.

Departamento de Investigación Operativa
Instituto de Computación - Facultad de Ingeniería
Universidad de la República (UdelaR)
Montevideo - Uruguay

Proyecto de Grado
Ingeniería en Computación

Simulador de transporte público urbano colectivo

Autores:
Patricia Aldaz
Gastón De León

Tutores:
MSc. Antonio Mauttone
MSc. Maria Urquhart

Resumen

El transporte público urbano colectivo es un elemento esencial para una ciudad, no sólo porque ayuda al desplazamiento de personas de un punto a otro, sino que también ayuda a disminuir la contaminación, reducir la congestión de tráfico y proveer más oportunidades de trabajo. Por este motivo, su desarrollo es muy importante. Uno de los mayores desafíos de la planificación del transporte público, consiste en asegurar un sistema operacional y económicamente eficiente, adecuadamente integrado al entorno. El ciclo de vida de los sistemas de transporte es un proceso continuo de diseño, evaluación, refinamiento y rediseño. En dicho proceso se tienen que resolver ciertas problemáticas como el trazado de los recorridos, la determinación de las frecuencias y tablas de horarios, asignación de la flota y de choferes a la operación de los diferentes servicios, estacionamiento y despacho de buses en los garajes. Por tal motivo, los planificadores y operadores necesitan herramientas que sirvan de ayuda para la toma de decisiones en la planificación del transporte público.

En este proyecto se plantea modificar la herramienta igoR-tp y desarrollar un simulador. IgoR-tp (Interfaz Gráfica para la Optimización de Recorridos del Transporte Público) es una herramienta de apoyo a la toma de decisiones para la optimización de recorridos y frecuencias para transporte público, en particular, permite la construcción de un modelo de red, la definición de recorridos sobre el mismo, calcular la matriz de demanda de pasajeros e interactuar con diferentes algoritmos. En este proyecto se plantea la modificación de esta herramienta, en particular, se cambió el modelo de red utilizado para permitir una construcción más detallada de la red sobre la cual se definen los recorridos. También, se desarrolló un simulador a eventos discretos, el cual modela la interacción de los pasajeros con los buses, dado un diseño del sistema de transporte público, un escenario particular de demanda de pasajeros y una determinada configuración de líneas y frecuencias. Más precisamente, permite modelar diferentes comportamientos de los pasajeros en cuanto a la elección de líneas. El simulador presenta resultados de la simulación, visualizando gráficamente su comportamiento y generando reportes. Esto permite al usuario que pueda utilizarlos como apoyo a la toma de decisiones.

Ambas herramientas se integraron de forma de que el simulador sea tratado como un algoritmo más para IgoR-tp. De esta forma el simulador podrá ser ejecutado desde igoR-tp tomando ciertos datos de entrada generados por este último.

Además, se construyó un caso de estudio referente a la ciudad de Rivera, en el cual se utilizaron las dos herramientas con el objetivo de validarlas. En particular, se construyó el modelo de red y los recorridos con igoR-tp y luego se evaluó el diseño construido con el simulador. Los resultados obtenidos son coherentes con los del sistema real de transporte público de la ciudad de Rivera y con los del modelo de asignación de Baaj y Mahmassani. También se realizaron cuatro experimentos que además de validar el simulador, muestran distintos estudios que se pueden realizar y las conclusiones que se pueden obtener a partir de éstos.

Índice

1. Introducción

1.1 Transporte Público

El término transporte público refiere al transporte de tipo comercial de personas. Es un servicio de transporte urbano y suburbano de pasajeros al que se accede mediante el pago de una tarifa fijada y que se lleva a cabo con servicios regulares establecidos en recorridos, horarios y puntos de acceso determinados. En este trabajo se propone el término transporte público urbano colectivo para la denominación de tal servicio, abreviado a efectos de simplificación como transporte público [36].

En los sistemas de transporte público intervienen tres actores: usuarios, operadores y autoridades. Los usuarios son las personas que hacen uso del servicio de transporte público, utilizándolo para trasladarse de un punto a otro de la ciudad. En este trabajo se refiere a ellos como pasajeros. En cambio, los operadores son las empresas que brindan el servicio de transporte público a los pasajeros, suministrando recursos económicos como son los vehículos, el combustible, la mano de obra y el mantenimiento. En este trabajo se les denomina empresas de transporte. Las autoridades son entidades planificadoras y reguladoras, cuyo principal objetivo es asegurar el servicio social del transporte a los habitantes de una ciudad. Esta regulación se ejerce a través de organismos gubernamentales, como por ejemplo las intendencias municipales. El acto de regulación implica establecer algunos componentes del sistema, principalmente los trazados de los recorridos, valores de las frecuencias y tarifas así como la satisfacción de la demanda [36].

El sistema de transporte público es un componente básico de la estructura social, económica y física de un área urbana, por lo que su desarrollo es un tema crucial. Su principal objetivo es mejorar la movilidad, ya que ofrece otra posibilidad de transporte para aquellos usuarios que lo elijan por razones de costo, rapidez, seguridad, conveniencia, para evitar el tráfico o por razones ambientales. Además, ha sido ampliamente reconocido por contribuir a la reducción de la polución del aire, ayudar a conservar la energía, reducir la congestión de tráfico, incrementar la productividad y proveer más oportunidades de trabajo.

El proceso de planificación de un sistema de transporte público se divide en 3 instancias según [14]:

Planificación estratégica: En esta instancia se toman las decisiones a largo plazo como es la determinación de los trazados de los recorridos. El objetivo de los problemas estratégicos es maximizar la calidad del servicio bajo restricciones presupuestarias.

Planificación Táctica: En esta instancia se toman las decisiones relacionadas con el servicio ofrecido a los pasajeros. Se asignan las frecuencias a los recorridos y también se determinan las tablas de horarios. Generalmente, estos problemas son resueltos por temporada, actualizándose ocasionalmente y al igual que en la planificación estratégica, también se enfocan en la calidad del servicio.

Planificación Operacional: se refieren a como las operaciones deben llevarse a cabo para ofrecer el servicio propuesto a un mínimo costo. Incluyen una gran variedad de problemas como asignación de la flota, asignación de choferes a la operación de los diferentes servicios, estacionamiento y despacho de buses en los garajes, y programación de mantenimiento. Los problemas operacionales son resueltos en varios intervalos de tiempo en un rango que va desde una vez por mes para la asignación de choferes a una vez por día para el estacionamiento y despacho de ómnibus en los garajes. En contraste a los objetivos de los problemas previos, el objetivo para los problemas operacionales es claramente el de minimizar el costo de operación.

Las entidades planificadoras y reguladoras tienen como misión y visión, proveer un servicio de buena calidad ajustando adecuadamente el sistema de transporte público, de forma de maximizar la calidad del servicio mientras minimizan los costos. Desde el punto de vista de los pasajeros, el sistema debe satisfacer la demanda proporcionando un servicio directo y barato. Desde el punto de vista de las empresas de transporte, el objetivo es maximizar las ganancias.

El mayor desafío en el planeamiento del transporte público urbano, es encontrar un equilibrio entre estos puntos de vista ya que los objetivos de los pasajeros y de las empresas de transporte son contrapuestos, destacando que las decisiones de los pasajeros impactan en los objetivos de las empresas de transporte y viceversa. Más específicamente, para definir los recorridos con sus frecuencias, los operadores necesitan conocer la demanda de los pasajeros. Y a su vez, la decisión de los pasajeros sobre que recorridos tomar, está basada en la definición de los mismos y sus respectivas frecuencias.

Debido a su gran complejidad, el ciclo de vida de los sistemas de transporte está constituido por un proceso continuo de diseño, evaluación, refinamiento y rediseño. Por tal motivo, los planificadores necesitan herramientas que sirvan de ayuda para la toma de decisiones en la planificación del transporte público.

1.2 Objetivos

En este proyecto se plantea la construcción de dos sistemas: igoR-tp v2.0 y un simulador. Estas herramientas tienen como objetivo ayudar a la toma de decisiones en la planificación estratégica y táctica. En particular, se plantea la construcción de un simulador a eventos discretos que modele la interacción de los pasajeros con los buses, dado un diseño del sistema de transporte público, un escenario particular de demanda de pasajeros y una determinada configuración de líneas y frecuencias. Más precisamente, deberá modelar diferentes comportamientos de los pasajeros en cuanto a la elección de líneas. El sistema deberá además presentar los resultados de una simulación, de forma que el usuario pueda utilizarlos como apoyo a la toma de decisiones.

IgoR-tp v1.0 [28] es una herramienta de planificación de recorridos y frecuencias desarrollado bajo el marco del proyecto de investigación PDT 48/02 [46] y en un Proyecto de Grado de la carrera Ingeniería en Computación [28]. En este proyecto se plantea modificar esta aplicación, de manera que soporte un modelo de red más detallado que el de la versión 1.0. Los diseños del sistema de transporte público que se construyan con igoR-tp, deben poder ser evaluados con el simulador. Por tal motivo, éste debe ser integrado a igoR-tp v2.0.

Se deben validar los dos sistemas: igoR-tp v2.0 y el simulador, con un caso de estudio real referente a la ciudad de Rivera.

1.3 Resultados y aporte del trabajo

Como resultado de este proyecto se obtuvo un software que permite simular un sistema de transporte público urbano colectivo, modelando la interacción de los distintos pasajeros con un conjunto dado de líneas de ómnibus. También se obtuvo una versión 2.0 de igoR-tp. En esta nueva versión, cambió el nivel de detalle del modelo de red utilizado, con el objetivo de soportar una visión más detallada del problema. Ambos sistemas fueron integrados debido a que el simulador toma como entrada parte del trabajo realizado en igoR-tp v2.0. Se generó la documentación técnica y de usuario correspondientes a los dos sistemas. El manual de usuario de igoR-tp v2.0 se encuentra en la ayuda de la aplicación.

También se presentó un reporte con experimentos realizados utilizando ambos sistemas, sobre el caso de estudio referente al sistema de transporte público de la ciudad de Rivera. Esto fue realizado con el fin de validar el correcto comportamiento de los dos sistemas construidos.

1.4 Estructura del informe

El informe se encuentra estructurado de la siguiente manera: en el capítulo 2 se presenta el marco teórico donde se definen los conceptos utilizados en este documento. Luego, en el capítulo 3 se encuentra una breve descripción de igoR-tp v1.0, las motivaciones que impulsaron la implementación de la versión 2.0, y se detallan las nuevas funcionalidades. La especificación del modelo de simulación implementado, se describe en el capítulo 4. En este mismo capítulo, también se encuentra el diseño del simulador, las especificaciones de los eventos, el detalle de su implementación y sus funcionalidades. El capítulo 5 muestra la construcción del caso de estudio de la ciudad de Rivera y los resultados numéricos obtenidos. Posteriormente, se presentan las conclusiones y las posibles extensiones de este trabajo, en el capítulo 6. Luego, en el capítulo 7 se encuentra la bibliografía consultada para este proyecto. Por otro lado, el informe cuenta con diferentes apéndices. En el apéndice A se encuentra el estudio realizado sobre los simuladores similares existentes. En el apéndice B se describe el estudio realizado de las bibliotecas de simulación, mientras que en el apéndice C se encuentra la documentación técnica de igoR-tp v 2.0. Luego, podemos encontrar la documentación técnica del simulador en el apéndice D y en el apéndice E se encuentra el manual de usuario del simulador. En el apéndice F se incluyen las actas de las reuniones realizadas a lo largo de todo el proyecto.

2. Marco Teórico

En este capítulo se introducen conceptos que utilizaremos durante todo el informe. Se describen aquellos conceptos relacionados con el modelo de transporte, los cuales son de vital importancia para entender el resto del documento. También se describen conceptos relacionados con Sistemas de Información Geográfica (SIG), los cuales se pueden ver aplicados tanto en la construcción de la red vial como en el modelado del grafo que representa a la misma dentro del simulador. Y finalmente, se describen definiciones relacionadas con Simulación a Eventos Discretos, lo cual es sumamente importante a la hora de poder entender el desarrollo del simulador.

2.1 Modelo de transporte

El sistema de transporte público se describe en términos de nodos, conexiones, y recorridos [48]. En esta sección definiremos los términos y notaciones utilizadas de manera que sirvan como base para la formulación del modelo utilizado y para el mayor entendimiento de los siguientes capítulos.

2.1.1 Red Vial, Nodos Viales y Tramos

La red vial en un área urbana incluye calles (por donde circulan los vehículos) e intersecciones a los que llamaremos indistintamente cruces de calles o nodos viales. La conexión entre cada par de nodos viales se denomina tramo. Estos elementos pueden ser representados de forma natural en una estructura de nodos y aristas las cuales pueden incluir costos. Una medida relevante para el costo de la arista es el tiempo requerido en atravesar la misma.

El grafo que representa la red física no es único. Hay muchas redes que pueden ser usadas para representar la misma estructura física variando el nivel de detalle. Por ejemplo, las intersecciones de las calles pueden ser representadas de una manera sencilla mediante un nodo y calles que lleguen y partan de él como muestra la figura 2.1.1.1 o de una manera más detallada como muestra la figura 2.1.1.2. En ambos casos, se representa el doble sentido de una calle de la misma manera, con dos flechas de dirección opuesta. La diferencia radica en el mayor nivel de detalle de la Figura 2.1.1.2 ya que el costo de las conexiones que mantienen la misma dirección y el costo de las conexiones que doblan, pueden ser distintos. A su vez, los costos de doblar a la derecha o a la izquierda pueden ser distintos dentro de la misma representación de la red vial, como muestra la Figura 2.1.1.2. En contraposición, en la Figura 2.1.1.1 el costo de doblar (a la izquierda o derecha) o seguir de largo es el mismo. Con esta representación tampoco se pueden representar restricciones para doblar [48].

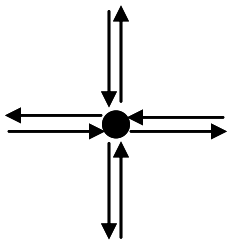


Figura 2.1.1.1 Representación sencilla del cruce de calle

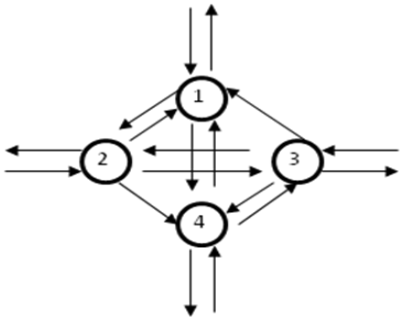


Figura 2.1.1.2 Representación detallada del cruce de calle

2.1.2 Zonas y Centroides

El proceso de planeación del transporte en el área urbana generalmente se basa en partir la ciudad en zonas de tráfico. El tamaño de cada zona puede variar (por ejemplo: una zona por cuadra o una zona para todo un barrio) y consecuentemente el número de zonas también varía.

El objetivo fundamental de la división en zonas es reconocer la posición de origen y destino de los desplazamientos entre zonas. Este elemento viene llamándose *centroide* y se define como el baricentro de una zona, donde se concentran los desplazamientos. Los centroides modelan el centro de la demanda, donde el tráfico se origina y hacia donde el tráfico es destinado. Es un nodo ficticio ya que no tiene un correspondiente geográfico con la realidad.

2.1.3 Demanda de viajes

La demanda de viajes de pasajeros se representa mediante una matriz origen-destino $D = \{d_{ij}\}$ de tamaño nxn, donde n es la cantidad de zonas y donde cada entrada d_{ij} representa la demanda de pasajeros desde la zona i hacia la zona j, expresada en personas por unidad de tiempo (1/minutos) en un determinado período de estudio [36].

2.1.4 Paradas

Las paradas de ómnibus son lugares dentro del recorrido de los ómnibus del transporte público en donde éstos se detienen para permitir el ascenso y descenso de los pasajeros.

2.1.5 Caminatas

Las caminatas modelan una conexión sobre la cual los peatones se trasladan desde el centroide a una parada (dentro de la misma zona) y viceversa, o desde una parada hacia otra (las cuales pueden estar en la misma o en distintas zonas). Sobre esta conexión interesa conocer el tiempo que le lleva al peatón ir desde un extremo al otro.

En la figura 2.1.5.1, se muestra un ejemplo de los conceptos: zonas, centroides, red, vial, cruces de calles, paradas y caminatas. Las zonas están representadas con polígonos de color celeste, la red vial con líneas de color gris, los cruces de calles con puntos de color turquesa, los centroides con puntos de color verde, las paradas con triángulos de color rosado y las caminatas entre paradas o entre parada y centroide con líneas de color violeta.

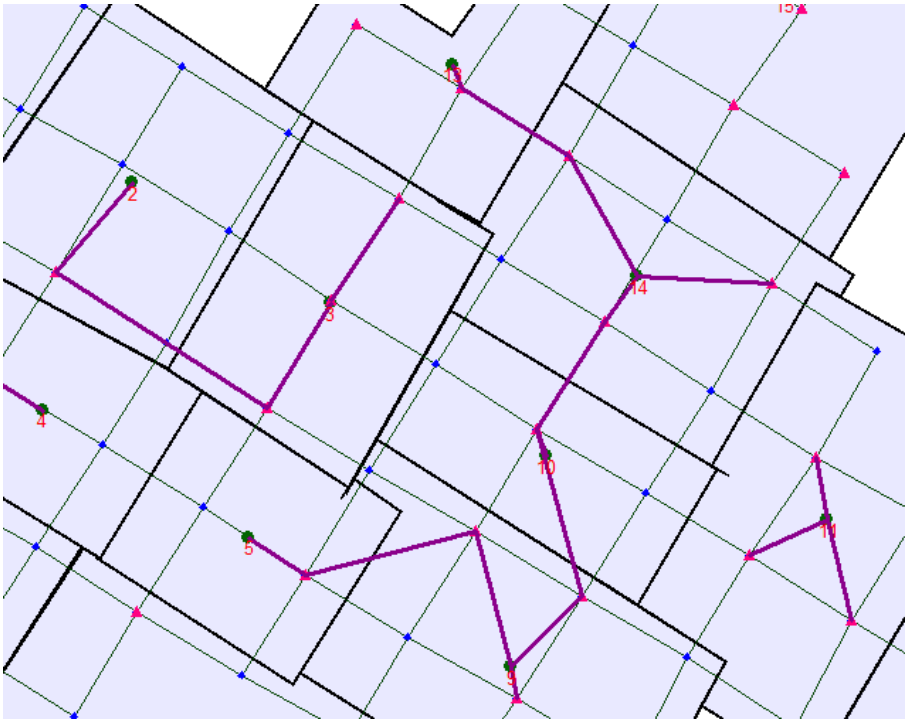


Figura 2.1.5.1: Representación de los conceptos: zonas, centroides, red, vial, cruces de calles, paradas y caminatas.

2.1.6 Líneas de ómnibus y recorridos

Las líneas de ómnibus a las cuales haremos referencia en este documento pueden estar compuestas por un recorrido circular o por dos recorridos: uno de Ida y otro de Vuelta. Los recorridos están formados por un conjunto ordenado de tramos o trayectos (conexiones entre nodos viales). En los recorridos circulares el ómnibus parte de un nodo origen, recorre una secuencia de nodos y termina en el nodo donde partió (nodo origen), como muestra la figura 2.1.6.2. En los recorridos de Ida y Vuelta, el ómnibus primero efectúa el recorrido de Ida. Para esto, parte del nodo origen de dicho recorrido, recorre una secuencia de nodos y termina en un nodo destino. Luego, efectúa el recorrido de vuelta. Para ello, se traslada al nodo origen del recorrido de vuelta, recorre una secuencia de nodos y finaliza el recorrido en el nodo destino. Un ejemplo del mismo se puede observar en la figura 2.1.6.1.

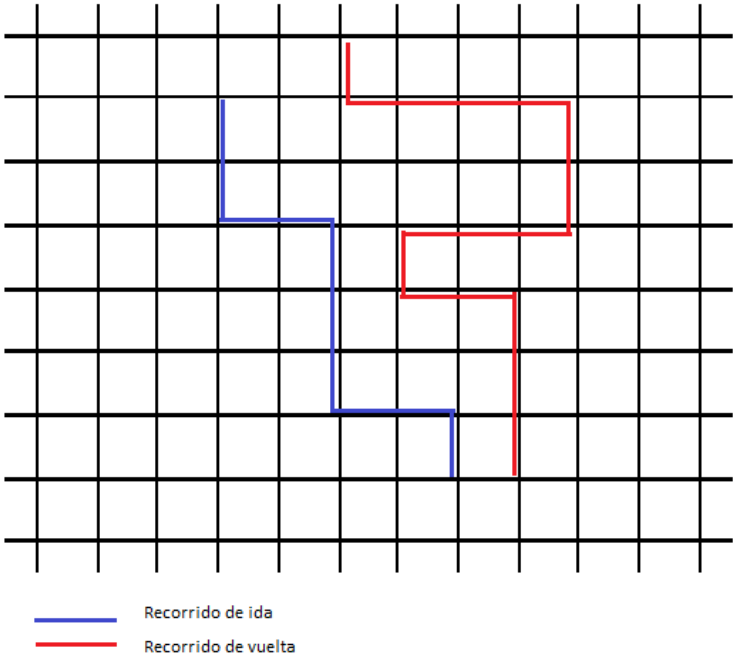


Figura 2.1.6.1

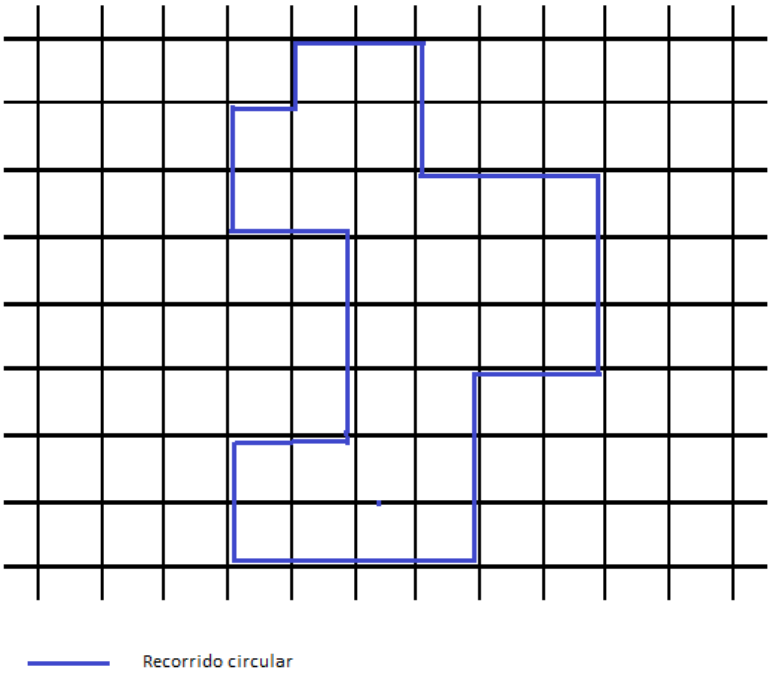


Figura 2.1.6.2

2.1.7 Asignación de buses a líneas de ómnibus

Implica asignar la flota de buses disponibles a la realización de las diferentes líneas de ómnibus.

2.1.8 Frecuencias de líneas de ómnibus

La frecuencia de una línea de ómnibus determina el intervalo de tiempo entre salidas de un bus asignado a dicha línea.

2.1.9 Tablas de horario de líneas de ómnibus

En la tabla de horarios se indica detalladamente los horarios de salida y llegada de los buses para las líneas existentes.

2.1.10 Modelo de Red de Transporte Público

La red sobre la que se definen los trazados de los recorridos se modela con un grafo G = (N;A), donde N es el conjunto de nodos y A es el conjunto de aristas conectando pares de nodos [36]. Los nodos del conjunto N pueden representar distintas entidades como ser: Cruces de calles (nodos viales), Paradas, Centroides de zonas. Las aristas del conjunto A pueden ser tramos o caminatas.

Algunas herramientas informáticas existentes que permiten representar estos modelos son por ejemplo CUBE de Citilabs [9] y EMME/2 de INRO [20]. El primero se trata de una familia de productos de software que forman un sistema de viajes, el cual brinda capacidades para la planificación de los sistemas de transporte. Asimismo, el segundo se trata de un sistema de demanda de viaje para la planificación del transporte urbano, regional y nacional.

2.2 Sistema de información geográfica

2.2.1 Definición de SIG

Un Sistema de Información Geográfica es la combinación de cinco componentes: personas especializadas, datos descriptivos y espaciales, métodos analíticos, hardware y software; organizados para analizar, manipular, procesar, almacenar, generar y visualizar todo tipo de información referenciada geográficamente. Un SIG procesa y analiza los datos convirtiéndolos en información digital que puede utilizarse para la toma de decisiones, el análisis espacial y la producción cartográfica. La implementación de un SIG permite la actualización de la información en tiempo real y la síntesis de las variables relevantes, facilitando la interpretación y comunicación de resultados. Esta tecnología permite contestar preguntas y resolver problemas mediante la observación de la información la cual se muestra fácil de comprender y compartir [31].

Los datos en un SIG representan los objetos del mundo real (por ejemplo: carreteras). Existen tres formas de almacenar estos datos: raster, tin y vectorial. En este proyecto se utiliza la representación vectorial. No sólo porque es usado por igoR-tp v1.0, sino porque el interés de las representaciones se centra en la precisión de la localización de los elementos geográficos sobre el espacio.

El formato vectorial define objetos geométricos (puntos, líneas y polígonos) mediante la codificación explícita de sus coordenadas. Los puntos se codifican en formato vectorial por un par de coordenadas en el espacio, las líneas como una sucesión de puntos conectados y los polígonos como líneas cerradas (formato orientado a objetos) o como un conjunto de líneas que constituyen las

diferentes fronteras del polígono (formato Arco/nodo) [49]. Para almacenar estos datos en formato digital, utilizamos capas o shapefiles.

La razón fundamental para utilizar un SIG es la gestión de información espacial. El sistema permite separar la información en diferentes capas temáticas y las almacena independientemente, permitiendo trabajar con ellas de manera rápida y sencilla. Además, brinda la posibilidad de relacionar la información existente a través de la topología de los objetos, con el fin de generar otra nueva que no podríamos obtener de otra forma.

2.2.2 Shapefiles

Un Shapefile es un formato vectorial de almacenamiento digital desarrollado por ESRI (*Enviromental Systems Research Institute*) [21], donde se guarda la localización de los elementos geográficos y los atributos asociados a ellos. El formato no almacena información topológica lo cual tiene ciertas ventajas como por ejemplo, ocupa menos espacio en disco. Un shapefile es generado por varios archivos. El número mínimo requerido es de tres y tienen las siguientes extensiones:

- shp: es el archivo que almacena las entidades geométricas de los objetos.
- shx: es el archivo que almacena el índice de las entidades geométricas.
- dbf: dBASE, o base de datos, es el archivo que almacena la información de los atributos de los objetos.

Además de estos tres archivos requeridos, opcionalmente se pueden utilizar otros para mejorar el funcionamiento en las operaciones de consulta a la base de datos, información sobre la proyección cartográfica, o almacenamiento de metadatos. Estos archivos son:

- .sbn y .sbx: Almacena el índice espacial de las entidades
- .fbn y .fbx : Almacena el índice espacial de las entidades para los shapefiles que son inalterables (solo lectura)
- .ain y .aih: Almacena el índice de atributo de los campos activos en una tabla o el tema de la tabla de atributos.
- .prj: Es el archivo que guarda la información referida a sistema de coordenadas.
- .shp.xml: Almacena los metadatos del shapefile.

Los tipos de shapefiles mencionados en este documento son: puntos, líneas y polígonos. Actualmente, en un shapefile se almacena el mismo tipo de figura no permitiendo combinar distintos tipos dentro de la misma capa. -

Dos herramientas informáticas existentes que utilizan shapefiles son ArcView [3] y MapWindow [35]. Particularmente, el formato shapefile fue introducido con ArcView a comienzos del año 1990.

2.3 Simulación a eventos discretos

La Simulación a Eventos Discretos (S.E.D) refiere a sistemas que pueden ser representados por una secuencia o serie de eventos. La simulación describe cada evento discreto, moviéndose de uno a otro, a medida que el tiempo transcurre [12].

Para poder comenzar a modelar este tipo de sistemas es importante poder previamente entender ciertos conceptos y terminología utilizada, conocer los diagramas de descripción y poder tener una idea del método de estructuración utilizado. Estos conceptos y terminologías que se presentan a continuación, están basados en [12] y [10].

2.3.1 Conceptos y terminologías

El modelo de S.E.D. es aquel en el que los cambios de estado del modelo ocurren en puntos discretos del tiempo (eventos). Cada punto del tiempo en que ocurre uno o más eventos se llama golpe de reloj. La medida de tiempo de la simulación se da en unidades de tiempo, apropiadas para el sistema en cuestión. La duración de la simulación es el período durante el que transcurre la simulación, medido en unidades de tiempo (independiente del tiempo que le lleva a la computadora realizarlo). Un programa de simulación comienza en tiempo 0 y ejecuta todos los eventos en el orden que deben ocurrir, avanzando de un evento a otro hasta que:

1. No hay más eventos a ejecutar.
2. El tiempo de ejecución del próximo evento supera el máximo previsto como “duración” de la simulación.
3. Se ejecutó algún evento que ponga fin a la simulación.

Se denomina “Entidad” a aquellos objetos o individuos cuyas actividades modelamos, los cuales pueden estar identificados individualmente por sus atributos. En los atributos se incluye un número (a veces llamado reloj de entidad) que indica el tiempo del próximo evento. Un evento ocurre en un punto particular del tiempo, cuando una entidad hace algo o le ocurre algo, provocando un cambio en el estado del sistema. Existen dos tipos de eventos:

- Evento fijo o agendado (B events): su ocurrencia es predecible y puede ser agendado.
- Evento condicionado o eventual (C events): su ocurrencia depende del cumplimiento de ciertas condiciones (ej. disponibilidad de ciertos recursos).

Cuando una entidad va a participar de un evento, el tiempo de ese evento (la hora en que ocurrirá) es escrito en su reloj.

Una actividad generalmente comienza con un evento condicionado y termina con un evento fijo. El número y el tiempo del evento fijo son agendados desde el evento condicionado que inició la correspondiente actividad. Una entidad ocupada está comprometida en una actividad. Una vez creada una entidad, ésta puede pasar por diferentes estados:

- Ocupada: cuando está agendada para algún evento fijo.
- En cola: en espera del turno para que alguna condición sea satisfecha. El criterio más común utilizado para seleccionar entidades de una cola es first-in-first-out (FIFO).
- Desocupada: son aquellas que se encuentran inactivas u ociosas, son las entidades que no están ni ocupadas ni en cola.

2.3.2 Diagrama de descripción

Para poder describir la vida de las distintas entidades en el sistema y su interacción con otras entidades, se utilizan diagramas de actividad. Cuando se utilizan entidades temporales, la secuencia de las entidades puede ser más lógico representarla como un flujo de actividades de principio a fin. Esta representación se denomina diagrama de flujo de actividades. En cambio si se trata de un ciclo cerrado de actividades se denomina diagrama de ciclo de actividades y se utilizan las entidades de manera permanente. Las colas son representadas mediante círculos grandes y las actividades por rectángulos. Los recursos se muestran a través de círculos pequeños. Las líneas dirigidas muestran el orden en que las entidades participan en las actividades o los recursos que se entregan. Finalmente la creación y destrucción de las entidades se representan mediante líneas en zigzag.

A modo de ejemplo proponemos un caso de un hospital. El número de camas en una sala del hospital es una restricción sobre el uso de las instalaciones hospitalarias. Los pacientes son ingresados a una sala y, después del tratamiento, son dados de alta. En la figura 2.3.2.1 se puede observar el diagrama de actividad asociado a éste ejemplo. El primer círculo representa la cola de pacientes esperando para ser atendidos, mientras que el rectángulo representa la actividad en el que el paciente entra a la sala para ser atendido.

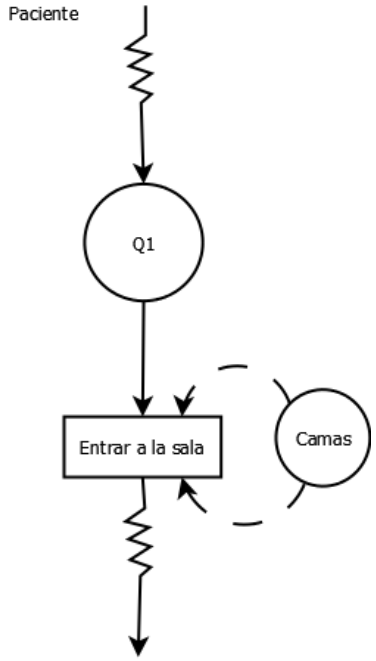


Figura 2.3.2.1

El procedimiento (programa) que se encarga de que los eventos sucedan en el orden correcto, se denomina ejecutivo, el cual también se puede denominar mecanismo de avance del tiempo.
Para eso se necesita un calendario que puede ser visto como una lista de entidades que identifican los próximos eventos a ejecutar o puede ser visto como una lista de eventos identificando las entidades participantes en esos eventos. El calendario está ordenado según los tiempos de los relojes de las entidades ("entity clock"). Para que la simulación comience debe existir por lo menos una entidad en el calendario (alimentador o inicializador).

