PLATAFORMA DE MICROSIMULACIÓN PARA SISTEMAS DE DESPACHO CON DEMANDA ON-LINE.

Gabriel Moreno C., Doris Sáez H.
Departamento de Ingeniería Eléctrica
Universidad de Chile, Santiago, Chile
Fono: (56-2) 978 4922, Fax: (56-2) 672 0162
E-mail: gmoreno@ing.uchile.cl, dsaez@ing.uchile.cl

Cristián E. Cortés
Departamento de Ingeniería Civil
Universidad de Chile, Santiago, Chile
Fono: (56-2) 978 4380, Fax: (56-2) 689 4206
E-mail: ccortes@ing.uchile.cl

RESUMEN

En este trabajo se presenta el desarrollo de una plataforma de simulación microscópica para validar en forma empírica el desempeño de un sistema de despacho de vehículos con demanda de pasajeros *on-line* y desconocida. Se simula tanto la flota vehicular que provee el servicio como la red vial correspondiente a la zona de cobertura de la flota. En el futuro se implementarán estrategias de control predictivo híbrido para la gestión de la flota.
El simulador se desarrolla usando la plataforma de simulación *QUADSTONE PARAMICS*, cuyas funcionalidades fueron extendidas mediante un *plugin* hecho a medida. Para el desarrollo de dicho *plugin* se utilizó el *API* ofrecido por *QUADSTONE PARAMICS*, compatible con los lenguajes de programación *C/C++*.
Se presentan pruebas en lazo abierto que permiten evaluar y validar el desempeño del microsimulador.

ABSTRACT

In this work, we present a microscopic simulation platform in order to empirically validate the performance of a vehicle dispatch system with online and unknown passenger demand. We simulate both, the fleet that provides the service and the details of the network (infrastructure) corresponding to the fleet deployment. In the future, we plan to implement hybrid predictive control strategies for the online management of the fleet.
The simulator is developed on a simulation platform *QUADSTONE PARAMICS*, whose functionalities were extended by means of a problem-specific *plugin*. For the development of such a *plugin* we used the available APIs provided by the software, compatible with the in the *C/C++* programming languages.
Open loop tests are presented, which allow us to evaluate and validate the performance of the microsimulation tool.

1. INTRODUCCIÓN

En esta investigación, el objetivo es desarrollar una plataforma de simulación detallada a nivel microscópico, para validar en forma empírica el desempeño de un sistema de despacho de vehículos con demanda de pasajeros *on-line* y desconocida. El sistema a representar es por lo tanto un sistema dinámico, el cual debe ser actualizado en tiempo real bajo reglas específicas de optimización, para lo cual hemos desarrollado en el último tiempo enfoques de control predictivo, y algoritmos de solución online eficientes para lograr mejores soluciones dinámicas de ruteo que benefician tanto a los operadores como a los usuarios que solicitan el servicio (Sáez et al., 2008; Cortés et al., 2008; 2009). La mejor forma de evaluar el rendimiento de un esquema de este tipo, es mediante una simulación detallada del sistema de transporte, representando las condiciones reales de tráfico y su evolución en el tiempo. Además, el funcionamiento de los vehículos de la flota será altamente sensible a cambios tanto en la operación a nivel de terminales, como a nivel de paradas (ya sea para dejar o recoger pasajeros), lo cual hace imprescindible el detalle microscópico para evaluar su rendimiento. Notar que los cambios en la operación específica de vehículos debieran ser relevantes para la actualización online de las reglas de operación y decisiones de despacho, que en definitiva, son determinantes en la evaluación global del sistema.

Por otra parte, a nivel de desarrollo e implementación de algoritmos, existen ciertas decisiones que deben ser tomadas a nivel de rutas (*path level*), y en la actualidad el excesivo detalle de codificación de los modelos microscópicos hace muy difícil que estos sean capaces de mantener en memoria la ruta completa de cada uno de sus vehículos en todo instante. Así por ejemplo, se puede mencionar el esfuerzo computacional para calcular dinámicamente la ruta mínima desde la posición de un vehículo controlado por el operador hasta el lugar de una llamada por servicio, si las redes de decisión se mantienen a un nivel muy detallado. Oh et al. (2000) desarrollaron un enfoque para simular *Advanced Traffic Management and Information Systems* (ATMIS) en redes de gran tamaño a nivel microscópico, integrando la capacidad para procesar rutas, propias de modelos macroscópicos. Ellos desarrollaron un esquema híbrido de simulación integrando el modelo de microsimulación PARAMICS (PARAllel MICroscopic Simulation) desarrollado en Escocia (Duncan, 1995; Quadstone, 2007), con los modelos de comportamiento y elección de ruta implementados dentro del modelo mesoscópico de asignación dinámica DYNASMART (*Dynamic Network Assignment Simulation Model for Advanced Road Telematics,* Jayakrishnan et. al, 1994), de tal forma que el simulador integrado podía evaluar esquemas de información en la ruta a nivel microscópico usando modelos de comportamiento y elección de ruta a nivel macroscópico. Este enfoque se basa en la integración de redes en dos niveles de abstracción y comunicación respecto de la posición de los vehículos entre la red detallada (requerida por PARAMICS) y la red más abstracta (requerida por DYNASMART), cuyo detalle sólo incorpora aquellos nodos en los cuáles debe tomarse una decisión a nivel de ruta. Las decisiones de ruta para cada vehículo son procesadas a nivel de la red más abstracta, y son entonces transmitidas a la simulación a nivel de red mucho más detallado donde se controlan los vehículos a nivel microscópico. Más adelante Jayakrishnan et al. (2003) y Cortés et al. (2005) desarrollaron una propuesta de simulación para distintas clases de vehículos en redes urbanas usando la plataforma establecida por Oh et. al(2000), con la potencialidad de controlar los vehículos tomando decisiones en la red más agregada.

