantidad de vehículos lograría cruzar la intersección durante el ciclo.

2.4. **Conclusiones**

En el presente capítulo se planteó una estrategia para determinar los tiempos de verde en intersecciones basada en la solución de un problema de programación cuadrática. La estrategia propuesta balancea las densidades de tráfico en las calles de entrada a una intersección tomado en cuenta el efecto posible en las intersecciones vecinas dado el flujo de salida autorizado. Con esto no es necesaria la comunicación con intersecciones vecinas.

Capítulo 3

Implementación de control de tráfico basado en programación cuadrática

---

Resumen: En este capítulo se muestra cómo fue implementada la estrategia de control propuesta en el simulador SUMO. Se describe la arquitectura definida y los módulos desarrollados y se presenta el procedimiento para realizar una simulación a partir de un caso de estudio dado.

3.1. **Simuladores de tráfico**

Para predecir la eficiencia de estrategias de control de tráfico aplicadas en un sistema de tráfico urbano real, los ingenieros de transporte y analistas de planeación urbana han hecho uso de la simulación como herramienta de soporte en la evaluación de desempeño de estos dispositivos, ya que permite evaluar el desempeño de estrategias de control mediante el análisis del flujo vehicular [López, 2009].

Para el propósito de evaluar el desempeño del sistema de control propuesto se buscó un simulador que cumpliera con algunas características específicas, como son:

- Micro simulación, ya que simula el tráfico a nivel de vehículos individuales.
- Código abierto que permita la implementación de control dinámico de semáforos.
- Facilidad para poder implementar la estrategia de programación cuadrática.
- Documentación y manuales de uso disponibles
- Preferentemente simulación gráfica, para observar de manera intuitiva el comportamiento del tráfico bajo el sistema de control.
- Libre acceso

Para llevar a cabo la simulación del controlador propuesto, requisitos como la simulación a gran velocidad o de una red de caminos muy amplia no eran estrictamente necesarios.

Para evaluar la eficiencia del sistema de control propuesto se analizaron varias opciones existentes en cuanto a simuladores de tráfico; a continuación se mencionan algunas:

- SimTraffic
- CORSIM
- AIMSUN
- SCATTER
- MATSIM-1
- Simulación distribuida de STU
- SUMO
- VISSIM
- VISUM
- METACOR

3.2. **Simulador SUMO**

Para conocer el desempeño del método de control propuesto se eligió usar el simulador de tráfico SUMO (Simulation of Urban MObility), debido a que en su momento se observó que contaba con las siguientes características:

- Documentación disponible en línea, en *http://sumo.sourceforge.net*
- Gratuito
- Código abierto
- Entorno gráfico de simulación
- Lista de distribución para retroalimentación entre usuarios
- Amplia comunidad de usuarios
- Lista de distribución sobre noticias de nuevas versiones y documentación
- Simulación microscópica

SUMO es un paquete de simulación de tráfico multi-modal, microscópico y de código abierto que es desarrollado desde el año 2000 por el Institute of Transportation Research en el German Aerospace Centre. La idea detrás del desarrollo de un simulador de tráfico de código abierto es apoyar a la comunidad en investigación de tráfico con un simulador que proporcione resultados de investigaciones más comparables y facilite la prueba de algoritmos desarrollados en la investigación de tráfico [Krajzewicz, 2006].

3.2.1. **Implementación del sistema de control en simulador SUMO**

3.2.1.1. **Construcción de la red de tráfico**

SUMO ofrece la posibilidad de importar mapas existentes almacenados en una base de datos ArcView, redes desarrolladas en VISSIM, redes desarrolladas en VISUM, bases de datos TIGER (Topologically Integrated Geographic Encoding and Referencing) o desde una descripción en un archivo XML.

Para someter a prueba el método de control propuesto se optó por realizar la descripción de la red de calles mediante archivos XML; la descripción de la red se debe codificar a mano, cuestión que resultaría muy poco práctica para implementar la red completa de una ciudad o incluso de una zona de la misma, pero para los escenarios de prueba en este trabajo fue la mejor opción, ya que consisten en una zona con un máximo de 6 intersecciones.

Para realizar el archivo XML que describe la red a utilizar en la simulación es necesario construir primero por lo menos dos archivos XML: uno para las intersecciones (nodos) y otro para las calles (aristas) que hay entre ellos.

A continuación se muestra como ejemplo algunas líneas de un archivo que describe las intersecciones de una red, este archivo deberá tener la extensión *.nod.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
 <nodes> <!-- The opening tag -->
 <node id="1" x="0.0" y="0.0" type="traffic_light"/>
 <node id="2" x="+500.0" y="0.0" type="traffic_light"/>
  <node id="3" x="+1000.0" y="0.0" type="traffic_light"/>
 <node id="4" x="0.0" y="-500.0" type="traffic_light"/>
 <node id="5" x="+500.0" y="-500.0" type="traffic_light"/>
 <node id="6" x="+1000.0" y="-500.0" type="traffic_light"/>
 <node id="7" x="0.0" y="+500.0" type="priority"/>
 <node id="8" x="+500.0" y="+500.0" type="priority"/>


 </nodes> <!-- The closing tag -->
```

Aunque no es posible aún visualizar en el simulador el archivo descrito anteriormente, debido a que se requiere la red completa, en la figura 3.1 se muestran los nodos definidos en el ejemplo mostrado y las coordenadas donde se establecen; nótese que en esta etapa es donde se define cuales intersecciones poseerán un semáforo.

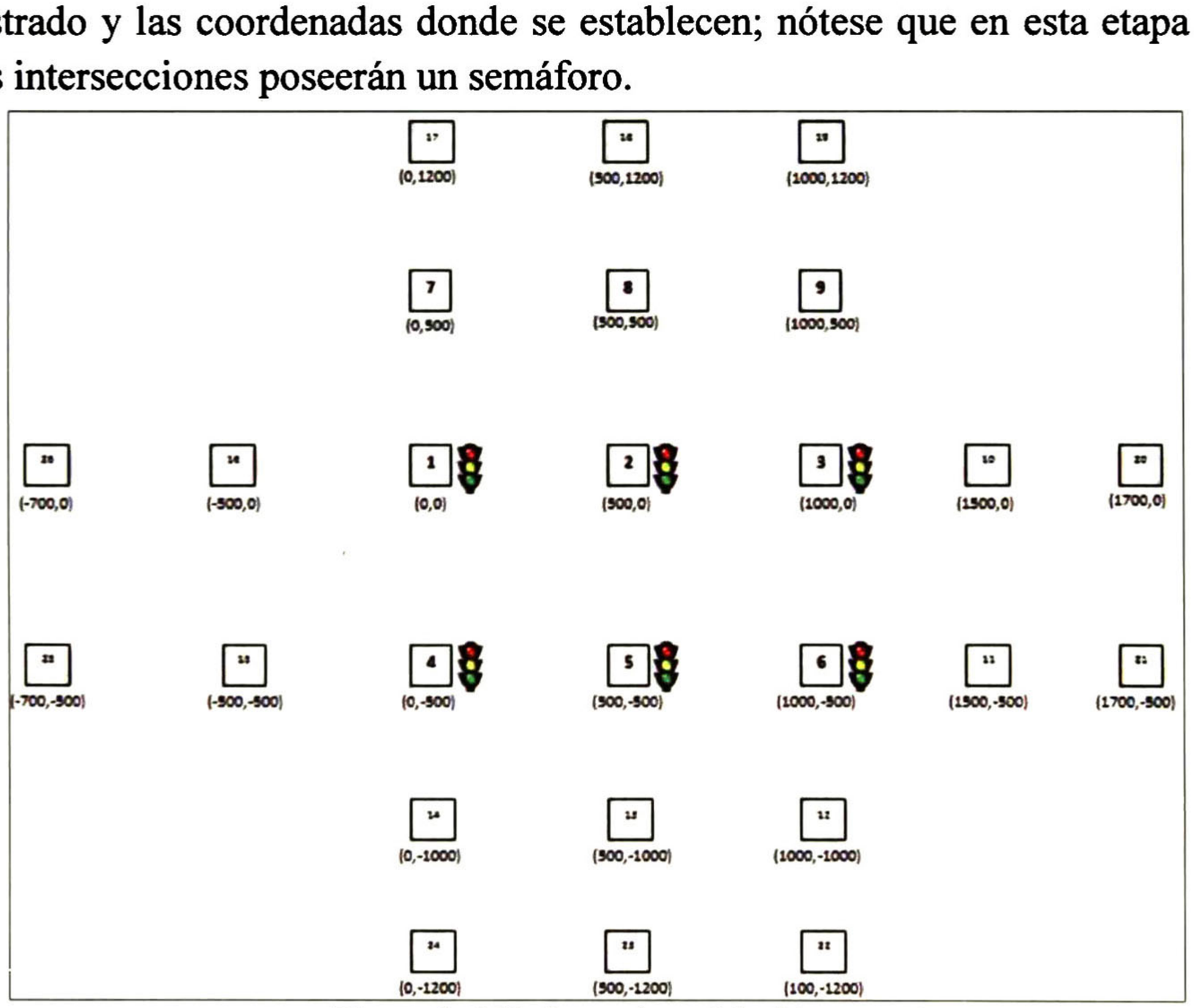


Figura 3. 1 Nodos en la red de 6 intersecciones

En el archivo que describe las calles se especifica cuál es el nodo origen y cuál es el nodo destino, estableciendo a la vez el sentido de circulación de los autos. A continuación se muestran algunas líneas de un archivo que describe las calles de una red, este archivo deberá tener la extensión *.edg.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<edges>
 <!--centrales-->
 <edge id="1-2" fromnode="1" tonode="0" priority="2" nolanes="2" speed="13.89"/>
 <edge id="2-1" fromnode="0" tonode="1" priority="2" nolanes="2" speed="13.89"/>
 <edge id="2-3" fromnode="2" tonode="0" priority="2" nolanes="2" speed="13.89"/>
 <edge id="3-2" fromnode="0" tonode="2" priority="2" nolanes="2" speed="13.89"/>
 <edge id="3-6" fromnode="3" tonode="0" priority="2" nolanes="2" speed="13.89"/>
 <edge id="6-3" fromnode="0" tonode="3" priority="2" nolanes="2" speed="13.89"/>


<edge id="26-16" fromnode="2" tonode="2a" priority="2" nolanes="2" speed="13.89"/>
</edges>
```

Una vez definidos los archivos de intersecciones y el archivo de calles, la aplicación *netconvert* de SUMO genera el archivo que representa la red de caminos a utilizar por el simulador. Sin embargo, esta aplicación construye la red bajo ciertas reglas de tránsito diferentes a las que se utilizan en ciudades mexicanas, por ejemplo que el carril del extremo derecho sea exclusivo para quienes darán vuelta hacia la derecha. Para solucionar esto y establecer en la red de caminos que la circulación de autos sea como en ciudades mexicanas es necesario construir otro archivo XML, en donde se especifica cada carril de cada intersección desde cual y hacia cual es permitida la circulación de los autos. A continuación se muestra un fragmento de código de un archivo de conexiones, cuya extensión debe ser *.con.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<connections>
 <!--carril 1i-->
 <connection from="1i" to="4o" lane="1:0"/> <!--vuelta a la izq. y seguir recto-->
  <connection from="1i" to="4o" lane="1:1"/>
  <connection from="1i" to="2o" lane="1:0"/>
  <connection from="1i" to="2o" lane="1:1"/>
  <connection from="1i" to="2o" lane="0:0"/><!--vuelta a la derecha y seguir recto-->
  <connection from="1i" to="2o" lane="0:1"/>
  <connection from="1i" to="3o" lane="0:0"/>
<connection from="1i" to="3o" lane="0:1"/>
<connection from="1i" to="1o" lane="1:0"/><!--vuelta en u-->
<connection from="1i" to="1o" lane="1:1"/>

<!--carril 2i-->
<connection from="2i" to="4o" lane="0:0"/>  <!--vuelta a la der y seguir recto-->
<connection from="2i" to="4o" lane="0:1"/>
<connection from="2i" to="1o" lane="0:0"/>
<connection from="2i" to="1o" lane="0:1"/>


</connections>
```

Con los archivos xml de intersecciones, calles y conexiones, la aplicación netconvert de SUMO generará la red a utilizar por el simulador.

En la figura 3.2 se muestra la red que es construída por *netconvert* especificando los nodos, aristas y conexiones entre calles; se muestra la vista detallada de una de las intersecciones con semáforo, recordando que las 6 intersecciones que cuentan con semáforo son similares.

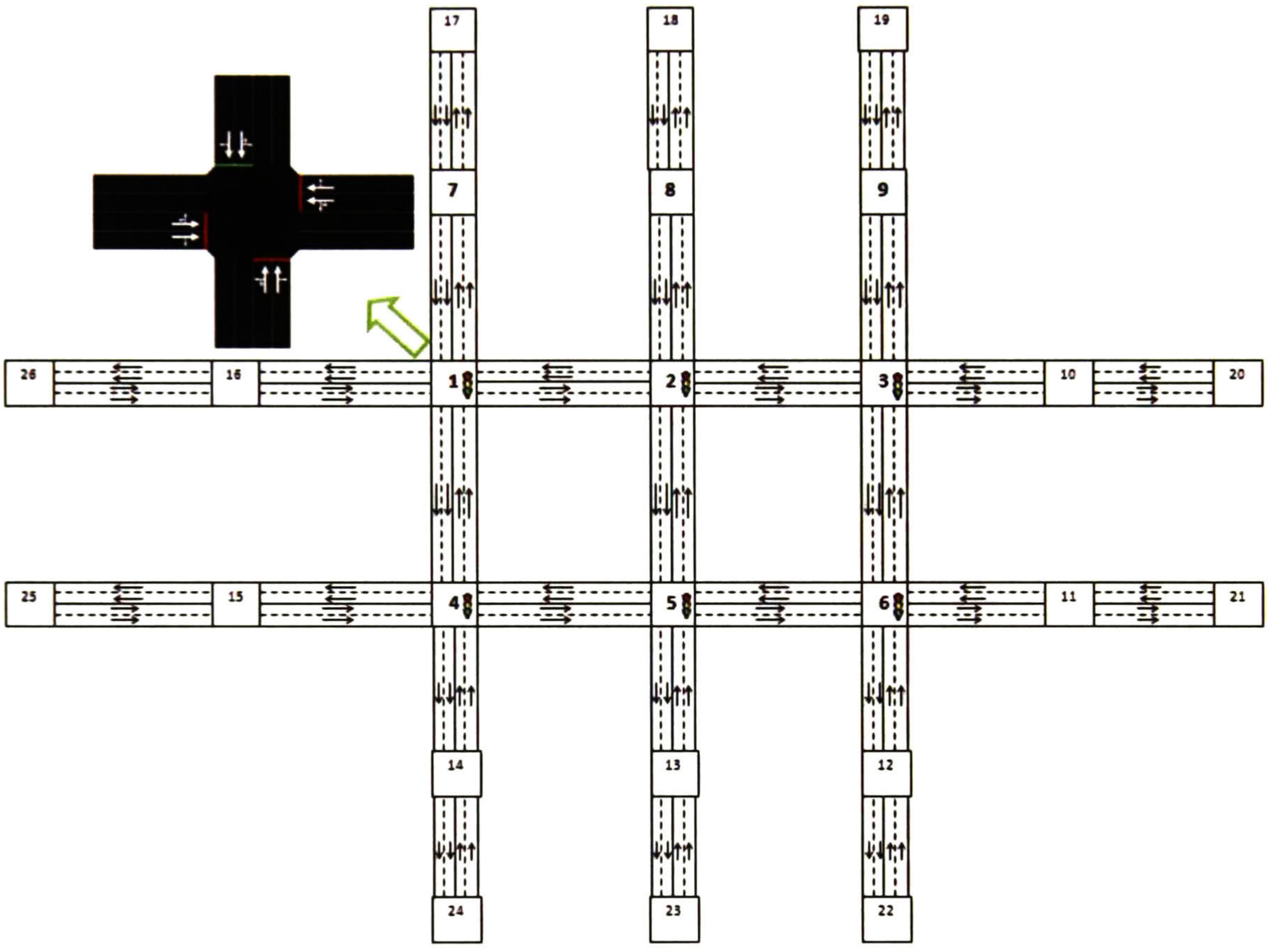


Figura 3. 2 Red de 6 intersecciones generada

3.2.1.2. **Definición de flujos de entrada**

Un método para establecer el flujo de vehículos de entrada en los diferentes escenarios de prueba, es construir un archivo XML en donde se especifica para cada calle de entrada, la calle de salida hacia donde se dirige ese flujo, el periodo de tiempo en segundos en que se estará emitiendo ese flujo, el número de vehículos que conforman este flujo, y el color de los autos que pertenecen a él. Los tiempos de salida se distribuyen uniformemente en el intervalo de tiempo

especificado. A continuación se muestra un fragmento de código de un archivo que define flujos de entrada para una red, su extensión es *.flows.xml:

```
<flowdefs>

<flow id="0" from="1ai" to="2ao" begin="0" end="2999" no="825" color="1,0,0"/>
 <flow id="1" from="2ai" to="1ao" begin="0" end="2999" no="600" color="0,1,0"/>
 <flow id="2" from="3ai" to="4ao" begin="0" end="2999" no="600" color="0,0,1"/>
 <flow id="4" from="4ai" to="3ao" begin="0" end="2999" no="825" color="0.5,0,0.5"/>

 <flow id="5" from="1ai" to="2ao" begin="3000" end="3999" no="325" color="1,0,0"/>
 <flow id="6" from="2ai" to="1ao" begin="3000" end="3999" no="200" color="0,1,0"/>
 <flow id="7" from="3ai" to="4ao" begin="3000" end="3999" no="138" color="0,0,1"/>
 <flow id="8" from="4ai" to="3ao" begin="3000" end="3999" no="275" color="0.5,0,0.5"/>

 <flow id="9" from="1ai" to="2ao" begin="4000" end="5000" no="375" color="1,0,0"/>
 <flow id="10" from="2ai" to="1ao" begin="4000" end="5000" no="200" color="0,1,0"/>
 <flow id="11" from="3ai" to="4ao" begin="4000" end="5000" no="75" color="0,0,1"/>
 <flow id="12" from="4ai" to="3ao" begin="4000" end="5000" no="275" color="0.5,0,0.5"/>l

</flowdefs>
```

Opcionalmente se puede especificar para cada intersección, el porcentaje de vehículos que llegan por cada calle se dirigirán en línea recta, hacia la derecha, hacia la izquierda o cualquier otro camino de los posibles, esto mediante un archivo cuya extensión debe ser *.turn.xml, a continuación se muestra un fragmento de código de un archivo de este tipo:

```
<?xml version="1.0" encoding="utf-8"?>
<turn-defs>
    <interval begin="0" end="5000">
        <!-- -------- NODO 1 ---------------------------->
        <fromedge id="16-1">
                <toedge id="1-2" probability="0.4"/>
                <toedge id="1-7" probability="0.3"/>
                <toedge id="1-4" probability="0.3"/>
        </fromedge>
        <fromedge id="4-1">
                <toedge id="1-7" probability="0.4"/>
                <toedge id="1-16" probability="0.3"/>
                <toedge id="1-2" probability="0.3"/>
        </fromedge>
        <fromedge id="2-1">
                <toedge id="1-16" probability="0.4"/>
                <toedge id="1-7" probability="0.3"/>
                <toedge id="1-4" probability="0.3"/>
        </fromedge>



