Universidad de la República

PROYECTO DE GRADO

Ingeniería en Computación

Algoritmos Evolutivos en sincronización de semáforos en el Corredor Garzón

Alvaro Acuña
Efraín Arreche

2014

Supervisor: Sergio Nesmachnow

Centro de Cálculo - Instituto de Computación
Montevideo, Uruguay

Algoritmos Evolutivos en sincronización de semáforos en el Corredor Garzón
Acuña, Alvaro - Arreche, Efraín
Proyecto de Grado
Instituto de Computación - Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay, Marzo de 2015

Algoritmos evolutivos en sincronización de semáforos en el Corredor Garzón

Resumen

Este proyecto propone el estudio de la sincronización de semáforos como problema de optimización multiobjetivo, y el diseño e implementación de un algoritmo evolutivo para resolverlo con alta eficacia numérica y desempeño computacional. Se toma como aplicación la sincronización de semáforos en el Corredor Garzón (Montevideo, Uruguay). La cantidad de cruces, calles, tráfico y semáforos lo hace un problema interesante desde el punto de vista de su complejidad. Además, las autoridades responsables admitieron la existencia de problemas relacionados con la sincronización de los semáforos, por lo que todavía hay espacio para la mejora de los tiempos promedio de los viajes en el Corredor Garzón.

El problema de sincronización de semáforos es un problema de optimización NP-difícil por lo que los métodos computacionales exactos solo son útiles en instancias de tamaño reducido. Se buscará mediante un algoritmo evolutivo llegar a una configuración aceptable de los semáforos, maximizando la velocidad media tanto de ómnibus como de otros vehículos. El enfoque seguido comprende la obtención de datos reales relacionados con la red vial, el tráfico y la configuración de los semáforos, y la utilización del simulador de tráfico SUMO para generar los datos requeridos por el algoritmo evolutivo.

El análisis experimental consiste en comparar los resultados numéricos del algoritmo evolutivo, con los valores del escenario base que modela la realidad actual. Complementariamente se desarrolla un escenario alternativo cuyo objetivo es mejorar la velocidad promedio de ómnibus y otros vehículos. Los resultados muestran que el algoritmo evolutivo propuesto logra una mejora de hasta 24.2 % (21.40 % en promedio) en el valor de *fitness* comparando con la realidad actual, mientras que al aplicar el algoritmo sobre el escenario alternativo se obtiene una mejora de hasta 37.1 % (34.7 % en promedio) en el valor de *fitness*.

Palabras clave: Algoritmo Evolutivo, Sincronización de semáforos, Optimización de tráfico, Corredor Garzón

Índice general

Índice de figuras

Índice de cuadros

Índice de Algoritmos

Capítulo 1

Introducción

He llamado a este principio, por el cual cada pequeña variación, si útil, es preservada, con el término de Selección Natural

— Charles Darwin, El origen de las especies

Este capítulo pretende introducir al lector en el contexto general donde se enmarca el problema de sincronización de semáforos en el Corredor Garzón. Inicialmente se describen las motivaciones y el enfoque seguido para el desarrollo del proyecto. A continuación se especifican los objetivos propuestos, las limitaciones y su alcance. Para finalizar se delinea la estructura del documento presentando una breve descripción del contenido de cada capítulo.

1.1. Motivación y contexto

En gran parte del mundo, el parque automotor está creciendo de forma sostenida desde hace varios años. Este crecimiento provoca serios problemas relacionados con la congestión del tráfico, afectando el desarrollo de las ciudades y la calidad de vida de las personas (Bull, 2003). Las congestiones de tráfico producen una progresiva disminución en la velocidad media de circulación, así como también un aumento del consumo de combustible, impactando directamente en la contaminación atmosférica y sonora. Uruguay, en particular su capital Montevideo no escapa a este fenómeno global. Aunque la situación en Montevideo no sea tan crítica como en otras ciudades del mundo, las autoridades municipales han tomado medidas para solucionar este problema, implementando un Plan de Movilidad Urbana que pretende mejorar la eficiencia del transporte público (IMM, 2010).

Uno de los puntos principales del Plan de Movilidad Urbana de la ciudad de Montevideo involucra la construcción de corredores urbanos de tránsito con carriles exclusivos para ómnibus. El primer corredor exclusivo implementado fue el Corredor Garzón, ubicado en la ciudad de Montevideo, con una extensión de 6.5 km, incluyendo 24 cruces semaforizados y vías exclusivas para ómnibus. Desde su inauguración en el año 2012 el Corredor Garzón ha recibido críticas por no cumplir uno de sus principales objetivos: el de agilizar el transporte público. La Intendenta de Montevideo Ana Olivera admitió que la duración de los viajes en el Corredor Garzón aumentó considerablemente. En palabras al Diario El País (2015) indicó:

> "En el momento más crítico llegaron a haber 24 minutos más, lo medimos, porque hubo un momento que se confundió mucho o se partidizó la crítica y nosotros queríamos tener los datos objetivos. Y aquí el dato objetivo era que los ciudadanos de la zona y en particular los de Lezica, no era que se sentían perjudicados, era que habían sido perjudicados. Y nosotros lo asumimos"

Los métodos para optimizar el tráfico se pueden dividir en dos categorías. Por un lado, los métodos enfocados en la modificación de las rutas, por ejemplo agregando nuevas vías de circulación, o ensanchando las existentes. Esta alternativa permite lograr mejoras en la circulación del tránsito, pero como punto negativo exigen un alto costo monetario (Litman, 2009) y disponer de espacio físico para implementar las modificaciones viales. Por otro lado se encuentran los métodos orientados a influir en el comportamiento de los conductores, que incluyen las técnicas para configurar los semáforos y agregar señalizaciones de tránsito, entre otros (McKenney y White, 2013). Estos métodos son muchas veces la única opción viable cuando no se dispone de espacio físico o el capital de inversión que exigen la modificación de las vías de tránsito. Por lo tanto, el estudio de estrategias para la sincronización eficiente de semáforos con el objetivo de mejorar la velocidad promedio de los viajes se presenta como un aporte interesante y necesario para el desarrollo ordenado de las ciudades.

El problema de sincronización de semáforos es un problema de optimización NP-difícil (Yang y Yeh, 1996), por lo que los métodos computacionales exactos son solo útiles en instancias de tamaño reducido. Por este motivo deben utilizarse métodos heurísticos y metaheurísticos para resolver instancias realistas del problema. Los algoritmos evolutivos han demostrado su utilidad en la resolución de problemas complejos, y resultan candidatos interesantes para resolver el problema de sincronización de semáforos.

En el marco del proyecto, se pretende diseñar un algoritmo evolutivo capaz de mejorar la velocidad promedio de los vehículos en la zona del Corredor Garzón, modificando la configuración de sus semáforos, con el objetivo de aportar una solución eficiente e innovadora para el desarrollo de la ciudad y mejorar la calidad de vida de los ciudadanos de Montevideo. Las particularidades del Corredor Garzón lo convierten en un reto complejo desde el punto de vista de la investigación, como consecuencia de la extensión del tramo, de la cantidad de cruces, del número de semáforos, las reglas de exclusividad, los diferentes tipos de tráfico, entre otras.

1.2. Objetivos

Los objetivos que se plantearon al inicio del proyecto incluyeron:

1. Estudio del problema del tráfico y la sincronización de semáforos.
2. Relevamiento de información sobre trabajos relacionados en el ámbito de control de tráfico y sincronización de semáforos.
3. Diseño e implementación de un algoritmo evolutivo multiobjetivo, capaz de resolver eficientemente el problema de sincronización de semáforos en la zona del Corredor Garzón.
4. Creación de instancias realistas del problema, incluyendo un mapa y datos precisos sobre la configuración relativa a semáforos, tráfico y reglas de tránsito obtenidos de la realidad actual.

5. Aplicación de técnicas de computación de alto desempeño para mejorar el rendimiento computacional de la solución implementada.

1.3. Enfoque

La metodología de resolución del problema de sincronización de semáforos en el Corredor Garzón propone como primer punto el modelado del problema. En este modelo se incluye: la creación de instancias realistas basadas en datos recabados de la realidad del Corredor Garzón, y el uso de un simulador de tráfico para analizar el comportamiento de variables útiles para el modelado, como son la velocidad promedio de ómnibus y de otros vehículos. Se plantea como un objetivo importante recabar datos precisos de la realidad, por lo tanto se incluyen tres fuentes para obtener y validar la información: datos recabados in-situ, reuniones con responsables y técnicos de la Intendencia Municipal de Montevideo (IMM), y finalmente información disponible públicamente. Como segundo punto de la metodología se plantea el diseño y desarrollo de un algoritmo evolutivo multiobjetivo que basado en el modelo anterior y modificando la configuración de los semáforos, sea capaz de mejorar la velocidad promedio de ómnibus y de otros vehículos en la zona del Corredor Garzón. El tercer punto propone la aplicación de técnicas de alto desempeño para mejorar la eficiencia computacional de la solución, al considerarse que el problema y las instancias desarrolladas son complejas e insumirán un gran consumo de recursos computacionales.

1.4. Limitaciones y alcance

El alcance geográfico que se plantea comprende la zona del Corredor Garzón, que incluye toda la extensión del Corredor, y dos paralelas a cada lado del mismo. Se pretende desarrollar un modelo preciso de la realidad pero se tiene en cuenta que es usual que algunos elementos de la realidad sean eliminados o simplificados, por tanto se buscará que estas simplificaciones no afecten los resultados de la solución propuesta. El modelado incluye el relevamiento manual de datos en el Corredor Garzón. Debido a los errores intrínsecos que pueden ocurrir en la toma de datos, no se puede asegurar la exactitud de los mismos, por este motivo se realizarán verificaciones con el objetivo de minimizar el impacto que pudieran tener.

1.5. Aportes

Los aportes del proyecto incluyen: estudio del estado del arte y el marco teórico para el tratamiento del problema del tráfico vehicular, creación de instancias realistas del problema utilizando datos recabados in-situ, y el diseño e implementación de un algoritmo evolutivo multiobjetivo que resuelve el problema de sincronización de semáforos en el Corredor Garzón. Este algoritmo utiliza técnicas de computación de alto desempeño y permite mejorar la velocidad promedio de ómnibus y otros vehículos para distintas densidades de tráfico (baja, media y alta). Estos aportes se complementan con el desarrollo de un sitio web que se puede acceder en la siguiente dirección: `http://www.fing.edu.uy/inco/grupos/cecal/hpc/AECG`, los interesados pueden consultar información sobre el proyecto y sus resultados. Adicionalmente, para enriquecer el trabajo, se agrega la redacción de un artículo de síntesis en idioma inglés de 10 páginas,

con el objetivo de tomarlo como base para la redacción de un articulo de investigación para presentar en una conferencia internacional. Finalizando, se destaca el diseño de un póster con la descripción del proyecto, que fue presentado en Ingeniería deMuestra 2014.

1.6. Estructura del documento

El capítulo 2 presenta los fundamentos teóricos necesarios para comprender el resto del trabajo. Comienza describiendo el problema del tráfico vehicular y algunas formas de atacarlo entre las que se encuentra la creación de planes de movilidad y la sincronización de semáforos. El capítulo incluye información sobre corredores urbanos de tránsito y en concreto sobre el Corredor Garzón, así como una descripción acerca de algoritmos evolutivos y simuladores de tráfico. Complementariamente, se ofrece una reseña de los principales trabajos relacionados, haciendo especial foco en algoritmos genéticos para la sincronización de semáforos.

El capítulo 3 explica la estrategia de resolución del problema de sincronización de semáforos implementada en este trabajo. Se presenta el modelado del problema que se basa en: la creación de un mapa digital de la zona del Corredor Garzón compatible con el simulador de tráfico SUMO y el trabajo de campo realizado para obtener datos de la realidad actual, que serán usados en las simulaciones. El capítulo detalla el diseño e implementación del algoritmo evolutivo y el modelo de paralelismo utilizado.

El capítulo 4 presenta el análisis experimental del algoritmo evolutivo, describe los escenarios utilizados en su evaluación y los resultados numéricos obtenidos. Se especifica el escenario que modela la situación actual del Corredor Garzón y el escenario alternativo que presenta modificaciones sobre el escenario anterior con el objetivo de mejorar la velocidad promedio de circulación. Se ofrece un análisis de los resultados obtenidos al variar la función de *fitness* y se realiza un breve estudio de la eficiencia computacional del algoritmo evolutivo.

El capítulo 5 plantea las conclusiones finales del trabajo, tanto sobre la metodología utilizada como de los resultados obtenidos. Para finalizar, el capítulo describe las posibles lineas de investigación para desarrollar en el futuro.

Capítulo 2

Marco Teórico

Este capítulo aborda el marco teórico necesario para comprender el desarrollo de los capítulos posteriores. Se analiza el problema del tráfico en general y las soluciones propuestas para manejarlo incluyendo la construcción de corredores exclusivos para el transporte público. En este contexto se presenta la descripción del Corredor de Garzón y sus problemas. Luego se presenta un relevamiento sobre simuladores de tráfico y la teoría detrás de los algoritmos evolutivos. Para finalizar, se incluye un relevamiento de trabajos relacionados para mostrar y comentar otras variantes del problema y soluciones propuestas en la literatura del área.

2.1. Problema del tránsito vehicular

En gran parte del mundo se está produciendo un crecimiento sostenido del parque automotor, lo cual ocasiona una serie de problemas relacionados con el agravamiento de las congestiones vehiculares que afectan la calidad de vida de las personas (Bull, 2003). Este problema tiene un gran impacto en el desarrollo de las ciudades, por lo que es un componente principal en los planes estratégicos para su crecimiento.

La congestión ocasiona una progresiva merma de la velocidad promedio de circulación, con la consecuencia del incremento en la duración de los viajes y del consumo de combustible. Esto repercute en la contaminación atmosférica y sonora que impacta directamente en la salud de las personas; además, se genera una exigencia en las vías de tránsito que ocasiona un deterioro mayor de calles y rutas.

Uruguay y en particular Montevideo, no escapa a este fenómeno. El aumento del parque automotor en la capital del país está en ascenso constante desde el 2005 (INE, 2014), y según proyecciones el crecimiento seguiría en un promedio de 4.5 % anual hasta el 2020 (BBVA Research, 2013). Este crecimiento viene de la mano con el sostenido aumento de las ventas de vehículos desde el 2003 como se aprecia en la figura 2.1.

Los expertos indican que la situación de congestión ya está instalada en la ciudad y la infraestructura vial no acompasó este crecimiento. Montevideo es la ciudad con más semáforos por automóvil en Latinoamérica, con más de 620 cruces semaforizados, alguno de los cuales no están coordinados (Subrayado, 2013).

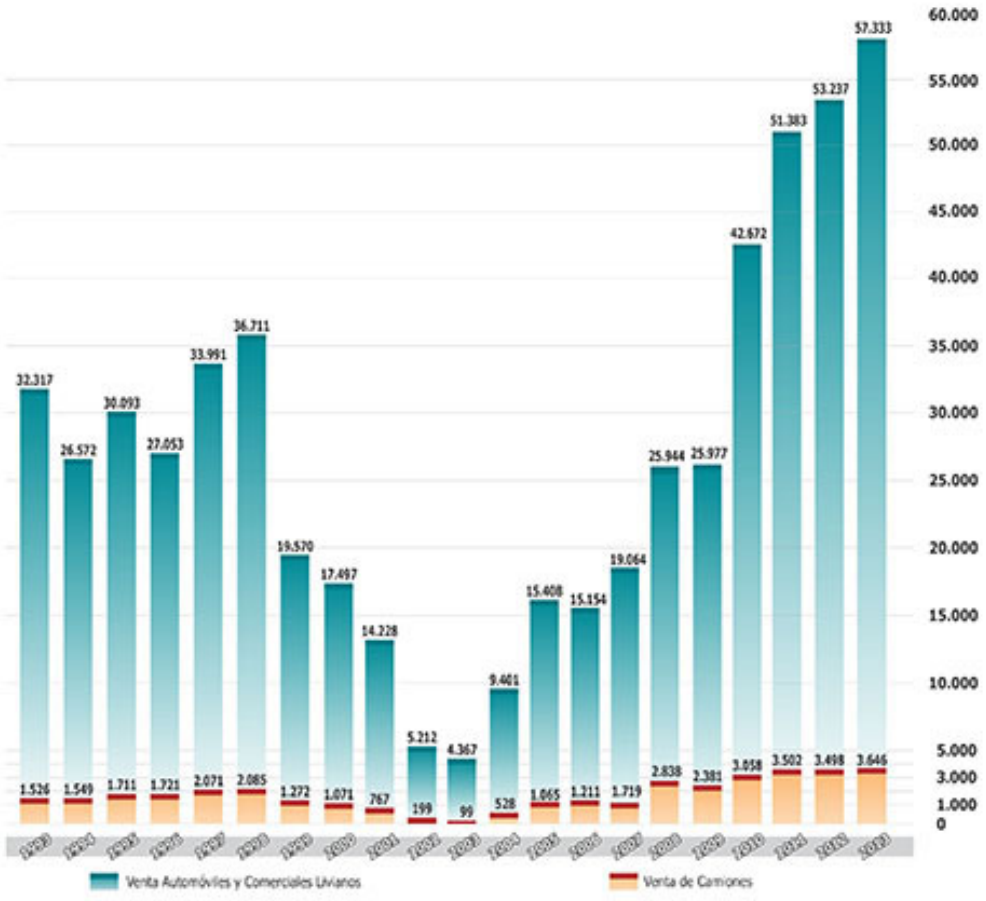

Figura 2.1: Evolución de la venta de automóviles por año. El valor más bajo corresponde al año 2003, y el más alto 2013. Imagen extraída de http://www.autoanuario.com.uy .

En un contexto global, el crecimiento en la circulación de automóviles provoca que baje el nivel de aceptación del transporte público, cuyo servicio en general es ineficiente. Para resolver este problema las autoridades suelen optar por instrumentar sistemas de transporte costosos como los *metros*, pero se ha demostrado que existen otras opciones viables como el BRT (Bus Rapid Transit / ómnibus de tránsito rápido)(Bañobre y Romero, 2009).

En el caso de la ciudad de Montevideo, para solucionar este problema se está implementando el Plan de Movilidad Urbana (IMM, 2010) con el objetivo de mejorar la eficiencia del transporte público y democratizar el acceso al mismo. El sistema de transporte está inspirado en un BRT con la construcción de varios corredores exclusivos en la ciudad. En la siguiente sección se presenta más información sobre los BRT, corredores urbanos y en particular el Corredor de Garzón.

2.2. Corredores urbanos de tráfico

El corredor urbano de tráfico, también llamado *corredor segregado*, se caracteriza por una separación física entre el carril de circulación de los ómnibus y los carriles para el resto del tráfico. Esta es la principal diferencia con un concepto similar llamado *carril*

de sólo bus, en donde se separan los carriles por líneas horizontales pintadas en la calle indicando que sólo pueden circular ómnibus. Los carriles de *sólo bus* suelen ser poco exitosos como medida de agilizar el tráfico de transporte colectivo por falta de control que evite que el tráfico los invada.

En el caso de Montevideo el carril *sólo bus* está sobre la derecha, por lo que los vehículos privados lo invaden al virar a la derecha y los taxis lo necesitan invadir para levantar/dejar pasajeros. Al ubicar los corredores alineados en el medio de las vías de tráfico, se evitan esos problemas.

Las ciudades de países en vías de desarrollo deben estudiar si pueden instalar corredores, su objetivo fundamental es para conseguir mayores velocidades en el sistema. Existen muchas posibilidades de implementación y no todas pueden ser aplicables por el contexto zonal, cultural o de inversión necesaria.

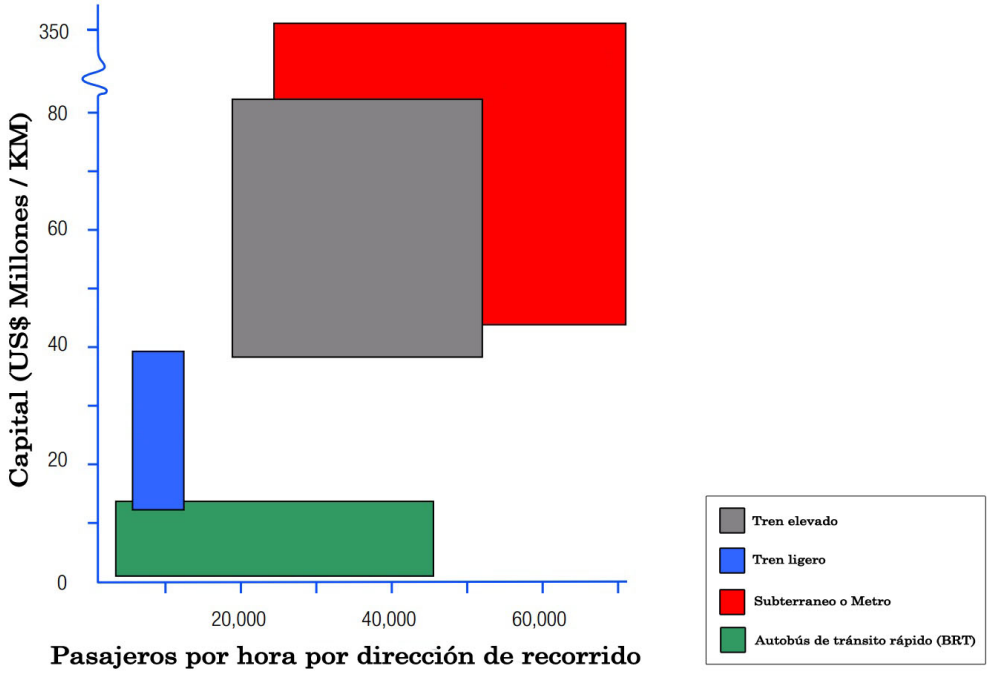

Figura 2.2: Gráfica de costos de transporte en función de la gente transportada.- Imagen original extraída de (Institute for Transportation & Development Policy (ITDP), 2007)

BRT es una solución innovadora, de alta capacidad y de menor costo para el transporte público que puede alcanzar el rendimiento y los beneficios de sistemas ferroviarios, con un costo significativamente menor. Se trata de un sistema integrado de movilidad basado en ómnibus para el transporte de los pasajeros a sus destinos de manera rápida y eficiente. Al mismo tiempo, ofrece la flexibilidad necesaria para satisfacer una variedad de condiciones locales. Los elementos del sistema de BRT pueden ser fácilmente personalizados a las necesidades de la comunidad e incorporan tecnologías de última generación de bajo costo que atraen a más pasajeros y en última instancia ayudan a reducir la congestión de tráfico en general.