En este trabajo, hemos actualizado la plataforma de simulación propuesta, basándose en el mismo

concepto de agregación de red para tomar decisiones online en base a diversos algoritmos de optimización diseñados ad-hoc. Se mantiene la estructura de simulación utilizando PARAMICS, donde los códigos que controlan los vehículos se conectan vía API (*Application Programming Interface*), mientras que las operaciones detalladas de los vehículos simuladas a nivel micro sobre la red de transporte las realiza el simulador comercial PARAMICS, en este caso.

El artículo está estructurado como sigue. En la siguiente sección, se describe el sistema de despacho de vehículos que se desea simular en detalle. En la sección 3 se detalla los distintos módulos del simulador desarrollado, poniendo énfasis en la forma de controlar los vehículos, decidir sus rutas y detenerlos adecuadamente; además, se describe con detalle la forma de comunicar la red a nivel agregado (donde se toman las decisiones de ruteo) con la red a nivel desagregado (donde se mueven los vehículos). Luego, en sección 4 se muestra un ejemplo de la aplicación computacional, cerrando en sección 5 con conclusiones y líneas de investigación futura.

2. DESCRIPCIÓN DE UN SISTEMA DE DESPACHO DE VEHÍCULOS CON DEMANDA *ON-LINE*.

El sistema que se simuló consiste en una flota de vehículos que ofrece el servicio de transporte colectivo de pasajeros desde y hacia direcciones arbitrarias en la zona de cobertura del sistema. Las direcciones de origen y destino son previamente establecidas por cada cliente al momento de solicitar el servicio, vía telefónica, a la central de operación de flota y son completamente desconocidas por dicha central en forma previa a la realización de la llamada. El uso de los vehículos es compartido por los pasajeros, es decir en cada vehículo pueden viajar varios clientes, cada uno con sus propias direcciones de origen y destino.

Las características del servicio ofrecido requieren de la generación eficiente de recorridos por parte del despachador de la flota, de acuerdo a las condiciones presentes de tráfico y a los distintos requerimientos de viaje pendientes. La asignación de requerimientos a cada vehículo se debe actualizar en forma dinámica, cada vez que se recibe una llamada solicitando el servicio, de otra forma los tiempos de espera y de viaje de los pasajeros serían excesivamente grandes, lo que desencadenaría insatisfacción en los clientes. Si no se dispone de recorridos eficientes también se incurriría en una distancia recorrida acumulada por cada vehículo excesivamente grande y en el uso ineficiente del espacio abordo. Esto último produciría aumentos en los costos operacionales de la flota. Tanto la insatisfacción de los pasajeros, así como el aumento en los costos operacionales son factores que harían inviable el servicio desde el punto de vista económico.

En la Figura 1 se observa las ubicaciones de origen y destino de cuatro clientes que solicitaron ser atendidos por el servicio. En este caso la flota consiste en dos vehículos, cuya ubicación en la zona de cobertura también se puede observar en la misma figura.

De acuerdo a las ubicaciones tanto de los vehículos como de los requerimientos, además de las condiciones actuales de tráfico y de la topología de la red, una posible solución óptima al problema podrían ser las dos secuencias que se muestran en la Figura 2.
Dado que la naturaleza del servicio es dinámica, cada vez que se genera un nuevo requerimiento

es necesario actualizar al menos la secuencia de un vehículo para que dicho requerimiento sea atendido. Cada actualización de secuencias involucra la resolución de un nuevo problema de optimización, ya que se debe procurar que la inclusión del nuevo pasajero no aumente demasiado los tiempos de espera y de viaje de los pasajeros que se encontraban previamente en el sistema, además de ofrecer un servicio de calidad al nuevo pasajero.

