# Algoritmos Evolutivos en la Planificación de Estrategias de Señalización

## Reducción de Emisiones Vehiculares

Matias Péres y German Ruiz
Instituto de Computación
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay

***Abstract*— El transporte vehicular se ha convertido por lejos en la principal fuente de contaminación ambiental y congestión del tráfico en zonas urbanas. El concepto de optimizar ciclos de semáforos para reducir las emisiones vehiculares fue tratado anteriormente con herramientas macroscópicas y analíticas, y no se consideraba el comportamiento individual de conducción. En años recientes se han desarrollado poderosas aplicaciones para el modelado del tráfico y modelos de emisiones. Sin embargo, la cantidad de semáforos operativos hace de la planificación conjunta un problema cada vez más complejo debido a la gran cantidad de combinaciones posibles de estados de los semáforos. En este contexto, dada la complejidad y la naturaleza estocástica del sistema a optimizar, se propone la utilización de un algoritmo evolutivo paralelo para planificación de ciclos con el objetivo de reducir las emisiones vehiculares más importantes utilizando como escenarios zonas de Montevideo, ciudad con más semáforos por automóvil en toda Latinoamérica. Se proponen operadores evolutivos específicos y un modelo maestro-esclavo de modo de incrementar la eficiencia computacional. Se realizan comparaciones con otras técnicas comprobando como el algoritmo propuesto es capaz de obtener reducciones significativas.**

***Keywords—Contaminación Ambiental Vehicular; Planificación de Ciclos de Semáforos; Emisiones Vehiculares; Algoritmos Evolutivos; Paralelismo; SUMO;***

### I. INTRODUCCIÓN

La preocupación por la preservación del Medio Ambiente se ha convertido en un tema de importancia capital en las sociedades de nuestro tiempo. Dentro de este ámbito la contaminación atmosférica, o contaminación del aire, ocupa un lugar preferente en la atención de la comunidad científica y en las inquietudes de la ciudadanía [1].

El transporte vehicular se ha convertido por lejos en la principal fuente de contaminación ambiental y congestión del tráfico en zonas urbanas. El continuo crecimiento del tráfico ha levantado preocupación sobre el impacto de las emisiones de tráfico en la salud humana y en la calidad del ambiente urbano, impulsando la demanda de un sistema normativo coherente para la gestión del tráfico, calidad del aire y emisión a nivel urbano así como a escalas regionales y nacionales [2].

Aunque las emisiones vehiculares han mejorado substancialmente en las últimas dos décadas, permanecen como principales contribuyentes de $CO$, $NO_x$ y smog (compuestos orgánicos volátiles, $HC$ o $VOC$) a nivel nacional. Además, la combustión de gasolina es responsable de una fracción significante de emisiones de gases de efecto invernadero ($CO_2$) [3].

Montevideo concentra la mayor parte de la actividad industrial y comercial del Uruguay. En los últimos años se ha promovido el estudio de los efectos de estas actividades en la calidad del aire y en la población brindándose informes semanales según el ICAire (Indice de Calidad del Aire). En Montevideo las condiciones geomorfológicas y climáticas son favorables a la dispersión natural de las emisiones contaminantes. Estas características hacen que la contaminación del aire no sea un problema grave como en otras ciudades de América Latina y el Caribe. Sin embargo, se han detectado situaciones que justifican atención especial en zonas específicas de la ciudad.

La contribución del sector transporte a la contaminación del aire se debe al aumento de vehículos y a las características y estado de los mismos, así como a la calidad y composición de los combustibles utilizados (el mal estado de las unidades suele significar poca eficiencia de combustión, provocando combustión incompleta) .

Datos del año 2013 estiman que unos 450.000 autos circulan por día en las calles de Montevideo mientras que en 2005 circulaban 250.000. La compra de autos cero kilómetro sigue en aumento y por día comienzan a circular unos 150 autos nuevos en la capital. Los expertos coinciden en que el congestionamiento es algo ya instalado y que la infraestructura vial no acompasó el crecimiento del parque automotor. Respecto al transporte de pasajeros, en Montevideo hay 135 líneas urbanas y suburbanas, pero es importante destacar que hay 4450 paradas, siendo cada una un punto fijo de emisión de contaminantes. Además, una velocidad promedio de 16 kilómetros por hora ocasiona considerables emisiones de contaminantes por los caños de escape [4].

Según un estudio llevado a cabo por la Facultad de Ciencias de la Universidad de la República con el apoyo del Instituto de Geofísica de la Universidad Autónoma de México (UNAM), la contaminación en el aire de Montevideo es mayor que en Roma y similar a la de una ciudad China de 8 millones de habitantes. Cabe mencionar que La República Popular China, lidera la

tabla de países con mayores niveles de emisión de gases de efecto invernadero, según las estadísticas informadas por la Organización Mundial de la Salud (OMS) [5].

Históricamente, las herramientas de optimización de ciclos de semáforos eran desarrolladas para reducir los retrasos y paradas experimentadas por conductores urbanos. El concepto de optimizar ciclos de semáforos para reducir el consumo de combustible y las emisiones fue tratado por primera vez por Robertson et al. [6]. Sin embargo, en ese momento el tráfico era simulado por herramientas macroscópicas y analíticas, y no se consideraba el comportamiento individual de conducción. La relación entre la actividad del tráfico, el consumo de combustible y las emisiones vehiculares era una simplista y lineal.

En años recientes se han desarrollado poderosas aplicaciones para el modelado del tráfico y modelos de emisiones. Los modelos de emisiones se desarrollaron para estimar segundo a segundo las emisiones individuales de los vehículos basándose en modos de un ciclo de conducción común. Estos dos tipos de modelos microscópicos han sido acoplados para estimar las emisiones instantáneas basadas en las actividades segundo a segundo de vehículos con comportamiento individual [7].

Montevideo es la ciudad con más semáforos por automóvil en toda Latinoamérica. La gran cantidad de semáforos operativos hace de la planificación conjunta un problema cada vez más complejo debido a la gran cantidad de combinaciones posibles de estados de los semáforos. En este sentido, se propone la utilización de algoritmos evolutivos para planificación de ciclos con el objetivo de reducir emisiones utilizando como escenarios zonas de Montevideo, dada la complejidad y la naturaleza estocástica del sistema a optimizar.

Para la evaluación de los programas de ciclos generados hemos utilizado el simulador de tráfico microscópico *SUMO* (*Simulator of Urban Mobility*). Este simulador trabaja a modo de caja negra que tiene como entrada el programa de ciclos para una determinada instancia de área urbana y devuelve los valores de adecuación de dicho programa para la instancia evaluada.

## II. Contexto Teórico

### A. *Algoritmos Evolutivos*

Muchos problemas de optimización caracterizados por una un espacio de búsqueda extenso y no lineal que aparecen en los ámbitos de las ingenierías son muy difíciles de solucionar por medio de técnicas tradicionales, por lo que a menudo se aplican algoritmos evolutivos, inspirados en la naturaleza, que recogen un conjunto de modelos basados en la evolución de los seres vivos.