    </interval>

<!-- son las aristas donde los vehiculos dejaran la red -->
<sink>7-17</sink>
<sink>8-18</sink>
```

<sink>9-19</sink>

</turn-defs>

Teniendo ya los archivos de red, de flujos y el que define las tasas de vuelta, la aplicación *jtrrouter* de SUMO genera el archivo de rutas que utilizará el simulador, su extensión deberá ser *.rou.xml.

3.2.1.3. **Datos de salida de la simulación**

Para obtener los parámetros de salida de la simulación, tales como densidades de tráfico y espacios libres en las calles, se construyen archivos xml especificando el tipo de detector que se desea colocar en la simulación, el espacio físico que se desea vigilar, el periodo de tiempo en que se tomarán las mediciones y el archivo de salida donde se acumularán los datos medidos.

Para las simulaciones realizadas en esta tesis se utilizaron tres tipos de especificaciones de salida:

- **Detector tipo e-2 de área por carriles**

Para cada carril del camino que se desea vigilar se construye un archivo, a continuación se muestra un ejemplo:

<e2-detector id="1" lane="1-2_0" pos="0" length="483" tl="1" file="1-2_0.txt" />

Aquí se especifica cual carril y la longitud del mismo que se desea vigilar, en este caso se estaría vigilando el carril completo; también se especifica que la longitud del periodo de tiempo durante el cual se toman mediciones es de acuerdo a la duración de las fases de el semáforo identificado con el número uno y al final se especifica el nombre del archivo de salida donde se colocarán los resultado de la medición.

Un archivo de salida de un detector tipo e-2 contiene líneas como la siguiente:

<interval begin="0" end="25" id="1" nSamples="0" meanSpeed="-1.00" meanOccupancy="0.00" maxOccupancy="0.00" meanMaxJamLengthInVehicles="0.00" meanMaxJamLengthInMeters="0.00" maxJamLengthInVehicles="0" maxJamLengthInMeters="0.00" jamLengthInVehiclesSum="0" jamLengthInMetersSum="0.00" meanHaltingDuration="0.00" maxHaltingDuration="0" haltingDurationSum="0" meanIntervalHaltingDuration="0.00" maxIntervalHaltingDuration="0" intervalHaltingDurationSum="0" startedHalts="0.00" meanVehicleNumber="0.00" maxVehicleNumber="0" />

Para realizar la optimización se toma en cuenta el número de espacios libres, por lo que es necesario extraer del archivo de salida mostrado la variable *meanVehicleNumber* para el intervalo de tiempo de la fase deseada. Con el valor de esta variable y la capacidad del carril de todo el segmento de calle, se calcula el número de espacios libres.

- **Estados de red basados en aristas**

Especificando este tipo de salida al simulador SUMO, se escriben en un archivo de texto varios datos sobre cada uno de los carriles que existen en la red completa, tomados a intervalos de tiempo especificados.

Se define de la siguiente manera:

<meandata-edge id="medir-densidad" freq="100" file="densi-opti-hasta-4900.txt" />

En este caso se especifica una frecuencia de registro de datos de 100 segundos, ya que para los escenarios de prueba esa es la duración de todas las fases.

A continuación se muestra un ejemplo del archivo que genera este tipo de especificación de salida:

<edge id="8-18" traveltime="51.11" sampledSeconds="8.20" density="0.12" occupancy="0.04" noStops="0" speed="13.70" entered="16" emitted="0" left="0"/>

<edge id="8-2" traveltime="74.37" sampledSeconds="946.19" density="19.23" occupancy="7.21" noStops="344" speed="6.88" entered="14" emitted="3" left="14"/>

<edge id="9-19" traveltime="51.60" sampledSeconds="8.02" density="0.11" occupancy="0.04" noStops="0" speed="13.57" entered="15" emitted="0" left="0"/>

<edge id="9-3" traveltime="76.38" sampledSeconds="945.13" density="19.21" occupancy="7.20" noStops="343" speed="6.74" entered="14" emitted="2" left="15"/>

Para realizar la optimización y mostrar resultados se toma en cuenta también la densidad en las aristas, por lo que de este archivo se extrae el valor de la variable *density* para cada una de las aristas en el ciclo que se esté optimizando. La densidad es el número de vehículos por kilómetro, este valor es normalizado por el número de movimientos observados, la longitud de la arista y el número de carriles.

- **Detector tipo e-1: ciclo de inducción**

Para cada carril del camino que se desea vigilar se construye un archivo, a continuación se muestra un ejemplo:

<e1-detector id="e1-16-1_0" lane="16-1_0" pos="491" freq="100" file="e1-16-1_0.txt"/>

Se especifica cual carril se desea vigilar, la posición donde se desea el detector y el intervalo de tiempo durante el cual se agregarán registros al archivo. Para el escenario de pruebas utilizado en este trabajo la longitud de los carriles es de quinientos metros, por lo que al

especificar la posición como en el ejemplo mostrado, se estaría colocando el detector justo en la línea que marca el alto en la intersección.

El archivo que esta salida genera contiene líneas como las siguientes:

<interval begin="0" end="99" id="e1-16-1_1" nVehContrib="0" flow="0.00" occupancy="0.00" speed="-1.00" length="-1.00" nVehEntered="0"/>

<interval begin="100" end="199" id="e1-16-1_1" nVehContrib="7" flow="252.00" occupancy="45.61" speed="6.13" length="7.50" nVehEntered="7"/>

<interval begin="200" end="299" id="e1-16-1_1" nVehContrib="9" flow="324.00" occupancy="87.88" speed="4.87" length="7.50" nVehEntered="9"/>

El flujo de vehículos se utiliza para medir y comparar los resultados de la optimización, por lo que de los archivos de salida como el mostrado anteriormente se extrae el valor de la variable *flow*, el cual se encuentra expresado en vehículos por hora.

3.2.1.4. **Implementación de escenarios de prueba**

Para implementar el control de tiempos en semáforos fue necesario interactuar con otras aplicaciones, ya que si bien el simulador SUMO ofrecía bastante documentación y código abierto, no contaba con una interfaz directa al simulador que ofreciera control dinámico de semáforos; además el código fuente del simulador poseía una documentación muy general, por lo que se recurrió a aplicaciones externas.

En la figura 3.3 se ilustra de manera general la implementación del sistema de control propuesto en el simulador SUMO; se alimenta el simulador con 3 elementos:

- **Red**. Consiste en la descripción física de las calles, se forma mediante la especificación de nodos, aristas y conexiones entre calles.
- **Rutas**. Especifica las tasas de entrada de vehículos a la red. Se forma mediante la especificación de tasas y rutas a seguir por cada grupo de vehículos.
- **Detectores**. Establece el tipo de salida que el simulador generará, para este caso, se utilizaron detectores de densidad, espacios ocupados y flujo de vehículos.

El simulador arroja datos de salida para cada uno de los elementos de su red, por lo que es necesario filtrar estos datos para conservar solamente los que se utilizarán como entrada para el sistema de control.

Mediante la aplicación matemática MAPLE se resuelve el sistema de optimización y la variación de tiempos de fase a aplicar se escribe en el archivo que especifica la red, ya que este archivo debe contener los tiempos de fase para cada intersección a simular.

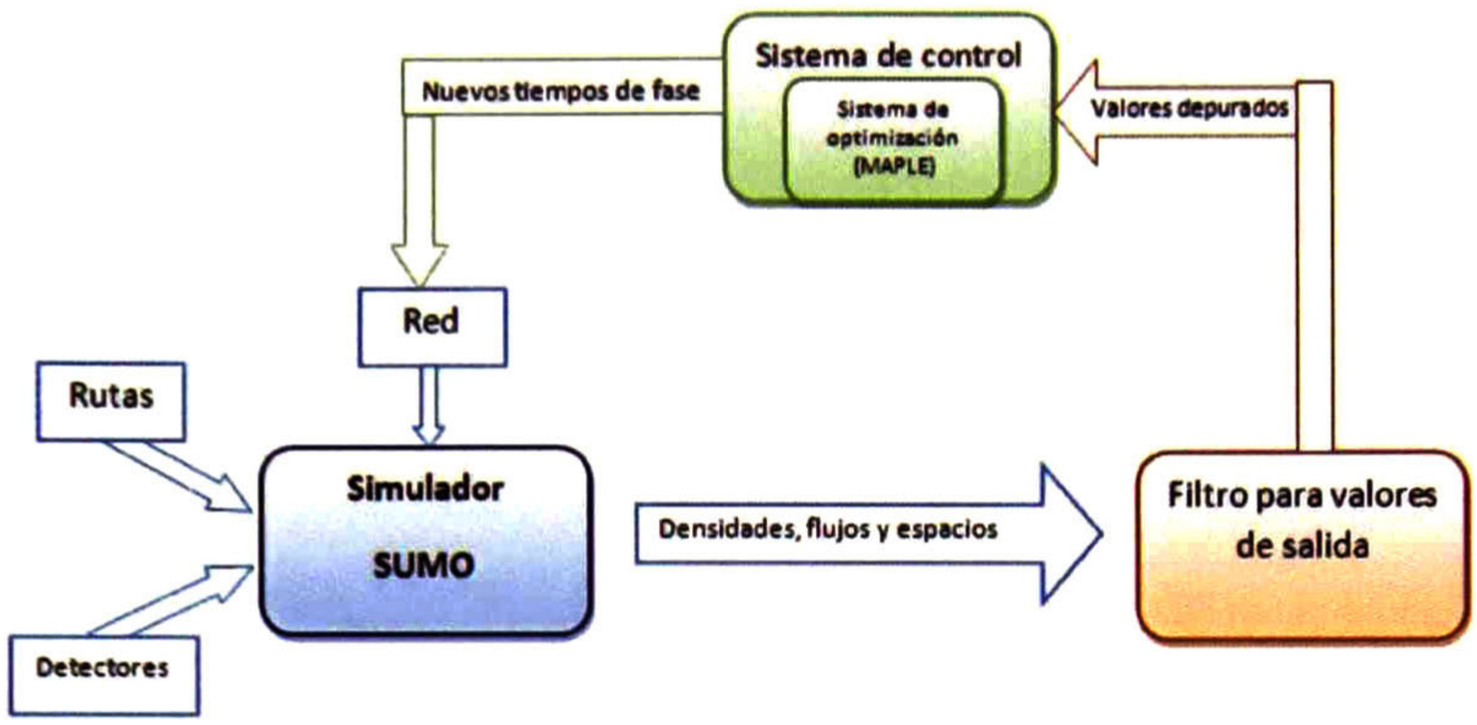


Figura 3. 3 Implementación del sistema de control en SUMO

En la figura 3.4 se muestra la arquitectura la cual describe de manera más detallada cómo se llevó a cabo la implementación del sistema de control propuesto en el simulador SUMO. En ésta figura se sombrea cada área con los colores utilizados en cada una de las tres etapas mostradas en la figura 3.3, para ilustrar a detalle cada una de ellas.

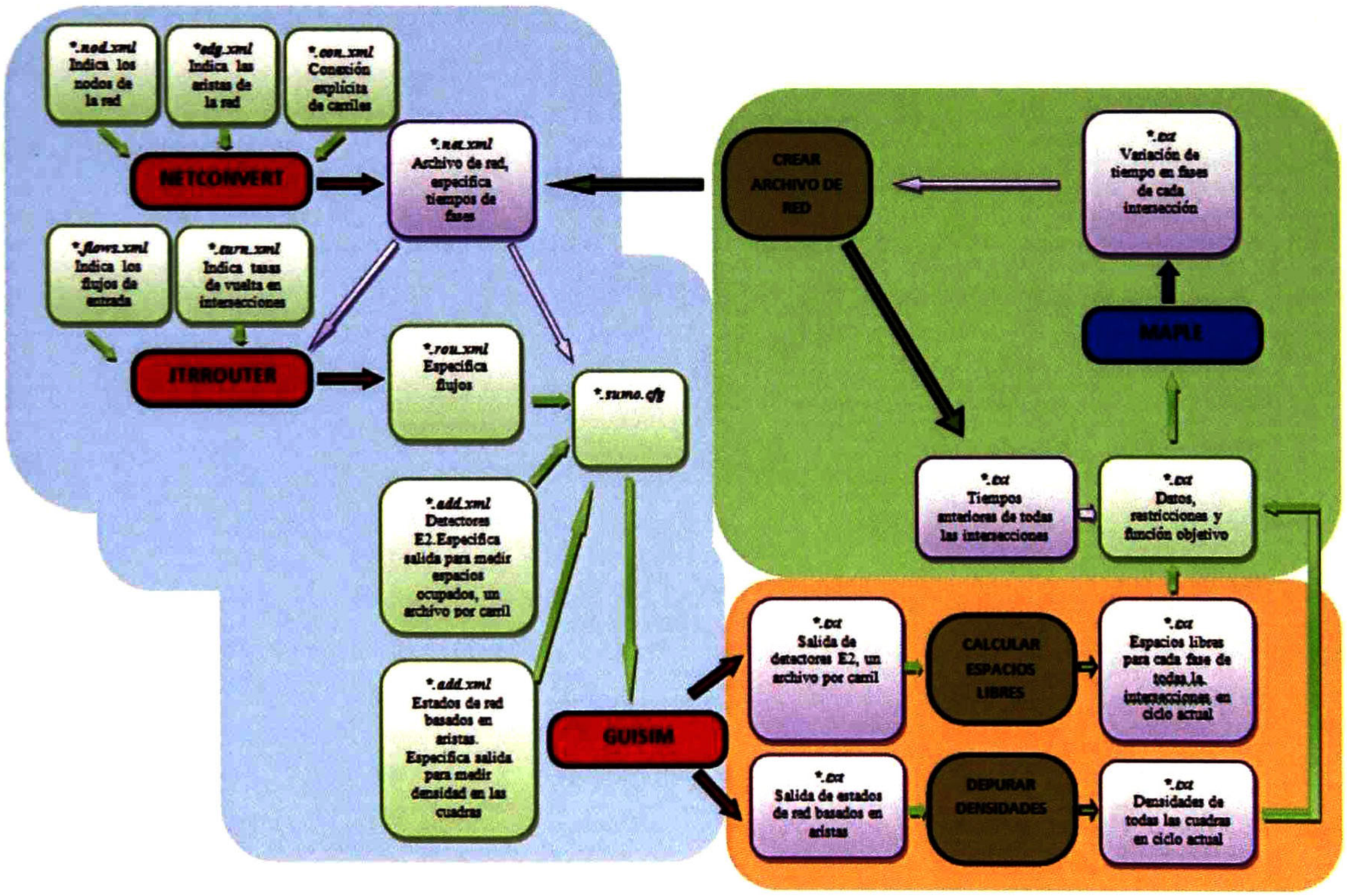


Figura 3. 4 Esquema detallado de la implementación del sistema de control en SUMO

Se realizaron programas en lenguaje C para extraer de los archivos de salida sólo las variables necesarias y proporcionarlas a la aplicación matemática MAPLE para resolver la optimización con estas variables; posteriormente, otro programa realizado en C tomaba la salida que MAPLE generaba, para construir el archivo xml que especifica la red y los tiempos en semáforos al simulador SUMO.

3.3. **Conclusiones**

Existen varios simuladores con el objetivo de que los investigadores en el área de ingeniería de tráfico prueben los sistemas de control desarrollados, no existe el "mejor" simulador, sino que dependiendo del caso y las necesidades de estudio se seleccionará el simulador que por sus características sea el más adecuado.