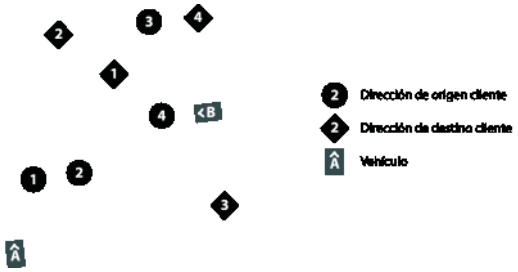

Figura 1. Flota de dos vehículos con cuatro requerimientos por atender y sus respectivas ubicaciones.

En la Figura 3 se muestra la llegada de un nuevo requerimiento en un instante posterior al expuesto en las Figuras 1 y 2.

Existen diferentes formas de incorporar el nuevo requerimiento al sistema. Cada una ofrece beneficios y perjuicios de diferente magnitud al sistema completo, magnitudes que son medibles en términos de disminución o aumento en los tiempos de servicio de la flota. En las Figuras 4(a) y 4(b) se muestran dos posibles soluciones, las cuales no necesariamente son únicas.

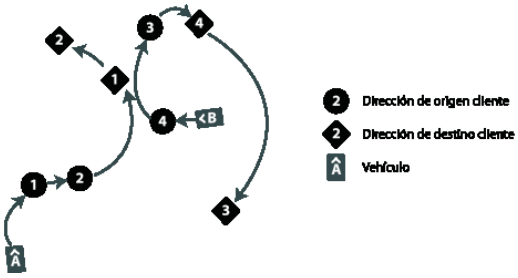

Figura 2. Posibles secuencias óptimas.

De entre todos los conjuntos de nuevas secuencias posibles, el operador deberá elegir aquel que minimice sus costos operacionales y que mantenga un nivel de calidad de servicio aceptable para sus clientes.

Figura 3. Llegada de un nuevo requerimiento.

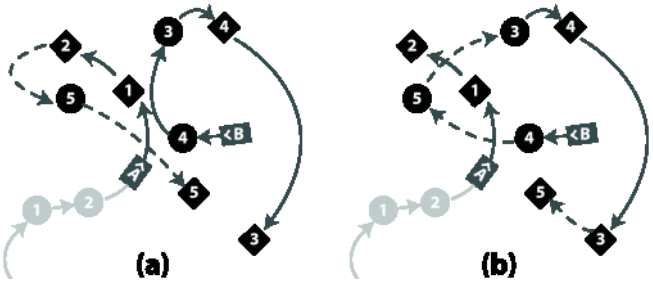

Figura 4. Dos posibles inserciones del requerimiento número 5.

3. DESCRIPCIÓN DE LA PLATAFORMA DE MICROSIMULACIÓN DESARROLLADA.

3.1. Antecedentes.

Un simulador para un sistema de despacho de vehículos requiere de dos componentes: un modelo de la red que cuente con vehículos que se comporten de acuerdo a la realidad y un modelo de la flota que provee el servicio.

- Variables de Salida:
Las variables de salida del bloque son la Posición (*X*) y carga (*L*) de cada uno de los vehículos de la flota, calculados para toda la secuencia asignada que queda por recorrer.

- Variable Manipulada:
Conjunto de secuencias actualizadas (*S*) de cada vehículo. Consiste en una tabla que contiene la lista de requerimientos asignados a cada vehículo de la flota para que éstos los satisfagan. Cada secuencia debe consistir en un itinerario optimizado por un sistema de control inteligente. El controlador debe realizar la optimización de las secuencias tomando como base la información provista por las variables de salida y el estado actual de la red.

- Perturbaciones:
Las perturbaciones del sistema son la demanda (η) y la distribución de velocidades actual en la red (φ). La demanda corresponde al conjunto de llamadas de los clientes. Una llamada consiste en una solicitud de servicio que ocurre en un instante de tiempo definido, involucra a una cantidad de pasajeros determinada, además de una dirección de origen y otra de destino.

En el entorno de simulación, el Proceso de Ruteo que se implementó consiste en la flota que se desea controlar y todas sus interacciones con la red. Este bloque recibe como entrada una tabla estandarizada [*S(k-1)*] que contiene las secuencias con requerimientos de cada vehículo. Al recibir una nueva tabla con secuencias, la flota debe reaccionar adecuando las trayectorias de cada vehículo de tal forma que se satisfagan los requerimientos solicitados por la entrada. En un entorno de microsimulación se deben simular las detenciones de los vehículos al llegar a las direcciones planificadas. Dichas detenciones involucran cambios de pista para acercar el vehículo a la vereda para recoger a los pasajeros, tal como ocurriría en una red real, y una detención por un intervalo de tiempo suficiente para que los pasajeros aborden o desciendan de los vehículos.

A partir del comportamiento la flota interactuando con la red se obtienen los valores de las variables de salida del sistema que, como fue mencionado, corresponden a la Posición [*X(k)*] y carga [*L(k)*] de cada vehículo, calculados para cada uno de los requerimientos que quedan por recorrer de las secuencias de los vehículos.