Un paso importante a la hora de diseñar el programa de la simulación es la escritura del pseudocódigo. El pseudocódigo que se utiliza es un subconjunto del lenguaje Inglés escrito en formato rígido sin calificativos. El lenguaje es menos preciso y más transparente que un programa en Pascal pero es fácil de traducir a cualquier lenguaje de programación estructurado.

A la hora de poder recolectar resultados sobre la simulación, se utilizan Histogramas. Los mismos proveen una representación visual de la distribución de frecuencias. Se utilizan para mostrar la distribución de variables de salida durante la ejecución de la simulación, o mostrar las tendencias de ciertas variables particulares a lo largo del tiempo. Los histogramas se pueden visualizar mientras se está ejecutando la simulación o pueden ser utilizados para esbozar los resultados al final de la misma.

2.3.3 Método de estructuración

El ejecutivo se estructura según tres métodos de estructuración de la simulación o "enfoques de mundo":

- Tres fases.
- Dos fases.
- A procesos.

En el enfoque de Tres Fases, los eventos condicionados y fijos se programan como procedimientos separados.
Las tareas del ejecutivo bajo este método de estructuración son las siguientes:

1. Avanzar el reloj al tiempo del próximo evento fijo.
2. Ejecutar todos los eventos fijos que ocurren a esa hora.
3. Verificar todos los eventos condicionados y ejecutar aquellos cuyas condiciones se satisfagan.

En el enfoque de Dos fases los procedimientos que describen los eventos fijos incluyen todos los eventos condicionados que van a suceder como resultado de los propios eventos agendados o fijos.
Finalmente, en el enfoque a procesos las acciones adoptadas por cada entidad se asignan como un proceso. El proceso es una descripción del ciclo o flujo de la entidad.
El método de estructuración utilizado en este proyecto es el de Tres Fases.

A modo de ejemplo, PascalSIM [12] y EOSimulator [11] son dos bibliotecas de simulación que manejan estos enfoques. En particular PascalSIM puede ser utilizado para cualquiera de los tres enfoques, mientras que EOSimulator puede ser utilizado para un enfoque de tres fases o para uno de dos fases.

3. IgoR-tp

Debido a la complejidad de la planificación estratégica y táctica del transporte público urbano, es necesario contar con herramientas que sirvan de apoyo a la toma de decisiones para la optimización de recorridos y frecuencias. IgoR-tp v1.0 es una herramienta desarrollada por estudiantes en el marco de un Proyecto de Grado de la carrera Ingeniería en Computación [28]. El proyecto siguió las líneas de trabajo del Departamento de Investigación Operativa acerca de la aplicación de metodologías y herramientas de la Investigación Operativa a problemas de transporte. En el momento en que se desarrolló, se llevó a cabo un proyecto de investigación PDT 48/02 y un Doctorado en Informática PEDECIBA en temas de optimización de recorridos y frecuencias del transporte público.

IgoR-tp v1.0, Interfaz Gráfica para la Optimización de Recorridos del Transporte Público, toma como entrada la capa de calles de la ciudad, información de la demanda y la velocidad media de la flota. Estos datos son necesarios para permitirle al usuario construir el grafo que representa la red del transporte público, donde los pasos a seguir son: zonificar la ciudad y construir la conectividad entre las distintas zonas. También permite calcular las matrices de demanda para los distintos períodos del día, debido a la variación de la demanda que puede haber en los mismos. Sobre el grafo construido, se definen los recorridos (a los cuales se le asigna una frecuencia) que pueden ser creados por el usuario o generados por un algoritmo. La herramienta puede interactuar con algoritmos para evaluar el diseño construido. Debido a la comunicación con algoritmos externos a la aplicación, es necesario indicar los algoritmos que se pueden invocar y para cada uno de ellos, definir su ubicación (programa que lo implementa) y los parámetros de entrada.

La utilización de la herramienta se puede dividir en tres etapas: construcción de la red de transporte público, definición y evaluación de los recorridos y gestión de los algoritmos. Hay dos restricciones en cuanto al orden de ejecución de las mismas. Una restricción indica que para generar los recorridos es necesario contar con el grafo que representa la red, por lo que se debe completar esta tarea previamente. La segunda restricción hace referencia a que para evaluar recorridos o generar los mismos con algún algoritmo, es necesario haber dado de alta alguno previamente. Las tareas comprendidas en cada etapa son distintas y salvo las restricciones descritas anteriormente, las tareas son independientes entre sí. Por este motivo, se dividió el sistema en tres módulos independientes, con un componente madre que los centraliza [28]. Estos módulos son “Módulo de construcción”, “Módulo de manipulación” y “Módulo de algoritmos”. A continuación se detalla el funcionamiento de cada uno de ellos.

Módulo de Construcción:
Para crear un proyecto de construcción, se deberá contar como datos de entrada, con una capa de red vial de la ciudad la cual representa las calles, una capa de puntos de demanda que contiene los datos de la demanda de los viajes y la velocidad media de los ómnibus. Una vez creado el proyecto, se pueden definir un conjunto de zonas. Éstas se almacenan en una capa y son representadas con polígonos simples, es decir que no contienen islas ni agujeros. Además no se permiten zonas que encierren a otras zonas. Automáticamente cuando se crea una zona, se crea el centroide de la misma que es representado por un punto y es almacenado en una capa de puntos. El usuario puede crear conexiones entre dos zonas, uniendo mediante una línea, los respectivos centroides. Cada conexión tiene asociada un peso que se corresponde con el tiempo que demora un ómnibus en ir desde un extremo al otro. Este valor se calcula tomando la distancia vial entre los centroides (la cual es calculada por la herramienta utilizando una implementación del algoritmo de “caminos mínimos” de Dijkstra) y dividiéndola sobre la velocidad media del bus.

También se cuenta con una funcionalidad que permite calcular las matrices de demanda a partir de los datos de la demanda entre las zonas o lo que es lo mismo entre los centroides. Las distintas matrices representan la demanda en distintas horas del día.

Además, el módulo debe poder persistir la construcción del caso de estudio descrita brevemente en las líneas anteriores, de forma que pueda ser retomada en otro momento o que sirva como dato de entrada para la creación de un proyecto del módulo de manipulación [28].

Módulo de Manipulación:

Este módulo toma como entrada la matriz de demanda, la capa de centroides y la capa de conexiones creadas en el módulo de construcción, las cuales son utilizadas para la creación y evaluación de soluciones.

Cada solución consta de un conjunto de recorridos. Cada recorrido es un conjunto ordenado de nodos de la red (centroides) donde cada par de nodos adyacentes en el grafo, definen un tramo. Un tramo es la arista que conecta a dos centroides de las respectivas zonas. Los recorridos pueden ser no dirigidos (doble sentido: ida y vuelta sobre la misma definición del recorrido), dirigidos (un solo sentido) o circular, y tienen asociado una frecuencia que es el tiempo entre pasadas de los ómnibus de dicho recorrido. La herramienta permite crear y editar soluciones. Para cada solución permite crear, editar y eliminar recorridos y para cada recorrido, agregar tramos, quitar tramos y asignarle una frecuencia. En la figura 3.1, se muestra como se visualizan los recorridos en igoR-tp v1.0. Los centroides de las zonas están representados por círculos rosados y las conexiones entre centroides indicadas por líneas grises. Hay 5 recorridos definidos, de los cuales hay sólo dos visibles representados con líneas de color verde y azul.

Luego de definir todos los recorridos que conforman la solución, ésta se puede evaluar utilizando un algoritmo que haya sido previamente dado de alta en el sistema con el módulo de algoritmos. El módulo de manipulación invoca al algoritmo y luego permite visualizar la evaluación de la solución proporcionada, mostrando los valores de los parámetros de salida del algoritmo en caso de que haya alguno y un histograma de perfil de cargas que muestra, para los tramos de los recorridos, la ocupación de los ómnibus.

También se pueden crear soluciones a partir de un algoritmo que genere recorridos, las cuales pueden ser posteriormente evaluadas como se mencionó anteriormente.

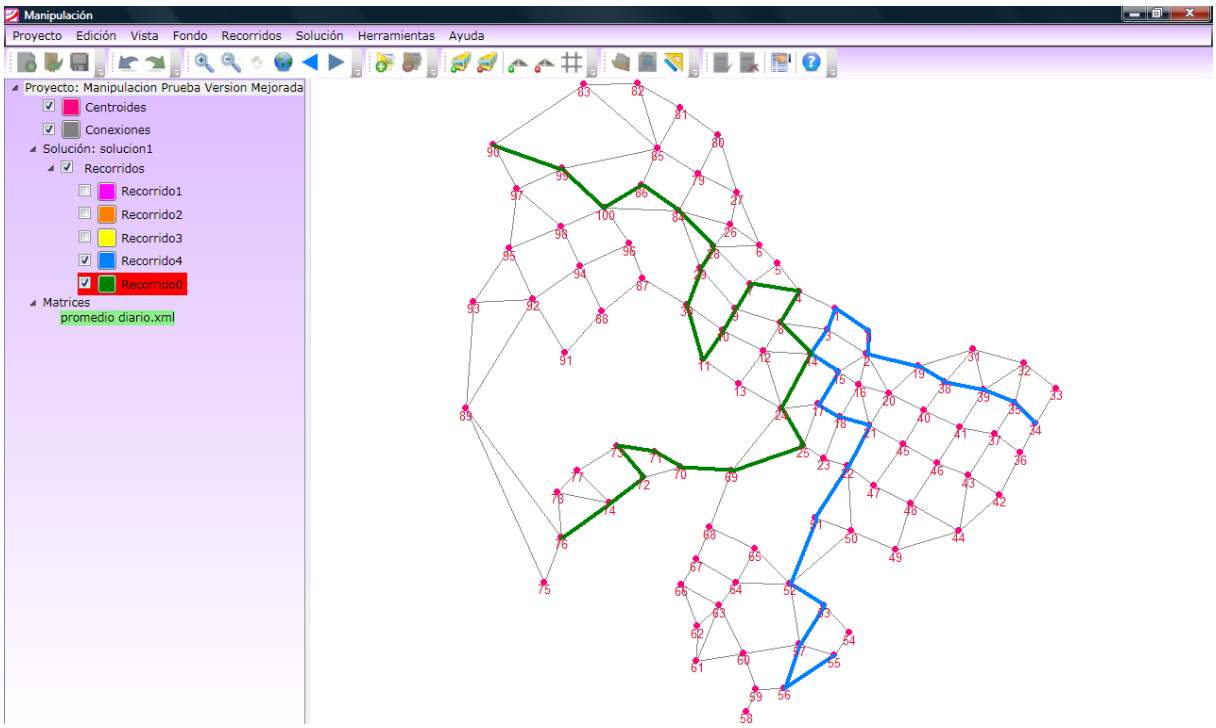


Figura 3.1. La imagen muestra como se representan los recorridos en IgoR-tp v1.0.

Módulo de Algoritmos:
En este módulo se pueden dar de alta o baja algoritmos, creando o eliminando archivos de configuración para los algoritmos respectivamente.

Los archivos de configuración almacenan su nombre, la ubicación del algoritmo, la cantidad de parámetros que recibe el algoritmo y la información de los mismos (nombre de los parámetros, tipo: caracteres, entero, real, y observación).
Estos archivos son utilizados en el módulo de manipulación para invocar a los algoritmos ya sea para generar recorridos o para evaluar soluciones.

3.1. Motivación para extender IgoR-tp v1.0

En igoR-tp v1.0, el modelo de red está representado por un grafo, en donde los nodos corresponden a los centroides de las zonas y las aristas a las conexiones entre centroides de las distintas zonas. Sobre estas conexiones se definen los recorridos de las líneas de ómnibus, donde cada recorrido está compuesto por tramos, siendo éstos la conexión entre dos centroides.

Una de las principales motivaciones para cambiar el modelo de red, fue poder representar el comportamiento de los pasajeros, no sólo entre zonas, sino que también dentro de las mismas. Más precisamente, poder modelar la elección de la parada origen y destino y tener una representación más detallada del tiempo que le insume a los pasajeros desplazarse por la ciudad y esperar en las paradas. También es interesante poder estudiar por parada, la cantidad de personas que acuden en determinados períodos del día.

Se decidió cambiar el modelo de red utilizado de forma que contemple la siguiente lógica: los pasajeros parten del centroide origen, caminan hacia una parada dentro de la misma zona del centroide, abordan un ómnibus y se bajan en una parada. Ésta puede ser la parada destino o una

parada intermedia. En caso de que sea la parada destino, el pasajero camina hacia el centroide destino. Pero si es una parada intermedia, el pasajero puede hacer transbordo ya sea tomándose otro ómnibus en esa misma parada o caminando hacia otra parada para tomarse otro ómnibus. Para representar esta realidad se mantienen los conceptos de zonas, centroides y red vial de igoR-tp v1.0 y se agregan tres conceptos nuevos: nodos viales, paradas y caminatas.

El grafo que representa el modelo de red utilizado en este proyecto, está formado por los siguientes tipos de vértices:

- Centroides: baricentro de la zona
- Nodos viales: cruce de calles
- Paradas: paradas de ómnibus

Y las siguientes aristas:

- Conexión vial: es la conexión entre dos nodos viales y el costo de la arista está dado por el tiempo que demora el ómnibus de ir de un nodo vial al otro.
- Caminatas: Es la conexión existente entre centroides y paradas o entre paradas. Su costo está dado por el tiempo que demora el peatón en ir de un extremo al otro.

En igoR-tp v1.0, los recorridos de los ómnibus se definen sobre las conexiones entre centroides. Esta versión es adecuada para una planificación no detallada de los recorridos, en el contexto de una planificación estratégica. En igoR-tp v2.0 los recorridos se definen sobre la red vial, indicando para cada recorrido, los tramos que lo componen. Es decir, se indica detalladamente las cuadras de la red vial por la que transita un bus asignado a un determinado recorrido. Las funcionalidades incluidas en la versión 2.0 permiten mayor nivel de detalle. La figura 3.1.1 muestra la red utilizada por la versión 1.0 de igoR-tp. Los centroides están representados con puntos de color rosado (el número indica el número de zona al cual corresponde) y las conexiones entre centroides están representadas por líneas en color gris. También, se definió el recorrido Línea Empalme desde Crespo con color turquesa. En la figura 3.1.2, se visualiza la red utilizada por igoR-tp v2.0. Los centroides están definidos con puntos de color verde, las paradas con puntos de color rosado, las caminatas con líneas violetas, los nodos viales con puntos azules, la red vial con líneas de color gris y el recorrido Línea Empalme desde Crespo con color turquesa. Observando las dos figuras (3.1.1 y 3.1.2) se puede apreciar la diferencia comentada anteriormente sobre el nivel de detalle en la definición de los recorridos. Además, en la Figura 3.1.2 se representan nuevos conceptos como nodos viales, paradas y caminatas, lo cuales enriquecen la representación del comportamiento del pasajero.

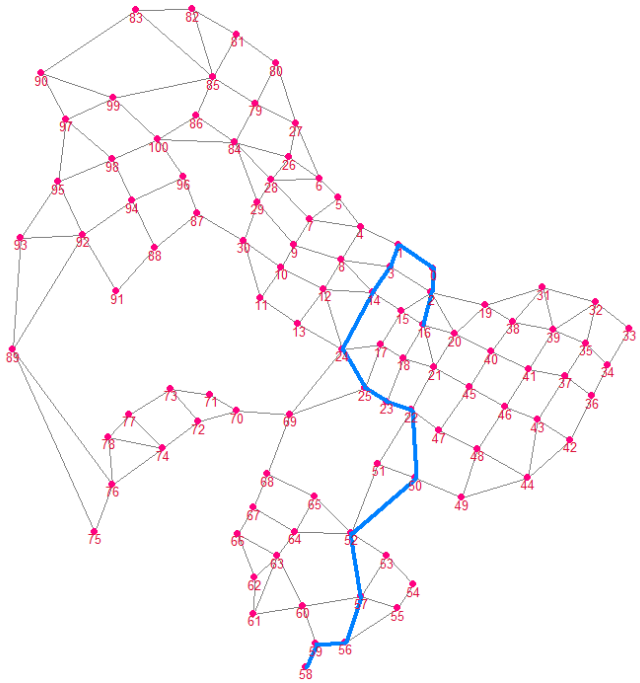


Figura 3.1.1 Red de transporte público en IgoR-tp v1.0 con el recorrido correspondiente a Línea Empalme desde Crespo visible.

Figura 3.1.2 Red de transporte público en IgoR-tp v2.0 con el recorrido correspondiente a Línea Empalme desde Crespo visible.

3.2. Requisitos Funcionales para IgoR-tp v2.0

De las reuniones realizadas con los usuarios se desprenden los siguientes requerimientos.

3.2.1. Cambios en el módulo de construcción

Introducción a los cambios en el módulo de construcción

Al igual que en la versión 1.0, para crear un proyecto de construcción, se deberá contar como datos de entrada: con la capa de red vial de la ciudad que representa las calles, una capa de puntos de demanda, la cual contiene los datos de demanda de los viajes, y la velocidad media de los ómnibus. Además, se deberán asignar valores a los parámetros: velocidad media del peatón y al tipo de distancia el cual puede ser Euclídea (longitud del segmento de recta que une los extremos de la caminata) o Manhattan (es la suma de las distancias horizontal y vertical entre los extremos de la caminata) [13]. Estos parámetros son utilizados para determinar el costo de las caminatas. En la nueva versión también se le permite al usuario agregar zonas, creándose automáticamente su centroide. Además, se quitaron las conexiones entre centroides que modelaba las conexiones entre las distintas zonas y se eliminaron las funcionalidades asociadas a la misma. Se agregaron los elementos: nodos viales, paradas y caminatas (entre dos paradas y entre un centroide y una parada de la misma zona). Los nodos viales se generan automáticamente, marcándose un nodo vial por cruce de calle. Asimismo, éstos pueden ser marcados como paradas, simplificando la localización geográfica de las mismas. Las paradas se pueden conectar con otras paradas por medio de caminatas o se pueden conectar con el centroide de la zona a la cual pertenecen. El costo de la caminata representa el tiempo promedio que le lleva al peatón ir desde un extremo de la caminata al otro.

Las diferencias en los pasos para la creación de un proyecto de construcción en las 2 versiones de igoR-tp se muestran en la tabla 3.2.1.1.

IgoR-tp v1.0	IgoR-tp v2.0
Asignar nombre de proyecto y ubicación	**Asignar nombre de proyecto y ubicación**
Seleccionar capa de puntos de demanda	**Seleccionar capa de puntos de demanda**
Seleccionar capa de red vial	**Seleccionar capa de red vial**
Ingresar el factor que convierte la unidad de la capa de calles a metros **o** **Seleccione la unidad de la capa de calles**	**Ingresar el factor que convierte la unidad de la capa de calles a metros** **o** **Seleccione la unidad de la capa de calles**
Asignar un valor a Velocidad media de la flota	**Asignar un valor a Velocidad media de la flota**
	Asignar un valor a Velocidad media del peatón
	Asignar un valor al tipo de Distancia. Los valores posibles son: Euclídea o Manhattan

Tabla 3.2.1.1. La tabla muestra para las dos versiones los pasos a seguir para crear un proyecto de construcción en su correspondiente versión.

La tabla 3.2.1.2 muestra los distintos elementos que maneja cada versión.

IgoR-tp v1.0	IgoR-tp v2.0
Centroides	**Centroides**
Puntos de Demanda	**Puntos de Demanda**
Red Vial	**Red Vial**
Zonas	**Zonas**
Conexiones	**Nodos Viales**
	Paradas de buses
	Caminatas

Tabla 3.2.1.2. La tabla muestra los conceptos que maneja cada versión.

Descripción de los cambios realizados en el módulo de construcción

Debido a que los ómnibus no circulan más de centroide a centroide sino que circulan por la red vial, se eliminó la capa de conexiones entre centroides y las funcionalidades asociadas a la misma (editar las propiedades de la capa y deshacer-rehacer de conexiones). A su vez se modificaron las siguientes funcionalidades que involucraban a esta capa: abrir y guardar proyecto e información del mapa, la cual muestra una tabla con información (atributos y sus valores) de las capas.

Se agregó la funcionalidad “Generar Nodos Viales”. A partir de la capa de red vial, se agrega un nodo vial a la capa de nodos viales por cruce de calle. Como los nodos viales se generan a partir de la capa de red vial, es responsabilidad del usuario construir la red vial de acuerdo a sus necesidades. Esta capa se genera una única vez por proyecto, por lo que esta funcionalidad queda deshabilitada luego de generados los nodos viales. La figura 3.2.1.1 muestra la red vial en color negro mientras se están generando los nodos viales. En la figura 3.2.1.2 se muestra la red vial en color negro, y los nodos viales ya generados, representados por puntos de color rosado.

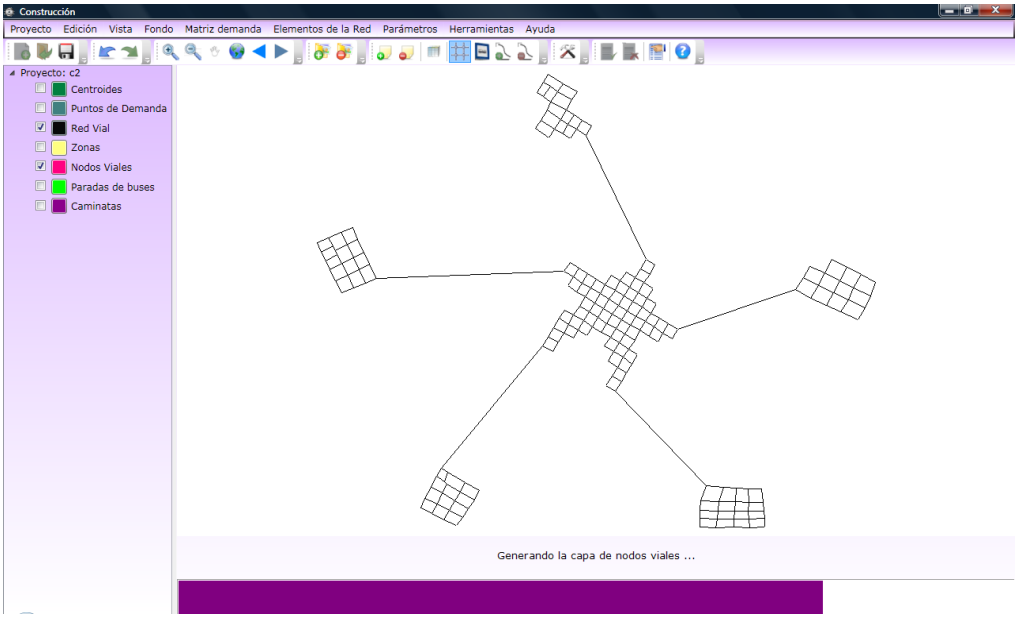


Figura 3.2.1.1: La figura muestra la capa de calles.

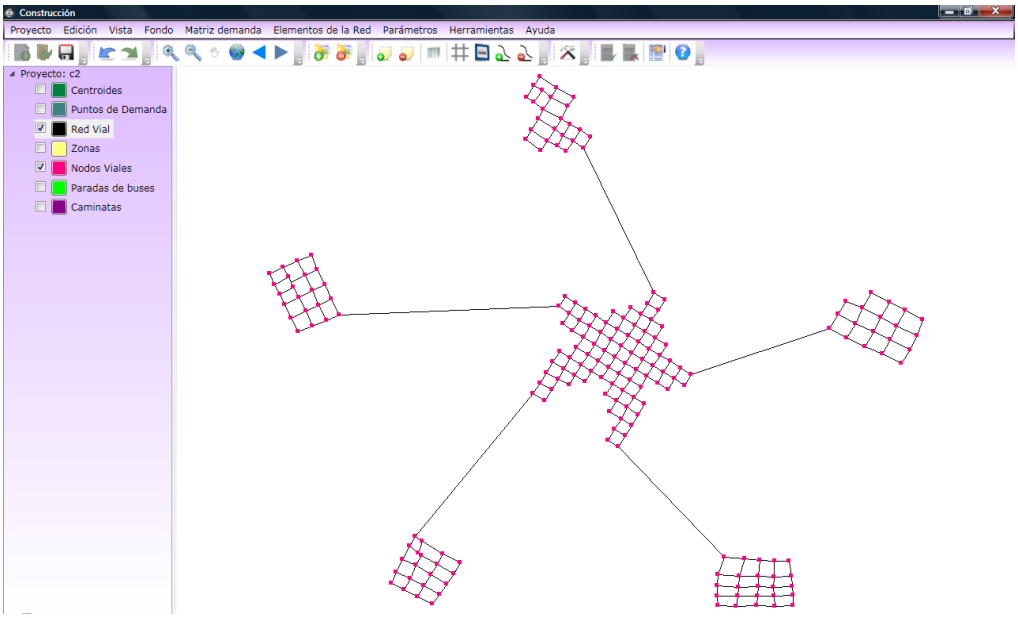


Figura 3.2.1.2: La figura muestra la red vial en negro, y los nodos viales en rosado.

Se agregó una capa de paradas. Como su nombre lo indica, representa las paradas de ómnibus, donde los pasajeros descienden de los mismos y/o esperan para abordarlos. Tomando como referencia la capa de nodos viales, el usuario de la aplicación, debe seleccionar la funcionalidad "Marcar/Desmarcar paradas" y hacer clic encima de un nodo vial. Si ese nodo vial no está marcado como parada, entonces se marca como parada y se agrega a la capa de paradas. En caso contrario, se desmarca como parada y se elimina de la capa de paradas. La figura 3.2.1.3 muestra como se representan los conceptos: red vial, nodos viales, paradas, zonas, centroides. En particular, la red vial se visualiza en color gris, las zonas en amarillo, los centroides en verde, los nodos viales en azules y las paradas en rosado.

Esta funcionalidad está deshabilitada si el proyecto no cuenta con la capa de nodos viales. Si el usuario quiere marcar paradas y no cuenta con la capa de nodos viales, debe seleccionar la funcionalidad "Generar Nodos Viales" y posteriormente se habilitará "Marcar/Desmarcar Parada".

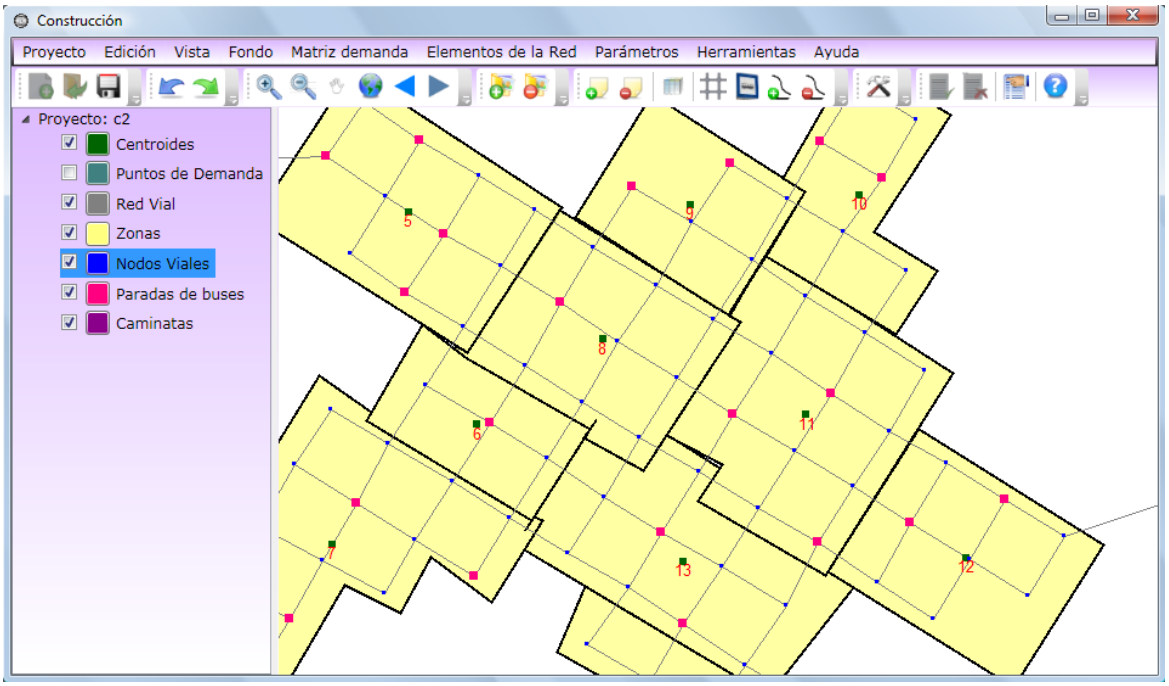


Figura 3.2.1.3: La figura muestra la red vial en gris, las zonas en amarillo, los centroides en verde, los nodos viales en azules y las paradas en rosado.

Se agregó una capa para representar las conexiones entre dos paradas o entre un centroide y una parada, a las cuales llamaremos caminatas. Para agregar una caminata, el usuario debe seleccionar la funcionalidad “Agregar Caminata” y luego partiendo desde un nodo, dibujar una línea hacia el otro nodo. Sólo se permiten caminatas de los tipos mencionados anteriormente. O sea, no se permiten caminatas entre centroides, ni donde uno de los extremos sea un nodo vial, y tampoco se permiten caminatas donde los nodos extremos sean el mismo nodo. Tampoco se permiten caminatas entre centroides y paradas de distintas zonas pero si entre paradas que pertenezcan a distintas zonas. El costo de la caminata, denominado tiempo de acceso, guarda el tiempo en minutos que le lleva al peatón ir de un extremo al otro. Este valor se obtiene dividiendo la longitud de la caminata sobre la velocidad media del peatón. La longitud de la caminata se puede hallar de dos maneras distintas, según la distancia euclídea o la distancia Manhattan. Donde la distancia euclídea entre dos puntos $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$, indica la longitud del segmento de recta que une P_1, P_2 y se define de la siguiente manera: $d(P_1, P_2) = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$. La distancia Manhattan entre dos puntos $P_1(x_1, y_1)$ y $P_2(x_2,y_2)$ es la suma de las distancias horizontal y vertical entre P_1, P_2 y se calcula de la siguiente manera: $|x_2 - x_1| + |y_2 - y_1|$ [13].
La velocidad media del peatón y el modo de hallar la distancia (euclídea o Manhattan), es asignada por el usuario teniendo la posibilidad de cambiarlas en tiempo de ejecución cuantas veces desee. Esto se realiza para que el costo de la caminata represente de manera más fiel la realidad según la zona geográfica donde se agreguen estas caminatas. Por ejemplo, en el centro de la ciudad la velocidad media del peatón puede ser más alta que en los barrios residenciales. O para una determinada zona puede representar mejor la realidad calcular la distancia Manhattan que la distancia euclídea.

Para eliminar una caminata, basta con seleccionar la funcionalidad “Eliminar Caminata” y seleccionar la conexión de tipo caminata a eliminar.

La figura 3.2.1.4 muestra la representación de la red vial en color gris, las paradas en rosado, en amarillo las zonas, en verde los centroides y en celeste las caminatas (centroide - parada, entre paradas de la misma zona y entre paradas de distintas zonas).

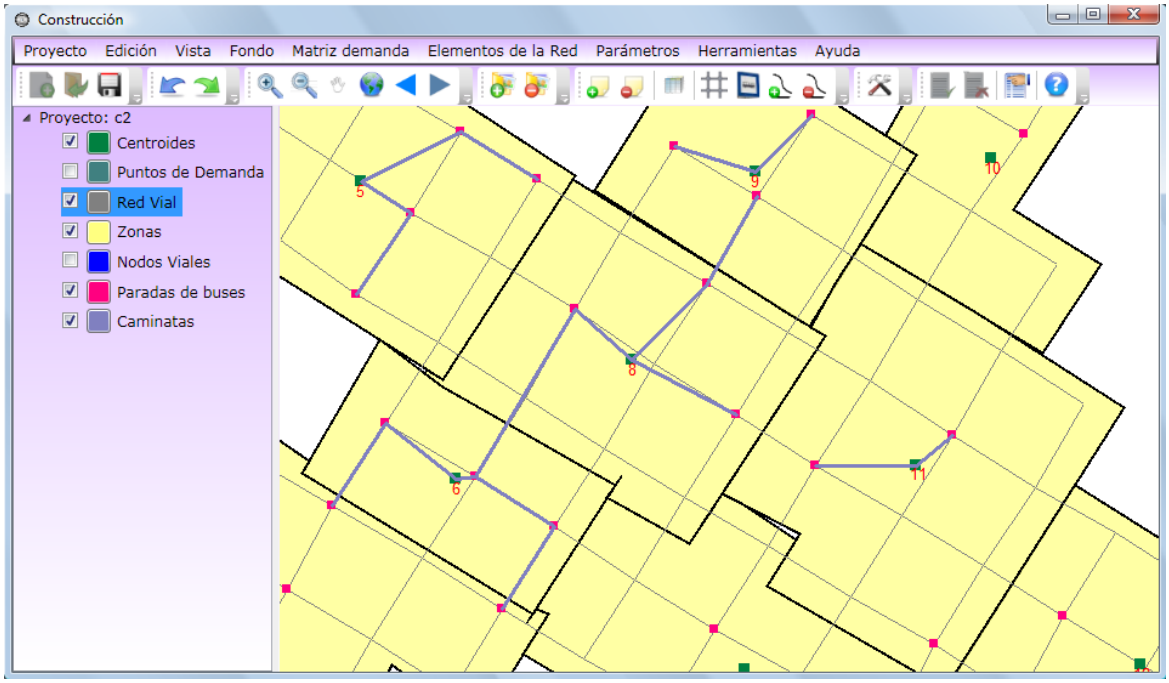


Figura 3.2.1.4: Representación de caminatas, paradas, centroides, zonas y red vial.

