Simulador de tráfico microscópico utilizando HPC

Presentación final proyecto Computación de alta performance

1st Steven Estramil
Universidad de la Republica
Facultad de Ingenieria
Montevideo,Uruguay
email: edison.estramil@fing.edu.uy

2nd Daniel Padrón
Universidad de la Republica
Facultad de Ingenieria
Montevideo, Uruguay
email: daniel.padron@fing.edu.uy

***Resumen*—En el presente documento se detallará la construcción de un simulador de tráfico microscopio, capaz de representar el tráfico en diferentes ciudades y número de vehículos. A diferencia de un simulador macroscópico, que representa al tráfico como un flujo, un simulador microscopio considera el estado y comportamiento de cada uno de los vehículos en tránsito. Debido elevado costo computacional, el simulador utiliza técnicas de programación paralela vía tecnologías como OpenMP (Open Multi-Processing) y MPI (Message Passing Interface).**

Palabras claves: Simulador de tráfico microscópico, MPI, OpenMP, Openstreetmap

I. Introducción

En las ciudades modernas, la gestión eficiente del tráfico es fundamental para mantener la movilidad y la calidad de vida de los ciudadanos. Sin embargo, a medida que crece la población y aumenta el número de vehículos en las carreteras, también lo hace la complejidad de mantener tiempos de viaje predecibles y razonables para los conductores. Este problema se agrava con el paso del tiempo, ya que la demanda de movilidad urbana continua en aumento. El tiempo medio de viaje - considerando una distancia fija de 10 km - es un indicador crucial para evaluar la infraestructura vial ante la creciente población de una ciudad o región. Según el ranking elaborado por TomTom [1], Londres, Inglaterra, encabeza la lista con el mayor tiempo promedio de viaje en 10 km, alcanzando los 37 minutos. En contraste, ciudades como Osaka, Japón, presentan un tiempo promedio considerablemente menor, alrededor de 21 minutos.

El cálculo del tiempo medio de viaje es realizado por medios experimentales.

Se propone como problema a resolver la creación de un simulador, el cual dado un número de vehículos y una estructura vial - determinada por los usuarios - sean capaz de registrar el tiempo y distancia que recorrió cada vehículo, estimando así el tiempo medio de viaje. Permitiendo el cálculo del anterior indicador en ciudades o regiones no existentes o modificadas, o con un número de vehículos distinto al de la actualidad.

II. Descripción del problema

A continuación se especificará el problema que se propone resolver, en particular se trata de un simulador de tráfico de una estructura vial y un número de vehículos personalizables. Existen diferentes tipos de representación del tráfico , principalmente son dos las variantes [2]: Macroscópicos y Microscopios. Los simuladores con representación Microscopia simula a cada uno de los vehículos, administrando su posición, velocidad, y demás propiedades. Por otro lado, los simuladores Macroscópicos representan a los vehículos como un flujo, ignorando los detalles particulares de cada vehículo y calle que circula. Actualmente, existen múltiples simuladores de ambos tipos, un ejemplo que utiliza representación microscopia es SUMO [3]. De todas formas, cuando se busca simulaciones a gran escala y cuya simulación sea en tiempo real, la representación Macroscopía es la indicada.

Dado que el proyecto es desarrollado en el contexto del curso de Computación de Alta Performance, se propone la resolución de un simulador que utilice una representación Microscopia, ya que es más demandante computacionalmente, y como se explicara en la sección V, utilizar técnicas de programación paralela para reducir el tiempo de cómputo.

De todas formas, también debido al contexto en que se desarrolla el proyecto, el problema a resolver se enfocara en los conceptos propios de HPC y en consecuencia se modelará de forma muy simplificada el comportamiento de los vehículos. Existen extensas investigaciones [4] que intentan describir el comportamiento de los conductores, aspectos como la velocidad segura, retrasos por giros, condiciones para el cambio de ruta planificada, cuando se produce un cambio de carriles, entre otros.

La información de la ciudad requerida será la siguiente: las calles y sus conexiones deberán ser representadas como un grafo; de cada calle, su largo, número de carriles y velocidad máxima; de cada cruce entre las calles, su ubicación geográfica y al barrio al que pertenece. Finalmente, se debe especificar el porcentaje de los vehículos generados en cada barrio, y para uno de ellos, qué porcentaje de sus vehículos tendrán como destino un barrio determinado. Por ejemplo, dado el barrio Cuidad Vieja, qué porcentaje de los vehículos generados ahí tendrán como destino cada uno de los barrios. La obtención de dichos datos será explica en la sección IV.

A diferencia de otros simuladores, los cuales generan constantemente vehículos, el simulador a implementar introduce los vehículos al tráfico lo más rápido posible (según la disponi-

bilidad de las calles), permitiendo así estudiar la tolerancia del sistema vial ante un veloz incremento en el número vehículos, similar al dado durante las conocidas ”horas picos”.

Además, se tiene como objetivo el cálculo de tiempo medio de viaje (ajustado a 10Km), el mismo deberá ser calculado como el promedio de los tiempos de viaje de cada uno de los vehículos una vez que todos ellos hayan alcanzado su destino.

El camino que un vehículo recorrerá desde su origen hasta su destino deberá ser generado para cada uno de los vehículos considerando los datos estructura vial. La intención es intentar emular que un conductor previo a iniciar su recorrido calculó la ruta que debería recorrer para minimizar su tiempo de viaje, y dado que cada vehículo hizo lo mismo, ver el efecto que esto tiene sobre el tránsito. De todas formas, notar que en la vida real, aplicaciones las cuales sugieran rutas de viaje (como Google Maps), tiene en cuenta variables como la congestión esperada de las calles en un rango de horas particular, lo que permitirá obtener tiempos medios de viaje menores.

Finalmente, debido a que el simulador calcula las rutas al inicio de la simulación, es posible que se generen ciclos de dependencias entre los vehículos, lo que provocaría que el tráfico quede indefinidamente detenido. Entonces se debe permitir que si un vehículo estuvo esperando un tiempo considerablemente elevado para poder ingresar a la siguiente calle que su ruta le indica, poder cambiar de ruta, buscando una que minimice los tiempos de viaje considerando la congestión y velocidad media de las calles al momento del cambio de ruta.

III. Justificación del uso de HPC

En las grandes ciudades del mundo se registran en el orden de cientos de miles de vehículos por ciudad, además de las miles de bifurcaciones de calles que existen, por ejemplo en el caso de Montevideo que es una ciudad bastante pequeña cuenta con alrededor de 50.000 bifurcaciones de calles, si suponemos que los vehículos van a una velocidad de 45km/h equivalentes a 12.5 m/s y en particular 1m cada 0.080s, una decisión natural sería tomar que cada época sea equivalente a 0.080s para que el avance de un vehículo sea a lo sumo de 1 metro por época, esto nos deja con un total de 10.000 épocas por vehículo para completar los 10km de recorrido en el caso ideal donde hace el recorrido sin parar. Esto es únicamente para un vehículo, además se pueden generar tantos vehículos como el usuario desee, donde cuantos más vehículos se generen más congestiones se van a dar y más ejecuciones van a ser necesarias para que cada auto llegue a su destino.

Dado que al aumentar la cantidad de vehículos y el tamaño de la ciudad computada, surge la necesidad de reducir el tiempo de cómputo de cada una de estas, dado que el avance de los vehículos en una calle es independiente de aquellas calles que no se bifurcan con la misma, podemos calcular el movimiento de los vehículos de una calle de manera separada y distribuir esta actividad de forma paralela.

Además, el problema crece en tamaño cuando el mapa se compone de un mayor número de calles, por lo cual el uso de HPC ayuda a disminuir los tiempos de simulación ante el aumento de vehículos (y por ende de congestión, lo que redunda en épocas) o en el tamaño del mapa.

IV. Preprocesamiento de los datos

Para la generación del archivo JSON necesario para construir el grafo correspondiente a la ciudad, se realizó el siguiente procedimiento:

1 - Obtención de información con OpenStreetMap (OSM): Para obtener la información de la estructura vial de la ciudad se hizo uso de la librería proporcionada por OSM, osmnx, para Python. Osmnx provee una función que, a partir del nombre de la ciudad que se pasa como entrada, genera una salida en formato JSON. Esta salida contiene nodos que representan las intersecciones entre calles. Estos nodos tienen un identificador, latitud y longitud de su ubicación. La salida también genera asociaciones entre nodos que representan las calles. Estas asociaciones incluyen identificadores de nodo inicio y destino, nombre de la calle, número de carriles, longitud del carril, velocidad máxima en la calle y si es doble vía.

2 - Asignación de barrios a los nodos: Como Osmnx no proporciona la información de a qué barrio pertenece cada nodo para Montevdieo, fue necesario hacer uso de la información provista por el catálogo de datos geográficos de Montevideo [6], que ofrece información sobre la delimitación de los barrios realizada en 2011. Con esta información y la latitud y longitud de cada nodo, se pudo calcular a qué barrio pertenece cada nodo.