Las perturbaciones que recibe el bloque son simuladas de diferente forma. Por una parte la distribución de velocidades en la red es generada en forma automática por la plataforma de microsimulación como producto de la interacción existente entre todos los vehículos presentes en la red y la topología de la red. La demanda por servicio debe ser simulada en forma externa a la plataforma de simulación.

La plataforma de microsimulación Quadstone PARAMICS ofrece varias aplicaciones orientadas al desarrollo y análisis de los distintos aspectos de un microsimulador. Particularmente hay dos herramientas que son útiles para el desarrollo de este modelo: Modeller y Programmer. Modeller provee las operaciones fundamentales para la implementación de modelos, simulación de tráfico y visualización del estado de la simulación, todo a través de una interfaz gráfica. Por otra parte, Programmer ofrece la posibilidad de extender las características que vienen implementadas por defecto en PARAMICS mediante la implementación de *plugins (extensiones)* desarrollados en C/C++.

Una vez que se dispone de un modelo de red, es posible modificar el comportamiento de sus componentes con un *plugin* basado en la API de Programmer. En este simulador se usó esa característica para definir el subconjunto de vehículos que representa a la flota y además para alterar a voluntad su comportamiento para que cada vehículo siga rutas controladas, las cuales podrán cambiar dinámicamente durante la simulación, debido a la aparición de nuevos requerimientos de servicio.

3.2. Modelación de la red.

En la etapa de desarrollo se trabajó con una red existente y calibrada. En PARAMICS las redes viales se modelan como grafos con costos en los arcos y en los nodos.
Las estructuras más importantes para efectos del desarrollo y funcionamiento del simulador son los nodos, arcos y vehículos. Los costos en los arcos del grafo se calculan a partir de las condiciones de tráfico, capacidad de la calzada, forma de las calles, entre otros. Los costos en los nodos representan los costos de viraje al pasar de un arco a otro. El costo de viraje depende de las condiciones de tráfico, presencia de semáforos, prioridad de paso en cada intersección, entre otros.

Desde el punto de vista de la programación misma, para intervenir con mayor facilidad sobre las distintas componentes de una red PARAMICS, existen distintas estructuras de datos que representan los distintos "objetos" presentes en la red. Algunas de estas estructuras son VEHICLE (para reprensentar un vehículo), NODE (para representar un nodo), LINK (para representar un arco), entre otras. Cada vehículo, cada nodo y cada arco en la red PARAMICS tienen asociados una única instancia de la estructura que los representa en forma inconfundifle. La idea es acceder e intervenir estas estructuras a través de las funciones ofrecidas por la API de PARAMICS, para en definitiva intervenir sobre la simulación. Esto último es lo que permite la implementación del bloque Proceso de Ruteo, explicado en la sección anterior.

Dado que los vehículos están restringidos a circular sólo los por arcos que forman la red, se formó un sistema de coordenadas basado en los identificadores de los arcos y en la distancia medida desde el nodo aguas arriba de los arcos. Si se considera que los arcos son de una dimensión, cualquier posición dentro de la red puede ser referenciada a través de este sistema de coordenadas. Una posible extensión del sistema, para representar posiciones en arcos de dos dimensiones, consiste en incorporar el número de pista en que se encuentra la posición que se quiere referenciar. Para este simulador no es necesario considerar que los arcos son de dos dimensiones, porque todas las paradas relevantes ocurren en la primera pista de la calzada.

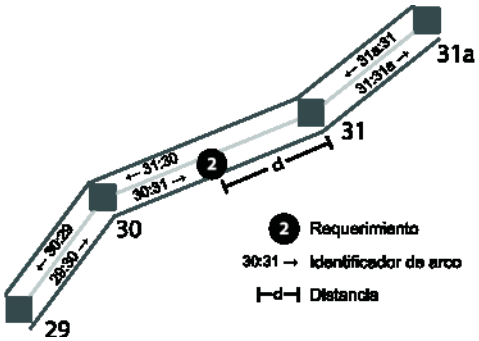

Figura 5. Ejemplo del sistema de coordenadas

3.3. Implementación de las reglas de despacho.

Para la implementación de un simulador de un sistema de despacho de vehículos se requiere que los vehículos sigan las secuencias asignadas por el despachador. En el entorno de simulación, para que un vehículo se desplace de un punto a otro por una ruta determinada, es necesario dirigir al vehículo arco a arco desde el punto de origen hasta el de destino.

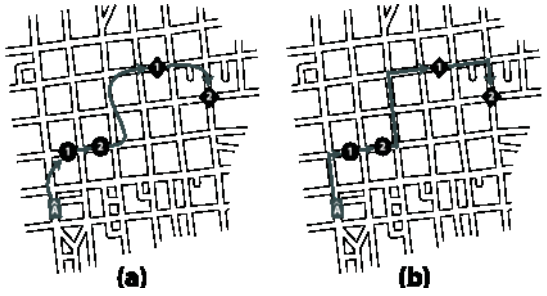

Figura 6. Ruteo a partir de lista de requerimientos.