A la capa de red vial se le agregó un atributo que almacena la información del tiempo (en minutos) que demora el ómnibus en recorrer cada tramo (arista formada por dos nodos viales). Este valor se obtiene de dividir el largo del tramo (información proporcionada por la capa de red vial) sobre la velocidad media del bus.

Al crear un nuevo proyecto se le pide al usuario que ingrese el factor que convierte la unidad de la capa de red vial a metros o debe seleccionar cual es la unidad de dicha capa, dentro de las listadas. También debe asignar valores a la velocidad media de la flota, y a los parámetros agregados en la versión 2.0 los cuales son: velocidad media del peatón y tipo de distancia. La figura 3.2.1.5 muestra el diálogo que se despliega al iniciar un proyecto para que se asignen los valores a los parámetros mencionados anteriormente. Cuando se abre un proyecto previamente guardado, igoR-tp toma los valores que asignó el usuario para esos parámetros para la instancia guardada.

Se agregó una funcionalidad “Modificar Parámetros” para que el usuario pueda cambiar los valores de los parámetros cuando desee. En caso de que cambie el tipo de distancia y la velocidad media del peatón, sólo las nuevas caminatas que se agreguen calcularán su costo según los nuevos valores de estos parámetros. En caso de que se cambie la velocidad media del ómnibus, se recalcula el costo para todos los tramos de las calles que se encuentran en la capa de red vial. La figura 3.2.1.6 muestra el diálogo que se despliega modificar los valores de los parámetros mencionados anteriormente.

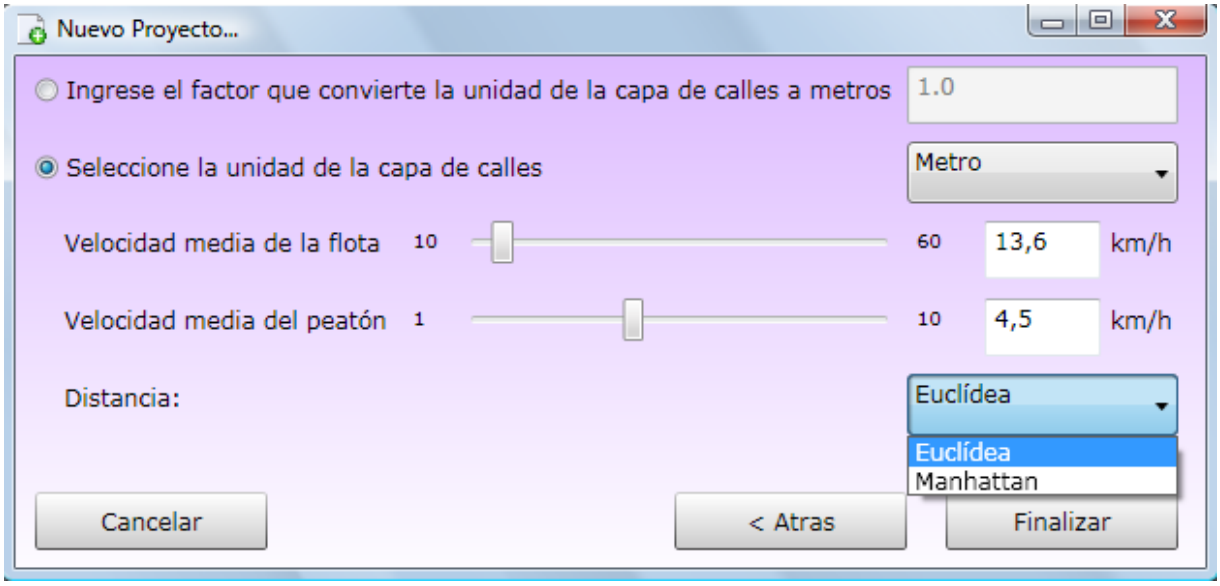


Figura 3.2.1.5: La figura muestra el cuadro que se despliega al iniciar un proyecto en el cual se deben asignar los valores para los parámetros.

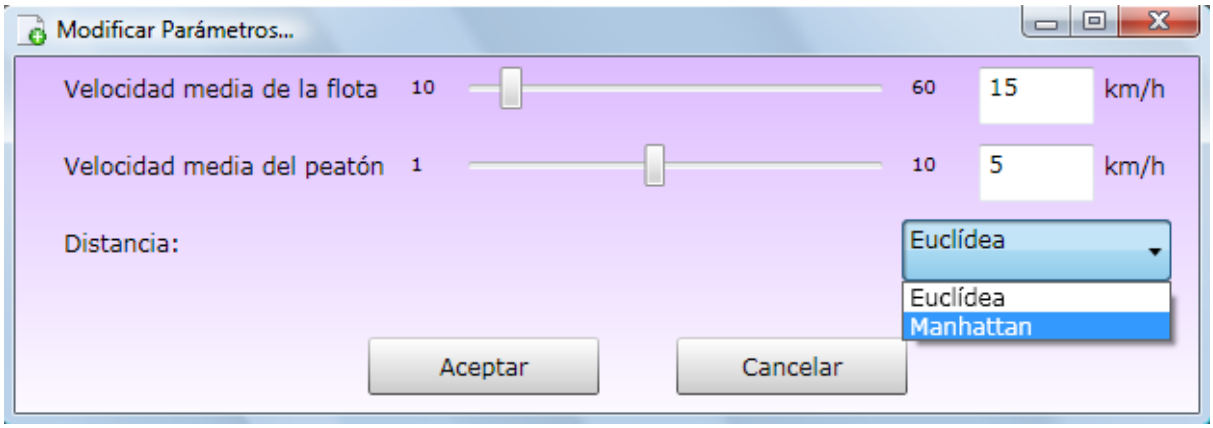


Figura 3.2.1.6: La figura muestra el cuadro que se despliega para modificar los valores de los parámetros.

Dado que se agregaron nuevas capas (Nodos viales, Paradas, Caminatas), y nuevos parámetros (velocidad media del peatón, distancia euclídea o Manhattan), se necesitó modificar “Guardar proyecto” y “Abrir Proyecto”, para que guarde y cargue la información agregada. También se permite abrir proyectos creados con la versión anterior para seguir trabajando a partir de estos. En este caso el parámetro velocidad media del peatón toma el valor por defecto 1km/h y se toma por defecto la distancia euclídea. Se recuerda que estos valores pueden ser modificados cuando el usuario desee.

Se extendió la funcionalidad “Información del mapa” la cual muestra una tabla con información (atributos y sus valores) de las capas. Se removió la información de la capa de conexiones y se implementó este requisito para las capas: nodos viales, paradas y caminatas.

Se quitó la funcionalidad “Ver camino mínimo sobre la red vial calculado para las conexiones” ya que se quitó la capa de conexiones entre centroides.

Se agregó la posibilidad de cambiarle las propiedades a las capas agregadas en esta nueva versión. Para las capas Nodos Viales y Paradas se puede cambiar: Estilo, Color y Tamaño del punto. Mientras que para la capa Caminata se puede cambiar: Estilo, Color y Ancho de la línea.

Se extendió la funcionalidad Deshacer/Rehacer para las propiedades de las capas agregadas en la nueva versión. Además se puede efectuar deshacer o rehacer de las siguientes funcionalidades: “Marcar/Desmarcar Parada”, “Agregar Caminata”, “Eliminar Caminata”.

3.2.2. Cambios en el módulo de manipulación:

Introducción a los cambios en el módulo de manipulación

Uno de los principales cambios en este módulo es el concepto de tramo. En igoR-tp v1.0 el elemento tramo hace referencia a una conexión entre dos centroides, mientras que en igoR-tp v2.0 el tramo es una conexión entre dos nodos viales. Recordamos que los recorridos están compuestos por una secuencia ordenada de tramos. Como consecuencia de lo anterior, los recorridos en esta nueva versión están definidos sobre la red vial y no sobre las conexiones entre centroides. Este cambio implicó quitar la capa de conexiones entre centroides, agregar la capa de nodos viales y red vial. Al igual que en la versión 1.0, se le permite al usuario crear soluciones (las cuales también pueden ser generadas por algún algoritmo) y evaluarlas. Otro cambio muy importante, fue la incorporación de las capas de paradas y caminatas. De esta manera con las capas de nodos viales, red vial, centroides, paradas y caminatas obtenemos toda la información correspondiente al modelo de red utilizado en este proyecto.

Las diferencias en los pasos para la creación de un proyecto de manipulación en las 2 versiones de igoR-tp se muestran en la tabla 3.2.2.1.

IgoR-tp v1.0	IgoR-tp v2.0
Asignar nombre de proyecto y ubicación	**Asignar nombre de proyecto y ubicación**
Seleccionar capa de centroides, conexiones y la ruta de la carpeta donde se encuentran las matrices de demanda	**Seleccionar capa de nodos viales, red vial y la ruta de la carpeta donde se encuentran las matrices de demanda**
	Seleccionar capa de paradas, centroides y caminatas.

Tabla 3.2.2.1. La tabla muestra los pasos a seguir para crear un proyecto de manipulación en las dos versiones de igoR-tp.

La tabla 3.2.1.2 muestra las capas que se muestran en cada versión luego de creado el proyecto de manipulación.

IgoR-tp v1.0	IgoR-tp v2.0
Centroides	**Nodos Viales**
Conexiones	**Red Vial**
	Centroides
	Paradas
	Caminatas

Tabla 3.2.2.2. La tabla muestra las capas que se visualizan en la versión 1.0 y 2.0 luego de creado el proyecto.

Descripción de los cambios realizados en el módulo de manipulación

Se removió la capa de conexiones entre centroides y las funcionalidades asociadas a la misma. Anteriormente, los recorridos de las líneas de los ómnibus se definían sobre la capa de conexiones entre centroides. En la nueva versión, éstos se definen sobre la capa de red vial.

Se modificó la funcionalidad “Nuevo Proyecto”. Éste se puede crear seleccionando las capas (Nodos Viales, Paradas, Caminatas, Red Vial, Centroides) y la ubicación de la carpeta donde se encuentran las matrices de demanda o a partir de un proyecto creado en el módulo de construcción con la versión 2.0 de igoR-tp.

Se modificaron las funcionalidades “Abrir Proyecto” y “Guardar Proyecto”, añadiéndoles la información de las capas que se agregaron a este módulo para cargarlas o guardarlas respectivamente. Estas capas son: Red Vial, Caminatas, Nodos Viales y Paradas. Al abrir un proyecto se muestra una advertencia al usuario, para informarle que si el proyecto de construcción en el cual está basado el proyecto que se desea abrir fue modificado, entonces las capas pueden estar desactualizadas.

Ya sea al crear un nuevo proyecto o al abrir uno existente, se chequea que todos los centroides tengan una caminata hacia alguna parada de su zona. De no ser así, se despliega un mensaje de error al usuario y no se le permite continuar.

Se implementó el cambio de las propiedades de las capas de puntos: Nodos Viales y Paradas, en donde las propiedades son “Estilo”, “Color”, “Tamaño”. También se implementó el cambio de las propiedades de las capas de polilíneas: Caminatas, Red Vial, en donde las propiedades son “Estilo”, “Color”, “Ancho”.

Se extendieron las funcionalidades Deshacer y Rehacer para que contemple los cambios en las propiedades de las capas: Nodos Viales, Red Vial, Paradas y Caminatas.

Se modificó el archivo donde se almacena el grafo (grafo.txt), añadiendo la información de las capas agregadas y eliminando la información de la capa de conexiones entre centroides.
El archivo grafo.txt tiene la siguiente estructura:
Id Nodo:Tipo_Nodo:coord._x:coord._y Adyacente[1]:Tipo adyacente[1]:Tipo_arista:costo-Adyacente[2]:Tipo adyacente[2]:Tipo_arista:costo …

Los algoritmos desarrolados para igoR-tp v1.0, toman como entrada el archivo grafo.txt. Es responsabilidad del usuario implementarlos nuevamente para que soporte la nueva visión del problema.

En la figura 3.2.2.1 se muestra la capa de red vial en color gris, las paradas representadas con puntos rosados, los centroides indicados con puntos verdes, las caminatas con líneas violetas y se encuentra visible un recorrido (LíneaComeriEstivaDesdeMaximilianoLuz) en color celeste.

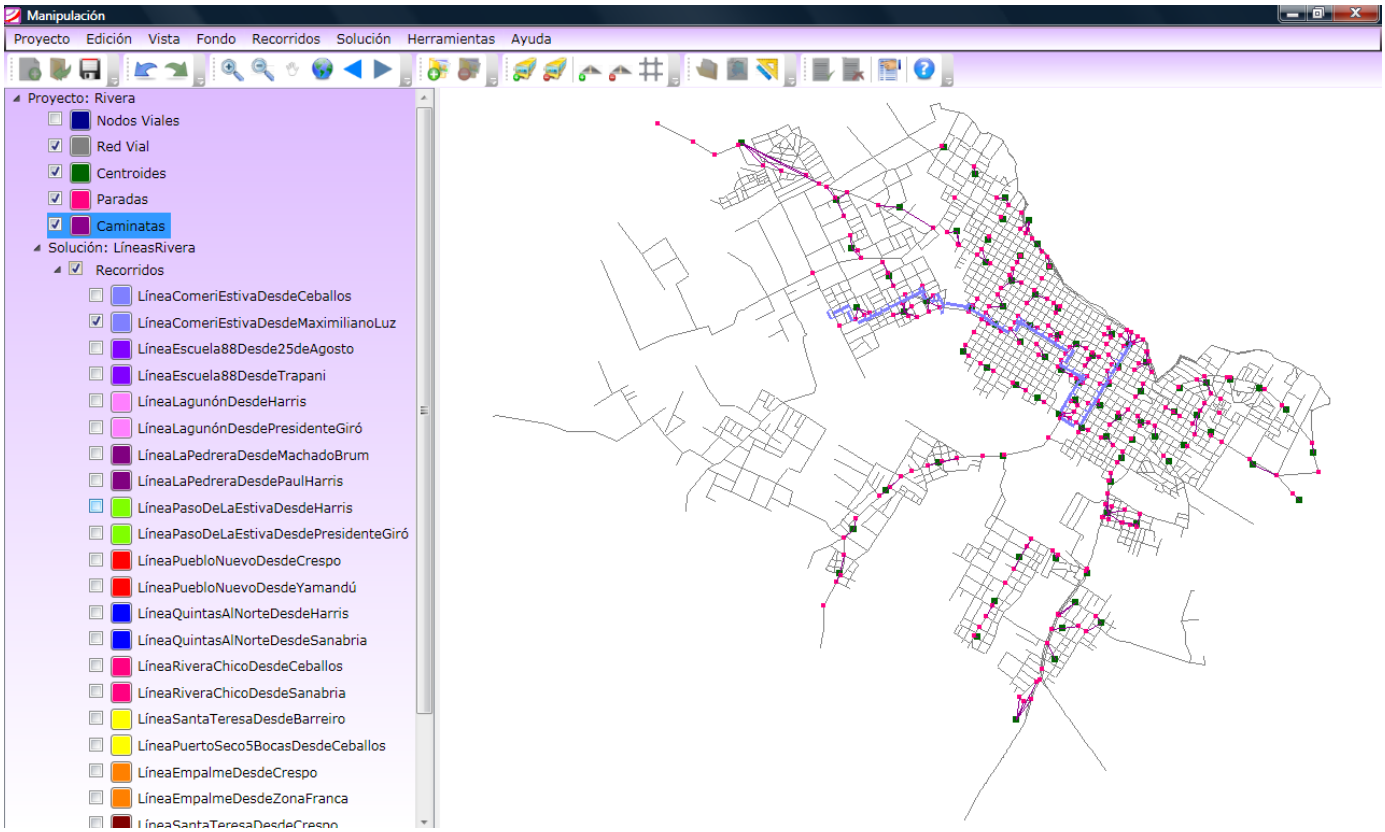


Figura 3.2.2.1: Se muestra la red vial en gris, las paradas en rosado, los centroides en verde, las caminatas en violeta y un recorrido en celeste.

3.2.3. Cambios en el módulo de algoritmos:

No fue necesario realizar ningún cambio en este módulo.

3.3. Requisitos No funcionales

Se debió respetar el diseño de IgoR-tp v 1.0.

Para implementar esta nueva versión, se tuvieron que utilizar las mismas tecnologías de desarrollo que fueron utilizadas para desarrollar igoR-tp v1.0. El lenguaje de programación es C#, con el entorno de desarrollo Visual Studio 2008 y el componente geográfico MapWinGIS. Para desarrollar las interfaces gráficas se utilizó WPF como en la versión 1.0 de igoR-tp.

3.4. Diseño

No hubo cambios de diseño con la versión 1.0. Se hace referencia a la documentación de IgoR-tp v1.0 [28] sección Diseño y Arquitectura.

4. Simulador

4.1. Introducción

Una vez modificado igoR-tp, se procedió a la construcción del simulador. El mismo modela la interacción de los pasajeros con los buses, dado un diseño del sistema de transporte público, un escenario particular de demanda de pasajeros y una determinada configuración de líneas y frecuencias. Más precisamente, modela diferentes comportamientos de los pasajeros en cuanto a la elección de líneas. Además, el simulador presenta resultados de la simulación, de forma que el usuario pueda utilizarlos como apoyo a la toma de decisiones. El mismo está concebido para ser utilizado desde igoR-tp, pero podría utilizarse de forma independiente.

En la construcción del simulador pasamos por diferentes etapas, siguiendo una determinada metodología en el desarrollo del mismo. En primer lugar nos focalizamos en la especificación del modelo y en las consideraciones tomadas, de forma de detallar el alcance del mismo. Luego pasamos a una etapa de diseño de la solución. Una vez realizado el diseño de la estructura estática del simulador, realizamos los diagramas de actividades de las entidades detectadas y en función de eso, especificamos los eventos necesarios. Posteriormente, pasamos a la etapa de implementación del simulador abarcando lógica e interfaz gráfica. El siguiente punto a resolver fue la recolección de datos y resultados. Esto involucra los histogramas generados, los reportes y aquellos datos de interés que consideramos importantes para la validación del simulador. El último punto que resolvimos fue la entrada de datos. Esto refiere a la definición del pasaje paramétrico de los datos de entrada, punto de gran importancia para resolver la comunicación con igoR-tp. En las siguientes secciones se detallan cada uno de los puntos mencionados anteriormente.

4.2. Especificación del modelo

Se desea modelar un sistema de transporte público urbano colectivo que represente la interacción de pasajeros con un conjunto dado de líneas de ómnibus, teniendo en cuenta información de la red vial y la demanda de transporte público para un escenario dado.

El sistema maneja un conjunto de líneas de ómnibus donde cada una de ellas tiene asociada una frecuencia y a su vez pueden ser de dos tipos: “Ida y Vuelta” o “Circular”. En el primer caso, la línea está conformada por un recorrido de ida y otro de vuelta. En el segundo caso, está conformada por un único recorrido. Estos recorridos están formados por una secuencia de cruces y paradas, los cuales pueden formar parte de varios recorridos. Tanto los cruces como las paradas los denominaremos nodos viales. Sobre las aristas formadas por los nodos viales, circularán ómnibus a una determinada velocidad de los que se tendrá la siguiente información: cantidad de pasajeros sentados y parados, capacidad máxima de pasajeros parados, capacidad máxima de pasajeros sentados y cantidad de puertas sólo para descender. Se considera que todos los ómnibus poseen una única puerta para ascender, la cual podrá ser utilizada para descender. Estos también contendrán un conjunto de atributos genéricos los cuales tienen un nombre y un valor. El comportamiento del ómnibus estará determinado por reglas operativas. En esta instancia se deben modelar las reglas operativas de forma de poder agregar nuevas reglas fácilmente. A modo de

ejemplo una regla operativa podría establecer que si el ómnibus va atrasado, entonces no para en la parada (a menos que haya un pasajero para descender). Asimismo, otra regla operativa podría establecer que si un ómnibus va adelantado, entonces permanece un tiempo extra en la parada. Por otro lado, todo ómnibus forma parte de una línea, por lo que cada uno de ellos va a seguir un recorrido. Si el ómnibus forma parte de una línea de "Ida Y Vuelta", entonces el ómnibus seguirá el recorrido de ida hasta llegar al último nodo vial de dicho recorrido. Allí espera un tiempo paramétrico (dependiente de la línea) para luego continuar con el recorrido de vuelta hasta su fin. Por otro lado, si el ómnibus forma parte de una línea "Circular" el ómnibus debe seguir el recorrido hasta su final. El primer nodo vial del recorrido de las líneas circulares debe coincidir con el último nodo vial de dicho recorrido.

Se debe contar con un componente que centralice el despacho de los ómnibus, que determine los recorridos que tendrán, así como también, controlar que se cumplan los horarios de salida de cada una de las líneas o en su defecto, las frecuencias de las mismas. En este último caso, si no se cuenta con un ómnibus disponible para cumplir con la frecuencia de la línea, se despachará un ómnibus nuevo.

Los pasajeros parten de su centroide origen con el fin de llegar al centroide destino según una estrategia determinada. El simulador debe tener la capacidad de agregar fácilmente nuevas estrategias y de que cada pasajero pueda elegir cual seguir. En esta instancia contaremos con una única estrategia específica, que minimiza los tiempos de viaje y de espera.

Esta estrategia se divide en 2 etapas. La primera se da en el momento del arribo del pasajero al centroide origen. En ese instante éste debe tomar la decisión de cuál parada elegir para ir a tomar el ómnibus. Para eso, elige aquella parada en la que pase la línea que le lleve menos tiempo en llevarlo a su destino (esto incluye el tiempo de caminata de la parada final al centroide destino). La segunda etapa de la estrategia se da una vez que el pasajero se encuentra en la parada esperando por el ómnibus. Éste tomará el primero que le sirva para llegar a su destino. En caso de que en el mismo instante de tiempo llegue más de un ómnibus que le sirva, tomará el que le lleve menos tiempo. Una vez que el pasajero elije el ómnibus para tomar, no cambiará de opción por más que mientras espera para subir venga otro que le sirva también. El tiempo mínimo de viaje es determinado por la sumatoria de los tiempos fijos entre nodos viales que forman el recorrido desde la parada origen a la destino, más el tiempo de caminata de las paradas a los centroides correspondientes.

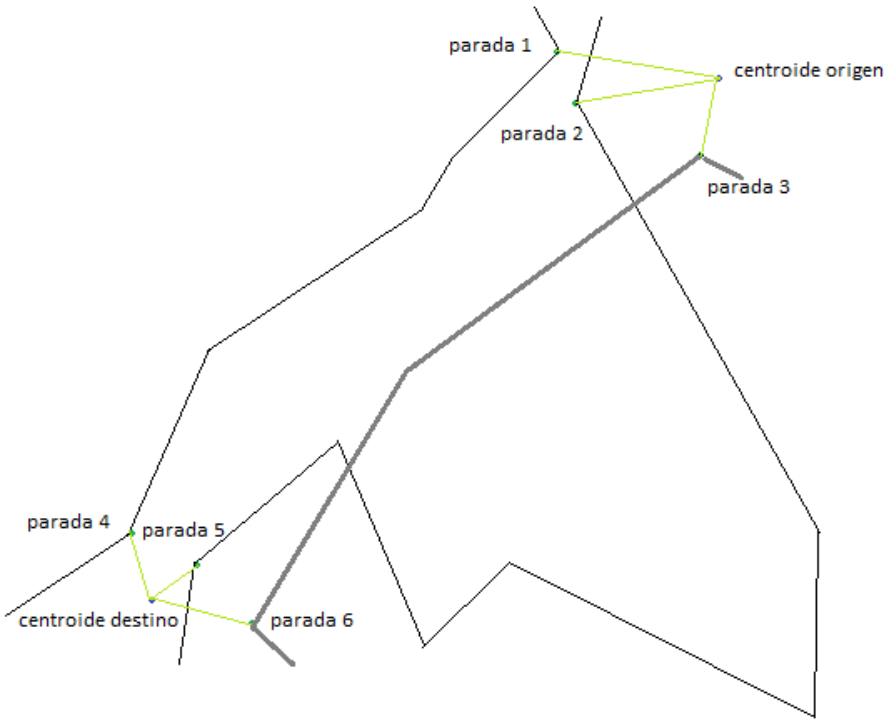


Figura 4.2.1

La Figura 4.2.1 muestra un ejemplo del comportamiento del pasajero determinado por la estrategia. Esta establece que una vez que el mismo arriba al centroide origen cumple con dos etapas:

1. Elije la parada 3 para ir a esperar al ómnibus ya que esa parada contiene la línea que le lleva menos tiempo en llegar al centroide destino.

2. Cuando arriba el ómnibus, lo toma y viaja hacia la parada 6 donde desciende y se dirige al centroide destino. En caso de existir otra línea que también pase por la parada 3 y lo lleve al centroide destino, tomará la primera que pase. Si ambos ómnibus arriban al mismo tiempo, tomará el que le lleve menos tiempo en llegar al centroide destino.

Cabe destacar que otras posibles estrategias existentes que se podrían llegar a implementar se pueden ver en [52].

El pasajero también debe contener un conjunto de atributos genéricos los cuales tienen un nombre y un valor. Estos atributos están para poder agregar de forma genérica nuevas propiedades a los pasajeros. Los mismos podrían ser utilizados como parte de una estrategia a la hora de decidir si un cierto ómnibus le sirve para tomar o no. A modo de ejemplo, un atributo genérico podría ser la edad del pasajero. La estrategia podría establecer que si se trata de un pasajero mayor de edad, éste toma el ómnibus solo si hay algún asiento libre.

En el simulador son utilizadas una serie de distribuciones de probabilidad que detallamos a continuación:

- Arribos de pasajeros: Todo pasajero parte de un centroide origen y tiene un determinado centroide destino al cual desea ir. El tiempo entre arribos de pasajeros a los centroides origen está modelado mediante una distribución exponencial negativa, cuya media es igual a la demanda (frecuencia de arribo) de los pasajeros que hay entre el centroide origen y el centroide destino. Esta demanda es dato de entrada del simulador y se detalla en la sección 4.10.

- Elección de puertas para descender: En el momento que un pasajero desea descender, éste debe tomar la decisión de que puerta elegir. Esto es modelado mediante una distribución Uniforme que toma valores entre 1 y la cantidad de puertas que posee el ómnibus. Es decir, cada ómnibus tiene asociado una distribución uniforme que determina, para cada pasajero, por cual puerta deberá bajar.

- Tiempo entre nodos viales: Las conexiones existentes entre cada par de nodo vial tienen asociado un tiempo de viaje, el cual es fijo y es un dato de entrada para el simulador. En la realidad existen ciertas variabilidades que se dan en estos tiempos de viaje, por lo tanto, lo más acorde es que el tiempo de viaje entre cada par de nodos viales sea producto de una distribución. En este sentido, se decidió utilizar una Distribución Normal, cuya media sea igual al tiempo que hay entre cada par de nodos viales y cuya desviación estándar sea igual a 1/5 de la media. En caso de dar valores negativos, se considera tiempo cero.

Esta herramienta debe permitir:

1) Evaluar tiempo de espera de los pasajeros y tiempo de viaje promedio de las distintas líneas de ómnibus.
2) Evaluar tiempo de viaje promedio de los pasajeros.
3) En caso de contar con la tabla de horarios, evaluar las líneas con retraso y el retraso total.
4) Poder contabilizar la cantidad de ómnibus necesarios para cubrir las líneas y sus frecuencias.
5) Tener la posibilidad de poder generar en cualquier momento del simulador un reporte. Dicho reporte debe contener información del estado actual de las distintas líneas de ómnibus así como también información referente a las distintas paradas.
   De las líneas interesa detallar los ómnibus que las componen, teniendo en cuenta su ocupación, el último nodo vial visitado y saber si el mismo se encuentra en viaje o esperando que la frecuencia o la tabla de horarios lo despache. Por otro lado, de las paradas interesa saber la cantidad de pasajeros que se encuentran esperando en la misma.
6) Gráficamente se debe poder observar el grafo que representa a la red de transporte público. El movimiento de los ómnibus a lo largo de la misma, identificando a cada uno de estos por un color diferente, dependiendo de la línea a la cual pertenezcan. También se debe poder ver el comportamiento de los pasajeros, desde su creación en los centroides, su caminata a la parada, su espera en la misma para tomar el ómnibus, su descenso en la parada final y la caminata al centroide destino. Se quiere tener un sector en donde se muestren datos actualizados durante la simulación del tiempo promedio de viaje y tiempo promedio de espera en las paradas de las distintas líneas de ómnibus. También se desea contar con un sector en donde se referencien funcionalidades sobre la simulación. Estas son:

- Poder mostrar y ocultar los ómnibus de las distintas líneas presionando una tecla. Ej: Al presionar la tecla “1”, se ocultan/muestran todos los ómnibus de la Línea 1.
- Poder mostrar con mayor o menor nivel de detalle la red vial. Esto es, poder ver la red vial completa o solo ver aquellas calles que formen parte del recorrido de alguna línea de ómnibus. Esto se realiza con el objetivo de darle más claridad a la simulación.
- Poder mostrar y ocultar a las paradas y centroides.
- Mostrar la referencia de que tecla presionar para generar el reporte.
- Mostrar la referencia de que tecla presionar para aumentar y disminuir la velocidad de la simulación.
- Mostrar la referencia de que tecla presionar para poder abortar la simulación.

4.3. Consideraciones

Para modelar el sistema tuvimos que tener en cuenta ciertas consideraciones:

- Si no contamos con la tabla de horarios de las diferentes líneas de ómnibus, no manejaremos franjas horarias, fines de semana, feriados, etc. En caso de utilizar frecuencias como forma de despacho de los ómnibus, las mismas se mantendrán durante toda la simulación.

- El tiempo de espera en la parada de un pasajero, se considera como el tiempo que demora esperando en la parada, hasta que llega un ómnibus que le sirva. El tiempo que demora en subir al ómnibus no se considera dentro del tiempo de espera.

- Se considera el mismo tiempo de subida al ómnibus para todos los pasajeros. Este valor se define como una constante del sistema.

- Se considera el mismo tiempo de bajada del ómnibus para todos los pasajeros. Este valor se define como constante del sistema.

- La simulación siempre comienza en el mismo estado. Es decir, si no se cuenta con una tabla de horarios, los primeros ómnibus de las distintas líneas en ser despachados, lo hacen en tiempo cero y luego siguen su respectiva frecuencia. Mientras que los primeros pasajeros en arribar a los centroides no llegan en tiempo cero, sino que llegan en el tiempo que determine la matriz de demanda.

4.4. Diseño

Para describir la estructura del sistema, sus clases, atributos, métodos y las relaciones entre ellos se realizó un diagrama de clases (Figura 4.4.1 y 4.4.2), en el cual se utilizó UML [53] como lenguaje de modelado.