Capítulo 4

Pruebas y resultados

Resumen: En este capítulo se presentan los casos de prueba realizados, en donde se muestra que el sistema de control propuesto es capaz de adaptarse a las diferentes demandas de tráfico con el objetivo de mejorarlo.

Para poner a prueba la eficiencia del método de control presentado se realizaron simulaciones con varios escenarios de prueba. Se considera como escenario de prueba a un conjunto de tasas de entrada en una red de tráfico dada. Primero se realizaron pruebas en una red de una sola intersección y posteriormente se realizaron pruebas en una red con 6 intersecciones.

Cabe señalar que algunas características de la red de caminos fueron modificadas para que las reglas aplicadas hicieran la simulación más semejante al tráfico en calles urbanas mexicanas, ya que la red que el simulador genera por default posee reglas que no se utilizan en México, por ejemplo que el carril de la derecha sea exclusivo para vehículos que darán vuelta a la derecha.

4.1. Control de tráfico en una intersección

4.1.1. Descripción general

En la figura 4.1 se muestra la red de caminos con una sola intersección utilizada para las primeras pruebas, algunas características de esta red son:

- Todas las calles son de doble sentido.
- Cada sentido de la calle consta de dos carriles.
- La longitud de todos los segmentos de entrada a la intersección es de 500 metros.
- La longitud de los segmentos de entrada de la red es de 200 metros.
- Para cada calle de entrada a la intersección, el carril de la izquierda es para dar vuelta en U, vuelta a la izquierda o continuar en línea recta y el carril de la derecha es para continuar en línea recta o dar vuelta a la derecha. En la figura 4.2 se muestra de manera más detallada la intersección.
- La intersección posee un semáforo para cada calle de entrada.
- El tiempo de ciclo en la intersección es de 100 segundos.
- La intersección posee 4 fases de 25 segundos cada una. Para cada fase los vehículos de entrada a la intersección pueden ir hacia todas direcciones. En la figura 4.3 se muestran de manera gráfica las 4 fases mencionadas.

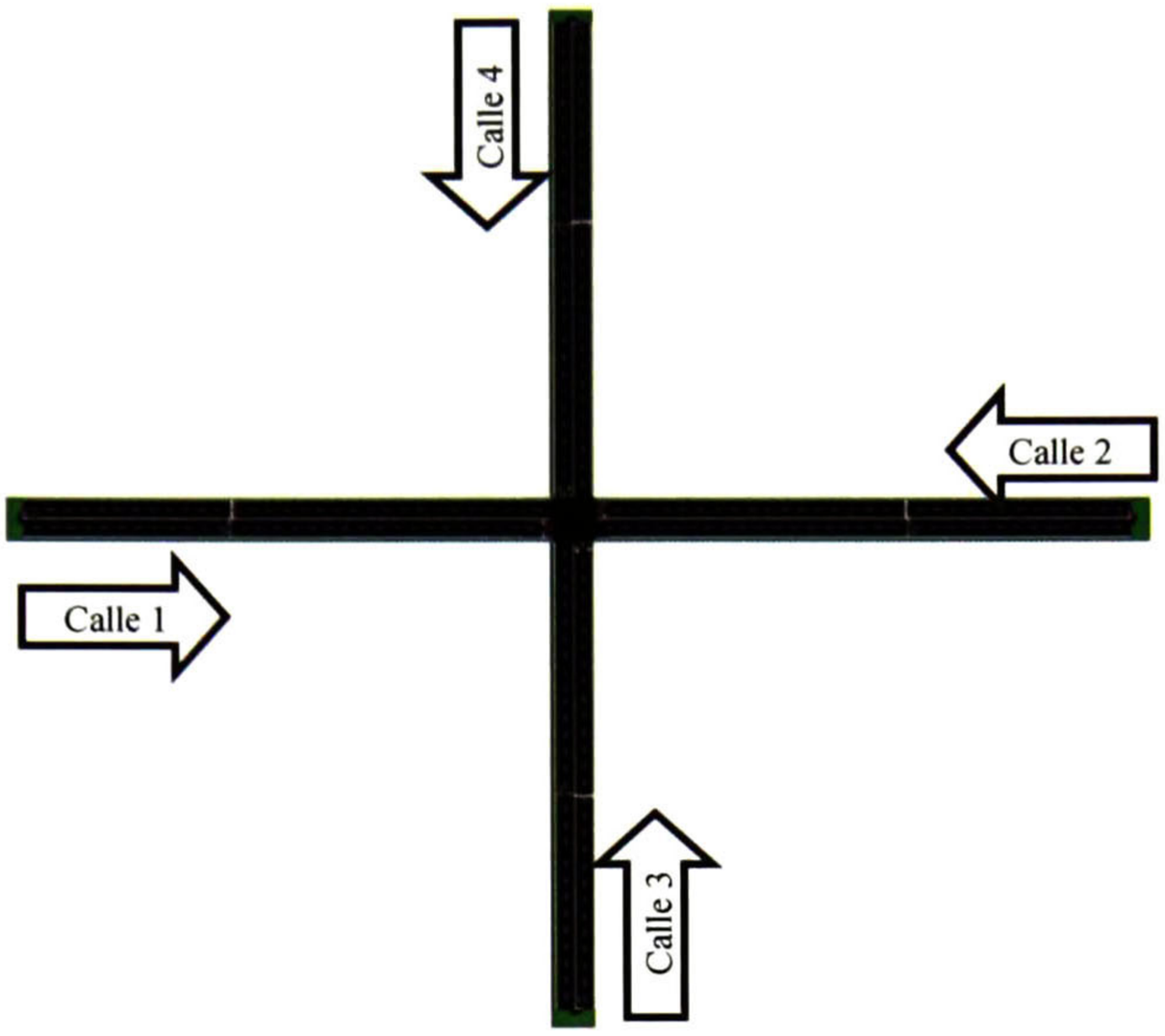


Figura 4. 1 Red de caminos con una intersección

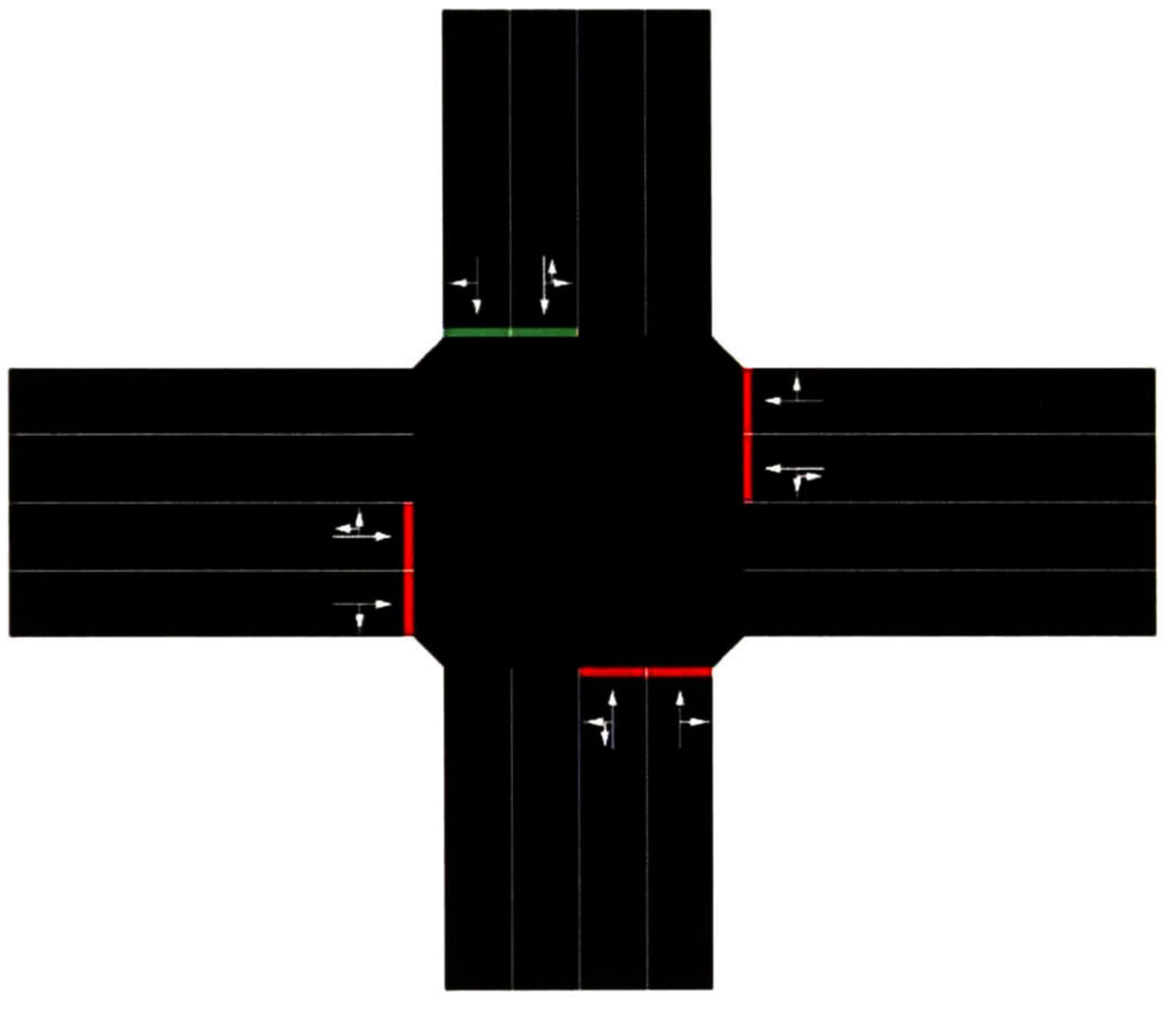


Figura 4. 2 Vista detallada de la intersección

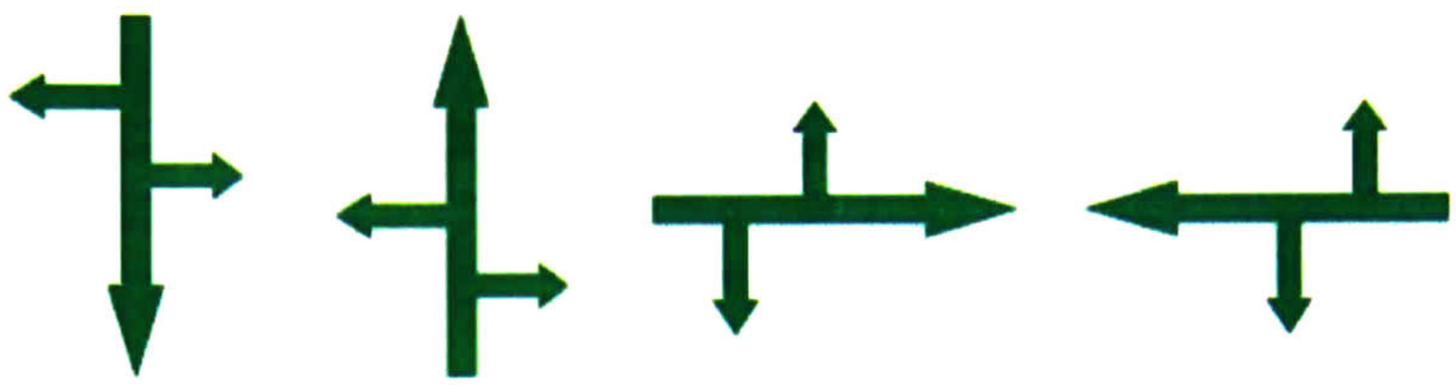


Figura 4. 3 Fases de la intersección

Además de las características de la red ya mencionadas, a continuación se mencionan las características del tráfico utilizadas; dichas características son conforme al modelo de seguimiento de vehículos desarrollado por Stefan Krauß [Krauß, 1998], las cuales se asignan por default a los vehículos en el simulador y tienen la opción de ser modificadas.

- Aceleración: 0.8 mt/s^2.
- Desaceleración: 4.5 mt/s^2.
- Imperfección del conductor (entre 0 y 1): 0.5.
- Longitud del vehículo: 5 mts.
- Máxima velocidad: 70 mt/s.

Todas las simulaciones realizadas para la red de caminos mostrada en la figura 4.1 se realizaron durante 5000 segundos, al inicio de la simulación la calle está vacía, por lo que las políticas de control se comienzan a aplicar hasta el instante 2000, para así trabajar con el tráfico ya estabilizado.

4.1.2. **Escenarios de prueba y resultados**

Para el caso de estudio de una sola intersección se tienen cinco escenarios con diferentes variaciones en sus tasas de llegada, para cada uno de los escenarios se analiza un esquema de control fijo y el esquema de control adaptable propuesto.

Para poder medir la eficiencia del sistema de control se optó por observar dos valores en la ejecución de las simulaciones; uno de ellos es la desviación estándar de las densidades (veh/km) existentes en las calles de entrada a la intersección y el otro valor es el promedio de flujo de vehículos (veh/hr) registrado en detectores posicionados en el extremo de cada segmento más cercano a la intersección.

Para cada escenario de prueba se comparan los valores mencionados tomados de un esquema de tiempos de fase fijos contra los mismos valores tomados bajo el esquema de tiempos de fase optimizados. Algunos de los datos que utiliza problema 2.1 son ajustados por el usuario; por ejemplo el tiempo que puede aumentar o disminuir una fase de un ciclo al siguiente (t_max)

y el porcentaje máximo o mínimo de ciclo permitido para una fase; (lim_max_fase y lim_min_fase).

Al realizar las simulaciones con los diferentes escenarios de prueba, se consideró que el porcentaje máximo de ciclo permitido para una fase es del 80% y el porcentaje mínimo de ciclo permitido para una fase es del 5%.

Escenario 1

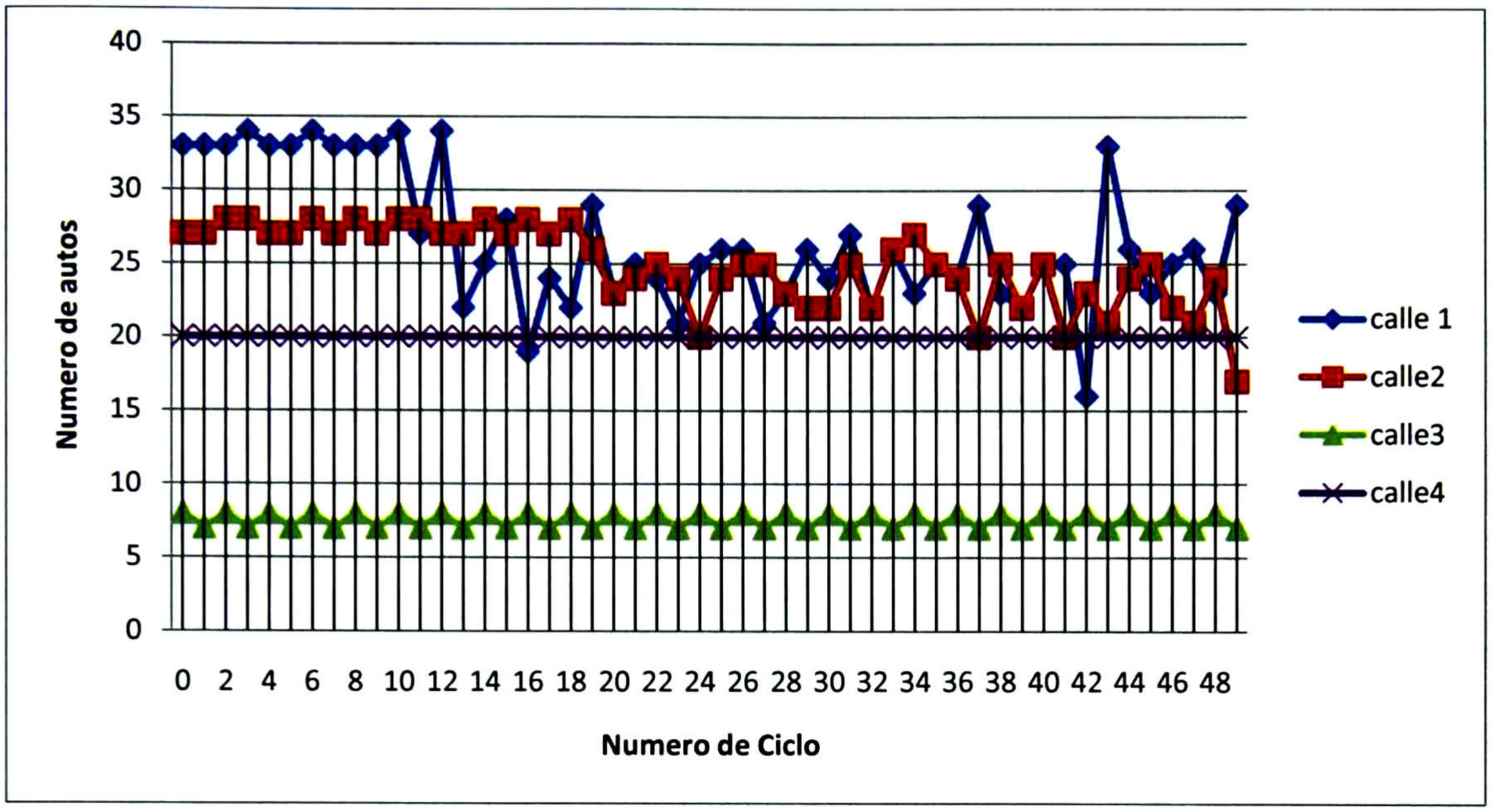
En la tabla 4.1 se muestran las tasas de llegada que se indicaron al simulador para el escenario 1, es este escenario se simuló un comportamiento de tráfico donde una calle de la intersección va saturada de tráfico mientras 2 calles se encuentran medianamente saturadas y 1 calle se encuentra con muy poco tráfico.

	Número de autos distribuidos uniformemente en el lapso de tiempo.			
Tiempo	**Calle 1**	**Calle 2**	**Calle 3**	**Calle 4**
0-2,999 seg.	1125	825	225	600
3,000-3,999 seg.	375	275	75	200
4,000-4999 seg.	375	275	75	200

Tabla 4. 1 Tasas de entrada de vehículos para escenario 1

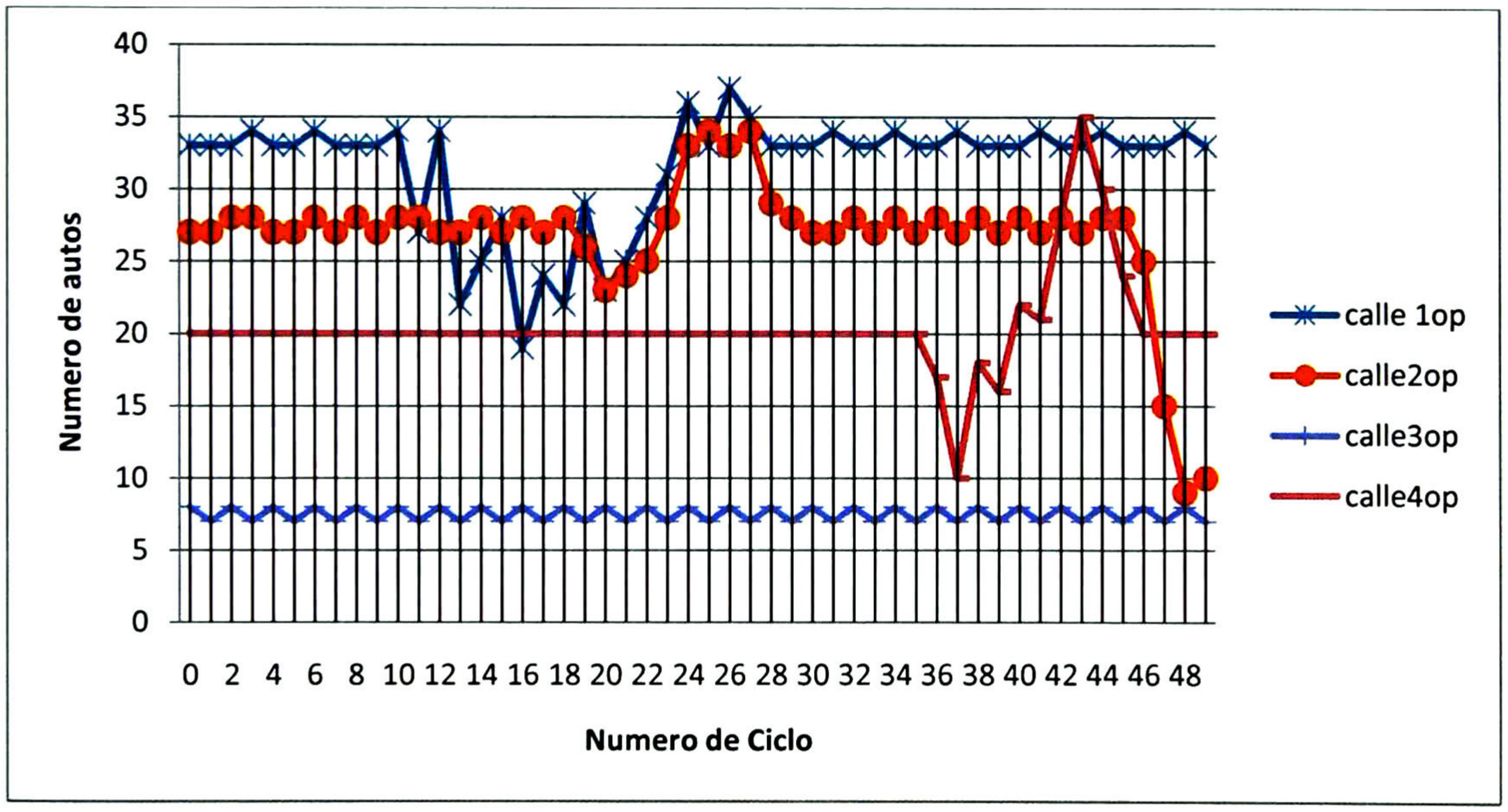
En la gráfica 4.1 se muestra la entrada de vehículos registrada en la simulación en el esquema de tiempos fijos.

Gráfica 4. 1 Entrada de vehiculos bajo escenario 1 con tiempos de fase fijos



En la gráfica 4.2 se muestra la entrada de vehículos registrada en la simulación en el esquema de tiempos óptimos.

Gráfica 4. 2 Entrada de vehiculos bajo escenario 1 con tiempos de fase optimizados



En las gráficas 4.1 y 4.2 se puede observar que los valores plasmados no corresponden con los valores en la tabla 4.1, esto es debido a que lo que se muestra en las tabla es una cota máxima indicada al simulador y lo que aparece en las gráficas es la entrada de autos que el simulador pudo llevar a cabo en la ejecución; dicho de otra forma, al simulador le podemos indicar que queremos ingresar 1000 vehículos en 100 segundos, pero en ejecución este solo puede ingresar 100 vehículos debido a factores como el tráfico que ya se encuentre en la red, velocidad máxima y capacidad de aceleración de los vehículos entre otros. Este comportamiento se observa también en el resto de los escenarios de prueba.

Bajo el escenario 1 se llevaron a cabo simulaciones con diferentes valores de *t_max*: de 5, 10 y 20 segundos para observar resultados parciales.

En la gráfica 4.3 se muestra la desviación estándar obtenida en la simulación del escenario 1 con un *t_max=5*. Cada una de las etiquetas de la gráfica corresponde a la simulación de 5000 segundos bajo las siguientes condiciones:

- Un esquema de tiempos de fase fijos durante toda la simulación. (Etiqueta *tiempo fijo*).

- Tiempos de fase fijos hasta el ciclo 21 (instante 2100), donde se aplicaron tiempos optimizados de acuerdo a las variables medidas en ciclo anterior (instante 2000 al 2100). Estos tiempos calculados persistieron en el resto de la simulación. (Etiqueta *2000*).

- Tiempos de fase fijos hasta el ciclo 21, donde se aplicaron tiempos optimizados de acuerdo al ciclo anterior y en el ciclo 22 se aplicaron tiempos optimizados de acuerdo al ciclo anterior. Estos tiempos calculados persistieron en el resto de la simulación. (Etiqueta *2100*).

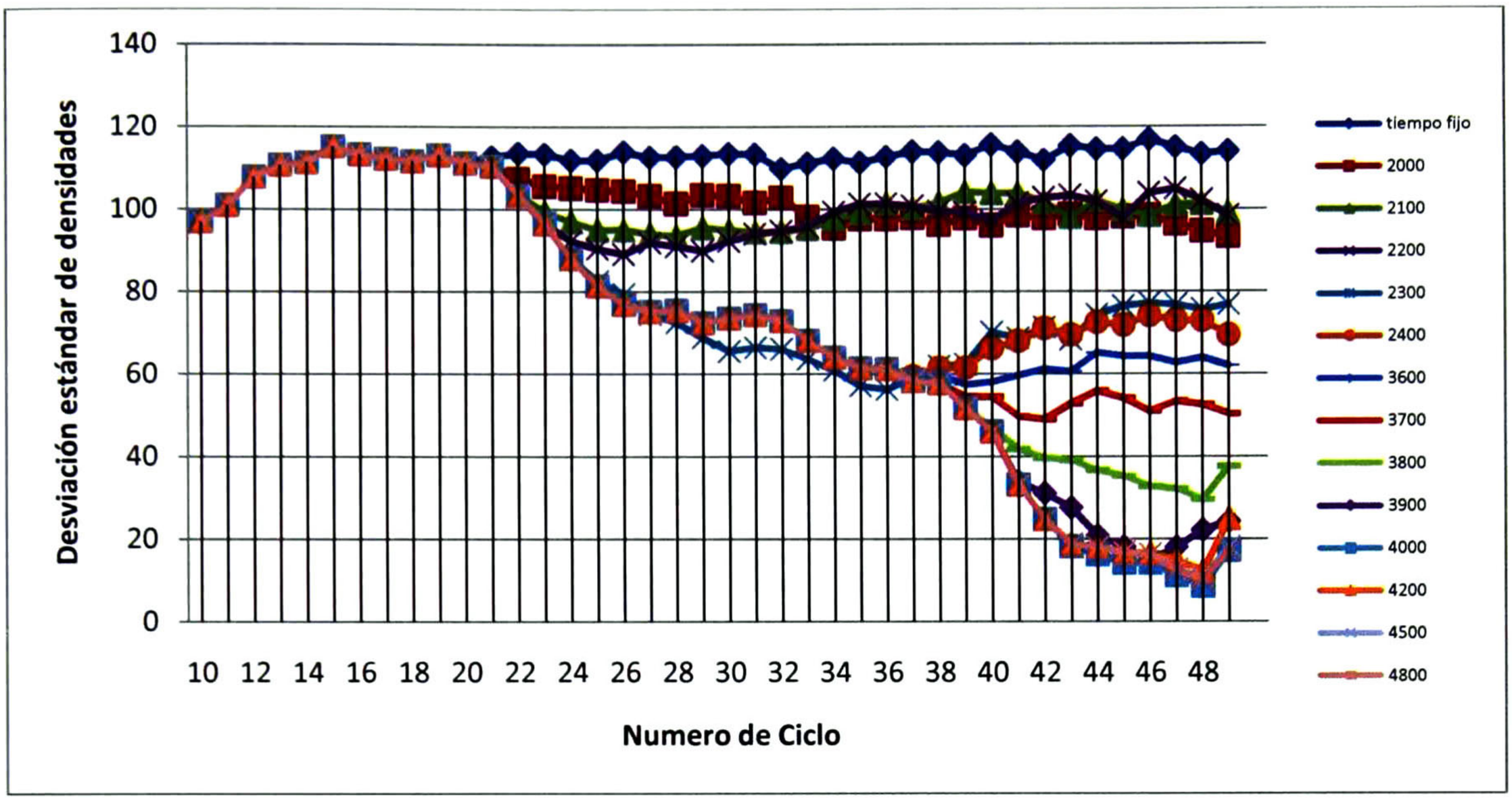
- Tiempos de fase fijos hasta el ciclo 21, donde se aplicaron tiempos optimizados de acuerdo al ciclo anterior y en el ciclo 22 y 23 se aplicaron tiempos optimizados de acuerdo al ciclo anterior correspondiente. Los tiempos calculados para el ciclo 23 persistieron en el resto de la simulación. (Etiqueta *2200*).

- De manera similar las condiciones restantes se fueron simulando, hasta llegar al punto en que del instante 2100 al 4900 cada ciclo aplique tiempos optimizados de acuerdo a la fase anterior (línea color rosa).

La comparativa ilustrada en la gráfica 4.3 tuvo como objetivo mostrar que a medida que el sistema va implementando los tiempos optimizados de una forma más constante se consigue disminuir la desviación estándar de las densidades de los segmentos de camino.