Los algoritmos evolutivos trabajan con una población de individuos, que representan las soluciones candidatas al problema. Esta población se somete a ciertas transformaciones con el fin de aumentar su *fitness*, medida relacionada a la función objetivo, y después a un proceso de selección, que favorece a los mejores. Cada ciclo de transformación y selección constituye una generación, de forma que después de cierto número de generaciones se espera que el mejor individuo de la población esté cerca de la solución buscada. Los algoritmos evolutivos combinan la búsqueda aleatoria, dada por las transformaciones de la población, con una búsqueda dirigida dada por la selección. Suele hablarse de tres paradigmas principales de algoritmos evolutivos:

- Programación evolutiva
- Estrategias evolutivas
- Algoritmos genéticos

La estrategia de evolución más conocida hoy en día son los algoritmos genéticos. Los algoritmos genéticos constituyen una técnica poderosa de búsqueda y optimización con un comportamiento altamente paralelo, inspirado en el principio darviniano de selección natural y reproducción genética. En este principio de selección de los individuos más aptos, tienen mayor longevidad y por tanto mayor probabilidad de reproducción. Los individuos descendientes de estos individuos tienen una mayor posibilidad de transmitir sus códigos genéticos a las próximas generaciones [8].

Se supone que los individuos (posibles soluciones del problema), pueden representarse como un conjunto de parámetros, los cuales agrupados forman una cadena de valores (a menudo referida como cromosoma). Si bien el alfabeto utilizado para representar los individuos no debe necesariamente estar constituido por el {0,1}, buena parte de la teoría en la que se fundamentan los Algoritmos Genéticos utiliza dicho alfabeto. La población inicial se genera utilizando un método aleatorio o con alguna heurística específica para el problema.

La *función de fitness* debe ser diseñada para cada problema de manera específica. Dado un cromosoma particular, la función de adaptación le asigna un número real, que se supone refleja el nivel de adaptación al problema del individuo representado por el cromosoma.

Durante la fase reproductiva se seleccionan utilizando una *estrategia de selección* a los individuos de la población para cruzarse y producir descendientes, que constituirán, una vez mutados, la siguiente generación de individuos. Cuanto mayor sea la adaptación de un individuo al problema, mayor seré la probabilidad de que el mismo sea seleccionado para reproducirse, cruzando su material genético con otro individuo seleccionado de igual forma. Según dicho esquema, los individuos bien adaptados se escogerán probablemente varias veces por generación, mientras que los pobremente adaptados al problema, no se escogerán más que de vez en cuando.

Una vez seleccionados dos padres, sus cromosomas se combinan, utilizando habitualmente los *operadores de cruzamiento y mutación*. Las formas básicas de dichos operadores se describen a continuación:

- El operador de cruce, toma dos padres seleccionados y corta sus cadenas de cromosomas en una posición

escogida al azar, para producir dos sub-cadenas iniciales y dos sub-cadenas finales. Después se intercambian las sub-cadenas finales, produciéndose dos nuevos cromosomas. Ambos descendientes heredan genes de cada uno de los padres. Este operador se conoce como *operador de cruzamiento de un punto.*

- El operador de mutación se aplica a cada hijo de manera individual, y consiste en la alteración aleatoria (normalmente con probabilidad pequeña) de cada gen componente del cromosoma.

Si bien puede en principio pensarse que el operador de cruce es más importante que el operador de mutación, ya que proporciona una exploración rápida del espacio de búsqueda, este último asegura que ningún punto del espacio de búsqueda tenga probabilidad cero de ser examinado, y es de capital importancia para asegurar la convergencia de los Algoritmos Genéticos.

Si el Algoritmo Genético ha sido implementado correctamente, la población evolucionará a lo largo de las generaciones sucesivas de tal manera que la adaptación media extendida a todos los individuos de la población, así como la adaptación del mejor individuo se irán incrementando hacia el óptimo global [9]. Se utiliza un criterio de parada que usualmente fija una cantidad determinada de generaciones o tiempo de ejecución. El algoritmo retorna la mejor solución encontrada según la función de fitness.

Las técnicas de procesamiento paralelo y distribuido se aplican al modelo clásico de Algoritmos Genéticos con el objetivo de obtener mejoras desde el punto de vista de la eficiencia y para perfeccionar la calidad de la búsqueda genética. Desde la perspectiva de la eficiencia, paralelizar un AG permite afrontar la lentitud de convergencia para problemas cuya dimensión motiva el uso de poblaciones numerosas, o múltiples evaluaciones de funciones de fitness costosas. Desde el punto de vista algorítmico, los Algoritmos Genéticos Paralelos (AGP) pueden explotar el paralelismo intrínseco del mecanismo evolutivo, trabajando simultáneamente sobre varias poblaciones semi-independientes para resolver el mismo problema. Eventuales intercambios de soluciones (*migraciones*) introducen diversidad para evitar problemas de convergencia en óptimos locales. Complementariamente, los AGP pueden aprovechar características de paralelismo propias del problema, analizando concurrentemente diferentes secciones del espacio de búsqueda.

El modelo que surge de distribuir funcionalmente el algoritmo, asignando a diferentes procesadores distintas etapas del mecanismo evolutivo, se conoce como modelo maestro-esclavo. Es usual distribuir la evaluación de la función de fitness, que generalmente involucra un tiempo de ejecución mayor que el de los sencillos operadores evolutivos. Por otra parte, el enfoque orientado a la distribución de datos da lugar a un modelo de población distribuida, organizada en sub-poblaciones semi-independientes. Estos dos enfoques, reconocidos desde las primeras etapas de la investigación en el área, han evolucionado y se han diversificado, dando lugar a múltiples modelos de AGP [10].

### *B. Simuladores de Tráfico*

Uno de los sistemas que se estudia mejor mediante una simulación por ordenador es una red de tráfico. Es más común que se experimente con las redes de tráfico en un entorno simulado de computadora porque experimentando con el tráfico en el entorno real no es práctico. La simulación de tráfico es muy estudiada y se puede clasificar en cuatro tipos: *macroscópicas*, *mesoscópica*, *microscópica* y *nanoscópica.*

Los simuladores macroscópicos modelan el flujo de tráfico utilizando modelos matemáticos alto nivel derivados a menudo de dinámica de fluidos, por lo tanto son simulaciones continuas. Este tipo de simulación se encarga de todos los vehículos de la misma manera y como un grupo. Utiliza agregado de entrada y salida de variables como la velocidad, flujo y densidad. Los simuladores macroscópicos son más útiles para la simulación de sistemas de tráfico de área amplia, que no requieren modelado detallado, como autopistas y carreteras interregionales. Este enfoque no es muy realista porque en la vida real hay muchos tipos diferentes de vehículos conducidos por personas diferentes que tienen sus propios estilos y comportamientos. Sin embargo, es rápida y precisa pero no está bien adaptado a los modelos urbanos en general.

Los simuladores microscópicos modelan entidades individuales (por ejemplo, vehículo, conductor etc.) por separado a un alto nivel de detalle y se clasifican como simulaciones discretas. Aquí, las interacciones generalmente se rigen por las lógicas auto-siguiente y el cambio de carril. Así, detalles de flujo de tráfico, generalmente asociados con simulación macroscópica son las propiedades emergentes de la simulación microscópica. Estos simuladores pueden modelar el flujo de trafico de manera más realística que los simuladores macroscópicos, debido a los detalles extras añadidos modelando las entidades individualmente. Los simuladores microscópicos son ampliamente utilizados para evaluar el nuevo control de tráfico y tecnologías de gestión, así como realizar análisis de las operaciones de tráfico existentes.