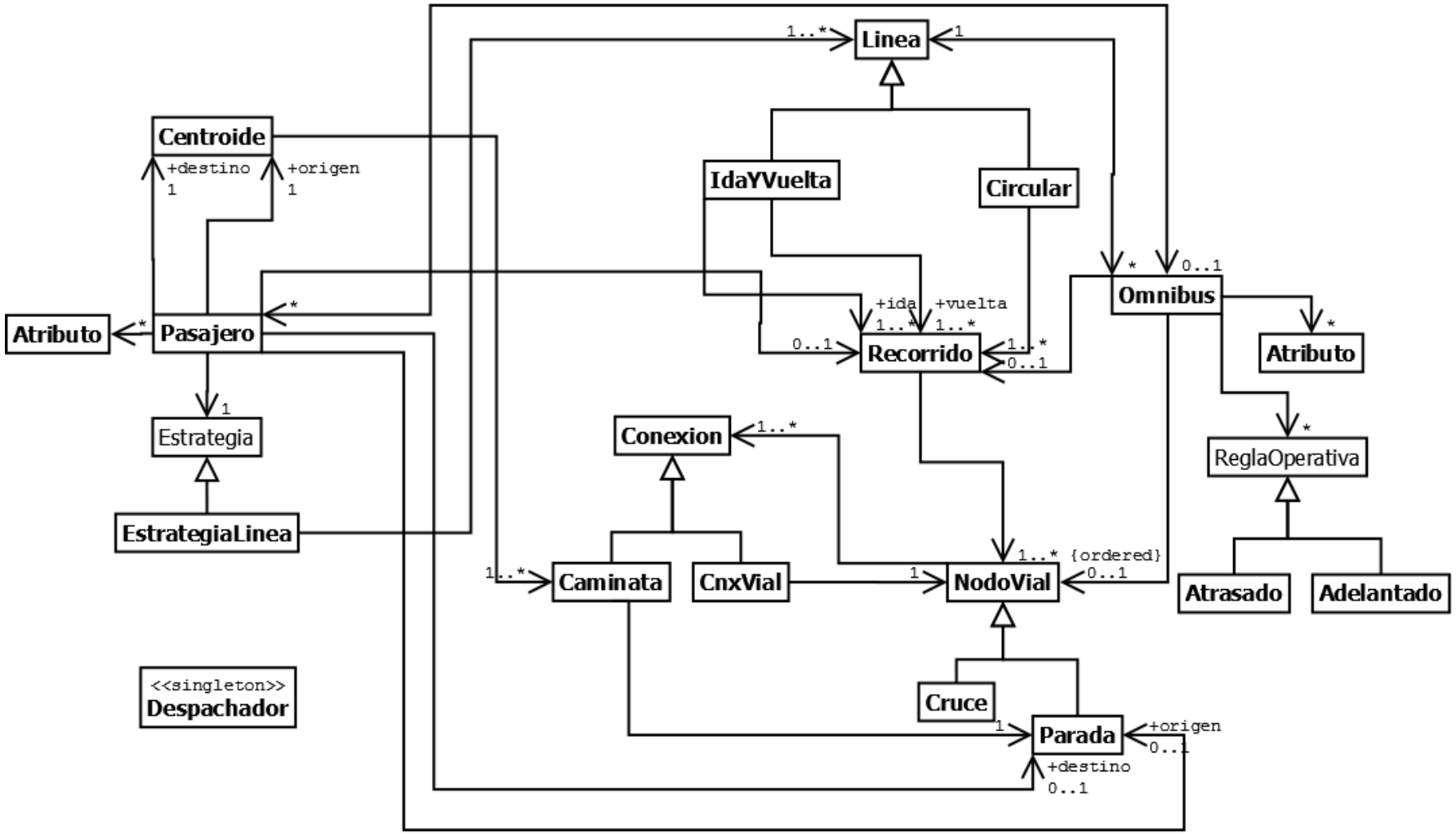

Linea
1..*
1
Centroide
+destino
1
+origen
1
IdaYVuelta
Circular
Omnibus
*
0..1
Atributo
*
Pasajero
*
+ida
1..*
+vuelta
1..*
0..1
Recorrido
1..*
0..1
Atributo
1
Estrategia
Conexion
1..*
ReglaOperativa
*
EstrategiaLinea
1..*
{ordered}
0..1
1..*
Caminata
CnxVial
1
NodoVial
Atrasado
Adelantado
<<singleton>>
Despachador
Cruce
1
Parada
+origen
0..1
+destino
0..1

Figura 4.4.1

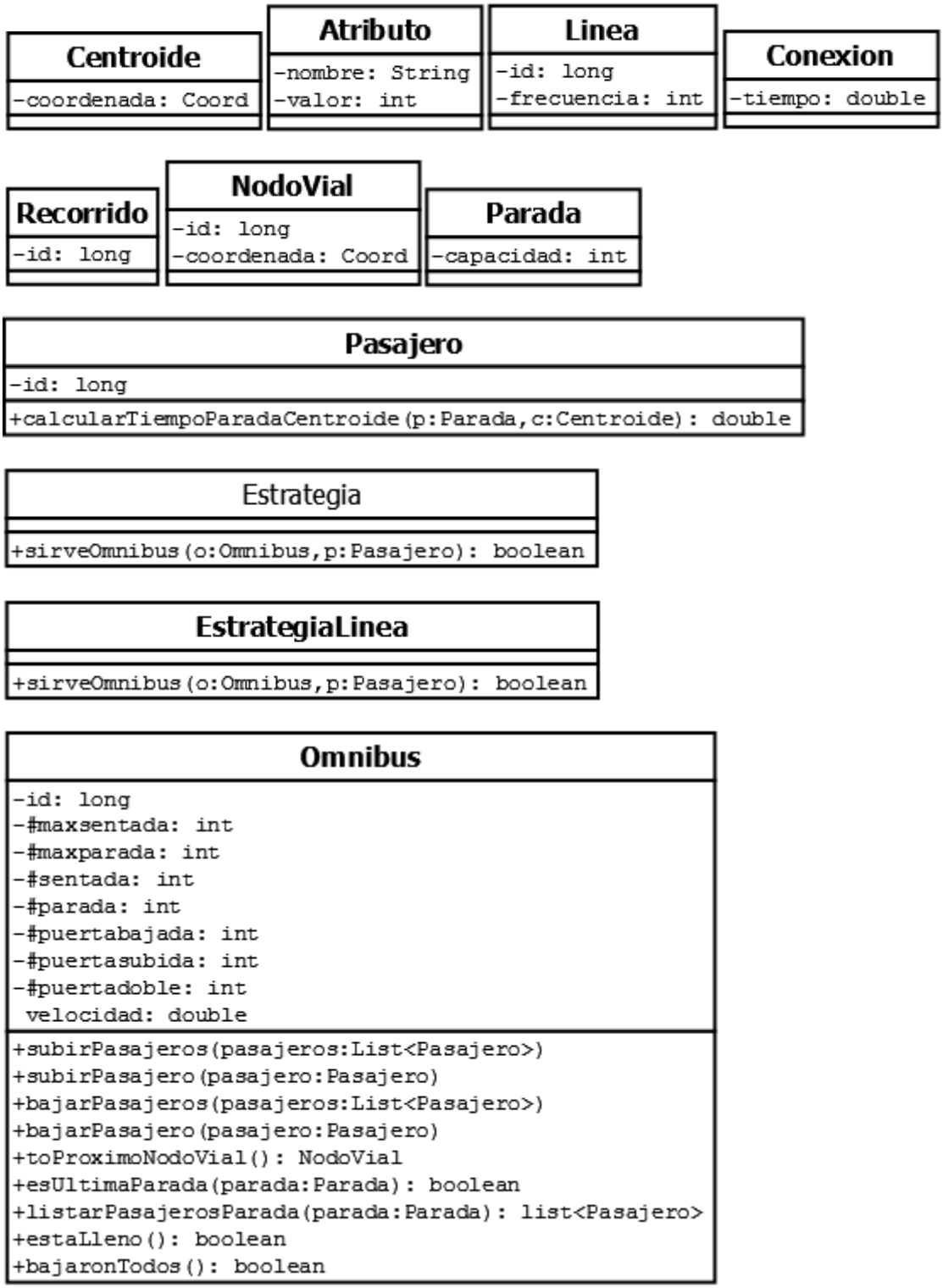

Centroide
-coordenada: Coord
Atributo
-nombre: String
-valor: int
Linea
-id: long
-frecuencia: int
Conexion
-tiempo: double
Recorrido
-id: long
NodoVial
-id: long
-coordenada: Coord
Parada
-capacidad: int
Pasajero
-id: long
+calcularTiempoParadaCentroide(p:Parada,c:Centroide): double
Estrategia
+sirveOmnibus(o:Omnibus,p:Pasajero): boolean
EstrategiaLinea
+sirveOmnibus(o:Omnibus,p:Pasajero): boolean
Omnibus
-id: long
-#maxsentada: int
-#maxparada: int
-#sentada: int
-#parada: int
-#puertabajada: int
-#puertasubida: int
-#puertadoble: int
velocidad: double
+subirPasajeros(pasajeros:List<Pasajero>)
+subirPasajero(pasajero:Pasajero)
+bajarPasajeros(pasajeros:List<Pasajero>)
+bajarPasajero(pasajero:Pasajero)
+toProximoNodoVial(): NodoVial
+esUltimaParada(parada:Parada): boolean
+listarPasajerosParada(parada:Parada): list<Pasajero>
+estaLleno(): boolean
+bajaronTodos(): boolean

Figura 4.4.2

Del diseño del simulador es interesante poder observar cómo fueron modelados los ómnibus y su comportamiento a lo largo de la red vial, cómo fueron modeladas las estrategias que pueden optar los diferentes pasajeros, como se modelaron las diferentes líneas de ómnibus y como se centralizó el despacho de los mismos.

Modelado de los ómnibus:

Cada ómnibus forma parte de una línea y tiene asociado un recorrido que debe seguir. El recorrido se forma por una secuencia ordenada de nodos viales los cuales pueden ser cruces o paradas. Para mantener el último nodo vial que el ómnibus visitó, se mantiene una asociación al mismo. De esta manera, podremos saber cuál es el siguiente nodo vial al cual debe moverse.

Estrategias de los pasajeros:
La estrategia que cada pasajero puede optar, fue modelada a través del patrón de diseño "Strategy". Este patrón permite mantener un conjunto de algoritmos de los cuales el objeto cliente puede elegir aquel que le conviene e intercambiarlo según sus necesidades [45]. En la Figura 4.4.3 se puede observar la estructura del mismo y los diferentes participantes que forman parte de este patrón de diseño. Estos son:

- **IStrategy:** declara una interfaz común para todas las variantes de uno o más algoritmos. Este rol lo cumple la clase "Estrategia" que es una interfaz común para las diferentes variantes que puden existir para el método `elegirParadaDesdeOrigen(l:list<Linea>, p:Pasajero):Parada` y para el método `elegirOmnibus(o:list<Omnibus>, p:Pasajero):Omnibus`

- **StrategyX:** implementa una variante para cada algoritmo. "EstrategiaLinea" cumple el rol de una de estas clases ya que define un posible algoritmo que sirve para decidir a qué parada origen ir (luego que se crea el pasajero en el centroide) y un posible algoritmo que sirve para decidir que ómnibus le sirve a un pasajero (cuando se encuentra esperando en la parada).

- **StrategyClient:** es el responsable de crear y mantener una referencia a una estrategia concreta. Este rol lo cumple la clase "Pasajero" ya que es él quien decide que estrategia elegir en cualquier momento.

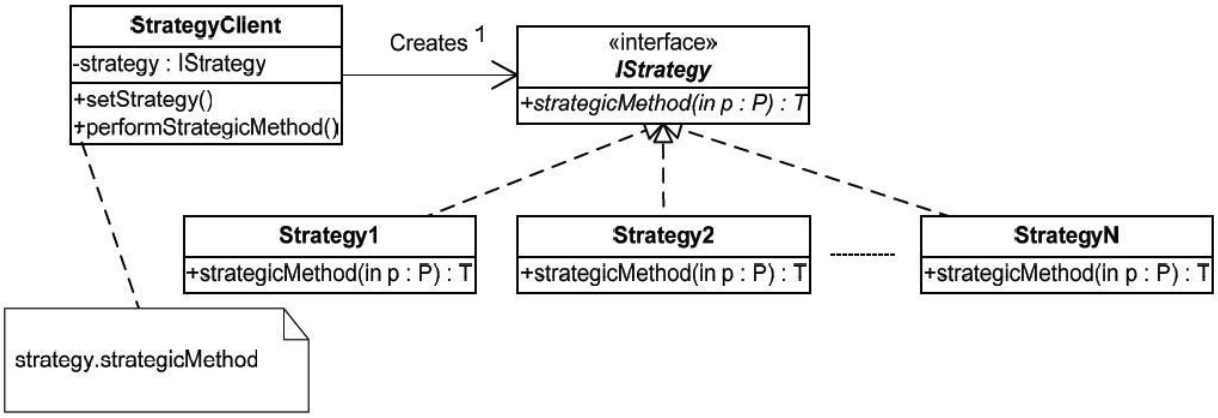


Figura 4.4.3 tomada de [45]

En esta versión del simulador contamos con una única estrategia implementada la cual denominamos "EstrategiaLinea". Ésta estrategia se divide en 2 etapas: La primera se da en el momento del arribo del pasajero al centroide origen, en la cual el método `elegirParadaDesdeOrigen` es invocado. Y la segunda se da una vez que el pasajero se encuentra en la parada esperando por el ómnibus, en la cual el método `elegirOmnibus` es invocado.

Un pasajero se crea en su centroide origen y posee un centroide destino al cual desea llegar. En ese momento el pasajero toma la decisión de cuál parada elegir para ir a tomar el ómnibus que lo lleve a su destino. Allí es invocado el método `elegirParadaDesdeOrigen`. El mismo se basa en elegir aquella parada en la que pase la línea que le lleve menos tiempo en llevarlo a su destino. Este tiempo incluye: tiempo de caminata del centroide origen a la parada origen, tiempo de viaje en el ómnibus y tiempo de caminata de la parada final al centroide final. El cálculo del tiempo de viaje es determinado por la sumatoria de los tiempos fijos entre nodos viales que forman el recorrido desde la parada origen a la parada destino. Una vez tomada esta decisión, el pasajero camina hasta la parada origen y permanece esperando por un ómnibus que le sirva. Ante el arribo de uno o más ómnibus, el pasajero debe tomar la decisión de que ómnibus escoger. Allí es invocado el método `elegirOmnibus`. Éste tomará el primero que le sirva para llegar a su destino. En caso de que en el mismo instante de tiempo llegue más de un ómnibus que le sirva, tomará el que le lleve menos tiempo. Esto incluye: tiempo de viaje y tiempo de caminata de la parada destino al centroide destino. Una vez que el pasajero elije el ómnibus para tomar, no cambiará de opción por más que mientras espera para subir venga otro que le sirva también. Luego de escogido el ómnibus, el pasajero se sube al mismo y realiza el viaje. Finalmente, cuando el ómnibus arriba a la parada destino del pasajero, éste desciende y camina al centroide final.

A continuación, se describen los pseudocódigos asociados a las implementaciones del método `elegirOmnibus(o:list<Omnibus>, p:Pasajero):Omnibus` y del método `elegirParadaDesdeOrigen(l:list<Linea>, p:Pasajero):Parada` para la subclase "EstrategiaLinea":

```
elegirParadaDesdeOrigen(l:list<Linea>, p:Pasajero):Parada
begin
   Parada paradaOrigen = "ninguna"
   foreach <parada "p" desde el centroide origen>
   begin
      foreach <línea "l">
      begin
         <calcular tiempo minimo de viaje de "l" por "p" al centroide final
+ caminata del centroide origen a "p">
         if <es el menor tiempo calculado>
         begin
            paradaOrigen = "p"
         end
      end
   end
   return paradaOrigen
end;
```

```
elegirOmnibus(o:list<Omnibus>, p:Pasajero):Parada
begin
   Omnibus omnibusViaje = "ninguno"
   foreach <ómnibus de la lista o, "omnibus">
   begin
      <calcular tiempo minimo de viaje de "omnibus" hasta parada final +
caminata de parada final al centroide destino>
      if <es el menor tiempo calculado>
      begin
         omnibusViaje = "omnibus"
      end
   end
   return omnibusViaje
end;
```

Modelado de líneas:

Como se observa en la Figura 4.4.1 las líneas se modelan como una generalización en la que la clase padre corresponde a "Linea" y las subclases corresponden a "Ida Y Vuelta" y "Circular". Las líneas de "Ida y Vuelta" se componen de dos recorridos (Clase "Recorrido"), uno que cumple el rol del recorrido de ida y otro que cumple el rol del recorrido de vuelta. Cada uno de estos recorridos está compuesto por una secuencia de nodos viales. Una vez asignado el ómnibus a una determinada línea, éste comienza su viaje en el primer nodo vial del recorrido de ida. Cuando el ómnibus visita al último nodo vial de éste recorrido, pasa al primer nodo vial del recorrido de vuelta hasta llegar al último y así completar su viaje. El tiempo que demora en cambiar de recorrido (de ida a vuelta), debe ser ingresado por parámetro y será un atributo de este tipo de línea.

Las líneas “Circular” se componen de un único recorrido (Clase “Recorrido”) el cual está compuesto por una secuencia de nodos viales que determinan la “circularidad”. Una vez asignado el ómnibus a una determinada línea, éste comienza su viaje en el primer nodo vial de la secuencia. Cuando el ómnibus visita el último nodo de dicha secuencia, el mismo finalizará su viaje. Para lograr la “circularidad” de este tipo de línea es necesario que el primer nodo vial coincida con el último nodo vial de la secuencia.

Despacho de ómnibus:

Se cuenta con una clase denominada “Despachador” que es la encargada de administrar todas las líneas que se definan sobre el simulador y asignarle a cada una de ellas los ómnibus que sean necesarios. Dado que cada línea de ómnibus tiene una frecuencia, si un ómnibus no llegó al destino y la frecuencia de la línea determina que tiene que salir nuevamente, entonces se saca otro ómnibus del despacho. Se asume que tenemos la cantidad de ómnibus que necesitamos. De todos modos, es importante llevar la cuenta de cuantos están circulando a la vez. No se tomarán más ómnibus que los disponibles y necesarios para cubrir las frecuencias de las líneas. Para el caso de las líneas de ida y vuelta, esta clase también es la encargada de “hacer esperar” un tiempo determinado entre que termina el recorrido de ida y comienza el recorrido de vuelta (este tiempo es un dato que poseen únicamente las líneas de ida y vuelta, no aplica para el caso de las líneas circulares). Si llegamos a contar con una tabla de horarios de las distintas líneas de ómnibus, es esta clase la encargada de asignarlos en los horarios que dicha tabla determine.

4.5. Diagrama de actividad

Se describe el diagrama de actividad referente a los ómnibus y pasajeros para mostrar el ciclo de vida de los mismos y la interacción entre ellos.

En la figura 4.5.1 se puede observar el ciclo de vida de los ómnibus. En el momento en que son creados, estos van al inicio de su recorrido o recorridos, dependiendo del tipo de línea al cual pertenecen. Luego de esto, comienzan el viaje recorriendo todos los nodos viales que conforman el recorrido. Si el nodo vial que visita se trata de una parada entonces espera a que bajen y suban todos los pasajeros de la misma. Posteriormente, luego de subir al último pasajero, el ómnibus continúa su viaje y repite el mismo proceso. En caso de que finalice su recorrido, el ómnibus se queda esperando para ser despachado y poder comenzar su recorrido nuevamente.

También se puede observar el ciclo de vida de los pasajeros. Una vez creados en su centroide origen, toman la decisión de que parada elegir para esperar el ómnibus, realizando su caminata a la misma. Una vez arribado a la parada, esperan hasta que un ómnibus que le sirva llegue. Allí se encola en Q1, esperando su momento para subir al ómnibus. Luego de subir, realiza su viaje hasta llegar a la parada destino. En dicha parada el pasajero procede a bajar por la puerta que eligió para descender. Finalmente cuando termina de bajar, camina a su centroide destino y se destruye.

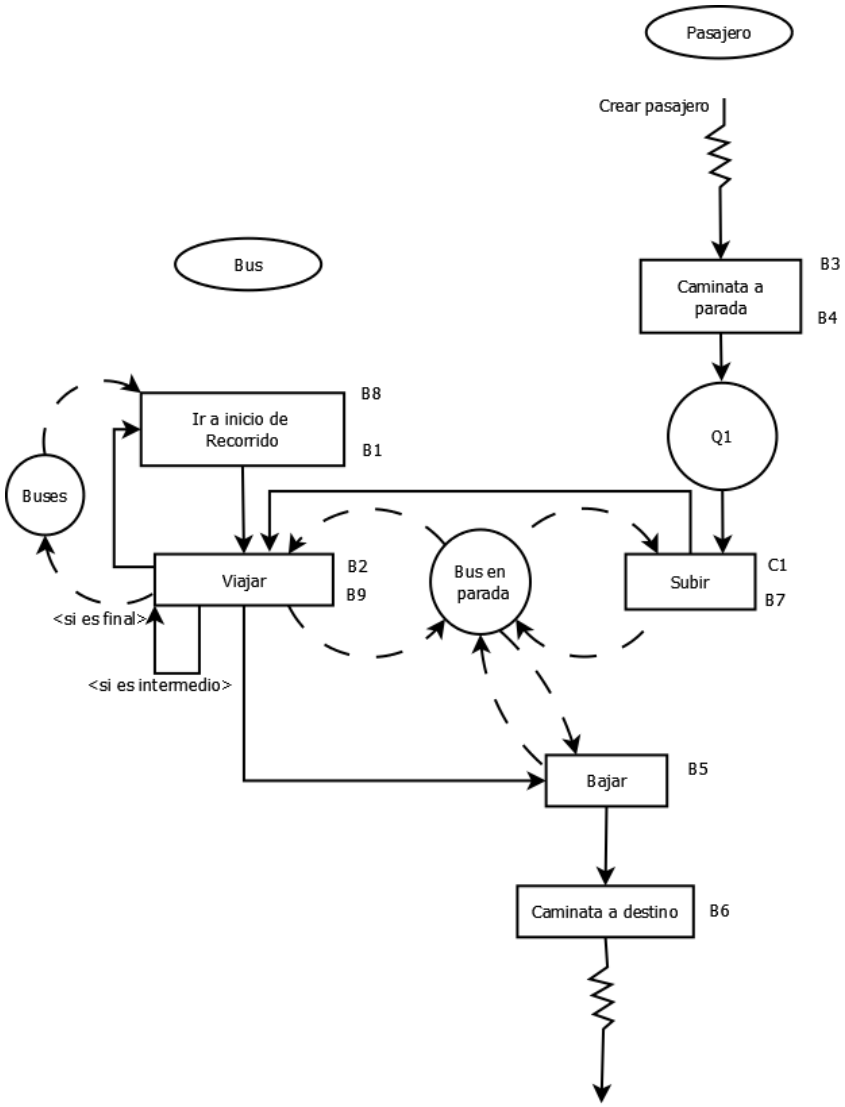

Pasajero
Crear pasajero
Bus
B3
Caminata a parada
B4
B8
Ir a inicio de Recorrido
B1
Q1
Buses
B2
Viajar
B9
Bus en parada
C1
Subir
B7
<si es final>
<si es intermedio>
B5
Bajar
Caminata a destino
B6

Figura 4.5.1

4.6. Especificación de eventos

En la construcción del simulador se especificaron los eventos existentes bajo el método de estructuración de tres fases. Por tal motivo, se definieron una serie de eventos fijos y condicionales.

A continuación, mostraremos el pseudocódigo de cada uno de ellos para poder entender de mejor manera su funcionalidad. La notación B_i corresponde al evento fijo "i", mientras que C_j corresponde al evento condicional "j". De esta forma se identifican los diferentes eventos, pudiendose observar sus correspondencias en el diagrama de actividad de la figura 4.5.1.

```
procedure comienzo-recorrido-omnibus; {B1}
begin
   <calculo tiempo del ómnibus al próximo nodo vial>
   <seteo tiempo de arribo del ómnibus al próximo nodo vial>
end;


procedure arribo-omnibus-nodo; {B2}
begin
   if <nodo actual es Parada> then
   begin
      while <hay pasajeros para bajar>
      begin
         <elegir puerta por la que baja el pasajero>
         <seteo tiempo bajada del pasajero>
      end
   end
   else
   begin
      <calculo tiempo del ómnibus al próximo nodo vial>
      <seteo tiempo de arribo del ómnibus al próximo nodo vial>
   end
end;


procedure comienzo-viaje-pasajero; {B3}
begin
   <calculo tiempo del pasajero del centroide origen a parada>
   <seteo tiempo de arribo del pasajero al la parada>
end;



procedure arribo-pasajero-parada; {B4}
begin
   <poner pasajero en cola Q1>
end;
```

```
procedure bajada-pasajero-omnibus; {B5}
begin
   <bajar pasajero del ómnibus>
   <calculo tiempo del pasajero de parada a centroide destino>
   <seteo tiempo de arribo del pasajero al centroide destino>
end;



procedure llegada-pasajero-centroide; {B6}
begin
   <elimino pasajero>
end;



procedure terminar-subida-pasajero; {B7}
begin
   <marco al pasajero que ya subió al ómnibus>
end;



procedure frecuencia-omnibus; {B8}
begin
   <consulto con el despachador si tiene un ómnibus que no esté en viaje>
   if <todos los ómnibus están en viaje> then
   begin
      <agregar uno nuevo>
   end
   else
   begin
      <pido un ómnibus al despachador>
   end
   <comienzo recorrido de omnibus>
end;

procedure cambio-linea-vuelta; {B9}
begin
   <comienzo recorrido de vuelta de ómnibus>
end;



procedure subida-pasajero-omnibus; {C1}
begin
   foreach <parada> do
   begin
      foreach <pasajero en la parada> do
      begin
         <elegir el ómnibus de la parada que le sirve al pasajero>
         if <encontró un ómnibus> then
         begin
            <sacar pasajero de cola Q1>
            <calculo tiempo que demora en subir al ómnibus>
            <seteo tiempo de subida del pasajero al ómnibus>
         end
```

```
      end
      foreach <ómnibus de la parada> do
      begin
         if <bajaron los pasajeros Y no hay pasajeros subiendo> then
         begin
            <calculo tiempo del ómnibus al próximo nodo vial>
            <seteo tiempo de arribo del ómnibus al próximo nodo vial>
         end
      end
   end
end;
```

4.7. Implementación

Con el objetivo de poder mostrar cómo se dividió el simulador en agrupaciones lógicas y también mostrar las dependencias entre las mismas, se construyó el siguiente diagrama de paquetes:

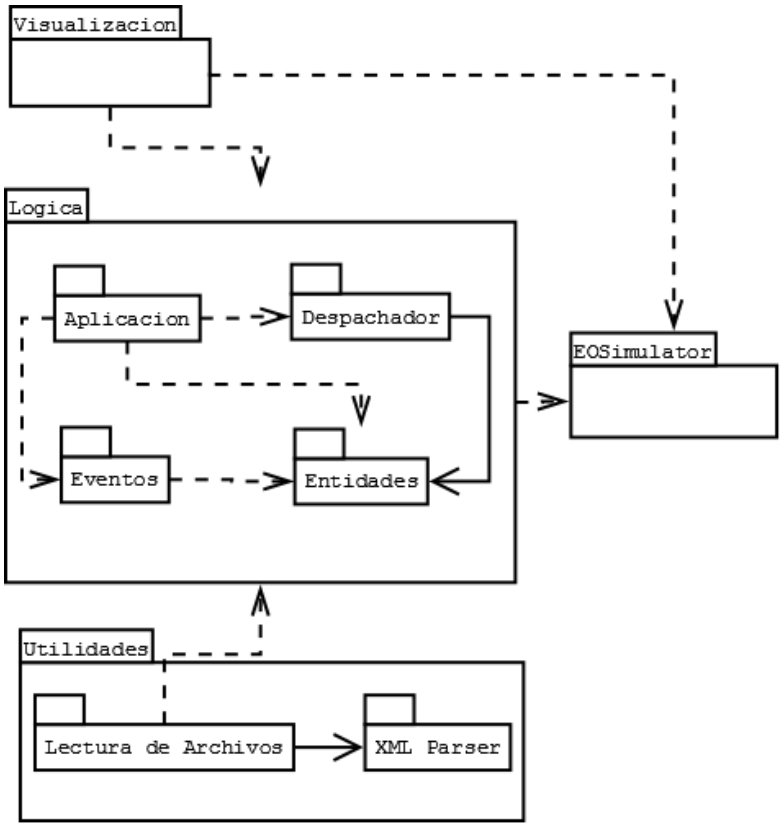


Figura 4.7.1

Como se puede observar en la Figura 4.7.1 separamos en paquetes los diferentes componentes.

Biblioteca de simulación y lenguaje de programación:

Luego de distintos estudios para elegir la biblioteca de simulación a utilizar (Anexo B), llegamos a la conclusión de que EOSimulator [2] sería la más adecuada para el desarrollo del simulador. Ésta reúne aquellas características que consideramos necesarias para llevar a cabo los requerimientos y a su vez nos brinda seguridad de soporte a través de integrantes del Departamento de Investigación de Operaciones de la Facultad de Ingeniería. Además, EOSimulator es una librería de simulación a eventos discretos escrita en C++. Esta dependencia nos hace desarrollar en este lenguaje de programación, el simulador de transporte público urbano.

Entorno de desarrollo:

A la hora de elegir el entorno de desarrollo a utilizar, Dev-C++ fue el que optamos, dado que teníamos un conocimiento previo del mismo.

Es un entorno de desarrollo integrado para programar en lenguaje C/C++. Usa MinGW que es una versión de GCC (GNU Compiler Collection) como su compilador.

El Entorno está desarrollado en el lenguaje Delphi de Borland. Tiene una página de paquetes opcionales para instalar, con diferentes bibliotecas de código abierto [16].

SDL:

Simple DirectMedia Layer (SDL) es un conjunto de bibliotecas desarrolladas con el lenguaje C que proporcionan funciones básicas para realizar operaciones de dibujado 2D, gestión de efectos de sonido y música, acceso a mouse y teclado, y carga y gestión de imágenes [51]. Pese a estar programado en C, tiene wrappers a otros lenguajes de programación como C++, Ada, C#, Basic, Erlang, Lua, Java, Python, etc.

EOSimulator utiliza internamente dicha librería para desplegar la salida gráfica pero haciendo uso de un subconjunto muy pequeño del total de las funcionalidades que SDL provee. Los eventos de teclado no son soportados por la librería de simulación por lo que tuvimos que utilizar nativamente SDL para poder agregar esta funcionalidad.

4.8. Salida gráfica

Ante el estudio de la diferentes bibliotecas de simulación (Anexo B), se pueden encontrar algunas que soportan salida gráfica y otras que no. Dentro de las que soportan una salida gráfica de la simulación, están aquellas que se destacan por sus animaciones tanto en 2D como en 3D, y están aquellas que contienen un conjunto de funcionalidades que permiten sencillas animaciones en 2D.

EOSimulator cae dentro de este último conjunto. Dado el nivel de los requerimientos (sección 4.2) en lo referente a la interfaz gráfica, ésta biblioteca reúne las funcionalidades básicas para poder desarrollarlos. La salida gráfica que provee EOSimulator posee la ventaja de que sus funcionalidades son relativamente fáciles de utilizar y a su vez bastante intuitivas. Pero como contrapartida, tiene ciertas limitaciones en cuanto a la cantidad de funcionalidades que provee, teniendo que entrar en el estudio de librerías graficas de C++ (SDL [51]) para su extensión.

Para poder satisfacer los requerimientos en lo referente a la interfaz gráfica, EOSimulator provee un módulo capaz de manejar simulaciones gráficas, el cual permite visualizar cómo se comporta el sistema. De lo contrario sólo podríamos guiarnos por datos recolectados durante la simulación. Este módulo se trata básicamente de una animación iconográfica en 2D, basado en librerías tales como Boost C++ Libraries, SDL, SDL_image, Antigrain Geometry y FreeType.

Dado los requerimientos del simulador y su salida gráfica tuvimos que modelar gráficamente los ómnibus, los pasajeros, las paradas, los centroides, las calles, las caminatas, y el movimiento de los ómnibus y pasajeros a lo largo de dichas calles y caminatas. Asimismo, se consideró importante poder visualizar en tiempo de simulación datos de interés y también poder mostrar un menú con referencias a funcionalidades de visualización y recolección de datos las cuales pueden ser invocadas a través del teclado.