Para otras ciudades, Osmnx proporcionaba las geometrías de los barrios, por lo que no fue necesario descargar la geometría de los barrios de una fuente externa.

V. Estrategia de resolución

En esta sección se desarrollará la estrategia de resolución abordada para el diseño e implementación del simulador. Como fue mencionado anteriormente, el objetivo es hacer uso de la programación paralela para acelerar los tiempos de cómputo, por lo tanto, en la primera subsección ”Paralelismo, elección de granularidad y cooperación” (V-A) se dará inicio a la resolución del problema discutiendo cuáles son las secciones del problema que conforman la parte paralela del mismo,cuál es la granularidad utilizada, y como los diferentes procesos cooperarán en la resolución del problema. En el resto de las subsecciones se desarrollará distintos aspectos necesarios a resolver para lograr el correcto funcionamiento del simulador. Finalmente, en la subsección ”Arquitectura y etapas de la simulación” (V-H) sé describirá la arquitectura del simulador y se sintetizará en un diagrama de secuencia cada una de las etapas del simulador con sus respectivas comunicaciones.

A. Paralelismo, elección de granularidad y cooperación

Un simulador de tráfico con representación microscopia representa el estado de cada uno de los vehículos en tráfico. Es importante tener una unidad de tiempo con el fin de poder sincronizar el estado de todos los vehículos. Llamaremos ”época” a una unidad de tiempo, la cual es equivalente a 100 milisegundos dentro del simulador. Como se profundizará

en la subsección ”Ejecución de una época” (V-D), debido a que estamos simulando un evento de la vida real, todos los vehículos debe existir en el mismo espacio temporal, y, por lo tanto, el número de épocas de los mismos no pueden diferir en más de una unidad.

Se identifican dos posibles estados en el que puede estar un vehículo: transitando dentro de una calle y esperando para ingresar a otra. En el primero caso, un vehículo debe estar sincronizado con el resto de los vehículos que circulan en la misma calle, ya que basándose en la posición de los demás determina su velocidad y por ende su nueva posición. Además, cuando se actualiza la posición de los vehículos de una calle (incrementa la época de los vehículos) es deseable que primero lo hagan los vehículos que se encuentran en la parte más próxima a la finalización de la calle, de forma tal que su actualización libera espacio para que el vehículo de detrás pueda avanzar. Mientras tanto, en el segundo caso, los vehículos que han alcanzó el final de la calle actual y desean desplazarse a la próxima calle de sus rutas, deberán esperar a que alguno de los carriles de la próxima calle esté disponible para efectuar el traspaso.

En consecuencia: el problema se puede paralelizar si se divide las calles a procesar en múltiples procesos, donde cada calle deberá incrementar la época de los vehículos que contiene, comenzando con los vehículos más próximos a la finalización de la calle.

Observar que la granularidad elegida es a nivel de calle, otra opción podría ser elegir una granularidad más fina, como por ejemplo a nivel de vehículo, pero consideramos que sería perjudicial dicha decisión. El motivo, ya ha sido mencionado con anterioridad: los vehículos dentro de una calle requiere estar sincronizados con el resto, ya que de ello depende la velocidad y posición futuras, además, que la actualización de los vehículos debe ser en un orden determinado. Todo lo anterior aumentaría los costos de sincronizaciones por las dependencias introducidas entre los vehículos.

Con el fin que los vehículos puedan completar su trayecto, los mismos deben avanzar dentro de cada calle y luego ser trasladado entre ellas, dicho traslado es de la forma en que las calles (y en consecuencia los procesos) cooperan. En particular, cada calle tendrá dos colas: una de solicitudes y otra de notificaciones, donde las solicitudes son mensajes enviados por la calle emisora a la receptora, solicitando espacio para uno de sus vehículos que han llegado al final de la calle actual. Mientras que las notificaciones son mensajes que las calles receptoras envían a las emisoras con el fin de informar que una solicitud en particular ha sido aceptada. Se profundizará en el mecanismo antes descrito en la sección ”Ejecución de una época” (V-D).

B. Estructura y División del mapa

Como fue explicado en la sección ”Descripción del problema” (II) solamente se conoce de la ciudad un grafo dirigido en el cual las aristas son las calles y los nodos son las intercepciones de las mismas. De cada nodo se conoce su latitud y longitud (cuyo uso se explicará en la siguiente subsección);

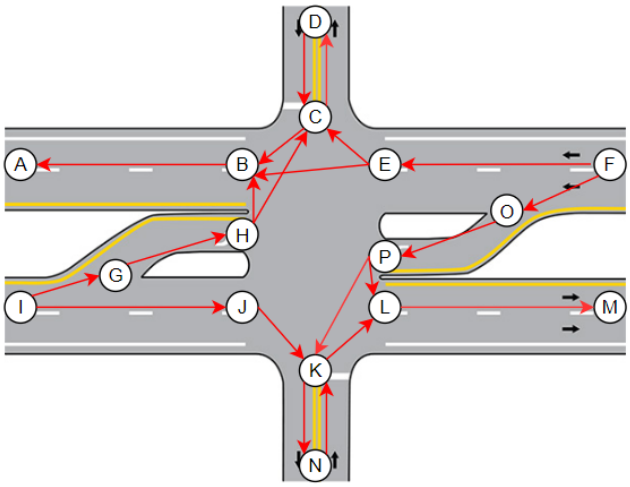

Fig. 1. Estructura de grafo de calles

mientras que de cada calle se conoce el nodo de inicio y destino (observar que no se permiten calles bidireccionales, en cuyo caso se requieren dos aristas), el número de carriles, la velocidad máxima a la que los vehículos pueden circular, y su largo.

En la figura 1 se presenta una ilustración de la estructura del grafo antes referido.

Además, se conocen los barrios que tiene la ciudad, y los nodos a los que pertenece cada uno.

Lo anterior permite dividir la ciudad en un conjunto de secciones, en particular diremos que una calle pertenece a un barrio, si su nodo inicial pertenece al barrio.

Con el fin de incrementar el número de procesos utilizados para la ejecución del simulador, se debe utilizar múltiples ordenadores, los cuales estarán comunicados vía MPI. Cada uno de estos ordenados serán llamados ”procesos MPI”. Y a cada uno de ellos se le delegará la responsabilidad del manejo de un barrio, y en consecuencia, de la ejecución de las calles que le perteneces. El motivo de utilizar barrios, y no asignar de forma aleatoria las calles a los procesos MPI, es con el objetivo de minimizar el volumen y tamaño de las comunicaciones producto del intercambio de vehículos entre calles de diferentes procesos MPI, procurando que la mayoría de dichas comunicaciones sean realizadas vía memoria compartida dentro de un mismo proceso MPI.

La asignación de un barrio a un proceso MPI es realizada mediante el procedimiento de balance de carga, detallado en la sección ”Balance de carga” (V-G).

C. Cálculo de caminos iniciales, distribución de segmentos y generación de vehículos

Como fue mencionado en la sección ”Descripción del Problema” (II) se cuenta por parte del usuario con el número de vehículos a generar, y qué porcentaje de los mismos serán generados en cada barrio. Además, el usuario también es responsable de suministrar una matriz, la cual contenga en

cada fila (que representa un barrio de origen) los porcentajes en que los vehículos generados en dicho barrio, tendrán como destino otro barrio, llamaremos matriz M a dicha matriz, definida formalmente como:

$$M_{i,j} = \text{Probailidad que un vehiculo generado en el barrio i tenga como destino el barrio j} \quad (1)$$

Como será detallado en la sección "Balance de carga" V-G al comienzo de la simulación el proceso MPI con rango 0 (a partir de este momento nos referiremos a el cómo "proceso máster") determina y comunica (según su estrategia de balance de carga) a qué proceso MPI estará asignado cada barrio y el número de vehículos que deben ser generados en cada uno de los mismos. Posteriormente, para cada barrio, el proceso MPI responsable sorteará de manera pseudoaleatoria un nodo dentro del barrio de origen, y utilizando la matriz M se escogerá el barrio de destino y seguidamente de forma pseudoaleatoria un nodo dentro de él.

El costo de las aristas del grafo (en otras palabras el costo de cada calle) que será utilizada en la búsqueda de la ruta está determinado de la siguiente manera:

$$\frac{\text{largo calle}}{\text{velocidad maxima} \times \sqrt{\text{numero carriles calle}}} \quad (2)$$

Notar que la ecuación (2) tiene como objetivo reflejar el menor tiempo que un vehículo permanecería en la calle, o el tiempo esperado si fuera a la máxima velocidad permitida en la misma. El factor $\sqrt{\text{numero carriles calle}}$ intenta priorizar que un vehículo recorra calles con un mayor número de carriles (como las avenidas y carreteras) y utilice las calles más estrechas de forma menos frecuente.