El tipo de modelación de la red que usa PARAMICS sugiere utilizar los algoritmos existentes de rutas mínimas en grafos para generar rutas óptimas que unan los distintos puntos, definidos por la secuencia, por los que se desea que los vehículos pasen. Sin embargo, el grafo con que PARAMICS modela la red no es apto para la utilización de los algoritmos recién mencionados. Es necesario hacer un tratamiento previo al grafo antes de aplicar algoritmos de rutas mínimas sobre él.

También es necesario seleccionar un grupo de vehículos para la representación de la flota y programar las detenciones y cambios de pista, necesarios para representar de forma adecuada las detenciones que ocurren al momento de satisfacer requerimientos.

En las próximas subsecciones se explica como se lograron los objetivos recién mencionados.

3.3.1. Agregación de la red para utilización de algoritmos de rutas mínimas.

PARAMICS es un software de microsimulación, por lo que la geometría de la red se hace relevante en la modelación. El grafo que maneja PARAMICS no sólo modela la conectividad entre intersecciones, sino que además se utiliza para modelar la geometría de la red. Cada nodo tiene una posición definida e invariable en el espacio, aportando información para la modelación de la geometría. Es por esa razón que en PARAMICS, normalmente, un tramo entre un par de intersecciones no se modela sólo por un par de nodos y un arco, si no que se modela con muchos nodos y arcos. Así se puede describir la geometría de las vías con mayor detalle.

Esta característica de PARAMICS tiene ventajas en la fidelidad del modelo final, pero complica altamente la aplicación de algoritmos de rutas mínimas sobre el grafo, aumentando los tiempos de computacionales y entregando resultados poco satisfactorios. La solución que se usó para este problema fue crear una red más abstracta, que agrega todos aquellos nodos y arcos que se utilizan para modelar la geometría de la vía entre dos intersecciones en un solo par de nodos y un arco. Al aplicar este procedimiento en toda la red, se obtiene una red simplificada sobre la cual sí se pueden aplicar los algoritmos conocidos de rutas mínimas.

Además de las función “agregadora”, se desarrolló una función “desagregadora”. Al alimentar a esta última con un arco de la red agregada, podemos recuperar los nodos y arcos originales de PARAMICS que forman el arco agregado.

Una vez que se dispone de la red agregada y el par de funciones de conversión entre la red de PARAMICS y la red abstracta, se pueden utilizar sin problemas los algoritmos de rutas mínimas más conocidos. En este caso se utilizó una implementación del algoritmo de Dijkstra tradicional y un algoritmo de ruta mínima entre dos puntos adaptado de Bertsekas (1998) por Cortés (2003), que utiliza el grafo de recubrimiento mínimo de la red generado anteriormente por Dijkstra.

Se desarrolló una función para automatizar el problema de generación de rutas mínimas y las sucesivas conversiones entre las redes agregada y original, además de llamar a los algoritmos de rutas mínimas en forma explícita.

La función utiliza las estructuras VEHICLE y LINK, mediante las cuales PARAMICS referencia a los vehículos y arcos, respectivamente.

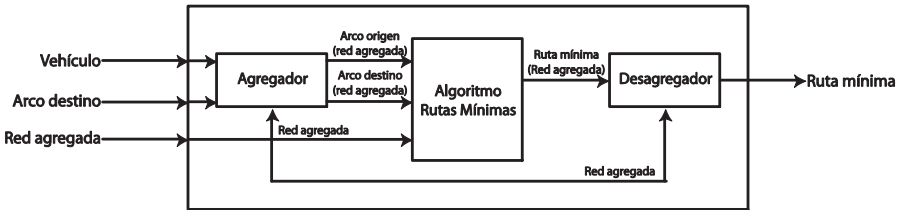

Figura 7. Diagrama de bloques del proceso de generación de rutas mínimas.

Las variables de entrada de la función son una referencia al vehículo que se desea rutear (Estructura VEHICLE), una referencia al arco en que ocurre la siguiente detención programada por la secuencia (Estructura LINK) y las estructuras de datos que definen la red agregada.

La variable de salida es una lista enlazada de LINKs. La lista consiste en una serie de referencias a arcos consecutivos entre sí que describen una ruta óptima entre la posición en que se encuentra el vehículo al momento de llamar a la función y el arco de destino.

Como paso inicial, el agregador determina la posición en que se encuentra el vehículo. Para eso se usa una la función *qpg_VHC_link* de PARAMICS. Una vez que se conoce ese arco, se considera que ese será el arco de origen. El agregador busca los equivalentes a los arcos de origen y destino en la red agregada y los entrega a la siguiente etapa del proceso, que corresponde al algoritmo de rutas mínimas.

El algoritmo de rutas mínimas calcula una ruta mínima entre los arcos de origen y destino agregados. Como resultado entrega una secuencia de arcos agregados al desagregador, que se encarga de generar la lista enlazada de LINKs que describe la ruta óptima que deber seguir el vehículo y que se usará posteriormente para realizar el ruteo.