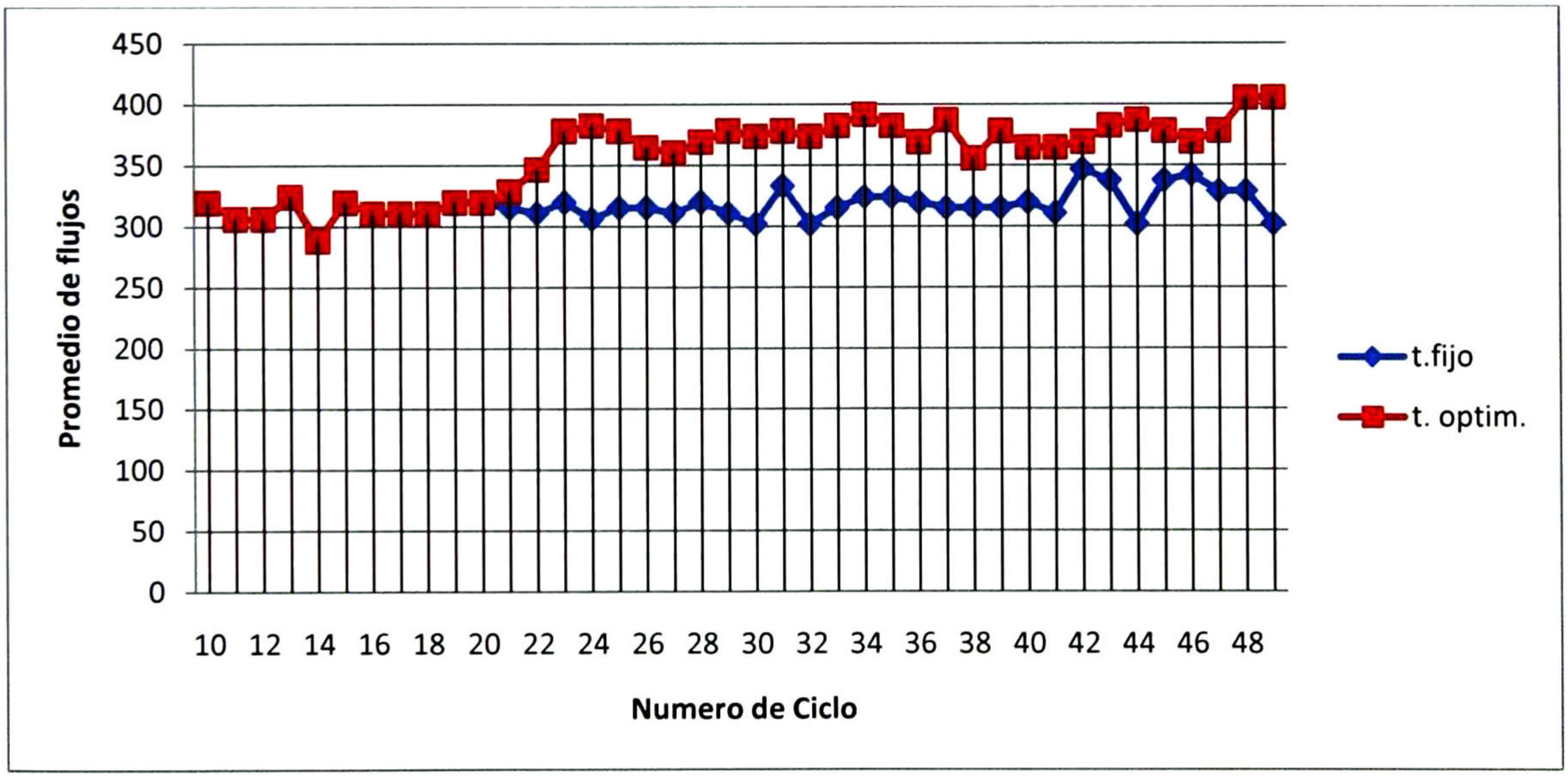
Las gráficas que ilustran la desviación estándar en todos los escenarios de una intersección poseen una organización similar a lo explicado para la gráfica 4.3.

Gráfica 4. 3 Comparativa de desviación estándar en simulación con el escenario 1 y t_max=5



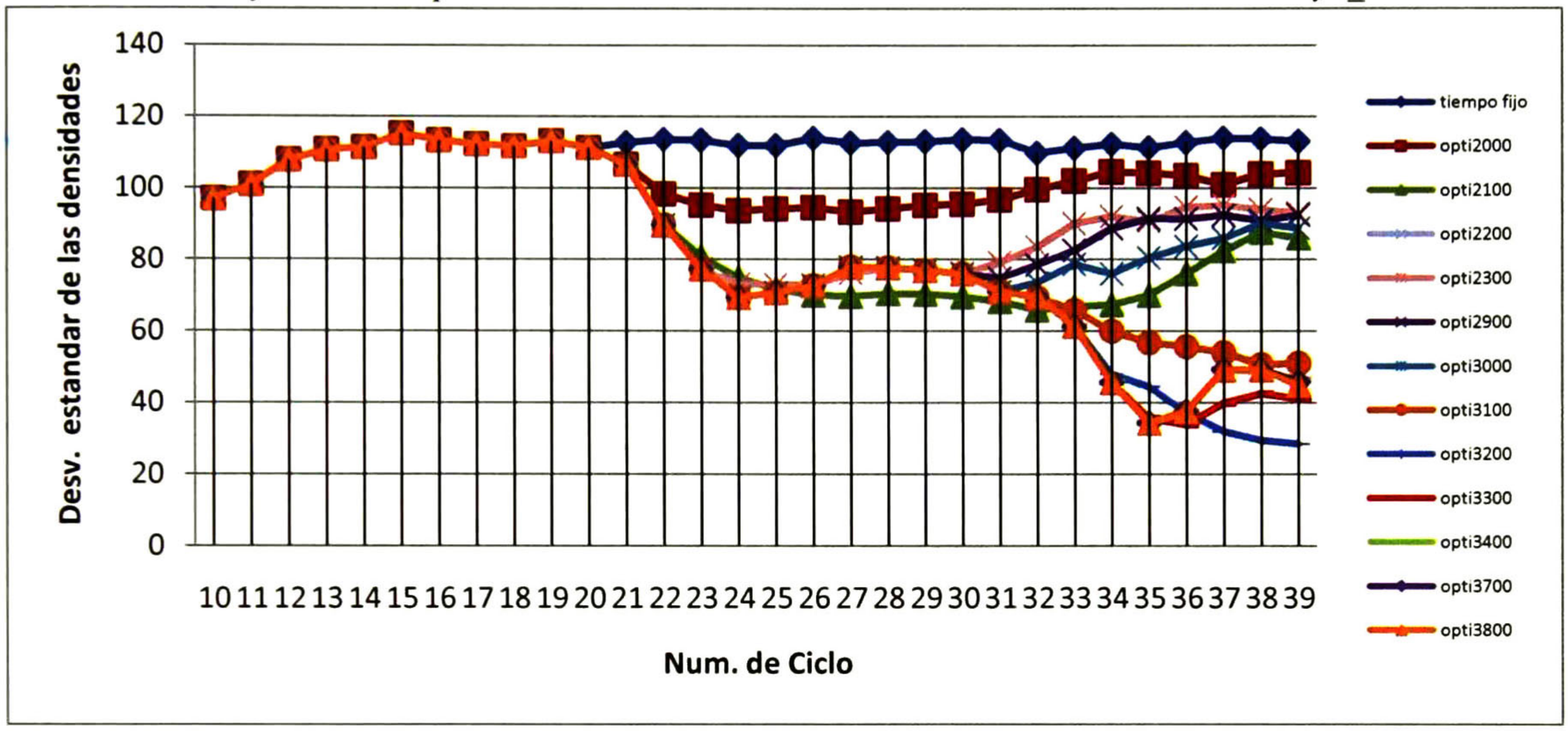
En la gráfica 4.4 se observa que el promedio de flujo de vehículos en el punto de salida de los segmentos de calle es mayor con la implementación del sistema de control propuesto.

Gráfica 4. 4 Comparativa de promedio de flujos en simulación con el escenario 1 y t_max=5

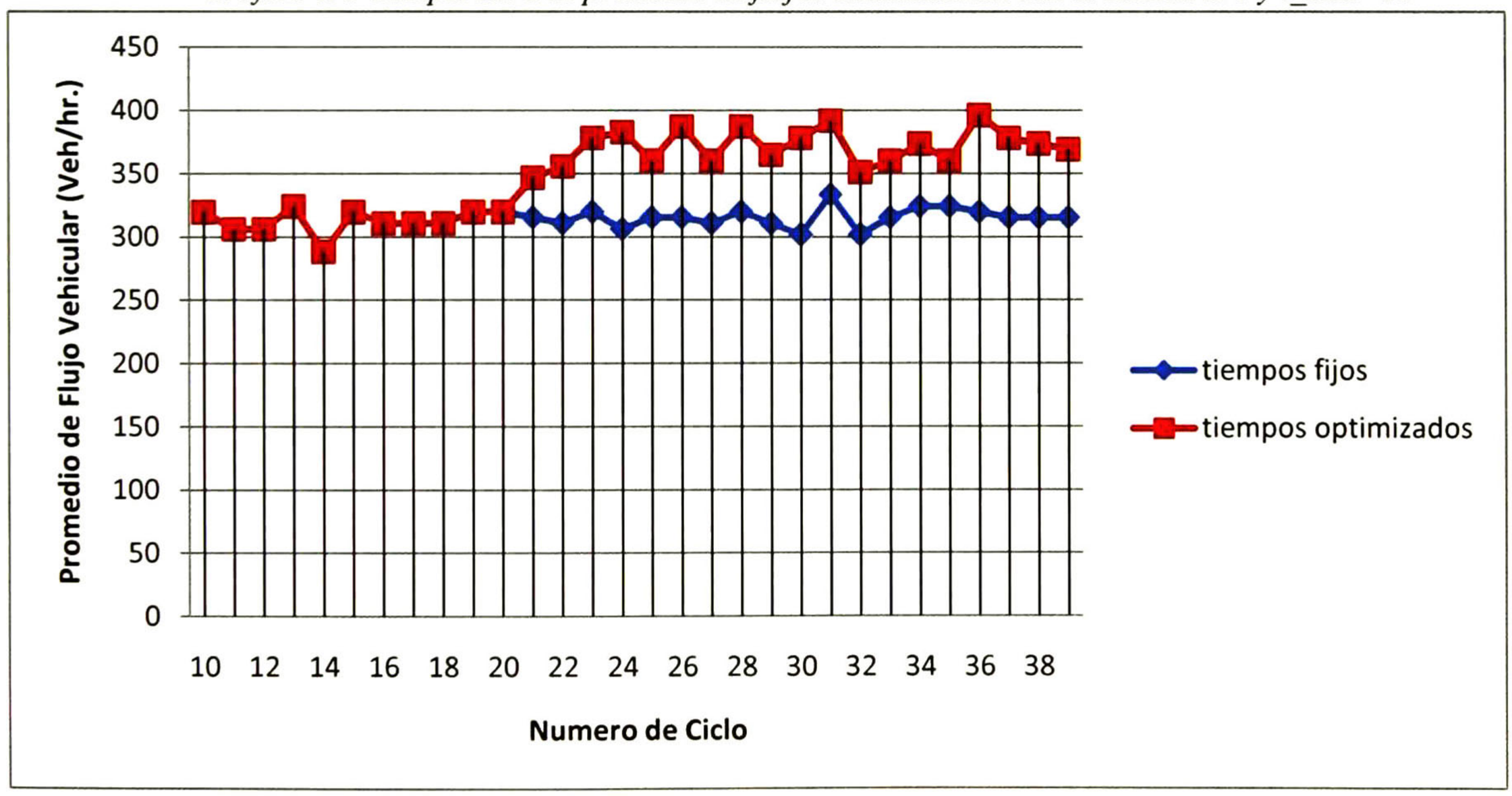


En las gráficas 4.5 y 4.6 se observan los resultados obtenidos nuevamente con el escenario 1, pero estableciendo el valor t_max=10. En cuanto al promedio de flujos, se observa que la mejora que se obtiene es similar con t_max=5 y t_max=10, pero al observar la desviación estándar la diferencia que se encuentra es que ésta se logra bajar en menos ciclos cuando t_max tiene valor de 10. Por ejemplo en la gráfica 4.3 se observa que 4 ciclos después de que inicia el sistema de control de tráfico la desviación estándar esta sobre 80, mientras que en la gráfica 4.5 está debajo de 80.

Gráfica 4. 5 Comparativa de desviación estándar en simulación con el escenario 1 y t_max=10

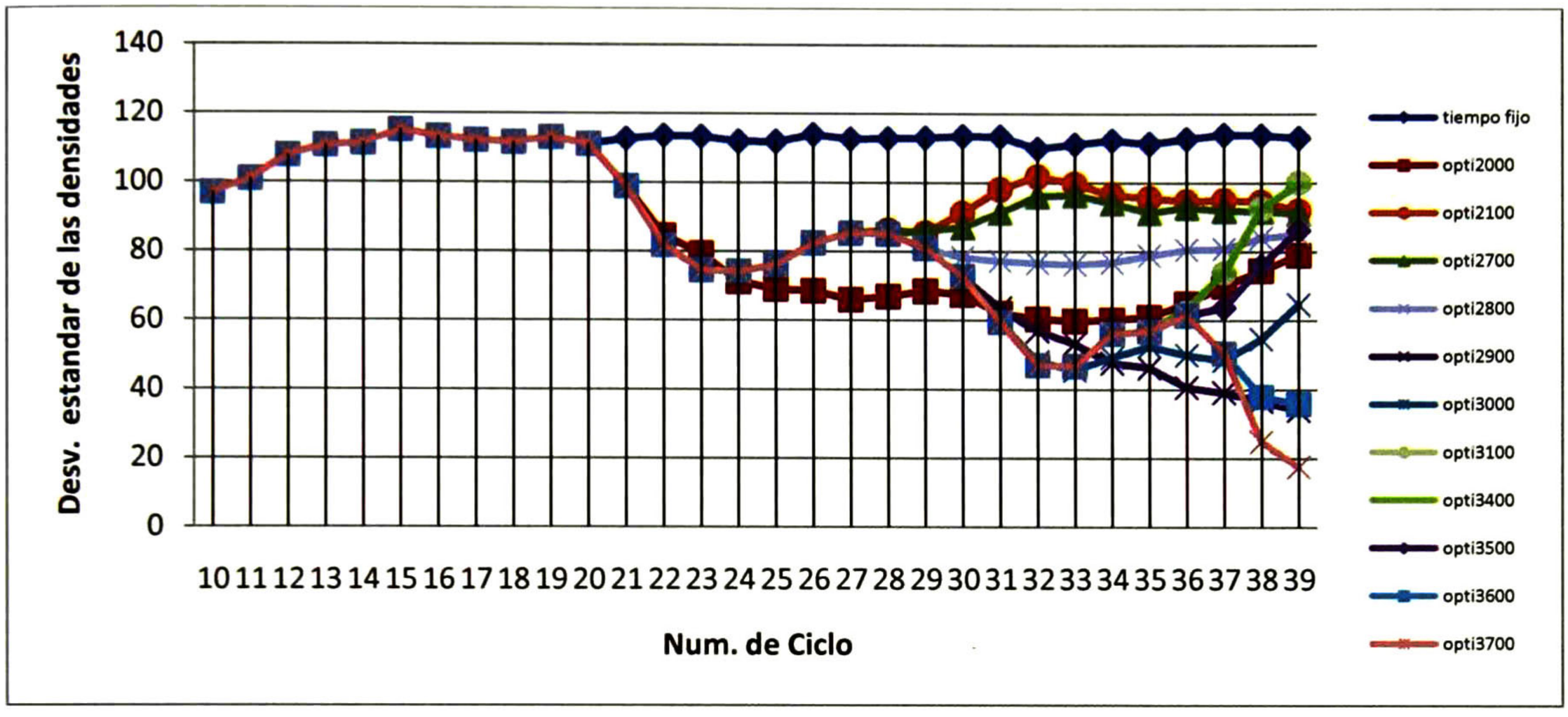


Gráfica 4. 6 Comparativa de promedio de flujos en simulación con el escenario 1 y t_max=10

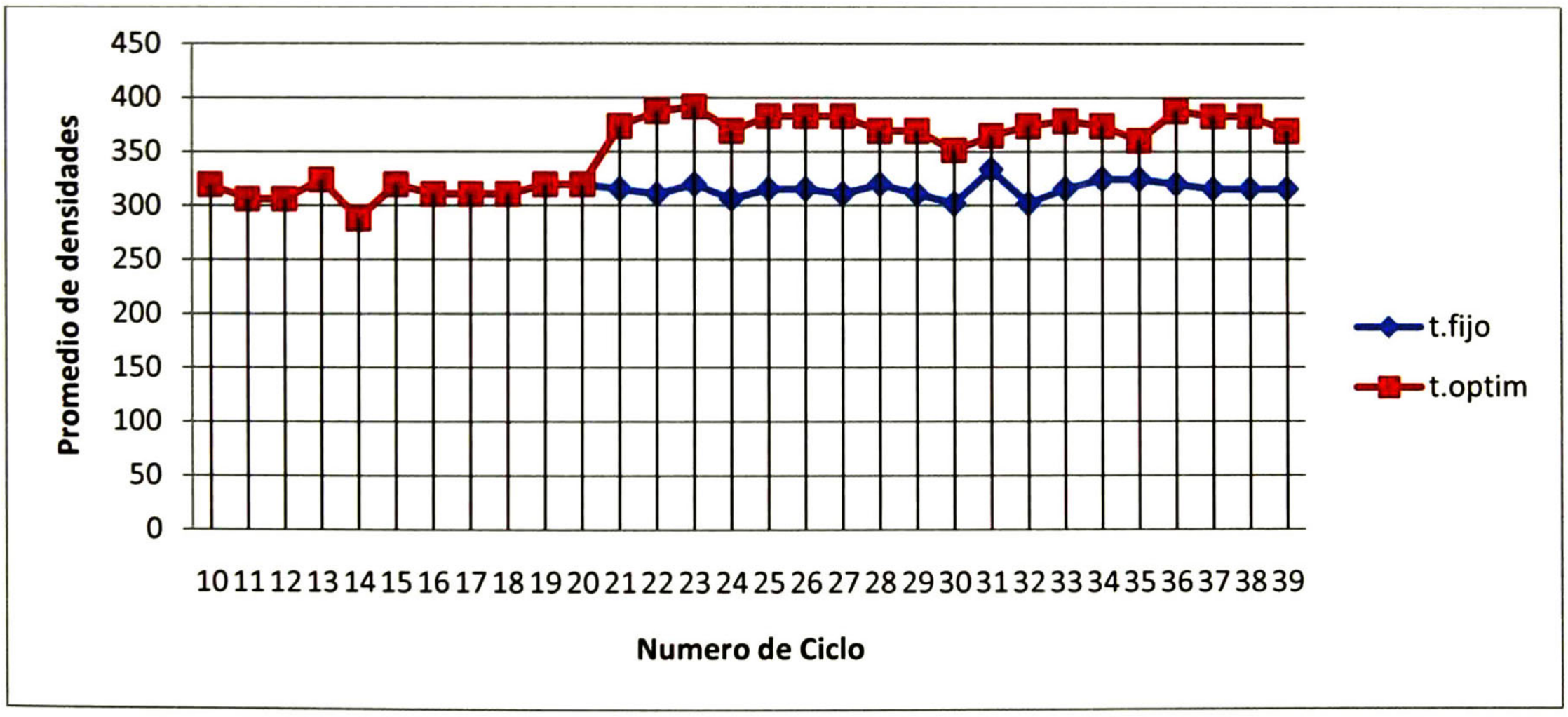


En las gráficas 4.7 y 4.8 se observan los resultados obtenidos nuevamente con el escenario 1, pero estableciendo el valor t_max=20, al igual que en la gráfica 4.5, se observa que 4 ciclos después de que inicia el sistema de control de tráfico la desviación estándar está debajo de 80.

Gráfica 4. 7 Comparativa de desviación estándar en simulación con el escenario 1 y t_max=20



Gráfica 4. 8 Comparativa de promedio de flujos en simulación con el escenario 1 y t_max=20



Después de observar estos resultados parciales se concluyó que el tomar el valor más amplio de los *t_max* mencionados le permite al sistema de control llegar a un mejor estado de tráfico en una menor cantidad de ciclos; Por lo que en el resto de los escenarios de prueba se utilizó *t_max*=20.

Escenario 2

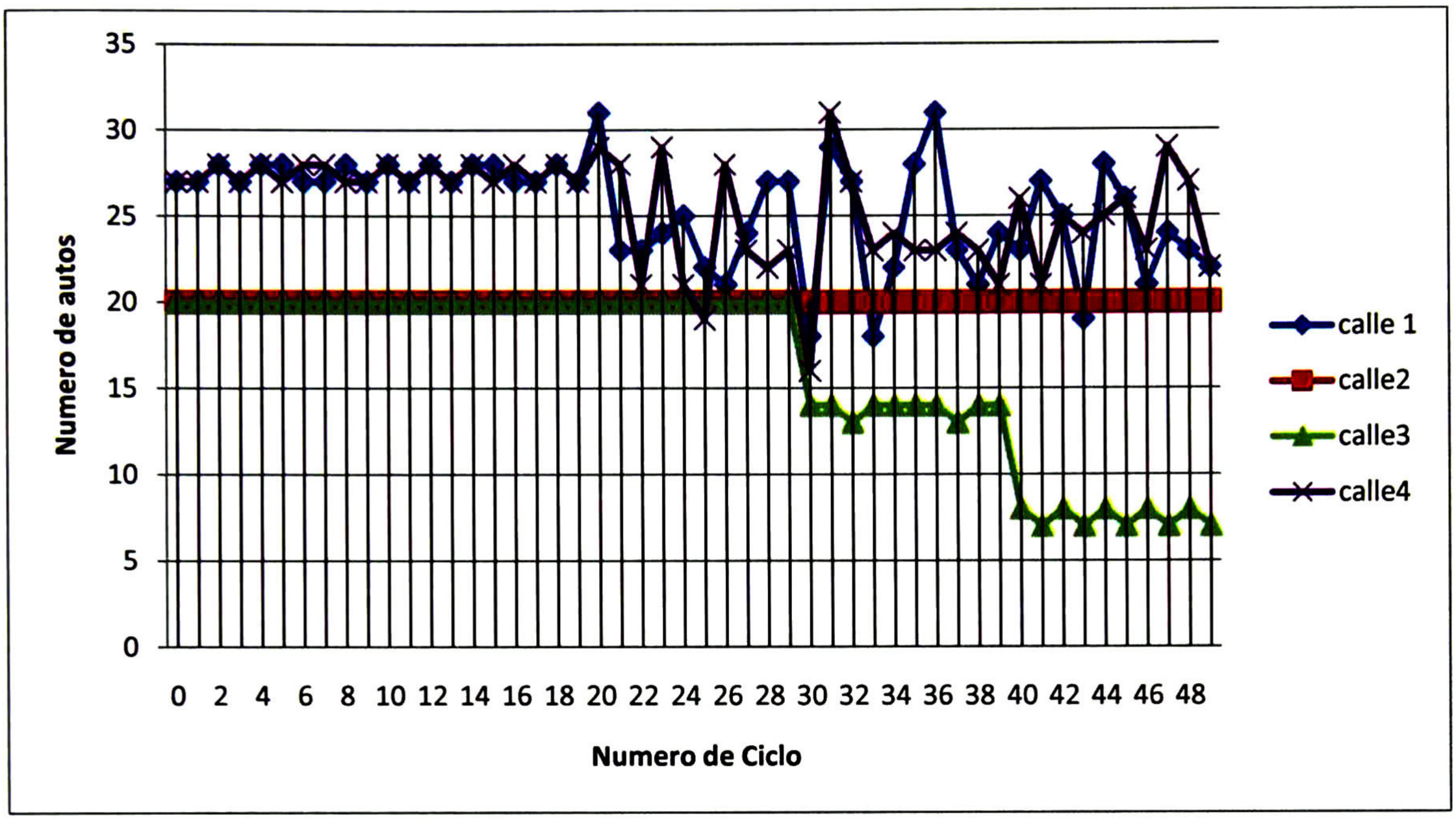
En la tabla 4.2 se muestran las tasas de llegada que se indicaron al simulador para el escenario 2, en este escenario se simuló un comportamiento de tráfico donde dos calles de la intersección se encuentran medianamente saturadas, y las otras dos calles poseen un poco más de saturación de tráfico.

	Número de autos distribuidos uniformemente en el lapso de tiempo.			
Tiempo	**Calle 1**	**Calle 2**	**Calle 3**	**Calle 4**
0-2,999 seg.	825	600	600	825
3,000-3,999 seg.	325	200	138	275
4,000-4999 seg.	375	200	75	275

Tabla 4. 2 Tasas de entrada de vehículos para escenario 2

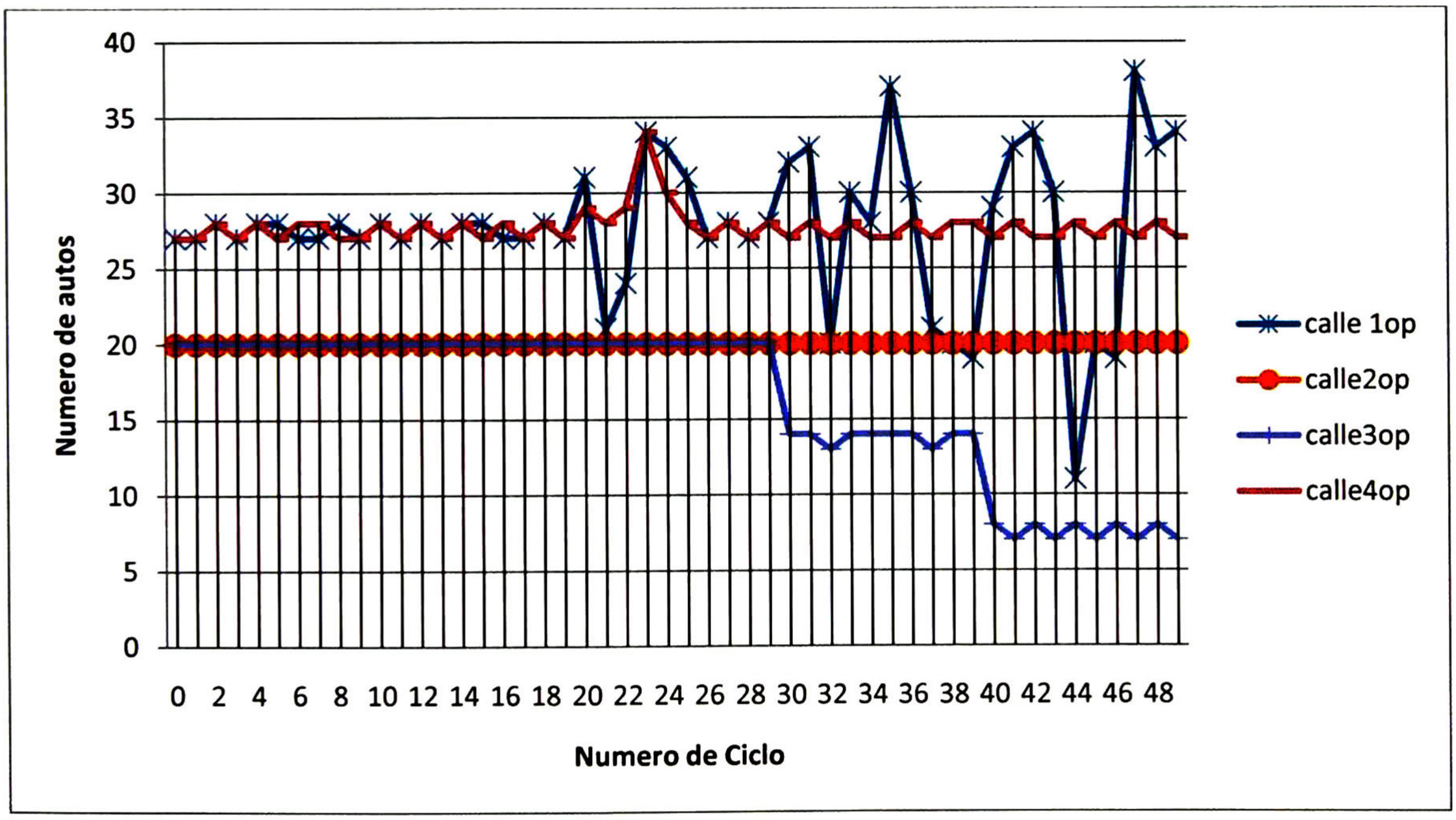
En la gráfica 4.9 se muestra la entrada de vehículos registrada en la simulación en el esquema de tiempos fijos.

Gráfica 4. 9 Entrada de vehículos bajo escenario 2 con tiempos de fase fijos



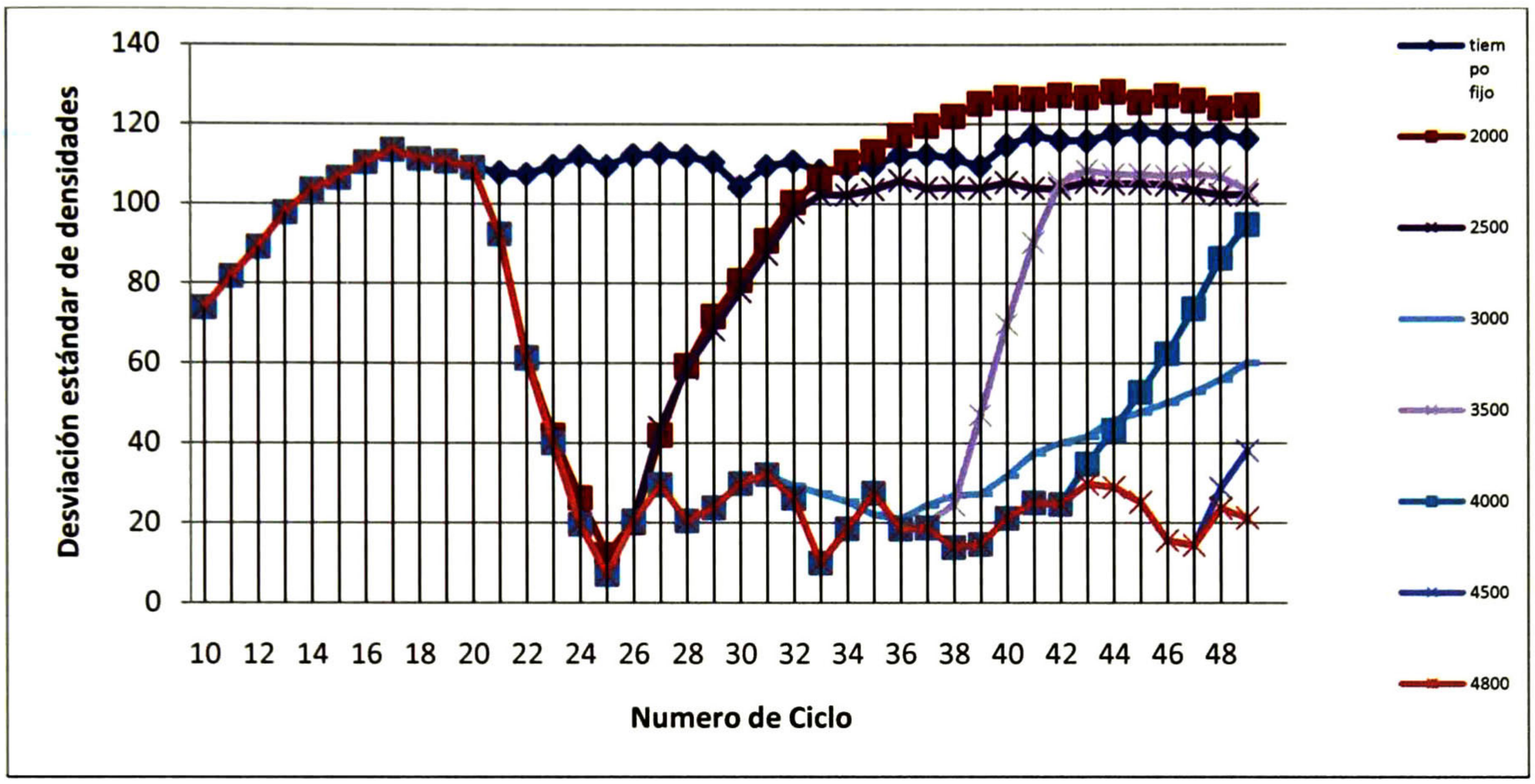
En la gráfica 4.10 se muestra la entrada de vehículos registrada en la simulación bajo el esquema de tiempos óptimos.

Gráfica 4. 10 Entrada de vehículos bajo escenario 2 con tiempos de fase optimizados



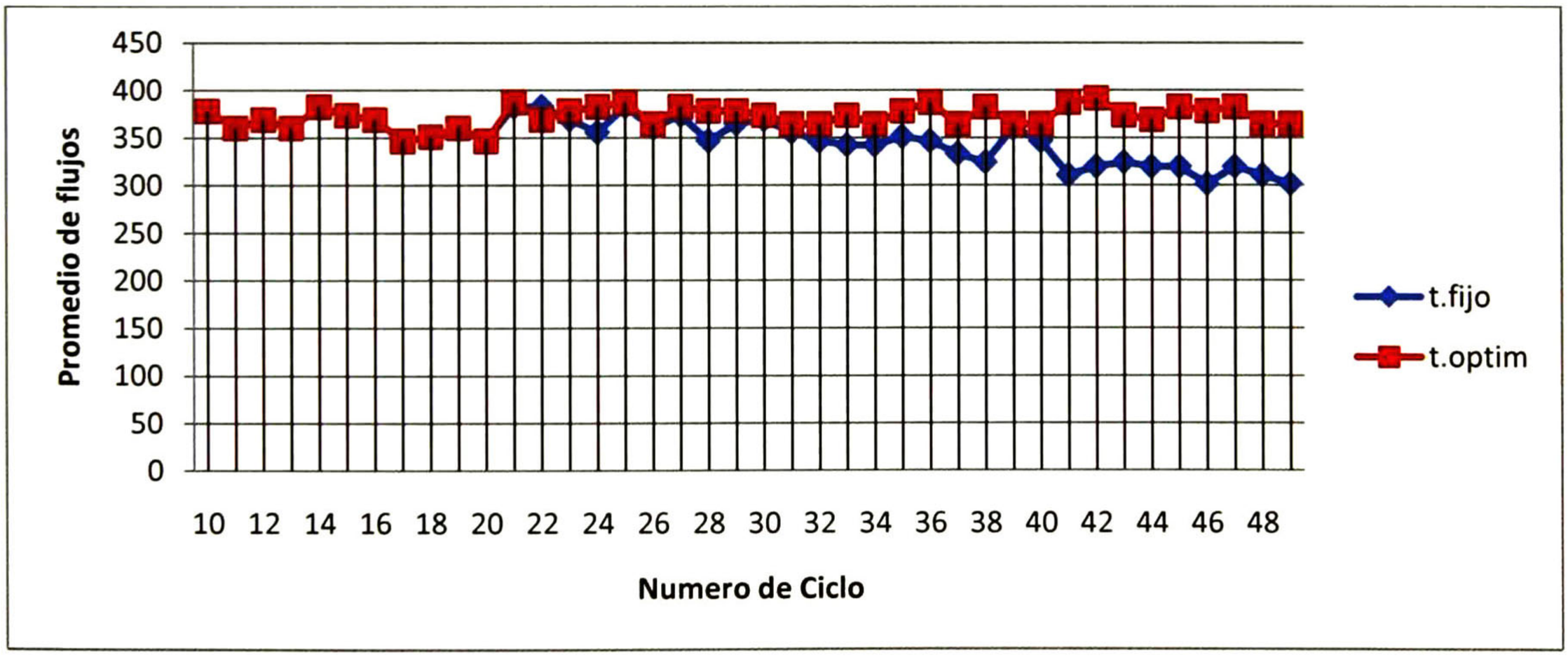
En la gráfica 4.11 se muestra la comparativa de la desviación estándar de las densidades de tráfico en las calles respecto al escenario 2, se observa que la desviación estándar con tiempos óptimos (línea color rosa) es mucho mejor respecto a los tiempos de fase fijos (línea azul).

Gráfica 4. 11 Comparativa de desviación estándar en simulación con el escenario 2



En la gráfica 4.12 se muestra la comparativa del flujo de vehículos en las calles bajo el escenario 2, se observa que el flujo de vehículos es mayor bajo las fases cuyo tiempo fue calculado por el sistema de control.

Gráfica 4. 12 Comparativa de promedio de flujos en simulación con el escenario 2



Escenario 3

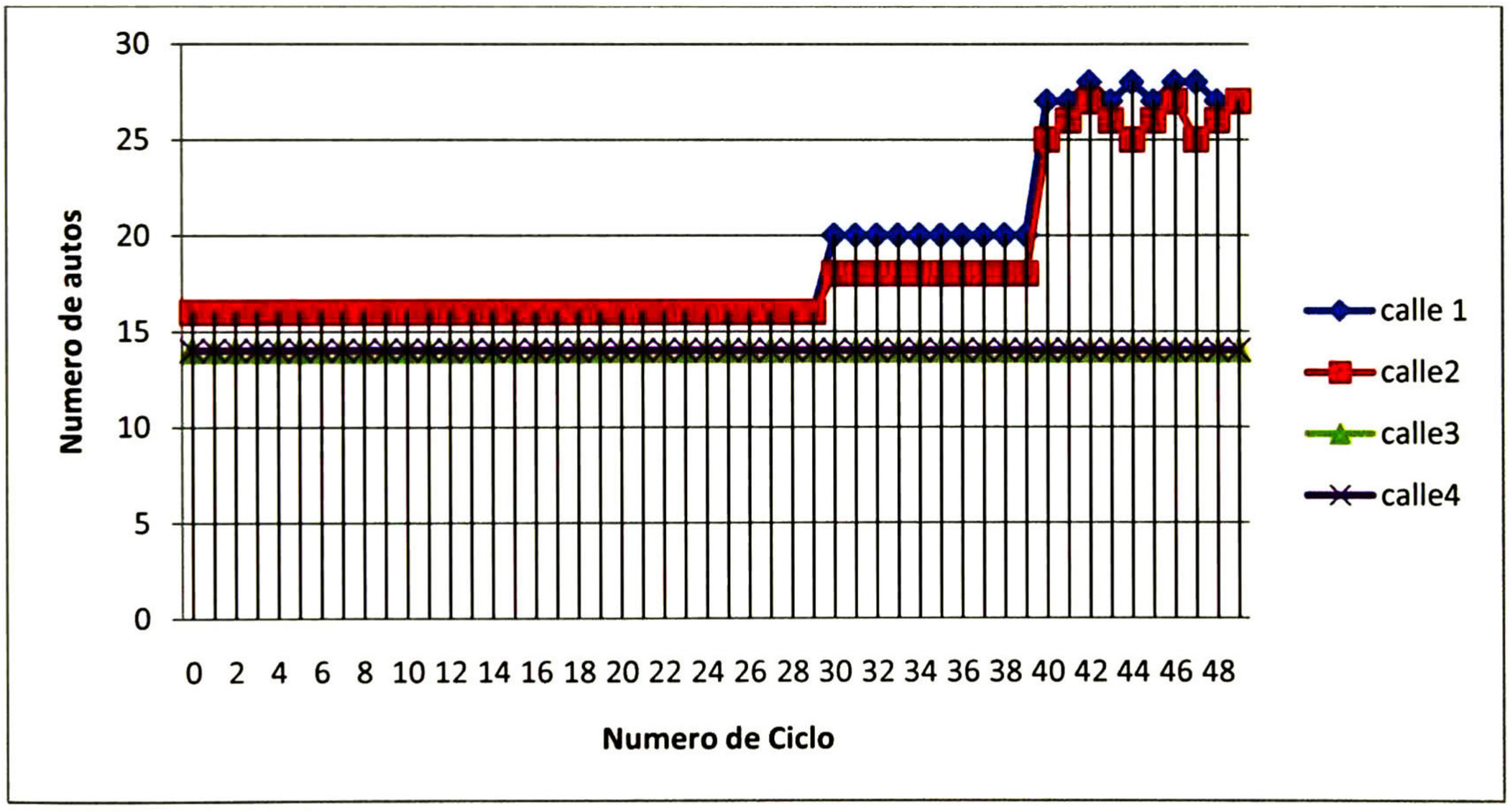
En la tabla 4.3 se muestran las tasas de llegada que se indicaron al simulador para el escenario 3, en este escenario se simuló un comportamiento de tráfico en donde las cuatro calles inician la simulación poco saturadas, pero dos de ellas comienzan a incrementar la densidad de tráfico a la mitad de la simulación.

	Número de autos distribuidos uniformemente en el lapso de tiempo.			
Tiempo	**Calle 1**	**Calle 2**	**Calle 3**	**Calle 4**
0-2,999 seg.	480	480	420	420
3,000-3,999 seg.	200	180	140	140
4,000-4999 seg.	275	260	140	140

Tabla 4. 3 Tasas de entrada de vehículos para escenario 3

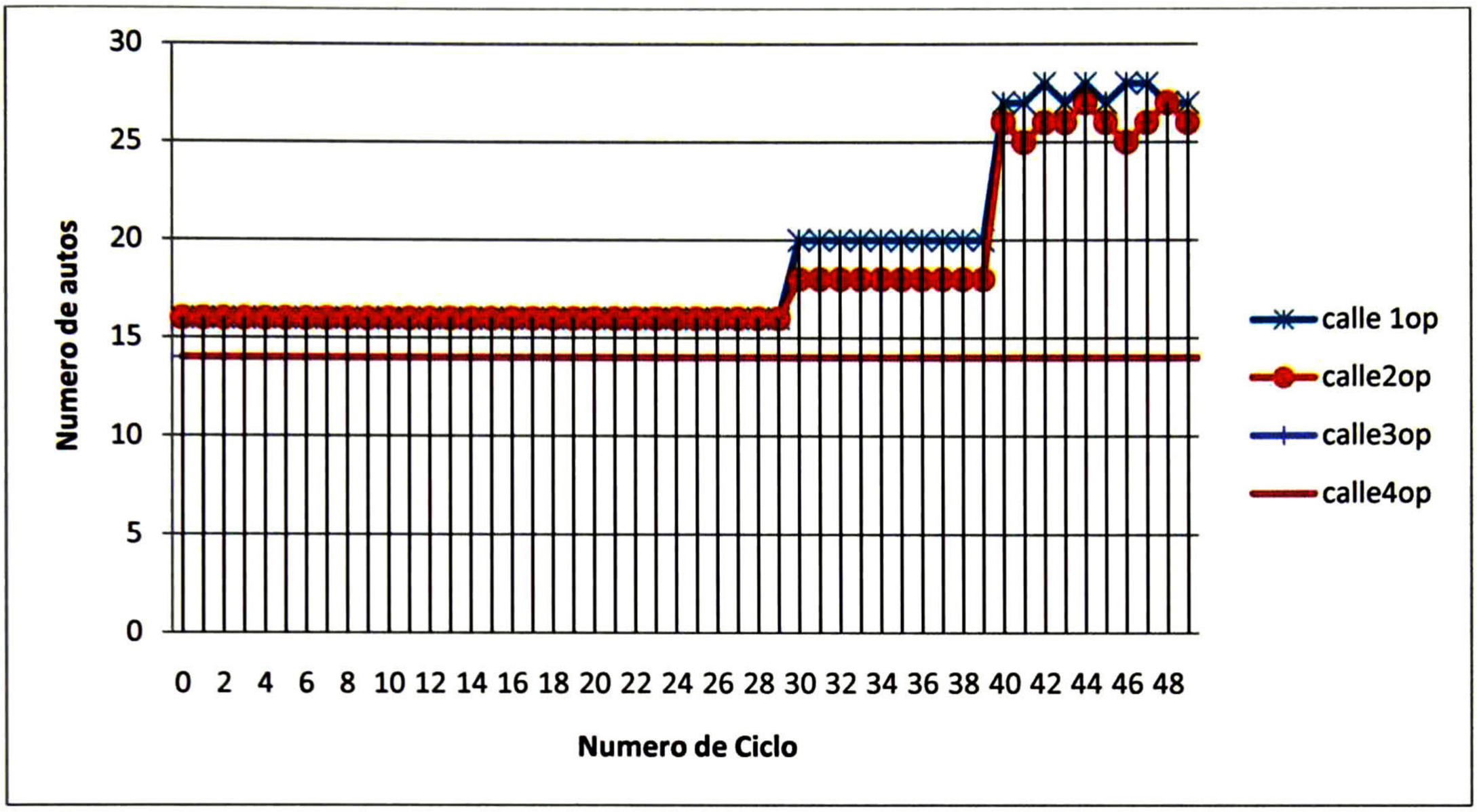
En la gráfica 4.13 se muestra la entrada de vehículos registrada en la simulación en el esquema de tiempos fijos bajo el escenario 3.

Gráfica 4. 13 Entrada de vehículos bajo escenario 3 con tiempos de fase fijos



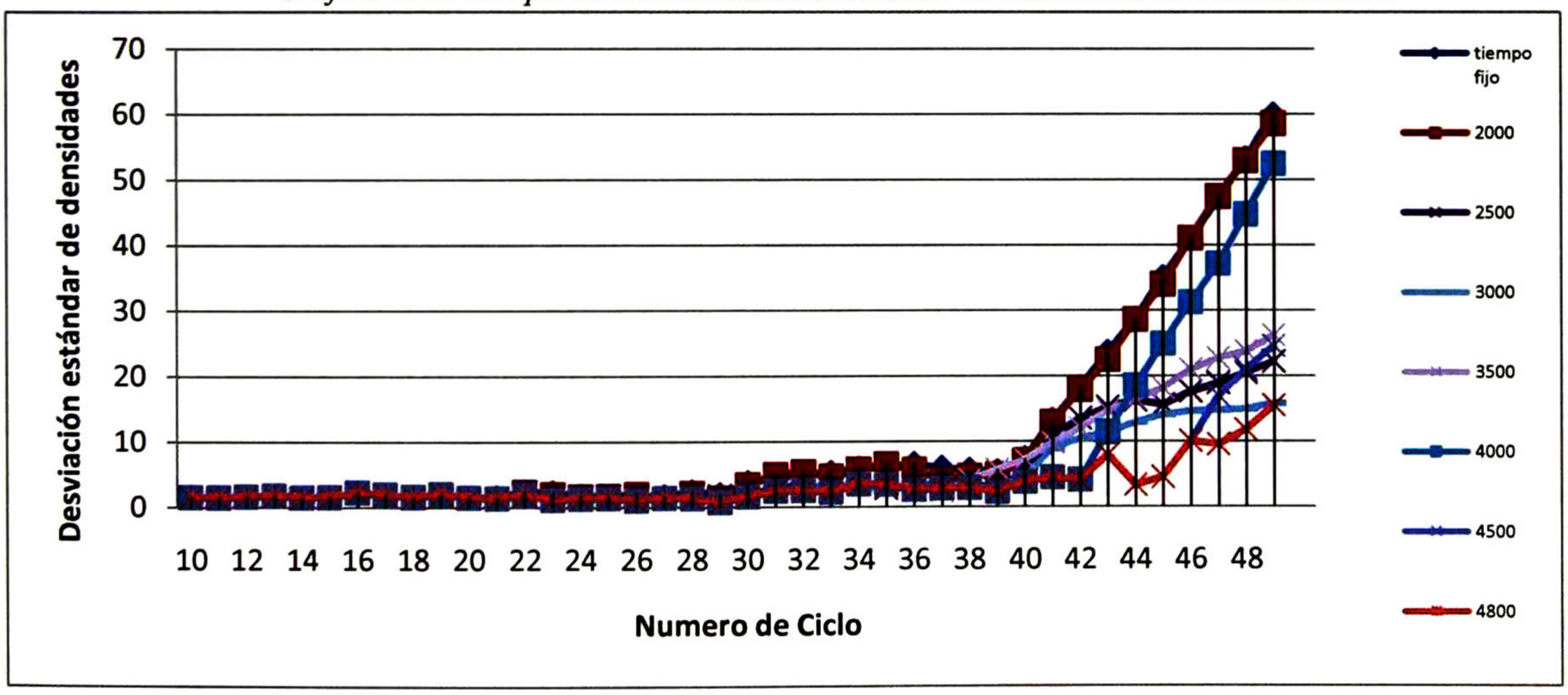
En la gráfica 4.14 se muestra la entrada de vehículos registrada en la simulación en el esquema de tiempos óptimos bajo el escenario 3.

Gráfica 4.14 Entrada de vehiculos bajo escenario 3 con tiempos de fase optimizados



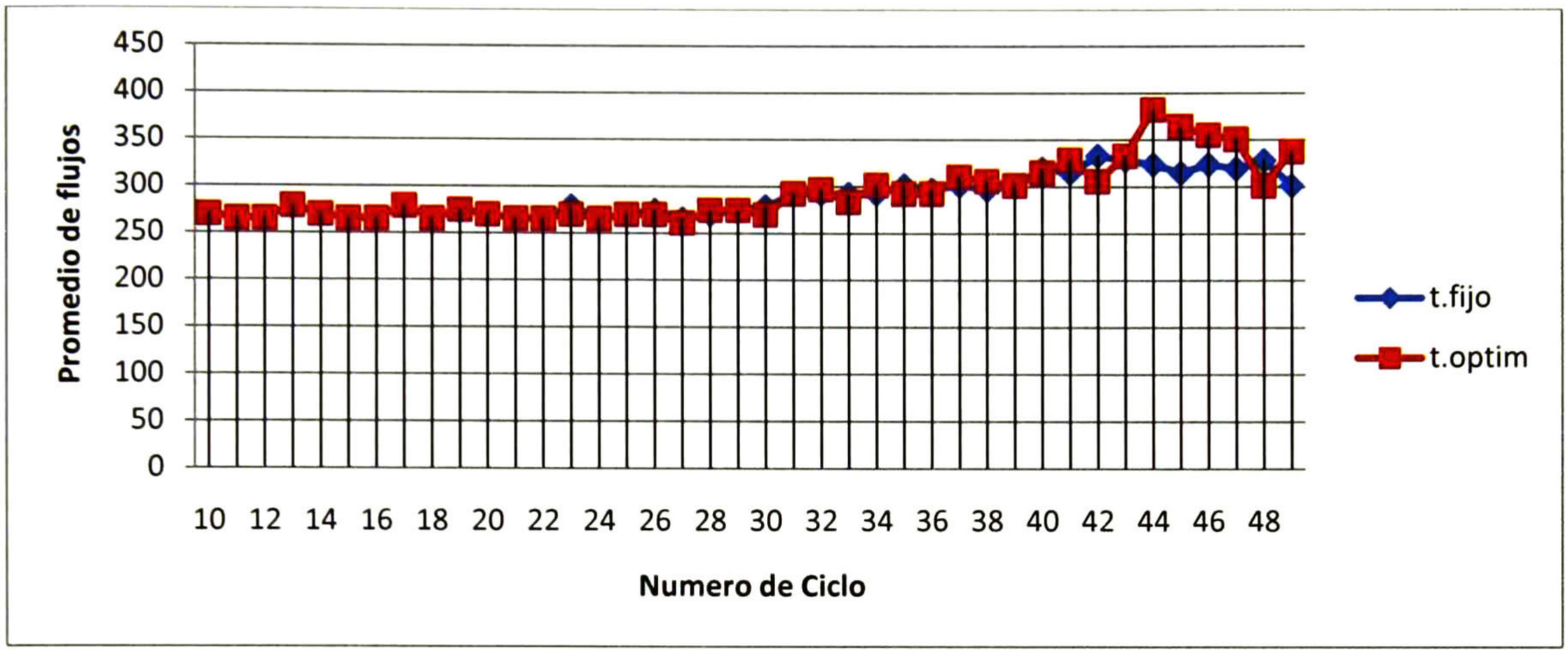
En la gráfica 4.15 se muestra la comparativa de la desviación estándar de las densidades de tráfico en las calles respecto al escenario 3, se observa que mientras el flujo de tráfico es similar en las cuatro calles tanto el tiempo fijo como los tiempos óptimos mantienen baja la desviación estándar, pero a partir del ciclo 40, cuando el flujo aumenta en algunas calles, los tiempos óptimos logran mantener baja la desviación estándar mientras que los tiempos fijos la elevan.

Gráfica 4. 15 Comparativa de desviación estándar en simulación con el escenario 3



En la gráfica 4.16 se muestra la comparativa de los flujos de tráfico respecto al escenario 3, se observa que bajo los dos esquemas el flujo de autos es similar, solamente a partir del ciclo 40 el flujo es mayor en algunos ciclos.

Gráfica 4. 16 Comparativa de promedio de flujos en simulación con el escenario 3



Escenario 4

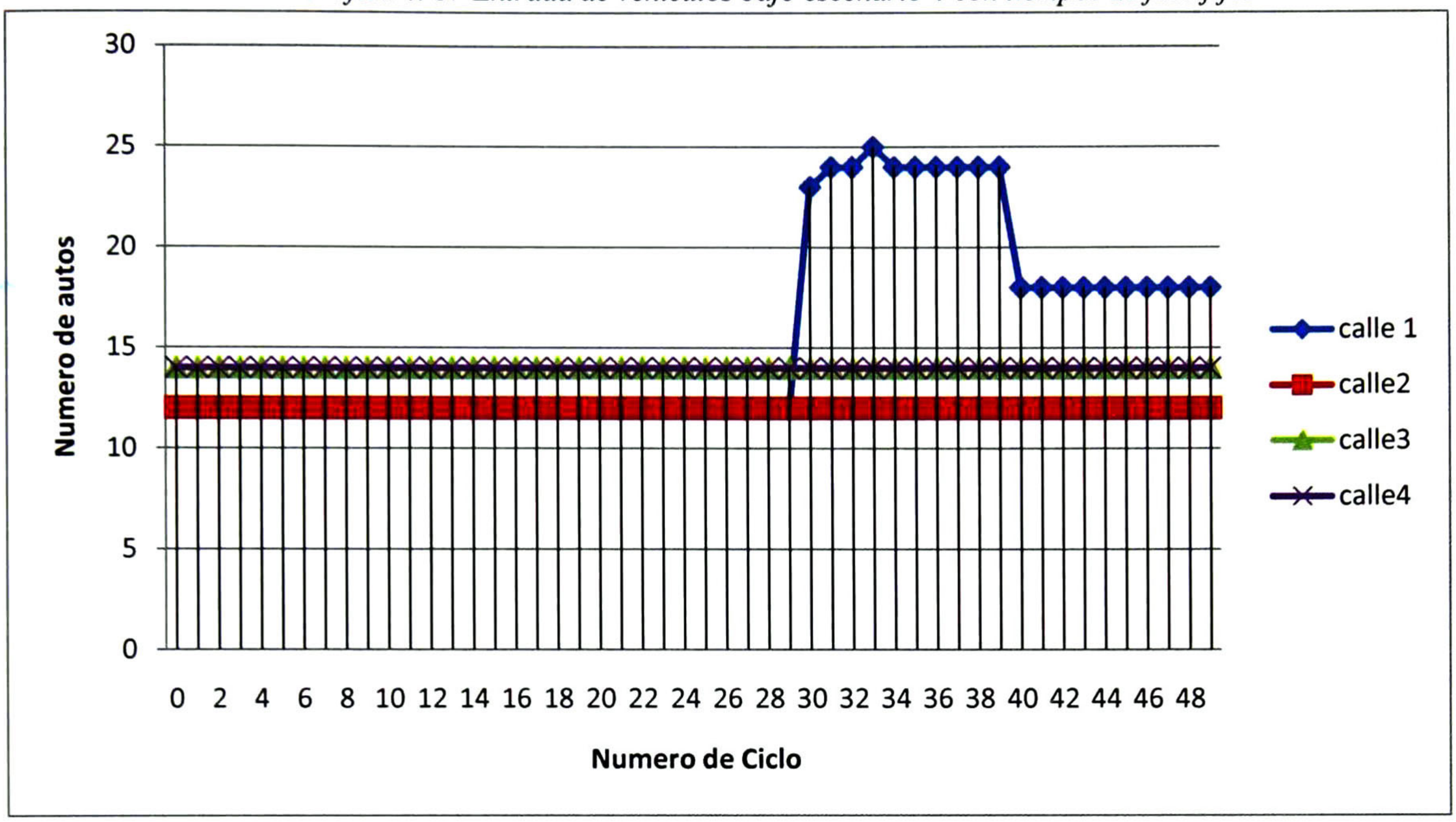
En la tabla 4.4 se muestran las tasas de llegada que se indicaron al simulador para el escenario 4, en este escenario se simuló un comportamiento de tráfico en donde las cuatro calles inician la simulación poco saturadas, pero una de ellas obtiene un aumento notable en la densidad de tráfico y 10 ciclos después recibe un decremento significativo.

	Número de autos distribuidos uniformemente en el lapso de tiempo.			
Tiempo	**Calle 1**	**Calle 2**	**Calle 3**	**Calle 4**
0-2,999 seg.	360	360	420	420
3,000-3,999 seg.	240	120	140	140
4,000-4999 seg.	180	120	140	140

Tabla 4. 4 Tasas de entrada de vehículos para escenario 4

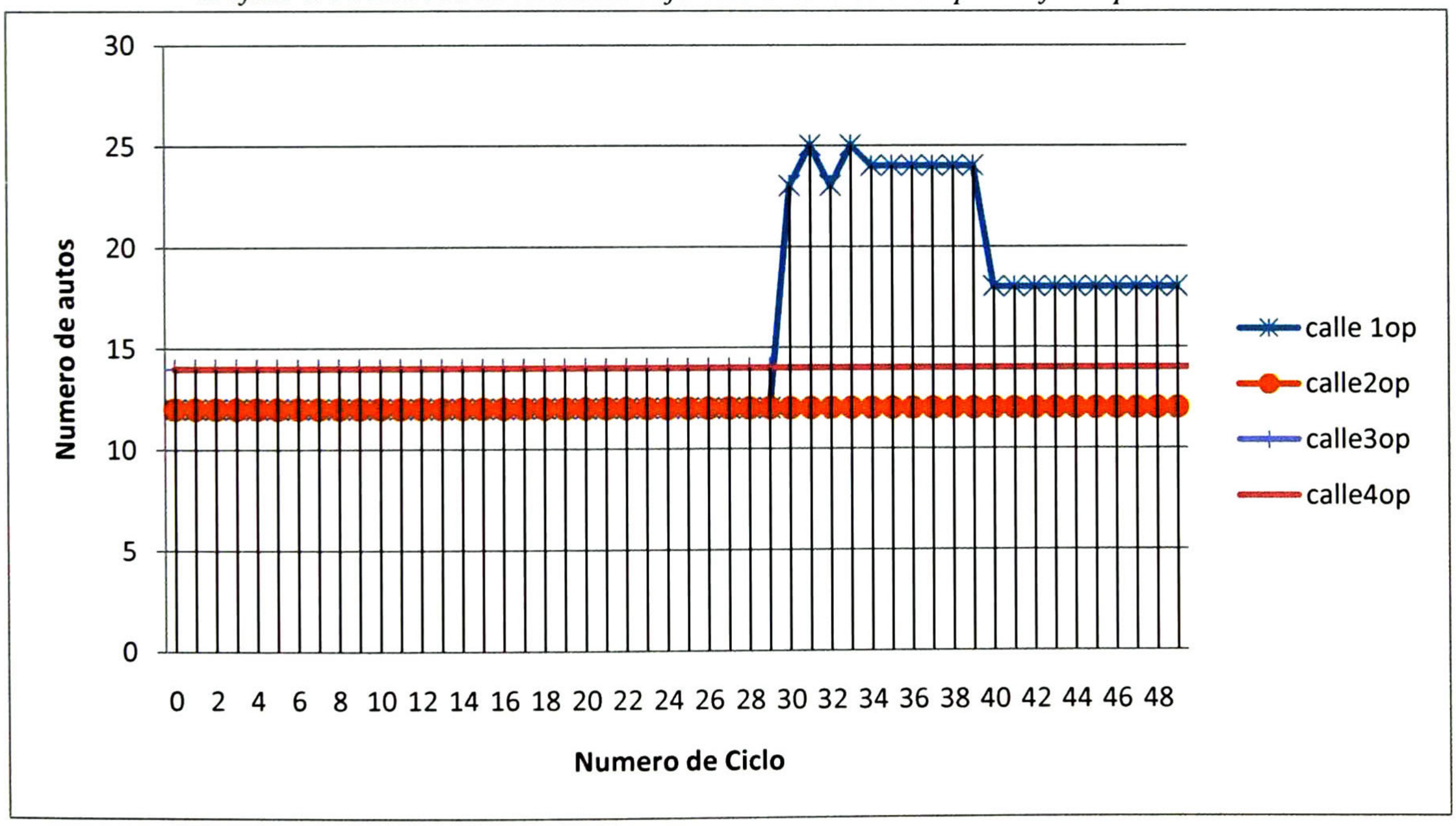
En la gráfica 4.17 se muestra la entrada de vehículos registrada en la simulación en el esquema de tiempos fijos bajo el escenario 4.

Gráfica 4. 17 Entrada de vehiculos bajo escenario 4 con tiempos de fase fijos



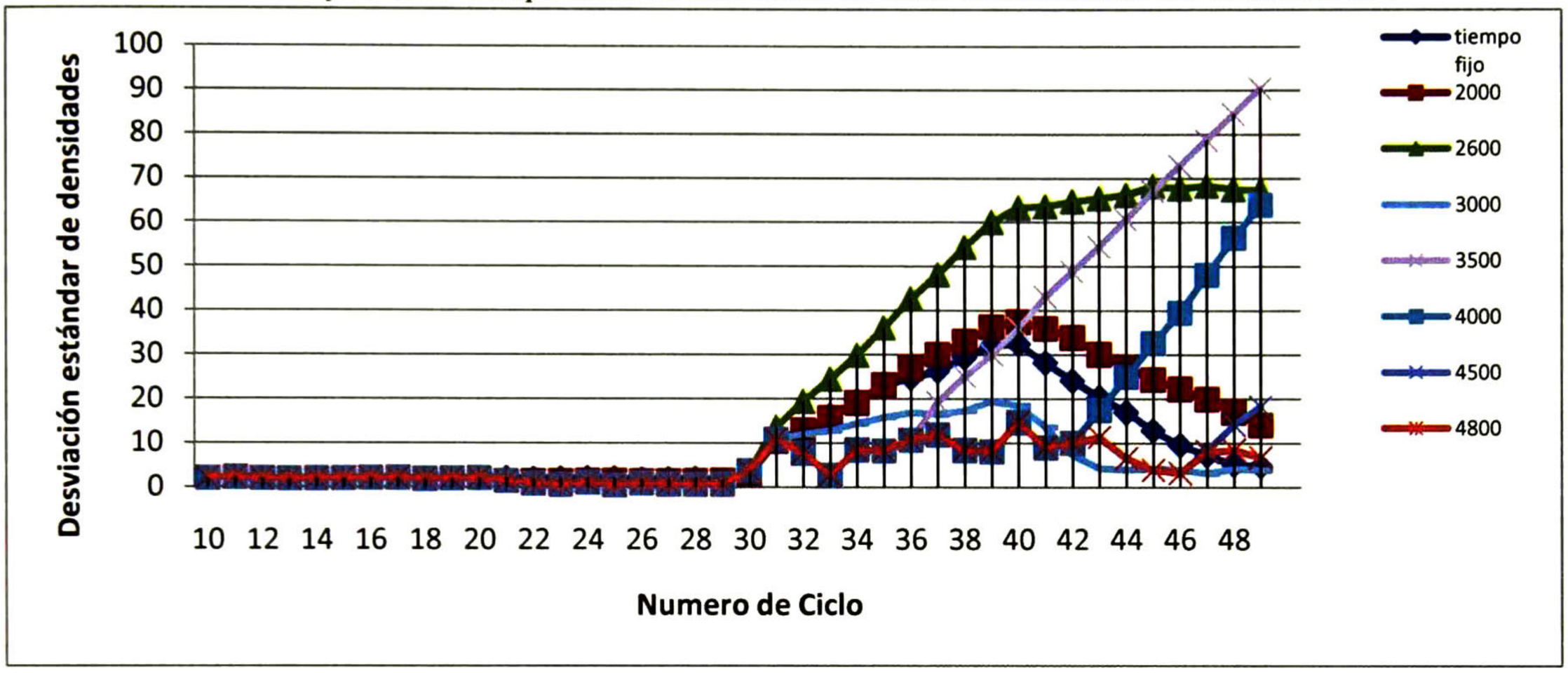
En la gráfica 4.18 se muestra la entrada de vehículos registrada en la simulación en el esquema de tiempos óptimos bajo el escenario 4.

Gráfica 4. 18 Entrada de vehículos bajo escenario 4 con tiempos de fase optimizados



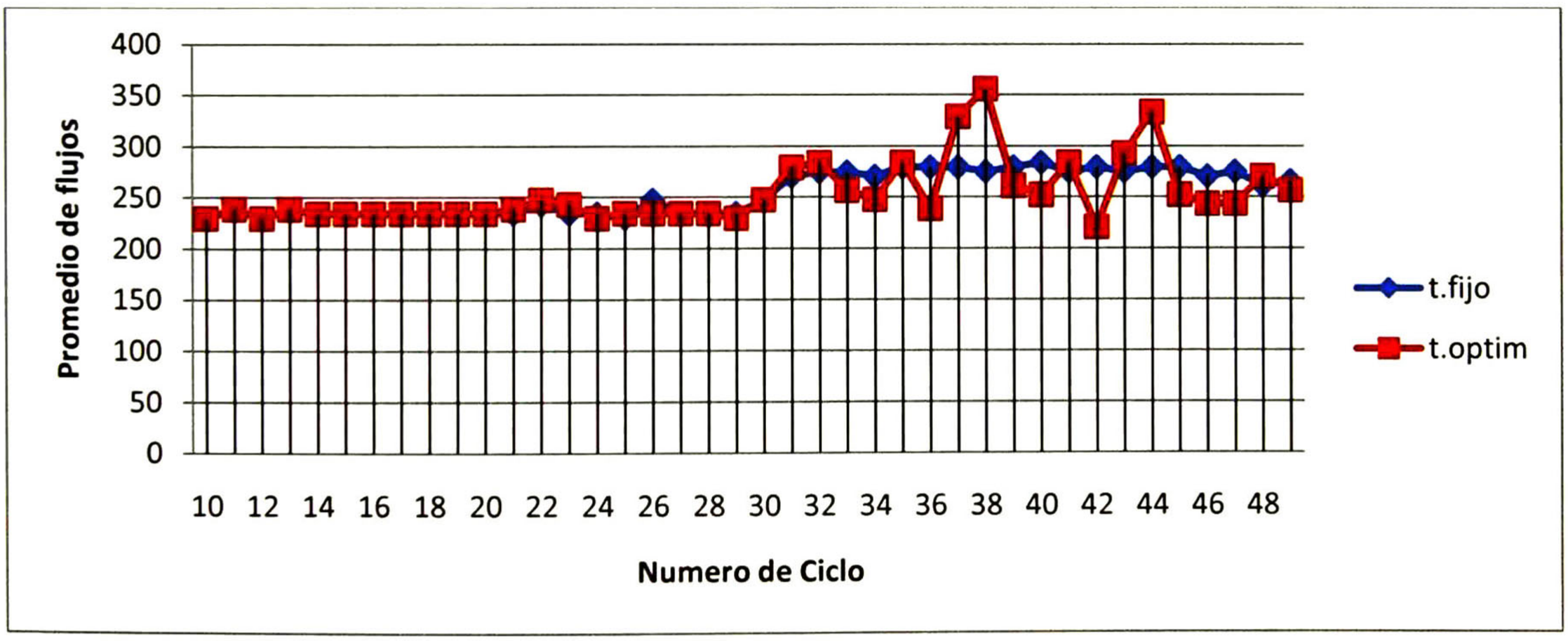
En la gráfica 4.19 se muestra la comparativa de la desviación estándar de las densidades de tráfico en las calles respecto al escenario 4, se observa que mientras el flujo de tráfico es similar en las cuatro calles tanto el tiempo fijo como los tiempos óptimos mantienen baja la desviación estándar (línea color rosa), pero a partir del ciclo 30, cuando el flujo aumenta en una de las calles, los tiempos óptimos logran mantener baja la desviación estándar mientras que los tiempos fijos la elevan (línea color azul).

Gráfica 4. 19 Comparativa de desviación estándar en simulación con el escenario 4



En la gráfica 4.20 se muestra la comparativa de los flujos de tráfico respecto al escenario 4, se observa que bajo los dos esquemas el flujo de autos es similar, solamente a partir del ciclo 30 el flujo es mayor en algunos ciclos.

Gráfica 4. 20 Comparativa de promedio de flujos en simulación con el escenario 4