Un BRT contiene características similares al tren ligero o al sistema de metro, por lo que es mucho más confiable, conveniente y más rápido que los servicios regulares de ómnibus. Entre sus características principales se destacan el uso de carriles exclusivos

para el ómnibus, con un alineamiento central del carril y un largo mínimo de 3 km. El uso de ómnibus de gran capacidad para transportar un número mayor de personas, la compra de pasajes fuera del ómnibus, que agiliza la entrada de pasajeros y que el ómnibus tenga prioridad sobre otros vehículos en las intersecciones. Con las especificaciones adecuadas, un BRT es capaz de evitar las causas de los retrasos que suelen tener los servicios regulares de ómnibus, como estar atrapado en el tráfico y hacer cola para pagar a bordo.

Para crear una definición común de BRT y calificar los sistemas existentes alrededor del mundo, evaluando su funcionamiento basado en las mejores practicas internacionales existe el standard BRT creado por The Institute for Transportation and Development Policy (ITDP) (2014). Su principal objetivo es brindar una mejor experiencia a los pasajeros, con un costo económico acorde e impacto ambiental positivo. El standard cuenta con un método para calcular el puntaje del corredor y determinar su nivel de calidad.

En la siguiente sección se presenta un análisis exhaustivo de varios puntos presentados en el standard BRT en relación con el caso de estudio del proyecto: el Corredor Garzón.

2.3. Corredor Garzón

El Corredor Garzón fue construido como parte del Plan de Movilidad Urbana que incluye otros corredores en la ciudad de Montevideo. (IMM, 2010). El corredor conecta los barrios de Colón, Sayago, Belvedere y Paso Molino. Tiene una extensión de 6.5km con 24 cruces semaforizados, en donde se encuentran calles importantes como Millán, que conecta con una autopista (Ruta 5), y Bulevar Batlle y Ordoñez, la cual tiene una gran densidad de tráfico. Es importante aclarar que el corredor Garzón no es sólo una conexión de extremo a extremo, ya que se encuentra en una zona densamente poblada cuyos barrios tienen al corredor como la principal vía de movilidad.

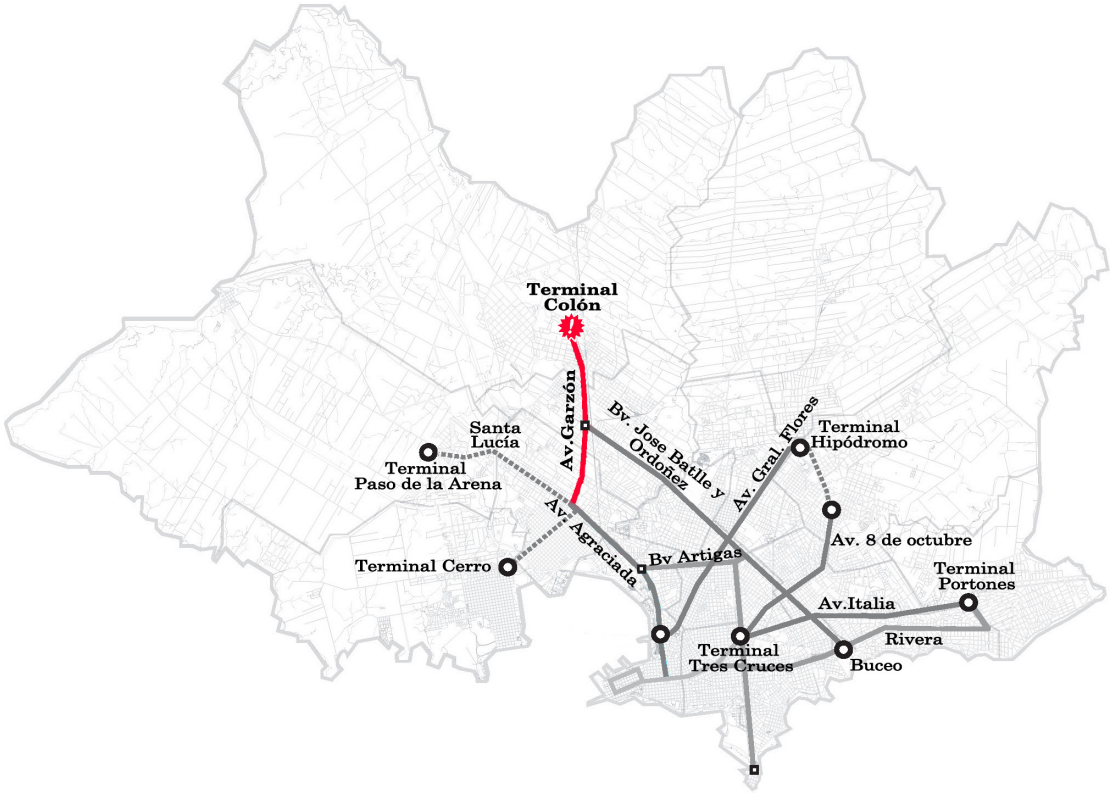

Figura 2.3: Localización del Corredor de Garzón en Montevideo, referenciado en color rojo - Imagen original extraída de www.montevideo.gub.uy

Como se aprecia en la figura 2.4, el corredor consiste básicamente de tres calles paralelas e independientes. Dos calles cuentan con dos carriles de una sola mano y entre medio de éstas se encuentra una calle de doble vía con un carril para cada vía, que es exclusivamente usado por ómnibus urbanos durante el día y por ómnibus urbanos y suburbanos en la noche.

Basándose en el standard antes mencionado, el Corredor de Garzón cumple con la definición de BRT por las siguientes características:

- Largo mínimo del corredor: 3 km de corredor exclusivo.
- Vía de ómnibus dedicada: al tener un 90 % de segregación física dentro del corredor.
- Alineación de la vía del ómnibus: por estar constituido por dos carriles centrales para el ómnibus en medio de los carriles de otros vehículos.
- Tratamiento de intersecciones: por prohibir algunos virajes a la izquierda y el carril de ómnibus tener prioridad en la mayoría de las intersecciones.

Analizando los puntos presentados en el standard como recomendaciones para la implementación de corredores urbanos, hay algunas características que no están totalmente aplicadas al Corredor Garzón, entre las que se destacan:

- Cobrar el boleto fuera del ómnibus: uno de los factores más importantes para mejorar la experiencia del usuario, así como también la velocidad en zonas de mucha

carga, es que existan al menos algunas estaciones (no solo paradas de ómnibus) donde el boleto se cobre (o se use tarjeta de transporte) al entrar a la estación.

- Servicios expresos o limitados: Una forma de mejorar las velocidades de operación consiste en crear líneas que no se detengan en todas las paradas (evitando aquellas paradas con menor demanda de pasajeros) y poner más ómnibus en la calle.

- Carril extra para adelantarse en paradas: este carril resulta crítico en sistemas de transporte colectivo de gran porte para poder manejar los servicios expresos. En sistemas de baja demanda es una buena inversión y en el caso del corredor de Garzón podría permitir que los servicios suburbanos funcionaran durante todo el día por el corredor, mejorando así el transporte público y privado.

- Distancia entre paradas e intersecciones: según el standard, la mínima distancia entre la intersección y la parada es de 26 m, pero idealmente deberían ser 40 m para evitar retrasos. En el Corredor Garzón las paradas están sobre las intersecciones y hay varias paradas donde se detiene más de una línea de ómnibus. Esto puede ocasionar problemas, dado que mientras uno o dos ómnibus ya realizaron la parada y esperan por el semáforo, los demás que lleguen generarán una fila aguardando por detenerse en la parada para levantar a los pasajeros y posiblemente luego, cuando lleguen al cruce, no tendrán la luz verde.

- Tener estaciones centrales o conexión entre paradas: la ausencia de paradas en el centro de los dos carriles de ómnibus hace que el diseño tenga una construcción más cara (hay que hacer dos paradas). El hecho de que no haya una conexión física por la cual el pasajero pueda cambiar de recorrido de un lado al otro sin tener que cruzar la calle, lo hace menos eficiente y más inseguro.

- Distancia entre estaciones: las paradas deberían estar a una distancia de entre 300 m y 800 m, siendo 450 m la distancia óptima tanto para el pasajero como para el transporte.

- Puertas de los ómnibus: con el fin de mejorar el flujo y volumen de pasajeros, sería conveniente contar con dos puertas anchas o más de tres puertas comunes.

- Semáforos: un corredor debe de funcionar al igual que una autopista, en el sentido de que una vez que se entra debería ser posible mantener la velocidad máxima sin tener que detenerse con frecuencia, por lo que no deberían de haber semáforos cerca uno de otro. De haberlos, la sincronización será la clave para minimizar los tiempos de espera.

- Calles paralelas: tener calles paralelas es de vital importancia para un corredor, ya que una de las formas de minimizar los tiempos de espera es prohibir los giros a la izquierda, y una calle paralela provee la facilidad de poder realizarlo.

- Giro a la derecha con luz roja: actualmente, en muchos países se encuentra reglamentada una ley que permite a los conductores doblar a la derecha con luz roja, ya que la misma (a menos que se especifique) es tomada como un cartel de pare *solamente* para doblar a la derecha. En donde esta aprobada esta ley acorta los tiempos de luz verde de las transversales, mejorando así la velocidad promedio en los corredores o calles importantes.

- Mejor calidad de estaciones: las paradas deberían ser a prueba del clima, con puertas corredizas que abren cuando hay un ómnibus (para que nadie caiga al corredor), con información al pasajero en tiempo real, etc. Estas especificaciones no harían al corredor más rápido pero si más seguro, cómodo y confiable.

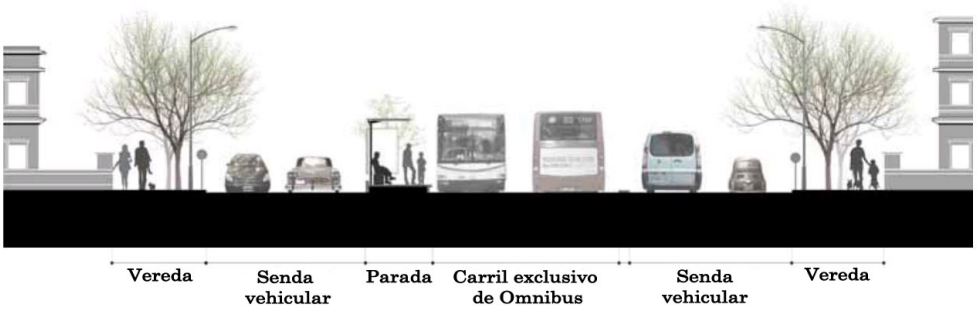

Figura 2.4: Perfil propuesto para el corredor Garzón desde San Quintín a Camino Colman - Imagen original extraída de (IMM, 2010)

Desde su construcción, el corredor ha recibido críticas debido a que su principal objetivo de agilizar el transporte público no fue cumplido. Luego de varios intentos de mejoras al respecto, se ha vuelto a una situación que implica la misma velocidad promedio para el transporte público que existía antes de realizar el corredor (El País, 2015).

Las autoridades municipales admitieron que se han cometido errores en el diseño del corredor, y que no se ha logrado sincronizar los semáforos en las vías de tránsito que lo componen (El País, 2013). Un buen funcionamiento de los semáforos es fundamental para asegurar que el tráfico circule con eficiencia y a la vez aporte seguridad a los peatones. A continuación se tratará específicamente el tema de sincronización de semáforos, que da la motivación para este proyecto.

2.4. Sincronización de semáforos

Los métodos utilizados para la optimización del tráfico tienen como objetivo mejorar el flujo de vehículos en una red vial. Estos métodos se pueden clasificar en dos categorías: influir en el comportamiento de los conductores (mediante la configuración de semáforos, introducción de señalizaciones, etc) o realizar modificaciones en las vías de tráfico (agregar nuevos carriles, ensanchar calles, etc). Las modificaciones de infraestructuras pueden producir mejoras drásticas, pero requieren una inversión monetaria y un espacio físico que muchas veces no está disponible. Por esta razón, los métodos destinados a influir en el comportamiento de los conductores se presentan como una mejor opción o inclusive como única opción en muchos escenarios.

Los métodos para la sincronización de semáforos se encuentran entre los más efectivos para agilizar el tránsito y no generar congestiones. Estas tecnicas permiten aumentar la velocidad promedio de los viajes y mejorar las perspectivas de desarrollo de la ciudad así como la calidad de vida de sus habitantes.

Un concepto importante en el manejo de semáforos es el de *fase*, que se refiere a una configuración especifica de luces de semáforos en una intersección de calles, que permiten el movimiento de ciertos flujos de tráfico. Como se ve en la Figura 2.5 cada intersección

puede tener diferente número de fases y también distintas duraciones. Las fases suelen ser configuradas y establecidas manualmente por técnicos especializados basados en su experiencia, aunque en ciertas ocasiones se utilizan simulaciones computacionales para obtener configuraciones apropiadas.

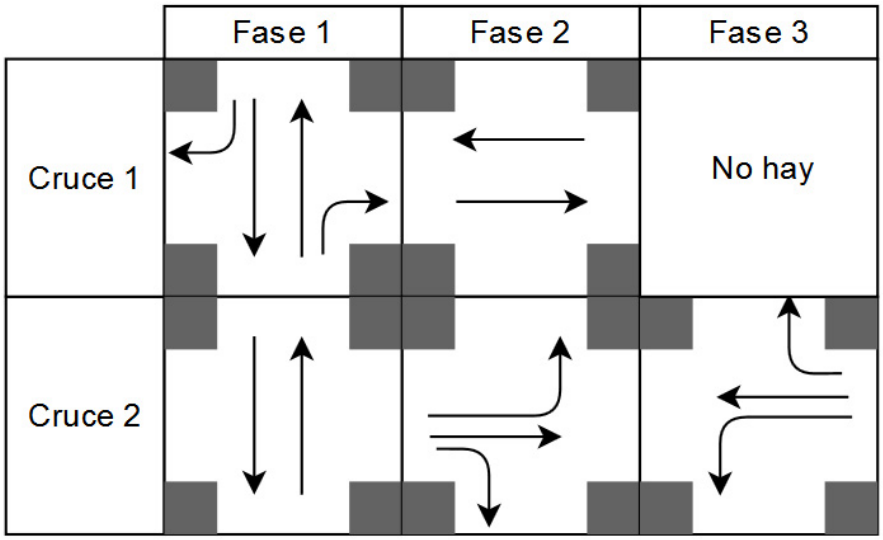

Figura 2.5: Distintas fases de semáforos para dos cruces en una red de tránsito.

Existen tres parámetros importantes a tener en cuenta que determinarán el comportamiento del sistema: la duración de fase, los ciclos de las luces y los valores de *offset*. A continuación se explica cada uno de ellos:

- Duración de fase: Se refiere a la duración en que una configuración especifica de las luces de los semáforos esta activa en una intersección. Un conjunto de las luces estará en verde y otras en rojo, determinando qué vía esta habilitada para cruzar en ese momento. La elección correcta de este parámetro es fundamental; las calles con una densidad de tránsito mayor tendrían que tener más tiempo asignado para cruzar. Realizar la optimización individual de cada intersección no tiene por qué conducir a una solución óptima de toda la red vial, ya que puede ocasionar cambios en el flujo vehicular que afecte otras secciones de la red. Por esta razón es conveniente utilizar un enfoque global al modelar una solución.

- Duración del ciclo: Un ciclo representa un conjunto de fases. En general la duración de un ciclo es la suma de las duraciones de las fases. Como indica su nombre, el ciclo se repetirá una vez que se completa. La duración puede incrementarse o decrementarse para permitir mayor cantidad de repeticiones de las fases.

- *Offset*: Indica en qué fase comienza el ciclo o en qué instante de tiempo, permitiendo que las intersecciones comiencen su ciclo en diferentes momentos. Este concepto es muy importante para sincronizar un flujo de tráfico en lo que se conoce como *línea verde*, en donde los vehículos logran pasar todas las intersecciones sin detenerse.

Los parámetros mencionados anteriormente pueden ser utilizados a la hora de sincronizar los semáforos de una zona, buscando una optimización global de la red vial.

También se debe tener en cuenta la implementación de soluciones seguras desde el punto de vista de las ordenanzas de tránsito, siendo lo más básico que no existan intersecciones donde los flujos vehiculares estén habilitados para cruzar al mismo tiempo.

Los métodos para lograr la coordinación necesaria entre semáforos incluyen van desde simples mecanismos de reloj a sistemas computarizados que se ajustan en tiempo real con la ayuda de sensores en la calle. Estos métodos pueden clasificarse como estrategias de tiempo fijo o de tiempo dinámico auto-ajustable.

Se considera al problema de sincronización de semáforos como un problema de optimización NP-difícil (Yang y Yeh, 1996), por lo que los métodos de resolución exactos solo son útiles para resolver instancias de tamaño reducido. Por esta razón se suelen utilizar diferentes métodos heurísticos, alguno de los cuales se describen en la sección de trabajos relacionados. Entre los métodos más desarrollados y efectivos para resolver el problema están los algoritmos evolutivos, en particular los algoritmos genéticos, que serán explicados a continuación.

2.5. Algoritmos Evolutivos

Los algoritmos evolutivos (AE) son un conjunto de técnicas metaheurísticas para la resolución de problemas complejos que se inspiran en la evolución natural. Los AE trabajan sobre una población de individuos que representan una solución y utilizan mecanismos de selección, reproducción y técnicas para mantener la diversidad para calcular soluciones de buena calidad para el problema (Spears, 2000).

Un AE se describe como una técnica iterativa que busca en cada paso mejorar las soluciones por medio de operadores de exploración y explotación, basado en un criterio predefinido a maximizar o minimizar.

Se pueden destacar cuatro etapas en la ejecución del AE:

- Evaluación: Para cada individuo de la población se determina un valor de aptitud *fitness* en relación a su capacidad para resolver el problema.
- Selección: Proceso en donde se eligen cuales son los individuos que sobrevivirán a la siguiente generación y sobre los cuales se aplicarán los operadores evolutivos.
- Operadores evolutivos: Se aplican combinaciones entre individuos (cruzamiento), y modificaciones aleatorias de individuos(mutación). Los operadores generan nuevos individuos que sustituirán a los existentes en la población.
- Reemplazo: Se produce el recambio generacional, sustituyendo a la antigua población por una nueva que podría tener sobrevivientes de la anterior o solamente nuevos individuados generados en la etapa de aplicación de los operadores evolutivos.

Se han desarrollado enfoques de algoritmos de optimización que aplican estos conceptos, las más relevantes se describen a continuación:

- Estrategias de Evolución: Propuesto por Rechenberg (1973), fue originalmente propuesto como un método de optimización utilizando individuos que codifican números reales en problemas relacionados al diseño de ingeniería. Su característica principal es que utilizan el operador de mutación como motor para la evolución. Sin

embargo, modelos más recientes agregan otro tipo de operadores, incluyendo cruzamiento.

- Programación genética: Intenta generar un programa de computación para resolver una tarea específica, aplicando técnicas evolutivas. Cada individuo puede ser un programa que es representado en forma de árbol, donde cada nodo del árbol tiene una operación y los nodos terminales un operando, de esta manera se facilita la evaluación de expresiones matemáticas. La función de aptitud se refiere a que tan cerca de resolver la tarea se encuentra el individuo (Koza, 1992).

- Algoritmo Genético: Es considerado el mas popular de los AE, dado su versatilidad a la hora de resolver problemas de optimización. El operador de cruzamiento es el principal operador evolutivo siendo el de mutación un operador secundario. Estos operadores se aplican sobre la población de soluciones potenciales en cada generación. En la siguiente sección se presentan en detalle las características de los algoritmos genéticos (AG), que es la técnica que se utiliza en este trabajo para resolver el problema de sincronización de semáforos.

2.5.1. Algoritmos Genéticos

En Algoritmo 1 muestra la formulación básica de un algoritmo genético (AG) que se basa en el esquema general de un AE. Comienza con una población inicial de individuos a los cuales en cada generación se le aplican operadores de cruzamiento y mutación, seleccionando a los mejores en base a su aptitud para resolver el problema. El operador de cruzamiento guía la búsqueda, mientras el operador de mutación se encarga de aportar diversidad a la exploración. Por tanto el objetivo es que con el paso de las generaciones se obtengan mejores soluciones, como se muestra la Figura 2.6 hasta detenerse usando un criterio de parada, ya sea el número de iteraciones o cuando ya no se puede mejorar más la solución. Los trabajos de Goldberg (1989) y Mitchell (1998) brindan más detalles sobre los AE.

Un individuo es una codificación de una solución que resuelve el problema. La población inicial puede generarse aleatoriamente o basándose en heurísticas que utilizan conocimiento especifico sobre el problema. El mecanismo de selección determina cuales individuos son adecuados para integrar la población de la siguiente generación. Este mecanismo utiliza el valor de la función de evaluación o *fitness* para definir que tan buena o apta es una solución en comparación con las demás. En cada iteración, la cual se conoce como generación, se aplican operadores de cruzamiento y mutación sobre los individuos. El cruzamiento permite combinar a dos individuos para obtener otros que potencialmente sean una mejor solución. La mutación aplica cambios aleatorios sobre los individuos. Estos operadores y el mecanismo de selección son probabilísticos, es decir, que su aplicación depende de una una tasa de probabilidad asociada al operador.

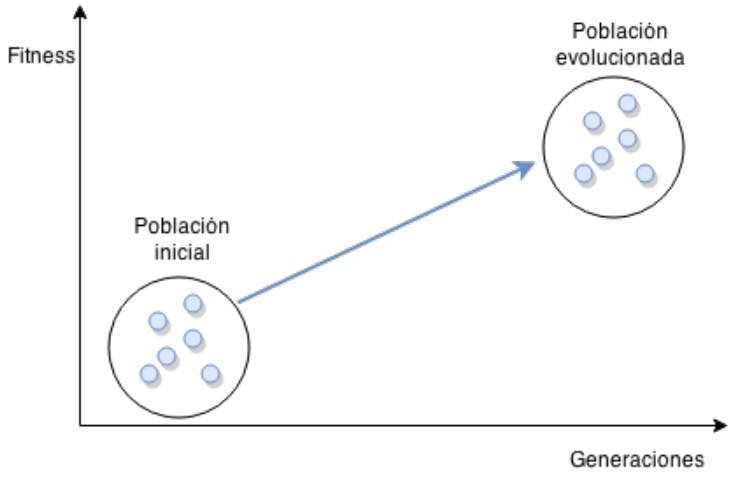

Figura 2.6: Esquema de la evolución del valor del fitness en un algoritmo evolutivo a través de las generaciones.

Por tanto se van seleccionando, combinando y cambiando las mejores soluciones en un proceso, que si el AG está bien diseñado, permite ir obteniendo mejores soluciones. El criterio de parada nos indica cuando termina este proceso, ya sea por que se alcanzó un número de generaciones predefinidos o por que la mejora no es evidente. Al final se devuelve la mejor solución encontrada en todo el proceso.

A continuación se presentan las principales características de los AG.