Por otro lado, los simuladores mesoscópicos llenan la brecha entre macro y micro simuladores. Normalmente describen entidades de tráfico a un mayor nivel de detalle, que modelos macroscópicos pero sus comportamientos e interacciones están en bajos niveles de detalle. Una nueva tendencia de simulación de tráfico son los modelos nanoscópicos que amplían la visión del vehículo, dividiéndolo en partes. Se utiliza sobre todo en la conducción autónoma y está en una

relación estricta con automatizada robótica, debido a la necesidad de simular los sensores.

En este trabajo nos enfocamos en los simuladores de tipo microscópico, siendo de más interés y utilidad el simulador *SUMO* por sus características.

*1) SUMO*

SUMO (*Simulation of Urban Mobility*) es un simulador de tráfico microscópico de código abierto, altamente portátil diseñado para manejar grandes carreteras. El simulador se mantiene en desarrollo en el Instituto de Sistemas de Transporte en el Centro Aeroespacial Alemán. Se halla licenciado bajo la GPL. Sus características incluyen: no colisión, movimiento de vehículo, multi-vía, calles con cambio de carril, velocidad de ejecución rápida, asignación dinámica de usuario y otros.

SUMO proporciona una fuente de información constante sobre el flujo de vehículos. Esto es fundamental para poner usar un algoritmo avanzado y hacer planificación automática. SUMO nos permite además, simular factores medioambientales basados en el modelo *HBEFA* (*HandBook Emission Factors for Road Transport*) [12]. A través de este modelo podemos simular diferentes condiciones de vehículos con información sobre aceleraciones, deceleraciones, frenado, velocidades máximas, así como su repercusión en las emisiones basadas en HBEFA: $CO$, $CO_2$, $HC$, $PM_x$ y $NO_x$. En este trabajo nos hemos centrado en los agentes contaminantes $CO$, $CO_2$, $HC$, y $NO_x$.

*C. Watchmaker Framework*

Watchmaker Framework es una librería extensible, de alta performance y orientada a objetos para implementar algoritmos evolutivos/genéticos independientes de la plataforma en Java.

Entre sus características se cuenta con modelos de evolución basado en islas o *steady-state*, procesamiento distribuido, etc. Resaltamos las utilizadas en el trabajo actual:

- *Evolution Engine Multi-Hilo*: toma ventaja del paralelismo para incrementar la performance de las computadoras multi-núcleo y multi-procesador.
- *No-Invasivo*: objetos de cualquier tipo pueden ser evolucionados sin que la clase evolucionante tenga que implementar una interfaz particular o extender de una clase base común. Esto significa que no hay restricciones a la hora de la implementación del tipo evolucionante y ninguna dependencia de las clases de la librería. El tipo evolucionante está completamente desacoplado.
- *Estrategias de Selección Acoplables*: *rueda de ruleta*, *selección por torneo* y muchas otras estrategias de selección son provistas por la librería. Alternativamente se puede implementar una estrategia de selección propia de forma rápida y fácil.

## III. Estrategia de Resolución

Se propuso una estrategia de optimización basada en un algoritmo genético de modo de obtener ciclos de semáforos eficientes desde un punto de vista ambiental, tomando en cuenta emisiones vehiculares y cantidad de combustible consumido, sobre varios escenarios montevideanos con un número considerable de semáforos y vehículos.

Durante el proceso se mantuvo una población de soluciones que evolucionan de acuerdo a operaciones de selección, cruzamiento, reemplazo y mutación, siguiendo la idea de la supervivencia de los individuos más aptos. El grado de adaptación de un individuo se evalúa de acuerdo al problema a resolver, mediante una función de fitness.

Cuando el algoritmo genético produce una nuevo individuo este es decodificado para ser introducido en la nueva configuración de SUMO que es iniciado para procesar el escenario de tráfico. Tras la simulación, SUMO devuelve los valores necesarios para computar la función de fitness.

*A. Representación*

Consideremos el siguiente conjunto de intersecciones: $I = \{c_1, c_2, \dots, c_n\}$. Los semáforos ubicados en una misma intersección deben estar sincronizados unos con otros, ya que de otro modo no cumplirían su función. Es por ello que se dispone de programación única para cada intersección, $P = \{p_1, p_2, \dots, p_n\}$. Cada programa $p_i$ consta de una serie de estados con determinada duración. Cada estado es una combinación válida de colores, es decir, debe seguir reglas específicas establecidas para no generar problemas.

El simulador utilizado SUMO ya nos provee de los posibles estados de luces válidos, que no pueden ser modificados durante la optimización y a los cuáles se les debe suministrar la duración. De esta forma se evita cualquier combinación inválida restringiendo la optimización a trabajar con los estados provistos.

Por ejemplo, supongamos que tenemos 2 semáforos A y B que impiden el colapso del tráfico proveniente de dos calles de sentido único que se intersecan. Este escenario da lugar al siguiente programa de 4 estados: V-R 20s; A-R 5s; R-V 30s; R-A 5s. Tenemos entonces la siguiente combinación: A en verde durante 20 segundos y B en rojo durante 20 segundos; A en amarillo durante 5 segundos y B en rojo durante 5 segundos; y así hasta que se termine el programa y luego se comienza desde el primer estado.

Siguiendo lo establecido por sumo, la representación utilizada es una cadena de naturales con las duraciones de los estados de los programas de cada intersección dispuestos en forma sucesiva, junto al tiempo de *offset* o desplazamiento del programa, tiempo inicial en que comienza el programa al inicio de la simulación.

$$\mathrm{o}_1, d_{11}, d_{12} \dots d_{1k_1}, \mathrm{o}_2, d_{21}, d_{22} \dots d_{2k_2} \dots \mathrm{o}_n, d_{n1}, d_{n2} \dots d_{nk_n}$$

Cada elemento de la representación se corresponde, por tanto, con la duración de un estado de colores válido de una intersección ($d_{ij}$) o el tiempo inicial del programa ($o_i$). Cabe mencionar que los programas pueden tener distinto número de estados ($k_i$). Los estados con luces amarillas tienen una duración constante de 5 segundos por lo que no se consideran en el número $k_i$ y no se incluyen en la representación.

### B. Inicialización

La población inicial de individuos se generó asignando a sus alelos valores aleatorios representando las duraciones de estado de modo de explorar todo el rango de posibles soluciones. Se utiliza el intervalo $[5,60] \in \mathrm{N}$ para la duración en segundos de los estados de los programas. A los offsets se les asignó un valor en el intervalo $[0,60 \times k_i + 5 \times k_i] \in \mathrm{N}$ intervalo que permite al programa iniciar en cualquiera de sus estados.

### C. Selección

Se propuso la utilización de selección por torneo con una estrategia elitista de modo de conservar el 10% de los mejores individuos encontrados hasta el momento. La selección por torneo tiene varias ventajas: es eficiente de codificar, trabaja en arquitecturas paralelas y permite ajustar fácilmente la presión de la selección variando el tamaño del torneo.

La estrategia de selección consiste en elegir un par de candidatos de forma aleatoria y luego elegir el candidato más apto con 50% probabilidad.

### D. Operadores de Cruzamiento

Se propuso la utilización del siguiente operador de cruzamiento:

- Operador de cruzamiento de 3 puntos elegidos en forma randómica conservándose las características de las soluciones del problema luego del cruzamiento.