En primer lugar decidimos organizar la pantalla en 3 sectores bien definidos, los cuales pueden observarse en la figura 4.8.1:

1) Sector de simulación
2) Sector de datos
3) Sector de referencias

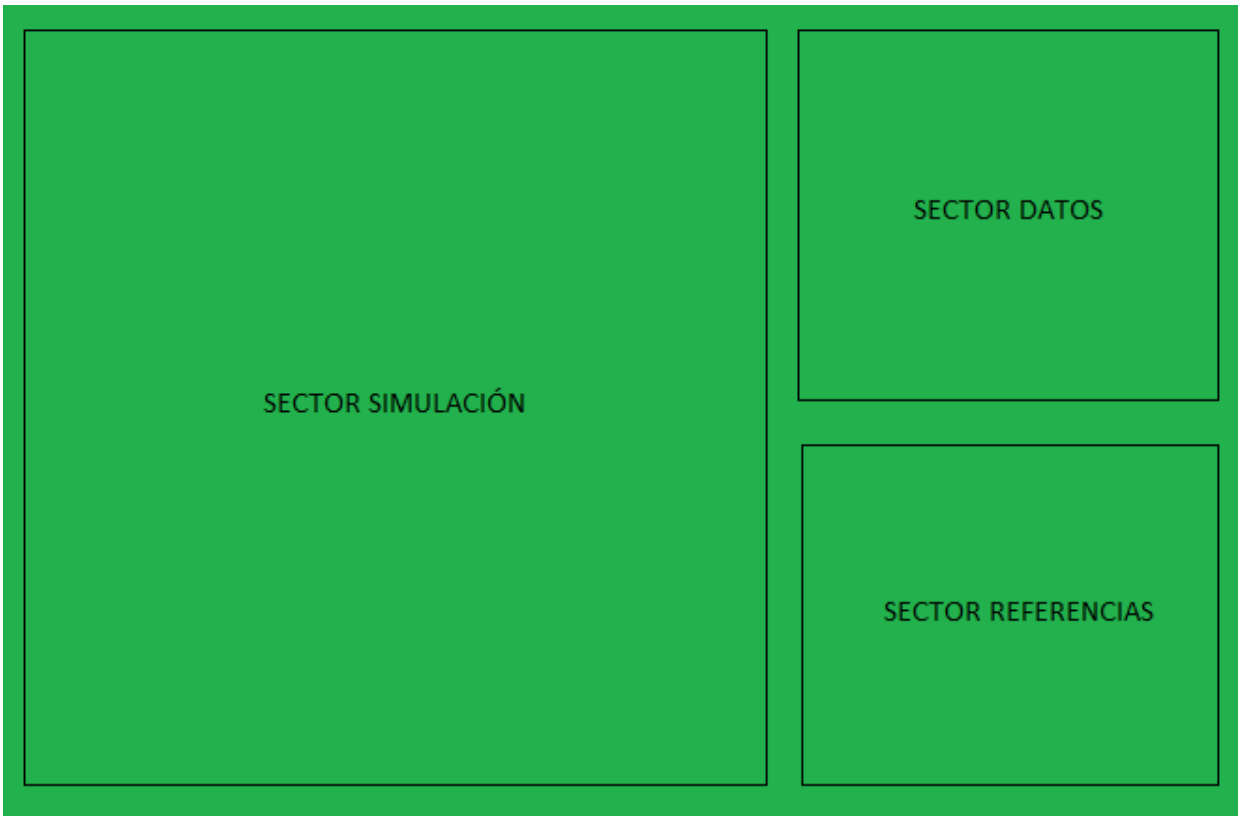


Figura 4.8.1

Sector simulación:

Este sector corresponde a la visualización de la simulación. Muestra la red vial, la ubicación de las paradas, la ubicación de los centroides, los ómnibus, los pasajeros y el comportamiento de los mismos a lo largo de la simulación. Este comportamiento incluye el movimiento de los ómnibus a lo largo de la red vial, la espera de los mismos en las diferentes paradas para que los pasajeros puedan bajar y/o subir del mismo, la generación de pasajeros en los centroides y su caminata a la parada para esperar el ómnibus, el descenso de los mismos del ómnibus y la caminata al centroide destino. Para llevar a cabo esto, EOSimulator cuenta con una clase (Sprite) para representar estos objetos de forma iconográfica en 2D y además provee funcionalidades sobre los mismos que permiten implementar el comportamiento de los mismos.

En la figura 4.8.2 se muestra como están representados los diferentes objetos:

Figura 4.8.2

Particularmente para el caso de los ómnibus se elije un color diferente dependiendo de la línea a la cual pertenece. Esto se hace con el objetivo de poder identificar fácilmente las diferentes líneas de ómnibus. Sobre estos íconos, EOSimulator (más específicamente la clase Sprite), nos permite moverlos de coordenada a coordenada así como también ocultarlos y mostrarlos, mediante las siguientes operaciones:

```
Sprite::setImage(const char *file)
```
Permite setearle una imagen para representar al objeto

```
Sprite::setMoves (MoveAction m)
```
Permite mover el objeto. MoveAction define el movimiento tomando como datos la coordenada origen, la coordenada destino y el tiempo que demora en mover el objeto del origen al destino.

```
Sprite::setVisible (bool v)
```
Permite mostrar y ocultar al objeto.

Sector datos:

En este sector se encuentran datos referentes a las líneas los cuales son actualizados en tiempo de simulación de forma automática.

Hay dos variables que medimos, “Promedio Tiempo en Viaje” y “Promedio Tiempo en Parada”. La primera hace referencia al promedio, por línea de ómnibus, del tiempo que le lleva a cada ómnibus perteneciente a la línea, en completar su viaje. No se contempla el tiempo que demora en las paradas para subir y bajar pasajeros y si se trata de una línea de Ida y Vuelta tampoco se contempla el tiempo de espera para comenzar el recorrido de vuelta. A modo de ejemplo (observando la tabla 4.8.3), el promedio del tiempo de viaje de la línea “Línea1”, es la suma de los tiempos de viaje de cada ómnibus que pertenece a la línea “Linea1” y que completa su viaje, divido por la cantidad de los mismos. La segunda variable, “Promedio Tiempo en Parada”, hace referencia al promedio, por línea de ómnibus, del tiempo que consume cada ómnibus perteneciente a la línea, en esperar por la subida y bajada de pasajeros en las paradas a lo largo de su viaje. Al igual que en el ejemplo anterior, el promedio del tiempo de espera de la línea “Linea1”, es la suma de los tiempos de espera en la paradas para subir y bajar pasajeros de cada ómnibus que pertenece a la línea “Linea1” una vez completado su viaje, dividido por la cantidad de los mismos.

Ambas mediciones son actualizadas al final del recorrido completo de la línea (en el caso de un línea de Ida y Vuelta el recorrido completo es el recorrido de ida más el de vuelta).

Esta tabla está organizada como se muestra en la tabla 4.8.3.

Línea	**Promedio Tiempo de Viaje**	**Promedio Tiempo en Paradas**
Línea1	[valor1]	[valor1]
Línea2	[valor2]	[valor2]
Línea3	[valor3]	[valor3]
…	…	…
…	…	…
…	…	…
LíneaN	[valorN]	[valorN]

Tabla 4.8.3

Sector referencias:
Este sector muestra las diferentes acciones que se pueden realizar sobre la simulación utilizando el teclado para llevarlas a cabo. En la figura 4.8.4 se observa cómo están desplegadas las diferentes opciones sobre la interfaz gráfica. A continuación pasamos a detallar cada una de ellas:

- Líneas de ómnibus: Se muestran las líneas de ómnibus existentes y se las asocia a un número/letra y también a un color. El número/letra hace referencia a la tecla que hay que presionar para ocultar o mostrar todos los ómnibus de la línea en cuestión. Esto se hace con el objetivo de poder visualizar, en un momento dado, sólo aquellas líneas que sean de interés. El color refleja cómo se visualiza los ómnibus de dicha línea durante la simulación.

- Calles: Asociado a la tecla "C". Al presionar esta tecla se muestra la red vial completa o se muestra parte de la misma. Esta parte de la red vial corresponde a aquellas calles que forman parte de algún recorrido de alguna línea. Esto se hace con el objetivo de darle claridad a la red vial y a los movimientos de los diferentes ómnibus. El hecho de poder cambiar el nivel de detalle de la red vial hace que la performance de la simulación varíe dado que ante un nivel de detalle mayor la computadora necesitará más recursos para poder correr la aplicación.

- Pasajeros: Asociado a la tecla "P". Al presionar esta tecla se muestran u ocultan los pasajeros del sistema. Los mismos pueden ser visualizados en el momento que un pasajero es creado en un centroide, su caminata a la parada origen, su espera en la misma, su descenso en la parada destino y finalmente su caminata al centroide final. Los pasajeros esperan en la parada en un radio pequeño de la misma, de forma de que puedan ser visualizados de mejor manera, de lo contrario todos irían a parar a un mismo punto.

- Centroides: Asociado a la tecla "T". Al presionar esta tecla se muestran o se ocultan todos los centroides del sistema y las caminatas que salen de los mismos.

- Paradas: Asociado a la tecla "G". Al presionar esta tecla se muestran o se ocultan todas las paradas del sistema.

- Aumentar Velocidad Simulación: Asociado a la tecla "I". Al presionar esta tecla aumenta la velocidad de la simulación. Este aumento no es infinito sino que está acotado por una constante que determina la velocidad máxima permitida.

- Disminuir Velocidad Simulación: Asociado a la tecla "K". Al presionar esta tecla disminuye la velocidad de la simulación. Esta disminución no es infinita sino que está acotada por una constante que determina la velocidad mínima permitida.

- Generar Reporte: Asociado a la tecla "M". Al presionar esta tecla se genera un reporte del estado actual de la simulación.

- Salir: Asociado a la tecla "Q". Al presionar esta tecla se aborta la simulación y finaliza la aplicación. Una vez que finaliza la simulación, es decir, se cumple el tiempo estipulado para

la misma, ésta queda pausada a la espera de que se presione “Q” para salir y finalizar la aplicación.

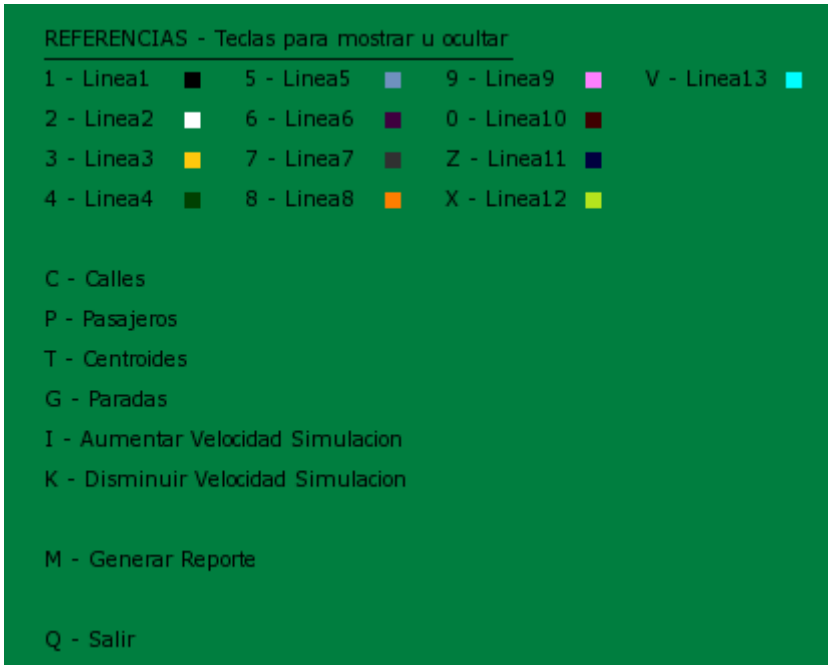


Figura 4.8.4

Mapeo de coordenadas

Dado que en la construcción del grafo, igoR-tp utiliza coordenadas geográficas, es necesario realizar una conversión de las mismas para poder representarlas en la interfaz gráfica del simulador. En primer lugar es importante tener en cuenta que la herramienta de visualización de igoR-tp utiliza un eje de coordenadas cartesiano orientado como muestra la figura 4.8.5 y en el cual las unidades se miden en pixeles.

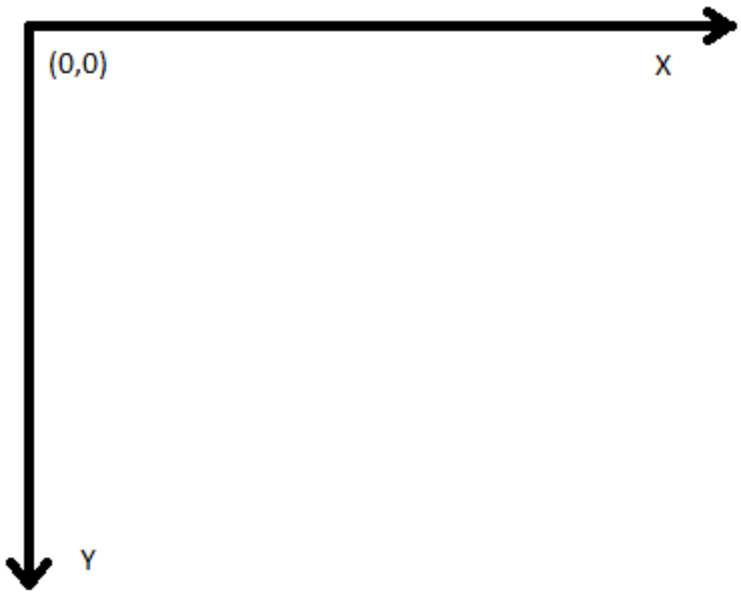


Figura 4.8.5

Luego se determinó una ecuación para calcular los valores de x e y correspondientes a la transformación de cada coordenada geográfica.

$X_{TRASFORMACION} = (X_{GEOGRAFICO} - X_{GEOGRAFICO\text{-}MINIMO}) / RATIO\_SCALE + OFFSET\_X$

$Y_{TRASFORMACION} = MAX\_Y_{GEOGRAFICO\text{-}MINIMO} - ((Y_{GEOGRAFICO} - Y_{GEOGRAFICO\text{-}MINIMO}) / RATIO\_SCALE + OFFSET\_Y)$

Pasamos a explicar cómo determinamos esta ecuación para poder determinar los valores de x e y para el simulador:

- $X_{GEOGRAFICO} - X_{GEOGRAFICO\text{-}MINIMO}$
  Mediante esta resta trasladamos la coordenada en x al origen.

- $(X_{GEOGRAFICO} - X_{GEOGRAFICO\text{-}MINIMO}) / RATIO\_SCALE$
  Mediante la división entre RATIO_SCALE escalamos x para que pueda ser visualizado completamente en la pantalla del simulador.

- $(X_{GEOGRAFICO} - X_{GEOGRAFICO\text{-}MINIMO}) / RATIO\_SCALE + OFFSET\_X$
  Mediante la suma de OFFSET_X definimos el desplazamiento en x del origen. Esto se hace para que no empiece en (0,0) y pueda ser visualizado de mejor manera.

- $Y_{GEOGRAFICO} - Y_{GEOGRAFICO\text{-}MINIMO}$
  Mediante esta resta trasladamos la coordenada en y al origen.

- $(Y_{GEOGRAFICO} - Y_{GEOGRAFICO\text{-}MINIMO})$ / RATIO_SCALE
  Mediante la división entre RATIO_SCALE escalamos y para que pueda ser visualizado completamente en la pantalla del simulador.

- $(Y_{GEOGRAFICO} - Y_{GEOGRAFICO\text{-}MINIMO})$ / RATIO_SCALE + OFFSET_Y
  Mediante la suma de OFFSET_Y definimos el desplazamiento en y del origen. Esto se hace para que no empiece en (0,0) y pueda ser visualizado de mejor manera.

- $MAX\_Y_{GEOGRAFICO\text{-}MINIMO} - ((Y_{GEOGRAFICO} - Y_{GEOGRAFICO\text{-}MINIMO})$ / RATIO_SCALE + OFFSET_Y)
  Esta resta lo que hace es permitir que la orientación de este eje se corresponda con la orientación en igoR-tp de la red vial. De esta manera podemos hacer la correspondencia visual entre el grafo bajo estas coordenadas y el grafo bajo las coordenadas geográficas de igoR-tp.

4.9. Recolección de datos y resultados

Reporte:

El reporte es una funcionalidad que provee el simulador capaz de recolectar datos en un momento dado y volcarlos en un archivo de texto con la finalidad de tener información relevante en cualquier momento de la simulación. Puede ser generado a través de la interfaz gráfica durante la simulación y a su vez es generado automáticamente al finalizar la misma, teniendo así un reporte final.

Este reporte tiene una estructura específica y es almacenado bajo la carpeta “reportes” en el directorio raíz del simulador. El archivo resultado es un archivo plano en formato TXT. Para llevar un control del mismo y referenciarlo, el nombre es autogenerado por el simulador y su formato, para el caso de los reportes generados durante la simulación, es el siguiente:

```
reporte<año><mes><día>_<hora><minutos><segundos>.txt
```

El formato del nombre del reporte cuando éste es generado al final de la simulación, es el siguiente:

```
reporte<año><mes><día>_<hora><minutos><segundos>FINAL.txt
```

En el reporte se almacenan datos referentes al tiempo de simulación en el que fue generado, datos de las líneas de ómnibus y de las paradas.

En el caso de las líneas de ómnibus, se almacena el identificador de la línea, el tipo de línea (Ida Y Vuelta o Circular) e información referente a los ómnibus que forman parte de la misma. Esto es, el identificador del ómnibus, la ocupación, el estado (En recorrido o Esperando) y el identificador del último nodo vial visitado (este identificador se corresponde con el mismo identificador generado sobre la capa de nodos viales en igoR-tp).

Para el caso de las paradas se registra el identificador de la parada (mismo identificador que la capa de paradas en igoR-tp) y la cantidad de pasajeros que se encuentran esperando el ómnibus en dicha parada.

La estructura del reporte es la siguiente:

```
TIEMPO SIMULACION = <tiempo>

Z1 = <tiempoZ1> //Únicamente para el reporte final

LINEA Id = <id_linea_1>
Tipo = <tipo_linea_1>
Omnibus Id = <id_omnibus_1>
Ocupacion = <ocupación_1>
Id Ultimo NodoVial = <id_nodoVial_1>
Estado = <estado1>
…
…
…
Omnibus Id = <id_omnibus_N>
Ocupacion = <ocupación_N>
Id Ultimo NodoVial = <id_nodoVial_N>
Estado = <estadoN>

PARADA Id = <id_parada_1>
Cantidad Pasajeros = <cantidad_1>
…
…
…
PARADA Id = <id_parada_N>
Cantidad Pasajeros = <cantidad_N>
```

- **<tiempo>:** tiempo de simulación
- **<tiempoZ1>:** resutado de Z1. Esta ecuación se encuentra explicada en este capítulo, bajo el título “Tiempo total de viaje de pasajeros”.
- **<tipo_linea>:** tipo de línea, puede ser Ida Y Vuelta o Circular
- **<Id_omnibus>:** identificador del ómnibus.

- **<ocupación>:** ocupación del ómnibus. Este dato viene en el formato "cantidad_de_pasajeros/cantidad_maxima_de_pasajeros"

- **<id_nodoVial>:** identificador del último nodo vial visitado por el ómnibus. Este identificador se corresponde con el de la capa de nodos viales en igoR-tp.

- **<estado>:** estado del ómnibus, puede ser Esperando o En recorrido. El estado Esperando se da cuando el ómnibus finalizó su recorrido y está a la espera de comenzar el recorrido nuevamente, una vez que el despachador de ómnibus lo disponga. El estado En recorrido se da cuando el ómnibus todavía no finalizó su recorrido.

- **<id_parada>:** identificador de la parada. Este identificador se corresponde con el mismo que tiene la capa de paradas en igoR-tp.

- **<cantidad>:** cantidad de pasajeros esperando el ómnibus en la parada.

El hecho de que los identificadores de los nodos viales y las paradas se correspondan con los de cada una de sus respectivas capas en igoR-tp, es para tener una referencia visual de los mismos. Dado que desplegar los identificadores en la interfaz gráfica del simulador los haría prácticamente imposible de visualizar para un usuario, fue que optamos por ésta forma de referenciarlos visualmente.

El simulador provee una funcionalidad (generarReporte) que precisamente se encarga de generar el mismo, siguiendo el siguiente pseudocódigo:

```
function generarReporte()
begin
   foreach Linea l
   begin
      l.generarReporte();
   end
   foreach Parada p
   begin
      p.generarReporte();
   end
end;
```

Mediante el pseudocódigo se puede observar que la generación del reporte de cada objeto es responsabilidad del mismo objeto. De esta forma, resulta sencillo poder agregar nuevos reportes de objetos diferentes y también poder agregar, modificar o eliminar información a la ya existente, yendo únicamente al método de generación de reporte correspondiente.

Cabe mencionar que el reporte se genera automáticamente cuando finaliza la simulación de forma de tener siempre un reporte final de la misma.

Histogramas

De forma de poder recolectar datos de la simulación, EOSimulator maneja histogramas. Cuenta con tres tipos de histogramas:

- Observation: Este tipo de histograma almacena el número de veces que el valor "x" fue medido en una variable.
- TimeSeries: Este tipo de histograma registra la evolución de una variable a lo largo del tiempo; una serie de valores en el tiempo.
- TimeWeighted: Este tipo de histogramas registra cuanto tiempo una variable obtuvo un determinado valor.

Para el caso de nuestro simulador el tipo de histograma utilizado es "Observation" recolectando datos sobre los tiempos de espera de los pasajeros y los tiempos de espera de los ómnibus en las paradas. Para el caso del tiempo de espera de los pasajeros, se mide la cantidad de pasajeros que esperaron un tiempo "t" en la parada. Mientras que para el caso del tiempo de espera de los ómnibus en las paradas, se mide la cantidad de veces que los ómnibus esperaron un tiempo "t" en las paradas.

Para poder obtener el histograma del simulador se utiliza el siguiente método:

```
void print (unsigned int cantCell, const char *path)
```

Recibe como parámetro la cantidad de barras que se quieren mostrar del histograma y el path donde se quiere almacenar el mismo. Ambos histogramas se almacenan en un archivo de texto, cada uno con su nombre correspondiente, bajo el directorio "histogramas" ubicado en la raíz del simulador.

Tiempo total de viaje de pasajeros

Otra variable que utilizamos como resultado en el simulador, es el tiempo total de viaje de los pasajeros (Z_1). Esto involucra, tiempo de viaje en el ómnibus, tiempo de espera y tiempo de transbordo de los pasajeros que van de un origen a un destino, ponderado a su vez por la demanda que ahí se establece [36].

La misma se calcula de la siguiente manera:

$$Z_1 = \sum_{i,j \in [1..n]} d_{ij} (tv_{ij} + te_{ij} + tt_{ij})$$

Donde,

d_{ij}: demanda de pasajeros que hay con centroide origen "i" y centroide destino "j".
tv_{ij}: tiempo de viaje del pasajero que va del centroide origen "i" al centroide destino "j".
te_{ij}: tiempo de espera en las paradas del pasajero que va del centroide origen "i" al centroide destino "j".
tt_{ij}: tiempo de transbordo del pasajero que va del centroide origen "i" al centroide destino "j".

Cabe destacar que en esta ecuación no se considera en tiempo de caminata de los pasajeros, ya que el modelo de red utilizado por Z_1 en [36] no considera tiempos ni distancias de caminanta.

El resultado numérico de esta variable se despliega una vez finalizada la simulación, en el reporte final generado.

4.10. Entrada de datos

Para correr el simulador es necesario pasarle una serie de parámetros. Estos son:

- Path absoluto del grafo: Este parámetro corresponde al path donde se encuentra el archivo que representa al grafo. Recordemos que igoR-tp puede generar este tipo de archivos. El formato del mismo está especificado en 3.2.2.

- Path absoluto de la demanda de pasajeros: Este parámetro corresponde al path donde se encuentra el archivo que contiene los datos de demanda de pasajeros. El mismo sigue el siguiente formato:

```
<centroide_origen1> <centroide_destino1> <demanda1>
<centroide_origen1> <centroide_destino2> <demanda2>
…
…
…
<centroide_origenN> <centroide_destinoN> <demandaN>
```

  centroide_origen: corresponde al identificador del centroide donde se crea al pasajero.
  centroide_destino: corresponde al identificador del centroide final.
  demanda: corresponde al valor de la demanda de pasajeros para el par de centroides origen-destino. La unidad de este valor está dado en 1/minutos

  Aquellos pares origen-destino que no se encuentren en este archivo serán considerados con demanda cero.

- Path absoluto de los recorridos: Este parámetro corresponde al path donde se encuentra el archivo que contiene los datos de los distintos recorridos. Recordemos que igoR-tp puede generar este tipo de archivos. El mismo tiene la siguiente estructura:

```
<?xml version="1.0" standalone="no"?>
<Solucion>
  <nombre>…</nombre>
  <ruta>…</ruta>
  <capacidad>…</capacidad>
```

```
  <frecuencias>
    <frecuencia>…</frecuencia>
    …
    <frecuencia>…</frecuencia>
  </frecuencias>
  <recorridos>
    <recorrido>
      <nombre>…</nombre>
      <ruta>…</ruta>
      <frecuencia>…</frecuencia>
      <activo>…</activo>
      <color>
        <R>…</R>
        <G>…</G>
        <B>…</B>
      </color>
      <visible>…</visible>
      <nodos>
        <nodo>…</nodo>
        …
        <nodo>…</nodo>
      </nodos>
      <circular>…</circular>
      <dirigido>…</dirigido>
    </recorrido>
  </recorridos>
  <evaluaciones />
  <evalActiva />
</Solucion>
```

Esta estructura es conservada de igoR-tp. De acá obtenemos únicamente la secuencia de nodos que componen cada recorrido.

- Path absoluto de las líneas: Este parámetro corresponde al path donde se encuentra el archivo que contiene los datos de las líneas de ómnibus. Este archivo especifica, los recorridos que forman parte de las distintas líneas, sus frecuencias y en caso de tratarse de líneas de Ida y Vuelta, cual es el tiempo que demora de pasar del recorrido de ida al de vuelta. El mismo tiene la siguiente estructura:

```
<tipo_linea1> <rec_ida1> <rec_vuelta1> <espera1> <frecuencia1>
<tipo_linea2> <rec_ida2> <rec_vuelta2> <espera2> <frecuencia2>
<tipo_linea3> <recorrido3> <frecuencia3>
…
…
<tipo_lineaM> <rec_idaM> <rec_vueltaM> <esperaM> <frecuenciaM>
…
…
<tipo_lineaN> <recorridoN> <frecuenciaN>
```

tipo_linea: número que corresponde al tipo de línea. Si es 0, corresponde a un tipo de línea “Ida Y Vuelta”. Y además, se sabrá que le seguirán 4 parámetros más, los cuales determinan el recorrido de ida, el recorrido de vuelta, el tiempo de espera para pasar del primero al segundo y la frecuencia de la línea. Si es 1, corresponde a un tipo de línea “Circular”. Luego le seguirán 2 parámetros más, los cuales determinan el recorrido y la frecuencia de la línea. Estos parámetros adicionales serán detallados posteriormente.

rec_ida: número de recorrido de ida que forma parte de la línea de “Ida y Vuelta”. Este número se corresponde con la posición de dicho recorrido en el archivo generado por igoR-tp y descrito en el punto anterior.

rec_vuelta: número de recorrido de vuelta que forma parte de la línea de “Ida y Vuelta”. Este número se corresponde con la posición de dicho recorrido en el archivo generado por igoR-tp y descrito en el punto anterior.

recorrido: número de recorrido forma parte de la línea “Circular”. Este número se corresponde con la posición de dicho recorrido en el archivo generado por igoR-tp y descrito en el punto anterior.

espera: tiempo de espera que demora una línea de “Ida y Vuelta” para pasar del recorrido de ida al de vuelta.

frecuencia: es la frecuencia de salida de la línea.

- Tiempo de simulación: Este parámetro corresponde a la duración de la simulación.

5. Caso de Estudio

Uno de los objetivos de este proyecto es aplicar las herramientas desarrolladas: igoR-tp v2.0 y el simulador, a un caso de estudio real. En este trabajo se utiliza como caso de estudio el sistema de transporte colectivo de la ciudad de Rivera. Mediante la construcción del mismo se logra testear igoR-tp y luego validar el modelo de simulación, comparando los resultados obtenidos con los del sistema real de transporte público (los cuales fueron obtenidos de [36]), con el modelo de asignación de Baaj y Mahmassani [55] y con diferentes experimentos realizados.

5.1. Descripción del caso de estudio

A continuación se detallará la descripción del caso de estudio, la cual fue tomada de [36], en la cual se relevó el sistema de transporte público urbano de la ciudad de Rivera en 2003/2004.

La ciudad de Rivera cuenta con una población de aproximadamente 65.000 habitantes. El transporte público urbano es un medio muy utilizado generalmente por estudiantes y jubilados, y en las horas pico, se destaca la presencia de trabajadores que se desplazan desde sus hogares a las localidades donde trabajan y viceversa. El desplazamiento en ómnibus es muy importante en esta ciudad por la geografía de la misma ya que gran parte de ésta se encuentra ubicada entre cerros lo que dificulta otros medios de transporte como la bicicleta. Además, debido al crecimiento de la ciudad y a los cambios en el uso del suelo, hay barrios que se encuentran bastante alejados del centro. Cabe destacar que desde la década del 50 existe oferta de líneas de transporte público. Empresas privadas de transporte invirtieron para lograr la implantación de un conjunto de líneas, cuyos recorridos cubren prácticamente toda el área urbana y suburbana de la ciudad.

Se relevaron 13 líneas como se indica en tabla 5.1.1, las cuales son operadas por 3 empresas privadas que transportan aproximadamente 7000 pasajeros por día. En 11 de las 13 líneas, los recorridos están estructurados en base a dos recorridos, uno de ida (desde el centro de la ciudad a un determinado barrio) y uno de vuelta (del barrio al centro). Las 2 líneas restantes tienen recorridos circulares, que comienzan y finalizan en una misma parada del centro de la ciudad, y se recorren en un solo sentido.

Líneas de ómnibus	Tiempo entre pasadas
Empalme	30
La Pedrera	60
Pueblo Nuevo	30
Comeri – Estiva	30
Santa Teresa	60
Rivera Chico	30
Quintas al Norte	30
Lagunón	60
Paso de la Estiva	20
Escuela 88	30
Mandubí	20
Cinco Bocas - Puerto Seco 1	60
Cinco Bocas - Puerto Seco 2	60

Tabla 5.1.1

El planeamiento de las líneas, incluyendo la definición de los recorridos junto con sus frecuencias, son administrados por el Departamento de Tránsito y Transporte de la Intendencia Municipal de Rivera junto con las empresas. La definición de los recorridos que componen las líneas de ómnibus se realizan en base a conocimientos cualitativos de la demanda y a criterios de los técnicos del Departamento de Tránsito y Transporte de la IMR. Generalmente, la extensión de recorridos hacia nuevas áreas de la ciudad, es causada por reclamos de la población al no contar con líneas a las cuales puedan acceder a una distancia razonable de caminata. Los recorridos de las distintas líneas, se realizan por determinadas vías de tránsito acondicionadas especialmente para esto.

Los servicios comienzan a operar a la hora 6:00 y finalizan a la hora 22:00 o 24:00, según la línea. Existen algunas modificaciones en los trazados de los recorridos en algunos turnos, así como en las frecuencias, que se reducen después de la hora 20:00.

Es de interés para las autoridades locales realizar un estudio sobre el transporte público urbano debido a que los técnicos responsables han detectado algunos problemas. Más precisamente, detectaron competencia entre muchas líneas para servir los mismos pares origen-destino, y baja utilización de los buses en determinados tramos de algunas líneas.

Figura 5.1.2: La figura muestra el plano de la ciudad de Rivera.

5.2. Construcción del caso de estudio

Para la construcción del caso de estudio de la ciudad de Rivera fueron utilizadas las herramientas desarrolladas en este proyecto: igoR-tp v2.0 y el simulador. En una primera etapa, se construyó el modelo de red utilizando el módulo de construcción. Sobre éste, se definieron los recorridos de los ómnibus a través del módulo de manipulación, para finalmente poder evaluar el caso de estudio con el simulador, con el objetivo de poder validarlo.

5.2.1 Módulo de construcción

Utilizando las funcionalidades provistas por este módulo, se contruye el modelo de red de la ciudad de Rivera, el cual está compuesto por la red vial, zonas, centroides, nodos viales, paradas y caminatas. Para la construcción del mismo se siguen los siguientes pasos:

Construcción del proyecto

Se creó un proyecto en el módulo de construcción. Para esto, fue necesario contar con la capa de calles de Rivera y con la capa de puntos de demanda, las cuales fueron obtenidas del proyecto PDT 48/02. Con la creación del mismo fue necesario ingresar el valor de la velocidad media del ómnibus, fijada en 13,6 km/h [36] y la velocidad media del peatón, fijada en 4 km/h [5]. Otro parámetro a ingresar fue el tipo de distancia utilizada (Manhattan o Euclídea). Observando el plano de la ciudad de Rivera, vemos que la mayoría de las manzanas son cuadradas por lo que es mejor tomar la distancia Manhattan. En la figura 5.2.1.1 se muestran los valores asignados a los parámetros. La figura 5.2.1.2 representa la visualización del módulo de construcción luego de creado el proyecto.

Se observa la capa de red vial representada con líneas en color gris y la capa de puntos de demanda representada por puntos en color rosado.

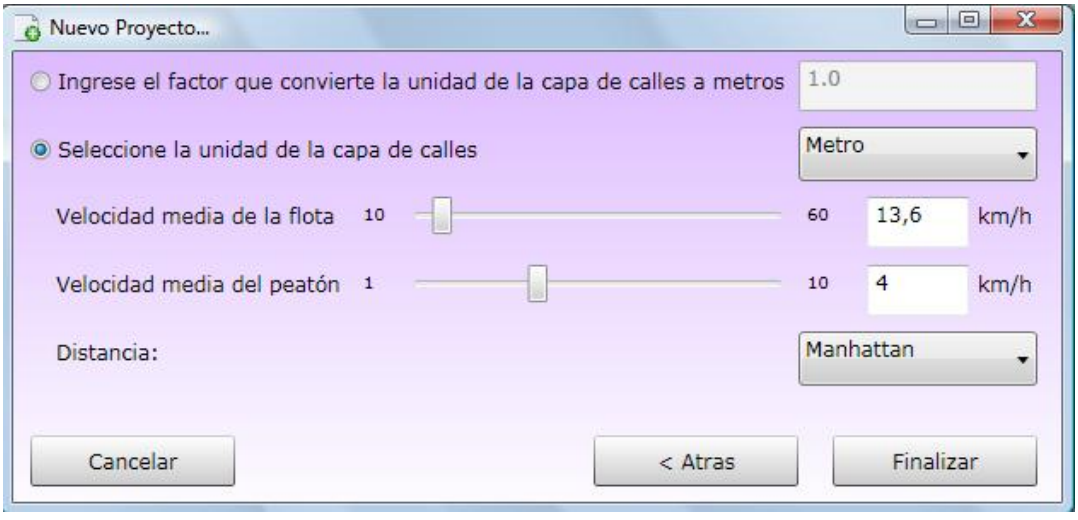


Figura 5.2.1.1: La figura muestra los parámetros y los valores asignados a los mismos.