Escenario 5

En la tabla 4.5 se muestran las tasas de llegada que se indicaron al simulador para el escenario 5, en este escenario se simuló un comportamiento de tráfico en donde dos calles tienen una densidad vehicular elevada, mientras que otra calle tiene densidad menor y en otra calle se incrementa de manera gradual.

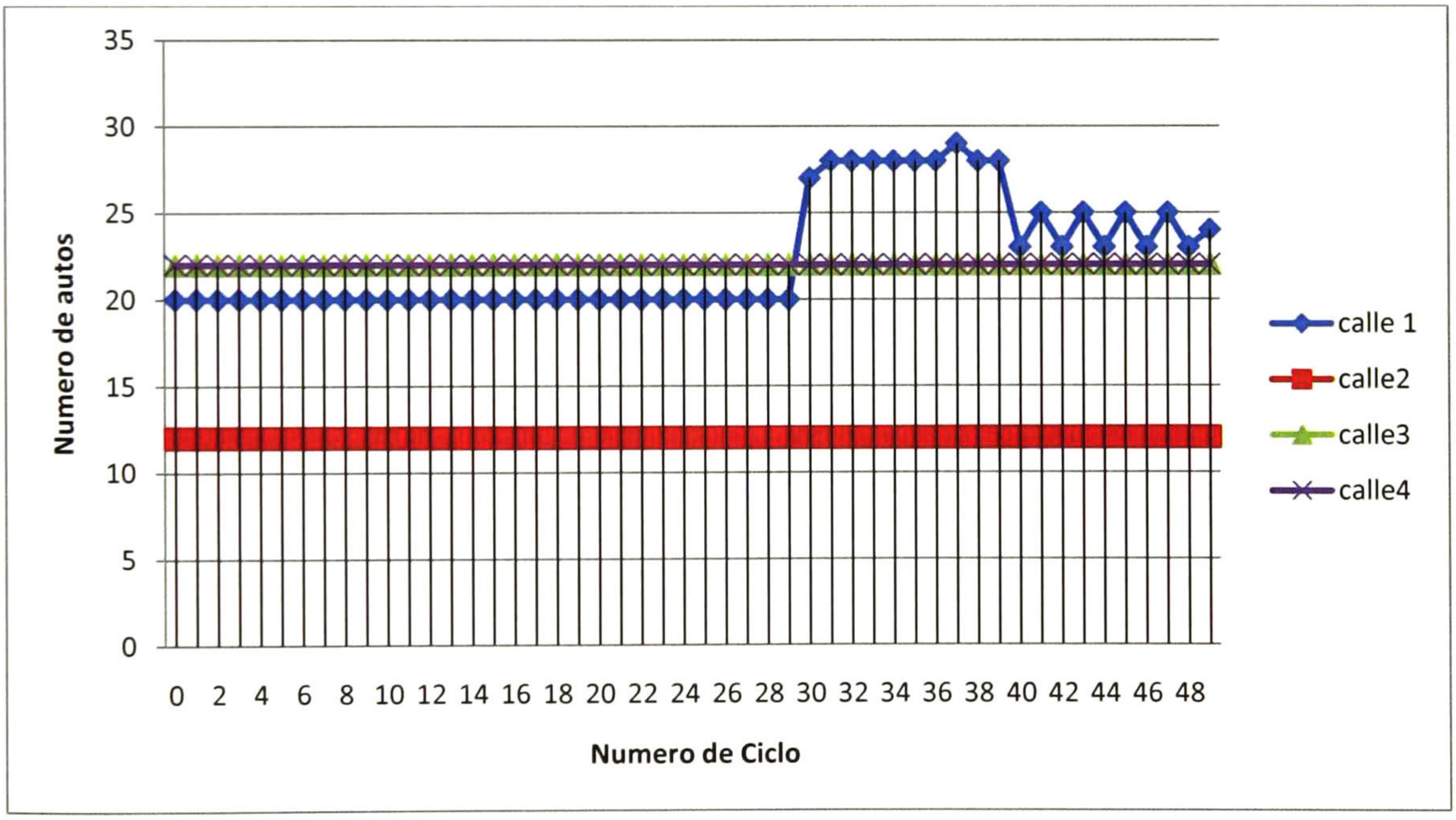
	Número de autos distribuidos uniformemente en el lapso de tiempo.			
Tiempo	**Calle 1**	**Calle 2**	**Calle 3**	**Calle 4**
0-2,999 seg.	600	360	660	660
3,000-3,999 seg.	280	120	220	220
4,000-4999 seg.	240	120	220	220

Tabla 4. 5 Tasas de entrada de vehículos para escenario

5

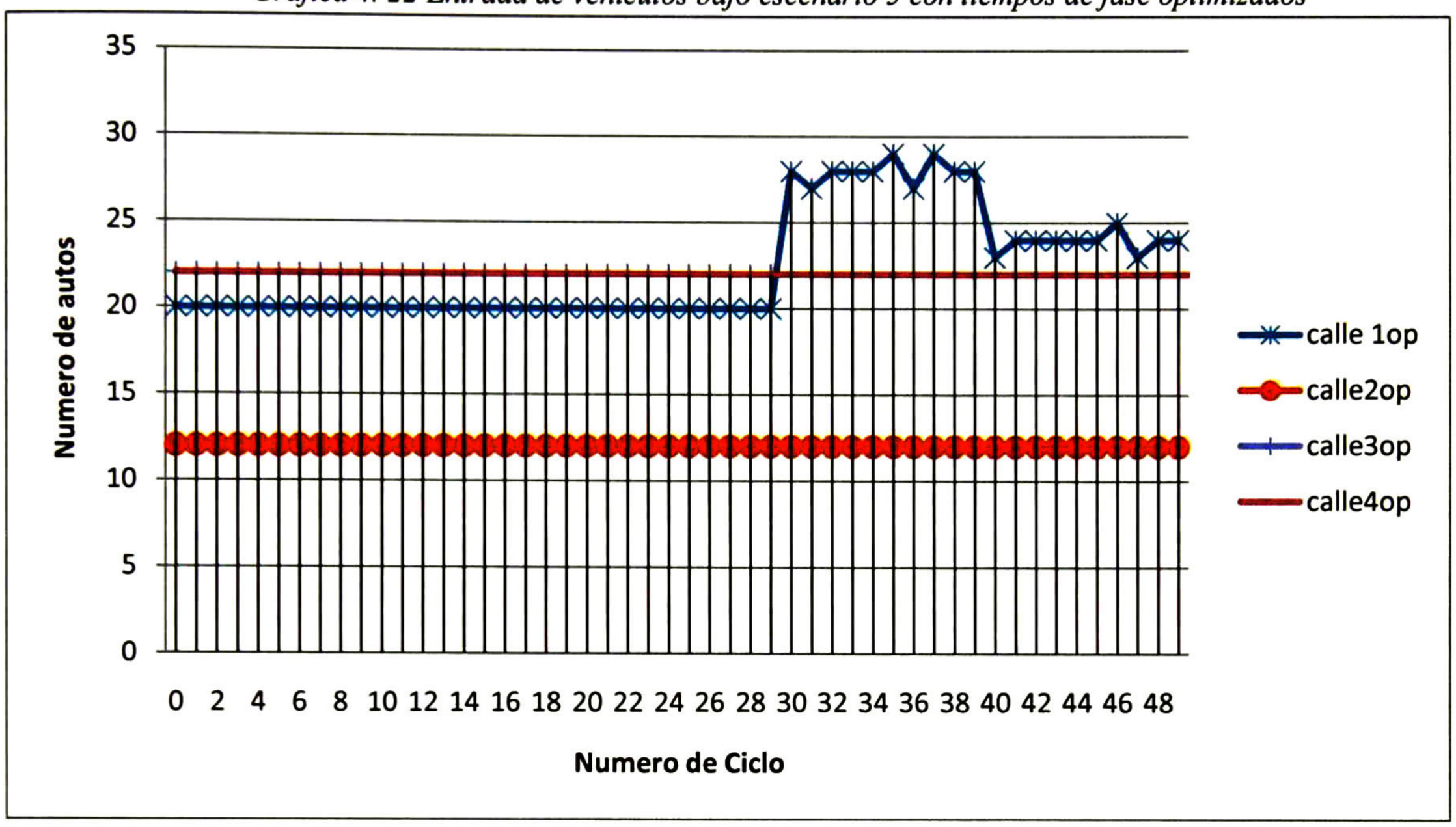
En la gráfica 4.21 se muestra la entrada de vehículos registrada en la simulación en el esquema de tiempos fijos bajo el escenario 5.

Gráfica 4. 21 Entrada de vehículos bajo escenario 5 con tiempos de fase fijos



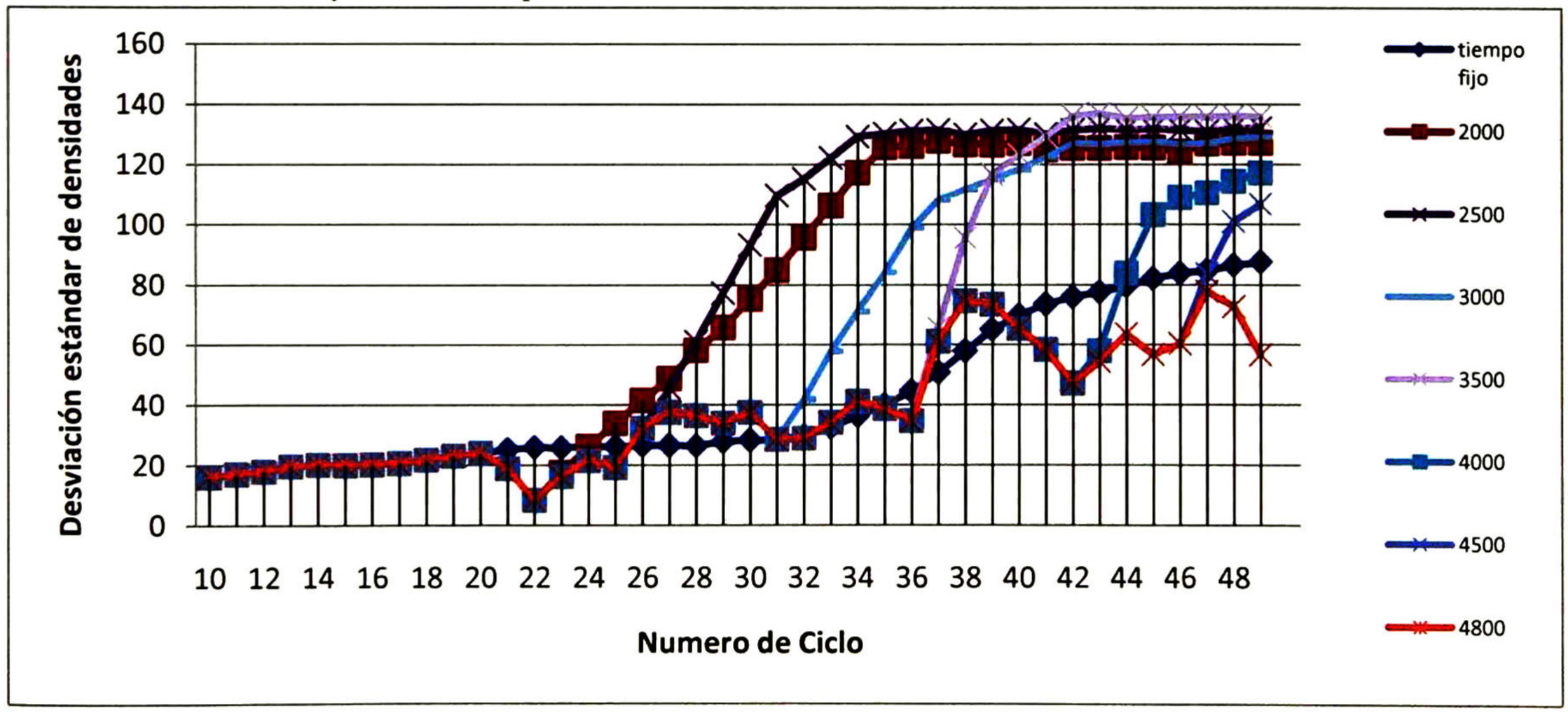
En la gráfica 4.22 se muestra la entrada de vehículos registrada en la simulación en el esquema de tiempos óptimos bajo el escenario 5.

Gráfica 4. 22 Entrada de vehiculos bajo escenario 5 con tiempos de fase optimizados



En la gráfica 4.23 se muestra la comparativa de la desviación estándar de las densidades de tráfico en las calles respecto al escenario 5. Se observa que bajo el esquema de tiempos óptimos (línea color rosa) solo en algunos puntos la desviación estándar es menor, esto debido a que en la simulación existen cambios drásticos en el flujo de una calle y al sistema de control le toma algunos ciclos llegar a un punto estable.

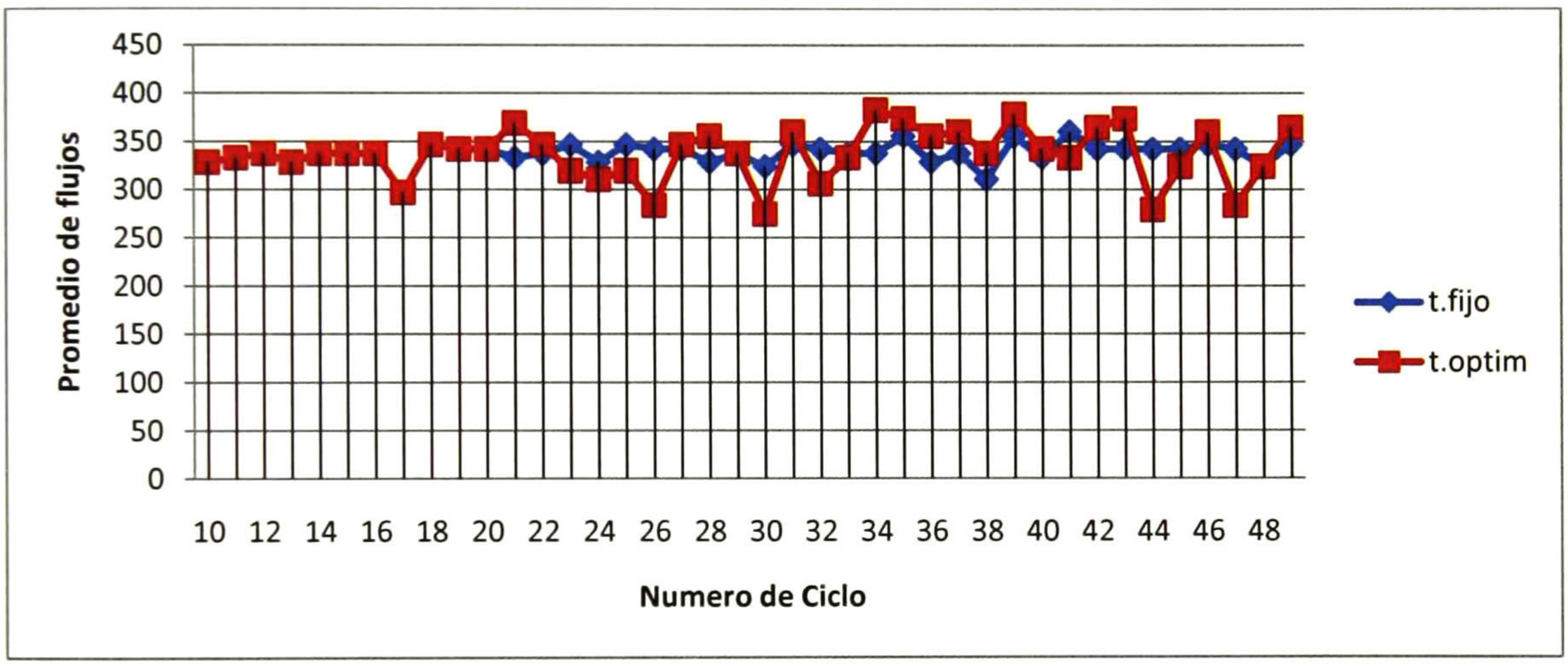
Gráfica 4.23 Comparativa de desviación estándar en simulación con el escenario 5



En la gráfica 4.24 se muestra la comparativa de los flujos de tráfico respecto al escenario 5, se observa que bajo el esquema de tiempos óptimos el promedio de flujos oscila respecto al de los tiempos fijos; aunque esta variación es evidente en la grafica, en la práctica no es significativa, ya que la unidad que se muestra en la gráfica para el promedio de flujos es en veh/hr y se ilustra la medida tomada durante 100 segundos.

Si se observa por ejemplo el ciclo numero 30 (donde se muestra la medida tomada en un periodo de 100 segundos), bajo el esquema de tiempos óptimos se tiene un valor de 274.5km/hr mientras que bajo el esquema de tiempos fijos se tiene un valor de 337.5 km/hr; esta diferencia no es realmente significativa, ya que si estas medidas de convierten a unidades de 100 segundos, se tiene una diferencia real de 1.75.

Gráfica 4. 24 Comparativa de promedio de flujos en simulación con el escenario 5



Para todos los escenarios de prueba en una red con sólo una intersección ocurre que en general, el esquema de tiempos optimizados logra disminuir la desviación estándar de los flujos, lo que se puede ver reflejado en una nivelación de colas de espera en semáforos o en distribución de densidad vehicular; ambos factores son indicadores de un mejor flujo de vehículos.

El aumento en flujos con la optimización es claramente notorio cuando existe alguna fase a la cual le está sobrando tiempo de verde, pero cuando se tiene un estado de tráfico donde todas las calles tienen vehículos saliendo de la intersección durante el total de tiempo de verde, el flujo de vehículos será similar en todas las fases, pero la diferencia hacia un mejor estado de tráfico será notoria en la nivelación de colas de espera en semáforo.

4.2. **Control de tráfico en un conjunto de intersecciones**

Después de haber realizado pruebas del sistema de control propuesto en una red de una sola intersección y observar sus resultados, se decidió probar el sistema mencionado en una red de caminos de 6 intersecciones, para observar resultados y evaluar la necesidad de complementar el sistema de control propuesto con un algoritmo de coordinación entre intersecciones.

4.2.1. **Descripción de la red de tráfico**

Se retoma la figura 2.4 en la figura 4.4, donde se muestra la red de caminos con 6 intersecciones utilizada para las siguientes pruebas, algunas características de esta red son:

- Todas las calles son de doble sentido.
- Cada sentido de la calle consta de dos carriles.
- La longitud de todos los segmentos de entrada a intersección es de 500 metros.
- Para cada calle de entrada a la intersección, el carril de la izquierda es para dar vuelta en U, vuelta a la izquierda o continuar en línea recta y el carril de la derecha es para continuar en línea recta o dar vuelta a la derecha. En la figura 4.2 se muestra una vista detallada de una intersección, las 6 utilizadas son similares a ésta.
- Cada intersección posee un semáforo para cada calle de entrada.
- El tiempo de ciclo en cada una de las intersecciones es de 100 segundos y no es variable.
- Todas las intersecciones inician la fase 1 en el mismo instante.
- Cada intersección posee 4 fases de 25 segundos cada una. Para cada fase los vehículos de entrada a la intersección pueden ir hacia todas direcciones. En la figura 4.3 se muestran de manera gráfica las 4 fases mencionadas.

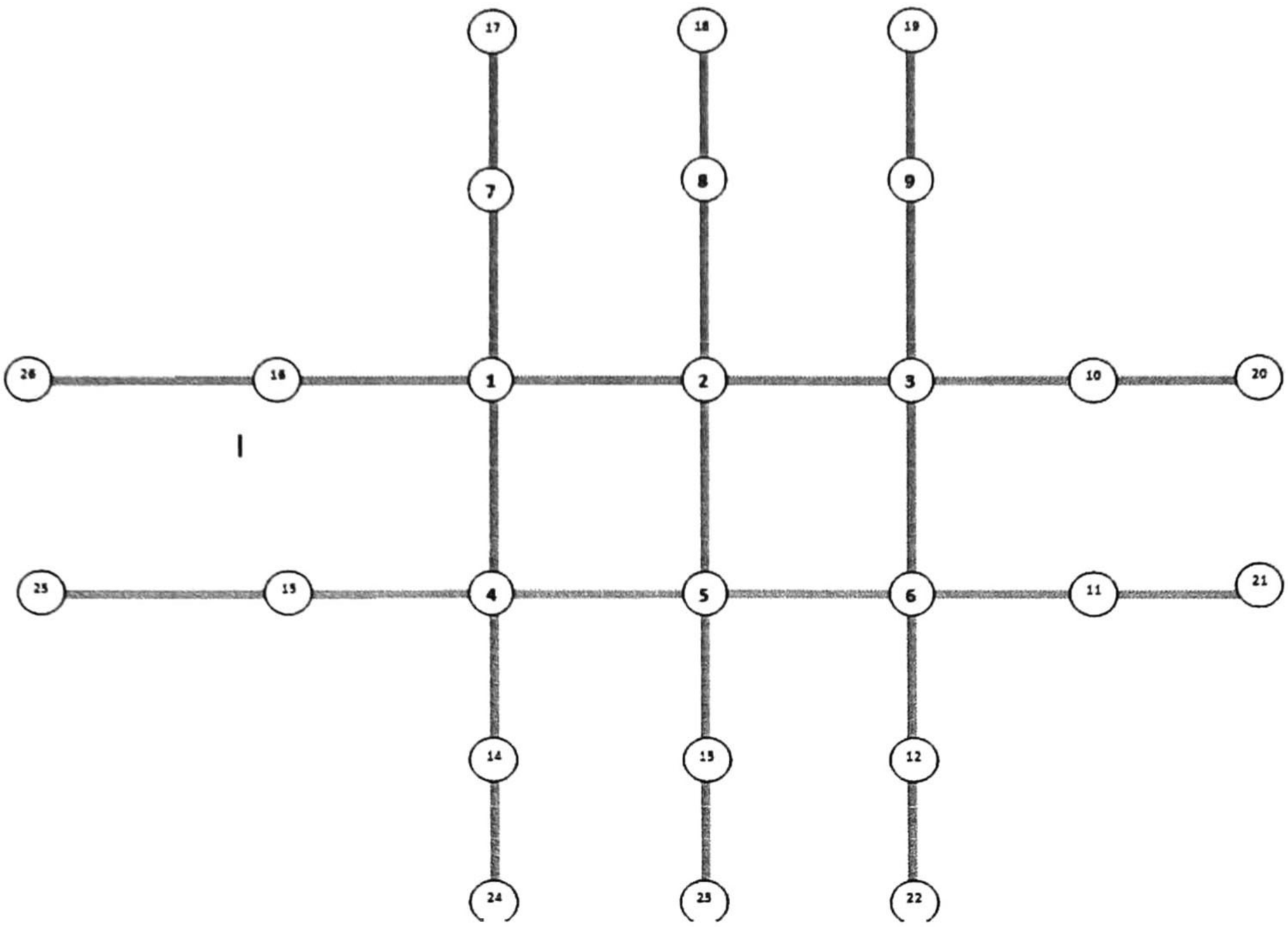


Figura 4. 4 Red de caminos con 6 intersecciones

Las características de tráfico utilizadas son las mismas que las especificadas en la sección 4.1. Todas las simulaciones realizadas en la red de caminos mostrada en la figura 4.4 se realizaron durante 5000 segundos. Al inicio de la simulación la calle está vacía, por lo que las políticas de control se comienzan a aplicar hasta el instante 2000, para así trabajar con el tráfico ya en su curso normal.

4.2.2. **Escenarios de prueba y resultados**

Para el caso de estudio de 6 intersecciones se tienen 3 escenarios con diferentes valores en sus tasas de llegada; para cada uno de los escenarios se analiza un esquema de control fijo y un esquema de control adaptable.

Para poder medir la eficiencia del sistema de control propuesto se optó por observar dos valores en la ejecución de las simulaciones, uno de ellos es la desviación estándar de las densidades (veh/km) existentes en las cuadras de entrada de todas las intersecciones y el otro

valor es el promedio de flujo de vehículos (veh/hr) registrado en detectores posicionados en el extremo de cada segmento más cercano la intersección.

Para cada escenario de prueba se comparan los valores mencionados tomados de un esquema de tiempos de fase fijos contra los mismos valores tomados bajo el esquema de tiempos de fase optimizados.

Al realizar las simulaciones con los diferentes escenarios de prueba, se consideró que el porcentaje máximo de ciclo permitido para una fase es del 80% y el porcentaje mínimo de ciclo permitido para una fase es del 5%. También se definió que el tiempo máximo que puede aumentar o disminuir una fase de un ciclo a otro sea de 20 segundos.

Escenario 1

En la tabla 4.6 se muestran las tasas de llegada que se indicaron al simulador para el escenario 1, en este escenario se simuló un comportamiento de tráfico en donde dos calles paralelas poseen una densidad de tráfico notable, mientras que las 3 calles que las cruzan tienen una baja densidad vehicular.

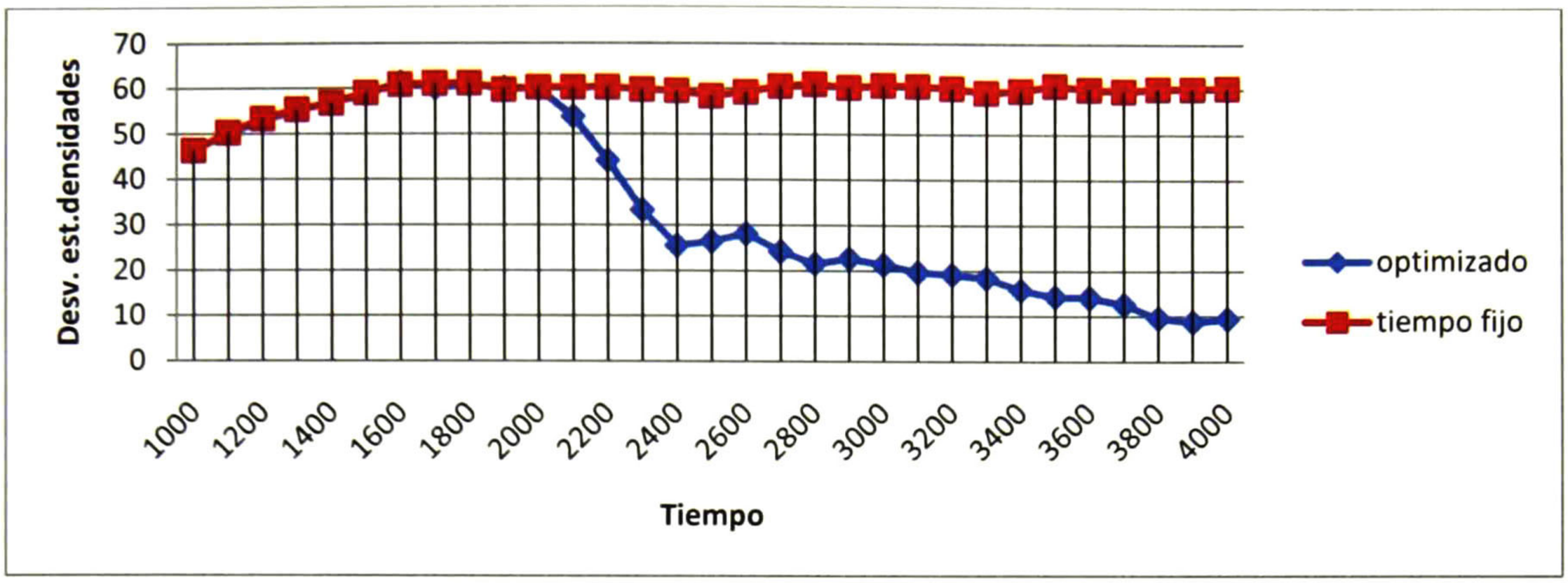
	Número de autos distribuidos uniformemente en el lapso de tiempo									
Tiempo	**calle 26-16**	**calle 17-7**	**calle 18-8**	**calle 19-9**	**calle 20-10**	**calle 21-11**	**calle 22-12**	**calle 23-13**	**calle 24-14**	**calle 25-15**
0-999	280	60	24	20	200	160	20	24	20	300
1000-1999	280	60	24	20	200	160	20	24	20	300
2000-2999	280	60	24	20	200	160	20	24	20	300
3000-3999	280	60	24	20	200	160	20	24	20	300
4000-4999	280	60	24	20	200	160	20	24	20	300

Tabla 4. 6 Tasas de entrada de vehículos para escenario 1 (6 intersecciones)

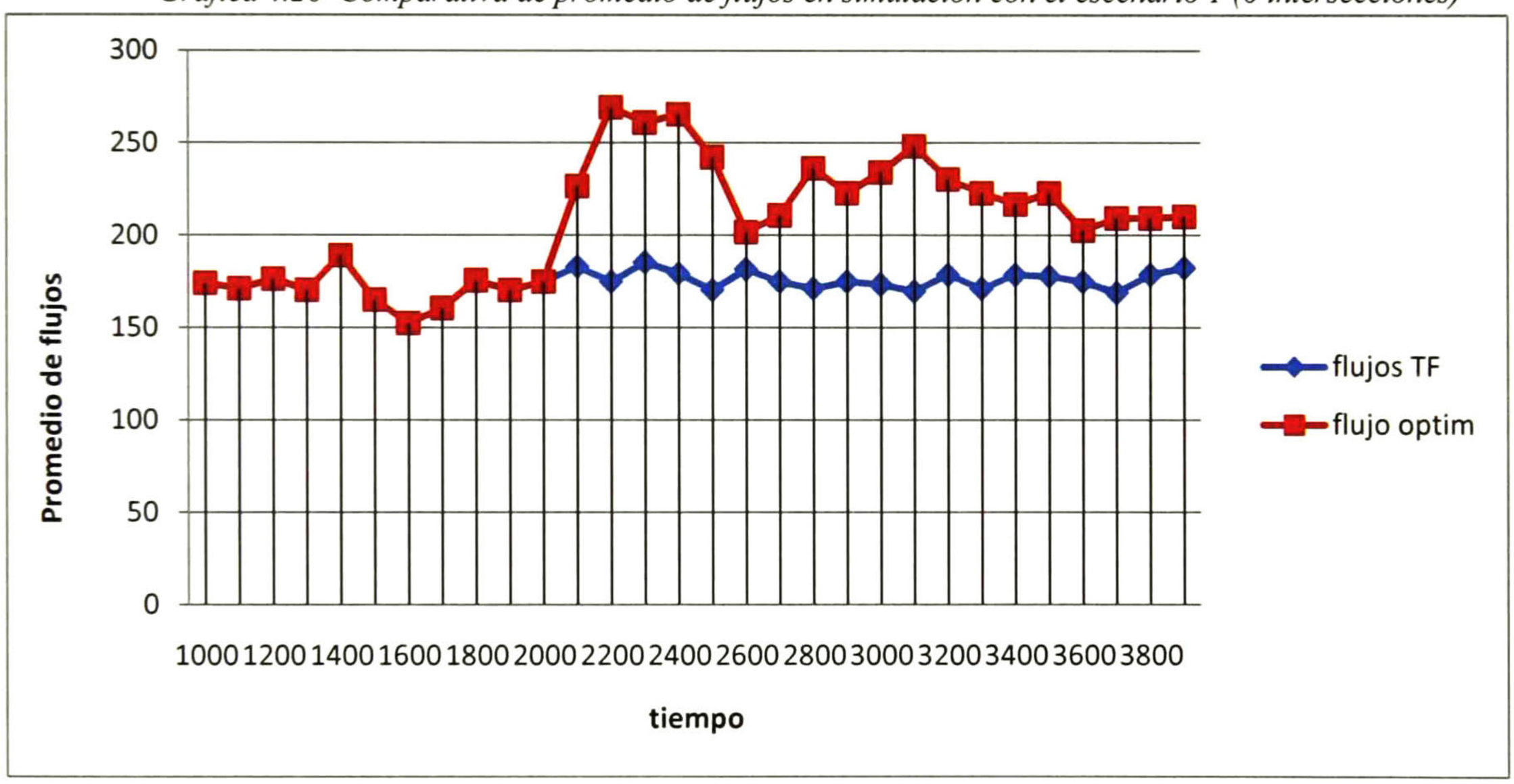
En las gráficas 4.25 y 4.26 se observa que bajo en el escenario 1 se obtienen mejoras tanto en la desviación estándar de densidades como en el promedio de los flujos; se logra bajar considerablemente la desviación estándar de las densidades y se logra también un incremento en el promedio de flujo de vehículos. Bajo este escenario la mejoría es bastante notoria debido a que la tasa de entrada de vehículos es grande en dos extremos de la red, mientras que en los otros extremos es muy baja, lo que ocasiona que bajo el esquema de tiempos fijos se tengan tiempos en verde mal utilizados, ya que ningún vehículo estaría pasando la intersección.