### E. Operadores de Mutación

Los operadores de mutación fácilmente podrían introducir características ajenas a las propuestas para las soluciones del problema. Es por ello que se propuso las siguiente mutación particular al problema en cuestión:

- Reemplazar el 5% de las duraciones del genotipo elegidas en forma aleatoria por valores en el intervalo $I$ correspondiente al intervalo de duraciones.

### F. Técnicas Avanzadas

*1) Escalado del Fitness:* se utilizó escalado Sigma, un mecanismo avanzado de la función de fitness de modo de moderar la presión de selección en el tiempo tal que no sea muy alta en generaciones tempranas o muy baja o débil una vez que la población se ha estabilizado y las diferencias de fitness entre los candidatos son menores. Se utiliza la desviación estandar del fitness de la población para escalar los valores de fitness y mantener una presión relativamente constante durante la vida del algoritmo evolutivo. Se busca de esta forma mantener la diversidad en la población para evitar problemas de convergencia prematura.

*2) Algoritmos Evolutivos Paralelos:* se adopto además un modelo *maestro-esclavo* distribuyendo la evaluación de la función de fitness de modo de mejorar la eficiencia computacional del algoritmo y beneficiarse de la arquitectura multi-núcleo del Cluster Fing. El modelo *maestro-esclavo* se organiza en una estructura jerárquica donde un proceso maestro guía la búsqueda mientras controla un grupo de de procesos esclavos que realizan las evaluaciones de fitness necesarias de distintas soluciones candidatas en forma paralela. El número de procesos esclavos es igual al número de núcleos disponibles en nodo de ejecución. En este trabajo siempre se solicitaron nodos de 24 núcleos.

### G. Función de Fitness

Para evaluar cada posible solución $x$ se considera la siguiente función de fitness:

$$f(x) = \frac{C(x)}{R(x)}$$

$$C(x) = \alpha \sum_e \|{CO_2}^{x,e}\|_{l(e),Ts} + \beta \sum_e \|CO^{x,e}\|_{l(e),Ts}$$

$$+ \gamma \sum_e \|HC^{x,e}\|_{l(e),Ts} + \delta \sum_e \|{NO_x}^{x,e}\|_{l(e),Ts}$$

$$R(x) = \varepsilon v(x)$$

Se divide la suma total de las emisiones de $CO_2, CO, HC$ y $NO_x$ por *edge* o calle del escenario normalizadas entre el largo de la calle y el tiempo de simulación (g/km/h), entre la cantidad de vehículos que llega a destino durante el tiempo de simulación ($Ts$). Se fijaron los pesos $\alpha = 1/100, \beta = 1, \gamma = 10, \delta = 100, \varepsilon = 2$ a priori de forma tal de dar consideración uniforme a las emisiones siendo el objetivo principal minimizar los niveles de emisiones en los escenarios estudiados sin descuidar las condiciones del flujo vehicular.

Todos los datos requeridos para computar $f(x)$ son devueltos por SUMO una vez completada la simulación que tiene como entrada el genotipo $x$ decodificado.

## IV. Análisis Experimental

A continuación se presentan los detalles sobre la plataforma computacional utilizada, los escenarios seleccionados para la realización de los experimentos, la configuración de parámetros y por último se analizan los resultados numéricos del algoritmo evolutivo propuesto y comparaciones.

Para la comparación de resultados entre algoritmos se realizaron pruebas de hipótesis de forma de determinar si los resultados obtenidos por un algoritmo son mejores o peores que los del otro.

### A. Desarrollo y Plataforma de Ejecución

El Algoritmo Evolutivo fue desarrollado en Java 1.6 JDK utilizando la librería de desarrollo Watchmaker Framework 0.7.1. La fase de simulación se realizó utilizando el simulador

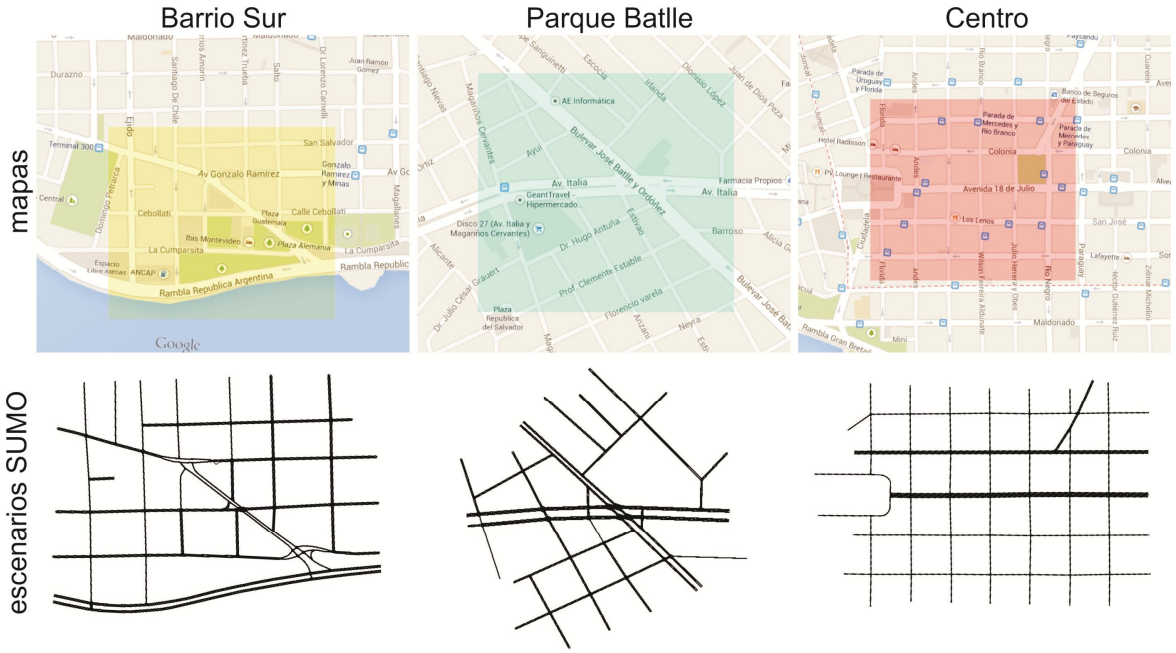


**Fig. 1. Escenarios de instancias de evaluación.**

SUMO en su versión 0.21.0 para el sistema operativo CentOS 6. El análisis experimental fue ejecutado en el Cluster Fing, infraestructura de cómputo de alto desempeño de la Universidad de la República, en nodos de 24 núcleos HP Proliant DL385 G7 y procesador AMD Opteron 6172 2.10 GHz, 24GB RAM y CentOS Linux 5.2. Todas las ejecuciones han sido realizadas solicitando los mismos recursos.

### *B. Instancias de Evaluación*

Para la realización de este trabajo se eligieron 3 grandes zonas de la ciudad de Montevideo con distintivas características de tramado y tráfico vehicular. Los escenarios comprenden parte de los barrios Centro, Parque Batlle y Barrio Sur respectivamente. Se utilizaron las reglas del tráfico regulado, señales, sentidos, carriles, etc. procurando asemejarse a la realidad en los aspectos más relevantes de acuerdo al proyecto.

Tenemos entonces los siguientes escenarios:

- Barrio Sur: escenario de trama compleja compuesto por muchas intersecciones. En total son 44 intersecciones de las cuales solo 13 están semaforizadas.
- Parque Batlle: escenario simple compuesto principalmente por dos grandes avenidas que se intersecan. Se tienen 19 intersecciones semaforizadas y 13 sin semaforizar.
- Centro: se trata del escenario más grande y complejo compuesto de avenidas y calles de distinta longitud, distinto número carriles y con parte de una rotonda. Posee 40 intersecciones de las cuales 34 están semaforizadas.