Esquema del algoritmo

El esquema básico de funcionamiento se presenta en el Algoritmo 1:

Algoritmo 1 Algoritmo Genético

```
1: Inicializo( Pob(0))
2: generacion = 0
3: while No llegue al criterio de parada do
4:    Evaluar Pob(generacion)
5:    Padres = Seleccionar(Pob(generacion))
6:    Hijos = Cruzamiento(Padres) y Mutacion(Padres)
7:    NuevaPob = Reemplazar Pob(generacion) con Hijos
8:    generacion++
9: end while
10: return  Mejor solución
```

Representación de soluciones

Los AG no trabajan directamente sobre las soluciones del problema, sino que utilizan una abstracción de las soluciones llamada cromosoma. Un cromosoma es un vector de genes donde el valor de cada gen se denomina alelo; nombres inspirados en la evolución natural biológica. En general los AG codifican las soluciones utilizando un vector de números binarios o reales de largo fijo.

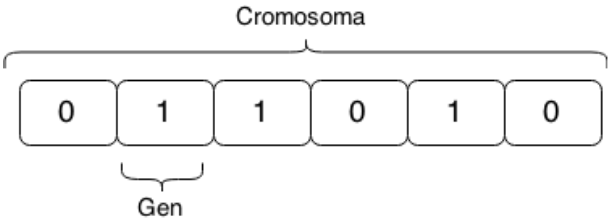

Figura 2.7: Representación binaria de un cromosoma.

Función de Evaluación

La función de evaluación indica qué tan bueno es un individuo para resolver el problema en cuestión, utilizando un valor conocido como *fitness*, por lo que también es llamada función de *fitness*. Este valor se utiliza para definir cuales son los mejores individuos y de esta forma guiar la exploración hacia la región que incluya las mejores soluciones. En general, la función de fitness es la que consume la mayor cantidad de tiempo del algoritmo genético, en comparación con los demás operadores.

Operador de Selección

Existen diversos operadores de selección, cuya función es mantener las mejores características de los individuos en las siguientes generaciones. Entre ellos se encuentran:

- Selección proporcional: Elige aleatoriamente individuos donde la probabilidad de selección es proporcional al valor del *fitness*. Los mejores individuos son elegidos con mayor probabilidad, pero los peores individuos también pueden ser elegidos, lo que permite mantener la diversidad en la población. Sea N la cantidad de individuos en la población y f_i el valor de *fitness* del i-ésimo individuo, la probabilidad asociada a su selección esta dada por la la ecuación 2.1 .

- Torneo: Se eligen aleatoriamente un determinado número de individuos y se selecciona un subconjunto de los individuos con mejor *fitness*.

- Rango: Se ordenan los individuos por el valor de *fitness* y se selecciona un determinado porcentaje(rango) de los mejores individuos.

$$F(x) = p_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{2.1}$$

Operador de cruzamiento

La función del operador de cruzamiento es combinar individuos con el objetivo de preservar las mejores características de los progenitores y así lograr construir mejores soluciones. El operador de cruzamiento se aplica de acuerdo a una tasa de probabilidad prefijada.

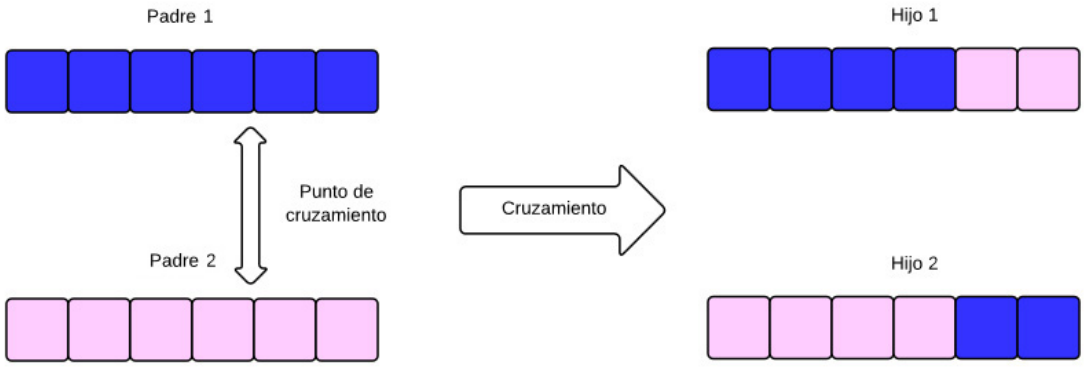

Figura 2.8: Cruzamiento de un punto

En general, los operadores de cruzamiento se pueden clasificar en:

- Cruzamiento de un punto: A partir de dos padres se selecciona un punto al azar de los cromosomas obteniendo dos trozos que se combinan para obtener dos hijos. Se explica en la Figura 2.8
- Cruzamiento multipunto: El método anterior se puede generalizar para obtener más puntos de corte y más recombinaciones.
- Cruzamiento uniforme: Para cada posición en el cromosoma se intercambian genes según una tasa de probabilidad, este mecanismo se muestra en la Figura 2.9 .

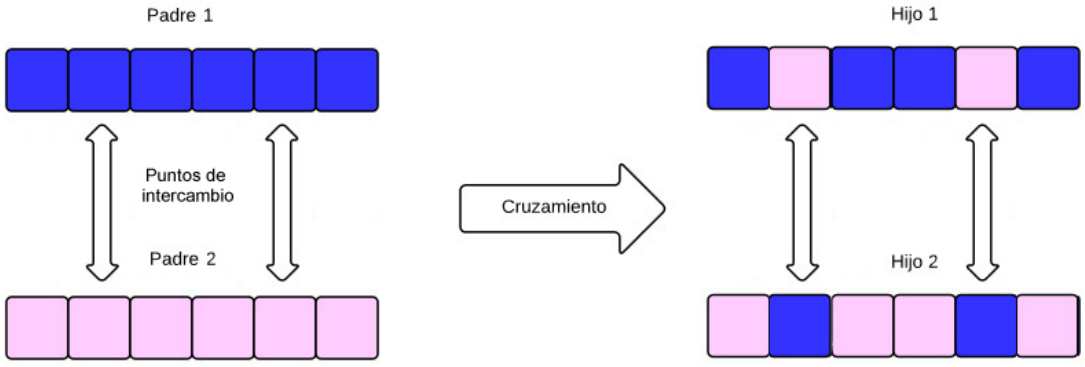

Figura 2.9: Cruzamiento uniforme

Operador de mutación

El operador de mutación es el método utilizado para modificar un individuo, con el objetivo de mantener y mejorar la diversidad y otorgar al AG un mecanismo que le permita escapar de óptimos locales. En general la mutación aplica una modificación aleatoria en el cromosoma, por ejemplo, en una representación binaria se invertiría aleatoriamente el valor de un alelo, como se muestra en la Figura 2.10.

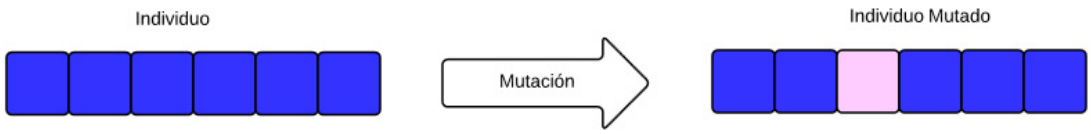

Figura 2.10: Mutación por inversión binaria

Criterio de reemplazo

Luego de que se aplica el cruzamiento se insertan nuevos individuos que aumentan la población original. Por lo tanto el operador de reemplazo indica cual es el criterio que se debe tomar para crear una nueva población con una cantidad de individuos igual a la original. Se podría reemplazar todos los padres por los hijos, o seleccionar cuales reemplazar, entre otros criterios.

Criterio de parada

El criterio de parada indica cuándo debe terminar el algoritmo. Puede definirse un número fijo de generaciones o determinar si el mejor valor de fitness se mantiene relativamente constante durante un número determinado de generaciones entre otros criterios. El objetivo será encontrar un compromiso entre un buen resultado y un tiempo acorde de ejecución, ya que el algoritmo no arrojará un valor exacto sino una buena aproximación.

2.5.2. Algoritmos evolutivos multiobjetivo

Los problemas de optimización multiobjetivo trabajan sobre un espacio multidimensional de funciones y no tienen una única solución. Por este motivo, el significado de óptimo cambia. Una solución es un *óptimo de Pareto* si ninguna de las funciones objetivo puede mejorar su valor sin degradar otro de los valores objetivo. Todas estas soluciones son consideradas igualmente buenas, ya que los vectores no se pueden ordenar completamente y representan diferentes valores de compromiso entre las funciones objetivo. Al conjunto de los valores funcionales de los óptimos de Pareto se les llama frente de Pareto.

Existen algoritmos evolutivos para resolver problemas de optimización multiobjetivo. Éstos son los llamados MOEA, por sus siglas en inglés *MultiObjetive Evolutionary Algorithm*. Sus propósitos son aproximarse al frente de Pareto y lograr obtener una gama de diferentes compromisos entre las funciones a optimizar para luego poder tomar la decisión de cual elegir.Para un análisis más detallado de los AE multiobjetivo se recomienda el trabajo de Deb (2001).

El Algoritmo 2 describe el esquema básico de un MOEA, donde se aprecia que existen dos operadores característicos: el operador de diversidad y el operador de asignación de *fitness*. El primero se aplica para evitar la convergencia a un sector en particular del frente de Pareto, mientras que el segundo intenta brindar mayor chance de perpetuar a los individuos con mejores características.

Los MOEA se pueden clasificar por el método de asignación de fitness. Por un lado se tienen los que no son basados en Pareto, que utilizan métodos sencillos de asignación de fitness y son adecuados cuando el problema tiene no más de tres funciones objetivo.

Algoritmo 2 Algoritmo Evolutivo MultiObjetivo. En rojo se indican las diferencias con el algoritmo evolutivo genérico.

```
1: Inicializo( Pob(0))
2: generacion = 0
3: while No llegue al criterio de parada do
4:    Evaluar Pob(generacion)
5:    Operador Diversidad (Pob(generacion))
6:    Asignar Fitness (Pob(generacion))
7:    Padres = Seleccionar(Pob(generacion))
8:    Hijos = Cruzamiento(Padres) y Mutacion(Padres)
9:    NuevaPob = Reemplazar Pob(generacion) con Hijos
10:   generacion++
11: end while
12: return Frente de Pareto
```

Un mecanismo popular entre los que no utilizan dominancia de Pareto es la combinación lineal de los objetivos. Por otro lado, los metodos de asignacion de fitness que se basan en Pareto dominancia de PAreto utilizan explícitamente la dominancia de Pareto al asignar el fitness.

2.5.3. Algoritmos evolutivos paralelos

Los problemas complejos suelen requerir una alta demanda computacional, por lo que paralelizar el algoritmo evolutivo es útil para lograr tiempos de ejecución menores. Pero este no es el único objetivo que se puede conseguir con un AE paralelo, entre otros existen: encontrar soluciones alternativas al mismo problema, búsqueda más eficiente aún sin *hardware* paralelo, facilidad en la cooperación con otros métodos y búsqueda paralela de múltiples puntos en el espacio (Alba y Tomassini, 2002).

En el caso de los algoritmos genéticos, gran parte del tiempo se ocupa en la etapa de evaluación de *fitness*, por esta razón es un buen método distribuir la carga en varios procesadores para que las evaluaciones se realicen en paralelo.

Un modelo muy utilizado es el *maestro-esclavo*. El proceso maestro es el encargado de ejecutar los operadores básicos del algoritmo y distribuir a procesos esclavos la evaluación de la función fitness para un conjunto de individuos. El esclavo devuelve el resultado y luego el maestro es el encargado de continuar con la evaluación, ejecutando los operadores. De este modo aumenta la eficiencia computacional del algoritmo ya que una de las funciones más costosas es distribuida entre varios procesos que ejecutan concurrentemente.

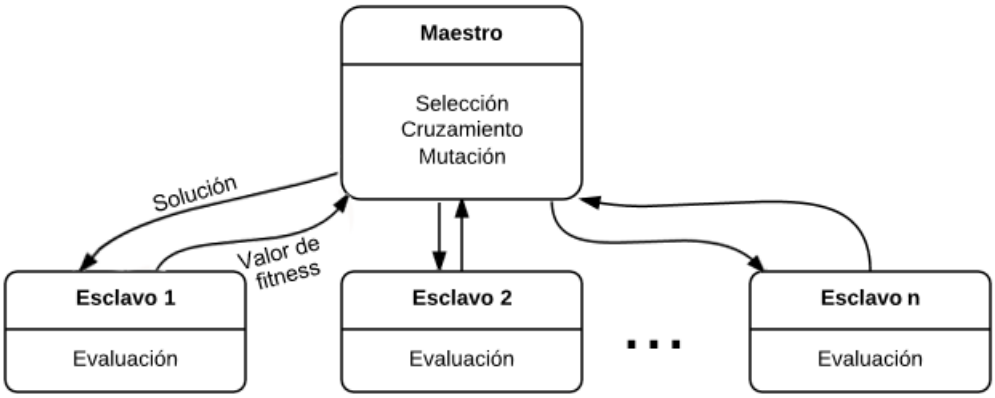

Figura 2.11: Modelo Maestro-Esclavo

2.5.4. Implementación y entornos de desarrollo

A la hora de implementar un algoritmo genético es importante tener en cuenta ciertos aspectos que dotarán a la solución de una mayor flexibilidad, eficiencia y robustez. En general se recomienda usar el paradigma de orientación a objetos, ya que aporta múltiples ventajas entre las que se encuentra: reusabilidad del código, abstracción de problemas, modularidad y legibilidad. Sin embargo, la orientación a objetos esto puede ocasionar pérdidas de eficiencia computacional, por lo que hay que balancear estos aspectos.

En el trabajo de Alba y Cotta (1997) se indican con más detalle los elementos importantes de una implementación de AG, entre ellas se destacan: no utilizar tamaño fijo para las poblaciones pues afectan la flexibilidad, evitar las múltiples evaluaciones de una misma solución, dar al usuario la posibilidad de extender las funcionalidades de manera modular, al implementar bibliotecas genéricas usar lenguajes orientados a objetos que permiten su extensión en forma sencilla. Por estas razones se han desarrollado entornos de desarrollo o*frameworks* para trabajar en la resolución de problemas utilizando algoritmos evolutivos y otras técnicas metaheurísticas. Los *frameworks* encapsulan de manera transparente al usuario los aspectos antes mencionados, lo que permite un desarrollo más rápido, sencillo y seguro.

Existe una gran variedad de *frameworks* para AE; a continuación se describen algunos de los más populares junto con sus principales características.

- JMetal: Es un *framework* orientado a objetos basado en Java para la resolución de problemas de optimización multiobjetivo utilizando metaheurísticas. Su arquitectura permite experimentar con técnicas embebidas y también desarrolladas por el usuario. Entre sus características destacan que al estan implementado en Java puede ser usado tanto en sistemas Windows como Linux. Posee una amplia variedad de algoritmos multiobjetivo listos para usar, y también algoritmos paralelos. Está en constante desarrollo y posee una documentación detallada (Durillo y Nebro, 2011).

- Galib: Es una biblioteca escrita en C++ que incluye objetos y herramientas útiles para la resolución de problemas usando algoritmos genéticos. Es gratuita y de código abierto, pudiendo ser ejecutada tanto en Linux como en Windows. No tiene un desarrollo sostenido, siendo su última actualización en el año 2000, esto no quita que todavía se siga usando por su sencillez y flexibilidad (Wall, 1996).

- Paradiseo: Es un *framework* orientado a objetos que utiliza C++. Su objetivo es el diseño de metaheurísticas tanto paralelas como distribuidas. Cuenta con algoritmos evolutivos, búsquedas locales y optimización basada en enjambres. Funciona tanto en plataformas Windows como Linux y posee varios módulos destinados a extender sus funcionalidades. Tiene una buena documentación y su ultima versión data del año 2012 (Cahon et al., 2004).
- Open Beagle: Es un *framework* en C++ utilizando algoritmos evolutivos. Brinda un entorno de alto nivel para trabajar con distintas técnicas, soportando programación evolutiva, algoritmos genéticos y estrategias evolutivas. Su arquitectura permite utilizar los principios de la programación orientada a objetos para lograr un código recusable y eficiente. Sus características apuntan a brindar un entorno amigable, eficiente, multi-plataforma y gratuito (Gagné y Parizeau, 2002).
- Mallba: Es una biblioteca escrita en C++ que brinda esqueletos de algoritmos exactos, metaheurísticas, e híbridos para la resolución de problemas de optimización. Mallba maneja el paralelismo de forma simple para el usuario. Posee una arquitectura flexible y extensible lo que permite agregar nuevos esqueletos de forma simple. No cuenta con documentación extensa pero si con varios ejemplos completos de implementaciones de los algoritmos. Mallba funciona tanto en ambiente Linux como Windows (Alba et al., 2006).
- Malva: Surge como un *branch* de la biblioteca Mallba, con modificaciones para ser ejecutado en ambientes actuales, por lo que cuenta con las mismas características aunque al estar en fase de desarrollo soporta solo algunos algoritmos, entre ellos algoritmos genéticos (Fagúndez, 2014).

Los *frameworks* son una herramienta importante a la hora de desarrollar un algoritmo evolutivo. Al enfocarse en el problema de la sincronización de semáforos también serán necesarias otras utilidades entre las que se encuentra los simuladores de tráfico, los cuales serán explicados en la siguiente sección.

2.6. Simulación de tráfico

Los simuladores de tráfico son programas que simulan el movimiento del flujo vehicular sobre una red terrestre, marítima o aérea. Son usados en proyectos de investigación, estudio de congestiones y análisis de impacto de obras. Existen varias razones para optar por esta herramienta, entre las que se encuentra: la rapidez en la obtención de resultados y el costo asociado de implementación.

Los simuladores se pueden dividir en dos grandes categorías, macroscópicos y microscópicos. En algunos casos se considera una tercera categoría híbrida de estas dos llamada mesoscópicos. En los simuladores *macroscópicos* el tráfico es modelado como un flujo continuo y es descrito de manera agregada usando características como la velocidad o densidad del flujo. En los simuladores *microscópicos* el tráfico se considera compuesto de partículas discretas. Cada partícula es actualizada según las propiedades de la red en ese momento, como limites de velocidad, vehículos cercanos y caminos a seguir. Los simuladores microscópicos utilizan un modelo de decisiones del conductor, lo que permite crear una distribución heterogénea del comportamiento de los vehículos. En general, los simuladores *mesoscópicos* representan a los individuos con alto nivel de detalle pero sus

interacciones y actividades con un bajo nivel de detalle, por ejemplo, agrupando vehículos en paquetes que se mueven por una red, considerándolos como una sola entidad.

Se considera que las simulaciones microscópicas se aproximan más a la realidad y que obtienen un nivel de granularidad mayor. Estas características pueden ser útil cuando se asignan propiedades sobre cada vehículo y se quiere observar el comportamiento cuando éstas cambian. Sin embargo, no implica que se dejen de usar simulaciones macroscópicas pues aunque no poseen tanto detalle si son más rápidas y en determinadas circunstancias podrían ser una mejor opción.

En la actualidad existe una gran variedad de simuladores disponibles, que se encuentran en las diferentes categorías que fueron mencionadas anteriormente. A continuación se ofrece una breve descripción de alguno de los simuladores presentados en el trabajo de Kotushevski y Hawick (2009) que presenta una lista detallada de simuladores de tráfico.

- SUMO: Simulador abierto, portable, microscópico diseñado para soportar grandes redes de tránsito. Es de los más populares, y utiliza una serie de archivos de configuración para representar las rutas, los vehículos y el tráfico (Krajzewicz et al., 2002).

- QuadStone Paramics Modeller: Simulador modular y microscópico capaz de modelar un amplio rango de problemas de tránsito y transporte.

- Aimsun: Paquete de simulación que integra varios tipos de modelos de transporte, por ejemplo herramientas para el tráfico estático y un simulador microscópico

- Trafficware SimTraffic: Es un simulador que forma parte del paquete *Trafficware's Syncro Studio*, que cuenta con una herramienta para la sincronización de semáforos.

- Corsim Trafvu: Parte del paquete *Tsis Corsim*, presenta las animaciones y los gráficos estáticos para las redes de tráfico utilizando *Corsim* como entrada.

Cabe destacar que de los cinco simuladores de tráfico mencionados anteriormente, solamente SUMO es abierto y gratuito, el resto son programas propietarios.

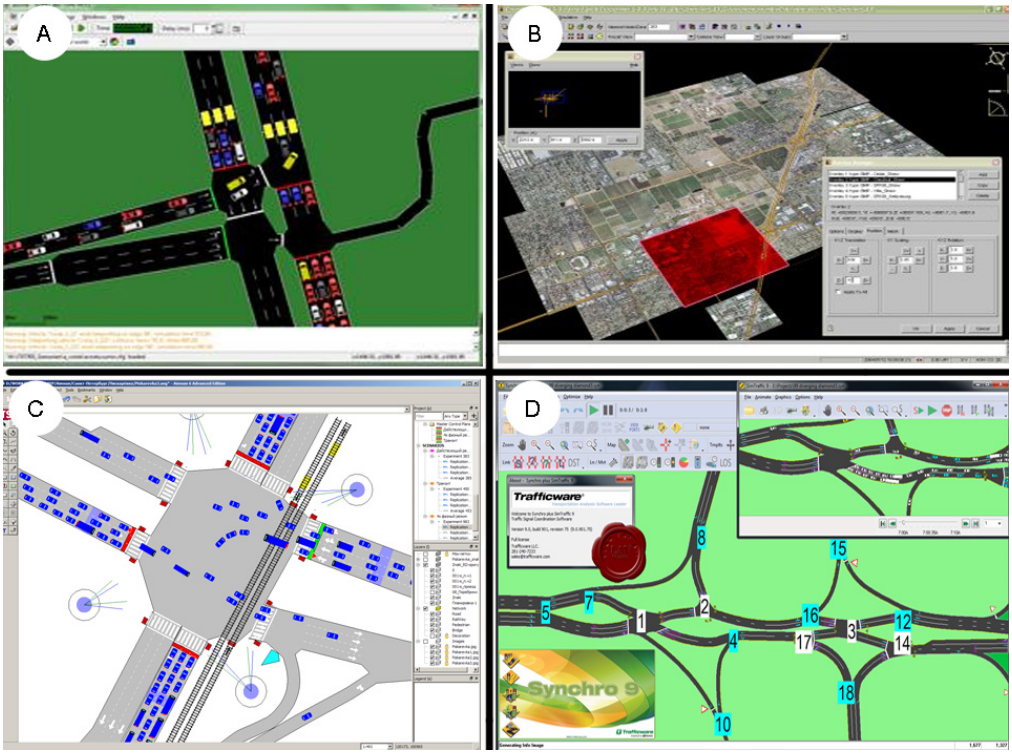

Figura 2.12: Algunos simuladores de tráfico. (A) Sumo, (B) QuadStone Paramics, (C) Aimsun, (D) Trafficware SimTraffic

Los simuladores requieren de la creación de una *red de tránsito*. En general, la red refiere a las propiedades que tendrán las vías de tránsito, por ejemplo la cantidad de carriles, el limite de velocidad, la ubicación, el largo, las conexiones entre calles, etc. Algunos simuladores obtienen ésta entrada desde archivos de texto, lo cual puede ser un proceso lento y propenso a errores; otros pueden importar redes viales de servicios, como Open Street Map (Haklay y Weber, 2008). También necesitan conocer las *rutas seguidas por los vehículos*. Los simuladores pueden tomar esta información de manera explicita, indicando para cada vehículo la ruta seguida, o utilizar otros métodos dinámicos indicando sólo los puntos de inicio y final del recorrido, cuya ruta será generada en tiempo de simulación. El procesamiento manual para generar los recorridos vehiculares puede ser un proceso verdaderamente complejo, por este motivo muchos simuladores brindan herramientas destinadas a facilitar esta tarea.