El problema mencionado se ve claramente solucionado mediante el esquema de control propuesto.

Gráfica 4. 25 Comparativa de desviación estándar en simulación con el escenario 1 (6 intersecciones)



Gráfica 4.26 Comparativa de promedio de flujos en simulación con el escenario 1 (6 intersecciones)



Escenario 2

En la tabla 4.7 se muestran las tasas de llegada que se indicaron al simulador para el escenario 2, en este escenario se simuló un comportamiento de tráfico en donde ninguna de las calles posee una baja densidad vehicular pero en el ciclo número 30, 4 extremos vecinos en la red de calles reciben un incremento en las tasas de entrada de autos.

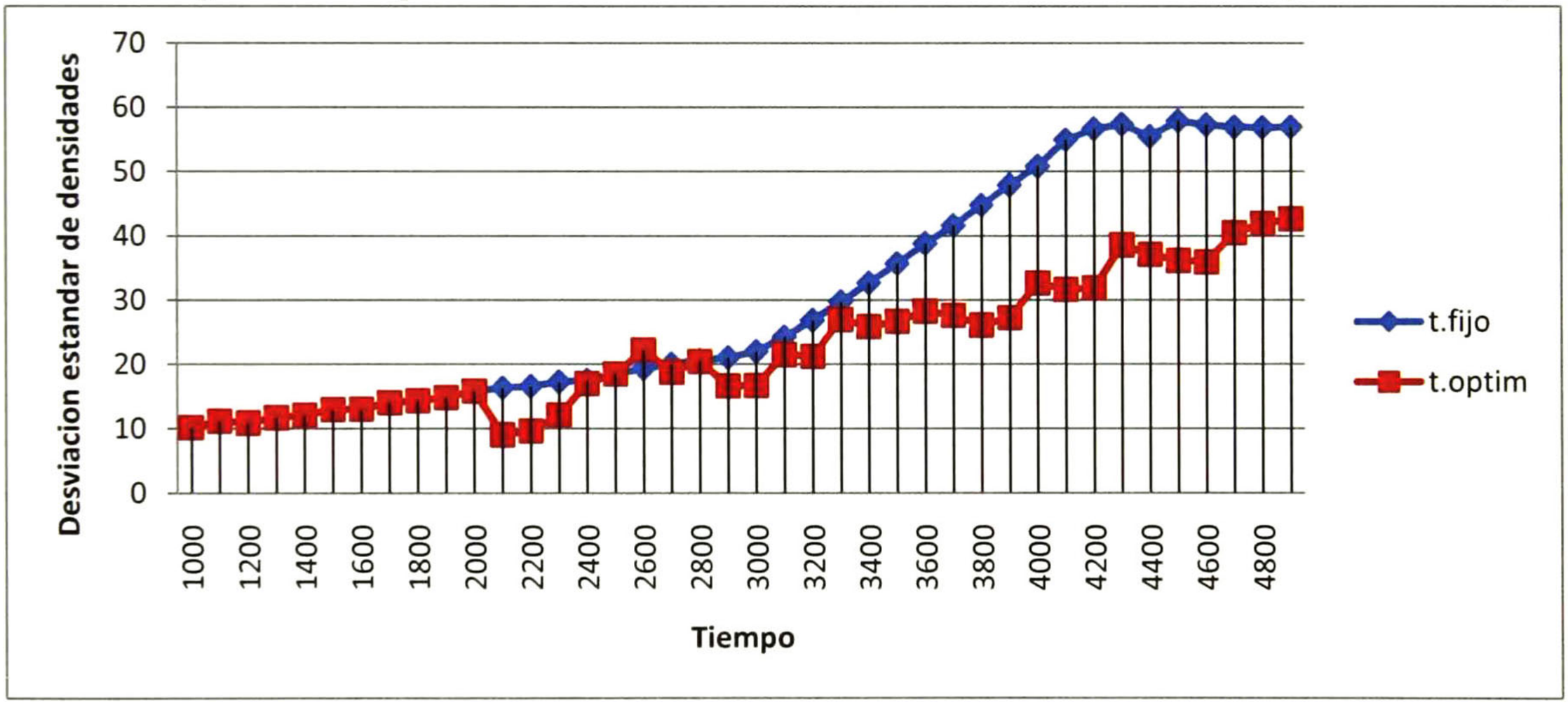
	Número de autos distribuidos uniformemente en el lapso de tiempo									
Tiempo	**calle 26-16**	**calle 17-7**	**calle 18-8**	**calle 19-9**	**calle 20-10**	**calle 21-11**	**calle 22-12**	**calle 23-13**	**calle 24-14**	**calle 25-15**
0-999	220	140	140	140	220	220	140	140	140	220
1000-1999	220	140	140	140	220	220	140	140	140	220
2000-2999	220	140	140	140	220	220	140	140	140	220
3000-3999	270	160	140	140	220	220	140	140	160	270
4000-4999	320	180	140	140	220	220	140	140	180	320

Tabla 4. 7 Tasas de entrada de vehículos para escenario 2 (6 intersecciones)

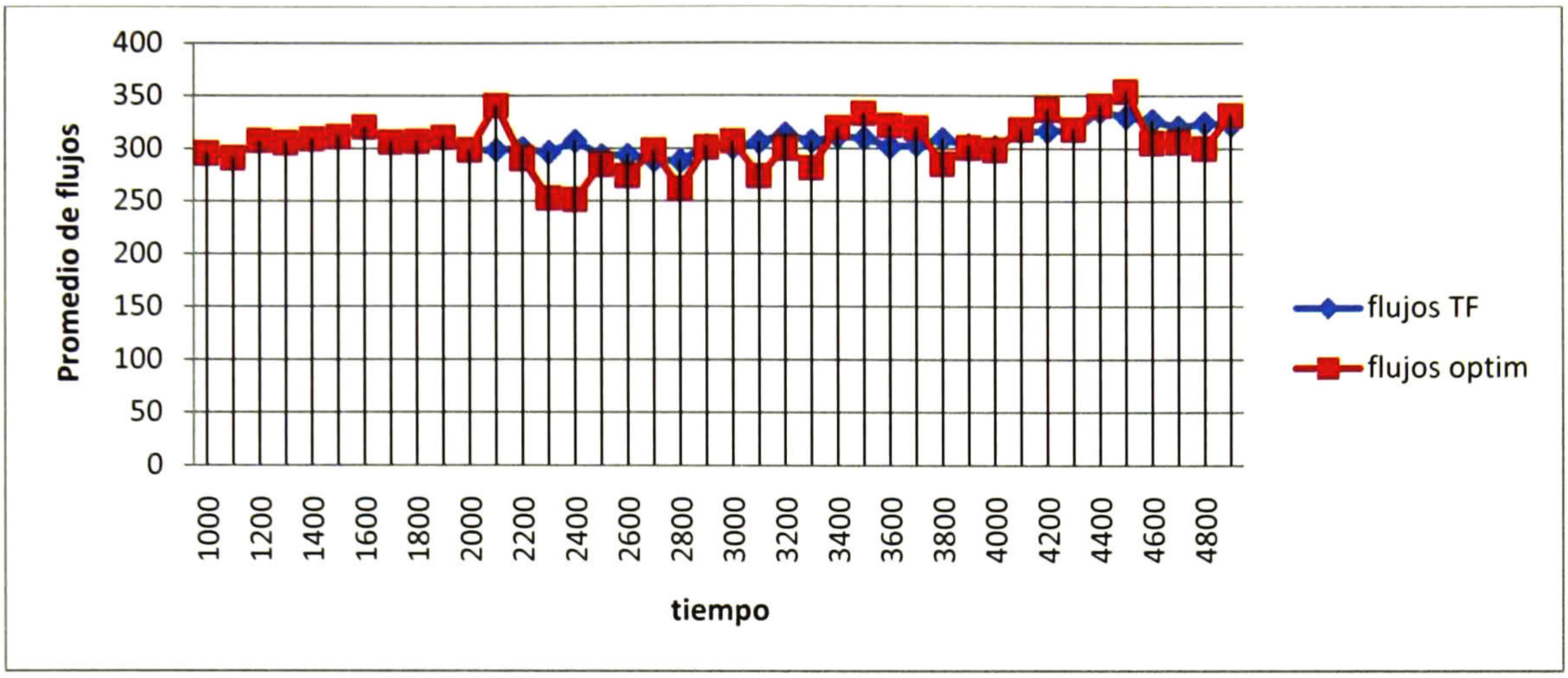
En las gráficas 4.27 y 4.28 se observan los resultados correspondientes a las pruebas con el escenario 2, para este caso no hay una mejoría visible en cuanto al promedio de flujos, esto es debido a que a diferencia del escenario 1, (en donde había diferencias muy marcadas en tasas de entrada por diferentes extremos) en el escenario 2 no hay tasas de entrada que sean muy bajas, por lo que durante el total de tiempo de verde de cualquier fase hay vehículos circulando, es decir, no ocurre el mal uso de tiempo de verde que se presenta bajo el escenario 1.

Lo observado respecto al flujo de vehículos no brinda información acerca de la longitud de colas de espera o distribución de densidad vehicular; la desviación estándar es la que nos da idea respecto a esto y para el escenario 2 este parámetro disminuye con el esquema de control propuesto.

Gráfica 4. 27 Comparativa de desviación estándar en simulación con el escenario 2 (6 intersecciones)



Gráfica 4. 28 Comparativa de promedio de flujos en simulación con el escenario 2 (6 intersecciones)



En la tabla 4.8 se muestran las tasas de llegada que se indicaron al simulador para el escenario 3. En este escenario se simuló un comportamiento de tráfico en donde ninguna de las calles posee una baja densidad vehicular pero en el ciclo número 30, 4 extremos vecinos en la red de calles reciben un incremento en las tasas de entrada de autos y posteriormente en 2 extremos de los mencionados las tasas de entrada vuelven a sus valores iniciales.

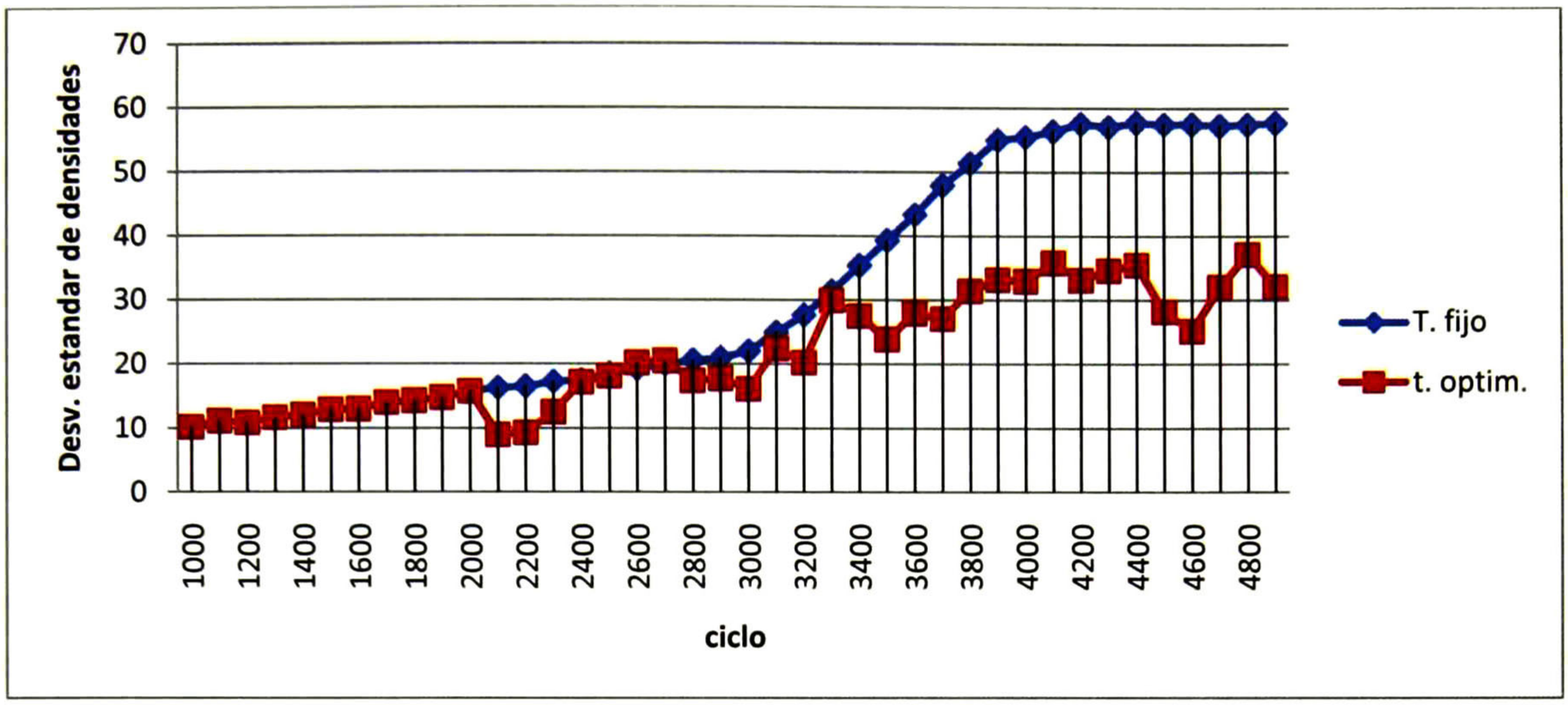
	Número de autos distribuidos uniformemente en el lapso de tiempo									
Tiempo	**calle 26-16**	**calle 17-7**	**calle 18-8**	**calle 19-9**	**calle 20-10**	**calle 21-11**	**calle 22-12**	**calle 23-13**	**calle 24-14**	**calle 25-15**
0-999	220	140	140	140	220	220	140	140	140	220
1000-1999	220	140	140	140	220	220	140	140	140	220
2000-2999	220	140	140	140	220	220	140	140	140	220
3000-3999	290	170	140	140	220	220	140	140	170	290
4000-4999	230	140	140	140	220	220	140	140	140	230

Tabla 4. 8 Tasas de entrada de vehículos para escenario 3 (6 intersecciones)

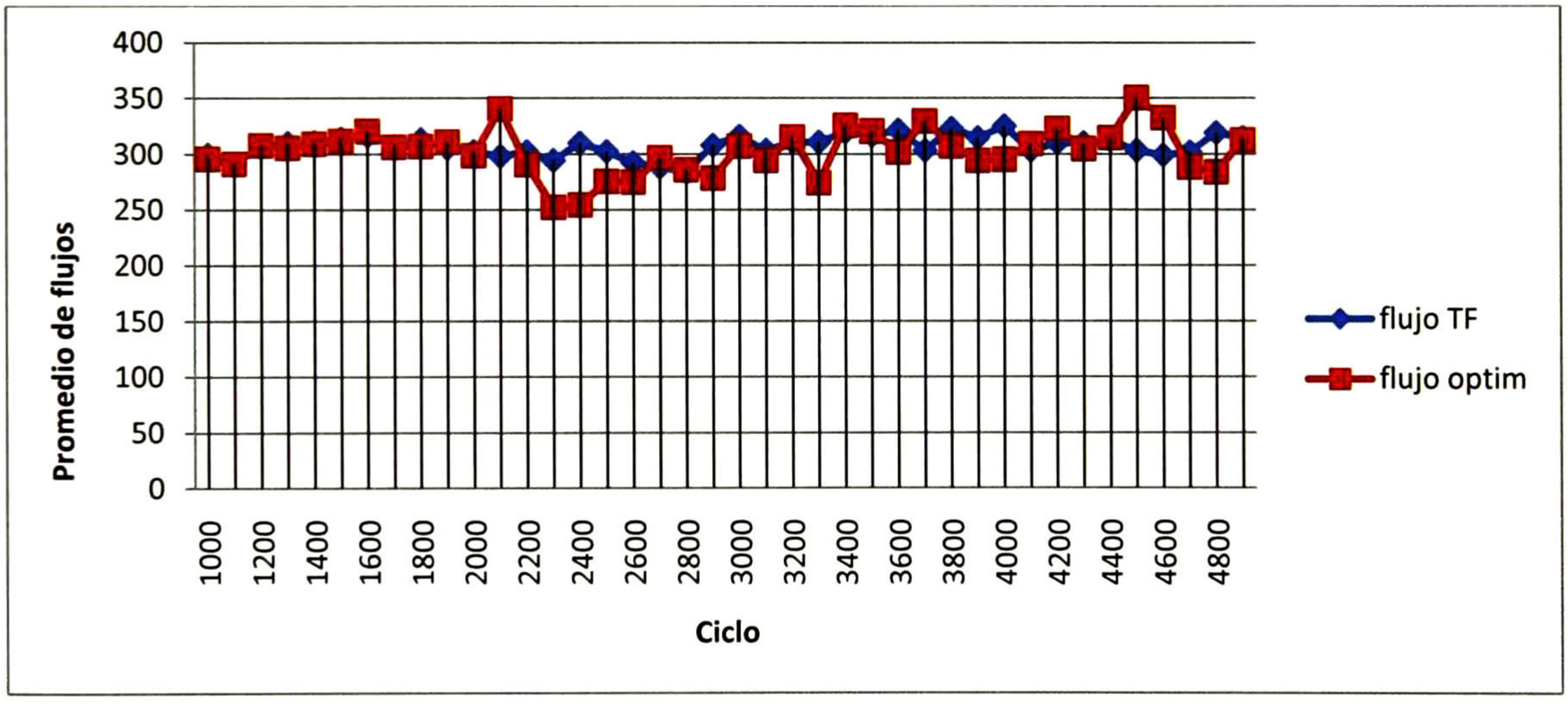
Las gráficas 4.29 y 4.30 muestran el resultado de la simulación del escenario 3, en donde se observa que también se logra mejorar la desviación estándar de densidades; en los flujos no se observa diferencia debido a que no se tienen tasas bajas de entrada por ningún extremo de la red de calles.

Se puede observar también que tanto en el escenario 2 como en el 3 ocurren incrementos en las tasas de entrada de algunas calles de la red y el sistema de control propuesto logra mejorar el tráfico aún bajo estas condiciones.

Gráfica 4. 29 Comparativa de desviación estándar en simulación con el escenario 3 (6 intersecciones)



Gráfica 4. 30 Comparativa de promedio de flujos en simulación con el escenario 3 (6 intersecciones)



4.3. **Conclusiones**

Después de observar las simulaciones en la red de una intersección y sobre todo en la red con 6 intersecciones se concluyó que el sistema de control propuesto no requiere complementarse con un algoritmo de coordinación entre intersecciones vecinas, ya que las reglas para que una intersección no afecte a la intersección hacia donde manda flujo están implícitas en las restricciones del problema de programación cuadrática que soluciona el sistema de control propuesto.

Si cada intersección se rige bajo las reglas de control propuestas se logra obtener un mejor estado de tráfico que mediante tiempos fijos.

Cuando todas las fases de una o varias intersecciones logran sacar autos durante sus tiempos de verde, tanto en tiempos fijos como con tiempos óptimos la diferencia en flujos no es notoria, sin embargo, cuando se disminuye la desviación estándar de las densidades en calles se obtiene una densidad de tráfico mejor distribuida, lo cual lleva a una mejor circulación de vehículos en la red de calles.

Conclusiones

En esta tesis se presentó un método de control adaptativo de tráfico urbano basado en programación cuadrática, en donde para cada intersección se resuelve un problema de optimización con la finalidad de obtener las variaciones de tiempo de fases de los semáforos que permitan nivelar la densidad de tráfico en las calles de entrada.

En el sistema propuesto cada intersección busca aplicar los tiempos de fase óptimos que le permitan nivelar la densidad de tráfico en sus calles de entrada, estos tiempos de fase son restringidos mediante valores generados por sensores en las calles de salida de la intersección, que a su vez son las calles de entrada para una intersección adyacente, lo que define de manera implícita una política de no afectar a las intersecciones vecinas. Si cada intersección se rige bajo las reglas de control propuestas se logra obtener un mejor estado de tráfico del área.

El método propuesto es distribuido, ya que cada una de las intersecciones a controlar aplica sus reglas basándose en datos arrojados por sensores en sus calles de entrada y salida. El mismo sistema se aplica en todas las intersecciones que se desee controlar. Este esquema resulta más factible de implementar que otros sistemas de control adaptativo propuestos, ya que al no tener la necesidad de negociación entre intersecciones, se elimina la instalación de dispositivos de comunicación.

La simulación del método fue mediante el simulador de tráfico urbano SUMO, donde se implementó el control de tráfico en una red de caminos con seis intersecciones bajo distintos escenarios de prueba y se observó que cuando en una intersección existe una fase durante la cual en el tiempo de verde no hay flujo de vehículos el sistema propuesto logra una buena mejoría en el promedio de flujo de vehículos. Cuando todas las fases de una o varias intersecciones logran sacar autos durante sus tiempos de verde, la diferencia en flujos no es notoria; sin embargo, en los dos casos mencionados disminuye la desviación estándar de las densidades en calles, por lo que se obtiene una densidad de tráfico mejor distribuida, lo cual lleva a una mejor circulación de vehículos en la red de calles.

Se ha presentado una primera aproximación hacia el control de tráfico urbano mediante programación cuadrática, por lo que se observa en los escenarios de simulación los resultados son positivos respecto a los tiempos fijos. Una siguiente etapa de este proyecto es analizar la implementación de esquemas de control similares al presentado, pero con variantes en sus restricciones y función objetivo, para ser aplicados bajo algún patrón de tráfico detectado de manera se mejore el desempeño del sistema de control respecto a la aplicación de un esquema general.

Respecto a las pruebas del controlador, es conveniente realizar la simulación con áreas de red de tráfico más complejas describiendo configuraciones y utilizando flujos de tráfico del mundo real.

APENDICE A. APLICACIONES DESARROLLADAS

A.1. Depurar Densidades