Cada simulación de SUMO tuvo un tiempo de estudio de 5 minutos, tiempo determinado a priori suficiente para que un vehículo pueda ir desde cualquier intersección del mapa a cualquier otra alcanzable a una velocidad promedio de 40 $km/h$. Además, para cada escenario se consideraron tres casos: durante el tiempo de simulación los vehículos ingresan al escenario a intervalos regulares de 1 segundos, 0.7 segundos ó 0.4 segundos, dando lugar a un total de 300, 430 y 750 vehículos aproximadamente. Se generaron rutas aleatorias para cada vehículos con una distancia mínima de 150 metros, un vez que el vehículo llega a destino no vuelve a aparecer durante el tiempo de estudio. En total, se utilizaron 9 instancias para evaluar el algoritmo desarrollado.

### *C. Configuración de Parámetros*

Para la configuración de parámetros se realizo un análisis experimental sobre instancias específicas o instancias de configuración, distintas a las instancias mencionadas

**Pruebas de normalidad**

| | | Shapiro-Wilk | | |
|---|---|---|---|---|
| Poblacion | | Estadístico | gl | Sig. |
| Fitness | IC1_50 | ,973 | 20 | ,820 |
| | IC1_75 | ,940 | 20 | ,241 |
| | IC1_100 | ,979 | 20 | ,916 |

**Tabla 1. Análisis de normalidad para configuración de población IC1.**

**ANOVA de un factor**

Fitness

| | Suma de cuadrados | gl | Media cuadrática | F | Sig. |
|---|---|---|---|---|---|
| Inter-grupos | 438,542 | 2 | 219,271 | 4,589 | ,014 |
| Intra-grupos | 2723,571 | 57 | 47,782 | | |
| Total | 3162,113 | 59 | | | |

**Tabla 2. Prueba ANOVA para configuración de población IC1.**

anteriormente de modo de evitar sesgos en la parametrización. Se utilizaron las siguientes instancias:

- Instancia de Configuración 1: se trata de un escenario con tramado simple que cuenta con una avenida principal semaforizada. Se tienen en total 24 intersecciones de las cuales solo 12 forman parte de la avenida principal y por tanto están semaforizadas. La densidad del tráfico es de 300 vehículos.
- Instancia de Configuración 2: mismo escenario que en la instancia 1 con otras 6 intersecciones semaforizadas y con densidad de tráfico de 600 vehículos.

Los parámetros estudiados fueron el tamaño de la población, la probabilidad de cruzamiento y la probabilidad de mutación. Esto fue realizado en dos etapas:

- Etapa 1: se fijo la probabilidad de cruzamiento y la probabilidad de mutación y se estudio el tamaño de la población.
- Etapa 2: una vez determinado el tamaño de la población se procedió a estudiar probabilidad de cruzamiento y probabilidad de mutación.

*1) Ajuste del Tamaño de la Población*

Para el estudio del tamaño de la población de genotipos se fijo la probabilidad de cruzamiento en 0.8 y la probabilidad de mutación en 0.075. Luego para cada una de las instancias de configuración se realizaron 20 ejecuciones independientes del algoritmo evolutivo utilizando diferentes valores de la semilla del generador de números aleatorios y variando el tamaño de la población, $t_p \in [50,75,100]$ . De este modo, para la instancia de configuración 1 se obtuvieron 20 resultados (mejor fitness) para cada tamaño de la población (muestras IC1_50, IC1_75, IC1_100).

En primer lugar, a las muestras IC1_50, IC1_75 y IC1_100 se les realizó una prueba de normalidad Shapiro-Wilk por tratarse de muestras pequeñas (n<50). Se comprobó que las 3 muestras siguen una distribución normal. Ver Tabla 1.

Luego se procedió a comparar las muestras para saber si existen diferencias significativas entre sus medias. Por tratarse de más de dos muestras se decidió utilizar el análisis de varianza ANOVA de un factor. Para ello fue necesario realizar una prueba de Levene de homogeneidad de varianzas que determino que no existen diferencias significativas entre las varianzas de las muestras. La prueba ANOVA si encontró diferencias significativas entre los promedios de las muestras. Ver Tabla 2. Es por ello que se procedió a realizar una prueba post hoc HSD de Tukey de forma de conocer las diferencias entre pares de muestras. Esta prueba determino que existe diferencia significativa entre los promedios de las muestras IC1_50 e IC1_100, sin embargo solo se observó una diferencia en la media de 6,6 mientras que la diferencia de medias para el tiempo de ejecución asciende a 1441,0 segundos.

Los pasos anteriores fueron repetidos para la instancia de configuración 2. Primero se comprobó que las muestras IC2_50, IC2_75 e IC2_100 fueran normales, luego se realizó la prueba de Levene y por último la prueba de ANOVA que esta vez encontró diferencias entre los promedios. Ver Tabla 3.

Dado que en la IC1 el incremento en el fitness era minúsculo al duplicar el tamaño de la muestra y dado que este incremento

**ANOVA de un factor**

Fitness

| | Suma de cuadrados | gl | Media cuadrática | F | Sig. |
|---|---|---|---|---|---|
| Inter-grupos | 198,123 | 2 | 99,062 | ,853 | ,431 |
| Intra-grupos | 6617,986 | 57 | 116,105 | | |
| Total | 6816,110 | 59 | | | |

**Tabla 3. Prueba ANOVA para configuración de población IC2.**

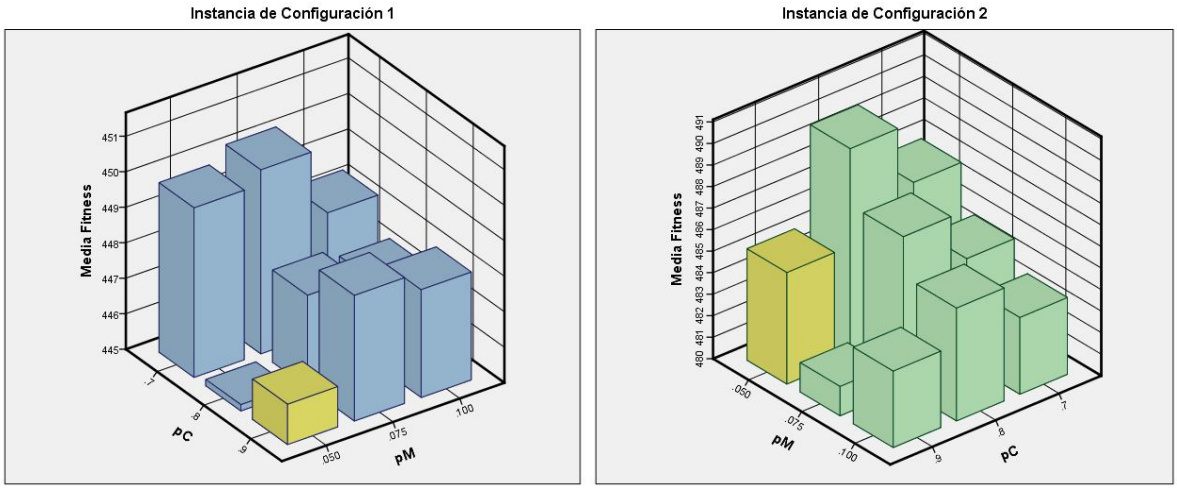


**Fig. 2. Comparación de Media de Fitness para configuración de probabilidades.**

conllevaba un incremento desproporcionado del tiempo de ejecución, las diferencias significativas estadísticas dadas por las pruebas no fueron tomadas en cuenta. Al no encontrarse diferencia en la IC2, finalmente se decidió utilizar una población de 50 individuos.