Un aspecto importante a tener en cuenta es la salida que genera el simulador, ya que es fundamental a la hora de analizar resultados y sacar conclusiones. En general, la información básica reportada incluye: la velocidad y tiempos de recorrido, aunque también puede contener más detalles incluyendo: la duración de detenciones, la densidad de tráfico, uso de combustible y/o la cantidad de emisiones contaminantes.

2.7. Trabajos relacionados

La investigación del estado del arte se realizó con dos objetivos en mente, el primero fue analizar las distintas soluciones que existen actualmente para el problema de sincronización de semáforos y el segundo fue para encontrar nuevas prácticas, algoritmos o

utilidades que pudieran fortalecer la solución a implementar.

El problema del tráfico optimizando las luces de los semáforos se puede resolver por diferentes métodos incluyendo redes neuronales (López et al., 1999), lógica difusa (Lim et al., 2001), redes de petri (Di Febbraro et al., 2002), entre otros. El número de artículos encontrados en el relevamiento de trabajos relacionados fue abundante y los tipos de propuesta fueron variadas, por este motivo se decidió enfocar la búsqueda en los trabajos que proponen soluciones cercanas al proyecto propuesto. En particular se relevaron aquellos trabajos que utilizan algoritmos genéticos.

Una reseña de los principales trabajos relacionados se presenta a continuación:

- J. J. Sanchez, M. Galan, y E. Rubio. Genetic algorithms and cellular automata: a new architecture for traffic light cycles optimization. En *Evolutionary Computation, 2004. CEC2004. Congress on*, volumen 2, p. 1668–1674. IEEE, 2004

  Este trabajo se basa en tres ideas fundamentales: El uso de algoritmos genéticos para la sincronización de los semáforos, la simulación de autómatas celulares para la función de evaluación del tráfico y un cluster para realizar ejecuciones del algoritmo en paralelo. El caso de estudio presentado es pequeño, con 5 calles de 2 vías que se intersectan. La codificación del cromosoma utiliza un vector de números enteros, donde se codifica para cada intersección cuál es la calle habilitada en cada ciclo. El AG utiliza una estrategia de selección elitista, donde los dos mejores individuos se clonan a la siguiente generación y el resto son generados aplicando un operador de cruzamiento de dos puntos.

  Para la evaluación del *fitness*, se considera el tiempo que transcurre desde el momento en que un vehículo entra en la red hasta que sale. Se ejecuta sobre un cluster en forma paralela con una estrategia maestro-esclavo. El maestro envía los cromosomas a los esclavos para que evalúen y devuelven el resultado, luego el maestro se encarga de generar la siguiente población. Se comparan los resultados obtenidos por el algoritmo con una simulación aleatoria y con una fija, obteniendo la solución propuesta mejores resultados en todos los casos evaluados.

  El mismo grupo de trabajo realizaron otros similares, expandiendo esta investigación, los cuales se presentan a continuación.

- J. Sánchez, M. Galán, y E. Rubio. Applying a traffic lights evolutionary optimization technique to a real case: "Las Ramblas" area in Santa Cruz de Tenerife. *Evolutionary Computation, IEEE Transactions on*, 12(1):25–40, 2008

  Este estudio aplica lo presentado en el trabajo anterior a un escenario real en Santa Cruz de Tenerife. Algunas mejoras introducidas en el modelo del problema incluyen una nueva codificación del cromosoma utilizando código de Gray, que segun los autores permite mejorar el desempeño computacional de los operadores de mutación y cruzamiento. La población inicial se compone de nueve soluciones provistas por la alcaldía de la ciudad. Las estrategias de selección, cruzamiento y mutación son similares a las aplicadas en el trabajo anterior.

  El escenario discretizado de la zona estudiada lo componen 42 semáforos, 26 vías de entrada y 20 de salida. Las soluciones provistas por la alcaldía se ejecutaron en el simulador y los resultados obtenidos se compararon con los valores alcanzados por el algoritmo, concluyendo que logra una mejora de 26 % en el valor del fitness.

- J. J. Sánchez-Medina, M. J. Galán-Moreno, y E. Rubio-Royo. Traffic signal optimization in "La Almozara" district in Saragossa under congestion conditions, using genetic algorithms, traffic microsimulation, and cluster computing. *Intelligent Transportation Systems, IEEE Transactions on*, 11(1):132–141, 2010

  Este trabajo contiene puntos de contacto con el presentado anteriormente. Un cambio introducido fue el análisis de cuatro funciones de fitness diferentes, evaluando: *i)* la cantidad de vehículos que llegaron a destino, *ii)* el tiempo de viaje promedio, *iii)* el tiempo de ocupación promedio y *iv)* la velocidad promedio global.

  El trabajo incorpora nuevas métricas correspondientes al gas total emitido por los vehículos, que tienen relación con la velocidad a la que circulan. El modelo discretizado de la zona de *La Almozara* cuenta con 17 semáforos, 7 intersecciones, 16 entradas y 18 salidas. El análisis experimental incluye 10 escenarios que van desde baja congestión de trafico hasta alta. Se analizan los resultados para las distintas funciones de fitness concluyendo que las mayores mejoras en el valor de *fitness* ocurren cuando la congestión de tráfico es más alta.

- R. H. J. Penner y C. Jacob. Swarm-based traffic simulation with evolutionary traffic light adaptation. En *Proceedings of Artificial Intelligence and Soft Computing*, 2002

  Este trabajo se centra en un modelo de simulación basado en enjambres. Utiliza un algoritmo genético con el objetivo de sincronizar los semáforos y así minimizar el tiempo de espera de los vehículos en toda la red vial. El cromosoma contiene la secuencia y duración de los semáforos, así como la relación con los semáforos complementarios. La mutación tiene en cuenta esa relación para no generar inconsistencias, por ejemplo: que ocurran dos luces verdes en la misma intersección. Existe mayor probabilidad de cruzamiento entre semáforos que se encuentran en la misma intersección. La función de *fitness* calcula la relación entre el tiempo total de viaje y el tiempo de espera de todos los vehículos.

  El primer escenario considerado en el análisis experimental es pequeño; cuenta con una ruta de tres carriles y tres intersecciones. En este escenario el AG logra mejoras significativas. Luego se resuelve un segundo escenario más complejo, de 28 semáforos y 9 intersecciones, logrando mejoras de hasta 26 % en el tiempo de espera total.

- D. H. Stolfi. Optimización del tráfico rodado en ciudades inteligentes. Master's thesis, Universidad de Málaga, 2012

  Este trabajo propone el concepto de ciudad inteligente, enfocado en la movilidad, indicando que la congestión de tráfico provoca tanto pérdidas económicas como contaminación ambiental. Plantea el desarrollo de un algoritmo inteligente, que toma en cuenta el estado de congestión de las rutas y sugiere al usuario la ruta más rápida a su destino con el objetivo de minimizar los tiempos de viaje de los vehículos que circulan por la red vial. Para detectar el nivel de congestión, utiliza un dispositivo en el automóvil que se enlaza por *wifi* con los semáforos, los cuales cuentan con sensores. El trabajo no se basa en la sincronización de los semáforos, sino que presenta un sistema de búsqueda de la mejor ruta.

  El escenario experimental presentado incluye una zona de la ciudad de Málaga, que cuenta con ocho entradas y ocho salidas. Para la simulación del tráfico utiliza

el simulador SUMO. Los vehículos modelados son: turismo, monovolumen, furgoneta y camión. Cada vehículo del modelo posee características diferentes como: la longitud, velocidad y probabilidad que entre en la red de tráfico.

Implementa un algoritmo genético, donde los cromosomas representan los sensores, que incluyen la información de los destinos y rutas posibles. La estrategia de selección consiste en tomar los dos peores individuos y reemplazarlos por los dos mejores hijos encontrados. La función de *fitness* tiene en cuenta los valores obtenidos de la simulación del trafico, incluyendo: la cantidad de viajes completados, el tiempo medio de viaje y el retraso medio.

Para el análisis experimental, obtiene los resultados bases simulando 64 itinerarios diferentes y los compara con la aplicación del algoritmo inteligente. Las simulaciones se realizan hasta con 800 vehículos, concluyendo que al aumentar la cantidad de vehículos (más de 400) en el sistema, el algoritmo mejora sustancialmente los resultados bases.

- K. T. K. Teo, W. Y. Kow, y Y. Chin. Optimization of traffic flow within an urban traffic light intersection with genetic algorithm. En *Computational Intelligence, Modelling and Simulation (CIMSiM), 2010 Second International Conference on*, p. 172–177. IEEE, 2010

  Este trabajo presenta un escenario simple de una intersección donde un algoritmo genético intenta optimizar los tiempos de las luces de los semáforos para lograr mejorar el tráfico vehicular. El cromosoma representa los tiempos de la luces verdes, mientras que la función de *fitness* se calcula teniendo en cuenta el largo de las colas generadas por los vehículos en las intersecciones. La ejecución de la simulación utiliza un tiempo fijo de 600 segundos por generación, pero no se describe el tipo de simulador utilizado. El análisis experimental compara los resultados del AG cuando el escenario tiene en cuenta un flujo continuo de vehículos, y cuando no. Entre las conclusiones del trabajo se indica, que la optimización de las luces de los semáforos usando algoritmos genéticos es una buena opción para resolver el problema de la congestión del tráfico.

- D. J. Montana y S. Czerwinski. Evolving control laws for a network of traffic signals. En *Proceedings of the First Annual Conference on Genetic Programming*, p. 333–338. MIT Press, 1996

  Esta propuesta utiliza un enfoque adaptativo, con sensores que analizan el tráfico en tiempo real. Un sensor contabiliza los autos que circulan y otro detecta el largo de la cola de vehículos generada en la intersección. Se consideran los cambios producidos respecto al caso promedio y se modifican los tiempos de las luces de los semáforos de forma acorde, con el objetivo de mejorar la circulación del tránsito.

  Se aplica un algoritmo híbrido entre programación genética, específicamente STGP (strongly typed genetic programming) (Montana, 1995) y un algoritmo genético, para resolver el problema de sincronización se semáforos. La medida básica de efectividad en la función de evaluación es el *delay*, que representa el total de tiempo perdido causado por las señales de tráfico.

  El análisis experimental se desarrolló es un escenario simple con cuatro intersecciones y tres tipos de tráfico diferente, utilizando una versión especial del simulador

TRAF-NETSIM (Rathi, 1990). En los experimentos se comparan los valores obtenidos utilizando una configuración fija de los semáforos, cuyos valores fueron obtenidos al aplicar un AG, contra el algoritmo adaptativo desarrollado. Los resultados indican que se aprecian mejoras en el flujo de tránsito (hasta 40 % en el valor de *fitness*) y destaca la buena adaptabilidad del algoritmo implementado en diferentes circunstancias. Sin embargo, recalca que el escenario es simple y de tamaño reducido, siendo una incógnita como el algoritmo funcionará en problemas más complejos.

- A. Vogel, C. Goerick, y W. Von Seelen. Evolutionary algorithms for optimizing traffic signal operation. En *Proceedings of the European symposium on intelligent techniques (ESIT)*, p. 83–91. Citeseer, 2000

  Este trabajo utiliza un enfoque auto-adaptativo para mejorar el tráfico, tanto en el corto como en el largo plazo, a través de la optimización de las señales de tráfico en las intersecciones de una red de rutas.

  Presenta la idea de que una configuración de semáforos particular, aún siendo optimizada usando simulaciones, tiene poca probabilidad de ser la mejor en todas las situaciones o en casos extremos (horas picos). Para solucionar ese problema, propone un sistema auto-adaptable que toma la información del tráfico actual usando detectores de vehículos y espacios disponibles.

  Propone el desarrollo de un algoritmo evolutivo donde cada individuo representa un sistema de fases de los semáforos, mientras la función de *fitness* tiene en cuenta las demoras en el tráfico, que se obtiene utilizando simulaciones. El escenario es relativamente pequeño, con una intersección de dos calles, cada una con tres líneas, donde la ruta principal tiene el doble de densidad vehicular. Los resultados obtenidos de este trabajo indican que la ventaja de usar conocimiento experto para configurar los parámetros iniciales es mínimo, ya que el algoritmo llega rápidamente a resultados similares. No se realizaron comparaciones con otros escenarios, solo se estudió el comportamiento del algoritmo evolutivo utilizando diferentes parámetros..

- N. M. Rouphail, B. B. Park, y J. Sacks. Direct signal timing optimization: Strategy development and results. En *XI Pan American Conference in Traffic and Transportation Engineering*. Citeseer, 2000

  Este trabajo estudia una pequeña red de tráfico con nueve intersecciones semaforizadas en la ciudad de Chicago (USA); el escenario incluye: la red vial, el tráfico de vehículos, y las paradas de ómnibus. Se toman valores reales de tráfico en horas pico, comprobando que las colas de vehículos generadas en la simulación coinciden con la realidad. Utiliza el programa Wallace et al. (1984) que permite visualizar mapas y un simulador de tráfico comercial llamado Halati et al. (1997). El objetivo es resolver el problema de sincronización de semáforos, utilizando un AG cuya función de evaluación que tiene en cuenta las demoras en la red y el largo de las colas producidas por los vehículos. El análisis experimental compara los valores de la realidad, con los resultados obtenidos por el AG. Los resultados indican que el AG mejora los valores del caso base, reduciendo las demoras simuladas en un 44 %.

2.7.1. Resumen

Esta sección presenta un breve análisis sobre los trabajos evaluados y como se relacionan con el presente proyecto.

El trabajo de Sanchez et al. (2004) incluye algunos puntos de contacto con el presente proyecto, ambos utilizan un cluster como plataforma de ejecución del algoritmo evolutivo y aplican el mismo método de paralelismo (maestro-esclavo) en el modelo de paralelismo. La principal diferencia entre ambos trabajos, es que el escenario que evalúan es pequeño y no se realizan comparaciones con modelos que representen un escenario real. El siguiente trabajo de Sánchez et al. (2008) expande lo anterior y lo aplica a un escenario real, en Santa Cruz de Tenerife. Este escenario presenta una complejidad similar al Corredor Garzón en términos de cantidad de cruces y semáforos. Los resultados obtenidos son muy positivos, obteniendo mejoras de hasta 26 % en el valor de *fitness*.

Se destaca de Sánchez-Medina et al. (2010) las pruebas realizadas con diferentes funciones de *fitness*, teniendo en cuenta diversos factores como: tiempo de viaje o velocidad promedio. Este trabajo inspiró al presente proyecto en considerar la realización de una función multiobjetivo que tuviera en cuenta la velocidad promedio de los autos y ómnibus.

En el trabajo de Stolfi (2012) no se optimiza la configuración de los semáforos, pero plantea una posibilidad interesante para mejorar el tráfico de una ciudad indicando a los conductores la mejor ruta a seguir. Esta novedosa opción se podría tomar como un elemento en trabajos futuros. Tanto Teo et al. (2010) como Stolfi (2012) plantean la simulación con un tiempo fijo, característica que se consideró para el presente proyecto.

Montana y Czerwinski (1996) y Vogel et al. (2000) proponen algoritmos adaptables en tiempo real, siendo un aporte interesante a tener en cuenta para trabajos a futuro.

En conclusión, el estudio de los trabajos relacionados permitió conocer en profundidad distintas soluciones y métodos que se consideraron en menor o mayor medida en la solución propuesta. El hecho de que todos los trabajos consultados obtuvieron mejoras significativas en sus resultados motivó aún más el desarrollo del presente proyecto.

Capítulo 3

Estrategia de resolución

Este capítulo explica la estrategia de resolución seguida para resolver el problema de sincronización de semáforos en el Corredor Garzón. La metodología se basa en tres puntos: el modelado del problema, el algoritmo evolutivo desarrollado y el uso de técnicas de alto desempeño. El capítulo comienza presentando el modelado del problema, que incluye el diseño del mapa de la zona y la creación de instancias realistas que serán usadas para simular el tráfico en el simulador SUMO. En esta primera sección se presenta el simulador de tráfico SUMO y el trabajo de campo realizado para recabar datos precisos de la realidad. Luego el capítulo detalla el diseño e implementación del algoritmo evolutivo multiobjetivo, que fue desarrollado utilizando la biblioteca Malva. Finalmente se especifican los parámetros, características y modelo de paralelismo del Algoritmo Evolutivo implementado.

3.1. Modelado del problema

El modelado del problema comprende la simplificación de la realidad y el uso de aquellos elementos que resulten útiles para resolver el problema de sincronización de semáforos. El modelado se basa en el diseño de un mapa de la zona e instancias realistas que serán usadas en el simulador de tráfico SUMO. A continuación se presenta el simulador, las herramientas necesarias utilizadas y el proceso seguido para desarrollar el modelo.

3.1.1. Simulador SUMO (Simulation of Urban MObility)

Como se explicó en el marco teórico, SUMO es un simulador de tráfico gratuito y de código abierto que permite modelar redes de calles, vehículos, transporte público, ciudadanos y semáforos. Este simulador sigue un modelo microscópico, ya que realiza la simulación individual explícita de cada elemento. SUMO incluye un conjunto de herramientas destinadas a facilitar la generación de tráfico y la construcción de mapas.

Puesto que existe un amplio abanico de posibilidades a la hora de elegir un simulador de tráfico, se efectuó un análisis relacionado con las posibilidades y objetivos del proyecto. A continuación se presentan las razones que sustentan la elección de SUMO como el simulador de tráfico utilizado:

- Portable: SUMO puede ser ejecutado tanto en Windows como en Linux. Esta característica es requerida, dado que Windows es una de las plataformas utilizadas

para implementar el proyecto y Linux (CentOs) es el sistema operativo utilizada por el Cluster Fing donde se desarrolla el análisis experimental.

- Modos de ejecución: SUMO puede ejecutarse sin interfaz gráfica, utilizando solamente la línea de comando, lo que aumenta sensiblemente la velocidad de ejecución. Esta funcionalidad es fundamental a la hora de diseñar el algoritmo evolutivo, ya que la interacción con el simulador de tráfico debe ser eficiente. Por otro lado, presenta la opción de ejecutarse con interfaz gráfica, lo que es indispensable a la hora de visualizar la simulación. Esta opción es útil sobre todo en el inicio del desarrollo, cuando se realizan ajustes y pruebas.

- Gratuito y abierto: SUMO se presenta como un proyecto de código abierto y sin costo, siendo un factor importante para un proyecto de investigación como el que se quiere desarrollar.

- Documentación y mantenimiento: Este simulador incluye una detallada documentación que hace más fácil su utilización. Cuenta con una comunidad muy activa que responde dudas en foros lo cual es útil a la hora de buscar soporte en caso de problemas o errores. SUMO también cuenta con un desarrollo activo que permite su mejora por medio de actualizaciones frecuentemente.

- Configuración simple: SUMO presenta un sencillo sistema de configuración, basada en archivos XML para la ejecución de las simulaciones. La modificación de estos archivos permiten alterar la configuración de los semáforos, las propiedades y rutas de los vehículos, entre otras opciones. Complementariamente cuenta con herramientas para importar mapas de servicios como OSM (Haklay y Weber, 2008).

- Información de salida: SUMO ofrece la opción de generar una amplia variedad de datos, producida por la simulación ejecutada. Entre estos datos se encuentran la velocidad de los vehículos y el largo del recorrido, datos que son requeridos por la solución propuesta.

- Eficiente: Soporta redes de tránsito muy grandes y está diseñado para ejecutar simulaciones a gran velocidad, siendo una característica deseable por la complejidad del problema a solucionar.

SUMO presenta un funcionamiento sencillo basado en tomar como entradas archivos de configuración que representan la red vial, los vehículos, el tráfico y los semáforos. Además, genera archivos de salida con información útil como el tiempo de simulación, la cantidad de vehículos, velocidad de los vehículos, duración del viaje, emisiones de gases contaminantes, etc.

Dada la complejidad del problema a enfrentar, se utilizan otras herramientas para realizar ciertas tareas de manera más eficiente y con menos errores. *NetConvert* viene integrado con SUMO y es utilizado para generar la red vial a partir del mapa obtenido de Open Street Map, transformándolo al formato que SUMO reconoce. Otra herramienta integrada con SUMO es *DUaRouter* que permite generar recorridos de vehículos basado en dinámicas definidas por el usuario. *Traffic Modeler* creado por Papaleondiou y Dikaiakos (2009), es útil para la generación de tráfico de manera visual que puede ser exportado al formato reconocido por SUMO.

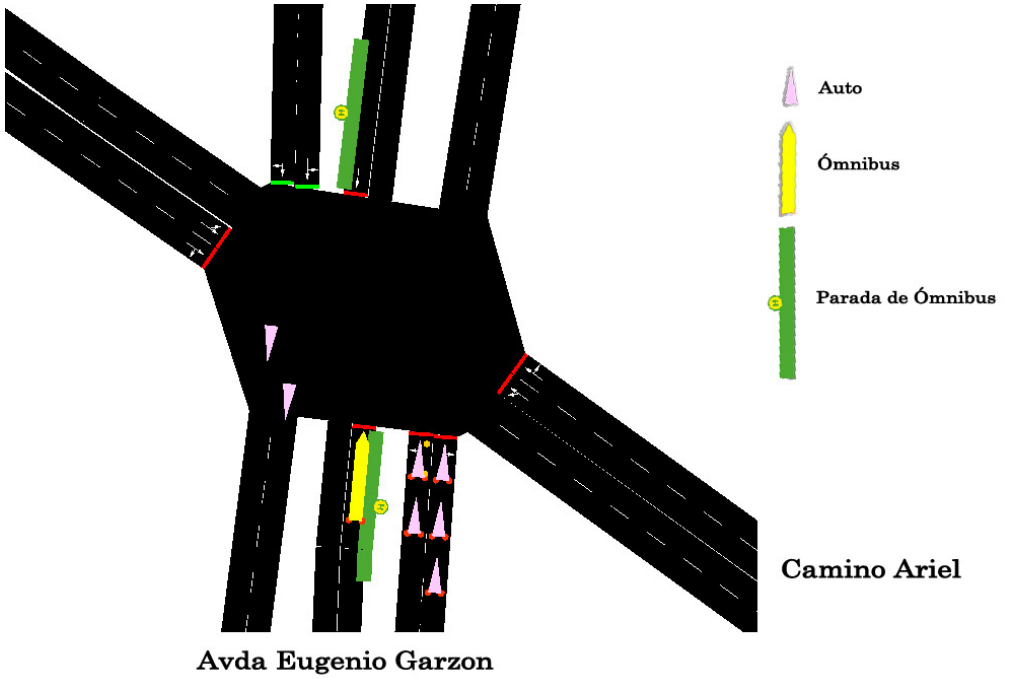

Figura 3.1: Simulación de tráfico en el simulador SUMO en el cruce entre Bulevar Battle y Ordoñez y el Corredor Garzón.