3.3.2. Definición de la flota proveedora del servicio en el entorno PARAMICS.

PARAMICS no tiene opciones de fábrica que permitan agrupar vehículos en flotas para su tratamiento en conjunto, sin embargo ofrece la posibilidad de “marcar” los vehículos que el programador estime conveniente con un “tag”. Mediante programación se puede establecer que las funciones de ruteo sólo afecten a aquellos vehículos marcados. Es esa manera se logra crear el

efecto de la existencia de una flota controlable en PARAMICS.

La flota requiere de la programación una serie de funciones y estructuras para el manejo de la flota, la recolección de datos estadísticos y la simulación de los usuarios del sistema con sus respectivos requerimientos. Estas funciones y estructuras también se encargan de recibir las secuencias provenientes del controlador, así como generar las variables de salida del sistema a partir de datos obtenidos de la red y de modelos de predicción.

3.3.3. Ruteo mediante el seguimiento de listas enlazadas con rutas óptimas.

Una vez que se dispone una ruta válida definida por una lista enlazada de LINKs, es necesario que el vehículo que se desea rutear siga la trayectoria definida por esa ruta. Para ese fin se programó un bloque denominado "Controlador de Rutas".

La forma en que PARAMICS define las rutas que siguen los vehículos es a través de vehículos virtuales (Dummy Vehicles). De esa forma se logra que los vehículos tengan un comportamiento "realista" en el entorno de simulación.

Cada vehículo presente en la simulación tiene asociado un vehículo virtual invisible. El vehículo invisible siempre circula dos o tres arcos aguas arriba con respecto al vehículo original con el objetivo que el vehículo original imite su trayectoria. El objetivo de esta estrategia es independizar la programación de la selección de rutas de la programación de cambios de pista, desaceleraciones y cualquier otra maniobra asociada a un viraje.

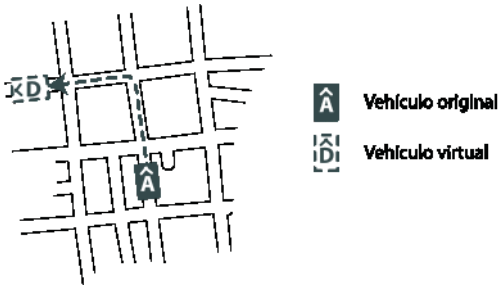

Figura 8. Ruteo mediante vehículo virtual.

La API ofrece la función "qpo_RTM_decision", que permite redefinir la lógica que PARAMICS usa para realizar el ruteo de los vehículos. La función es llamada en forma automática por PARAMICS cada vez que un vehículo virtual debe decidir un viraje. PARAMICS entrega como referencia a la función la estructura de datos que representa al vehículo virtual, además del arco en donde se encuentra éste. La idea es que el usuario reescriba la definición de dicha función para ajustarla a las necesidades de la simulación.

El tipo de retorno de esta función es un número entero, que es interpretado por PARAMICS de la siguiente forma:

Cuando el vehículo se aproxima a un nodo, dicho nodo puede tener conectado entre 1 y N arcos, además del arco por el que circula el vehículo es ese instante. Si la función "qpo_RTM_decision" retorna un entero "i" entre 1 y N, PARAMICS interpretará eso como que el próximo arco al que debe dirigirse el vehículo es el arco "i". Esto considerando que los arcos conectados al nodo aguas abajo están numerados de 1 a N en sentido inverso a las agujas del reloj, teniendo como referencia el arco en que se encuentra el vehículo.

Si se quiere que el vehículo siga la trayectoria definida por defecto por la rutina de elección de

rutas de PARAMICS, "qpo_RTM_decision" debe retornar el valor cero.

En la Figura 9 se muestra un ejemplo de decisión de rutas. En ese caso la función "qpo_RTM_decision" debe retornar el valor "2" para que el vehículo siga la trayectoria deseada, que se encuentra indicada en la figura.

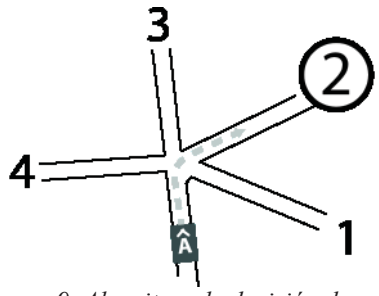

Figura 9. Algoritmo de decisión de rutas.

Se debe tener precaución al reescribir esta función, porque está pensada para redefinir la estrategia de ruteo de todos los vehículos de la red. En el contexto del simulador sólo interesa modificar el comportamiento de ruteo de los vehículos que forman parte de la flota. Los vehículos no pertenecientes a la flota se deben comportar de acuerdo a los criterios establecidos por defecto.