Se calculará el camino entre el nodo de origen y el nodo de destino utilizando el algoritmo A-star [9]. Dicho algoritmo es muy similar al algoritmo de los caminos más cortos de Dijkstra, pero con la diferencia que utiliza una heurística para estimar que tan distante es un nodo del grafo y el nodo objetivo. Basándose en el valor obtenido por dicha heurística, se prioriza explorar, en la búsqueda del camino más corto, aquellos nodos cuyo peso más el valor de la heurística sea el menor. El resultado es una disminución significativa en los costos computacionales en comparación a realizar Dijkstra, aunque no hay garantizas que el camino obtenido sea efectivamente el de menor peso. La heurística utiliza se presenta a continuación:

$$\frac{Haversine(nodo_1, nodo_2)}{80} \quad (3)$$

Donde "Haversine" es la fórmula de Haversine [10] utilizada para el cálculo en metros entre dos puntos cuya ubicación está expresada en longitud y longitud.

La generación de los vehículos (incluyendo el cálculo de su camino) dentro de un proceso MPI será realizada con los múltiples hilos disponibles en el nodo. Como el costo computacional del cálculo de un camino es variable (dependiendo de la distancia entre el nodo de origen y destino), entonces se emplea una asignación (scheduler) dinámica (*dynamic*). La asignación *guided* fue probada, pero no presento mejores resultados, posiblemente porque la diferencia de tiempo no es lo suficientemente dispar.

Una vez que un vehículo conoce el recorrido que debe realizar para llegar a su destino, se presenta una dificultad, ¿Qué debería ocurrir cuando un vehículo debe ser trasladado de una calle a otra, pero las dos calles son administradas por procesos MPI distintos?. Una posibilidad podría ser que en la solicitud de traspaso de un vehículo hacia una calle se encuentra incluida su ruta restante, pero en ciudades de un tamaño medio, dichas rutas pueden consistir de cientos de elementos, lo que haría que las comunicaciones entre procesos sean más lentas. La opción elegida consiste en dividir la ruta calculada en una serie de "segmentos", cada segmento corresponde al fragmento de la ruta que está incluida de forma continua dentro de un barrio. Por ejemplo, si la ruta de un vehículo atraviesa (en este orden) los barrios: Parque Batlle, Buceo, nuevamente Parque Batlle y finalmente Unión, entonces de dicha ruta se generaran 4 segmentos.

Cada uno de estos segmentos consiste de la siguiente información:

- Identificador de vehículo al que le corresponde la ruta segmentada,
- Identificador del barrio al que corresponde el segmento
- Si se trata del último segmento asociado a la ruta
- El nodo por el cual se ingresara al barrio
- El nodo por el cual se sale del barrio o, si trata del segmento final, el nodo donde finalizará el recorrido el vehículo.

Todos procesos MPI calculan dichos segmentos para cada uno de los vehículos que generen. Seguidamente, envían a cada uno del resto de los procesos MPI los segmentos que le correspondan según el barrio asociado al segmento y el proceso MPI responsable del barrio en cuestión.

El envío es realizado de forma no bloqueante (utilizando instrucción *MPI_Isend*), el tipo de dato enviado es un struct definido con MPI llamado *MPI_SegmentoTrayectoVehculoEnBarrio*. Luego de los envíos, cada nodo utilizando la instrucción *MPI_IProve* obtiene el número de segmentos a recibir por nada nodo, reserva la memoria requerida, y también de forma no bloqueante recibe los segmentos(Utilizando la instrucción *MPI_Irecv*). Finalmente, se utiliza la instrucción *MPI_Waitall* para esperar la finalización de las operaciones de envío y recepción. Los segmentos serán utilizados durante la ejecución de las épocas, la cual será descrita en detalle en la siguiente sección.

D. Ejecución de una época

En esta subsección se detallará el conjunto de pasos realizados para la ejecución de una época, como se mencionó en la sección "Paralelismo, elección de granularidad y cooperación" (V-A), el tiempo dentro de la simulación (época) en la que se encuentra cada vehículo no puede diferir en más de 1 época, lo que equivale a 10ms de simulación. Por lo tanto,

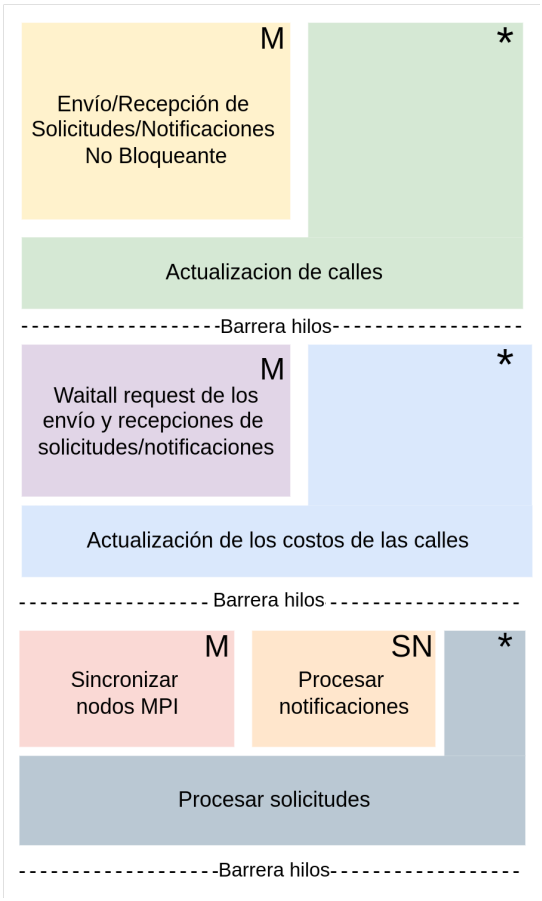

Fig. 2. Procesos de Ejecución de una época

la ”Ejecución en una época” consiste en actualizar el estado de cada vehículo activo en la simulación. El conjunto pasos a realizar para cumplir dicho objetivo se encuentra ilustrado en la figura 2. Cada bloque representa un paso a ser realizado, en aquellos bloques que se encuentran en la esquina izquierda superior una ”M”, significa que son únicamente ejecutados por el hilo máster; cuando es ”SN” significa que son ejecutados por cualquier thread y los demás no esperan su terminación (*single nowait*); mientras tanto cuando se encuentra un ”*” representa se divide la carga en los múltiples hilos que ingresen al bloque.

Procedemos a explicar el propósito y funcionamiento de cada uno de ellos a continuación, comenzando por **”Actualización de calles”**.

Utilizando todos los hilos disponibles en el proceso MPI, con una asignación dinámica (”chunk” de 150 elementos), se recorre la lista de calles de los barrios administradora por dicho proceso, y para cada calle un hilo será el encargado de actualizar la calle. La actualización de una calle consiste en la siguiente serie de pasos:

1) Eliminar los vehículos por los cuales se haya recibido una notificación.
2) Recorrer los vehículos que se encuentran en la calle en orden, primero lo que se encuentran más cerca de la terminación de la calle. Para cada uno de ellos actualizar su posición según la política de movimiento, en este caso, según la velocidad máxima permitida por la calle y la posición del vehículo que tiene en frente en del mismo carril.
3) Para aquellos vehículos que en el paso anterior alcanzaron el final de la calle, hay 2 posibles casos: Alcanzo el nodo de destino, resultando en la eliminación del vehículo, registrar el tiempo medio de viaje en 10 Km y descontar una unidad al contador de vehículos en curso que el proceso MPI posee; en el caso contrario debe ser enviada una solicitud a la próxima calle que intentará ingresar.
4) Para cada carril que tenga suficiente espacio para alojar un nuevo vehículo, el primer vehículo de la lista de solicitudes, donde se notificará a la calle emisora de la solicitud que el vehículo ha sido aceptado. Observar que en caso de que se tuviera información de los semáforos o un sistema de preferencias, es en este paso donde se aplicaría. Ya que basta que la calle que escoja la solicitud lo haga considerando dicho método de sincronización. De todas formas, en la implementación aquí presentada no se cuenta con tal sistema de prioridad o semáforos.

Es importante resaltar que cuando una calle debe enviar una solicitud a otra calle, existen una serie de casos:

- Si la calle receptora de la solicitud pertenece al mismo barrio que la emisora, se inserta directamente la nueva solicitud en la cola de solicitudes de la calle receptora. Tanto la cola de solicitudes como la de notificaciones de cada calle tiene su acceso mutuo excluido vía un mutex propio.
- Si la calle receptora pertenece a un barrio distinto al de la emisora, pero ambos barrios están bajo la responsabilidad del mismo nodo MPI, entonces: Se accede y elimina el segmento correspondiente al vehículo y barrio de la calle receptora, y se vuelve a calcular la ruta a recorrer, pero únicamente dentro del barrio, y entre el nodo de inicio y fin del segmento. Recordar que el segmento fue calculado durante la generación de camino del vehículo (descrito en la subsección ”Cálculo de caminos iniciales, distribución de segmentos y generación de vehículos” (V-C)). Como los barrios contiene un número reducido de nodos, el cálculo de la ruta del segmento es poco demandante. De todas formas, si una ruta entre dos nodos dentro de un barrio es calculada, la misma es almacenada, eliminando cálculos repetidos.
- Si el barrio de la calle emisora y receptora no es el mismo, y los dos barrios son administrados por procesos MPI diferentes, entonces la solicitud debe ser enviada al proceso MPI responsable de la siguiente calle. Todas las solicitudes que se encuentren en la misma condición serán almacenadas, y durante la ejecución de la próxima época enviada a sus correspondientes procesos MPI. Se describirá en detalle el proceso de intercambio de solicitudes y notificaciones en los dos próximos apartados del proceso de ”Ejecución de la época”.