3.1.2. Proceso de modelado

El objetivo del proceso de modelado es el desarrollo de instancias realistas del problema. Se comienza con el diseño del mapa que representa la zona del Corredor Garzón. Luego se describe el trabajo de campo realizado para obtener datos precisos de la realidad. Finalmente utilizando estos datos, se describe el desarrollo de las instancias realistas que serán utilizadas en el simulador de tráfico.

Diseño del mapa

El primer paso del modelado consiste en diseñar un mapa de la zona del Corredor Garzón que sea compatible con el simulador. El servicio de Open Street Map (Haklay y Weber, 2008) cuenta con mapas libres y actualizados por una comunidad muy activa. Este servicio se utiliza para importar la zona de interés y luego editarlo con el programa JOSM(Java OpenStreetMap Editor) para poder adecuarlo a las necesidades del problema. Se utilizaron otros servicios (Google Maps y Bing Maps) y un estudio de la zona para validar la exactitud del mapa importado desde OSM. Algunas inconsistencias fueron detectadas entre la realidad y el mapa importado, las cuales fueron corregidas, por ejemplo en la dirección de las rutas y el diseño de algunos cruces.

El alcance geográfico comprende al Corredor Garzón en toda su extensión y dos caminos paralelos a cada lado del mismo. Como se ve en el mapa de la figura 3.2, no existen calles paralelas reales, por lo que el proceso para diseñar el mapa consistió en: seleccionar las calles que construyen dos caminos paralelos a cada lado del Corredor Garzón, luego, seleccionar las calles internas entre las paralelas. Se verifica que cada camino paralelo incluya calles doble mano o dos calles de una sola mano, finalmente se

borran las demás calles que no están incluidas en la zona modelada.

Dado los cambios realizados en el mapa, como la eliminación de calles que no formaban parte de la zona modelada, no se pudo contribuir con la comunidad OSM subiendo el mapa con las modificaciones que corregían algunas inconsistencias encontradas.

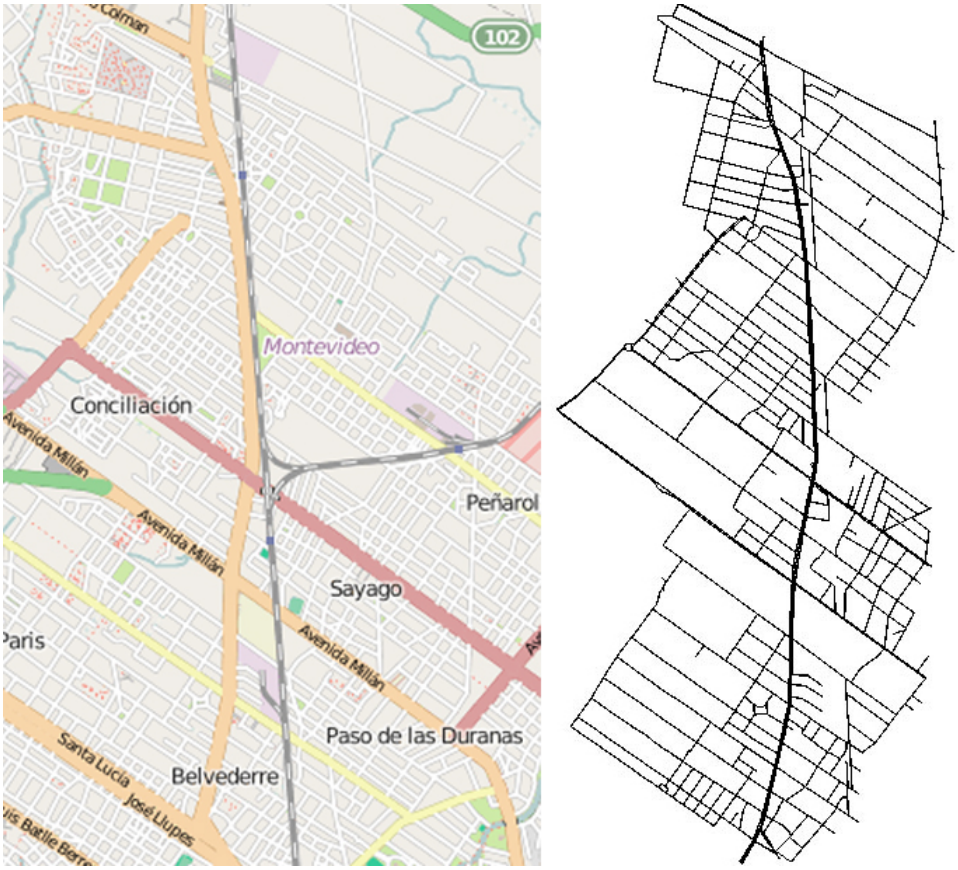

Figura 3.2: Mapa del corredor Garzón. A la izquierda el mapa extraído de OSM, a la derecha el mapa procesado listo para ser usado en SUMO. El corredor Garzón está en el centro de cada imagen.

La herramienta *Netconvert* fue utilizada para transformar el mapa obtenido desde OSM, al formato XML que reconoce el simulador SUMO. Esta herramienta no reconoce algunos datos incluidos en el formato OSM, por ejemplo: los números de las casas, vías de tren, vías de peatones, caminos, plazas, paradas y los separadores que tiene el corredor de las vías para autos, entre otros. Se destaca como punto importante que NetConvert no reconoce correctamente a los corredores, ya que confunde y superpone el corredor central con las vías para los autos, seguramente por no reconocer los separadores físicos que presenta el corredor. Para solucionar este problema se utilizó el programa JOSM para eliminar los datos no reconocidos por NetConvert y editar el mapa, separando aproximadamente un metro las calles del corredor central. Esta modificación permitió que NetConvert interpretara correctamente el funcionamiento del corredor, para que los vehículos no se trasladaran entre la vía de los autos y la del corredor de ómnibus. Pero este cambio implementado generó un problema, ya que cada cruce de la realidad pasó a ser representado por tres cruces distintos (uno por cada calle del corredor). Para solucionar

el problema anterior y otros errores, se realizaron ajustes manuales, modificando los archivos xml relacionados. Entre otros cambios, se especificó manualmente las reglas de movilidad correctas en algunos cruces, por ejemplo cuando estaba prohibido girar a la izquierda.

Al finalizar el proceso de diseño, se obtuvo un mapa fiel a la realidad, delimitado en la zona del Corredor Garzón que es compatible con el simulador de tráfico SUMO.

Trabajo de campo

Existen datos públicos disponibles sobre la cantidad de ómnibus y sus frecuencias, pero no se tienen datos sobre la densidad del tráfico vehicular en la zona del Corredor Garzón. Por este motivo se realizó un revelamiento in-situ en cinco cruces representativos. Los cruces seleccionados fueron: Camino Ariel, Battle y Ordoñez, Plaza Videla, Camino Colman y Aparicio Saravia. Estos cruces presentan diferentes características que son útiles para modelar variantes en la cantidad de vehículos circulantes.

Se siguieron las recomendaciones de los textos consultados relacionados con el conteo vehicular (Smith y McIntyre, 2002). El día seleccionado para efectuar el conteo fué el miércoles, con clima soleado y en el horario de 15:00 a 17:00 horas, para no obtener sesgos que se producen los fines de semana, en horas pico, o en días de lluvia. Se realizaron filmaciones de entre 15 y 30 minutos en los cruces y luego se analizaron los vídeos para efectuar el conteo manual con la posibilidad de enlentecer el vídeo para mayor facilidad. Luego se completó una planilla electrónica (figura 3.3) donde se incluyen: la cantidad de vehículos que recorren el Corredor Garzón, los que circulan por la calle que cruza y los que doblan.

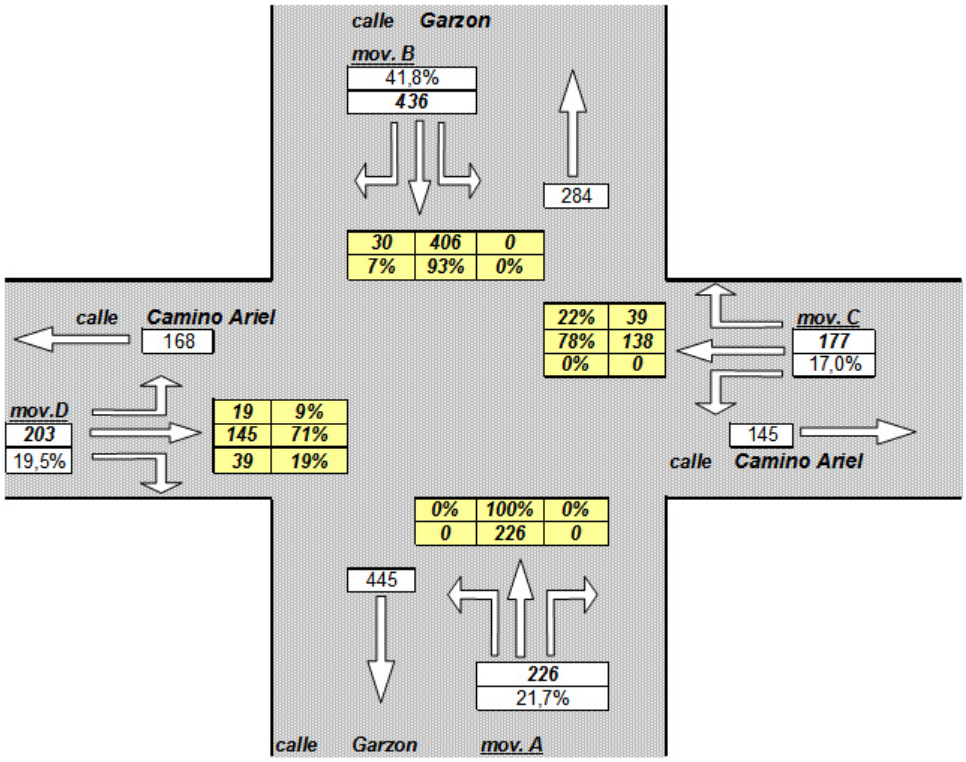

Figura 3.3: Ejemplo de planilla electrónica para el conteo manual de tráfico en la intersección entre el corredor Garzón y Camino Ariel

Cuadro 3.1: Resumen del revelamiento del tráfico en la zona del corredor Garzón. Se muestra la cantidad de vehículos y la orientación hacia donde circulan.

Intersección	Sur-Garzón	Norte-Garzón	Oeste	Este
Camino Colman	410	390	140	230
Plaza Vidiella	400	444	292	0
Aparicio Saravia	390	450	40	90
Battle y Ordoñez	510	390	470	300
Camino Ariel	436	226	177	203

La tabla 3.1 muestra la cantidad de vehículos en los distintos cruces; hay que tener en cuenta que solamente se contabilizan los autos, no se tienen en cuenta otro tipo de vehículos.

Se realizaron recorridas de punta a punta del corredor a una velocidad constante en auto para ser utilizados como datos a la hora de desarrollar las simulaciones de tráfico. Con este análisis se obtiene tanto la duración total del recorrido, como el tiempo que permanece detenido en los cruces semaforizados.

Para la configuración de los semáforos, se realizó un recorrido en bicicleta por toda la extensión del Corredor Garzón. Se cronometró la duración de las luces de los semáforos en cada esquina recorriendo el camino de ida, y para validar los tiempos también se relevaron las duraciones de las luces en el camino de vuelta. Estos tiempos fueron verificados por

los vídeos obtenidos anteriormente en el conteo del tráfico.

Desarrollo de instancias realistas del problema

Una vez completado el diseño del mapa compatible con el simulador de la zona del Corredor Garzón, y obtenidos los datos relevados de la realidad, se procede al desarrollo de instancias realistas que representen la situación actual en relación con el tráfico.

Existen varios modelos disponibles para representar la circulación de los vehículos. El modelo *aleatorio* genera recorridos para los vehiculos al azar. En el modelo *basado en áreas* se especifican las zonas donde el tráfico se origina y donde finaliza. Un modelo más complejo es el *basado en actividad* donde se definen la cantidad de casas, vehículos y población de una zona determinada. Luego el modelo genera la densidad de tráfico basado en los tipos de actividades que se realizan comúnmente, como ir al trabajo, hacer las compras, ir a la escuela, etc. Finalmente un modelo muy utilizado, sobre todo en escenarios de tamaño reducido, es el *definido por el usuario* que permiten determinar la ruta de cada vehículo con mayor detalle.

Se utilizó el programa Traffic Modeler (Papaleondiou y Dikaiakos, 2009) que se caracteriza por generar modelos de tráfico complejos de manera visual. Se optó por un modelo de movilidad entre áreas, lo que permite una buena granularidad al especificar la densidad de tráfico. El programa genera el recorrido del viaje para cada vehículo, que se mantienen constantes en las distintas ejecuciones del algoritmo evolutivo. La variación se produce en las velocidades desarrolladas por cada uno, ya que dependiendo de la configuración de los semáforos logrará mayores o menores velocidades.

Se plantea el uso de dos tipos de vehículos: autos y ómnibus. Cada tipo de vehículo posee diferentes características asociadas, como el tamaño, aceleración y velocidad máxima. No se especifican otros tipos de vehículos como motos o camiones ya que el trabajo se enfoca en los dos primeros tipos. Se agregaron las frecuencias y los distintos recorridos de los ómnibus obtenidos de datos públicos de la Intendencia Municipal de Montevideo(IMM). Las frecuencias incluyen las líneas urbanas: 'G', '409' ,'2', 'd5' y '148'. Las líneas de ómnibus suburbanas realizan un mismo trayecto las cuales no van a ser tomadas en cuenta en la optimización del algoritmo pero si aparecerán en la simulación.

La ubicación y líneas de cada parada se obtuvo del Servicio de Información Geográfica de Montevideo (http://sig.montevideo.gub.uy). Existen 14 paradas de líneas urbanas por el Corredor Garzón para el recorrido de ida y la misma cantidad para la vuelta. Se realizaron variantes en los recorridos dentro de una misma línea para simular el hecho de que no siempre se detienen en las mismas paradas. Se tuvo en cuenta el tiempo que demora un ómnibus al detenerse en una parada y que existen paradas donde por la cantidad de pasajeros se demora más tiempo. La duración de la detención en las paradas fueron obtenidas en el lugar, con tiempos de entre 15 a 35 segundos.

Se realizó un estudio basado en datos de GPS proporcionados por la IMM. Estos datos cuentan con la posición exacta, la velocidad instantánea y la referencia del ómnibus, para un conjunto de líneas seleccionadas tomadas en un período de una semana. Para este procesamiento se utilizó QGIS, un sistema de información geográfico capaz de visualizar, editar y realizar operaciones sobre elementos geográficos. Adicionalmente fue necesario el desarrollo de *scripts* para obtener las estadísticas necesarias, ya que la cantidad de información estudiada era muy grande (1.5Gb). El uso de QGIS permitió seleccionar, filtrar y relacionar los datos de posición y velocidad con las líneas que circulan por el Corredor Garzón. Luego de procesar los datos se constató una velocidad media de 14.5

km/h en el Corredor Garzón lo que permitió ajustar mejor el modelo.

Para configurar la simulación del tráfico se utilizan como entrada tres archivos de configuración en formato XML que son reconocidos por el simulador SUMO. El primero, es el archivo de *configuración de los semáforos* donde se detalla la duración de las luces, el comienzo de fase y la ubicación de los mismos en el mapa base. El segundo archivo representa la *ruta de los vehículos* que contiene el recorrido de cada vehículo. Finalmente el archivo con el *recorrido de los ómnibus* indica el recorrido de los mismos, su frecuencia, la ubicación de las paradas, en cuales se detienen y el tiempo que demoran en la parada.

Finalizando este proceso, se cuenta con todos los elementos necesarios para la creación de instancias realistas del problema, que serán utilizadas en el simulador para modelar la realidad. En el análisis experimental presentado en el próximo capítulo se describen los escenarios que hacen uso de estas instancias, modificando algunas variables como la densidad del tráfico para ajustarlo a las necesidades de la experimentación.

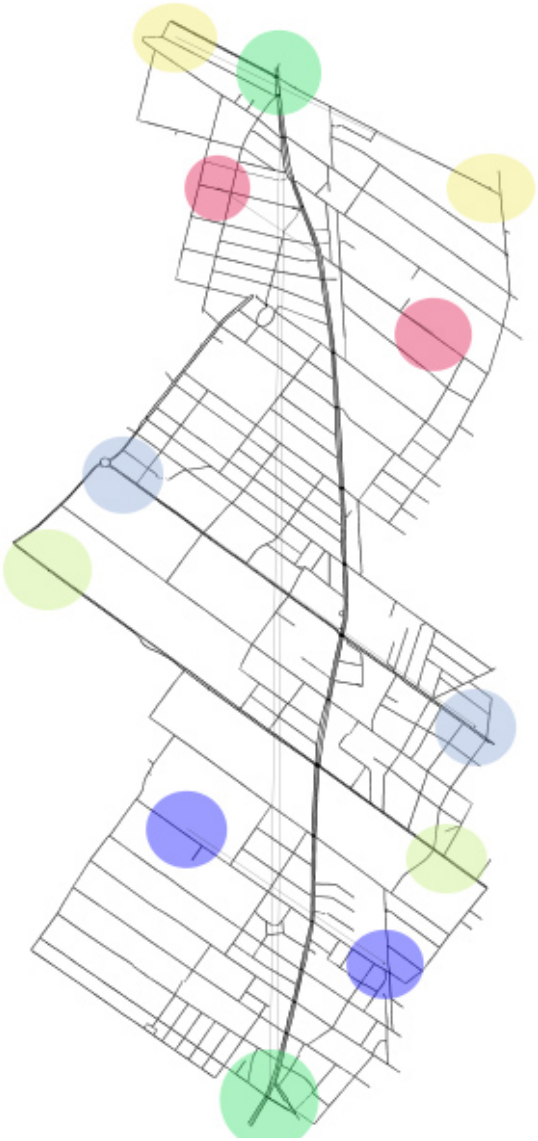

Figura 3.4: Mapa diseñado en el TrafficModeler para la generación de trafico. Los círculos del mismo color especifican el tráfico entre esas áreas.

3.2. Arquitectura de la solución

El esquema de la figura 3.5 muestra la arquitectura propuesta para el problema. La biblioteca Malva se utiliza para la implementación del algoritmo evolutivo. En cada evaluación de la función de fitness se realiza un llamado al simulador SUMO. El primer paso consiste en generar la población inicial del algoritmo evolutivo, estos individuos representan una configuración de semáforos para todo el Corredor Garzón, que incluye la duración de las fases y el *offset*. Luego por cada individuo se genera un archivo compatible con el simulador, que representa la configuración de semáforos. Este archivo se utiliza como entrada para ejecutar el simulador. Al finalizar la ejecución se generan archivos de salida que son procesados por el algoritmo evolutivo. Los datos de salida incluyen información necesaria para calcular el *fitness* de cada individuo como la velocidad media de ómnibus y otros vehículos. El algoritmo se detiene al alcanzar un número de generaciones determinado, en ese momento se crea un archivo con la mejor configuración de semáforos encontrada.

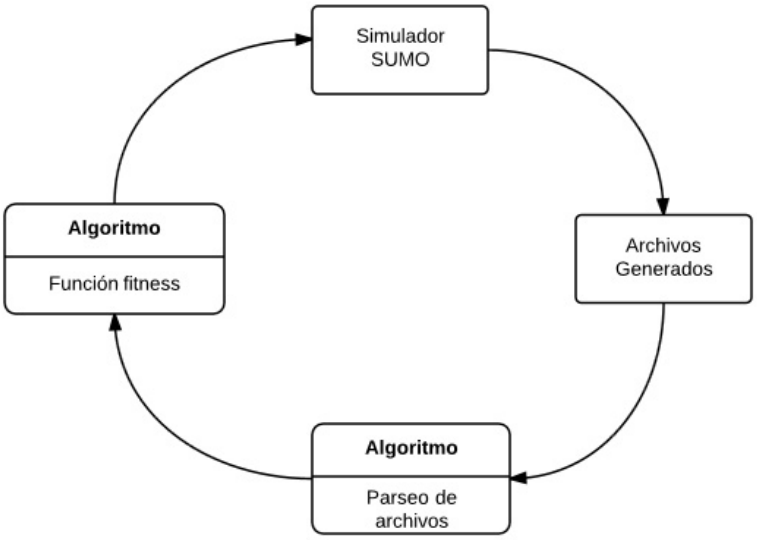

Figura 3.5: Arquitectura de la función de fitness

3.3. Implementación: Biblioteca Malva

Dada la complejidad del problema, se contempló desde el inicio del proyecto la utilización de un *framework* para el desarrollo del algoritmo evolutivo, de esta forma, se logra una ejecución eficiente y una implementación robusta pero flexible.

Como se detalló en el marco teórico, existen varias opciones disponibles para seleccionar un *framework*. Por las particularidades del problema planteado, existen ciertos requerimientos a la hora de seleccionar un *framework*, entre los que se destacan:

- Código abierto y gratuito: Al ser un proyecto de investigación de índole académico es conveniente que no existan costos económicos asociados. Además, es importante que se cuente con el código abierto con el fin de introducir modificaciones en el código base o corregir errores.

- Algoritmo genético: El *framework* debe facilitar el desarrollo de un algoritmo genético ya que es la base en la que se sustenta el proyecto.

- Algoritmo paralelo: Por la complejidad del problema y el alto consumo de recursos computacionales que insume, es fundamental que incluya la opción de ejecución en paralelo. Si no cuenta con funcionalidad nativa, al menos que exista la posibilidad de modificar el código base para agregarlo.