Cuando comienza el proceso de ruteo, ya se deben encontrar disponibles las listas enlazadas que contienen las rutas de cada uno de los vehículos de la flota. El primer paso del algoritmo es determinar si el vehículo que gatilló la función "qpo_RTM_decision" pertenece a la flota. Si el vehículo pertenece a ésta, se continúa con el algoritmo. En caso contrario, el algoritmo termina. Si se continúa con el algoritmo, se busca el identificador del arco sobre el que se encuentra el vehículo actualmente en su lista enlazada. Al encontrar el arco en la lista, basta con extraer el identificador del arco inmediatamente posterior a éste para saber a qué arco debe virar el vehículo al alcanzar el próximo nodo. Los virajes se realizan mediante el procedimiento descrito en la Figura 9.

3.3.4. Detenciones y cambios de pista.

Para modelar las detenciones y cambios de pista se usaron las funciones "qps_VHC_speed" y "qps_VHC_changelane", respectivamente.

En la Figura 10 se muestra un ejemplo de un vehículo que debe cumplir un requerimiento.

En cierto instante de tiempo (t_o) se tiene un vehículo en la posición x_o, moviéndose a velocidad v_o, y un requerimiento ubicado en x_f. El vehículo debe detenerse y realizar cambios de pista para lograr recoger al pasajero.

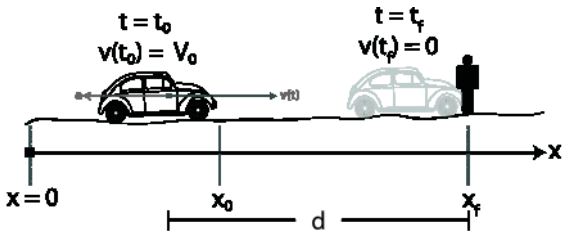

Figura 10. Ejemplo de detención.

La detención y los cambios de pista deben ser paulatinos, tal como ocurriría en la realidad. Cuando se detecta que el vehículo se encuentra a una distancia menor o igual a d de la próxima detención, comienzan las maniobras asociadas a la detención. Se decidió aproximar la maniobra de frenado como un movimiento uniformemente desacelerado discreto. La distancia de detención d debe ser suficientemente grande para evitar detenciones violentas.

Para lograr la detención en la posición x_f, a partir del instante t_o la velocidad del vehículo en cada instante de tiempo debe definirse de acuerdo a la siguiente función:

$$v_k = v_{k-1} - \frac{v_o^2}{2(x_f - x_o)}\Delta t$$

Para imponer la velocidad del vehículo, la función "qps_VHC_speed" debe actualizarse en cada instante de tiempo de acuerdo a la función anterior. Los cambios de pista comienzan en forma simultánea a la desaceleración. Para eso se debe indicar a qué pista debe moverse el vehículo mediante la función "qps_VHC_changelane".

4. PRUEBAS POR SIMULACIÓN.

Para probar el funcionamiento del simulador en lazo abierto (sin control retroalimentado), se diseñaron dos pruebas.

En la primera prueba, en el instante inicial de simulación se generan cinco solicitudes de servicio. En la Figura 11(a) se muestran las ubicaciones de origen y destino de cada cliente, así como la posición inicial del vehículo al cual fueron asignados los requerimientos. Los clientes son individualizados mediante un identificador correlativo. En la misma figura se muestra la ruta que fue asignada al vehículo inmediatamente después de la recepción de los requerimientos.

El vehículo fue capaz de circular por la red siguiendo la ruta establecida, pasando por todas las coordenadas indicadas por los requerimientos. Al alcanzar los puntos de abordaje o descenso de pasajeros, el vehículo se detuvo y, tras un intervalo de tiempo, fue capaz de acelerar para continuar con su marcha.

En la Tabla 1 se muestran los tiempos de espera y de viaje de cada cliente, obtenidos mediante la simulación.

Pasajero	Tiempo de Espera [S]	Tiempo de Viaje [S]
1	224.0	198.5
2	224.0	266.5
3	224.0	407.0
4	422.5	208.5
5	490.5	140.5

Tabla 1. Tiempos de espera y viaje, prueba número 1.

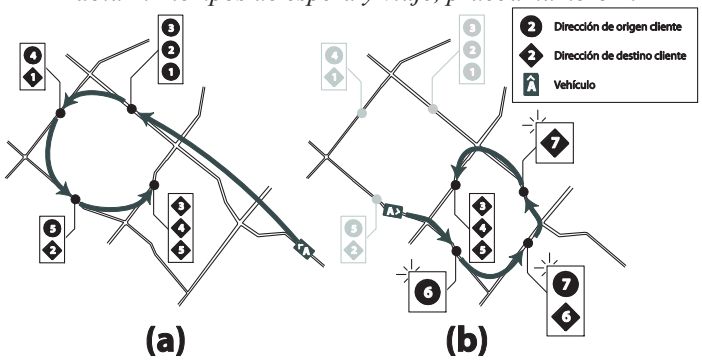

Figura 11. Pruebas realizadas.