El caso de las notificaciones es similar: Cuando una solicitud es aceptada, si la calle emisora de la solicitud está bajo la responsabilidad del mismo proceso MPI la notificación es directamente insertada en su cola de notificaciones, en caso contrario, será almacenada y enviada al proceso MPI responsable de la calle a notificar en la próxima época.

Dado un proceso MPI, se le llamará "procesos MPI vecinos" a los proceso MPI que tengan bajo su responsabilidad a algún barrio adyacente a algún otro barrio del proceso MPI en cuestión.

Al mismo tiempo que los hilos actualizan el estado de cada calle, el hilo master primero realiza el apartado de **Envío/recepción de Solicitudes/Notificaciones No bloqueantes**. Esto consiste primero en enviar de forma no bloqueante (utilizando *MPI_Isend*) a sus "procesos MPI vecinos" las solicitudes y notificaciones generadas durante la ejecución de la época pasada, y cuyo destino era una calle gestionada por otro proceso MPI. Seguido a ello se realizan las recepciones no bloqueantes de las solicitudes y notificaciones (utilizando *MPI_Irecv*) de sus "proceso MPI vecinos". Observar que el número de solicitudes y notificaciones que puede recibir un nodo está superiormente acotado por el número de calles entrantes a los barrios bajo su supervisión (multiplicando cada uno por el número de carriles), por lo tanto, los buffers de recepción tanto de notificaciones como de solicitudes son creados una vez que la asignación de los barrios a los procesos MPI haya sido determinada y distribuida.

El hilo máster no espera en ese momento a la finalización de los envíos y recepciones, sino que continúa actualizando las calles restantes junto al resto de los hilos. De esta forma, se solapan el envío y recepción de solicitudes y notificaciones junto con cómputo producto de la actualización de las calles.

Las solicitudes, al igual que los segmentos, son enviados vía MPI utilizando un tipo de dado struct definido, cuyo nombre en el simulador es *MPI_SolicitudTranspaso*. Este contiene: identificador del vehículo, la época en que el vehículo fue introducido a la simulación, el número de metros recorrido por el vehículo hasta el momento, el identificador de la calle emisora de la solicitud, el identificador del barrio de la próxima calle a recorrer, y el identificador del barrio en que el vehículo ha sido generado. Mientras tanto, la notificación simplemente consiste en el identificador del vehículo, resultando al uso del tipo de MPI *MPI_INT* para su envío y recepción.

Una vez que todos los hilos hayan finalizado la actualización de las calles, el hilo master verificará (de forma bloqueante con *MPI_Waitall*) la terminación de las operaciones de recepción y vía *MPI_Get_count* obtiene el número de solicitudes y notificaciones recibidas. En simultáneo, el resto de los hilos realiza cada 1000 épocas la **Actualización de los costos de las calles**. Dicha actualización será descrita en la próxima subsección "Recalculo de caminos con base en congestión y velocidad media" (V-E).

Finalmente, una vez que ambos apartados han sido finalizados, un hilo comienza **Procesando las notificaciones**, mientras el resto de los hilos (a excepción del master) **procesan las solicitudes**. El procesamiento de las notificaciones es simple: dado el código de un vehículo notificado, la calle en la que actualmente se encuentra el vehículo en el proceso MPI es notificada, aunque en este caso, cuando se actualice dicha calle en la siguiente época, eliminará la instancia del vehículo, ya que el mismo no pertenece más al proceso MPI que ha sido notificado.

Por el contrario, el procesamiento de las solicitudes no es una tarea instantánea, ya que se debe calcular la ruta que el vehículo realizará dentro del nuevo barrio. Para ello se recurre al segmento enviado durante la generación de los vehículos y sus rutas. El mecanismo es el mismo que se describió cuando un vehículo era transferido entre dos calles con diferentes barrios, pero administradas por el mismo nodo MPI; se calcula la ruta dentro del barrio únicamente utilizando los nodos del barrio en cuestión, entre el nodo inicial y final del segmento.

Como ya fue mencionado, el número de épocas en que se encuentra cada vehículo no puede diferir en más de una unidad, y por extensión se deben **sincronizar los nodos MPI**. Entonces, mientras las solicitudes y notificaciones son procesadas, el hilo master de cada nodo MPI se sincronizan vía *MPI_Barrier*, lo que garantiza que antes de comienzo del cómputo de una nueva época, todos los procesos MPI hayan finalizado y recibido las solicitudes y notificaciones de sus procesos MPI vecinos de la época pasada.

Como se explicará en la sección "Balance de carga" (V-G) esto representa una debilidad a la solución, ya que el tiempo de cómputo por época está supeditado al proceso MPI que más lentamente procede la época.

Junto a la sincronización de los procesos MPI, cada unas 250 épocas, se calcula el número de vehículos globalmente en curso, permitiendo conocer si la simulación a o no finalizado y el número de vehículos aún en curso. Dicho número es determinado por la suma del número de vehículos que cada proceso MPI contiene, la operación es realizada mediante la instrucción *MPI_Allreduce*. El número de vehículos contenido en un proceso MPI inicialmente es el número de vehículos que generó, cada vez que recibe una solicitud de otro proceso MPI incrementa en una unidad, mientras tanto decrece cuando recibe una notificación, o un vehículo cuya última calle es administra por el nodo ha alcanzado su destino. Observar que la suma de dicha cantidades puede ser superior al número de vehículos realmente pendientes, ya que: si un proceso MPI ha recibido una solicitud, pero el nodo emisor de la solicitud aún no recibió o procesado la notificación, dicho vehículo es considerado por los dos nodos. De todas formas, el número calculado nunca puede ser menor a la cantidad de vehículos aún activos en el simulador, y, por lo tanto, la simulación no finalizará hasta que realmente todos los vehículos hayan alcanzado destino.

E. Recalculo de caminos con base en congestión y velocidad media

Un inconveniente de calcular el camino para cada uno de los vehículos al inicio de la simulación, es que existe la posibilidad de que se genere un "ciclo de dependencias" entre las calles. Esto se refiere, a que en un determinado momento durante

de la simulación existe un conjunto de calles, $c_1, c_2, c_3, ..., c_n$, donde ninguna tiene más espacio disponible, y además los vehículos que se encuentran en el final de c_1 solicitan ingreso a la calle c_2. Por lo tanto, la calle c_1 está inmovilizada hasta que c_2 acepte a los dichos vehículos. Lo mismo ocurre que c_2, se encuentra inmovilizada por c_3, y así sucesivamente hasta que finalmente c_n también encuentra inmovilizada por c_1.

El caso antes descrito es frecuente cuando el número de vehículos el elevado, las calles son pequeñas, y las calles están frecuentemente incluidas en rutas de los vehículos.

En consecuencia, se propuso un sistema de recálculo de ruta dentro de un bario, basado en la congestión y velocidad media de los vehículos que se encuentran en el barrio.

Para una calle dada, se proponen los dos siguientes indicadores:

- Congestión: $\frac{\text{Cantidad de vehículos actualmente en la calle}}{\text{Máxima capacidad de vehículos de la calle}}$
- Velocidad media: promedio de la velocidad de los vehículos actualmente en la calle, si no hay vehículo el valor es cero.

Basándonos en los dos anteriores indicadores, se propone la siguiente función de costo para la calle utilizando la congestión y velocidad media:

$$B \times (1 + c \times (\frac{v_{max} - v_{media}}{v_{max}})) \quad (4)$$

Donde:

- B es la función de costo original de la calle, en otras palabras: $\frac{\text{largo calle}}{\text{velocidad maxima} \times \sqrt{\text{numero carriles calle}}}$.
- C es la congestión antes calculada.
- v_{max} es la velocidad máxima de la calle.
- v_{media} es la velocidad media antes calculada.

Observar que en caso de que una calle este llena, y los vehículos inmóviles, el costo de la calle se duplica comparándolo con el costo original. Durante la actualización de cada calle, se calcula y almacena el valor de dicha función de costo.

Una vez cada mil épocas, los hilos de un proceso MPI ejecutan la **Actualización de los costos de las calles** (ver figura 2). La misma consiste en que por cada arista (una calle) del grafo de calles de la ciudad, se actualiza su costo, cómo el promedio de los mil valores de la función de costo calculado durante las épocas pasadas. Esto permite que los nuevos vehículos que ingresen al barrio, cuando calculen su ruta dentro de él, lo hagan considerando el nuevo costo de cada calle, el cual incluye congestión y velocidad media.