- Plataforma: Es requerido que se pueda ejecutar en el Cluster Fing (CentOs - Linux), en Windows y otros sistemas Linux, ya que son las plataformas en donde se desarrolla y donde se realiza el análisis experimental del trabajo.

- Confiabilidad: Es deseable que el *framework* sea lo suficientemente estable como para tener confianza de que el algoritmo funcionará de manera correcta. Por ejemplo que existan casos de éxito usando el *framework*, documentación, ejemplos de código, etc.

- Lenguaje: Es deseable que esté desarrollado en C++ o Java por la experiencia que se cuenta en estos lenguajes.

- Multiobjetivo: Es deseable (aunque no requerido) que soporte algoritmos multiobjetivo, ya que la función que se quiere implementar aún siendo multiobjetivo es sencilla.

Luego de analizar los puntos anteriores, se selecciona la biblioteca Malva para la implementación del algoritmo evolutivo. A continuación se detalla su funcionamiento y características.

Como se explicó anteriormente, Malva (Fagúndez, 2014) surge como una variante del proyecto Mallba (Alba et al., 2006). Propone su actualización, mejora y desarrollo como un proyecto de código abierto colaborativo. Su objetivo es proveer de varios esqueletos de heurísticas de optimización que puedan ser utilizados y extendidos de manera fácil y eficiente. Los esqueletos se basan en separar dos conceptos: El problema concreto que se quiere resolver y por otro lado el método utilizado para resolverlo. Por tanto un esqueleto se puede ver como una instancia de una plantilla genérica para resolver un problema particular, manteniendo todas las funcionalidades genéricas.

Malva utiliza el lenguaje C++, con el objetivo de brindar modularidad y flexibilidad. Los esqueletos se ofrecen como un conjunto de clases requeridas que el usuario deberá modificar para adaptarlo a su problema, y las provistas que incluyen todos los aspectos internos del esqueleto siendo independientes del problema particular. Entre los algoritmos provistos se encuentra el de Algoritmos genéticos y CHC (Eshelman, 1991).

3.4. Especificación del Algoritmo Genético utilizado

Se utiliza el algoritmo genético provisto por la biblioteca Malva llamado NewGA con las modificaciones detalladas en la sección anterior. El siguiente esquema describe el funcionamiento del algoritmo utilizado:

Algoritmo 3 Algoritmo Genético de Malva.

```
1: t = 0
2: Inicializo( P(t))
3: Evaluar estructuras en ( P(t))
4: while No termine do
5:    t++
6:    Seleccionar C(t) de P(t-1)
7:    Recombinar estructuras en C(t) formando C'(t)
8:    Mutar estructuras en C'(t) formando C"(t)
9:    Evaluar estructuras en C"(t) generando un hilo de ejecucion por cada una
10:   Consolidar valores de la evaluacion
11:   Reemplazar P(t) de C"(t) y P(t-1)
12: end while
```

A continuación se realiza un resumen de las características del algoritmo implementado, que en la siguiente sección serán tratados detalladamente:

- Algoritmo paralelo: Utiliza el método maestro-esclavo donde en cada iteración el maestro genera un hilo por cada ejecución de la función *fitness* y luego espera a la terminación de todos los hilos para consolidar los datos.
- Función Multiobjetivo: Se intenta optimizar tanto la velocidad promedio de vehículos como la de ómnibus, teniendo cada uno un peso específico.
- Representación del cromosoma: Es un vector de números naturales que representan la duración de las fases de los semáforos y el *offset* para todos los semáforos del Corredor Garzón.
- Cruzamiento y mutación: Se implementa una variante del cruzamiento de un punto específico para este problema y se utilizan dos tipos de mutación para modificar la duración de fase y el *offset* de los semáforos.
- Selección y reemplazo: Reemplaza padres por hijos. La selección de los padres se realiza por el método de torneo de tres individuos y la selección de hijos por el método de ruleta.

3.4.1. Representación del cromosoma

Como se explicó en el marco teórico, el problema de sincronización de semáforos puede ser resuelto optimizando diferentes parámetros. Entre estos parámetros se encuentran la duración de fase, de ciclo y el *offset*. Dependiendo de cuáles se seleccionen deberán ser incluidos en la representación del cromosoma.

Para la solución propuesta se contempla tanto la duración de fase como el *offset*. El cromosoma se agrupa lógicamente en cruces, siendo el valor de cada gen la duración de una fase en un cruce, además se agrega para cada cruce el valor del *offset*. Se utiliza un vector de números naturales para favorecer la claridad en el desarrollo y la facilidad a la hora de aplicar los operadores. Por este motivo el tamaño del cromosoma depende de la cantidad de cruces y de la cantidad de fases que incluya cada cruce. Con esta representación se busca una optimización global de todo el sistema y no individualmente

para cada cruce, lo cual es fundamental para mejorar la velocidad promedio del Corredor Garzón y del resto de las calles que lo cruzan.

En la representación del cromosoma se omiten las luces amarillas ya que no modifican los tiempos reales del paso de vehículos.

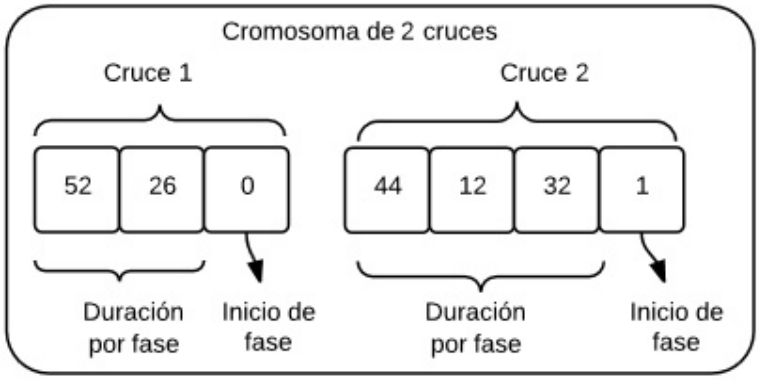

Figura 3.6: Cromosoma de dos cruces

Es importante que el algoritmo evolutivo genere solamente soluciones factibles, por lo tanto se tiene especial cuidado al ejecutar los operadores de cruzamiento y mutación.

```
<add>
    <tlLogic id="cluster_1894127318_917520037_917579892" type="static" offset="0">
        <phase duration="52" state="GGGGrrrGGGGrrr"/>
        <phase duration="3" state="yyyyrrryyyyrrr"/>
        <phase duration="26" state="rrrrGGgrrrrGGG"/>
        <phase duration="6" state="rrrryyyrrrryyy"/>
    </tlLogic>
    <tlLogic id="cluster_1858033111_917381626_917457905" type="static" offset="1">
        <phase duration="44" state="rrrrrrrGGGGrrr" />
        <phase duration="3" state="rrrrrrryyyyrrr" />
        <phase duration="12" state="rrrrGrrrrrrGrr" />
        <phase duration="3" state="rrrryrrrrrryrr" />
        <phase duration="32" state="GGGGrrrrrrrrrr" />
        <phase duration="3" state="yyyyrrrrrrrrrr" />
    </tlLogic>
</add>
```

Figura 3.7: Representación de Sumo

La figura 3.7 muestra el archivo de configuración de semáforos que utiliza el simulador SUMO para el cromosoma anterior, donde se representan las fases. Por ejemplo la fase *GGGGrrrGGGGrrr* indica la secuencia de luces y su duración de 52 segundos, *G* indica la luz verde, *r* la roja, *y* el Amarillo. El valor del *offset* indica el inicio de la fase.

La figura 3.8 presenta la numeración de los semáforos que se corresponden con la primera fase de este ejemplo, donde cada posición de la secuencia *estado* se corresponde con un color en particular.

Por tanto para el estado *GGGGrrrGGGGrrr* se tiene que:

- GGGG se corresponde a 1, 2, 3 y 4.
- rrr a 5, 6 y 7.
- GGGG a 8, 9, 10 y 11.
- rrr a 12, 13 y 14.

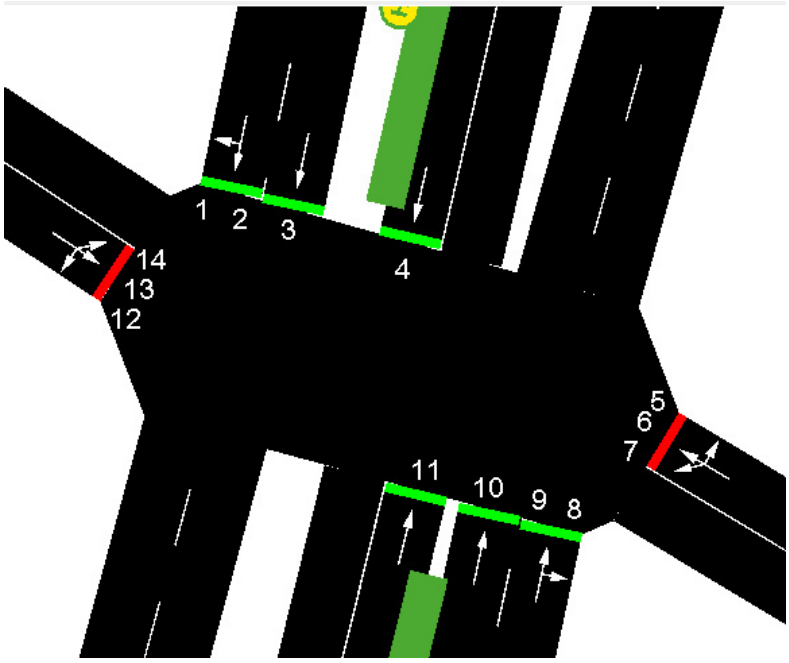

Figura 3.8: Representación de una fase de los semáforos para un cruce. Cada número se corresponde a una letra en la secuencia de *state* del archivo de configuración del simulador SUMO.

3.4.2. Población inicial

Para la inicialización de la población se toma como referencia la configuración obtenida a partir de los datos de la realidad, luego para cada cruce se varían las duraciones de las fases de manera aleatoria entre un rango de valores configurable. Además se selecciona la fase inicial aleatoriamente entre la cantidad de fases del cruce.

3.4.3. Función *fitness*

La evaluación de un individuo se realiza generando un archivo con la configuración de los semáforos en base a su cromosoma y ejecutando el simulador SUMO utilizando esta configuración para obtener los datos necesarios para calcular el *fitness*.

Se emplea una función multiobjetivo usando combinación lineal de la velocidad de los ómnibus y del resto de los vehículos, ya que es un método sencillo y adecuado cuando

son menos de tres objetivos. El *fitness* se calcula como una suma ponderada, con los pesos fijados a priori.

$$F(x) = \sum_{i=1}^{n} w_i f_i(x) \tag{3.1}$$

Se selecciona como objetivo la velocidad promedio de los ómnibus (vpb) y la velocidad promedio del resto de los vehículos (vpv). Esta métrica fue elegida pues es más adecuada para realizar las comparaciones con la realidad. Por ejemplo la cantidad de vehículos que completan su viaje, la duración promedio del recorrido o el tiempo de simulación son todas métricas que podrían usarse pero no son útiles a la hora de realizar comparaciones con la realidad.

La siguiente es la fórmula de *fitness* utilizada, donde x e y indican los pesos que se especifican en la función. En una primera instancia se establece x = y = 1, más adelante se experimentará con otros pesos.

$$f = x.vpb + y.vpv \tag{3.2}$$

Se plantea optimizar la velocidad promedio de los vehículos en toda la red vial, de manera global, esto quiere decir que se busca una mejor velocidad promedio tanto en autos que van por Garzón como aquellos que circulan por los cruces o calles paralelas. La optimización global es fundamental, ya que si cada cruce se optimiza individualmente podría generar problemas de congestión en otras zonas.

3.4.4. Operadores

Operador de Cruzamiento

Se utiliza cruzamiento de un punto, seleccionando del cromosoma una posición aleatoria entre dos cruces como punto de corte, por tanto si un tramo del corredor tiene un buen comportamiento se intenta mantener esa propiedad. En el escenario que plantea la siguiente figura, los padres cuentan con tres cruces. Se elige un punto de corte aleatoriamente entre dos cruces y se generan los hijos que son una combinación de los padres.

Figura 3.9: Visualización del cruzamiento entre individuos.

Operador de Mutación

Se implementaron dos tipos de mutación:

- Mutación de duración de fase: para cada fase de cada cruce se modifica su duración sumando o restando una cantidad dada de segundos, con una probabilidad dada. Teniendo en cuenta que el valor obtenido se encuentre dentro de un rango especificado para que no se produzcan casos irreales como un cruce de menos de 5 segundos.
- Mutación del *offset*: Se elige aleatoriamente una fase con la cual va a comenzar el cruce con una probabilidad dada.

Selección y reemplazo

Se utilizan los operadores y criterios provistos por el algoritmo genético newGA de Malva. La selección de padres se realiza por el método de torneo de tres individuos y la selección de hijos por el método de ruleta. La política de reemplazo indica que tanto padres como hijos pueden ser parte de la siguiente generación (no solamente los hijos), por lo que existe reemplazo de padres por hijos.

Parámetros del algoritmo

Los parámetros específicos del algoritmo se establecen en el siguiente capítulo, donde se realiza un análisis experimental para encontrar los más adecuados. Estos parámetros son: número de generaciones, tamaño de población, probabilidad de cruzamiento y de mutación.

3.5. Modelo de paralelismo e implementación

Uno de los requisitos planteados al inicio del presente trabajo era la creación de un algoritmo genético que soportara paralelismo, con el objetivo de reducir los tiempos de

ejecución. El motivo principal fue que los escenarios planteados se consideraron complejos y que utilizarían gran cantidad de recursos computacionales. Para este propósito, se utilizó como código base el algoritmo genético provisto por la biblioteca Malva llamado NewGA. Este código fue modificado para que el algoritmo soportara la ejecución en paralelo, específicamente se desarrolló una nueva forma de evaluar a los individuos, generando un hilo de ejecución por cada individuo.

Se utilizó el método maestro-esclavo para el modelo de paralelismo. El proceso maestro se encarga de la mayoría de las etapas del algoritmo evolutivo, al comenzar, inicializa la población y se encarga de distribuir la evaluación de los individuos hacia los esclavos, creando un hilo de ejecución por cada esclavo. Luego el proceso maestro espera a que las evaluaciones terminen para obtener los valores de *fitness* de los esclavos. Una vez obtenidos los valores, selecciona a los mejores individuos y le aplica los operadores de cruzamiento y mutación. Para finalizar ejecuta el criterio de reemplazo para generar la siguiente población. Mientras, los esclavos se encargan solamente de obtener el individuo enviado por el maestro y de efectuar la evaluación del individuo, que corresponde a ejecutar el simulador y obtener los datos de salida.

En el capítulo siguiente, en la sección de análisis experimental se realiza un estudio de la eficiencia computacional del algoritmo genético paralelo, para comparar los tiempos de ejecución entre la versión paralela y en serie.

Capítulo 4

Análisis Experimental

Este capítulo presenta la evaluación experimental del algoritmo evolutivo multiobjetivo propuesto para resolver el problema de sincronización de semáforos en el Corredor Garzón. Inicialmente se presenta una descripción de los escenarios utilizados en la evaluación experimental y la plataforma de ejecución en la que se llevo a acabo la experimentación. A continuación se detalla el análisis experimental, que se divide en dos etapas: primero se analiza la configuración paramétrica inicial del algoritmo evolutivo con el objetivo de obtener una mejor calidad de resultados y luego se realizan los experimentos de validación para los que se reportan los resultados numéricos. Finalmente se ofrece un breve análisis de la eficiencia computacional del algoritmo evolutivo propuesto.

4.1. Plataforma de ejecución y Desarrollo

Como se explicó en el capítulo anterior, la biblioteca Malva se utiliza para implementar el algoritmo evolutivo propuesto, específicamente se trabaja con el algoritmo genético newGA provisto por la biblioteca. El código base de este algoritmo fue extendida para soportar la creación de nuevos hilos de ejecución para lograr el funcionamiento en paralelo.

Dada la complejidad del problema, el análisis experimental se realiza en la plataforma Cluster Fing (sitio web: http://www.fing.edu.uy/cluster) con el objetivo de acelerar el tiempo real de procesamiento al ejecutar el algoritmo evolutivo en paralelo. De esta forma se distribuye la carga de procesamiento en varios núcleos. El cluster Fing cuenta con procesadores AMD Opteron 6272 de 2.09GHz, 48Gb RAM, 64 núcleos, con sistema operativo CentOS Linux 6.5.

Las pruebas incluidas en el análisis experimental utilizaron entre 4 y 32 núcleos de procesamiento. Dada la naturaleza de recursos compartidos del Cluster, no siempre es posible contar con una cantidad igual de recursos disponibles, además para la mayoría de las pruebas el número de núcleos no es relevante. El numero de núcleos utilizados sera tenido en cuenta cuando se realice el análisis de eficiencia computacional detallado al final de este capítulo.

4.2. Ajuste paramétrico

Se realiza un análisis previo para determinar el mejor valor de los parámetros del algoritmo evolutivo propuesto. Este análisis tiene como objetivo obtener una mejor calidad

de resultados. Los parámetros que se ajustan incluyen el *tiempo de simulación* que se refiere al parámetro del simulador SUMO para determinar la duración de la simulación. Luego el *criterio de parada* que indica el método usado para detener el algoritmo y el *tamaño de la población* que representa la cantidad de individuos de la población utilizada en el algoritmo evolutivo. Finalmente se ajustan los parámetros de *probabilidad de mutación y probabilidad de cruzamiento*. Estos parámetros serán descritos en detalle en las siguientes secciones.

Para comenzar el análisis del ajuste paramétrico se generan tres escenarios de tráfico diferentes. Estos escenarios no incluyen datos recabados de la realidad (como el tráfico vehicular o la frecuencia de los ómnibus), para no ajustar los parámetros a un caso particular. A continuación se presenta la cantidad de vehículos para cada uno de los tres escenarios de tráfico propuestos:

- Tráfico Bajo: 30 ómnibus y 500 vehículos
- Tráfico Medio: 60 ómnibus y 1000 vehículos
- Tráfico Alto: 120 ómnibus y 2000 vehículos

En el ajuste paramétrico los primeros parámetros que se definen son: el tiempo de simulación, el criterio de parada y el tamaño de población. Luego de establecidos los valores para los parámetros mencionados, se realizan las pruebas para todas las combinaciones de probabilidad de cruzamiento y mutación buscando los valores con los cuales el algoritmo evolutivo obtiene una mejor calidad de resultados.

La plataforma Cluster Fing donde se desarrolla el análisis experimental permite disponer tanto de la métrica del tiempo real que dura la ejecución, así como también el tiempo secuencial, es decir la suma del tiempo de procesamiento de todos los procesadores involucrados en la evaluación del algoritmo. Cuando se realizan comparaciones en los tiempos de ejecución se utiliza el tiempo secuencial que es independiente de la cantidad de procesadores utilizados en las pruebas.

4.2.1. Pesos de la función de fitness

Para el análisis experimental la función *fitness* del algoritmo (3.2) tendrá los pesos $x = y = 1$. Estos valores dan pesos equitativos tanto a ómnibus como a otros vehículos por lo que no existe prioridad para uno u otro. Más adelante se realizarán experimentos con otras variantes.

4.2.2. Tiempo de simulación

El tiempo de simulación refiere a la duración de una simulación usando SUMO. Es un parámetro que se puede configurar y permite tener un mejor control sobre los tiempos totales de ejecución del algoritmo.

Se establece un tiempo de simulación de 4000 *steps* (medida interna de tiempo del simulador). Este número representa 66 minutos en la simulación, mientras que el tiempo real de ejecución depende de la plataforma computacional utilizada y de la cantidad de vehículos considerados en el escenario. En las simulaciones realizadas en el marco de este proyecto se encuentra entre los 5 y 20 segundos. En el análisis experimental se tuvo en cuenta y validó que en cada escenario más del 80 % de los vehículos completaran la

simulación, es decir que llegaran a sus destinos. Se realizaron 10 ejecuciones del algoritmo para cada uno de los tres tipos de tráfico comprobando que se cumplía el criterio.

4.2.3. Criterio de parada

Se elige como criterio de parada un determinado número de generaciones. Este criterio permite estandarizar las pruebas y evaluar una comparación justa entre los distintos escenarios de tráfico. Para determinarlo se busca un compromiso entre un buen resultado y un tiempo de ejecución que no sea excesivo.

Se decide que por un lado la ejecución del algoritmo deberá estar comprendida entre 1 y 24 horas, y además comprobar experimentalmente que el valor de *fitness* no tiene una gran variación en las ultimas 100 generaciones. Se ejecutaron 10 ejecuciones del algoritmo por cada tipo de tráfico.

Luego de la realización de las pruebas se elige el número de 500 generaciones como criterio de parada. En la siguiente gráfica se aprecia como el valor de *fitness* no presenta grandes variaciones luego de la generación 400, además el tiempo de ejecución real está dentro del margen pautado. La gráfica no muestra todos los valores, solo algunos representativos para una mejor visualización.

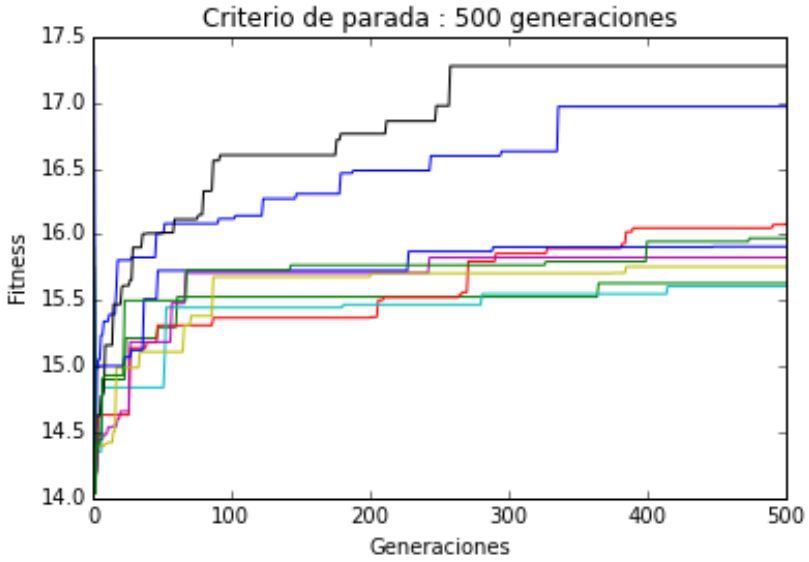

Figura 4.1: Gráfica de la evolución de los valores de fitness para establecer el criterio de parada.

4.2.4. Tamaño de la población

Para la elección del tamaño de la población se tendrán en cuenta tres elementos: el valor de *fitness* resultante de la ejecución del algoritmo genético, el tiempo de ejecución total y los recursos de la plataforma de ejecución.