*2) Ajuste de Probabilidades de Cruzamiento y Mutación*

Para el ajuste de probabilidades de cruzamiento y mutación se utilizó una población de 50 como se determinó en la etapa anterior. Para cada instancia se estudiaron todos las combinaciones posibles entre $p_m \in [0.05, 0.075, 0.1]$ y $p_c \in [0.7, 0.8, 0.9]$, obteniéndose para cada instancia un conjunto de 9 muestras de 20 resultados cada una.

Para ambos conjunto se comprobó la normalidad de las muestras, luego se realizó la prueba de homogeneidad de varianza y por último la prueba de ANOVA de un factor. En ninguno de los dos casos se encontraron diferencias significativas entre los promedios.

Se determinó utilizar la configuración ($p_c = 0.9, p_m = 0.05$) de buen rendimiento (menor media de fitness) tomando en cuenta ambas instancias. Ver Figura 1.

*D. Algoritmos Adicionales*

Con el objetivo de evaluar la calidad de las soluciones obtenidas por el algoritmo evolutivo desarrollado se han implementado además otros 3 algoritmos:

- CPH: se trata de un algoritmo que utiliza una aproximación humana o razonable. Este algoritmo genera 200 candidatos a solución dando prioridad a las avenidas o calles de varios carriles. Se asigna una duración aleatoria $d$ en el intervalo $[x - 10s, x, x + 10s]$ a los estados, donde $x$ es 50 segundos si el estado se corresponde con una luz verde de una calle principal y 20 segundos para todos los demás. Se evalúan todos los candidatos utilizando la función de fitness y retorna el mejor. Los candidatos son evaluados en forma paralela utilizando una cantidad de hilos igual a los núcleos del nodo solicitado.
- CPHD: este algoritmo propone el uso de una medida frecuentemente aplicada descongestionar calles. Esta es, aplicar la luz verde por dos o tres minutos cuando se detectan cantidades extraordinarias de vehículos en una esquina, a través de cámaras inteligentes. El algoritmo implementado toma la salida de CPH y aplica en forma individual la medida de descongestionamiento en todos los estados de luces. Esto da lugar a una cantidad de individuos igual a la cantidad de estados de luces que son evaluados. Se retorna el mejor.
- CPSA: se trata de un algoritmo de *enfriamiento simulado* o *simulated annealing*. El candidato inicial es generado de la misma manera que en el algoritmo evolutivo asignando un valor aleatorio en el intervalo $[5,60] \in \mathrm{N}$ para la duración en segundos de los estados de los programas. El vecindario $N(s)$ de la solución temporal $s$ es una variación de 5 segundos en un estado elegido aleatoriamente, siempre manteniéndose en el intervalo $[5,60]$. Primero se chequea si la solución vecina es mejor utilizando la función de fitness. Si lo es, la aceptamos incondicionalmente. Si es peor, miramos dos cosas: que tan mala es y en que iteración estamos. En las iteraciones iniciales es más probable aceptar

soluciones que son peores. La tomamos con probabilidad:

$$\exp\left(\frac{s.fitnes() - s^*.fitness()}{T}\right)$$

donde $T$ va de 200 a 1 teniendo de este modo una condición de terminación de 200 iteraciones.

Los tres algoritmos adicionales toman una duración de 5 segundos para los estados de luz amarilla y un valor de 0s para el offset de los programas de ciclo.

### *E. Resultados y Comparaciones*

El algoritmo evolutivo AE junto con los otros tres algoritmos adicionales CPH, CPHD y CPSA fueron ejecutados 30 veces cada uno (ejecuciones independientes variando la semilla del generador de números aleatorios) para cada una de las 9 instancias de evaluación mencionadas anteriormente, obteniendo 4 muestras de 30 resultados (mejor fitness) para cada una de las instancias junto al tiempo de ejecución. Todos los datos fueron luego analizados. Los resultados se reportan en la Tabla 4. Para cada par instancia, algoritmo se presenta el mejor fitness entre los 30, el promedio de los mejores fitness y la desviación estándar. Para los tiempos de ejecución se reporta el promedio y la desviación estándar. Por último se reporta el p-valor del test de Shapiro-Wilk realizado a los mejores fitness.

Como se puede ver en la tabla señalada, el algoritmo evolutivo encontró la solución óptima en todos los escenarios para las distintas cantidades de vehículos. Para saber si existen diferencias significativas entre sus medias se procedió a realizar pruebas estadísticas. En las instancias IR1, IR2, IR5 e IR9, algunas de las muestras no pasaron el test de normalidad de Shapiro-Wilk. En los casos restantes no se pudo comprobar la homogeneidad de varianzas utilizando la prueba de Levene, por lo que para las 9 instancias se realizó la prueba no paramétrica de Kruskal-Wallis. En todos los casos se encontraron diferencias estadísticamente significativas obteniéndose siempre un p-valor de 0,000. En la figura 3 se observa una gráfica de cajas del fitness y el tiempo para la instancia 1 donde es evidente el buen desempeño del algoritmo evolutivo frente a los demás en términos de soluciones óptimas, sin embargo su eficiencia computacional está muy lejos de ser la óptima a pesar de tener una aproximación enfocada en la mejora de la performance computacional utilizando el modelo-esclavo. Cabe señalar que los algoritmos CPH y CPHD evalúan su población inicial en forma paralela al igual que lo hace el AE, no así el CPSA, que se ejecuta en forma serial.

### *F. Análisis Particulares - Emisiones*

En esta sección analizaremos los resultados del algoritmo evolutivo para una caso representativo de ejecución.

En la figura 4 se observa la evolución de la función de fitness para el algoritmo evolutivo junto a la evolución de los distintos contaminantes analizados $CO_2, CO, HC$ y $NO_x$ en una ejecución de la instancia 1. Vemos además como incluso se