```
#include <stdio.h>
#include <conio.h>
#include <string.h>

int inStr(char *pattern, char *str) {
        int i, foundPos = 0;
        while(str[foundPos]!=0) {
                if (str[foundPos]==pattern[0]) {
                        i = 1;
                        while(pattern[i]!=0 && str[foundPos + i]!=0 && str[foundPos + i]==pattern[i]) { i++; }
                        if (pattern[i]==0) {
                                return foundPos;
                        } else if (str[foundPos + i]==0) {
                                return -1;
                        }
                }
                foundPos++;
        }
        return -1;
}

//las longitudes se expresan tomando en cuenta el cero
 //busca la cadena "buscando" en la cadena "cadena"
//regresa la posicion donde se encuentra o -1 si no se encuentra
int encuentra(char *cadena, char *buscando, int long_cadena, int long_buscando)
 {
 int i,busca=0,ini=0;
 for(i=0;i<=long_cadena;i++)
  {
  ini=i;
  while(cadena[i]==buscando[busca])//guardar la primera posicion
    {
    if (busca==long_buscando)
      return(ini);
     busca++;
     i++;
    }
  busca=0;
  i=ini;
  }
  return(-1);
  }

void copiarnn(char *destino, char *origen, int ini, int fin)
 {
 int i,desti=0;
 for(i=ini;i<=fin;i++)
  {
  destino[desti]=origen[i];
  desti++;
```

```
   }
  destino[desti]='\0';
  }
//-------------------------------------------------------------------------------------------------
int main(int argc, char *argv[]){
    char linea[100], encabezado[200],seg[200];
    int vete=0,EstoyEnCiclo=0,segu,a,aa,i1i,d1i,o1i,conta=0,longi,num_lanes=24;
    FILE *ar_leer,*ar_escribir;

    char lane0[10]="16-1",lane1[10]="2-1",lane2[10]="4-1",lane3[10]="7-1"; //interseccion 1
    char lane4[10]="1-2",lane5[10]="3-2",lane6[10]="5-2",lane7[10]="8-2"; //interseccion 2
    char lane8[10]="2-3",lane9[10]="10-3",lane10[10]="6-3",lane11[10]="9-3"; //interseccion 3
    char lane12[10]="15-4",lane13[10]="5-4",lane14[10]="14-4",lane15[10]="1-4"; //interseccion 4
    char lane16[10]="4-5",lane17[10]="6-5",lane18[10]="13-5",lane19[10]="2-5"; //interseccion 5
    char lane20[10]="5-6",lane21[10]="11-6",lane22[10]="12-6",lane23[10]="3-6"; //interseccion 6

    char densi0[10], densi1[10], densi2[10], densi3[10];
    char densi4[10], densi5[10], densi6[10], densi7[10];
    char densi8[10], densi9[10], densi10[10], densi11[10];
    char densi12[10], densi13[10], densi14[10], densi15[10];
    char densi16[10], densi17[10], densi18[10], densi19[10];
    char densi20[10], densi21[10], densi22[10], densi23[10];

     strcpy(encabezado,"seg 16-1 2-1 4-1 7-1 1-2 3-2 5-2 8-2 2-3 10-3 6-3 9-3 15-4 5-4 14-4 1-4 4-5 6-5 13-5 2-5 5-6
                                                                                    11-6 12-6 3-6 \n");
    ar_leer=fopen("densi-opti-hasta-2000.txt", "r");
    ar_escribir=fopen("densi-en-el-2000.txt","w");
    //printf("(depura)el argumento es: %s\n",argv[1]);getch();
    fputs(encabezado,ar_escribir);
    do{
       fgets(linea,100,ar_leer);
       //puts(linea);
       a=encuentra(linea,"interval begin",100,13);
       segu=encuentra(linea,argv[1],100,3);
       if(a!=-1 && segu!=-1)//----------------------------------interval begin
          {
          aa=encuentra(linea,"end",100,2);
          copiarnn(seg,linea,a+16,aa-3);
          conta++;
          EstoyEnCiclo=1;
          //printf("conta: %d  %s \n",conta,linea);  getch();
          }//if(a!=-1)
       //-------------------------------------------------------------------------------------
       longi=strlen(lane0);
       i1i=encuentra(linea,lane0,100,longi-1);
    if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
       {
       d1i=encuentra(linea,"density=",100,7);
       o1i=encuentra(linea,"occupancy",100,8);
       copiarnn(densi0,linea,d1i+9,o1i-3);
       conta++;
       //printf("conta: %d  %s \n",conta,linea); getch();
       }//if
    //-------------------------------------------------------------------------------------
    longi=strlen(lane1);
    i1i=encuentra(linea,lane1,100,longi-1);
```

```
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi1,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea);getch();
  }//if
//-------------------------------------------------------------------------------
longi=strlen(lane2);
i1i=encuentra(linea,lane2,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi2,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------
longi=strlen(lane3);
i1i=encuentra(linea,lane3,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi3,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------
longi=strlen(lane4);
i1i=encuentra(linea,lane4,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi4,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------
longi=strlen(lane5);
i1i=encuentra(linea,lane5,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi5,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------
longi=strlen(lane6);
i1i=encuentra(linea,lane6,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
```

```
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi6,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------
longi=strlen(lane7);
i1i=encuentra(linea,lane7,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi7,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------
longi=strlen(lane8);
i1i=encuentra(linea,lane8,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi8,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------
longi=strlen(lane9);
i1i=encuentra(linea,lane9,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi9,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------
longi=strlen(lane10);
i1i=encuentra(linea,lane10,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi10,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------
longi=strlen(lane11);
i1i=encuentra(linea,lane11,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
```

```
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi11,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------------
longi=strlen(lane12);
i1i=encuentra(linea,lane12,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi12,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------------
longi=strlen(lane13);
i1i=encuentra(linea,lane13,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi13,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------------
longi=strlen(lane14);
i1i=encuentra(linea,lane14,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi14,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------------
longi=strlen(lane15);
i1i=encuentra(linea,lane15,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi15,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------------
longi=strlen(lane16);
i1i=encuentra(linea,lane16,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
```

```
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi16,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------
longi=strlen(lane17);
i1i=encuentra(linea,lane17,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi17,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------
longi=strlen(lane18);
i1i=encuentra(linea,lane18,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi18,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------
longi=strlen(lane19);
i1i=encuentra(linea,lane19,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi19,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------
longi=strlen(lane20);
i1i=encuentra(linea,lane20,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi20,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------
longi=strlen(lane21);
i1i=encuentra(linea,lane21,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
```

```
  copiarnn(densi21,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------------
longi=strlen(lane22);
i1i=encuentra(linea,lane22,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi22,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if
//-------------------------------------------------------------------------------------
longi=strlen(lane23);
i1i=encuentra(linea,lane23,100,longi-1);
if(i1i!=-1 && linea[i1i-1]=="" && linea[i1i+longi]=="" && EstoyEnCiclo==1)
  {
  d1i=encuentra(linea,"density=",100,7);
  o1i=encuentra(linea,"occupancy",100,8);
  copiarnn(densi23,linea,d1i+9,o1i-3);
  conta++;
  //printf("conta: %d  %s \n",conta,linea); getch();
  }//if

if(conta==25 && EstoyEnCiclo==1)
  {
conta=0;
vete=1;
strcat(seg," "); strcat(seg,densi0);
strcat(seg," "); strcat(seg,densi1);
strcat(seg," "); strcat(seg,densi2);
strcat(seg," "); strcat(seg,densi3);
strcat(seg," "); strcat(seg,densi4);
strcat(seg," "); strcat(seg,densi5);
strcat(seg," "); strcat(seg,densi6);
strcat(seg," "); strcat(seg,densi7);
strcat(seg," "); strcat(seg,densi8);
strcat(seg," "); strcat(seg,densi9);
strcat(seg," "); strcat(seg,densi10);
strcat(seg," "); strcat(seg,densi11);
strcat(seg," "); strcat(seg,densi12);
strcat(seg," "); strcat(seg,densi13);
strcat(seg," "); strcat(seg,densi14);
strcat(seg," "); strcat(seg,densi15);
strcat(seg," "); strcat(seg,densi16);
strcat(seg," "); strcat(seg,densi17);
strcat(seg," "); strcat(seg,densi18);
strcat(seg," "); strcat(seg,densi19);
strcat(seg," "); strcat(seg,densi20);
strcat(seg," "); strcat(seg,densi21);
strcat(seg," "); strcat(seg,densi22);
strcat(seg," "); strcat(seg,densi23);
strcat(seg,"\n");
```

```
      printf("%s \n",seg);
      fputs(seg,ar_escribir);
        }//if conta
    }while(!feof(ar_leer) && vete==0);

//printf("el ultimo valor de conta es : %d\n",conta);
//printf("la ultima linea: \n %s \n",linea);
fclose(ar_leer);
fclose(ar_escribir);
//printf("\n Se escribieron densidades de ciclo ");
getch();
}
```

A.2. Calcular espacios libres

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>

int inStr(char *pattern, char *str) {
        int i, foundPos = 0;
        while(str[foundPos]!=0) {
                if (str[foundPos]==pattern[0]) {
                        i = 1;
                        while(pattern[i]!=0 && str[foundPos + i]!=0 && str[foundPos + i]==pattern[i]) { i++; }
                        if (pattern[i]==0) {
                                return foundPos;
                        } else if (str[foundPos + i]==0) {
                                return -1;
                        }
                }
                foundPos++;
        }
        return -1;
}

//las longitudes se expresan tomando en cuenta el cero
 //busca la cadena "buscando" en la cadena "cadena"
 //regresa la posicion donde se encuentra o -1 si no se encuentra
 int encuentra(char *cadena, char *buscando, int long_cadena, int long_buscando)
{
int i,busca=0,ini=0;

for(i=0;i<=long_cadena;i++)
  {
  ini=i;
  while(cadena[i]==buscando[busca])//guardar la primera posicion
    {
    if (busca==long_buscando)
      return(ini);
     busca++;
     i++;
    }
  busca=0;
  i=ini;
  }
  return(-1);
  }

void copiarnn(char *destino, char *origen, int ini, int fin)
  {
  int i,desti=0;
  for(i=ini;i<=fin;i++)
    {
    destino[desti]=origen[i];
    desti++;
    }
```

```
 destino[desti]='\0';
 }

//la siguiente funcion es para sacar el valor de meanVehicleNumber del archivo txt generado por SUMO
// el archivo que se proporciona tiene los datos de todas las fases, pero unicamente de un solo carril.
//en el segundo paramentro se especifica el numero de segundo de la simulacion, a partir de ese segundo
//se guardaran los valores de las 4 fases del ciclo, en las variables val"n"
void sacar_valores(char nom_archivo[], char tiempo[], char val1[],char val2[],char val3[],char val4[])
 {
  FILE *ar;
  int sale=0,longi_linea,a,aa,b,c;
  char linea[530];
ar=fopen(nom_archivo,"r");
while(!feof(ar) and sale==0){
  fgets(linea,530,ar);
  longi_linea=strlen(linea);
  a=encuentra(linea,tiempo,longi_linea,strlen(tiempo)-1);
  aa=encuentra(linea,"end",longi_linea,2);
  if(a!=-1 && a<=aa)  //----------------------------------interval begin
    {
    b=encuentra(linea,"meanVehicleNumber=",longi_linea,17);
    c=encuentra(linea,"maxVehicleNumber",longi_linea,15);
    copiarnn(val1,linea,b+19,c-3);
    fgets(linea,530,ar);
    fgets(linea,530,ar);
    longi_linea=strlen(linea);
    b=encuentra(linea,"meanVehicleNumber=",longi_linea,17);
    c=encuentra(linea,"maxVehicleNumber",longi_linea,15);
    copiarnn(val2,linea,b+19,c-3);
    fgets(linea,530,ar);
    fgets(linea,530,ar);
    longi_linea=strlen(linea);
    b=encuentra(linea,"meanVehicleNumber=",longi_linea,17);
    c=encuentra(linea,"maxVehicleNumber",longi_linea,15);
    copiarnn(val3,linea,b+19,c-3);
    fgets(linea,530,ar);
    fgets(linea,530,ar);
    longi_linea=strlen(linea);
    b=encuentra(linea,"meanVehicleNumber=",longi_linea,17);
    c=encuentra(linea,"maxVehicleNumber",longi_linea,15);
    copiarnn(val4,linea,b+19,c-3);
    sale=1;
    }
  }//while
fclose(ar);
 } //fin declaracion de la funcion sacar_valores
//-------------------------------------------------------------------------------------------------------
//  este programa debera ejecutarse despues de una simulacion con guisim, ya que ahi se estan creando los archivos
//correspondientes a espacios ocupados, (un archivo por carril) esos archivos son tomados por este programa
//para calcular los espacios libres correspondientes a cada fase de cada intersección
```

```
int main(int argc, char *argv[]){
  char encabezado[200],llinea[400], val1[20],val2[20],val3[20],val4[20],tiempo[100],nom_archivo[100],
linea_esp_libres[500],texto[20];
   int inte1[4][8],inte2[4][8],inte3[4][8],inte4[4][8],inte5[4][8],inte6[4][8];
   int capacidad, esp_libres_f1,esp_libres_f2,esp_libres_f3,esp_libres_f4;
   FILE *ar;
   ar=fopen("esp-libres.txt","w");
   //printf("(esp libres) el argumento tomado es: %s\n",argv[1]);getch();
   strcpy(tiempo,argv[1]);   // segundo en que inicia el ciclo del cual se quieren saber los espacios libres

   linea_esp_libres[0]='\0';
   strcpy(encabezado,"16-1 2-1 4-1 7-1 1-2 3-2 5-2 8-2 2-3 10-3 6-3 9-3 15-4 5-4 14-4 1-4 4-5 6-5 13-5 2-5 5-6 11-6
                                                                                    12-6 3-6 \n");
fputs(encabezado,ar);
capacidad=58;  //capacidad maxima para un carril en numero de vehiculoa
//  ----------------------------interseccion 1 ----------------------------------------------------------

// primero se guardaran en el arreglo bidimensional,

sacar_valores("1-16_0.txt",tiempo,val1,val2,val3,val4);
//printf("val1=%s  ,val2=%s, val3=%s, val4=%s, \n",val1,val2,val3,val4);
inte1[0][0]=atoi(val1) ;inte1[1][0]=atoi(val2);inte1[2][0]=atoi(val3);inte1[3][0]=atoi(val4);

sacar_valores("1-16_1.txt",tiempo,val1,val2,val3,val4);
//printf("val1=%s  ,val2=%s, val3=%s, val4=%s, \n",val1,val2,val3,val4);
inte1[0][1]=atoi(val1) ;inte1[1][1]=atoi(val2);inte1[2][1]=atoi(val3);inte1[3][1]=atoi(val4);
sacar_valores("1-2_0.txt",tiempo,val1,val2,val3,val4);
inte1[0][2]=atoi(val1) ;inte1[1][2]=atoi(val2);inte1[2][2]=atoi(val3);inte1[3][2]=atoi(val4);
sacar_valores("1-2_1.txt",tiempo,val1,val2,val3,val4);
inte1[0][3]=atoi(val1) ;inte1[1][3]=atoi(val2);inte1[2][3]=atoi(val3);inte1[3][3]=atoi(val4);
sacar_valores("1-4_0.txt",tiempo,val1,val2,val3,val4);
inte1[0][4]=atoi(val1) ;inte1[1][4]=atoi(val2);inte1[2][4]=atoi(val3);inte1[3][4]=atoi(val4);
sacar_valores("1-4_1.txt",tiempo,val1,val2,val3,val4);
inte1[0][5]=atoi(val1) ;inte1[1][5]=atoi(val2);inte1[2][5]=atoi(val3);inte1[3][5]=atoi(val4);
sacar_valores("1-7_0.txt",tiempo,val1,val2,val3,val4);
inte1[0][6]=atoi(val1) ;inte1[1][6]=atoi(val2);inte1[2][6]=atoi(val3);inte1[3][6]=atoi(val4);
sacar_valores("1-7_1.txt",tiempo,val1,val2,val3,val4);
inte1[0][7]=atoi(val1) ;inte1[1][7]=atoi(val2);inte1[2][7]=atoi(val3);inte1[3][7]=atoi(val4);


//estando en el arreglo, se hacen las sumas y restas correspondientes para calcular los espacios libres
//correspondientes a las calles,

esp_libres_f1=  (capacidad*6) - (inte1[0][2]+inte1[0][3]+inte1[0][4]+inte1[0][5]+inte1[0][6]+inte1[0][7]);
strcat(linea_esp_libres,itoa(esp_libres_f1,texto,10));strcat(linea_esp_libres," ");
 //"capacidad" se multiplica por -6, porque son tres carriles
//  printf("la fase 1 de la interseccion 1 tiene %d espacios libres \n",esp_libres_f1);getch();
 //en el mismo acomodo en que "depura-densi" acomoda las densidades de las calles
//aqui se estarian escribiendo en el archivo, los espacios libres que tiene cada calle en sus calles de salida.
//(o espacios libres que tiene cada fase)
 esp_libres_f2=(capacidad*6) - (inte1[1][0]+inte1[1][1]+inte1[1][4]+inte1[1][5]+inte1[1][6]+inte1[1][7]);
 strcat(linea_esp_libres,itoa(esp_libres_f2,texto,10));strcat(linea_esp_libres," ");
//  printf("la fase 2 de la interseccion 1 tiene %d espacios libres \n",esp_libres_f2);getch();

 esp_libres_f3=(capacidad*6) - (inte1[2][0]+inte1[2][1]+inte1[2][2]+inte1[2][3]+inte1[2][6]+inte1[2][7]);
 strcat(linea_esp_libres,itoa(esp_libres_f3,texto,10));strcat(linea_esp_libres," ");
```

```
//   printf("la fase 3 de la interseccion 1 tiene %d espacios libres \n",esp_libres_f3);getch();

   esp_libres_f4=(capacidad*6) - (inte1[3][0]+inte1[3][1]+inte1[3][2]+inte1[3][3]+inte1[3][4]+inte1[3][5]);
   strcat(linea_esp_libres,itoa(esp_libres_f4,texto,10));strcat(linea_esp_libres," ");
//   printf("%d %d %d %d %d %d",inte1[3][0],inte1[3][1],inte1[3][2],inte1[3][3],inte1[3][4],inte1[3][5]);
//   printf("la fase 4 de la interseccion 1 tiene %d espacios libres \n",esp_libres_f4);getch();

//   puts(linea_esp_libres);

   // ----------------------interseccion 2 ----------------------------------------------------

   sacar_valores("2-1_0.txt",tiempo,val1,val2,val3,val4);
   inte2[0][0]=atoi(val1) ;inte2[1][0]=atoi(val2);inte2[2][0]=atoi(val3);inte2[3][0]=atoi(val4);
   sacar_valores("2-1_1.txt",tiempo,val1,val2,val3,val4);
   inte2[0][1]=atoi(val1) ;inte2[1][1]=atoi(val2);inte2[2][1]=atoi(val3);inte2[3][1]=atoi(val4);
   sacar_valores("2-3_0.txt",tiempo,val1,val2,val3,val4);
   inte2[0][2]=atoi(val1) ;inte2[1][2]=atoi(val2);inte2[2][2]=atoi(val3);inte2[3][2]=atoi(val4);
   sacar_valores("2-3_1.txt",tiempo,val1,val2,val3,val4);
   inte2[0][3]=atoi(val1) ;inte2[1][3]=atoi(val2);inte2[2][3]=atoi(val3);inte2[3][3]=atoi(val4);
   sacar_valores("2-5_0.txt",tiempo,val1,val2,val3,val4);
   inte2[0][4]=atoi(val1) ;inte2[1][4]=atoi(val2);inte2[2][4]=atoi(val3);inte2[3][4]=atoi(val4);
   sacar_valores("2-5_1.txt",tiempo,val1,val2,val3,val4);
   inte2[0][5]=atoi(val1) ;inte2[1][5]=atoi(val2);inte2[2][5]=atoi(val3);inte2[3][5]=atoi(val4);
   sacar_valores("2-8_0.txt",tiempo,val1,val2,val3,val4);
   inte2[0][6]=atoi(val1) ;inte2[1][6]=atoi(val2);inte2[2][6]=atoi(val3);inte2[3][6]=atoi(val4);
   sacar_valores("2-8_1.txt",tiempo,val1,val2,val3,val4);
   inte2[0][7]=atoi(val1) ;inte2[1][7]=atoi(val2);inte2[2][7]=atoi(val3);inte2[3][7]=atoi(val4);

   esp_libres_f1=  (capacidad*6) - (inte2[0][2]+inte2[0][3]+inte2[0][4]+inte2[0][5]+inte2[0][6]+inte2[0][7]);
   strcat(linea_esp_libres,itoa(esp_libres_f1,texto,10));strcat(linea_esp_libres," ");
   esp_libres_f2=(capacidad*6) - (inte2[1][0]+inte2[1][1]+inte2[1][4]+inte2[1][5]+inte2[1][6]+inte2[1][7]);
   strcat(linea_esp_libres,itoa(esp_libres_f2,texto,10));strcat(linea_esp_libres," ");
   esp_libres_f3=(capacidad*6) - (inte2[2][0]+inte2[2][1]+inte2[2][2]+inte2[2][3]+inte2[2][6]+inte2[2][7]);
   strcat(linea_esp_libres,itoa(esp_libres_f3,texto,10));strcat(linea_esp_libres," ");
   esp_libres_f4=(capacidad*6) - (inte2[3][0]+inte2[3][1]+inte2[3][2]+inte2[3][3]+inte2[3][4]+inte2[3][5]);
   strcat(linea_esp_libres,itoa(esp_libres_f4,texto,10));strcat(linea_esp_libres," ");



// ----------------------interseccion 3 ----------------------------------------------------

   sacar_valores("3-2_0.txt",tiempo,val1,val2,val3,val4);
   inte3[0][0]=atoi(val1) ;inte3[1][0]=atoi(val2);inte3[2][0]=atoi(val3);inte3[3][0]=atoi(val4);
   sacar_valores("3-2_1.txt",tiempo,val1,val2,val3,val4);
   inte3[0][1]=atoi(val1) ;inte3[1][1]=atoi(val2);inte3[2][1]=atoi(val3);inte3[3][1]=atoi(val4);
   sacar_valores("3-10_0.txt",tiempo,val1,val2,val3,val4);
   inte3[0][2]=atoi(val1) ;inte3[1][2]=atoi(val2);inte3[2][2]=atoi(val3);inte3[3][2]=atoi(val4);
   sacar_valores("3-10_1.txt",tiempo,val1,val2,val3,val4);
   inte3[0][3]=atoi(val1) ;inte3[1][3]=atoi(val2);inte3[2][3]=atoi(val3);inte3[3][3]=atoi(val4);
   sacar_valores("3-6_0.txt",tiempo,val1,val2,val3,val4);
   inte3[0][4]=atoi(val1) ;inte3[1][4]=atoi(val2);inte3[2][4]=atoi(val3);inte3[3][4]=atoi(val4);
   sacar_valores("3-6_1.txt",tiempo,val1,val2,val3,val4);
   inte3[0][5]=atoi(val1) ;inte3[1][5]=atoi(val2);inte3[2][5]=atoi(val3);inte3[3][5]=atoi(val4);
   sacar_valores("3-9_0.txt",tiempo,val1,val2,val3,val4);
   inte3[0][6]=atoi(val1) ;inte3[1][6]=atoi(val2);inte3[2][6]=atoi(val3);inte3[3][6]=atoi(val4);
   sacar_valores("3-9_1.txt",tiempo,val1,val2,val3,val4);
```

```
inte3[0][7]=atoi(val1) ;inte3[1][7]=atoi(val2);inte3[2][7]=atoi(val3);inte3[3][7]=atoi(val4);

esp_libres_f1=  (capacidad*6) - (inte3[0][2]+inte3[0][3]+inte3[0][4]+inte3[0][5]+inte3[0][6]+inte3[0][7]);
strcat(linea_esp_libres,itoa(esp_libres_f1,texto,10));strcat(linea_esp_libres," ");
esp_libres_f2=(capacidad*6) - (inte3[1][0]+inte3[1][1]+inte3[1][4]+inte3[1][5]+inte3[1][6]+inte3[1][7]);
strcat(linea_esp_libres,itoa(esp_libres_f2,texto,10));strcat(linea_esp_libres," ");
esp_libres_f3=(capacidad*6) - (inte3[2][0]+inte3[2][1]+inte3[2][2]+inte3[2][3]+inte3[2][6]+inte3[2][7]);
strcat(linea_esp_libres,itoa(esp_libres_f3,texto,10));strcat(linea_esp_libres," ");
esp_libres_f4=(capacidad*6) - (inte3[3][0]+inte3[3][1]+inte3[3][2]+inte3[3][3]+inte3[3][4]+inte3[3][5]);
strcat(linea_esp_libres,itoa(esp_libres_f4,texto,10));strcat(linea_esp_libres," ");

//  puts(linea_esp_libres);

// ----------------------interseccion 4 ----------------------------------------------------------
sacar_valores("4-15_0.txt",tiempo,val1,val2,val3,val4);
inte4[0][0]=atoi(val1) ;inte4[1][0]=atoi(val2);inte4[2][0]=atoi(val3);inte4[3][0]=atoi(val4);
sacar_valores("4-15_1.txt",tiempo,val1,val2,val3,val4);
inte4[0][1]=atoi(val1) ;inte4[1][1]=atoi(val2);inte4[2][1]=atoi(val3);inte4[3][1]=atoi(val4);
sacar_valores("4-5_0.txt",tiempo,val1,val2,val3,val4);
inte4[0][2]=atoi(val1) ;inte4[1][2]=atoi(val2);inte4[2][2]=atoi(val3);inte4[3][2]=atoi(val4);
sacar_valores("4-5_1.txt",tiempo,val1,val2,val3,val4);
inte4[0][3]=atoi(val1) ;inte4[1][3]=atoi(val2);inte4[2][3]=atoi(val3);inte4[3][3]=atoi(val4);
sacar_valores("4-14_0.txt",tiempo,val1,val2,val3,val4);
inte4[0][4]=atoi(val1) ;inte4[1][4]=atoi(val2);inte4[2][4]=atoi(val3);inte4[3][4]=atoi(val4);
sacar_valores("4-14_1.txt",tiempo,val1,val2,val3,val4);
inte4[0][5]=atoi(val1) ;inte4[1][5]=atoi(val2);inte4[2][5]=atoi(val3);inte4[3][5]=atoi(val4);
sacar_valores("4-1_0.txt",tiempo,val1,val2,val3,val4);
inte4[0][6]=atoi(val1) ;inte4[1][6]=atoi(val2);inte4[2][6]=atoi(val3);inte4[3][6]=atoi(val4);
sacar_valores("4-1_1.txt",tiempo,val1,val2,val3,val4);
inte4[0][7]=atoi(val1) ;inte4[1][7]=atoi(val2);inte4[2][7]=atoi(val3);inte4[3][7]=atoi(val4);

 esp_libres_f1=  (capacidad*6) - (inte4[0][2]+inte4[0][3]+inte4[0][4]+inte4[0][5]+inte4[0][6]+inte4[0][7]);
strcat(linea_esp_libres,itoa(esp_libres_f1,texto,10));strcat(linea_esp_libres," ");
esp_libres_f2=(capacidad*6) - (inte4[1][0]+inte4[1][1]+inte4[1][4]+inte4[1][5]+inte4[1][6]+inte4[1][7]);
strcat(linea_esp_libres,itoa(esp_libres_f2,texto,10));strcat(linea_esp_libres," ");
esp_libres_f3=(capacidad*6) - (inte4[2][0]+inte4[2][1]+inte4[2][2]+inte4[2][3]+inte4[2][6]+inte4[2][7]);
strcat(linea_esp_libres,itoa(esp_libres_f3,texto,10));strcat(linea_esp_libres," ");
esp_libres_f4=(capacidad*6) - (inte4[3][0]+inte4[3][1]+inte4[3][2]+inte4[3][3]+inte4[3][4]+inte4[3][5]);
strcat(linea_esp_libres,itoa(esp_libres_f4,texto,10));strcat(linea_esp_libres," ");

//  puts(linea_esp_libres);


// ----------------------interseccion 5 ----------------------------------------------------------
sacar_valores("5-4_0.txt",tiempo,val1,val2,val3,val4);
inte5[0][0]=atoi(val1) ;inte5[1][0]=atoi(val2);inte5[2][0]=atoi(val3);inte5[3][0]=atoi(val4);
sacar_valores("5-4_1.txt",tiempo,val1,val2,val3,val4);
inte5[0][1]=atoi(val1) ;inte5[1][1]=atoi(val2);inte5[2][1]=atoi(val3);inte5[3][1]=atoi(val4);
sacar_valores("5-6_0.txt",tiempo,val1,val2,val3,val4);
inte5[0][2]=atoi(val1) ;inte5[1][2]=atoi(val2);inte5[2][2]=atoi(val3);inte5[3][2]=atoi(val4);
sacar_valores("5-6_1.txt",tiempo,val1,val2,val3,val4);
inte5[0][3]=atoi(val1) ;inte5[1][3]=atoi(val2);inte5[2][3]=atoi(val3);inte5[3][3]=atoi(val4);
sacar_valores("5-13_0.txt",tiempo,val1,val2,val3,val4);
inte5[0][4]=atoi(val1) ;inte5[1][4]=atoi(val2);inte5[2][4]=atoi(val3);inte5[3][4]=atoi(val4);
sacar_valores("5-13_1.txt",tiempo,val1,val2,val3,val4);
```

```
inte5[0][5]=atoi(val1) ;inte5[1][5]=atoi(val2);inte5[2][5]=atoi(val3);inte5[3][5]=atoi(val4);
sacar_valores("5-2_0.txt",tiempo,val1,val2,val3,val4);
inte5[0][6]=atoi(val1) ;inte5[1][6]=atoi(val2);inte5[2][6]=atoi(val3);inte5[3][6]=atoi(val4);
sacar_valores("5-2_1.txt",tiempo,val1,val2,val3,val4);
inte5[0][7]=atoi(val1) ;inte5[1][7]=atoi(val2);inte5[2][7]=atoi(val3);inte5[3][7]=atoi(val4);

esp_libres_f1=  (capacidad*6) - (inte5[0][2]+inte5[0][3]+inte5[0][4]+inte5[0][5]+inte5[0][6]+inte5[0][7]);
strcat(linea_esp_libres,itoa(esp_libres_f1,texto,10));strcat(linea_esp_libres," ");
esp_libres_f2=(capacidad*6) - (inte5[1][0]+inte5[1][1]+inte5[1][4]+inte5[1][5]+inte5[1][6]+inte5[1][7]);
strcat(linea_esp_libres,itoa(esp_libres_f2,texto,10));strcat(linea_esp_libres," ");
esp_libres_f3=(capacidad*6) - (inte5[2][0]+inte5[2][1]+inte5[2][2]+inte5[2][3]+inte5[2][6]+inte5[2][7]);
strcat(linea_esp_libres,itoa(esp_libres_f3,texto,10));strcat(linea_esp_libres," ");
esp_libres_f4=(capacidad*6) - (inte5[3][0]+inte5[3][1]+inte5[3][2]+inte5[3][3]+inte5[3][4]+inte5[3][5]);
strcat(linea_esp_libres,itoa(esp_libres_f4,texto,10));strcat(linea_esp_libres," ");

//  puts(linea_esp_libres);



 // ----------------------interseccion 6 ----------------------------------------------------------

 sacar_valores("6-5_0.txt",tiempo,val1,val2,val3,val4);
 inte6[0][0]=atoi(val1) ;inte6[1][0]=atoi(val2);inte6[2][0]=atoi(val3);inte6[3][0]=atoi(val4);
 sacar_valores("6-5_1.txt",tiempo,val1,val2,val3,val4);
 inte6[0][1]=atoi(val1) ;inte6[1][1]=atoi(val2);inte6[2][1]=atoi(val3);inte6[3][1]=atoi(val4);
 sacar_valores("6-11_0.txt",tiempo,val1,val2,val3,val4);
 inte6[0][2]=atoi(val1) ;inte6[1][2]=atoi(val2);inte6[2][2]=atoi(val3);inte6[3][2]=atoi(val4);
 sacar_valores("6-11_1.txt",tiempo,val1,val2,val3,val4);
 inte6[0][3]=atoi(val1) ;inte6[1][3]=atoi(val2);inte6[2][3]=atoi(val3);inte6[3][3]=atoi(val4);
 sacar_valores("6-3_0.txt",tiempo,val1,val2,val3,val4);
 inte6[0][4]=atoi(val1) ;inte6[1][4]=atoi(val2);inte6[2][4]=atoi(val3);inte6[3][4]=atoi(val4);
 sacar_valores("6-3_1.txt",tiempo,val1,val2,val3,val4);
 inte6[0][5]=atoi(val1) ;inte6[1][5]=atoi(val2);inte6[2][5]=atoi(val3);inte6[3][5]=atoi(val4);
 sacar_valores("6-12_0.txt",tiempo,val1,val2,val3,val4);
 inte6[0][6]=atoi(val1) ;inte6[1][6]=atoi(val2);inte6[2][6]=atoi(val3);inte6[3][6]=atoi(val4);
 sacar_valores("6-12_1.txt",tiempo,val1,val2,val3,val4);
 inte6[0][7]=atoi(val1) ;inte6[1][7]=atoi(val2);inte6[2][7]=atoi(val3);inte6[3][7]=atoi(val4);

 esp_libres_f1=  (capacidad*6) - (inte6[0][2]+inte6[0][3]+inte6[0][4]+inte6[0][5]+inte6[0][6]+inte6[0][7]);
 strcat(linea_esp_libres,itoa(esp_libres_f1,texto,10));strcat(linea_esp_libres," ");
 esp_libres_f2=(capacidad*6) - (inte6[1][0]+inte6[1][1]+inte6[1][4]+inte6[1][5]+inte6[1][6]+inte6[1][7]);
 strcat(linea_esp_libres,itoa(esp_libres_f2,texto,10));strcat(linea_esp_libres," ");
 esp_libres_f3=(capacidad*6) - (inte6[2][0]+inte6[2][1]+inte6[2][2]+inte6[2][3]+inte6[2][6]+inte6[2][7]);
 strcat(linea_esp_libres,itoa(esp_libres_f3,texto,10));strcat(linea_esp_libres," ");
 esp_libres_f4=(capacidad*6) - (inte6[3][0]+inte6[3][1]+inte6[3][2]+inte6[3][3]+inte6[3][4]+inte6[3][5]);
 strcat(linea_esp_libres,itoa(esp_libres_f4,texto,10));strcat(linea_esp_libres," ");
  puts(linea_esp_libres);
 fputs(linea_esp_libres,ar);
  fclose(ar);
  //printf("\n\n Los espacios libres se han escrito\n");
  getch();
  return (0);
  }
```