La máxima cantidad poblacional a estudiar está determinada por la infraestructura del Cluster Fing, donde la cantidad máxima de núcleos en el mismo nodo es 64. Teniendo en cuenta que la mejor distribución de trabajo es un individuo de la población por núcleo, se determina que la máxima cantidad de población será de 64 individuos.

Luego se eligen dos valores mas para completar el análisis, estos son 32 y 48 individuos de tamaño de población. Se tiene en cuenta que no son lo suficientemente bajos y son valores con los que se obtiene una distribución adecuada de individuos en la infraestructura. Para las pruebas se realizan 10 ejecuciones del algoritmo por cada tipo de tráfico obteniendo el promedio de esos valores.

La tabla 4.1 muestra los resultados obtenidos; como se aprecia no existen grandes diferencias en la elección de un número poblacional sobre otro. Por tanto se elige 32 como tamaño de la población, teniendo en cuenta que el tiempo de ejecución secuencial del algoritmo es el menor y que insume menos recursos al ejecutarse. Esto es importante al ejecutarse sobre el Cluster Fing que es utilizada por otras personas y con recursos limitados.

Cuadro 4.1: Comparación de los valores de *fitness* para distintas poblaciones.

Población	**Fitness**		**Tiempo ejecución serial (m)**
	mejor	promedio	
32	17.28	16.37±0.5	10184±526
48	16.19	15.84±0.3	6772±256
64	17.27	16.46±0.6	4853±155

4.2.5. Probabilidad de mutación y cruzamiento

Para configurar la probabilidad de cruzamiento (pc) se consideraron tres valores candidatos (0.5, 0.8, y 1) y para la probabilidad de mutación (pm) otros tres (0.01, 0.05, y 0.1). De las nueve combinaciones posibles, se realizaron tres ejecuciones independientes del algoritmo para cada uno de los tres tipos de tráfico (bajo, medio y alto).

Cuadro 4.2: Valores del fitness para las distintas combinaciones de probabilidad de cruzamiento(pc) y de mutación (pm)

p_C	p_M	Fitness promedio ± desviación estándar
0.5	0.01	16.09±0.30
0.5	0.05	15.60±0.17
0.5	0.1	16.16±0.42
0.8	0.01	16.04±0.55
0.8	0.05	15.85±0.32
0.8	0.1	16.08±0.34
1	0.01	16.08±0.45
1	0.05	15.82±0.34
1	0.1	16.04±0.25

Analizando la tabla y la gráfica se puede apreciar claramente que para una probabilidad de mutación de 0.05 se obtienen los peores resultados. Otro dato interesante es que no existe gran diferencia en el resto de las combinaciones.

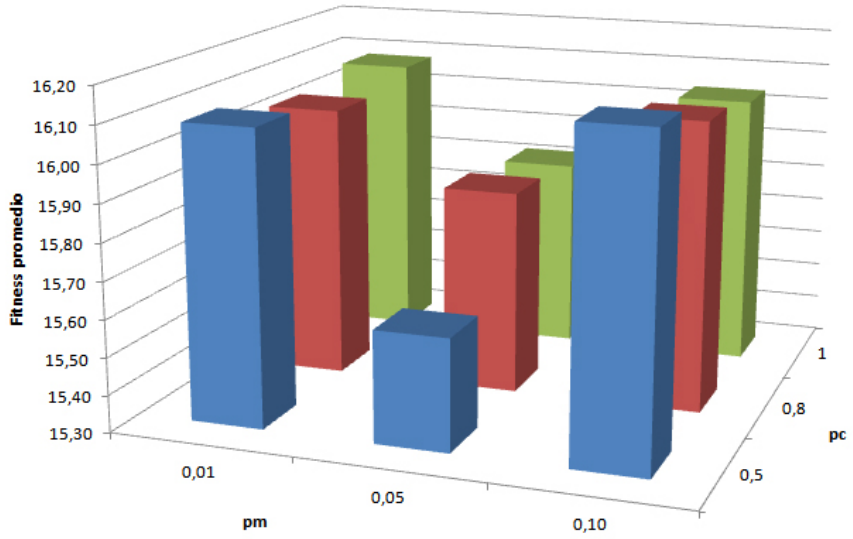

Figura 4.2: Gráfica con combinaciones de probabilidad de cruzamiento (pc) y de mutación (pm).

Se comprueba que todas las muestras siguen la distribución normal para poder aplicar el test de Student.

Si se compara la combinación del mejor promedio (0.5-0.1) y del peor (0.5-0.05) con el test de Student obtenemos t(x) = 0.07 que nos indica que para un nivel de significancia de 0,1 la hipótesis nula es rechazada, por lo tanto existe evidencia estadística para elegir la combinación con el mejor promedio (0.5-0.1) sobre la combinación con el peor promedio (0.5-0.05) para la ejecución del algoritmo. Para comprobar si es la mejor opción se toman las dos combinaciones con el mejor promedio (0.5-0.1) y (0.5-0.01) obteniendo en el test Student t(x) = 0.71 que nos indica que no existe una diferencia significativa entre ambas muestras por lo que elegir una sobre otra no implicaría grandes beneficios. En tal sentido podríamos elegir cualquiera de las dos, en este caso se elige la combinación (0.5-0.01) por su buen promedio y baja desviación estándar.

4.3. Descripción de escenarios

Esta sección presenta los escenarios que serán evaluados, el primero es el escenario base que representa la realidad actual y el segundo un escenario alternativo que contiene modificaciones con el objetivo de obtener mejores métricas que el escenario base.

4.3.1. Caso base o realidad actual del corredor

El caso base representa la situación actual en términos de tráfico, red vial y sincronización de semáforos del corredor Garzón. El objetivo es realizar una simulación para recabar mas información (velocidad promedio de vehículos y ómnibus) que sirva para comparar con los resultados del algoritmo utilizando los datos obtenidos in-situ (configuración de semáforos, cantidad de vehículos, etc). Se valida su correctitud comparando los tiempos obtenidos en la simulación con tiempos obtenidos in-situ de los recorridos de

ida y vuelta para los vehículos. Para el caso de los ómnibus se utilizan las frecuencias, cantidad y recorridos de los mismos que son de acceso público.

Se realizó un estudio sobre datos proporcionados por la IMM que contenían el posicionamiento de los ómnibus, velocidad instantánea y datos de la línea durante todo el día para una semana en particular. De esta forma se constató que para las líneas de ómnibus que pasan por Garzón la velocidad promedio de los ómnibus es de 14.5 km/h. Esto permitió calibrar el escenario modificando aspectos de la simulación relacionados con los ómnibus para mayor precisión.

Sobre el escenario base geográfico se realizarán tres escenarios de tráfico : bajo, medio y alto. El caso medio representa los datos obtenidos en el trabajo de campo, el bajo es disminuyendo el 50 % de vehículos y el tiempo de espera en las paradas de ómnibus teniendo en cuenta que en este caso existirá menos personas utilizando el transporte público. Las frecuencias de ómnibus se mantienen iguales ya que no son alteradas en la realidad. El caso de tráfico alto se aumenta 50 % los vehículos y el tiempo de espera en la parada de los ómnibus. El aumento y disminución del 50 % se obtuvo al analizar datos proporcionados por la IMM de la zona de Garzón de años anteriores.

A continuación el resumen de la cantidad vehicular para los tres escenarios de tráfico:

- Tráfico Alto: 3000 vehículos en la simulación y 70 ómnibus.
- Tráfico Medio: 2000 vehículos y 70 ómnibus.
- Tráfico Bajo: 1000 vehículos y 70 ómnibus.

4.3.2. Escenario alternativo

Para mostrar la utilidad que tienen las simulaciones sobre un escenario real, se realiza un escenario alternativo. Una de las ventajas principales es que no requiere gran inversión monetaria, de tiempo y que no afecta la situación actual de la realidad, por lo que se pueden generar distintas pruebas para encontrar aquellas que logren un beneficio.

Analizando los puntos que se entienden podrían atentar contra el buen funcionamiento del Corredor, se agregan algunas modificaciones al escenario base para intentar mejorarlo. El objetivo no es demostrar que será la mejor alternativa, sino dar una de las muchas alternativas que se pueden generar y probar con la simulación si se logran mejoras. Ya que pueden existir limitaciones o reglas que no estamos tomando en cuenta y que deben cumplirse en la realidad.

Entre los cambios estudiados se encuentran: eliminación de paradas y pasajes peatonales, alternar paradas y modificación de reglas de semáforos.

4.4. Resultados

Esta sección muestra los resultados obtenidos tanto de la simulación de la realidad, como de la aplicación del algoritmo. Se presenta la simulación del escenario alternativo y la posterior evaluación. Además se realizan estudios sobre cambios en la función de fitness del algoritmo y un breve análisis de la eficiencia computacional.

4.4.1. Valores numéricos del caso base

En la tabla 4.3 se pueden ver las métricas obtenidas para las diferentes instancias de tráfico simulado para el caso base. Estos datos representan la realidad actual del Corredor Garzón. Como se aprecia la velocidad promedio de los ómnibus en tráfico medio es de 14.6km/h siendo 14.5km/h el valor que se obtuvo de analizar los datos reales proporcionados por la IMM, lo que verifica que el modelo se aproxima a la realidad.

Cuadro 4.3: Resultados numéricos del caso base, mostrando la velocidad promedio ómnibus (vpb) y velocidad promedio vehículos(vpv) para los distintos tipos de tráfico

	$vbp(km/h)$	$vvp(km/h)$	Fitness
Tráfico Bajo	15.89	32.45	13.42
Tráfico Medio	14.59	28.81	12.05
Tráfico Alto	14.31	26.36	11.30

En el caso base no se representa la desviación standard ya que los resultados de su simulación son constantes, pues siempre se están probando los mismos recorridos de los vehículos y la configuración de los semáforos no cambia. Cuando se aplica el algoritmo obtenemos un valor diferente de velocidad y *fitness* en cada ejecución ya que tienen distintas configuraciones de semáforos, en este caso si se muestra la desviación standard.

4.4.2. Resultados numéricos de la evaluación

Como se aprecia en la tabla 4.3, el algoritmo mejora la velocidad promedio tanto de ómnibus como de otros vehículos en los tres tipos de tráfico estudiados. Además la velocidad media de los vehículos se mantiene en un rango mucho más ajustado que en el caso original al variar el tráfico. Las mejoras logradas en el *fitness* son de de hasta 24 %. A continuación se describe el análisis estadístico para comprobar la mejora.

Cuadro 4.4: Resultados numéricos del algoritmo evolutivo, mostrando velocidad promedio ómnibus (vpb) y de otros vehículos(vpv) para los distintos tipos de tráfico.

Tráfico	$vbp(km/h)$	$vpv(km/h)$	*Fitness*		Mejora *fitness* (%)	
			Promedio	Mejor	Promedio	Mejor
Bajo	17.92±0.18	34.30±0.40	14.50±0.14	14.88	8.04	10.8
Medio	16.95±0.32	33.29±0.29	13.95±0.15	14.19	15.70	17.7
Alto	16.51±0.61	32.90±0.25	13.72±0.17	14.04	21.40	24.2

Se realizaron 20 ejecuciones independientes para cada tipo de tráfico, comprobando que siguieran una distribución normal.Por tanto se puede aplicar el criterio de significancia estadística para validar los resultados. Este indica que el algoritmo A es mejor que B si los resultados de A y B cumplen:

$$|f_{avg}(A) - f_{avg}(B)| > max(std(f_A), std(f_B)) \tag{4.1}$$

En este caso, A representa el algoritmo y B el caso base. Esto indica que la diferencia del resultado promedio del algoritmo restado al resultado del caso base debe ser mayor a

la máxima desviación. Esto se cumple para todos los casos, por lo que se puede afirmar que existe evidencia estadística para indicar que los resultados del algoritmo son mejores al del caso base.

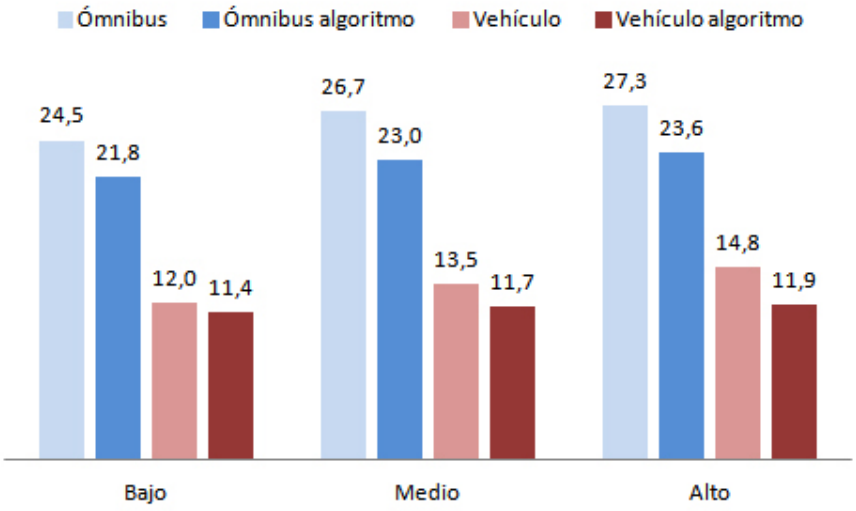

Figura 4.3: Comparación entre la duración en minutos de los viajes sobre el escenario base y al aplicar el algoritmo evolutivo al escenario alternativo, de ómnibus y otros vehículos en el recorrido completo del corredor Garzón para los diferentes tipos de tráfico.

Un resultado interesante es que la duración original de los viajes cuando el tráfico es bajo es casi igual a la duración de los viajes con tráfico alto luego de ejecutar el algoritmo. Para ómnibus tenemos 24.5m y 23.6m y para otros vehículos 12.0m y 11.9m.

4.4.3. Detalles del escenario alternativo

Los cambios propuestos incluyen eliminación de paradas, semáforos, pasajes peatonales y alternar paradas. Se estudiaron otras propuestas pero fueron descartadas por la poca viabilidad real de las mismas, como por ejemplo construir calles paralelas a Garzón o nuevas reglas en los cruces como existen en otros países.

Eliminación de paradas

Se consideraron dos paradas a eliminar que cumplieran con algunas características: no fueran cercana a una calle principal, que existiera otra parada cercana y que la eliminación de la parada no afecte en demasía a la gente en un traslado mayor. En este caso se seleccionaron las paradas en la calle Ariel y Casavalle.

Eliminación de pasajes peatonales

Hay tres pasajes peatonales en el corredor con semáforos que detienen el tráfico, dos de ellos solo manejan una esquina (sin pulsador en funcionamiento) donde en el escenario alternativo se implementó solamente mediante un *pare* en la calle transversal al corredor y el otro es netamente peatonal frente a la Facultad de Agronomía que fue totalmente eliminado. Una opción que mantiene los pasajes peatonales así como también los resultados obtenidos en el escenario alternativo sería implementar el pasaje peatonal

por encima del corredor. Al eliminar los pasajes peatonales se aumenta la velocidad media de todo el transporte.

Alternar paradas

Uno de los problemas del ómnibus es su baja aceleración, por lo que cada vez que este frena en un semáforo o en una parada demora en retomar una velocidad aceptable. Por tanto al reducir la cantidad de paradas que un ómnibus tiene que hacer se mejora la velocidad promedio. La línea G recorre a Garzón de punta a punta y es cubierta por las empresas Coectc y Cutcsa. Una posibilidad de alternancia de paradas consiste en dividir las paradas por empresa y compartir las ganancias del corredor u otro método para equiparar el pasaje transportado.

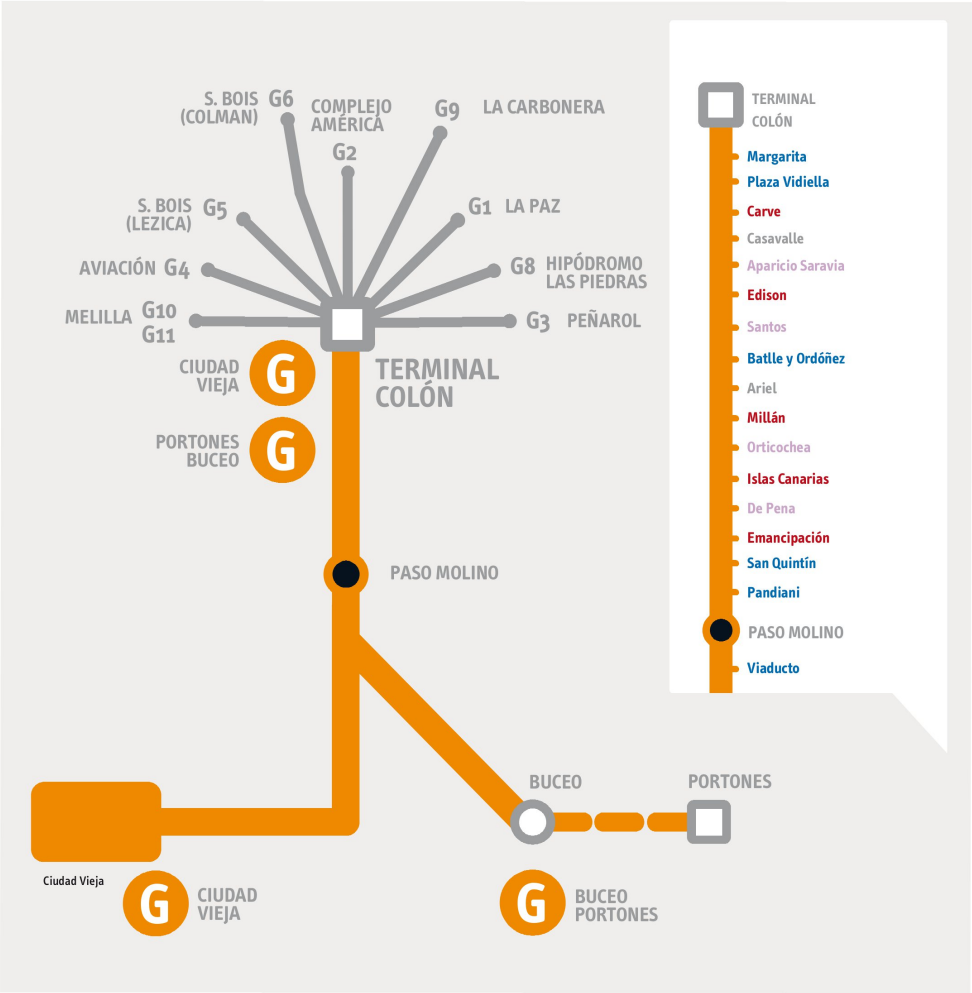

Figura 4.4: Gráfico de las paradas alternativas. Gris: Parada Eliminada. Azul: línea G de Coectc y de Cutcsa. Rojo: línea G de Coectc. Violeta: G de Cutcsa. - Imagen original extraída de montevideo.gub.uy

Una empresa se detendrá en las paradas pares y la otra en las impares y algunas de mayor aglomeración de pasaje/tiempo serán realizadas por las dos. Cada empresa viajará por el corredor a 4 minutos de frecuencia (como en la actualidad). Si se reduce

el número de paradas que hace un ómnibus, aumentará su velocidad promedio y no se deberá resentir en demasía el servicio ya que la disminución de la frecuencia en una parada se contrarresta con el aumento promedio de velocidad.

El cambio de alternar paradas es el que más aumenta la velocidad media y tal vez es uno de los más sencillos de implementar en la realidad.

Cambio básico de semáforos

Al hacer el relevamiento de los datos obtenidos de la configuración de los semáforos del Corredor Garzón, se constató que en todas las intersecciones en donde una línea de ómnibus que circula por el corredor tiene un viraje a la izquierda, se hace detener el tránsito de la derecha de la misma, cada vez que el corredor central tiene la luz verde. Esto no parece tener mucho sentido ya que podrían seguir circulando por el corredor sin ningún tipo de problema, el carril que hay que detener es el de la izquierda del ómnibus cuando una línea dobla a la izquierda pero no los dos carriles al mismo tiempo. No se tiene conocimiento si esto corresponde a un error en la configuración, un tema de costos o facilidad para manejar los dos semáforos de los carriles paralelos juntos. Como esto ocurre en varias intersecciones y en ambos sentidos este cambio mejora la velocidad promedio de los autos que circulan por los dos carriles.

Este cambio se aplicó en las siguientes intersecciones:

- Islas Canarias: dobla línea 409 hacia la izquierda, orientado a Colón (Norte).
- Camino Ariel: doblan líneas como la 2 y la 148 hacia la izquierda, orientado a Paso Molino (Sur).
- Camino Casavalle: dobla línea 174 hacia la izquierda, orientado a Paso Molino (Sur).

4.4.4. Valores numéricos al aplicar los cambios

Para determinar cuales son los cambios que logran mejores rendimientos se elabora la tabla 4.5. Esta se basa en el tráfico medio, ya que solo se quiere realizar una comparación sencilla de las mejoras realizadas. Estos cambios son acumulativos, por lo que se hacen uno después del otro. Se puede apreciar que el que logra una mayor diferencia es la utilización de paradas alternadas.

Cuadro 4.5: Valores numéricos del escenario alternativo con su velocidad promedio ómnibus (vpb) y velocidad promedio vehículos(vpv) comparando el *fitness* para el tráfico medio

	$vbp(km/h)$	$vvp(km/h)$	*Fitness*	Mejora(%)
Base	14.59	28.81	12.05	-
Eliminar Paradas	15.44	29.03	12.35	2.4
Eliminar Peatonales	16.02	29.32	12.59	4.4
Paradas alternadas	19.17	28.88	13.34	10.7
Cambio reglas	18.50	29.70	13.39	11.1

Una vez que se aplican todos las modificaciones sobre el escenario, se realiza un análisis para los demás tipos de tráfico. Como se ve en la tabla 4.6 se obtienen mejores rendimientos en todos los tipos de tráfico estudiados y el mejor rendimiento se obtiene cuando el tráfico es alto.

Cuadro 4.6: Mejoras del escenario alternativo para las velocidades promedio de los ómnibus(vpb) y de otros vehiculos (vpv) en el escenario alternativo para distintos tipos de tráficos

	$vbp(km/h)$	$vvp(km/h)$	Fitness	Mejora *fitness*(%)
Tráfico Bajo	20.72	33.18	14.97	11.5
Tráfico Medio	18.50	29.70	13.39	11.1
Tráfico Alto	18.60	27.17	12.7	12.6

Una vez que tenemos el escenario alternativo se procede a aplicarle el algoritmo cuyo resultado veremos a continuación.

4.4.5. Resultados de la evaluación sobre el escenario alternativo

El escenario alternativo supuso una mejora sustancial en comparación con el caso base (11 % en el valor de fitness). Se procede a aplicarle el algoritmo para determinar si aún hay posibilidad de mejorar los valores de velocidad y *fitness*.