En la segunda prueba es similar a la primera, salvo que un instante después de realizar la tercera parada, en la cual se dejó en destino al cliente 2 y se recogió al cliente 5, el vehículo recibe dos nuevas solicitudes por servicio. Los nuevos requerimientos, correspondientes a los clientes 6 y 7, obligan al vehículo a realizar tres paradas adicionales y a modificar, en forma dinámica, su ruta establecida. En la Figura 11(b) se pueden observar las ubicaciones de los requerimientos pendientes, así como la ubicación del vehículo y la nueva ruta generada en forma dinámica.

En la Tabla 2 se muestran los tiempos de espera y de viaje de cada cliente, obtenidos mediante la simulación.

Pasajero	Tiempo de Espera [S]	Tiempo de Viaje [S]
1	224.0	198.5
2	224.0	266.5
3	224.0	871.5
4	422.5	673.0
5	490.5	605.0
6	117.5	129.5
7	247.0	166.5

Tabla 2. Tiempos de espera y viaje, prueba número 2.

Al comparar las Tabla 1 y Tabla 2, se puede apreciar que la modificación de la ruta permitió incorporar al servicio a los clientes 6 y 7, sin embargo produjo un aumento en los tiempos de viaje de los clientes 3, 4 y 5.

5. CONCLUSIONES.

En este trabajo se presenta una plataforma de simulación para sistemas de despacho de vehículos con demanda de pasajeros online, sobre redes urbanas. La simulación de la operación de los vehículos y su interacción con la infraestructura es realizada con gran detalle por un paquete computacional comercial, el cual es intervenido vía API para controlar los vehículos y poder aplicar reglas de ruteo dinámico basadas en algoritmos de optimización con información en tiempo real de varias de las componentes del sistema. A su vez, se realizan pruebas en lazo abierto del simular que validan la dinámica del proceso del ruteo de vehículos.
Actualmente, se ha comenzado a trabajar en algoritmos sofisticados de optimización y reglas de ruteo dinámico específicas basadas en teoría de control predictivo híbrido, para evaluar con esta herramienta el rendimiento de tales sistemas bajo condiciones de tráfico realistas emulando una futura implementación de campo.

AGRADECIMIENTOS

Esta investigación ha sido parcialmente financiada por el Proyecto Fondecyt 1061261 y por el Instituto Milenio Sistemas Complejos de Ingeniería.

REFERENCIAS

Bertsekas, D. (1998). **Network Optimization**. Athena Scientific, Belmont, Massachussets.

Cortés, C.E., D. Sáez, A. Núñez, D. Muñoz-Carpintero (2009). Hybrid predictive control for a dynamic Pick-up and Delivery Problem, **Transportation Science** 43, 27-42.

Cortés C.E., A. Núñez, D. Sáez (2008). Hybrid Adaptive Predictive Control for a Dynamic Pickup and Delivery Problem including Traffic Congestion, **International Journal of Adaptive Control and Signal Processing** 22(2), 103-123.

Cortés, C.E., L. Pagès, R. Jayakrishnan (2005). Microsimulation of Flexible Transit System Designs in Realistic Urban Networks, **Transportation Research Record**, 1923, 153-163.

Cortés C.E. (2003) Ph D. Dissertation. **High-Coverage Point-to-Point Transit (HCPPT): A new design concept and simulation – evaluation of operational schemes for future technological deployment**. Henry Samuely School of Engineering and Institute of Transportation Studies. University of California at Irvine.

Duncan, G.I. (1995). PARAMICS wide area microscopic simulation of ATT and traffic management, **Proceedings of the 28th International Symposium on Automotive Technology and Automation (ISATA)**, Stuttgart, Germany, 475-484.

Jayakrishnan, R., C.E. Cortés, R. Lavanya, LaiaPagès (2003), Simulation of Urban Transportation Networks with Multiple Vehicle Classes and Services: Classifications, Functional Requirements and General-Purpose Modeling Schemes. **Proceedings of the 82th Transportation Research Board Annual Meeting**, Washington D.C, January 2003.

Jayakrishnan, R., H. Mahmassani y T-Y Hu (1994). An evaluation tool for advanced traffic information and management systems in urban networks, **Transportation Research** 2C, 129-147.

Oh, Jun-Seok, C.E. Cortés, R. Jayakrishnan and Der-Horn Lee, Microscopic Simulation with Large-Network Path Dynamics for Advanced Traffic Management and Information Systems, *Proceedings of the 6th* **International Conference on Applications of Advanced Technologies in Transportation Engineering, Singapore, June 2000**

Quadstone (2007).QuadstonePARAMICS V6.3. Modeller User Guide, Quadstone Ltd.

Sáez, D., Cortés C.E., Núñez A. (2008). Hybrid Adaptive Predictive Control for the Multi-vehicle Dynamic Pickup and Delivery Problem based on Genetic Algorithms and Fuzzy Clustering, **Computers and Operations Research** 35(11), 3412-3438.