Además de lo antes mencionado, en necesario proporcionar un mecanismo el cual permita resolver un "ciclo de dependencias" entre las calles cuando el mismo se genera.

Se ha implementado el siguiente mecanismo: Cuando un vehículo solicita el traspaso a otra calle, la cual se encuentra dentro del mismo barrio, un contador con una cantidad pseudoaleatoria de épocas es activado (entre 400 y 700 épocas). Luego de cada época en que el vehículo permanezca en el final de la misma calle (su solicitud aún no fue aceptada) el contador descenderá en una unidad. Cuando alcance cero se calculará nuevamente la ruta con el menor costo desde el punto en que se encuentra hasta el final del segmento actual (o sea, hasta el final del barrio o hasta el final de su recorrido), y en caso de diferir será modificada. Observar que se utilizan los nuevos costos de la calle, y, por lo tanto, se intenta evitar las calles con elevada congestión y con una baja velocidad media.

Es importante señalar que, si la ruta difiere, y la próxima calle de la nueva ruta no es a la que se le envió la solicitud, previo confirmar el cambio de ruta, se confirma la existencia y se elimina la solicitud antes ingresada. En caso de que la misma no se encuentre, es porque fue aceptada y la calle emisora de la solicitud de traspaso aún no recibió la notificación.

Utilizando el anterior sistema fue posible ejecutar pruebas con un número elevado de vehículos sin experimentar los ciclos antes mencionados.

Otra alternativa, la cual no fue implementada, pero sí analizada, es "flexibilizar" los puntos de finalización de cada barrio. Con lo anterior nos referimos que un vehículo no deba necesariamente salir de un barrio hacia el siguiente por exactamente el mismo nodo de finalización indicado en el segmento del barrio actual, sino por uno próximo.

Formalmente: cuando se calcula la ruta de un vehículo dentro de un barrio, se busca el camino con el menor costo desde el nodo en que se ingresó al barrio hasta cualquier punto del próximo barrio, pero se le adiciona al costo de las aristas limítrofe (su nodo inicial está en el barrio actual y el nodo final en el próximo barrio), un costo adicional proporcional a la distancia entre el nodo de finalización de la calle y la distancia al nodo de finalización del segmento

F. Cálculo del tiempo de viaje medio

Una vez que todos los vehículos han finalizado su trayecto, se calcula el tiempo medio de viaje en 10 KM, basándonos la diferencia entre la época en que alcanza su destino y la época en que ingreso al tráfico (época inicial), y luego ajustada por la distancia recorrida.

Cada proceso MPI realiza la anterior operación para cada uno de los vehículos que finalicen su trayecto en alguno de los barrios que controla. Observar que cada vehículo acumula el número de metros recorridos, y el mismo es incrementado con la longitud de la calle cada vez que llega al final de la misma. Cuando un vehículo es traspasado entre dos calles administras por diferentes procesos MPI, parte de la solicitud contiene el número de metros recorridos y la época inicial.

El proceso MPI con rango cero será el encargado del cálculo del promedio del tiempo de viaje medio de todos los vehículos, donde se realizará una reducción (*MPI_Reduce*) con el tiempo de recorrido y la cantidad de vehículos de cada proceso MPI.

G. Balance de carga

Como se menciona anteriormente la ciudad se divide en los barrios que la conforman, estos barrios se distribuyen entre los procesos MPI. Como la ejecución de las épocas de los distintos procesos MPI debe estar sincronizadas, el tiempo de ejecución de una época del simulador esa limitada a la

del proceso MPI más lento. Además, como los vehículos se desplazan entre barrios administrados por diferentes procesos MPI, la carga de cada uno de ellos es variable. Dado el anterior caso, sería razonable considerar la posibilidad, de transferir el control de los barrios entre los procesos MPI, de forma tal de balancear la carga dinámicamente a medida que los vehículos son trasladados. Se descartó dicho enfoque debido a que la transferencia de información que es administra localmente por cada proceso MPI acerca de sus barrios es considerablemente elevada, por ejemplo la misma incluye: las posiciones y velocidades de cada vehículo, el estado de las solicitudes y notificaciones, los nuevos costos de las calles, entre otros.

En consecuencia, asignación de los barrios a los procesos MPI será realiza al principio de la simulación, será el proceso MPI con rango cero el que determina según la estrategia de balance de carga elegida qué barrio le corresponde a cada proceso MPI. La asignación es enviada via broadcast de forma completa a todos los procesos MPI de la sección. Las estrategias de balance de carga diseñadas e implementadas son las siguientes:

1) **Distribución por cantidad de vehículos en barrio:** Esta forma de distribución consiste en tomar el número de vehículos a computar por barrio y distribuir los barrios de tal forma de que cada proceso MPI obtenga una cantidad equitativa de vehículos.
2) **Distribución por cantidad de calles en barrios:** Esta distribución es similar a la anterior pero se balancea de tal forma de que los procesos MPI obtengan similar cantidad de calles a procesar.
3) **Distribución por barrios adyacentes:** La motivación de este tipo de balance de carga surge de que en los dos métodos anteriores los barrios que recibe cada proceso MPI generalmente no son adyacentes, por lo que cada pasaje de barrio de un vehículo implica realizar mensajería MPI, para disminuirla se propone el balance de carga por barrios, este consiste en la realización de una partición de la ciudad de tal forma que los procesos MPI tengan que gestionar barrios que sean adyacentes. En la figura 3 se puede observar una posible distribución para 6 procesos MPI.

La decisión de qué balance de carga utilizar es dependiente de la ciudad en la que se quiera trabajar, dado que hay ciudades que tienen un fuerte acoplamiento de personas en barrios. Al iniciar la simulación, los vehículos son creados en los barrios de forma proporcional a la cantidad de personas que habitan cada barrio, luego los barrios se van asignando de tal forma de que las asignaciones de los barrios a procesos MPI sean lo más parejas en vehículos. La segunda forma tiene la ventaja de que cada proceso MPI revise la misma cantidad de calles que computar, mientras que la cantidad de vehículos puede ir aumentando o disminuyendo, en un proceso MPI la cantidad de calles siempre es fija. Por último la distribución por barrios tiene una pérdida en tiempo en cuanto a la distribución de caminos pero tiene menor cantidad de mensajes MPI que transmitir entre los procesos.

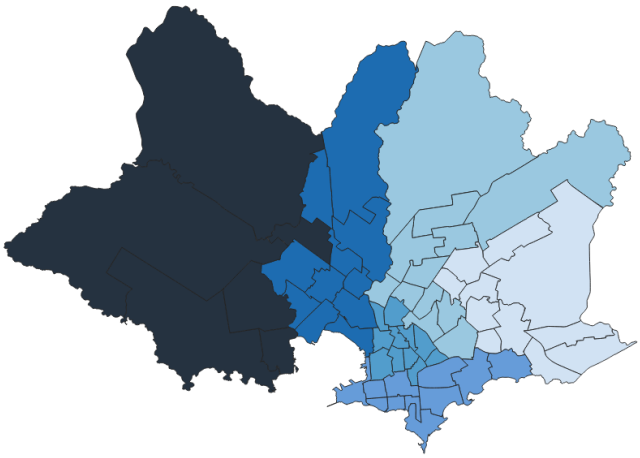

Fig. 3. Distribución de los barrios para 6 procesos MPI, utilizando balance de carga por barrios

A continuación se realiza una prueba en Montevideo con 100.000 vehículos y 6 procesos MPI utilizando las tres formas de balancear la carga, por cada balance se realizaron tres pruebas con las que se realiza un promedio de los resultados para ese balance. Para el caso de por barrios se utiliza la distribución que se muestra en la Figura 3.

Los campos de la Tabla significan: Tiempo de simulación (excluyendo el cálculo de caminos, ya que de esta forma es más notorio el efecto de la sincronización entre procesos MPI, y por ende las ventajas del balance de carga), el tiempo total de ejecución, y el tiempo de cálculo de caminos, en ”min” el proceso MPI que primer termina y en ”max” el que termina el cálculo de dichos caminos por último, claramente esto es dependiente de la cantidad de vehículos que tenga que procesar, la centralidad del barrio y la distribución de los barrios destino para los barrios de origen.

TABLE I
RESULTADOS BALANCES DE CARGA

Balance	Tiempo simulación	Tiempo Total	caminos [min,max]
Vehículos	165s	202s	32 ; 34 s
Calles	169s	210s	27 ; 39 s
Barrios	174s	225s	20 ; 48 s

En la Tabla I se puede ver que, como es de esperar, el balance por vehículos es más estable en el tiempo que le lleva calcular los caminos, dado que a cada proceso le toca una cantidad similar de vehículos.

Un resultado interesante es que el balance de carga por barrios está siendo el peor, incluso en tiempo de simulación, lo que nos indica que al utilizar mensajes de un tamaño pequeño, dicha comunicación no presenta un overhead elevado.