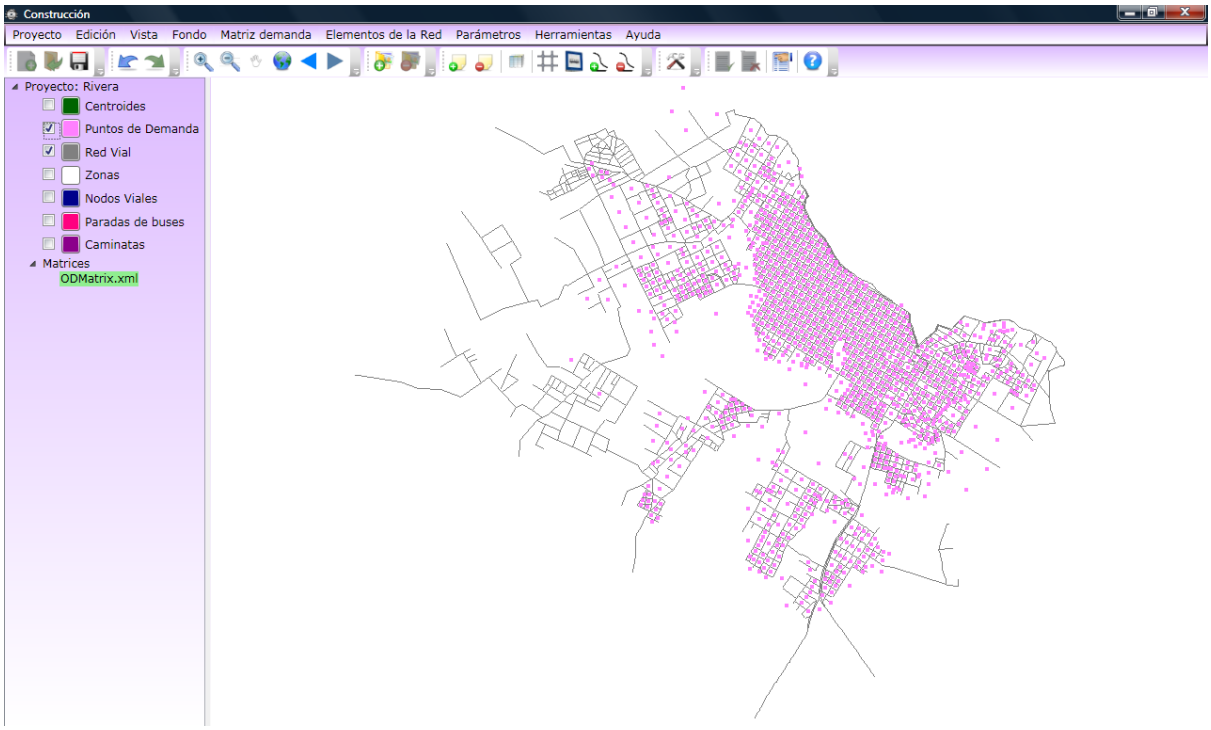


Figura 5.2.1.2: La figura muestra la capa de red vial representada con líneas en color gris y la capa de puntos de demanda representada por puntos en color rosado.

Digitalización de zonas

Para la digitalización de las zonas se utilizó una copia impresa de un archivo AutoCAD con la información de las distintas zonas (imagen de las zonas sobre las calles). El mismo fue creado en [36] y se proporcionó con el objetivo de armar un caso que sea consistente con dicha tesis, de modo de poder comparar y validar los resultados. A partir del archivo AutoCAD y tomando como referencia la capa de calles, se digitalizaron las 84 zonas, creándose automáticamente los centroides de cada una.

En la figura 5.2.1.3 se muestran las 84 zonas con sus respectivos centroides creadas en el módulo de construcción.

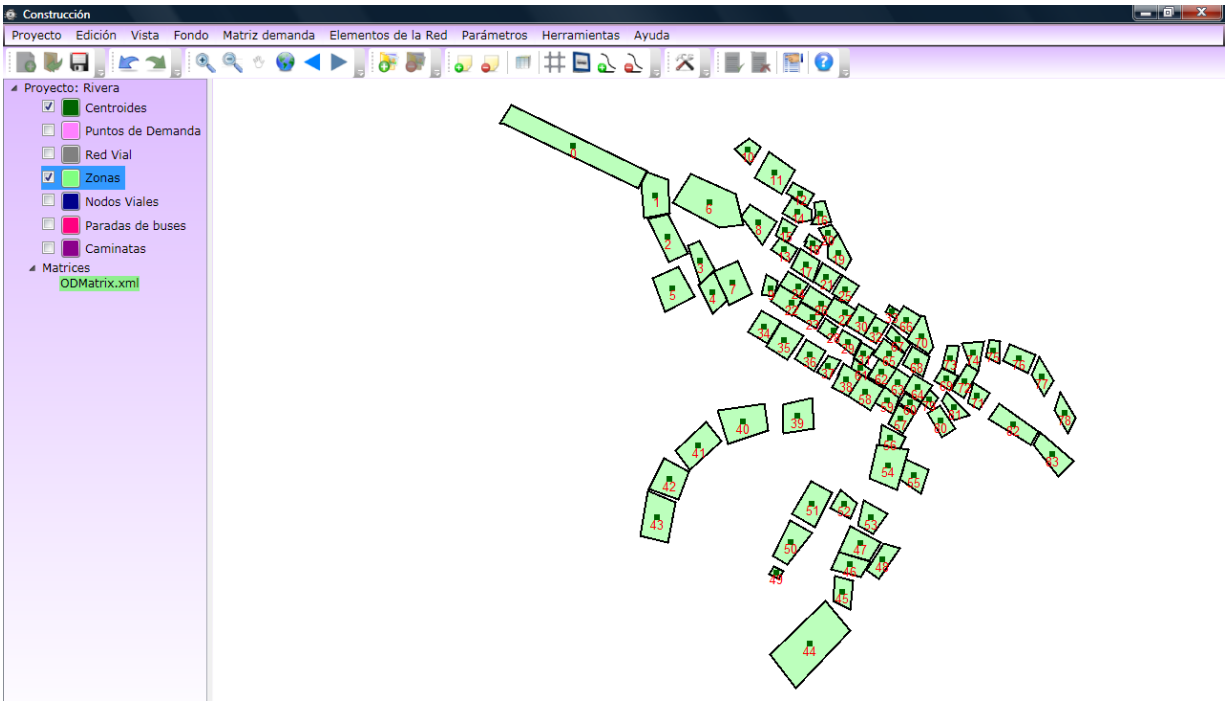


Figura 5.2.1.3: La figura muestra las 84 zonas con sus respectivos centroides.

Generación de nodos viales

Los nodos viales son generados automáticamente por igoR-tp v2.0, a partir de la red vial. En la figura 5.2.1.4 se muestra la capa de nodos viales y en la figura 5.2.1.5 se superpone la capa de nodos viales con la de red vial. Se observa que se generó un nodo vial por cruce de calle.

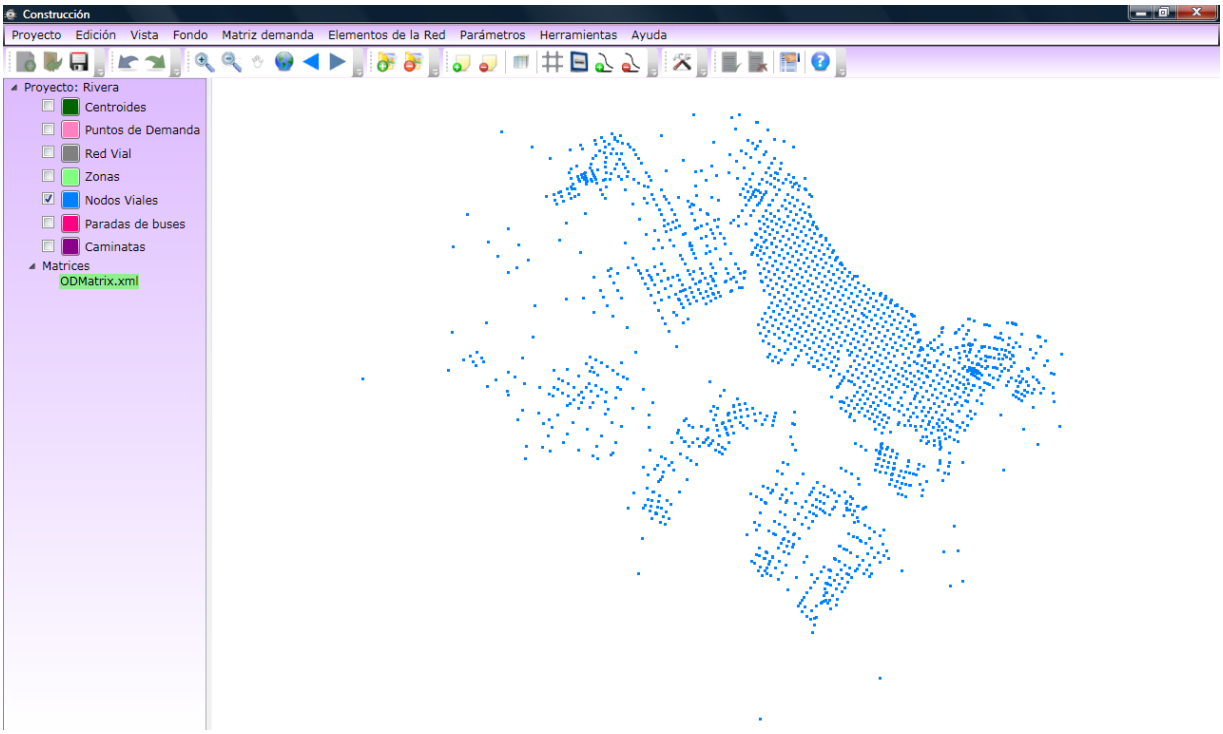


Figura 5.2.1.4: La figura muestra la capa de nodos viales.

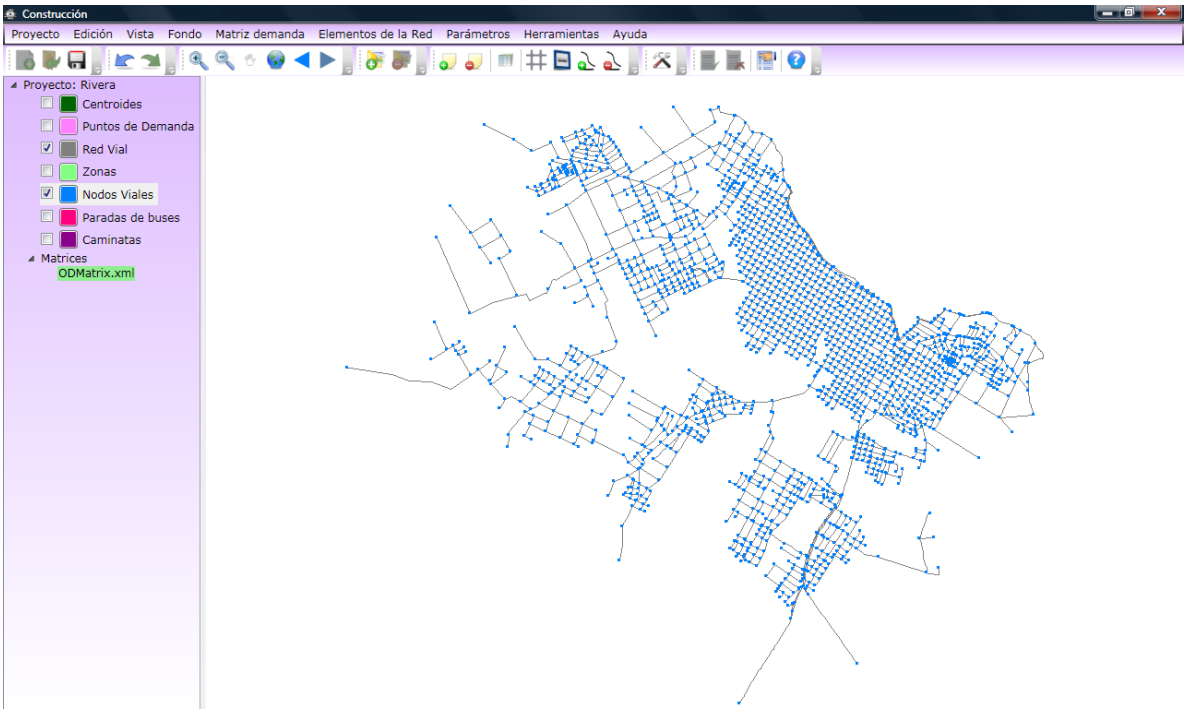


Figura 5.2.1.5: La figura muestra la capa de nodos viales y la capa de calles de la ciudad de Rivera.

Digitalización de paradas

Se cargó como fondo la capa de paradas proporcionada por el proyecto PDT 48/02 la cual fue tomada como referencia para digitalizar las paradas. En IgoR-tp v2.0 las paradas son localizadas sobre los nodos viales. En algunos casos, la localización física de las paradas no coincidía con la de algún nodo vial. Esto es debido a que algunas paradas no se encuentran en la esquina misma, sino que cercanas a ésta. Entonces, se marcó como parada aquel nodo vial que se encontrara más próximo visualmente a la parada a digitalizar.

En la figura 5.2.1.6 se muestra el resultado de la digitalización de la capa de paradas. En la figura 5.2.1.7 se visualizan los nodos viales representados por puntos en color azul, la red vial en color gris y las paradas en color rosado.

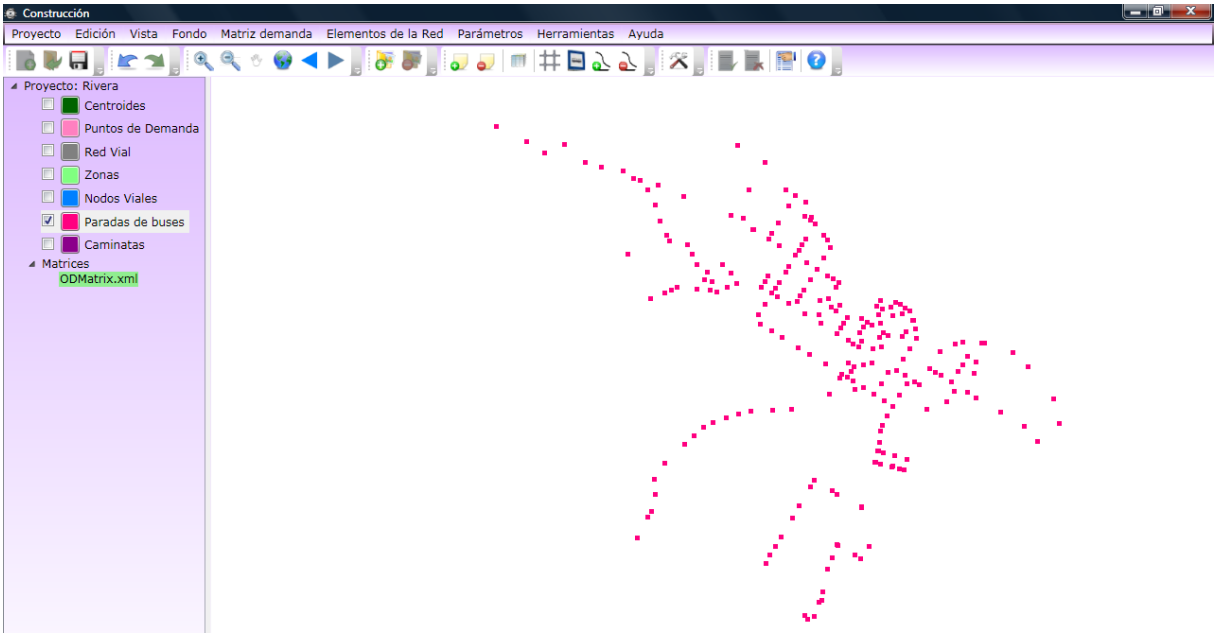


Figura 5.2.1.6: La figura muestra la capa de paradas.

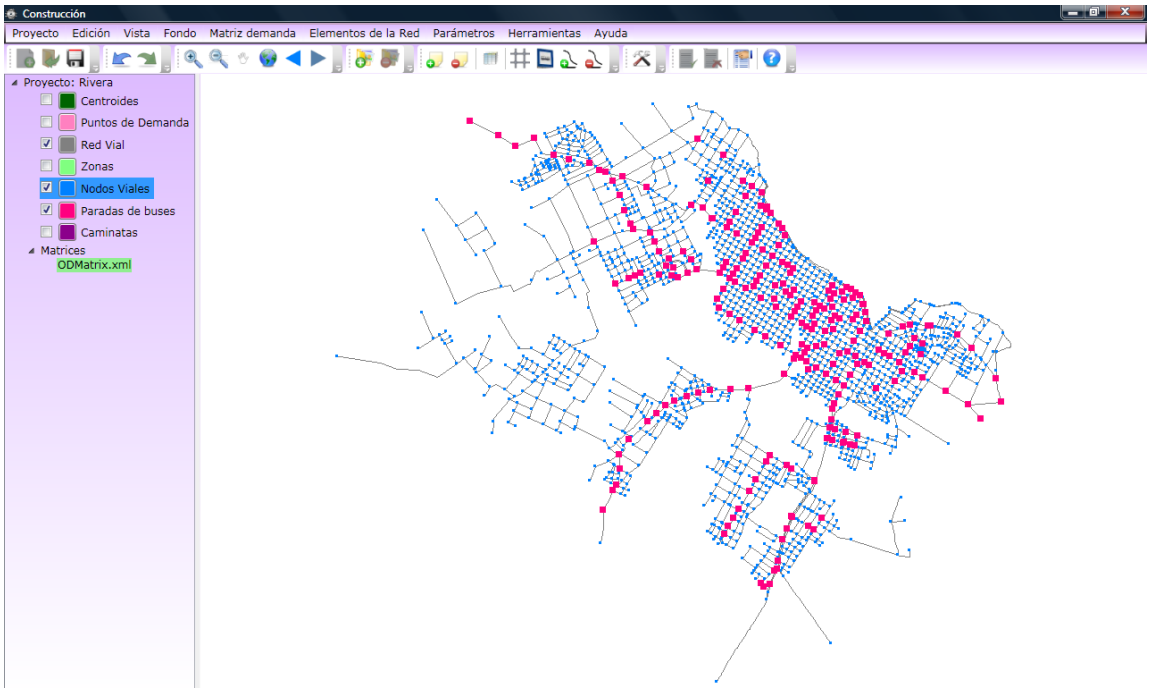


Figura 5.2.1.7: La figura muestra los nodos viales representados por puntos en color azul, la red vial en color gris y las paradas en color rosado.

Construcción de caminatas

Se conectaron todas las paradas con los centroides de la zona a la cual pertenecen como muestra la figura 5.2.1.8.

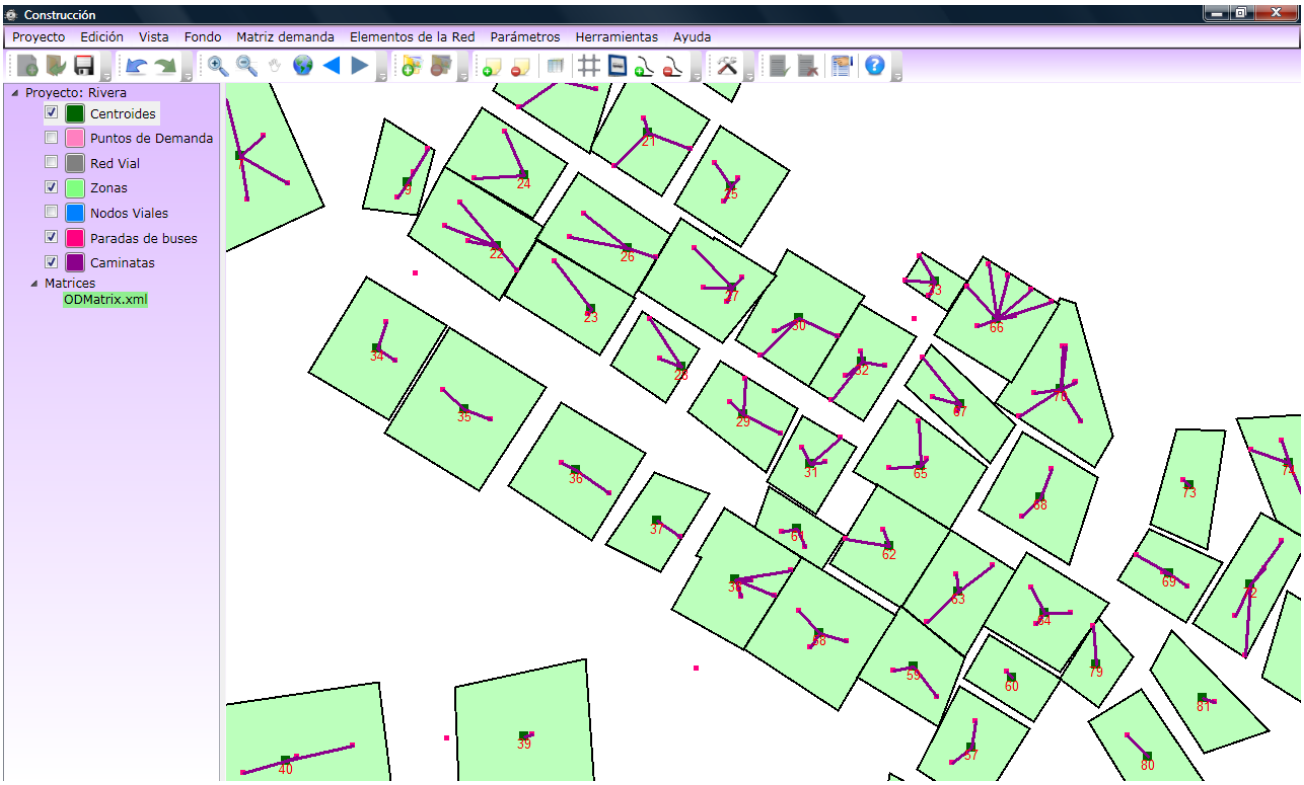


Figura 5.2.1.8: La figura muestra una vista parcial de las zonas de Rivera, con sus paradas y caminatas desde las paradas a los centroides de dichas zonas.

Matriz de demanda

La matriz origen-destino no fue generada con el módulo de construcción, sino que utilizamos la matriz de demanda calculada en el proyecto PDT 48/02. Por lo que ubicamos el archivo que la contenía, bajo la carpeta “Matrices” que se encuentra en la raíz del proyecto de construcción. Para que se mantenga la correspondencia entre la matriz origen-destino mencionada anteriormente y la numeración de las distintas zonas, éstas últimas se generaron en el mismo orden que las de la copia impresa del archivo AutoCAD. De esta manera no fue necesario realizarle modificaciones a la matriz de demanda.

Luego de realizados estos pasos, concluimos las acciones en el módulo de construcción.

5.2.2 Módulo de Algoritmos

En este módulo damos de alta el simulador como un algoritmo más en el sistema. Para esto, se realizaron los siguientes pasos.

Creación del archivo de configuración

Para dar de alta al Simulador como algoritmo en igoR-tp v2.0, abrimos el módulo de algoritmos y seleccionamos la funcionalidad "Crear archivo de configuración de algoritmo". Luego ingresamos el nombre del algoritmo *Simulador*, la ubicación del mismo (ejecutable del algoritmo) y la cantidad de parámetros de entrada que éste necesita. Finalmente, para cada parámetro completamos el nombre del mismo, el tipo (caracteres, entero, real) y observación, como se muestra en la Tabla 5.2.2.1.

Nombre	*Tipo*	*Observación*
Ruta del archivo grafo.txt	*Caracteres*	
Ruta del archivo solución de las líneas	*Caracteres*	
Ruta del archivo demanda.txt	*Caracteres*	
Ruta del archivo lineas.txt	*Caracteres*	
Tiempo de simulación (segundos)	*Caracteres*	

Tabla 5.2.2.1: La tabla muestra la definición de los nombres de los parámetros de entrada para el simulador.

Esta información se guarda en un archivo xml en la carpeta "Algoritmos" (ubicada en la raíz de la aplicación igoR-tp). La estructura del archivo es la siguiente:

```
<Algoritmo>
      <nombre>…</nombre>
      <ruta>…</ruta>
      <parametros>
      <parametro>
       <nombre>…</nombre>
       <tipo>…</tipo>
                     <observacion />
              </parametro>
      …
      …
      …
      <parametro>
       <nombre>…</nombre>
       <tipo>…</tipo>
                     <observacion />
              </parametro>
       </parametros>
</Algoritmo>
```

5.2.3 Módulo de Manipulación

Partiendo del proyecto creado en el módulo de construcción y utilizando las funcionalidades que implementa el módulo de manipulación, se creó la solución la cual contiene los recorridos que componen las líneas de ómnibus de la ciudad de Rivera. Luego, se evaluó la solución creada con el simulador obteniéndose así los resultados para el mismo. Más específicamente, los pasos que se realizaron utilizando este módulo fueron los siguientes:

Creación del proyecto

Creamos un proyecto nuevo en el módulo de manipulación a partir del proyecto creado en el módulo de construcción. Esto tiene como consecuencia, que el módulo de manipulación carga automáticamente las capas nodos viales, paradas, red vial, caminatas, centroides (con las que trabajamos en el proyecto de construcción) y la matriz de demanda. En la figura 5.2.3.1 se muestra como se visualiza el proyecto de manipulación creado, a partir del proyecto de construcción referente a la ciudad de Rivera.

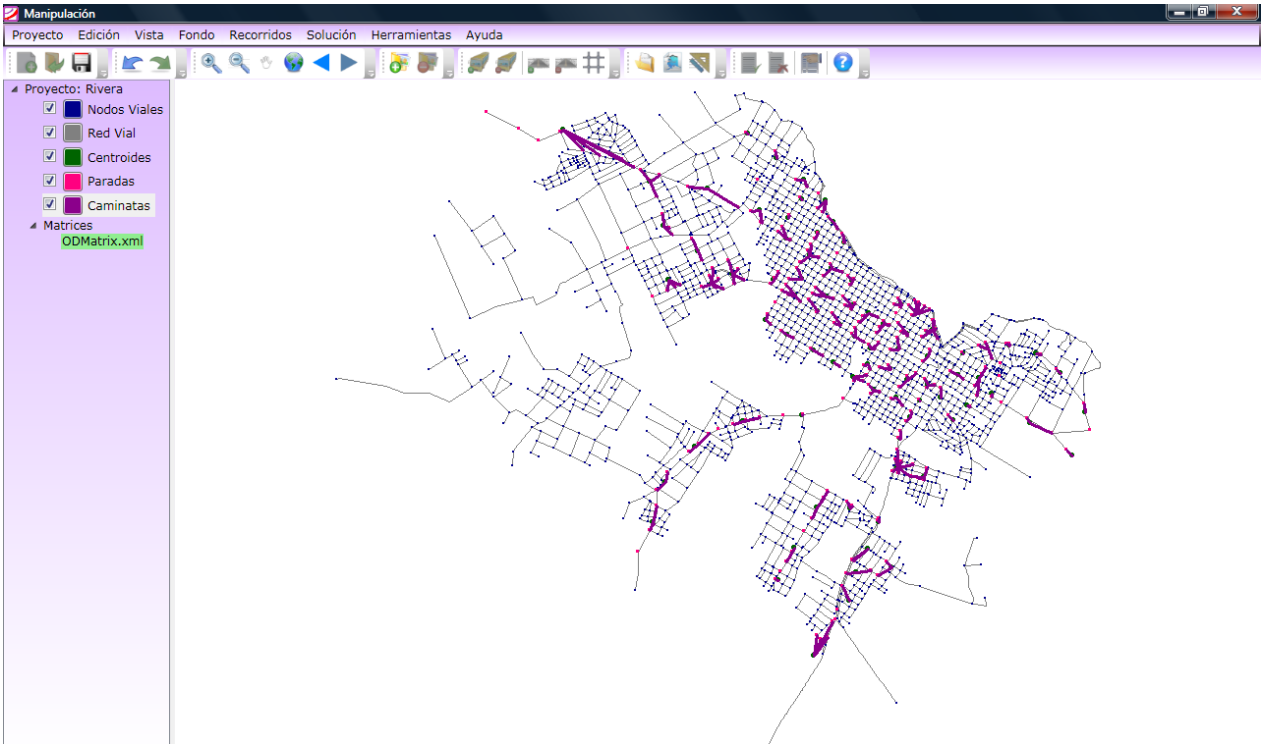


Figura 5.2.3.1: La figura muestra el nuevo proyecto de manipulación creado a partir del proyecto de construcción que se describió en esta sección.

Creación de la solución

Creamos una solución la cual llamamos *LíneasRivera* y definimos los recorridos de las líneas de la ciudad de Rivera que conforman la solución. Se cargó como fondo las capas de recorridos de la ciudad de Rivera proporcionadas por el proyecto PDT 48/02 y tomando éstas como referencia, se digitalizaron los recorridos. Para el caso de la líneas de ida y vuelta, se creó cada recorrido por

separado, siendo el archivo líneas.txt el encargado de llevar la información de los recorridos que forman parte de dicha línea. La estructura del mismo se puede observar en la sección 4.10.

En las figuras 5.2.3.2, 5.2.3.3, 5.2.3.4 y 5.2.3.5 se muestran como quedaron los recorridos digitalizados.


ComeriEstivaDesdeCeballos
ComeriEstivaDesdeMaximilianoCruz
EmpalmeDesdeCrespo
EmpalmeDesdeZonaFranca
Escuela88Desde25deAgosto
Escuela88DesdeTrapani

Figura5.2.3.2

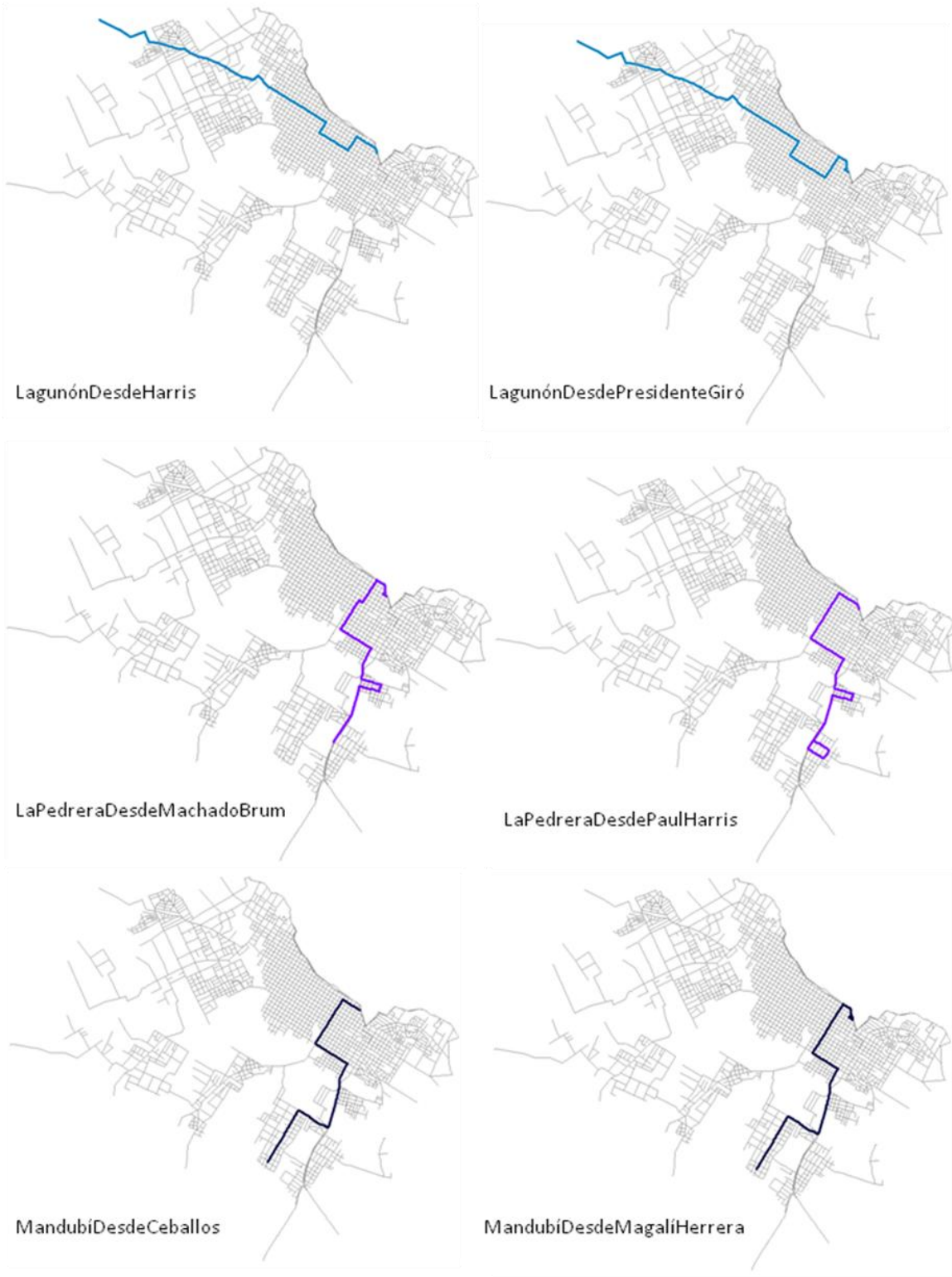

LagunónDesdeHarris
LagunónDesdePresidenteGiró
LaPedreraDesdeMachadoBrum
LaPedreraDesdePaulHarris
MandubíDesdeCeballos
MandubíDesdeMagalíHerrera

Figura5.2.3.3


PasodelaEstivaDesdeHarris
PasodelaEstivaDesdePresidenteGiró
PuebloNuevoDesdeCrespo
PuebloNuevoDesdeYamandú
PuertoSeco5BocasDesdeCeballos
PuertoSeco5BocasDesdeCeballosB

Figura5.2.3.4


QuintasAlNorteDesdeHarris
QuintasAlNorteDesdeSanabria
RiveraChicoDesdeCeballos
RiveraChicoDesdeSanabria
SantaTeresaDesdeBarreiro
SantaTeresaDesdeCrespo

Figura5.2.3.5

Creación del archivo líneas.txt

Una vez definidos todos los recorridos de las líneas, dejamos en segundo plano el módulo de manipulación y creamos un archivo llamado líneas.txt que determina para cada línea, que recorridos (tomados de la solución) la componen. Recordar que esto se realizó debido a que igoR-tp maneja el concepto de recorridos pero no de líneas. Mediante este archivo podemos unificar ambos conceptos y relacionarlos entre sí, especificando los recorridos que forman parte de las distintas líneas, sus frecuencias y en caso de tratarse de líneas de Ida y Vuelta, cual es el tiempo que demora de pasar del recorrido de ida al de vuelta.