Los resultados obtenidos en la tabla 4.7 mejoran claramente el rendimiento del escenario alternativo y por supuesto del caso base en todos los tipos de tráfico. Comparando con la realidad actual se logran mejoras de hasta 37 %.

Al comparar los resultados obtenidos se aprecia que cuanto más densidad de tráfico, mayor es el porcentaje de mejora. Además un resultado interesante es que las diferencias entre los valores de los distintos tipos de tráfico se redujo.

Cuadro 4.7: Mejoras obtenidas al aplicar el algoritmo evolutivo sobre el escenario alternativo, comparando las velocidades de ómnibus(vpb), otros vehículos(vpv) y el *fitness* con cada tipo de tráfico contra el escenario base.

Tráfico	$vbp(km/h)$	$vpv(km/h)$	*Fitness*		Mejora *fitness* (%)	
			Promedio	Mejor	Promedio	Mejor
Bajo	23.15±0.36	34.43±0.33	15.99±0.08	16.10	19.1	19.90
Medio	21.83±0.50	33.89±0.22	15.47±0.09	15.65	28.3	29.87
Alto	21.46±0.54	33.41±0.38	15.24±0.19	15.50	34.8	37.10

En la gráfica 4.5 se puede apreciar la comparación en la duración de los viajes. Se produce una gran reducción en la duración de los viajes de los ómnibus en los tres tipos de tráfico, mientras para el caso de los vehículos la mayor diferencia ocurre cuando el tráfico es alto.

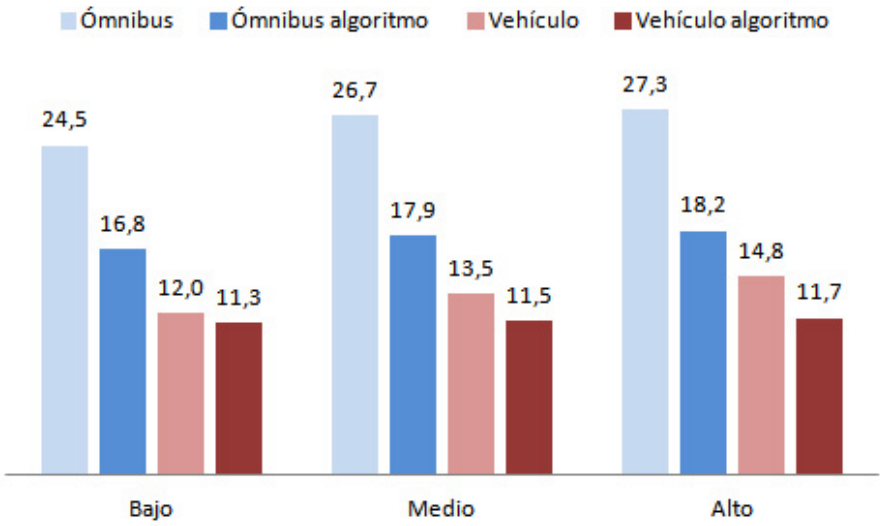

Figura 4.5: Comparación de la duración en minutos de los viajes realizados sobre el escenario base y al aplicar el algoritmo evolutivo al escenario alternativo, de ómnibus y otros vehículos en el recorrido completo del corredor Garzón, para los diferentes tipos de tráfico.

Otra vez podemos aplicar el criterio de significancia estadística(4.1) para comprobar que la mejora es significativa tanto al comparar con los valores del caso base como con los del alternativo.

4.4.6. Variación de la función de *fitness*

La función *fitness* (3.2) utilizaba los pesos $x = y = 1$ lo que representa un balance equitativo para ómnibus y vehículos. Estos pesos pueden ser variados en función de lo que se necesite, por lo que se realizaron pruebas con dos tipos de pesos para comparar como varían las velocidades cuando se da más peso a un tipo de vehículo sobre el otro.

Prioridad ómnibus

Este caso dará más prioridad a los ómnibus. Esto se sostiene en el hecho que uno de los objetivos buscados por la IMM es que se utilice más el transporte colectivo como parte de su Plan de Movilidad Urbana (IMM, 2010). Con la premisa que al mejorar la duración del viaje en ómnibus en relación al del auto por el corredor, las personas que utilizan auto para sus viajes optarán por el transporte colectivo. Por tanto se experimentó cambiando los pesos de la función *fitness* con un peso de 70 % para los ómnibus y 30 % al resto de los vehículos.

Prioridad a otros vehículos

Este caso asigna 70 % del peso a los vehículos y 30 % a los ómnibus. Es el caso opuesto al anterior y resulta útil para poder comparar como varían los valores de las velocidades.

Resultados

La siguiente tabla compara las velocidades promedio de ómnibus y vehículos para los tres tipos de pesos que se probaron y por cada tipo de tráfico. El caso 50-50 es el caso base donde los pesos son iguales, 70-30 es el caso con más prioridad para los ómnibus y el 30-70 más prioridad a los otros vehículos. Se analiza cuanto varían las velocidades de ómnibus (var. vpb) y otros vehículos(var. vpv) comparando contra el caso 50-50 de cada tipo de tráfico.

Cuadro 4.8: Valores obtenidos al modificar los pesos de la función de *fitness*, analizando las variaciones en la velocidad promedio de ómnibus (vpb), otros vehículos (vpv) y *fitness*.

Tráfico	pb(%) pv (%)	vpb	vpv	fitness	var. vpb(%)	var. vpv(%)	var. *fitnesss*(%)
Bajo	50-50	17.92±0.18	34.30±0.40	14.50±0.14	-	-	-
	70-30	17.93±0.23	34.06±0.17	12.65±0.11	+0.07	-0.70	-12.79
	30-70	17.55±0.23	34.71±0.21	16.42±0.10	-2.06	+1.18	+13.21
Medio	50-50	16.95±0.32	33.29±0.29	13.95±0.15	-	-	-
	70-30	17.29±0.27	33.08±0.14	12.24±0.12	+2.0	-0.62	-12.30
	30-70	16.71±0.42	33.79±0.31	15.92±0.11	-1.41	+1.49	+14.11
Alto	50-50	16.51±0.60	32.90±0.25	13.72±0.17	-	-	-
	70-30	16.72±0.14	32.79±0.26	13.75±0.07	+1.24	-0.33	+0.19
	30-70	15.48±0.42	33.20±0.25	15.49±0.16	-6.23	+0.92	+12.87

Los resultados indican que al variar los pesos de la función *fitness* las velocidades promedio de los vehículos se ve afectada. En el caso de dar más prioridad a los ómnibus se produce como cabía esperar un aumento en su velocidad promedio y una leve baja en la velocidad promedio del resto de los vehículos. Cuando el tráfico es bajo este cambio casi no aumenta la velocidad de los ómnibus. Una explicación posible de este comportamiento es que ya se llegó a un limite máximo y no se puede mejorar más.

Al dar más prioridad a los otros vehículos se produce un aumento en su velocidad y una disminución en la velocidad de los ómnibus la cual es muy evidente en el caso de tráfico alto. Este resultado permite apreciar como estos valores son fuertemente afectados por la densidad de tráfico que se estudie.

En general las variaciones en las velocidades no son grandes pero suficientemente apreciable para tener cierta libertad al plantear distintos objetivos que tiendan a favorecer un tipo u otro de vehículos.

4.4.7. Eficiencia computacional

Se realiza un estudio de la eficiencia computacional del algoritmo para analizar los tiempos de ejecución cuando se usan varios procesadores y como se comporta su capacidad de paralelismo.

Se evalúan nueve ejecuciones del algoritmo; tres con cada tipo de tráfico: alto, medio y bajo, para estudiarlo en diferentes contextos. El algoritmo utiliza 32 hilos de ejecución por lo que utilizamos esa cantidad de núcleos.

Las pruebas fueron realizadas sobre el node40 del Cluster Fing, con un procesador AMD Opteron 6272 2.09GHz, 48 GB RAM y 32 núcleos utilizados.

El *speedup* (S) mide la mejora de rendimiento de una aplicación al aumentar la cantidad de procesadores comparando con el rendimiento al usar un solo procesador.

$$S = \frac{T_1}{T_N} \tag{4.2}$$

Donde T_1 es el tiempo de ejecución del algoritmo serial o secuencial, y T_N el tiempo del algoritmo ejecutado sobre N procesadores.

La eficiencia computacional (E) corresponde al valor normalizado del *speedup* (entre 0 y 1) respecto a la cantidad de procesadores. Los valores cercanos a uno indican una alta eficiencia computacional.

$$E = \frac{T_1}{N * T_N} = \frac{S}{N} \tag{4.3}$$

Cuadro 4.9: Análisis de la eficiencia computacional comparando los tiempos de ejecución en serial y paralelo en minutos.

Instancia	Serial(m)	Paralelo(m)	*Speedup*	Eficiencia
bajo1	1572	59	26.64	0.83
bajo2	1571	59	26.62	0.83
bajo3	1183	44	26.88	0.84
medio1	3002	119	25.22	0.78
medio2	2195	82	26.76	0.83
medio3	3007	120	25.05	0.78
alto1	2920	110	26.5	0.82
alto2	4365	183	23.85	0.74
alto3	4276	177	24.15	0.75
		Promedio	25.7±1.1	0.80±0.03

El algoritmo paralelo logra una mejora sustancial en los tiempos de ejecución con un valor promedio del *speedup* de 25.7 y eficiencia promedio de 0.8, lo cual puede considerarse como buenas métricas.

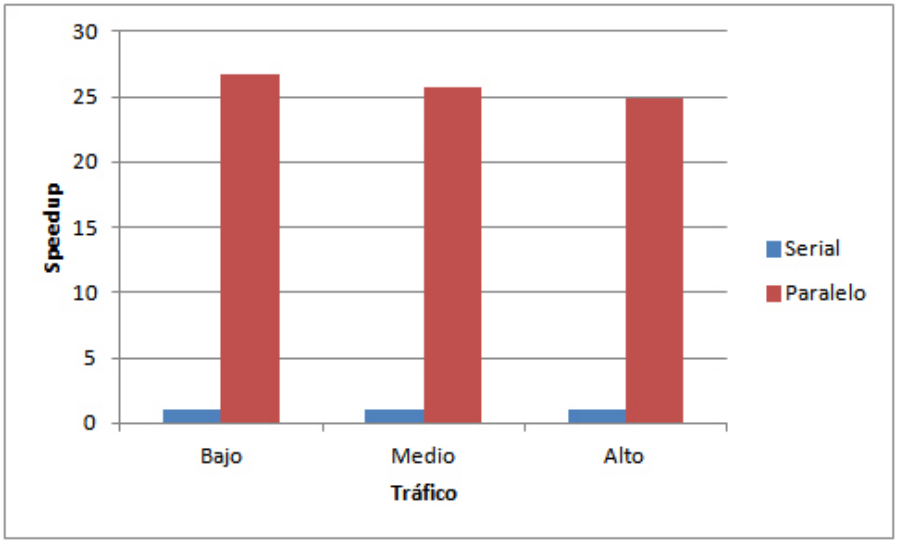

Figura 4.6: Comparación de los *speedup* promedios para cada tipo de tráfico. Se representa el caso serial que corresponde al *speedup* = 1 para fines de comparación.

La gráfica 4.6 muestra como a medida que aumenta el tráfico disminuye el speedup. Esto sucede por que está influenciado por los accesos al disco duro, al tener más vehículos circulando en la simulación se tiene que leer y escribir más información en los archivos, lo que aumenta el tiempo de ejecución del algoritmo, aunque como se ve no tiene un gran impacto.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

Luego de analizar los objetivos planteados al inicio del trabajo se puede afirmar que se cumplieron satisfactoriamente. La investigación de los trabajos relacionados permitió encontrar información relevante que ayudó a mejorar la solución presentada. Al estudiar el problema del tráfico, se constató que afecta a la población y al desarrollo de las ciudades. Por este motivo se considera imprescindible la búsqueda de nuevas soluciones. En el contexto de nuestro país (Uruguay), no se encontraron soluciones similares, por lo que este trabajo se plantea como un aporte interesante, que demuestra que existen las herramientas y el conocimiento necesario para realizarlo.

Las simulaciones del tráfico demostraron su utilidad al dar la flexibilidad necesaria para probar distintas variantes de forma sencilla, y poder diseñar un escenario alternativo con modificaciones agregadas que logró una mejora del 11 % en el valor de *fitness* comparando con el escenario base que representa la realidad.

A pesar de que el problema de sincronización de semáforos es un problema difícil de abordar, los resultados obtenidos muestran la capacidad de los algoritmos genéticos para resolver problemas de este tipo, obteniendo evidencia estadística que valida las mejoras producidas. En general el algoritmo genético propuesto logra una mejora de hasta 24.2 % (21.40 % en promedio) del valor de fitness comparando con la realidad actual, mientras el escenario alternativo obtiene una mejora de hasta 37.1 % (34.7 % en promedio) en el valor de fitness.

El enfoque multiobjetivo, aún siendo básico, permitió analizar las diferentes velocidades medias de ómnibus y otros vehículos, que fueron utilizadas para realizar análisis comparativos que enriquecieron el trabajo.

El desarrollo de algoritmos evolutivos con capacidad de paralelización son fundamentales, sobre todo en problemas complejos que requieren mucho poder de computo como el abordado. El algoritmo evolutivo obtuvo buenas métricas de *speedup* sin las cuales hubiera sido muy difícil realizar la cantidad de pruebas presentadas.

5.2. Trabajo futuro

El diseño de mapas para la simulación del tráfico requiere la realización de modificaciones para que sean reconocidos por el simulador, en algunos casos se aplicaron manualmente ya que las herramientas no brindaban la granularidad necesaria. Además,

la edición de los archivos que representan la configuración de semáforos, las líneas y paradas de ómnibus supone un proceso lento y propenso a errores. Por estos motivos, para un futuro, se sugiere el desarrollo o búsqueda de nuevas herramientas que automaticen o agilicen este trabajo.

El algoritmo evolutivo desarrollado, puede ser aplicado a otros lugares geográficos, modificando los datos de entrada: mapa, tráfico, configuración de los semáforos y recorrido de ómnibus. El alcance del trabajo solo se enfocó en la zona del Corredor Garzón pero sería interesante aplicarlo en otros escenarios para determinar su rendimiento.

Los trabajos de Montana y Czerwinski (1996) y Vogel et al. (2000) proponen la adaptabilidad del algoritmo en tiempo real, aunque esto requiere del agregado de sensores a la red, podría resultar en una mejora importante sobre todo, en zonas de gran densidad de tráfico.

Bibliografía

E. Alba y C. Cotta. On-line tutorial on evolutionary computation. En `http://neo.lcc.uma.es/TutorialEA/semEC/main.html`, 1997. Disponible online, consultado Abril 2015.

E. Alba y M. Tomassini. Parallelism and evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 6(5):443–462, 2002.

E. Alba, F. Almeida, M. Blesa, C. Cotta, M. Diaz, I. Dorta, J. Gabarró, J. González, C. León, L. Moreno, J. Petit, J. Roda, A. Rojas, y F. Xhafa. MALLBA: A library of skeletons for combinatorial optimisation. *Parallel Computing*, 32(5-6):415–440, 2006.

E. Bañobre y Á. Romero. Los BRT en corredores segregados como sistema óptimo de transporte urbano. En *Administrando en entornos inciertos = managing in uncertain environment*, 2009.

BBVA Research. Situación Automotriz Uruguay. En `https://www.bbvaresearch.com`, 2013. Disponible online, consultado noviembre 2014.

A. Bull. *Congestión de tránsito: el problema y cómo enfrentarlo.* Number 87. United Nations Publications, 2003.

S. Cahon, N. Melab, y E.-G. Talbi. Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3):357–380, 2004.

K. Deb. *Multi-objective optimization using evolutionary algorithms*, volumen 16. John Wiley & Sons, 2001.

A. Di Febbraro, D. Giglio, y N. Sacco. On applying petri nets to determine optimal offsets for coordinated traffic light timings. En *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, p. 773–778. IEEE, 2002.

J. J. Durillo y A. J. Nebro. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771, 2011.

El País. «Garzón: Olivera admitió errores». En `http://www.elpais.com.uy/`, 30 Julio 2013. Disponible online, consultado febrero 2015.

El País. «Ana Olivera: "No hay justificación, nos equivocamos con Garzón"». En `http://www.elpais.com.uy/`, 10 Enero 2015. Disponible online, consultado febrero 2015.

L. Eshelman. The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination. En *Foundations of Genetics Algorithms*, p. 265–283. Morgan Kaufmann Publishers, Inc., 1991.

G. Fagúndez. The Malva Project: A framework for comutational intelligence in C++. En `http://themalvaproject.github.io/`, 2014. Disponible online, consultado julio 2014.

C. Gagné y M. Parizeau. Open beagle: A new versatile c++ framework for evolutionary computation. En *GECCO Late Breaking Papers*, p. 161–168, 2002.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1 edition, Jan. 1989. ISBN 0201157675.

M. Haklay y P. Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.

A. Halati, H. Lieu, y S. Walker. Corsim-corridor traffic simulation model. En *Traffic congestion and traffic safety in the 21st century: Challenges, innovations, and opportunities*, 1997.

IMM. Plan de movilidad urbana. En `http://www.montevideo.gub.uy`, Febrero 2010. Disponible online, consultado febrero 2015.

INE. *Uruguay en cifras 2014*, capítulo Transporte y Comunicaciones (11), p. 93–99. 2014. URL `http://www.ine.gub.uy/`. Disponible online, consultado enero 2015.

Institute for Transportation & Development Policy (ITDP). Bus rapid transit planning guide. En `https://www.itdp.org`, Junio 2007. Disponible online, consultado febrero 2015.

R. H. J. Penner y C. Jacob. Swarm-based traffic simulation with evolutionary traffic light adaptation. En *Proceedings of Artificial Intelligence and Soft Computing*, 2002.

G. Kotushevski y K. A. Hawick. A review of traffic simulation software. Technical Report CSTN-095, Computer Science, Massey University, Albany, North Shore 102-904, Auckland, New Zealand, 2009.

J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems)*. A Bradford Book, 1 edition, 12 1992. ISBN 9780262111706.

D. Krajzewicz, G. Hertkorn, C. Rössel, y P. Wagner. Sumo (simulation of urban mobility). En *Proc. of the 4th middle east symposium on simulation and modelling*, p. 183–187, 2002.

G. Lim, J. Kang, y Y. Hong. The optimization of traffic signal light using artificial intelligence. En *Fuzzy Systems, 2001. The 10th IEEE International Conference on*, volumen 3, p. 1279–1282. IEEE, 2001.

T. Litman. Transportation cost and benefit analysis. 2009.

S. López, P. Hernández, A. Hernández, y M. García. Artificial neural networks as useful tools for the optimization of the relative offset between two consecutive sets of traffic lights. En J. Mira y J. V. Sánchez-Andrés, editors, *IWANN (2)*, volumen 1607 of *Lecture Notes in Computer Science*, p. 795–804. Springer, 1999.

D. McKenney y T. White. Distributed and adaptive traffic signal control within a realistic traffic simulation. *Engineering Applications of Artificial Intelligence*, 26(1):574–583, 2013.

M. Mitchell. *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*. A Bradford Book, reprint edition, 2 1998. ISBN 9780262631853.

D. J. Montana. Strongly typed genetic programming. *Evolutionary Computation*, 3(2): 199–230, 1995.

D. J. Montana y S. Czerwinski. Evolving control laws for a network of traffic signals. En *Proceedings of the First Annual Conference on Genetic Programming*, p. 333–338. MIT Press, 1996.

L. G. Papaleondiou y M. D. Dikaiakos. Trafficmodeler: A graphical tool for programming microscopic traffic simulators through high-level abstractions. En *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, p. 1–5. IEEE, 2009.

A. K. Rathi. Urban network traffic simulation:traf-netsim program. *Journal of Transportation Engineering*, 116(6):734–743, November-December 1990.

E. Rechenberg. *Ingo Rechenberg Evolutionsstrategie Optimierung technischer Systeme nach Prinzipien der biologishen Evolution*. mit einem Nachwort von Manfred Eigen, Friedrich Frommann Verlag, Struttgart-Bad Cannstatt, 1973.

N. M. Rouphail, B. B. Park, y J. Sacks. Direct signal timing optimization: Strategy development and results. En *XI Pan American Conference in Traffic and Transportation Engineering*. Citeseer, 2000.

J. Sánchez, M. Galán, y E. Rubio. Applying a traffic lights evolutionary optimization technique to a real case: "Las Ramblas" area in Santa Cruz de Tenerife. *Evolutionary Computation, IEEE Transactions on*, 12(1):25–40, 2008.

J. J. Sanchez, M. Galan, y E. Rubio. Genetic algorithms and cellular automata: a new architecture for traffic light cycles optimization. En *Evolutionary Computation, 2004. CEC2004. Congress on*, volumen 2, p. 1668–1674. IEEE, 2004.

J. J. Sánchez-Medina, M. J. Galán-Moreno, y E. Rubio-Royo. Traffic signal optimization in "La Almozara" district in Saragossa under congestion conditions, using genetic algorithms, traffic microsimulation, and cluster computing. *Intelligent Transportation Systems, IEEE Transactions on*, 11(1):132–141, 2010.

D. Smith y J. McIntyre. Handbook of simplified practice for traffic studies. Technical report, 2002.

W. M. Spears. *Evolutionary algorithms: the role of mutation and recombination*. Springer Science & Business Media, 2000.

D. H. Stolfi. Optimización del tráfico rodado en ciudades inteligentes. Master's thesis, Universidad de Málaga, 2012.

Subrayado. «Expertos en tránsito: "el colapso está establecido" en Montevideo». En `http://www.subrayado.com.uy`, 27 Mayo 2013. Disponible online, consultado noviembre 2014.

K. T. K. Teo, W. Y. Kow, y Y. Chin. Optimization of traffic flow within an urban traffic light intersection with genetic algorithm. En *Computational Intelligence, Modelling and Simulation (CIMSiM), 2010 Second International Conference on*, p. 172–177. IEEE, 2010.

The Institute for Transportation and Development Policy (ITDP). The BRT Standard, 2014 edition. En `https://www.itdp.org/`, 2014. Disponible online, consultado noviembre 2014.

A. Vogel, C. Goerick, y W. Von Seelen. Evolutionary algorithms for optimizing traffic signal operation. En *Proceedings of the European symposium on intelligent techniques (ESIT)*, p. 83–91. Citeseer, 2000.

M. Wall. GAlib: A C++ library of genetic algorithm components. *Mechanical Engineering Department, Massachusetts Institute of Technology*, 87:54, 1996.

C. E. Wallace, K. Courage, D. Reaves, G. Schoene, y G. Euler. Transyt-7f user's manual. Technical report, 1984.

C.-B. Yang y Y.-J. Yeh. The model and properties of the traffic light problem. 1996.