H. Arquitectura y etapas de la simulación

Ahora que ya ha sido detallado los distintos apartados que confirman el simulador, se detallara la arquitectura del

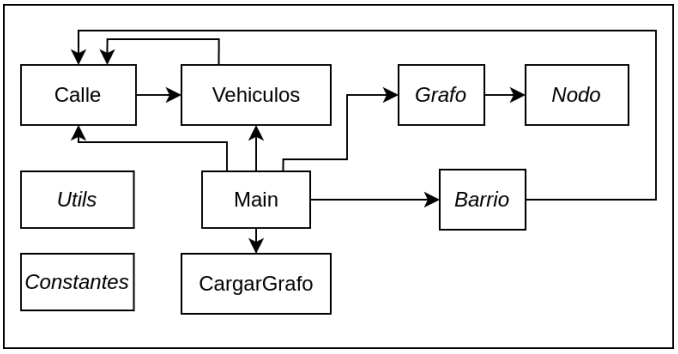

Fig. 4. Diagrama de clases del simulador

mismo, mediante un diagrama de clases. En la figura 4 ilustra los ficheros implementados. Todos los ficheros son clases, a excepción de ”Utils”, ”Constantes”, ”CargarGarfo y ”Main”. La finalidad de cada misma se describe a continuación:

- **Main**: Contiene la lógica principal del programa. Esta incluye la creación de los vehículos, envío cursado de segmentos, asignación de barrios a procesos MPI, procesamiento y la ejecución de las épocas.
- **Barrio**: Contiene las calles que integra el barrio y cuáles son los barrios vecinos.
- **Vehículo**: Contiene la representación del vehículo, en particular la distancia recorrida, la calle actual, el barrio de origen y la ruta.
- **Grafo y Nodo**: Contiene la representación del grafo de las calles explicado en la sección ”Estructura y División del mapa” (V-B). Permite el cálculo de los caminos más cortos utilizando A-star y Dijkstra.
- **Calle**: Contiene la representación de la calle, la lógica de movimiento de los vehículos y generación de solicitudes y notificaciones.
- **Utils**: Contiene los procedimientos comunes utilizados en el resto de la lógica del simulador. Por ejemplo, el cálculo de distancia en Haversine, o la definición de los struct de Segmento, Solicitud y Asignación a Barrio.
- **CargarGrafo**: Contiene la lógica necesaria para la lectura del JSON que contiene la estructura del grafo de las calles de la cuidad.

Por último, en la figura 5 se ilustra en un diagrama secuencial las principales etapas del simulador y la comunicación entre los distintos procesos MPI. Como cada parte ha sido descrita en detalle en las anteriores subsecciones, procederemos a una descripción general del proceso:

1) El proceso MPI con rango 0, según el balance de carga, asigna los barrios a los distintos procesos MPI, junto con la cantidad de vehículos que cada uno debe generar. La asignación es conocida por todos los procesos. Para más detalle, leer la subsección ”Balance de carga” (V-H).
2) Se generan los vehículos para cada barrio, para cada uno de ellos se eligen el nodo de origen (en el mismo barrio) y el destino según la matriz M. Se generan las rutas utilizando el algoritmo A-star para cada vehículo, y se envían los Segmentos a los demás nodos se forma cruzada. Para más detalle, leer la subsección ”Cálculo de caminos iniciales, distribución de segmentos y generación de vehículos” (V-C).
3) Mientras algún vehículo aún no haya terminado, se computa ejecuta la época en cada proceso MPI, consistiendo él la actualización de las calles, el envío y recepción de solicitud y notificaciones dentro y entre los procesos MPI, sincronizar dichos procesos y contabilizar el número de vehículos pendientes. Para más detalle, leer la subsección ”Ejecución de una época” (V-D) y leer la subsección ”Recalculo de caminos con base en congestión y velocidad media” (V-E).
4) Por último, el proceso MPI con rango cero calcula el tiempo medio de viaje mediante los resultados parciales del resto de los procesos MPI. Para más detalle, leer la subsección ”Cálculo del tiempo de viaje medio” (V-F)

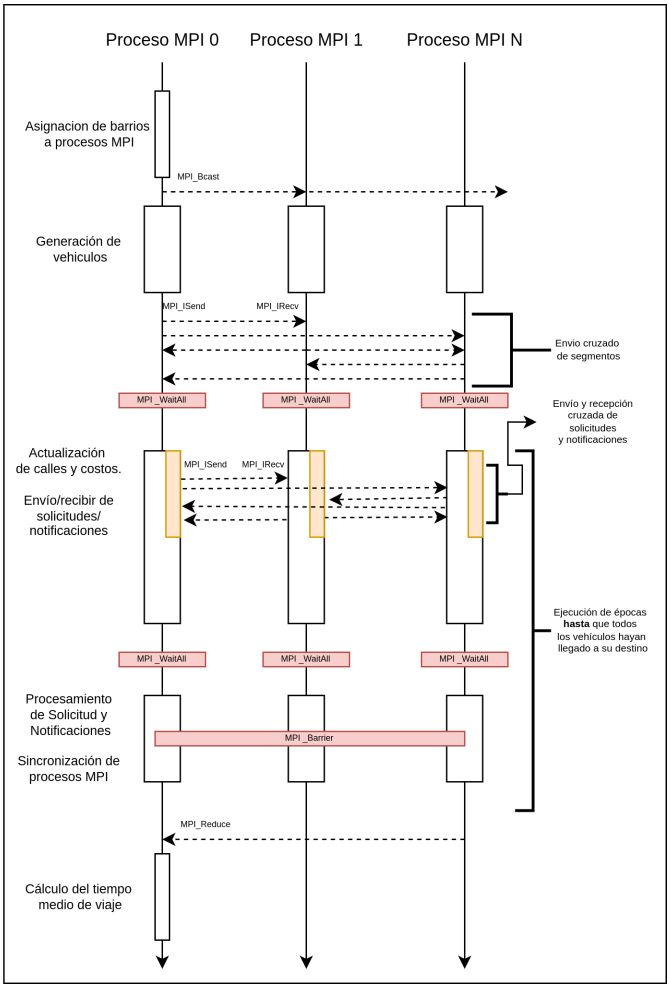

Fig. 5. Diagrama secuencial del simulador

I. *Tolerancia a fallos*

En el desarrollo del sistema, se plantea implementar una estrategia de tolerancia a fallos mediante la creación de puntos de control (checkpoints) que se registrarán cada una cierta can-

tidad de épocas. Estos puntos de control utilizarán un logger para registrar el estado del sistema, lo que permitirá recuperar el progreso en caso de un fallo. Posteriormente, se realizará un proceso de restauración desde el último checkpoint disponible. Esta funcionalidad es una prioridad para el trabajo futuro y contribuirá a mejorar la robustez del sistema, pero por falta de tiempos no fue implementada en el simulador descrito.

VI. Evaluación experimental

En esta sección experimentará con el simulador desarrollado con el fin de explorar su correcto funcionamiento, simular diferentes escenarios en Montevideo, analizando lo mismos y el aumento de rendimiento del simulador (en particular la paralelicibilidad) debido al uso de programación paralela.

A. Evaluación del Modelo y experimentación

En esta sección se presentan variantes para las que puede ser de utilidad el simulador creado, así como mostrar el correcto funcionamiento del mismo. Por último se mostrará el resultado de como aumenta el tiempo medio de viaje en Montevideo al aumentar la cantidad de vehículos.

La simulación captura para cada vehículo el tiempo y la distancia que tuvo que recorrer para llegar a su destino, computando para cada vehículo la siguiente relación:

$$\frac{tiempo}{distancia} * 10000 \text{ metros}$$

obteniendo el tiempo que le toma a ese vehículo recorrer una distancia ajustada a 10km. Las entradas modificables al simulador son las siguientes:

- Número de vehículos que van a ser simulados.
- Matriz M que indica la probabilidad que tiene un vehículo que parte de un barrio ir a otro.
- Ciudad sobre la que se quiere lanzar la ejecución.
- Archivo con extensión CSV que indica la cantidad de personas que habitan cada barrio de la ciudad elegida. Esta información es usada para generar en los barrios una cantidad proporcional a las personas que viven en el.

A modo de ejemplo tomaremos la ciudad de Montevideo, sobre la cual se simulan 100.000 vehículos en diferentes situaciones. La información respecto a la cantidad de personas que habitan cada barrio se obtuvo del censo 2011 [8], cuya distribución es bastante uniforme como se muestra en la Figura 6.

En primer lugar, veamos que sucede en una prueba simple donde la matriz M se resuelve de manera equiprobable, es decir, que la probabilidad de ir a un barrio a los otros es la misma. Los resultados obtenidos se reflejan en la Figura 7 El graduado para un barrio b1 índica el tiempo en relación con 10km que le lleva llegar a un vehículo que parte de un barrio b2, pero que termina en b1. En dicha figura se muestra la tendencia de que los barrios que están en el centro tiendan a demorar más tiempo, esto tiene sentido, ya que al ser probabilidad equiprobable, los vehículos tienden a ir hacia el centro, por lo que la congestión en dicha será mayor, además de estos resultados se puede concluir que el simulador se está comportando de manera esperada.