| | Escenario Barrio Sur | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 300 Vehículos - IR1 | | | | | | 430 Vehículos - IR2 | | | | | | 750 Vehículos IR3 | | | | | |
| Alg. | $f_{mejor}$ | $\bar{f}$ | $f_\sigma$ | SW pValor | $\bar{t}$ | $t_\sigma$ | $f_{mejor}$ | $\bar{f}$ | $f_\sigma$ | SW pValor | $\bar{t}$ | $t_\sigma$ | $f_{mejor}$ | $\bar{f}$ | $f_\sigma$ | SW pValor | $\bar{t}$ | $t_\sigma$ |
| AE | **1297** | 1380 | 65 | *0,041* | 365,0 | 10,0 | **1274** | 1375 | 62 | 0,646 | 307,4 | 21,1 | **1296** | 1423 | 82 | 0,267 | 448,9 | 13,7 |
| CPH | 1935 | 2066 | 59 | 0,835 | 3,8 | 0,4 | 1990 | 2086 | 59 | *0,034* | 4,7 | 0,7 | 2059 | 2147 | 56 | 0,106 | 7,1 | 0,3 |
| CPHD | 1549 | 1729 | 137 | *0,000* | 6,1 | 1,1 | 1393 | 1625 | 105 | 0,970 | 6,5 | 0,8 | 1508 | 1717 | 118 | 0,215 | 9,7 | 0,5 |
| CPSA | 1592 | 1910 | 163 | 0,609 | 61,0 | 0,8 | 1670 | 1922 | 150 | 0,654 | 84,4 | 1,6 | 1717 | 2123 | 233 | 0,678 | 137,2 | 2,0 |
| | Escenario Parque Batlle | | | | | | | | | | | | | | | | | |
| | 300 Vehículos - IR4 | | | | | | 430 Vehículos - IR5 | | | | | | 750 Vehículos IR6 | | | | | |
| Alg. | $f_{mejor}$ | $\bar{f}$ | $f_\sigma$ | SW pValor | $\bar{t}$ | $t_\sigma$ | $f_{mejor}$ | $\bar{f}$ | $f_\sigma$ | SW pValor | $\bar{t}$ | $t_\sigma$ | $f_{mejor}$ | $\bar{f}$ | $f_\sigma$ | SW pValor | $\bar{t}$ | $t_\sigma$ |
| AE | **1054** | 1191 | 71 | 0,544 | 384,0 | 21,9 | **1066** | 1238 | 75 | 0,934 | 445,6 | 153,1 | **1123** | 1305 | 88 | 0,422 | 501,3 | 23,7 |
| CPH | 1574 | 1707 | 59 | 0,555 | 5,1 | 1,3 | 1544 | 1709 | 70 | 0,492 | 5,6 | 1,1 | 1685 | 1887 | 72 | 0,803 | 8,2 | 0,4 |
| CPHD | 1304 | 1569 | 90 | 0,071 | 7,0 | 1,3 | 1399 | 1610 | 89 | *0,016* | 7,6 | 0,5 | 1684 | 1794 | 78 | 0,072 | 11,4 | 0,6 |
| CPSA | 1707 | 1977 | 193 | **0,034** | 74,4 | 1,3 | 1540 | 1956 | 206 | 0,952 | 95,9 | 1,0 | 1695 | 2172 | 231 | 0,991 | 149,1 | 1,8 |
| | Escenario Centro | | | | | | | | | | | | | | | | | |
| | 300 Vehículos - IR7 | | | | | | 430 Vehículos - IR8 | | | | | | 750 Vehículos IR9 | | | | | |
| Alg. | $f_{mejor}$ | $\bar{f}$ | $f_\sigma$ | SW pValor | $\bar{t}$ | $t_\sigma$ | $f_{mejor}$ | $\bar{f}$ | $f_\sigma$ | SW pValor | $\bar{t}$ | $t_\sigma$ | $f_{mejor}$ | $\bar{f}$ | $f_\sigma$ | SW pValor | $\bar{t}$ | $t_\sigma$ |
| AE | **337** | 388 | 10 | 0,415 | 497,1 | 48,4 | **399** | 419 | 12 | 0,610 | 481,1 | 21,9 | **444** | 477 | 17 | 0,850 | 413,9 | 19,1 |
| CPH | 443 | 457 | 6 | 0,485 | 4,9 | 1,3 | 471 | 487 | 7 | 0,383 | 5,1 | 1,0 | 563 | 581 | 8 | 0,840 | 6,0 | 0,0 |
| CPHD | 426 | 440 | 9 | 0,127 | 7,8 | 1,8 | 457 | 474 | 10 | 0,234 | 7,5 | 1,3 | 536 | 559 | 13 | 0,401 | 9,6 | 0,5 |
| CPSA | 461 | 525 | 28 | 0,931 | 59,4 | 0,7 | 491 | 593 | 29 | 0,670 | 80,1 | 0,9 | 591 | 656 | 45 | *0,046* | 124,5 | 1,6 |

**Tabla 4. Resultados experimentales para los algoritmos AE, CPH, CPHD y CPSA**

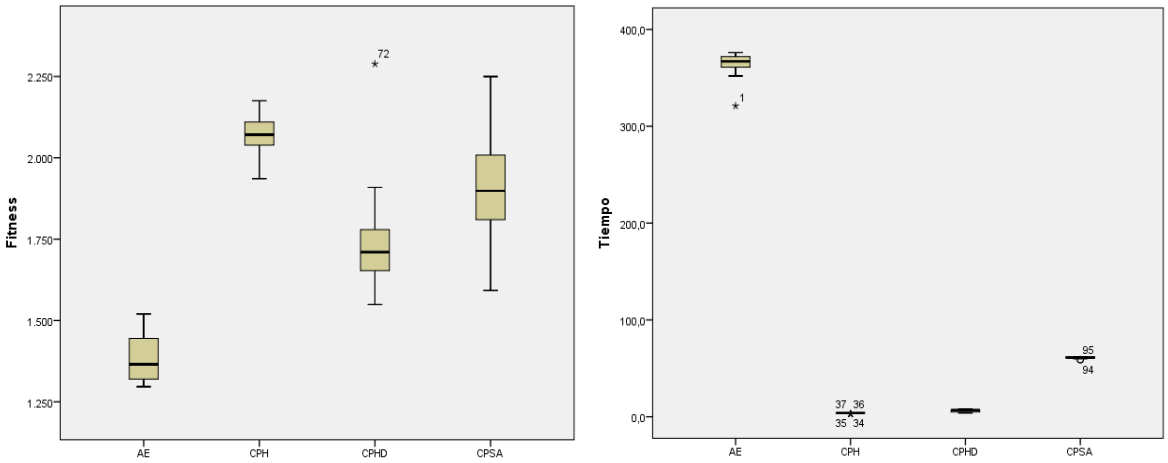


**Fig. 3. Gráfico de cajas para las muestras de fitness y tiempo de la instancia 1**

incrementa la cantidad de vehículos que llegan a destino, mejorando las condiciones del tráfico. Este sin embargo no es el caso para todas las ejecuciones. También es posible observar casos donde el algoritmo opta por aumentar la cantidad de vehículos únicamente mientras que las emisiones se mantienen constantes o en algunos casos incluso aumentan. Esto es debido a la naturaleza mono-objetivo del algoritmo que se satura al tener que considerar tantos objetivos dispuestos en la función de fitness.

## V. Conclusiones y Trabajo Futuro

Este artículo presento una aproximación al problema de programación de ciclos de semáforos para reducción de emisiones vehiculares utilizando un algoritmo evolutivo paralelo.

El algoritmo propuesto fue diseñado para proveer soluciones precisas al problema siguiendo casos de estudio realistas de la ciudad de Montevideo, ciudad con un alto porcentaje de semáforos en comparación con otros países de América Latina. Se propuso la reducción de las emisiones vehiculares más importantes como ser $CO_2, CO, HC$ y $NO_x$ y al mismo tiempo mejorar las condiciones del tráfico. Las instancias de evaluación fueron basadas en zonas reales de la ciudad de Montevideo ubicadas en los barrios Centro, Parque Batlle y Barrio Sur variando la cantidad de vehículos entre 300 y 750. Se adoptó además un modelo maestro-esclavo de modo de aumentar la eficiencia computacional.