A continuación se muestra el contenido del archivo líneas.txt:

```
0   1   2   0   1800
0   3   4   0   1800
0   5   6   0   3600
0   7   8   0   3600
0   9   10  0   1200
0   11  12  0   1800
0   13  14  0   1800
0   15  16  0   1800
0   17  21  0   3600
0   19  20  0   1800
0   22  23  0   1200
1   18  3600
1   24  3600
```

Evaluación de la solución

Después de finalizar la tarea anterior, seguimos trabajando en el módulo de manipulación y procedemos a evaluar la solución *LíneasRivera* seleccionando como algoritmo el *"Simulador"* como se muestra en la figura 5.2.3.6. Se despliega un diálogo como muestra la figura 5.2.3.7, en el cual se debe ingresar los valores para los parámetros de entrada del simulador.

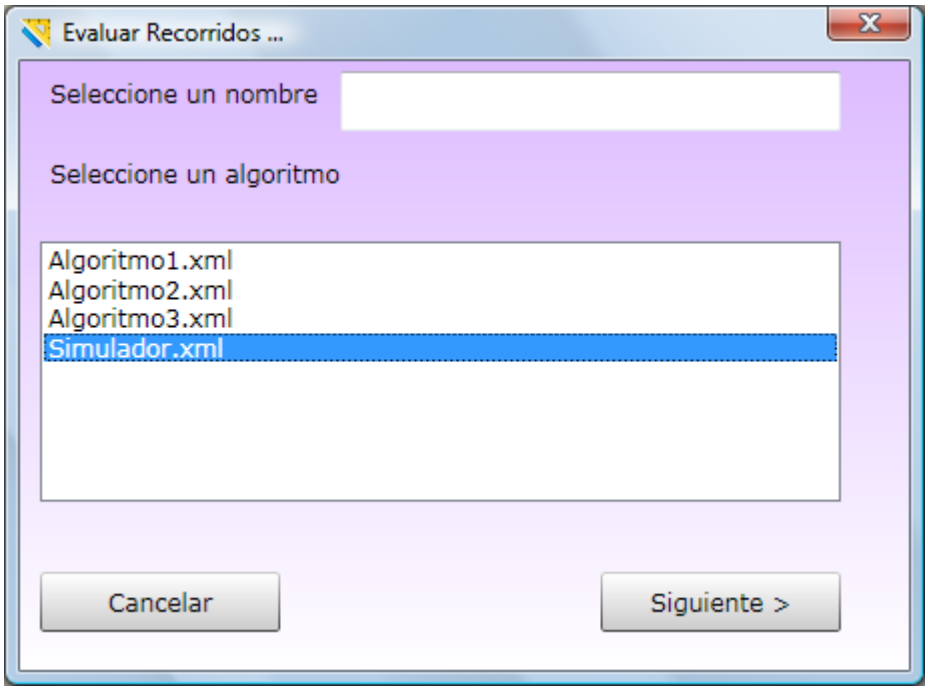


Figura 5.2.3.6: La figura muestra el diálogo que se despliega para que el usuario seleccione el algoritmo.

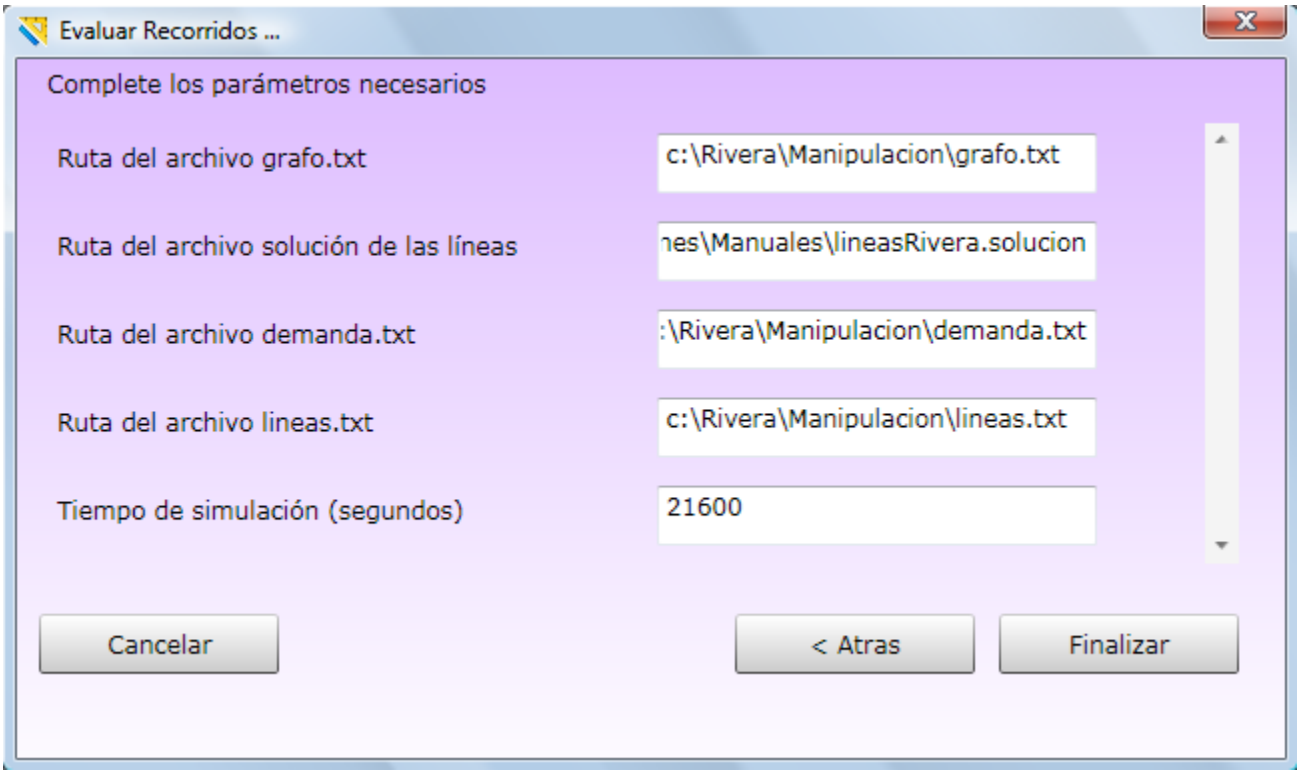


Figura 5.2.3.7: La figura muestra el diálogo que se despliega para que el usuario complete los valores de los parámetros para el algoritmo Simulador.

Los archivos grafo.txt y demanda.txt se encuentran bajo la carpeta donde está almacenado el proyecto de manipulación sobre el cual se esté trabajando. El archivo solución de las líneas, se encuentra bajo la carpeta cuyo nombre se corresponde con el nombre de la solución creada, la cual puede estar ubicada bajo la carpeta “Soluciones/Manuales”, si los recorridos fueron creados por el usuario o “Soluciones/Ejecuciones” si los recorridos fueron generados por un algoritmo. Ambas carpetas se encuentran bajo el directorio donde esté almacenado el proyecto. El archivo líneas.txt debe crearse y puede guardarse en la ubicación que el usuario desee. El tiempo de simulación es un valor real mayor a cero.

Cuando se presiona finalizar, el módulo de manipulación invoca al ejecutable del simulador y queda esperando a que el proceso que lo atiende termine. Mientras tanto, se visualiza la simulación como se muestra en la figura 5.2.3.8. Una vez que termina, retoma la ejecución normal del módulo de manipulación y despliega un mensaje indicando que la solución fue evaluada.

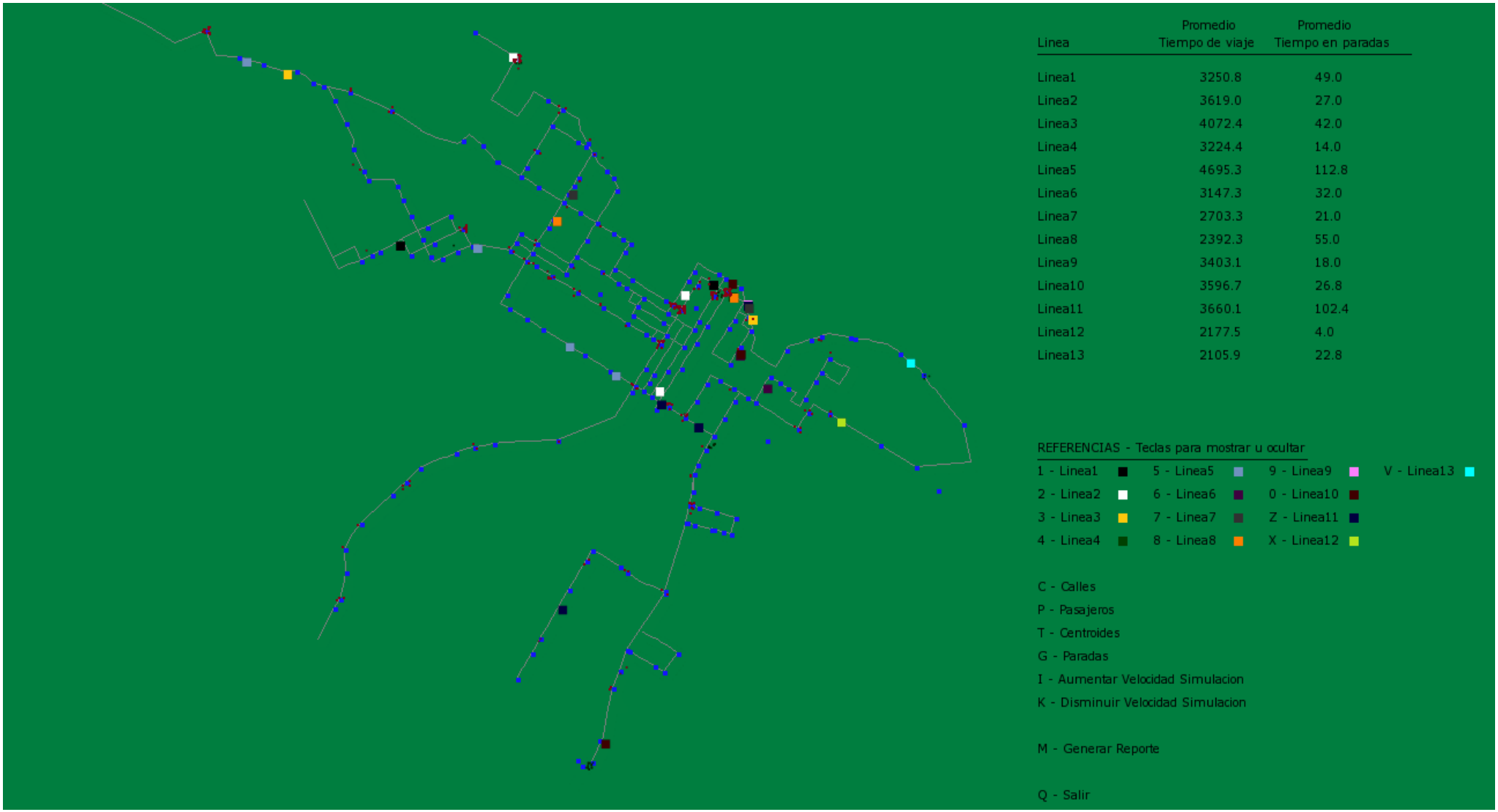


Figura 5.2.3.8: La figura muestra la ejecución del Simulador

5.3. Resultado del experimento

Una vez construido el caso de estudio, se lanza el simulador realizando una única replicación independiente y utilizando siempre la misma semilla. Luego de finalizada se obtienen los siguientes resultados:

Líneas	Tiempo Promedio de Viaje	Flota
Comeri Estiva	57	3
Escuela	57	2
Lagunón	49	3
Pedrera	50	3
Paso de la Estiva	64	2
Pueblo Nuevo	41	2
Quintas al Norte	49	2
Rivera Chico	77	2
Santa Teresa	77	5
Empalme	57	3
Mandubí	58	4
Puerto Seco 5 Bocas A	46	1
Puerto Seco 5 Bocas B	44	1

Tabla 5.3.1

En la tabla 5.3.1 se observan los tiempos de viajes promedio de las distintas líneas (en minutos) y la cantidad de ómnibus necesarios para poder cubrir la frecuencia de cada una.

La información referente a los tiempos de viajes promedios es tomada del sector de datos ubicada en la interfaz del simulador. El mismo se encuentra detallado en la sección 4.8.

Por otro lado, la información de la cantidad de flota necesaria para cubrir la frecuencia de cada línea se obtiene del reporte generado por el simulador una vez que finaliza su ejecución. Esto se encuentra detallado en la sección 4.9.

	Tiempo promedio de viaje (segundos)
	Simulador
Z1	**435**

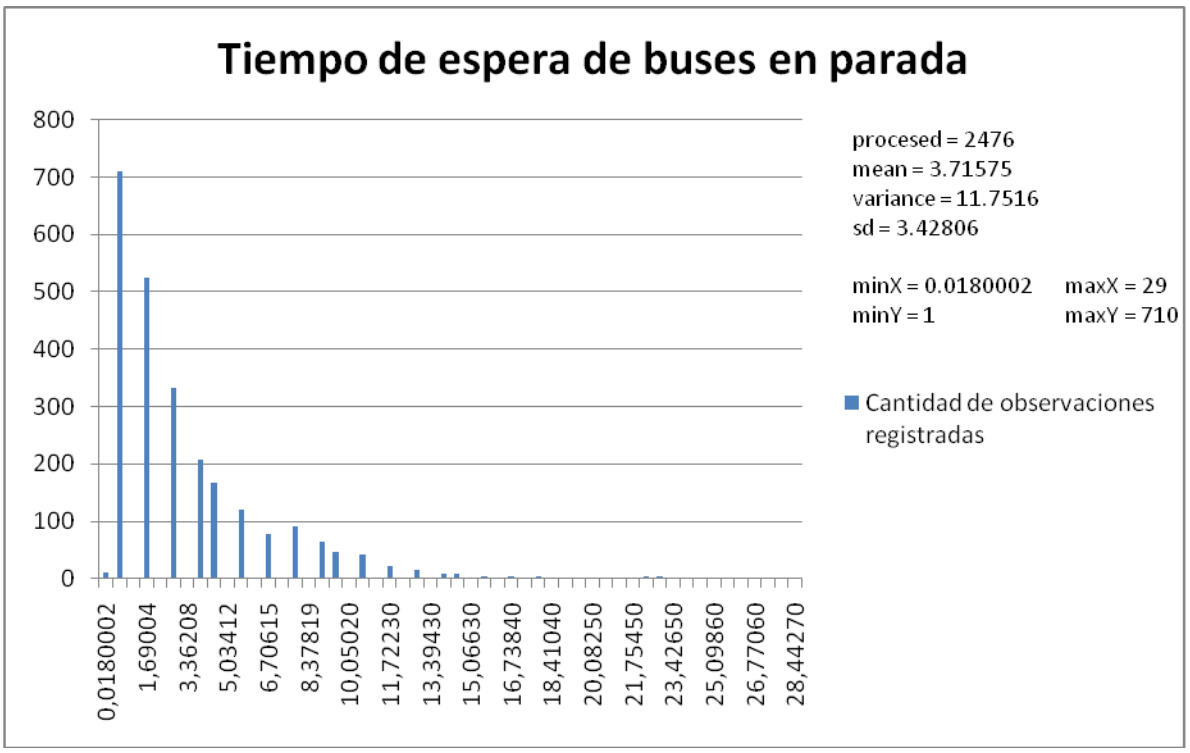
Tabla 5.3.2

También se obtuvo como resultado el valor de Z1 (tiempo promedio de viaje de los pasajeros) el cual se puede observar en la tabla 5.3.2. El cálculo de Z1 se especifica en la sección 4.9.

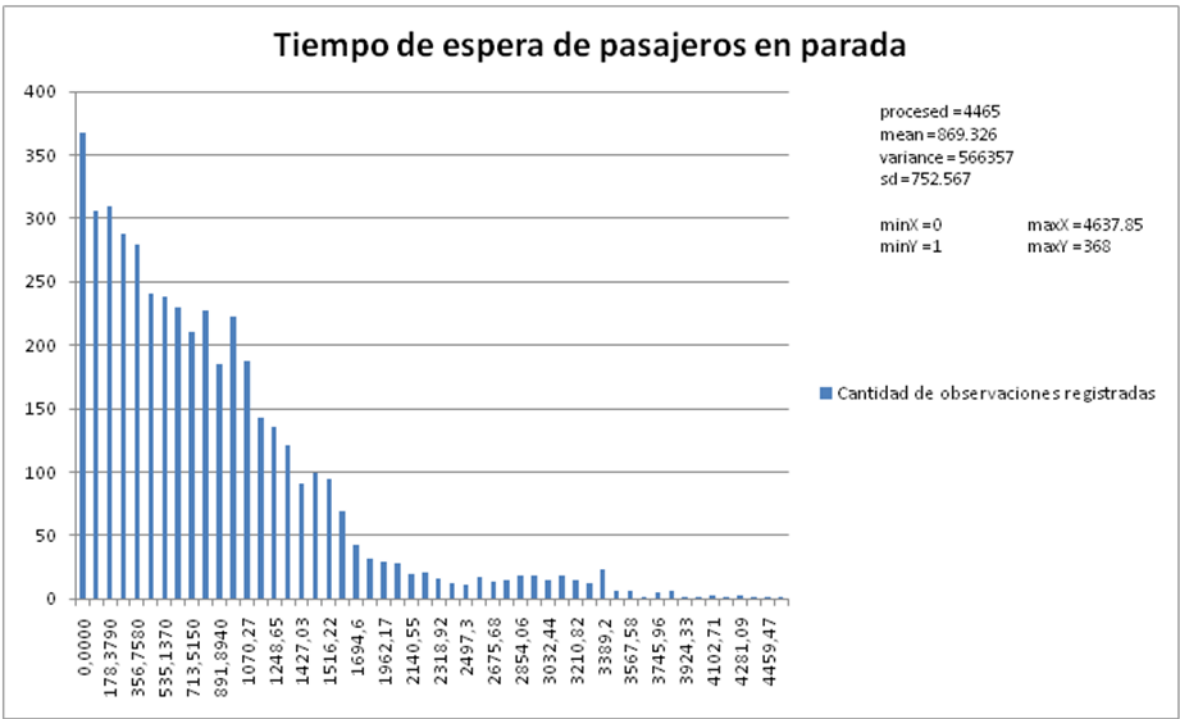
Con los resultados de la tabla 5.3.1 y 5.3.2 podemos observar que los mismos son parecidos a los resultados obtenidos en [36], mas allá de que este experimento haya sido realizado bajo condiciones diferentes. Los tiempos de caminata, así como también los tiempos de subida y bajada de los pasajeros a los ómnibus, no son nulos en este experimento. Estos resultados nos permiten realizar una validación preliminar del simulador.

5.3.1 Histogramas

En los histogramas 5.3.1.1 y 5.3.1.2 se miden los tiempos de espera de los ómnibus en las paradas a lo largo de sus viajes y los tiempos de espera de los pasajeros en las paradas respectivamente. El tiempo en ambos casos se mide en segundos.



Histograma 5.3.1.1



Histograma 5.3.1.2

5.3.2 Interpretación de los resultados

En el histograma 5.3.1.1 se puede observar los tiempos de espera de los ómnibus en las paradas. Estos resultados están atados a los valores de los tiempos de subida y bajada de los pasajeros. Gran parte de los ómnibus demoran poco tiempo esperando en las paradas. Esto se traduce a que sólo unos pocos pasajeros toman un determinado ómnibus a la vez.

Es importante notar que la espera de los ómnibus nunca superó los 29 segundos y que en promedio el tiempo de espera ronda en los 3.7 segundos.

En el histograma 5.3.1.2 se puede observar los tiempos de espera de los pasajeros en las paradas. Como conclusión se puede ver que hay una cantidad grande de pasajeros cuyo tiempo de espera está entre los 0 y 178 segundos. En promedio, el tiempo de espera de los pasajeros ronda en los 869 segundos (14 minutos aproximadamente). Es interesante observar que hay un porcentaje menor de pasajeros que esperan alrededor de los 4000 segundos. Es razonable que el tiempo no supere dicho valor ya que los mayores tiempos entre pasadas planificados son de 60 minutos. Estas situaciones se dan, por lo general, cuando a un pasajero le sirve una única línea para llegar a su destino y a su vez le sirve únicamente el recorrido de vuelta. El pasajero arriba a la parada, pasa la línea en su recorrido de ida (ya que no le sirve para la ida, no lo toma) y recién cuando pasa nuevamente, en su recorrido de vuelta, lo toma.

5.4. Validación

Como estrategia para la validación del simulador se optó por el uso de datos estadísticos de diferentes fuentes para su comparación, de forma de confirmar que la simulación produce resultados similares. Dichas comparaciones fueron realizadas contra datos reales y el modelo de asignación de Baaj y Mahmassani [55].

5.4.1. Comparación con datos reales y modelo de asignación

Con el objetivo de hacer una validación del modelo del simulador construido, realizamos una comparación de los resultados del caso de estudio con datos reales y con los resultados del modelo de asignación de Baaj y Mahmassani, tomados de [36].

Para poder llevar a cabo esta comparación tuvimos que adaptar algunos datos de entrada de modo que se asemeje al modelo del cual se obtuvieron los resultados con los que vamos a comparar:

- Tiempo de caminata cero: Para realizar esto, se editó el atributo costo de la capa de caminatas asignándole el valor cero para cada una de éstas. Esto se realiza con el objetivo de generar nuevamente el archivo grafo.txt, que sirve como entrada para el simulador.

- Tiempo de subida y bajada cero: Los tiempos de bajada y subida de los pasajeros a los ómnibus son seteados con el valor cero.

- Tiempo de cambio de línea Ida y Vuelta cero: El tiempo que demora una línea de "Ida Y Vuelta" en pasar del recorrido de Ida al de Vuelta es seteado en cero.

- Capacidad del bus: Se considera una capacidad máxima de 28 pasajeros sentados y 14 pasajeros parados por ómnibus.

- Velocidad del bus: Se considera que la velocidad del ómnibus es de 13,6 km/h.

Bajo estas condiciones se realiza un nuevo experimento. En las tablas 5.4.1.1, 5.4.1.2 y 5.4.1.3, se comparan los resultados obtenidos por el simulador, los datos reales y los resultados provenientes del modelo de asignación.

Línea	Tiempo promedio de viaje		
	Simulador	Real	Asignación
Empalme	**57**	**60**	**67**
La Pedrera	**56**	**60**	**64**
PuebloNuevo	**48**	**60**	**55**
Comeri-Estiva	**49**	**60**	**49**
SantaTeresa	**63**	**60**	**70**
RiveraChico	**40**	**60**	**50**
Quintas al Norte	**49**	**60**	**46**
Lagunón	**77**	**60**	**74**
Paso de la Estiva	**70**	**60**	**108**
Escuela 88	**56**	**60**	**63**
Mandubí	**57**	**60**	**62**
PuertoSeco5BocasA	**44**	**40**	**61**
PuertoSeco5BocasB	**44**	**40**	**47**

Tabla 5.4.1.1

Línea	Flota		
	Simulador	Real	Asignación
Empalme	**3**	**2**	**3**
La Pedrera	**2**	**1**	**2**
PuebloNuevo	**3**	**2**	**2**
Comeri-Estiva	**3**	**2**	**2**
SantaTeresa	**2**	**1**	**2**
RiveraChico	**2**	**2**	**2**
Quintas al Norte	**2**	**2**	**2**
Lagunón	**2**	**1**	**2**
Paso de la Estiva	**5**	**3**	**6**
Escuela 88	**3**	**3**	**3**
Mandubí	**4**	**3**	**4**
PuertoSeco5BocasA	**1**	**1**	**2**
PuertoSeco5BocasB	**1**	**1**	**1**

Tabla 5.4.1.2

	Flota		
	Simulador	Real	Asignación
Flota total	**33**	**23**	**33**

Tabla 5.4.1.3

Otra variable que utilizamos para comparar el Simulador con el modelo de asignación, es el tiempo total de viaje de los pasajeros (Z_1) y se puede observar la comparación en la tabla 5.4.1.4. Esta variable fue explicada en el punto 4.9.

	Tiempo promedio de viaje (segundos)	
	Simulador	Asignación
Z1	**371.4**	**406**

Tabla 5.4.1.4

Las variaciones que se dan entre los distintos resultados pueden estar dados porque:

- El modelo de red utilizado es distinto. Para el caso del modelo de asignación se consideran como nodos viales los centroides y como calles las conexiones entre esos centroides. Mientras que para el simulador, se consideran como nodos viales las paradas y los cruces, y como calles las conexiones entre los mismos. En el caso de simulador los centroides son utilizados únicamente para modelar las caminatas a las distintas paradas. En estas condiciones, los recorridos de las distintas líneas de ómnibus son definidos sobre modelos diferentes. Posiblemente ésta sea la principal razón por el cual el valor de Z1 en el simulador es menor que la del modelo de asignación. Esto se acentúa con las diferencias existentes en los tiempos de viajes de la tabla 5.4.1.1. En general se puede observar que los tiempos de viaje de los ómnibus en el modelo de asignación son significativamente mayores que los del simulador. Esto hace que suceda lo mismo con los tiempos de viaje a bordo de los pasajeros y en consecuencia en Z1.

- Capas diferentes. Una posible razón para que los resultados obtenidos con el simulador y el modelo de asignación sean diferentes es que las capas de paradas y de red vial son distintas. A todo esto, sumamos el hecho de que para el caso del simulador se tuvieron que agregar paradas de forma estratégica para que no hayan zonas sin paradas y también para que todos los recorridos tengan al menos una parada en todas las zonas por las que pasa.

- La estrategia utilizada por los pasajeros. La utilización de diferentes estrategias por parte de los pasajeros a la hora de elegir que ómnibus tomar, provoca cambios en los resultados. En el caso del simulador, tendería a provocar que los tiempos de espera de los pasajeros sean mayores respecto al modelo de asignación. Esto no se ve reflejado en Z1, posiblemente, porque las diferencias entre los modelos de red

(observado en el primer ítem) sea más fuerte y haga una diferencia mayor en los tiempos de espera a favor del modelo de asignación, que la elección de la estrategia utilizada.

- En el simulador, el tiempo de viaje entre cada par de nodos viales es producto de una distribución Normal, cuya media es igual al tiempo que hay entre cada par de nodos viales. En este sentido, existen diferencias en estos tiempos por lo que repercuten en los resultados finales comparándolos con la realidad y con el modelo de asignación.

- Utilizamos una distribución sobre la matriz de demanda. Al aplicarle una distribución a la demanda de los pasajeros en los distintos centroides, se generan ciertas diferencias en los tiempos de creación de pasajeros.

- Estado inicial. Cuando se inicia la simulación se lanzan al mismo tiempo todas las líneas de ómnibus, siguiendo luego cada una su respectiva frecuencia. Esto en la realidad no es así por lo que este punto puede provocar algunas diferencias.

- Subida y bajada de pasajeros. Más allá de que los tiempos de subida y bajada de pasajeros en el simulador fueron seteados en cero (es una constante que se puede modificar) para asemejarlo a las condiciones del modelo de asignación, es claro que esto en la realidad no sucede. Es más, en la realidad el tiempo de subida y bajada de las personas depende en gran medida del tipo de persona (adulto, joven, lesionado, etc.

5.5. Experimentos con el caso de estudio

Tomando como base los datos del caso de estudio de la ciudad de Rivera y utilizando los resultados obtenidos en la sección 5.3 (Resultado del experimento) como referencia para la comparación, bajo las mismas condiciones experimentales, realizamos 4 experimentos. La situación base para la comparación considera que los tiempos de subida y bajada de pasajeros son de 1km/h, el tiempo que demora cualquier línea de “Ida y vuelta” en pasar del recorrido de ida al de vuelta es nulo, los tiempos de caminatas no son nulos y además todos los ómnibus poseen una capacidad máxima de 28 pasajeros sentados y 14 parados.

Estos experimentos fueron realizados con el objetivo de poder mostrar la utilización del simulador y así observar distintos resultados de comportamiento que se pueden obtener, variando ciertos parámetros de entrada. Con esto reforzamos la validación del simulador y nos aseguramos que el modelo produce resultados razonables cuando los factores e hipótesis son modificados.

5.5.1. Experimento 1

Este primer experimento consistió en aumentar la frecuencia de las distintas líneas de ómnibus en un 20%. Para eso fue necesario cambiar el archivo líneas.txt y correr nuevamente el simulador. Este nuevo archivo es el siguiente:

0	1	2	0	1440
0	3	4	0	1440
0	5	6	0	2880
0	7	8	0	2880
0	9	10	0	960
0	11	12	0	1440
0	13	14	0	1440
0	15	16	0	1440
0	17	21	0	2880
0	19	20	0	1440
0	22	23	0	960
1	18	2880		
1	24	2880		

Una vez finalizada la simulación se obtuvieron nuevos resultados. En primer lugar, observando la Tabla 5.5.1.1, vemos que los tiempos promedio de viaje de la mayoría de las líneas disminuyeron o se mantuvieron iguales. Esto se debe a que se despachan más ómnibus en menos tiempo, provocando que cada uno de estos demore menos tiempo en las paradas subiendo y bajando pasajeros y en consecuencia, se demore menos tiempo en terminar cada viaje.

En las tablas 5.5.1.2 y 5.5.1.3, observamos que ante el aumento de la frecuencia de las distintas líneas, es necesario mayor cantidad de ómnibus para cubrirlas. Este resultado era esperable ya que se necesitan despachar más ómnibus en menos tiempo. En total, vemos que se necesitan 6 ómnibus más para cubrir las frecuencias.