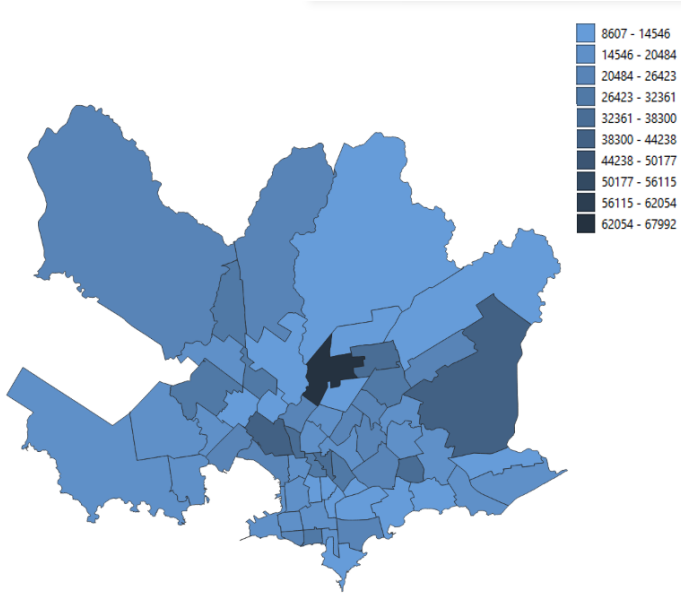

Fig. 6. Cantidad en miles de personas por barrios de Montevideo - Censo 2011

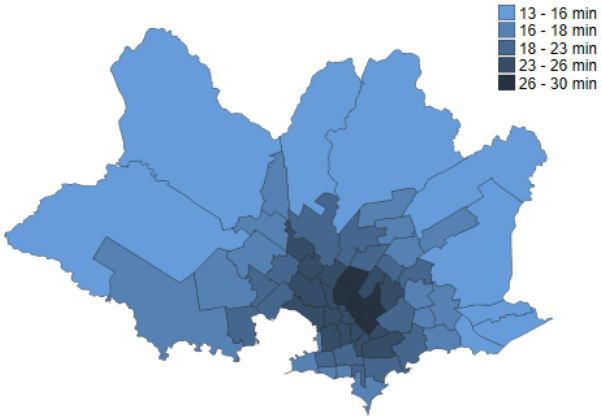

Fig. 7. distribución de demoras con matriz M equiprobable

La siguiente prueba consiste en modificar la matriz M de tal forma de que los vehículos tiendan a ir hacia los barrios más concurridos de Montevideo, estos son Tres Cruces, Cordón, Punta Carretas, Ciudad Vieja y Centro. Estos barrios se pueden ver en amarillo en la Figura 8. En esta prueba los vehículos de todos los barrios tendrán un 50% de probabilidad de tener como destino uno de los anteriores barrios mencionados (en particular 10% a cada uno) y equiprobabilidad al resto. El graduado de la Figura 8 es igual que el anterior, aquí se nota que los barrios que estan mas lejos de zona centro al estar menos congestionados van a tener tiempos buenos de viaje mientras que los barrios cercanos a zona centro sufren los embotellamientos de aquellos autos que van hacia zona centro.

La próxima prueba consiste en modelar una situación en la que ocurre un evento importante en uno de los barrios de la

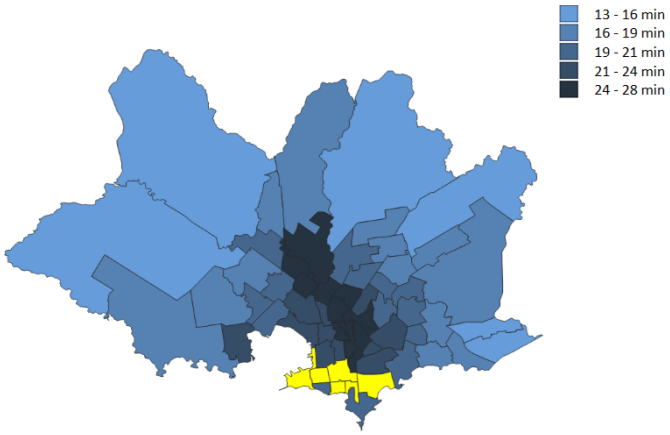

Fig. 8. Distribución de demora con probabilidad 0.5 a zona centro

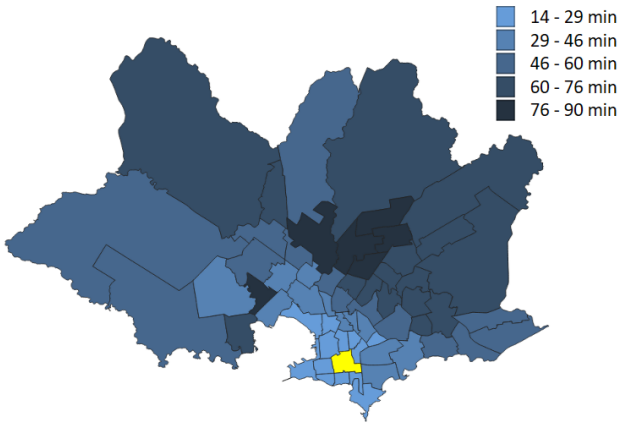

Fig. 9. Distribución de demora donde todos los barrios deciden con probabilidad 0.9 ir al Centro

ciudad, supongamos que este evento se da en Cordón, ¿cuanto sería el tiempo que le llevaría desde cada uno de los barrios acudir al evento? Además, supongamos que la decisión de acudir o no es del 90%. Para realizar esta prueba necesitamos configurar la matriz M de tal forma que la probabilidad de ir a cordón sea de 90% y repartir el 10% restante entre los demás barrios. Además, como se quiere ver como se afecta cada barrio al ir a Cordón se cambiará la semántica de la gráfica, cada graduado significa cuanto tiempo le lleva en promedio a los vehículos que tienen como barrio origen ese barrio llegar a Cordon. Los resultados obtenidos se expresan en la Figura 9, nuevamente el patrón que se cumple tiene sentido ya que los barrios que tienen que atravesar más barrios van a tener una mayor demora ya que todos quieren ir hacia el mismo lugar, además se puede observar como esta escala aumenta considerablemente, pasando a demorar 70-90 minutos, un tiempo considerable para solamente 10km.

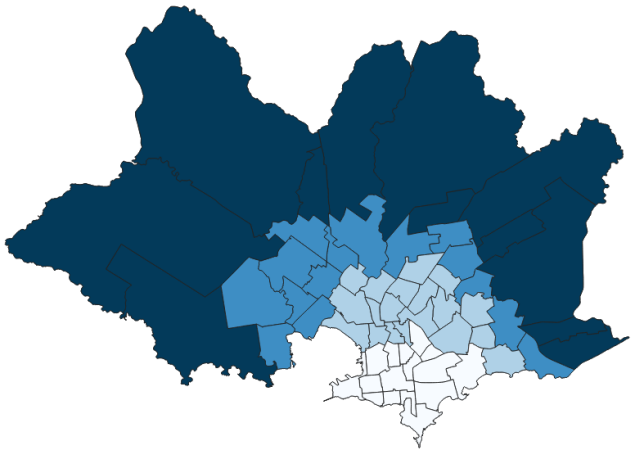

Fig. 10. División de Montevideo en 4 zonas

B. Aumento de tiempo de Traslado

Una de las motivaciones de realizar el simulador era determinar el aumento del tiempo de traslado al aumentar la cantidad de vehículos que se simulaban, utilizando nuevamente la ciudad de Montevideo se realizan pruebas con distintos números de vehículos y se capturan los tiempos medios de traslado, esta información se puede visualizar en la Figura 11 donde es interesante notar el considerable aumento que ocurre pasando los 100.000 vehículos, los embotellamientos empiezan a tener efecto y producen que el tiempo se dispare hasta llegar a 60 minutos, imaginemos una situación donde el trabajo de una persona se encuentre a 20km y transite en una ciudad como Montevideo con 120.000 vehículos, tardaría un total de dos horas en solo acudir a trabajar! Las pruebas realizadas aquí hacen uso de una matriz M que se cree representa una mejor visión de la realidad, si bien no hay datos hasta la fecha de este informe, que represente las probabilidades de barrios a barrios, se realizó de forma analítica cual era la forma de hacer que represente mejor la realidad. Por lo que la ciudad de Montevideo fue dividida en 4 zonas como se muestra en la Figura 10 donde se define una probabilidad de ir de una zona a la otra, por lo que la matriz M se ajusta cada barrio dependiendo de la zona a la que pertenezca. La Zona 1 es representada en la figura 10 con un tono blancuzco, consecutivamente las zonas 2 y 3 hasta llegar a la zona 4 que es la más oscura en la figura. En la tabla II se referencian las probabilidades correspondientes a las zonas.