A.3.Crear archivo de red

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>

int encuentra(char *cadena, char *buscando, int long_cadena, int long_buscando);
void ProduceLineaFase(char tiempo[],int segundos,int NumeroFase);
void ProduceLineaWaut(char LineaWaut[],int segundo);
void CreaArchivoNet(int NumOrigen,int NumDestino,int TiemposAplicar[6][4]);
void LeerTiempos(int tiempo, int TiemposAplicar[6][4]);

int main()
 {
 int i,TiemposAplicar[6][4];
 char EjecutarRed1[530],EjecutarRed2[530],segu[10],DepuraDensi[530],EspaciosLibres[530];

 for (i=2000;i<=4800;i=i+100)
   {
   EjecutarRed1[0]='\0';
   EjecutarRed2[0]='\0';
   itoa(i,segu,10);
   strcat(EjecutarRed1,"sumo -n=red-opti-hasta-");
   strcat(EjecutarRed1,segu);
   strcat(EjecutarRed1,".net.xml -r=rutas1.rou.xml -
a=i.add.xml,1i.add.xml,2.add.xml,22.add.xml,3.add.xml,33.add.xml,4.add.xml,44.add.xml,5.add.xml,55.add.xml,6.
add.xml,66.add.xml,7.add.xml,77.add.xml,8.add.xml,88.add.xml,9.add.xml,99.add.xml,10.add.xml,1010.add.xml,11
-.add.xml,11-11.add.xml -b 0 -e 5000");
   strcat(EjecutarRed2,"sumo -n=red-opti-hasta-");
   strcat(EjecutarRed2,segu);
   strcat(EjecutarRed2,".net.xml -r=rutas1.rou.xml -
a=12.add.xml,1212.add.xml,13.add.xml,1313.add.xml,14.add.xml,1414.add.xml,15.add.xml,1515.add.xml,16.add.x
ml,1616.add.xml,17.add.xml,1717.add.xml,18.add.xml,1818.add.xml,19.add.xml,1919.add.xml,20.add.xml,2020.ad
d.xml,21.add.xml,2121.add.xml,22-.add.xml,22-
22.add.xml,23.add.xml,2323.add.xml,24.add.xml,2424.add.xml,medir-densidad.add.xml -b 0 -e 5000");
 system(EjecutarRed1);
 system(EjecutarRed2);
   //system("sumo -n=red-opti-hasta-2500.net.xml -r=rutas1.rou.xml -
a=1.add.xml,11.add.xml,2.add.xml,22.add.xml,3.add.xml,33.add.xml,4.add.xml,44.add.xml,5.add.xml,55.add.xml,6.
add.xml,66.add.xml,7.add.xml,77.add.xml,8.add.xml,88.add.xml,9.add.xml,99.add.xml,10.add.xml,1010.add.xml,11
-.add.xml,11-
11.add.xml,12.add.xml,1212.add.xml,13.add.xml,1313.add.xml,14.add.xml,1414.add.xml,15.add.xml,1515.add.xml,
16.add.xml,1616.add.xml,17.add.xml,1717.add.xml,18.add.xml,1818.add.xml,19.add.xml,1919.add.xml,20.add.xml,
2020.add.xml,21.add.xml,2121.add.xml,22-.add.xml,22-
22.add.xml,23.add.xml,2323.add.xml,24.add.xml,2424.add.xml,medir-densidad.add.xml -b 0 -e 5000");

   DepuraDensi[0]='\0';
   strcat(DepuraDensi,"depura-densiV2-escribe-un-ciclo ");
   strcat(DepuraDensi,segu);
 system(DepuraDensi);
   //system("depura-densiV2-escribe-un-ciclo 2500");

   EspaciosLibres[0]='\0';
   strcat(EspaciosLibres,"espacios-libresV2 ");
```

```
  strcat(EspaciosLibres,segu);
system(EspaciosLibres);
  //system("espacios-libresV2 2500");

system("cmaple opti-6-intersecciones-.txt");

LeerTiempos(i,TiemposAplicar);   //maple siempre esta escribiendo en tiempo-aplica-al-2000.txt
                                 //y leyendo densi-en-el-2000.txt y esp-libres.txt y tiempos-fases.txt
CreaArchivoNet(i,i+100,TiemposAplicar);

  }

getch();
}//main



//las longitudes se expresan tomando en cuenta el cero
 //busca la cadena "buscando" en la cadena "cadena"
 //regresa la posicion donde se encuentra o -1 si no se encuentra
 int encuentra(char *cadena, char *buscando, int long_cadena, int long_buscando)
   {
   int i,busca=0,ini=0;

   for(i=0;i<=long_cadena;i++)
     {
     ini=i;
     whiie(cadena[i]==buscando[busca])//guardar la primera posicion
        {
        if (busca==long_buscando)
          return(ini);
        busca++;
        i++;
        }
     busca=0;
     i=ini;
     }
  return(-1);
  }

//La funcion ProduceLineaFase se encarga de formar la linea de "phase duration" con el numero de segundos
//especificados, y almacena esta linea en el parametro "tiempo"
void ProduceLineaFase(char tiempo[],int segundos,int NumeroFase)
  {
  char texto[10],phase[40],brake[40],yellow[40];
  int longi;
  tiempo[0]='\0';
  texto[0]='\0';
  if (NumeroFase==1)
     {strcpy(phase,"11111111000000000000000000000000");
     strcpy(brake,"00000000111111111111111111111111");
     strcpy(yellow,"00000000000000000000000000000000");}
  else if (NumeroFase==2)
     {strcpy(phase,"00000000000000001111111100000000");
     strcpy(brake,"11111111111111110000000011111111");
     strcpy(yellow,"00000000000000000000000000000000");}
```

```
else if (NumeroFase==3)
   {strcpy(phase,"00000000111111110000000000000000");
    strcpy(brake,"11111111000000001111111111111111");
    strcpy(yellow,"00000000000000000000000000000000");}
else if (NumeroFase==4)
   {strcpy(phase,"00000000000000000000000011111111");
    strcpy(brake,"11111111111111111111111100000000");
    strcpy(yellow,"00000000000000000000000000000000");}

strcat(tiempo,"    <phase duration=");
longi=strlen(tiempo);tiempo[longi]="";tiempo[longi+1]='\0';
itoa(segundos,texto,10);
strcat(tiempo,texto);//el valor del arreglo//);
longi=strlen(tiempo);tiempo[longi]="";tiempo[longi+1]='\0';
strcat(tiempo," phase=");
longi=strlen(tiempo);tiempo[longi]=""; tiempo[longi+1]='\0';
strcat(tiempo,phase);
longi=strlen(tiempo);tiempo[longi]=""; tiempo[longi+1]='\0';
strcat(tiempo," brake=");
longi=strlen(tiempo);tiempo[longi]="";tiempo[longi+1]='\0';
strcat(tiempo,brake);
longi=strlen(tiempo);tiempo[longi]="";tiempo[longi+1]='\0';
strcat(tiempo," yellow=");
longi=strlen(tiempo);tiempo[longi]="";tiempo[longi+1]='\0';
strcat(tiempo,yellow);
longi=strlen(tiempo);tiempo[longi]="";tiempo[longi+1]='\0';
strcat(tiempo,"/>\n");
}//fin "ProduceLineaFase"

void ProduceLineaWaut(char LineaWaut[],int segundo)
 {

 char texto[20];
 int longi;
 LineaWaut[0]='\0';
 texto[0]='\0';
 itoa(segundo,texto,10);
 strcat(LineaWaut,"<wautSwitch time=");
 longi=strlen(LineaWaut);LineaWaut[longi]="";LineaWaut[longi+1]='\0';
 strcat(LineaWaut,texto);
 longi=strlen(LineaWaut);LineaWaut[longi]="";LineaWaut[longi+1]='\0';
 strcat(LineaWaut," to=");
 longi=strlen(LineaWaut);LineaWaut[longi]="";LineaWaut[longi+1]='\0';
 strcat(LineaWaut,"opti_aplica_");
 strcat(LineaWaut,texto);
 longi=strlen(LineaWaut);LineaWaut[longi]="";LineaWaut[longi+1]='\0';
 strcat(LineaWaut,"/>");
 }

//########crear el archivo .net.xml con los tiempos almacenados en la matriz "tiempos anteriores"###########
//NOTA:falta lo de la matriz

void CreaArchivoNet(int NumOrigen,int NumDestino,int TiemposAplicar[6][4])
  {
  char
LineaWaut[500],tiempo[530],linea[530],CadenitaOrigen[50],CadenitaDestino[50],marcador[50],ciclo[50],segu[10];
```

```
int interseccion,fase,longi,conta,segundos;
FILE *destino,*origen;
//origen=2000;
//destino=2100;
//-----crear el nombre de archivo origen-----
CadenitaOrigen[0]='\0';
strcat(CadenitaOrigen,"red-opti-hasta-");
CadenitaOrigen[15]='\0';
itoa(NumOrigen,segu,10);
strcat(CadenitaOrigen,segu);
strcat(CadenitaOrigen,".net.xml");
///---crear el nombre de archivo destino-------
CadenitaDestino[0]='\0';
strcat(CadenitaDestino,"red-opti-hasta-");
CadenitaDestino[15]='\0';
itoa(NumDestino,segu,10);
strcat(CadenitaDestino,segu);
strcat(CadenitaDestino,".net.xml");
//----------crear el marcador ---------"<!--NNNN-->"
marcador[0]='\0';
strcat(marcador,"<!--");
marcador[4]='\0';
itoa(NumDestino,segu,10);
strcat(marcador,segu);
strcat(marcador,"-->");

destino=fopen(CadenitaDestino,"w");
origen=fopen(CadenitaOrigen,"r");

conta=0;
strcpy(ciclo,marcador);  // inicio del ciclo donde los tiempos se van a establecer
segundos=NumDestino;
//printf("se trabajara con el segundo %d\n",segundos);getch();
fgets(linea,530,origen);
do
  {
  fputs(linea,destino);
  fgets(linea,530,origen);
  }
while (encuentra(linea,ciclo,strlen(linea)-1,strlen(ciclo)-1)==-1); //mientras no llegue al ciclo buscado
//printf("sali del primer while\n");
conta=1;
fase=1;
interseccion=1;
do
   {
   if (encuentra(linea,"phase duration",strlen(linea)-1,13)==-1)
      {fputs(linea,destino);}
   else
      {
      ProduceLineaFase(tiempo,TiemposAplicar[interseccion-1][fase-1],fase); //void ProduceLineaFase(char
tiempo[],int segundos,int NumeroFase)
      fase++;
      if (fase==5){ fase=1; interseccion++;}
      fputs(tiempo,destino);
      conta++;}
```

```
 fgets(linea,530,origen);
 }
while(conta<=24);
//printf("sali del while,valor de conta: %d\n",conta);getch();

fputs(linea,destino);
while(!feof(origen))
 {
 fgets(linea,530,origen);
 if(encuentra(linea,ciclo,strlen(linea)-1,strlen(ciclo)-1)!=-1)
   {
   ProduceLineaWaut(LineaWaut,segundos);//void ProduceLineaWaut(char LineaWaut[],int segundo)
    strcat(linea,LineaWaut);strcat(linea,"\n");
    }
  fputs(linea,destino);
  }
 //printf("Se supone ya es fin de archivo"); getch();
 fclose(origen);
 fclose(destino);
}//fin de la funcion

//esta funcion lee los valores de incremento/decremento del archivo  "tiempo-aplica-al-'tiempo'.txt"
//y coloca los valores definitivos a aplicar en la matriz "TiemposAplicar[][]"
//al inicio lee los tiempos de la fase anterior del archivo "tiempos-fases.txt" y al final escribe en este
//mismo archivo los tiempos a aplicar, que serán los anteriores de la siguiente fase. (el maple los necesita
//en el archivo)
void LeerTiempos(int tiempo, int TiemposAplicar[6][4])
 {
 FILE *ar,*tf;
int i,suma;
char cade1[100],cade2[100],NombreArchivo[50],segu[10];
float i1f1,i1f2,i1f3,i1f4;
int tiempos[6][4],TiemposAnteriores[6][4];

NombreArchivo[0]='\0';
strcat(NombreArchivo,"tiempo-aplica-al-2000.txt");

ar=fopen(NombreArchivo,"r");
//ar=fopen("tiempo-aplica-al-2600.txt","r");   // tiempos de incremento/decremento
tf=fopen("tiempos-fases.txt","r");          // tiempos de la fase anterior

printf("\n - - - - - CICLO %d - - - - -\n",tiempo);
for (i=0;i<6;i++)// para las 6 intersecciones
  {
  //en estas variables los tiempos de incremento/decremento
  fscanf(ar,"%s%s%f %f %f %f",cade1,cade2,&i1f1,&i1f2,&i1f3,&i1f4);
  //en estas variables los tiempos que se apicaban en el ciclo anterior
  fscanf(tf,"%d %d %d %d
",&TiemposAnteriores[i][0],&TiemposAnteriores[i][1],&TiemposAnteriores[i][2],&TiemposAnteriores[i][3]);
  //mostrar los tiempos de incremento/decremento con decimales
  printf("%s %s,                    %.2f, %.2f, %.2f, %.2f\n",cade1,cade2,i1f1,i1f2,i1f3,i1f4);
  //pedir al usuario valores sin decimales
  do{printf("Captura valores enteros: (xx  xx  xx  xx): ");  ///un arreglo bidimensional  para tener poco codigo
   scanf("%d  %d  %d  %d",&tiempos[i][0],&tiempos[i][1],&tiempos[i][2],&tiempos[i][3]);
   suma=tiempos[i][0]+tiempos[i][1]+tiempos[i][2]+tiempos[i][3];
   if(suma!=0)
```

```
    {printf("La suma no es correcta, vuelve a capturar (%d)\n",suma);}}
  while(suma!=0);
  //sumar los tiempos de incremento/decremento a los tiempos de ciclo anterior
  TiemposAplicar[i][0]=TiemposAnteriores[i][0]+tiempos[i][0];
  TiemposAplicar[i][1]=TiemposAnteriores[i][1]+tiempos[i][1];
  TiemposAplicar[i][2]=TiemposAnteriores[i][2]+tiempos[i][2];
  TiemposAplicar[i][3]=TiemposAnteriores[i][3]+tiempos[i][3];
  }//for
fclose(ar);
fclose(tf);

tf=fopen("tiempos-fases.txt","w");
  fprintf(tf,"%d %d %d %d
",TiemposAplicar[0][0],TiemposAplicar[0][1],TiemposAplicar[0][2],TiemposAplicar[0][3]);
  fprintf(tf,"%d %d %d %d
",TiemposAplicar[1][0],TiemposAplicar[1][1],TiemposAplicar[1][2],TiemposAplicar[1][3]);
  fprintf(tf,"%d %d %d %d
",TiemposAplicar[2][0],TiemposAplicar[2][1],TiemposAplicar[2][2],TiemposAplicar[2][3]);
  fprintf(tf,"%d %d %d %d
",TiemposAplicar[3][0],TiemposAplicar[3][1],TiemposAplicar[3][2],TiemposAplicar[3][3]);
  fprintf(tf,"%d %d %d %d
",TiemposAplicar[4][0],TiemposAplicar[4][1],TiemposAplicar[4][2],TiemposAplicar[4][3]);
  fprintf(tf,"%d %d %d %d
",TiemposAplicar[5][0],TiemposAplicar[5][1],TiemposAplicar[5][2],TiemposAplicar[5][3]);
  fclose(tf);
  }//fin de LeerTiempos
```

A.4. Depurar flujos

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>

int inStr(char *pattern, char *str) {
        int i, foundPos = 0;
        while(str[foundPos]!=0) {
                if (str[foundPos]==pattern[0]) {
                        i = 1;
                        while(pattern[i]!=0 && str[foundPos + i]!=0 && str[foundPos + i]==pattern[i]) { i++; }
                        if (pattern[i]==0) {
                                return foundPos;
                        } else if (str[foundPos + i]==0) {
                                return -1;
                        }
                }
                foundPos++;
        }
        return -1;
}

//las longitudes se expresan tomando en cuenta el cero
 //busca la cadena "buscando" en la cadena "cadena"
 //regresa la posicion donde se encuentra o -1 si no se encuentra
 int encuentra(char *cadena, char *buscando, int long_cadena, int long_buscando)
{
int i,busca=0,ini=0;

for(i=0;i<=long_cadena;i++)
  {
  ini=i;
  while(cadena[i]==buscando[busca])//guardar la primera posicion
     {
     if (busca==long_buscando)
       return(ini);
      busca++;
      i++;
     }
  busca=0;
  i=ini;
  }
  return(-1);
  }

void copiarnn(char *destino, char *origen, int ini, int fin)
  {
  int i,desti=0;
  for(i=ini;i<=fin;i++)
    {
    destino[desti]=origen[i];
    desti++;
    }
```

```
 destino[desti]='\0';
 }


void sacar_valores(char nom_archivo[], char val1[], char val2[], char val3[], char val4[], char val5[])
 {
  FILE *ar;
  int sale=0,longi_linea,a,aa,b,c;
  char linea[530];
  ar=fopen(nom_archivo,"r");

  while(!feof(ar) && sale!=1){
fgets(linea,530,ar);
longi_linea=strlen(linea);
a=encuentra(linea,"interval begin=",longi_linea,14);
 if(a!=-1)
   {
  aa=encuentra(linea,"flow=",longi_linea,4);
  b=encuentra(linea,"occupancy=",longi_linea,9);
  copiarnn(val1,linea,aa+6,b-3);

  fgets(linea,530,ar);
  aa=encuentra(linea,"flow=",longi_linea,4);
  b=encuentra(linea,"occupancy=",longi_linea,9);
  copiarnn(val2,linea,aa+6,b-3);


  fgets(linea,530,ar);aa=encuentra(linea,"flow=",longi_linea,4);
  b=encuentra(linea,"occupancy=",longi_linea,9);
  copiarnn(val3,linea,aa+6,b-3);


  fgets(linea,530,ar);aa=encuentra(linea,"flow=",longi_linea,4);
  b=encuentra(linea,"occupancy=",longi_linea,9);
  copiarnn(val4,linea,aa+6,b-3);


  fgets(linea,530,ar);aa=encuentra(linea,"flow=",longi_linea,4);
  b=encuentra(linea,"occupancy=",longi_linea,9);
  copiarnn(val5,linea,aa+6,b-3);
  sale=1;

   }
  }//while
fclose(ar);
 } //fin declaracion de la funcion sacar_valores

//OJO...los valores que se proporcionan en el archivo de salida tienen truncados los decimales.

int main(){
   char encabezado[230],llinea[400],
val1[20],val2[20],val3[20],val4[20],val5[20],tiempo[100],nom_archivo[100],linea_esp_libres[500],texto[20];
   char linea1[200], linea2[200], linea3[200], linea4[200], linea5[200];
   int capacidad, esp_libres_f1,esp_libres_f2,esp_libres_f3,esp_libres_f4;
   float n1,n2,n3,n4,n5;
   FILE *ar;
```

```
ar=fopen("flujos.txt","w");
if (ar==NULL) printf("error al abrir archivo de escritura escritura\n");getch();

linea_esp_libres[0]='\0';
linea1[0]='\0';
linea2[0]='\0';
linea3[0]='\0';
linea4[0]='\0';
linea5[0]='\0';
 strcpy(encabezado,"seg. 16-1 2-1 4-1 7-1 1-2 3-2 5-2 8-2 2-3 10-3 6-3 9-3 15-4 5-4 14-4 1-4 4-5 6-5 13-5 2-5 5-6
                                                                                                    11-6 12-6 3-6 \n");
fputs(encabezado,ar);

// ----------------------interseccion 1 ----------------------------------------------------------

sacar_valores("e1-16-1_0.txt",val1,val2,val3,val4,val5);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-16-1_1.txt",val1,val2,val3,val4,val5);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1,"0 "); strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2,"1000 "); strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3,"2000 "); strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4,"3000 "); strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5,"4000 "); strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");


sacar_valores("e1-2-1_0.txt",val1,val2,val3,val4,val5);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-2-1_1.txt",val1,val2,val3,val4,val5);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");


sacar_valores("e1-4-1_0.txt",val1,val2,val3,val4,val5);
```

```
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);


sacar_valores("e1-4-1_1.txt",val1,val2,val3,val4,val5);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");


sacar_valores("e1-7-1_0.txt",val1,val2,val3,val4,val5);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-7-1_1.txt",val1,val2,val3,val4,val5);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");


// ----------------------------interseccion 2 ----------------------

sacar_valores("e1-1-2_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-1-2_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
```

```
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");

sacar_valores("e1-3-2_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-3-2_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");

sacar_valores("e1-5-2_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-5-2_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");

sacar_valores("e1-8-2_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-8-2_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
```

```
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");



// ----------------------------interseccion 3 ----------------------

sacar_valores("e1-2-3_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-2-3_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");

sacar_valores("e1-10-3_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-10-3_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");

sacar_valores("e1-6-3_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-6-3_1.txt",tiempo,val1,val2,val3,val4);
```

```
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");

sacar_valores("e1-9-3_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-9-3_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");


// ----------------------------interseccion 4 ----------------------
sacar_valores("e1-15-4_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-15-4_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");

sacar_valores("e1-5-4_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
```

```
n5=atof(val5);

sacar_valores("e1-5-4_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");

sacar_valores("e1-14-4_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-14-4_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");

sacar_valores("e1-1-4_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-1-4_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");


// ----------interseccion 5 ----------
```

```
sacar_valores("e1-4-5_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-4-5_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");

sacar_valores("e1-6-5_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-6-5_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");

sacar_valores("e1-13-5_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-13-5_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");
```

```
sacar_valores("e1-2-5_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-2-5_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");



// ----------------------------interseccion 6 ----------------------

sacar_valores("e1-5-6_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-5-6_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");

sacar_valores("e1-11-6_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-11-6_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
```

```
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");

sacar_valores("e1-12-6_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-12-6_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1," ");
strcat(linea2, itoa(n2,texto,10));strcat(linea2," ");
strcat(linea3, itoa(n3,texto,10));strcat(linea3," ");
strcat(linea4, itoa(n4,texto,10));strcat(linea4," ");
strcat(linea5, itoa(n5,texto,10));strcat(linea5," ");

sacar_valores("e1-3-6_0.txt",tiempo,val1,val2,val3,val4);
n1=atof(val1);
n2=atof(val2);
n3=atof(val3);
n4=atof(val4);
n5=atof(val5);

sacar_valores("e1-3-6_1.txt",tiempo,val1,val2,val3,val4);
n1=(n1+atof(val1))/2;
n2=(n2+atof(val2))/2;
n3=(n3+atof(val3))/2;
n4=(n4+atof(val4))/2;
n5=(n5+atof(val5))/2;
strcat(linea1, itoa(n1,texto,10));strcat(linea1,"\n");
strcat(linea2, itoa(n2,texto,10));strcat(linea2,"\n");
strcat(linea3, itoa(n3,texto,10));strcat(linea3,"\n");
strcat(linea4, itoa(n4,texto,10));strcat(linea4,"\n");
strcat(linea5, itoa(n5,texto,10));strcat(linea5,"\n");

fputs(linea1,ar);
fputs(linea2,ar);
fputs(linea3,ar);
fputs(linea4,ar);
fputs(linea5,ar);

 fclose(ar);
 printf("\n\n Los flujos se han escrito\n");
 getch();
}
```

REFERENCIAS

[Breherton, 2004] Bretherton, D., Bodger, M., Baber, N. "SCOOT the future". Road Transport Information and Control Conference, 2004. RTIC 2004. 12th IEE International Conference on Volume , Issue , 20-22 April 2004 Pags.: 301 306

[Castelan, 2006] Cartelar Rodrigo, Kraus Werner, Caponogara Eduardo. "Combinig The TUC Urban Traffic Control Strategy With Bandwidth Maximization" 11th IFAC Symposium on Control in Transportation Systems, Netherlands, 2006.

[Di Febraro, 2006] Di Febraro Angela, Giglio Davide. "Urban Traffic Control In Modular/Switching Deterministic-Timed Petri Nets" 11th IFAC Symposium on Control in Transportation Systems, Netherlands, 2006.

[Dinopoulou, 2002] Dinopoulou V., Diakaki C., Papageorgiu M. "Applications of the Urban Traffic Control Strategy TUC". Proceedings of the 9th Meeting of the EURO Working Group on Transportation. 2002

[Friedrich, 2002] Firedrich B. "Adaptative Signal Control: An overview " Proceedings of the 13th mini-Euro Conference. 2002

[García, 2000] García L., "Diseño e Implementación de una Arquitectura Multiagente para la Ayuda a la Toma de Decisiones en un Sistema de Control de Tráfico Urbano" Tesis de Doctorado, Univesitat Jaume I, mayo del 2000.

[Gartner, 2001] Garther N. "Optimized Policies for Adaptive Control (OPAC)" Workshop on Adaptive Traffic Signal Control Systems Transportation Research Board. Washington D.C. Enero 2001

[Gartner,1990] Gartner N., Assmann S., Lasaga F., Hou D. "MULTIBAND A variable-bandwidth arterial progression scheme" Transportation Research Record No. 1287. Washington, DC, USA. 1990

[Herena, 2007] Herena Ulloa Z., "Un Esquema Multiagentes para la Coordinación de Tráfico Urbano" Tesis de Maestría, CINVESTAV unidad Guadalajara, Octubre del 2007.

[Hewage, 2004] Hewage K., Ruwanpura J. "Optimization of traffic signal light timing using simulation" Winter Simulation Conference. Washington, D.C. 2004.

[Krajzewicz, 2006] Krajzewicz D., Bonert M.,Wagner P,. "The Open Source Traffic Simulation Package SUMO" RoboCup 2006 Infrastructure Simulation Competition, Germany, Bremen. 2006.

[Krauß, 1998] Krauß S., "Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics". Deutsches Zentrum f¨ur Luft– und Raumfahrt. Köln, Abril 1998.

[Little, 1981] Little J., Kelson M., Gartner N., "MAXBAND a versatile program for setting signal on arterires and triangular networks". Massachusetts Intitute of Technology, Department of civil engineering, Univertity of Lowell. Enero 1981.

[López, 2009] López Neri E., "Simulación Distribuida de Tráfico Urbano basada en Sistemas Multi-Agente". Tesis de Doctorado, CINVESTAV unidad Guadalajara, Octubre del 2009.

[Mizuno, 2008] Mizuno Kazunori, Fukui Yukio, Nishihara Seiichi. "Urban Traffic Signal Control Based on Distributed Constraint Satisfaction". 41st. Hawaii International Conference on System Sciences, 2008.

[SCATS, 2010] SCATS, Sydney coordinated adaptative traffic system. 10 de marzo 2010. < http://www.scats.com.au/index.html>

[Slinn, 2005] Slinn M., Matthews P., Guest P. Traffic Engineering Design. 2a. ed. UK. Elsevier. 2005. 2 p. ISBN: 0 7506 5865 7.

[Swida, 2006] Swida P., Kozlak J., Creput J,. " Modeling and Optimization of City Traffic With the Agent Approach". 12th IFAC Symposium. Saint-Etienne, France. 2006

[TRANSYT, 2010] TRL Software. Transist13. 15 de febrero 2010 < http://www.trlsoftware.co.uk/products/detail.asp?aid=4&c=2&pid=66>

[Wegener, 2008] Wegener Axel, Hellbruck Hort, Wewetzer, Christian. "VANET Simulation Environment with Feedback Loop and its Application to Traffic Light Assistance". GLOBECOM Workshops, 2008 IEEE

[Zand, 2007] Zand Arash, Kheyruri Hadi. "Independent Agents for Urban Traffic Control Problem with mobile-Agent Coordination" Department of Computer science, Birjand University, Iran. 2007

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

"2011, Año del Turismo en México"

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Control Adaptativo de Tráfico Urbano Mediante Programación Cuadrática

del (la) C.

Sandra MERCADO PÉREZ

el día 09 de Junio de 2011.

Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

Dr. Félix Francisco Ramos Corchado
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Antonio Ramírez Treviño
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Mario Angel Siller González
Pico
Investigador CINVESTAV 2C
CINVESTAV Unidad Guadalajara