En comparación con otros algoritmos que aplican una aproximación humana o razonable e incluso una metaheurística frecuentemente utilizada, el algoritmo evolutivo consiguió las mejores soluciones siempre. Analizando un caso en particular se observó la disminución en todos las emisiones estudiadas e incluso un gran incremento en la cantidad de vehículos que llega a destino, requiriendo sin embargo un tiempo de ejecución mucho mayor al utilizado por los algoritmos adicionales.

Como trabajo futuro sería conveniente incorporar a las simulaciones más datos reales de las zonas elegidas, como ser el flujo de vehículos por calle y distinciones entre tipos de vehículos. Además sería conveniente adoptar una implementación multi-objetivo dado que no siempre el algoritmo es capaz de disminuir las emisiones y la cantidad de vehículos dado que es superado por la cantidad de objetivos dispuestos en la función de fitness.

## References


[1] Ernesto Martínez Ataz, Yolanda Díaz de Mera Morales. Contaminación atmosférica. España: Ediciones de la Universidad de Castilla-La Mancha, 2004, pp. 11.

[2] F. Costabile, I. Allegrini. A new approach to link transport emissions and air quality: An intelligent transport system based on the control of traffic air pollution. Environmental Modelling & Software 2008;23: pp. 258-267.

[3] Keoleian, G.A., Kar, K., Manion, M.M. and Bulkely, J.W., 1997. Industrial ecology of the automobile: a life cycle perspective. Society of Automobile Engineers Inc., Warrendale, USA. pp. 65,128.

[4] Geo Montevideo Informe Ambiental 2004. Programa de las Naciones Unidas para el Medio Ambiente. pp. 76.

[5] Sanchez-Bertucci L., Darre E., Reyes B.A., Gogichaishvili A., Morales J., Bautista F. Estudio magnético en líquenes de la ciudad de Montevideo. Latinmag Letters, Volume 3, Special Issue (2013), OD07, 1-7. Proceedings Montevideo, Uruguay.

[6] Robertson, D. I., C. F. Lucas, and R. T. Baker. Coordinating Traffic Signals to Reduce Fuel Consumption. TRL Report LR934. Transport Research Laboratory, Crowthorne, Berkshire, United Kingdom, 1980.

[7] Aleksandar Stevanovic, Jelka Stevanovic, Kai Zhang, and Stuart Batterman. Optimizing Traffic Control to Reduce Fuel Consumption and Vehicular Emissions. Transportation Research Record: Journal of the Transportation Research Board, No. 2128, Transportation Research Board of the National Academies, Washington, D.C., 2009, pp. 105–113.

[8] Algoritmos Evolutivos y Algoritmos Genéticos. Alfonso Mateos Andaluz - Universidad Carlos III de Madrid. Inteligencia en Redes de Comunicaciones.

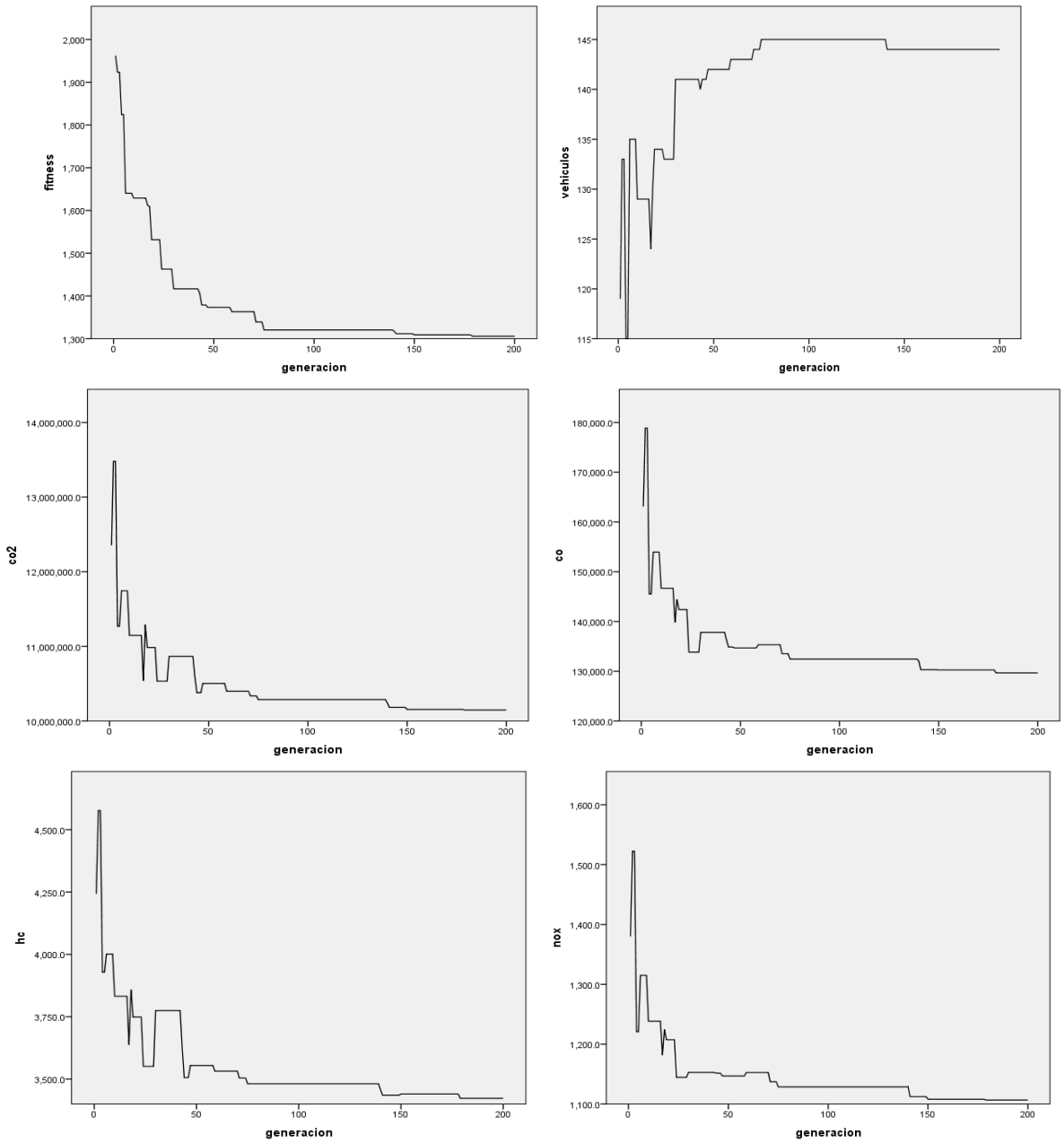


**Fig. 4. Evolución de fitness, vehículos y contaminantes para una ejecución particular de I1**

[9] Algoritmos Genéticos - Abdelmalik Moujahid, Iñaki Inza y Pedro Larrañaga - Departamento de Ciencias de la Computación e Inteligencia Artificial - Universidad del País Vasco–Euskal Herriko Unibertsitatea. pp. 1-4.

[10] Evolución en el diseño y clasificación de Algoritmos Genéticos Paralelos - Sergio Nesmachnow -Universidad de la República, Instituto de Computación, pp. 2.

[11] A Review of Traffic Simulation Software - G. Kotushevski and K. A. Hawick, Computer Science, Institute of Information and Mathematical Sciences, 2009.

[12] M. Keller. Handbook of Emission Factors for Road Transport (HBEFA) 3.1. quick reference. Technical report, INFRAS, 2010.