TABLE II
TABLA DE PROBABILIDAD DE IR DE UNA ZONA A OTRA

-	**Zona1**	**Zona2**	**Zona3**	**Zona4**
Zona1	0.4	0.3	0.2	0.1
Zona2	0.35	0.30	0.20	0.15
Zona3	0.4	0.2	0.2	0.2
Zona4	0.1	0.15	0.45	0.30

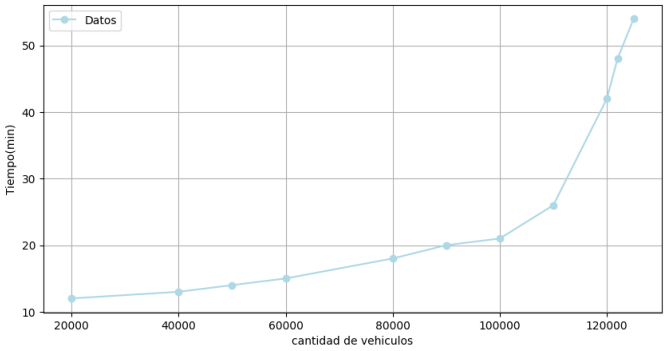

Fig. 11. Tiempo Traslado de distintas cantidades de vehiculos

C. Rendimiento del Simulador

Speedup algorítmico se define como

$$S_n = \frac{T_1}{T_n}$$

donde T_1 es el tiempo de cómputo del algoritmo secuencial y T_n el tiempo del algoritmo paralelo en N recursos de cómputo. Dado que la implementación no es serial, no es posible realizar esta métrica directamente. En cambio, la Paralecibilidad se define como:

$$P = \frac{TP_1}{TP_n}$$

siendo TP_1 el tiempo que toma a un algoritmo paralelo en un único recurso de cómputo, y TP_n el tiempo que toma al mismo computador paralelo ejecutar el mismo algoritmo en N recursos de cómputo, esta métrica tiene más sentido en este problema.

En primer lugar, de analista en la ciudad de Montevideo, simulando 100.000 vehículos en diferentes cantidades de equipos y núcleos. En la tabla III se puede observar el número de computadoras y núcleos utilizados, el tiempo de cómputo total del simulador y el mismo excluyendo el cálculo de los caminos iniciales. Notemos que se realiza esta distinción debido a que las técnicas de balance de carga, solapamiento de envíos con cómputo, y demás técnicas ya explicas se pueden apreciar mejor si no se considera en el tiempo el cálculo de los caminos, que son actividades que no requiere comunicaciones ni sincronizaciones. Finalmente, se presenta la ”paraleliciblidad de simulación”, y ”paralecibilidad total” (considerando el tiempo total del simulador).

Los tiempos en single core realizados en Montevideo fueron de alrededor de 30 minutos para 100.000 vehículos, por lo que sería interesante realizar un cálculo de paralecibilidad para una ciudad más grande que Montevideo como puede ser Cuidad de Buenos Aires (CABA), por lo que se procedió a simular 100.000 vehículos en esta ciudad. Los resultados obtenidos son los expresados en la Tabla IV, se puede observar que los tiempos en single core aumentan más del doble, a 1 hora 20 minutos aprox. pasando a durar 5 minutos en 30 cores. La gráfica de paralecibildad se puede apreciar en la Figura 12 Existen ciudades extremadamente más grandes que

TABLE III
PARALELICIBILIDAD PARA MONTEVIDEO

Equipos	**Núcleos**	**Tiempo sim(s)**	**Tiempo total(s)**	**Paral. Sim.**	**Paral. Total**
1	1	1660	2194	1	1
1	4	439	577	3,78	3,80
2	4	236	307	7,03	7,15
3	4	174	223	9,54	9,84
4	4	147	184	11,29	11,92
5	4	130	160	12,77	13,71
6	4	114	140	14,56	15,67
7	4	110	133	15,09	16,50
8	4	102	122	16,27	17,98

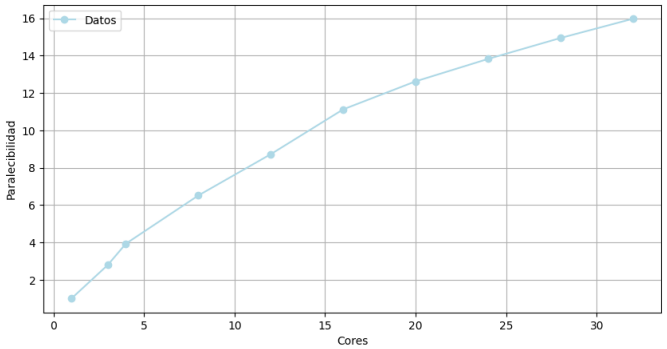

Fig. 12. Gráfica de paralelicibilidad

Buenos Aires como puede ser Tokio, a efectos de analizar la paralecibilidad del algoritmo se considera que ya se puede apreciar en ciudades pequeñas, ciudades como Tokio al ser mas grande requieren mayores niveles de preprocesamiento que no es el objetivo principal de este trabajo.

TABLE IV
DATOS DE SIMULACIÓN Y PARALELIZACIÓN

Equipos	**Núcleos**	**Tiempo sim(s)**	**Tiempo total(s)**	**Paral. Sim.**	**Paral. Total**
1	1	3.123	4.596	1.00	1
1	3	1.116	1.646	2.80	2.79
1	4	813	1.170	3.84	3.93
2	4	529	706	5.90	6.51
3	4	402	527	7.77	8.72
4	4	320	413	9.77	11.12
5	4	287	364	10.89	12.62
6	4	267	333	11.70	13.82
7	4	247	308	12.64	14.94
8	4	236	288	13.23	15.97

Finalmente, observar que si el número de vehículos aumenta, el tiempo requerido para la simulación aumenta considerablemente, observar en la Figura 13 los tiempos de ejecución cuando el simulador es ejecutado utilizando 16 núcleos. Eso justifica aún más el uso de HPC para este problema.

VII. CONCLUSIONES

Se logró diseñar y construir un simulador con representación microscopia que consigue una paralelización considerable. Las

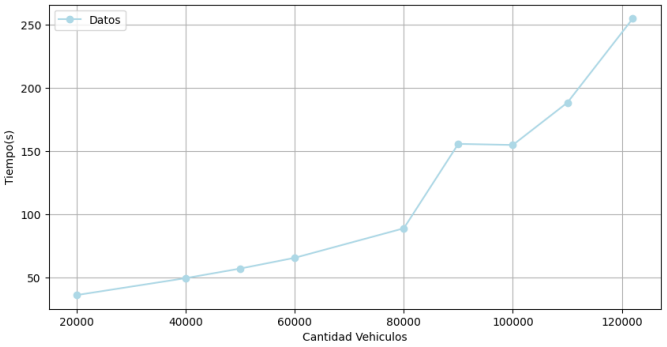

Fig. 13. aumento de la cantidad de vehículos usados para 16 núcleos

bases de la paralelización ya están desarrolladas, y aunque no se pudieron implementar los comportamientos avanzados de los vehículos y calles, la infraestructura necesaria para poder hacerlo utilizando las ventajas de HPC ya han sido desarrolladas. Con el fin de obtener resultados más realistas, el enfoque a futuro debería ser en la ingeniería de tráfico, reglamentación de la ciudad simulada, cálculo de camino más adecuados, entre otros.

REFERENCIAS

[1] Web oficial TomTom (https://www.tomtom.com/traffic-index/ranking/)
[2] ”An overview of microscopic and macroscopic traffic models”, Bachelorondersoek Technische Wiskunde, Países Bajos, Julio 2013, (https://fse.studenttheses.ub.rug.nl/11050/1/Bachelorproject.pdf)
[3] Web oficial software SUMO (https://eclipse.dev/sumo/)
[4] ”Traffic Flow Dynamics”, Martin Treiber y Arne Kesting, ISBN:9783642324598
[5] Web oficial software QGRIS (https://www.qgis.org/)
[6] Sistema de Información Geográfica (https://sig.montevideo.gub.uy/)
[7] ”Informe Censos 2011 :Montevideo y Área Metropolitana”, ”Intendencia de Montevideo, Uruguay, Noviembre 2023, (https://montevideo.gub.uy/sites/default/files/informe_censos_2011_mdeo_y_area_metro.pdf)
[8] ”Informe Censos 2011 :Montevideo y Área Metropolitana”, ”Intendencia de Montevideo, Uruguay, Noviembre 2023, (https://montevideo.gub.uy/sites/default/files/informe_censos_2011_mdeo_y_area_metro.pdf)
[9] ”Algoritmo de búsqueda A*”, Portal de Wikipedia,(https://es.wikipedia.org/wiki/Algoritmo_de_b
[10] ”Haversine formula”, Portal de Wikipedia,(https://es.wikipedia.org/wiki/F