Tiempo promedio de viaje		
Línea	Situación base	Experimento 1
Empalme	**57**	**57**
La Pedrera	**57**	**56**
PuebloNuevo	**49**	**49**
Comeri-Estiva	**50**	**49**
SantaTeresa	**64**	**57**
RiveraChico	**41**	**41**
Quintas al Norte	**49**	**48**
Lagunón	**77**	**77**
Paso de la Estiva	**77**	**76**
Escuela 88	**57**	**57**
Mandubí	**58**	**58**
PuertoSeco5BocasA	**46**	**45**
PuertoSeco5BocasB	**44**	**40**

Tabla 5.5.1.1

Flota		
Línea	Situación base	Experimento 1
Empalme	**3**	**3**
La Pedrera	**2**	**2**
PuebloNuevo	**3**	**3**
Comeri-Estiva	**3**	**3**
SantaTeresa	**2**	**2**
RiveraChico	**2**	**3**
Quintas al Norte	**2**	**3**
Lagunón	**2**	**2**
Paso de la Estiva	**5**	**6**
Escuela 88	**3**	**4**
Mandubí	**4**	**5**
PuertoSeco5BocasA	**1**	**2**
PuertoSeco5BocasB	**1**	**2**

Tabla 5.5.1.2

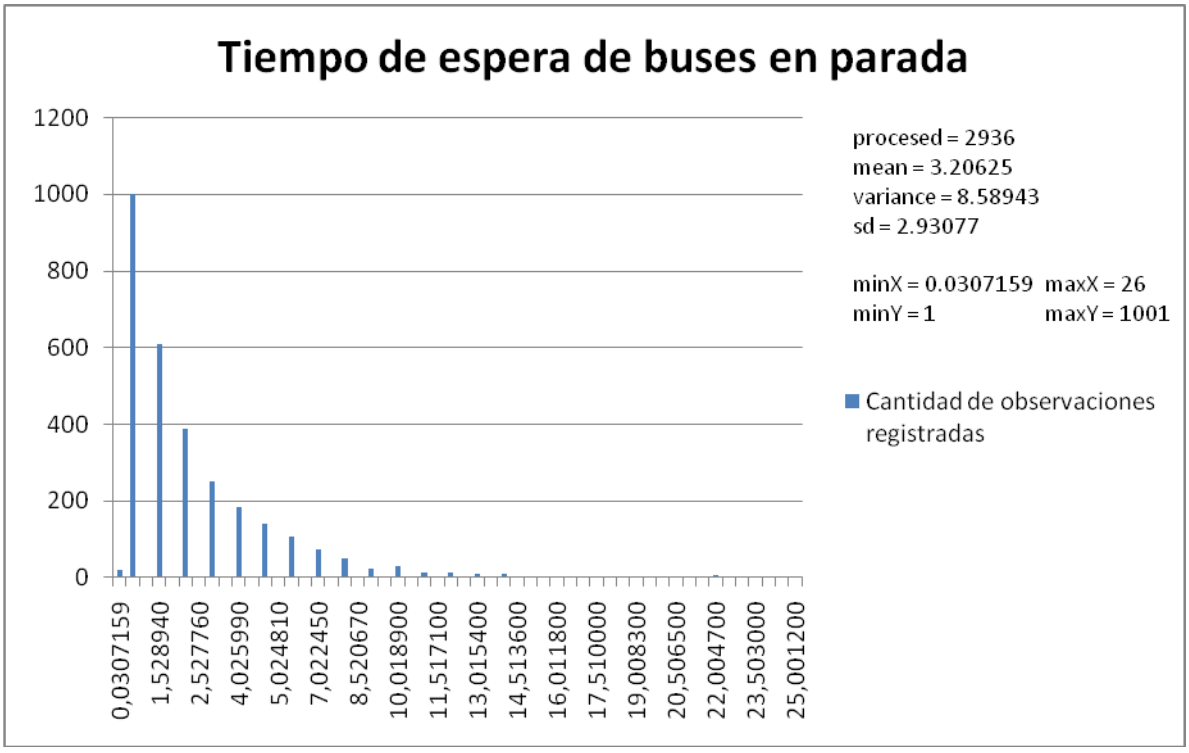
	Flota	
	Situación base	Experimento 1
Flota total	**33**	**39**

Tabla 5.5.1.3

Con respecto a los histogramas (5.5.1.4 y 5.5.1.5), se puede observar que con el aumento en las frecuencias de las líneas de ómnibus, el tiempo de espera promedio de los pasajeros en las paradas disminuye de 14 minutos a 11 minutos. Esto se debe a que se despachan una mayor cantidad de ómnibus en menos tiempo, por lo que cada pasajero tendrá que esperar menos tiempo para tomárselo.

Por otro lado, el tiempo de espera promedio de todos los ómnibus en las paradas disminuyó de 3.7 a 3.2 segundos.

Histograma 5.5.1.4



Histograma 5.5.1.5

5.5.2. Experimento 2

Este segundo experimento consistió en disminuir la frecuencia de las distintas líneas de ómnibus en un 20%. Para eso fue necesario cambiar el archivo líneas.txt y correr nuevamente el simulador. Este nuevo archivo es el siguiente:

0	1	2	0	2160
0	3	4	0	2160
0	5	6	0	4320
0	7	8	0	4320
0	9	10	0	1440
0	11	12	0	2160
0	13	14	0	2160
0	15	16	0	2160
0	17	21	0	4320
0	19	20	0	2160
0	22	23	0	1440
1	18	4320		
1	24	4320		

Una vez finalizada la simulación se obtuvieron nuevos resultados. En primer lugar, observando la Tabla 5.5.2.1 vemos que los tiempos promedio de viaje de las distintas líneas de ómnibus aumentaron. Esto es debido a que al disminuir la frecuencia de salida de los ómnibus, menos ómnibus de cada línea van a pasar en un mismo tiempo, haciendo que se aglomere una mayor cantidad de pasajeros en las paradas y en consecuencia cada ómnibus demore más tiempo subiendo pasajeros. Asimismo, al subir más pasajeros, cada ómnibus va a tener más pasajeros para bajar en las paradas, contribuyendo a aumentar el tiempo promedio de viaje del mismo.

En las tablas 5.5.2.2 y 5.5.2.3 vemos que necesitamos una cantidad igual o menor de ómnibus para cubrir las distintas líneas. Este resultado era esperable ya que es necesario despachar menos ómnibus en un mismo período de tiempo. Cabe recordar que se agrega un nuevo ómnibus a la flota, si no hay ómnibus libres para despachar cuando la frecuencia lo disponga. Teniendo en cuenta esto, la disminución de la frecuencia provoca que haya ómnibus que terminen sus viajes antes que la frecuencia disponga lanzar otro ómnibus, reutilizando así los mismos y evitando agregar nuevos a la flota. Dado los resultados vemos que bajo estas condiciones se podrían disminuir en 7 la cantidad total de la flota.

Línea	Tiempo promedio de viaje	
	Situación base	Experimento 2
Empalme	**57**	**63**
La Pedrera	**57**	**60**
PuebloNuevo	**49**	**52**
Comeri-Estiva	**50**	**56**
SantaTeresa	**64**	**65**
RiveraChico	**41**	**45**
Quintas al Norte	**49**	**52**
Lagunón	**77**	**78**
Paso de la Estiva	**77**	**84**
Escuela 88	**57**	**65**
Mandubí	**58**	**60**
PuertoSeco5BocasA	**46**	**42**
PuertoSeco5BocasB	**44**	**33**

Tabla 5.5.2.1

Línea	Flota	
	Situación base	Experimento 2
Empalme	**3**	**2**
La Pedrera	**2**	**1**
PuebloNuevo	**3**	**2**
Comeri-Estiva	**3**	**2**
SantaTeresa	**2**	**1**
RiveraChico	**2**	**2**
Quintas al Norte	**2**	**2**
Lagunón	**2**	**2**
Paso de la Estiva	**5**	**4**
Escuela 88	**3**	**3**
Mandubí	**4**	**3**
PuertoSeco5BocasA	**1**	**1**
PuertoSeco5BocasB	**1**	**1**

Tabla 5.5.2.2

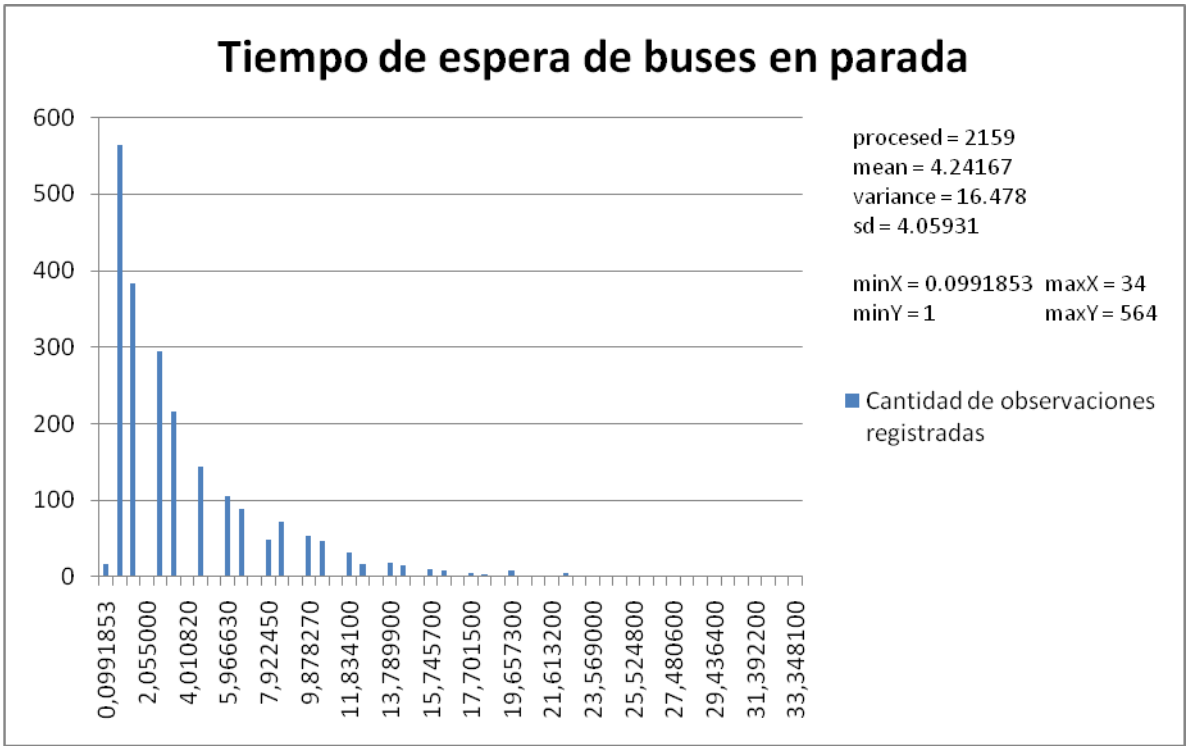
	Flota	
	Situación base	Experimento 2
Flota total	**33**	**26**

Tabla 5.5.2.3

Con respecto a los histogramas (5.5.2.4 y 5.5.2.5), se puede observar que ante la disminución de la frecuencia de las líneas de ómnibus, el tiempo de espera promedio de los pasajeros en las paradas aumenta de 14 minutos a 17 minutos. Esto se debe a que se despachan menos ómnibus en relación al tiempo, por lo que cada pasajero tendrá que esperar más tiempo para tomárselo.

Por otro lado el tiempo de espera promedio de todos los ómnibus en las paradas aumentó de 3.7 a 4.2 segundos.

Histograma 5.5.2.4



Histograma 5.5.2.5

5.5.3. Experimento 3

Este tercer experimento consistió en aumentar cada uno de los elementos de la matriz de demanda de pasajeros en un 5%. Con esto se intenta determinar cuál es el comportamiento de la simulación ante un posible aumento de la demanda de pasajeros.

Una vez finalizada la simulación se obtuvieron nuevos resultados. En primer lugar, observando la Tabla 5.5.3.1 vemos que los tiempos promedios de viaje de las distintas líneas de ómnibus aumentaron o permanecieron iguales. Esto es debido a que al aumentar la demanda de pasajeros, las paradas se van a ver afectadas con una mayor cantidad de pasajeros y en consecuencia, cada ómnibus va a permanecer más tiempo esperando para subirlos y/o bajarlos, contribuyendo a aumentar el tiempo promedio de viaje de los mismos.

En las tablas 5.5.3.2 y 5.5.3.3 vemos que más allá de aumentar los tiempos promedios de viaje, no es necesario agregar más ómnibus a las distintas líneas para cubrir la demanda. Este resultado se da porque el aumento de la demanda de pasajeros no es lo suficientemente grande como para requerir aumentar la cantidad de la flota.

Línea	Tiempo promedio de viaje	
	Situación base	Experimento 3
Empalme	**57**	**59**
La Pedrera	**57**	**59**
PuebloNuevo	**49**	**53**
Comeri-Estiva	**50**	**52**
SantaTeresa	**64**	**65**
RiveraChico	**41**	**42**
Quintas al Norte	**49**	**51**
Lagunón	**77**	**78**
Paso de la Estiva	**77**	**78**
Escuela 88	**57**	**58**
Mandubí	**58**	**60**
PuertoSeco5BocasA	**46**	**46**
PuertoSeco5BocasB	**44**	**44**

Tabla 5.5.3.1

Línea	Flota	
	Situación base	Experimento 3
Empalme	**3**	**3**
La Pedrera	**2**	**2**
PuebloNuevo	**3**	**3**
Comeri-Estiva	**3**	**3**
SantaTeresa	**2**	**2**
RiveraChico	**2**	**2**
Quintas al Norte	**2**	**2**
Lagunón	**2**	**2**
Paso de la Estiva	**5**	**5**
Escuela 88	**3**	**3**
Mandubí	**4**	**4**
PuertoSeco5BocasA	**1**	**1**
PuertoSeco5BocasB	**1**	**1**

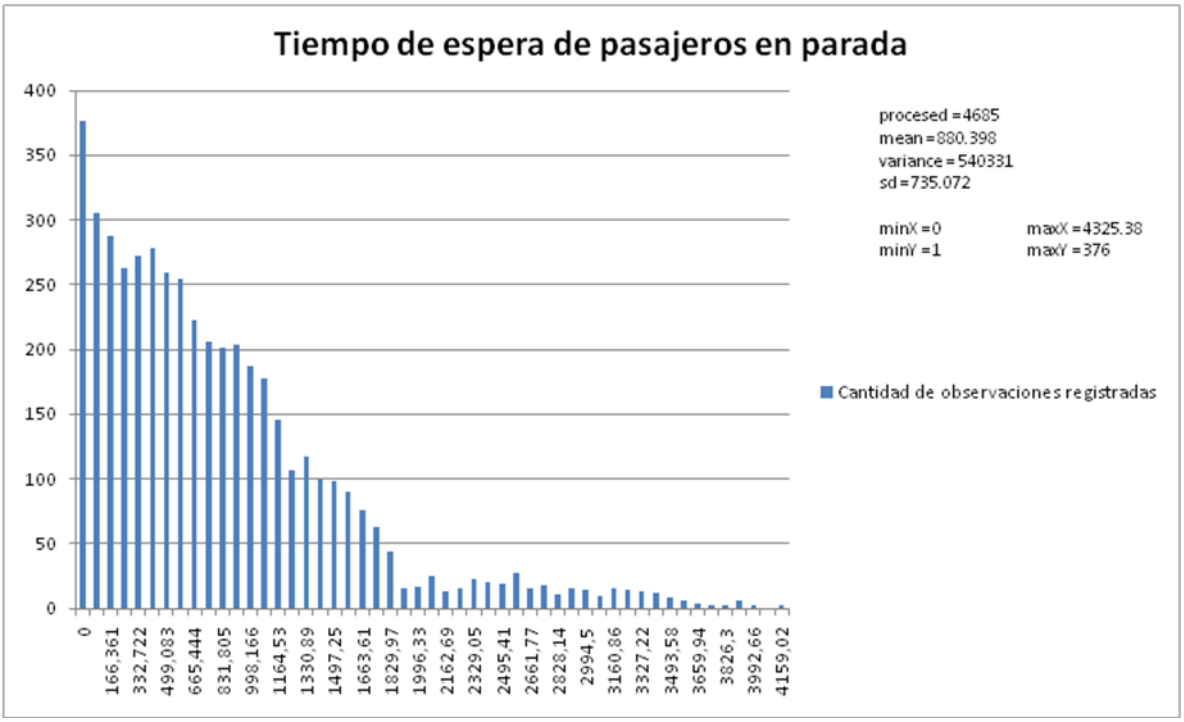
Tabla 5.5.3.1

	Flota	
	Situación base	Experimento 3
Flota total	**33**	**33**

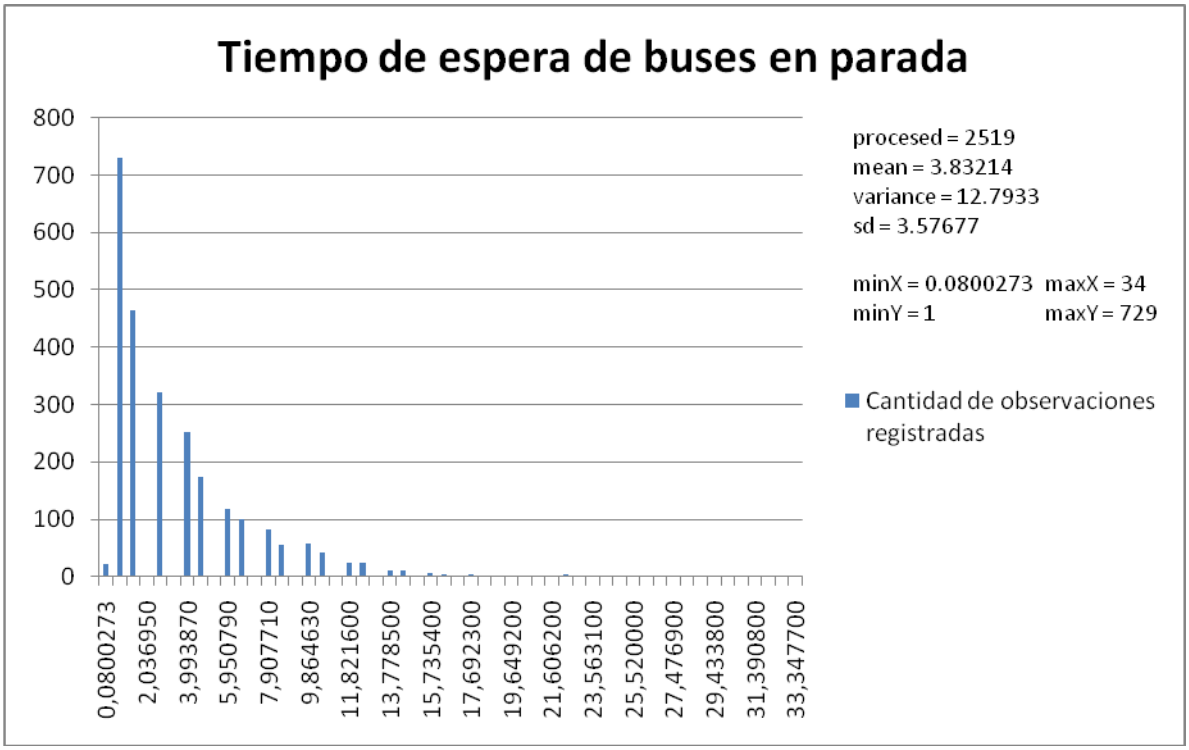
Tabla 5.5.3.3

Con respecto a los histogramas (5.5.3.4 y 5.5.3.5), se observa que con el aumento de la demanda de pasajeros, el tiempo de espera promedio de los mismos en las paradas permanece prácticamente incambiado en 14 minutos. Observando en detalle los resultados, se ve que el aumento es de tan solo 11 segundos en promedio. Esto se debe a que el aumento de la demanda no es lo considerablemente grande como para distorsionar los resultados.

Por otro lado, el tiempo de espera promedio de todos los ómnibus en las paradas aumentó de 3.7 a 3.8 segundos debido a que, al aumentar la demanda de pasajeros, cada ómnibus va a demorar más tiempo promedialmente para que suban y bajen los pasajeros en sus respectivas paradas.



Histograma 5.5.3.4



Histograma 5.5.3.5

5.5.4. Experimento 4

Este cuarto experimento consistió en aumentar cada uno de los elementos de la matriz de demanda de pasajeros en un 5% más sobre aumento realizado en el experimento 3. Con esto se intenta determinar cuál es el comportamiento de la simulación aumentando la demanda de pasajeros y también comparar los resultados con los obtenidos en el experimento 3.

Una vez finalizada la simulación se obtuvieron nuevos resultados. En primer lugar, observando la Tabla 5.5.4.1 vemos que los tiempos promedios de viaje de las distintas líneas de ómnibus aumentaron en mayor proporción. Esto es debido a que al aumentar la demanda de pasajeros, las paradas se van a ver afectadas con una mayor cantidad de pasajeros y en consecuencia, cada ómnibus va a permanecer más tiempo esperando para subirlos y/o bajarlos, contribuyendo a aumentar el tiempo promedio de viaje de los mismos.

En las tablas 5.5.4.2 y 5.5.4.3 vemos que hay dos líneas que necesitan más ómnibus para cubrir las frecuencias. Con las modificaciones sobre la demanda de pasajeros, vemos que se necesitan dos ómnibus más en total para cubrir las frecuencias de las líneas.

Línea	Tiempo promedio de viaje		
	Situación base	Exp. 3	Exp. 4
Empalme	**57**	**59**	**62**
La Pedrera	**56**	**59**	**62**
PuebloNuevo	**48**	**53**	**53**
-Comeri-Estiva	**49**	**52**	**56**
SantaTeresa	**63**	**65**	**70**
RiveraChico	**40**	**42**	**46**
Quintas al Norte	**49**	**51**	**51**
Lagunón	**77**	**78**	**82**
Paso de la Estiva	**70**	**78**	**83**
Escuela 88	**56**	**58**	**65**
Mandubí	**57**	**60**	**65**
PuertoSeco5BocasA	**44**	**46**	**49**
PuertoSeco5BocasB	**44**	**44**	**47**

Tabla 5.5.4.1

Línea	Flota		
	Situación base	Exp. 3	Exp. 4
Empalme	**3**	**3**	**3**
La Pedrera	**2**	**2**	**2**
PuebloNuevo	**3**	**3**	**3**
Comeri-Estiva	**3**	**3**	**3**
SantaTeresa	**2**	**2**	**2**
RiveraChico	**2**	**2**	**2**
Quintas al Norte	**2**	**2**	**2**
Lagunón	**2**	**2**	**2**
Paso de la Estiva	**5**	**5**	**6**
Escuela 88	**3**	**3**	**4**
Mandubí	**4**	**4**	**4**
PuertoSeco5BocasA	**1**	**1**	**1**
PuertoSeco5BocasB	**1**	**1**	**1**

Tabla 5.5.4.2

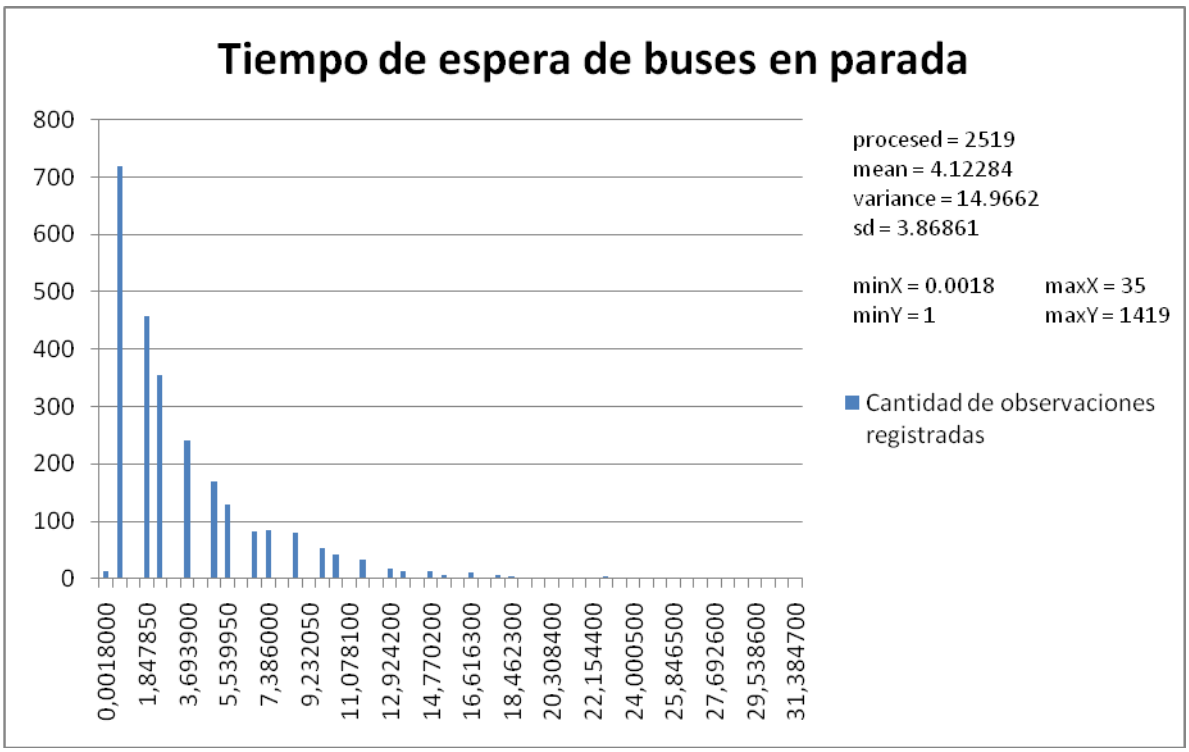
	Flota		
	Situación base	Exp. 3	Exp. 4
Flota total	**33**	**33**	**35**

Tabla 5.5.4.3

Con respecto a los histogramas (5.5.4.4 y 5.5.4.5), se observa que con el aumento de la demanda de pasajeros, el tiempo de espera promedio de los mismos en las paradas permanece prácticamente incambiado en 14 minutos. Un posible motivo por el cual ocurre esto, es porque ninguno de los ómnibus se completa como para no permitir subir pasajeros.

Por otro lado, el tiempo de espera promedio de todos los ómnibus en las paradas aumentó de 3.8 a 4.1 segundos en comparación con el experimento 3 debido a que, al aumentar la demanda de pasajeros con respecto al experimento anterior, cada ómnibus va a demorar más tiempo promedialmente para que suban y bajen los pasajeros en sus respectivas paradas.

Histograma 5.5.4.4



Histograma 5.5.4.5

Dados todos los experimentos realizados y las conclusiones sacadas de cada uno de los resultados, se pudo validar el correcto comportamiento del sistema y asegurarnos que ante diferentes cambios estratégicos de entrada de datos al simulador (como ser cambios en frecuencias de líneas y en la demanda de pasajeros), el modelo produce resultados que son razonables.

6. Conclusiones y trabajos futuros

Se cumplieron los objetivos del proyecto. Más concretamente, se construyó un simulador a eventos discretos que modela la interacción de los pasajeros con los buses, dado un diseño del sistema de transporte público, un escenario particular de demanda de pasajeros y una determinada configuración de líneas y frecuencias. El sistema presenta los resultados de forma que el usuario pueda utilizarlos como apoyo a la toma de decisiones en la planificación de recorridos del transporte público urbano. Consideramos importante haber desarrollado el simulador de forma genérica ya que hace que el mismo sea extensible, pudiendo agregar nuevas estrategias para los pasajeros, nuevas reglas operativas para los ómnibus y nuevos atributos de forma de poder experimentar de forma sencilla con distintas realidades.

El simulador se integró a IgoR-tp v2.0. Esta herramienta es una nueva versión de IgoR-tp v1.0, la cual soporta una visión más detallada del modelo de red. Destacamos la importancia del desarrollo de esta nueva versión de igoR-tp, ya que con la versión 1.0, el simulador perdería realismo debido a que no se podrían modelar los comportamientos de las caminatas de los pasajeros a las paradas y además los ómnibus no viajarían por la red vial sino que lo harian por las conexiones entre centroides.

Utilizando igoR-tp v2.0 y el simulador, se construyó un caso de estudio real, referente a la ciudad de Rivera con el cual se validaron las herramientas construidas. El resultado obtenido en la evaluación del caso de estudio, es similar al resultado obtenido por el modelo de asignación de Baaj y Mahmassani y al resultado de la solución relevada del sistema de transporte público de Rivera. Esto, junto a todos los experimentos realizados, nos permitió verificar y validar el correcto funcionamiento del simulador.

El desarrollo de ambos sistemas implicó el trabajo con distintas tecnologías, aportándonos experiencia en el uso de las mismas. No se encontraron limitantes por parte de éstas a la hora de implementar todos los requerimientos de los sistemas.

Un aporte que destacamos de este proyecto es que el simulador no sólo muestra los resultados obtenidos, sino que también permite visualizar gráficamente el comportamiento del sistema. Otro gran aporte es que con las herramientas construidas, es posible experimentar diversas soluciones con un menor costo que si lo implementaran en la práctica. No sólo porque con la herramienta se pueden obtener los resultados en menor tiempo, sino que también en algunos casos llevarlo a cabo en la realidad podría ser muy costoso o incluso imposible.

Trabajos futuros

En un trabajo futuro sugerimos las siguientes consideraciones sobre el modelo de red utilizado:

- Sería bueno tener la flexibilidad de ubicar las paradas sobre la red vial según su localización física, en cualquier sector de la calle y no sólo en los cruces de calles como se hace actualmente. Representando las paradas en los cruces de calles permite tomar ómnibus en 4 direcciones (como muestra la figura 6.1). En cambio, representando la parada sobre la calle, a cierta distancia del cruce, se permite tomar ómnibus en sólo 2 direcciones (como muestra la figura 6.2).

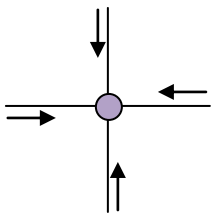


Figura 6.1

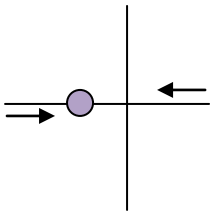


Figura 6.2

- Otro aspecto a mejorar sería que las caminatas entre paradas se modelen por la red vial y no en línea recta. Esto traería como consecuencia un modelo más detallado que represente más fielmente la realidad.
- Sería conveniente mejorar la lógica de los centroides permitiendo mover los centroides para que se adapte mejor a la realidad. Por ejemplo, si en una zona hay un hospital pero este no se encuentra en el baricentro de la misma, sería bueno mover el centroide para que refleje la gran demanda hacia ese punto. Otra consideración, sería poder ubicar más de un centroide por zona o manejar otro concepto como el de puntos de atracción. Un ejemplo de esto es cuando en una zona hay un hospital, un banco y una escuela ubicados distanciadamente dentro de la misma zona. Creemos conveniente que el modelo de red para un trabajo futuro permita modelar estos casos. Este cambio impacta en la estructura de la matriz origen destino, la cual debe cambiar para modelar la demanda hacia y desde los puntos de atracción de las zonas.

En cuanto a las funcionalidades de igoR-tp destacamos las siguientes modificaciones o nuevos requerimientos que serían buenos incluirlos en un trabajo futuro:

- Permitirle al usuario trabajar con distintos modelos de red en igoR-tp. Por ejemplo, que al crear un nuevo proyecto en el módulo de construcción, tenga la posibilidad de elegir entre el modelo de red utilizado en igoR-tp v1.0 y el utilizado en igoR-tp v2.0.

- Trabajar con calles flechadas a nivel de la aplicación. El usuario puede trabajar con calles flechadas pero él mismo debe ser responsable de utilizar correctamente esa información. Sería bueno que IgoR-tp pudiera manejar eso, al determinar los costos de la red vial y a la hora de definir los recorridos en el módulo de manipulación.

- Para unificar los conceptos entre igoR-tp y el simulador, sugerimos que el módulo de manipulación de IgoR-tp maneje el concepto de líneas y que cada línea puede tener un recorrido de ida y otro de vuelta. Esto haría más directo y amigable la entrada de la

correspondencia entre las líneas y recorridos, eliminando así el archivo generado por el usuario “líneas.txt”.

Destacamos para el simulador los siguientes trabajos futuros a realizar:

- Una funcionalidad que creemos importante agregar, sería poder realizar zoom en ciertas regiones de la red para obtener una mejor visualización en zonas de interés. Hay ciertas limitaciones tecnológicas con la salida gráfica de EOSimulator que no permiten cumplir con este requerimiento, pero se podría implementar utilizando nativamente SDL como librería gráfica, por lo que ésta puede ser una buena alternativa para sacarle más provecho a la visualización.

- Sería conveniente incluir más datos que interesen estudiar en la generación de los reportes.

- Implementar otras estrategias de forma de generar distintos comportamientos de pasajeros en la simulación. También, sería conveniente implementar reglas operativas para los buses para modelar los comportamientos de los mismos. Tanto los pasajeros como los buses tienen una lista de atributos genéricos. Sería bueno que éstos fueran utilizados como parte de alguna estrategia (en el caso de los pasajeros) o utilizados por reglas operativas (en el caso de los buses). Es importante recordar que el simulador fue desarrollado de forma genérica y las validaciones fueron realizadas sobre la lógica principal del mismo. La validación de todas las extensiones que se deseen realizar quedan a cargo de sus respectivos usuarios.

7. Bibliografía

[1] Aimsun. *http://www.aimsun.com/site*

[2] S. Alaggia, A. Mauttone, M. Urquhart. *EOSimulator: biblioteca de simulación para un curso de simulación a eventos discretos,* 2007.

[3] ArcView. *http://www.esri.com/software/arcview/index.html*

[4] M. Baaj y H. Mahmassani. *TRUST: A LISP program for the analysis of transit route configurations*, 1990.

[5] L. Bañón Blázquez, J. Beviá García. *Manual de Carreteras vol.1,* 2000.

[6] J. Barceló, H. Grzybowska, S. Pardo. *Vehicle routing and scheduling models, Simulation and City Logistics,* 2005.

[7] C++. *http://www.cplusplus.com*

[8] CORSIM. *http://mctrans.ce.ufl.edu/featured/tsis/Version5/corsim50.htm*

[9] CUBE Citilabs. *http://www.citilabs.com/products.html*

[10] Curso de Simulación a Eventos Discretos. *http://www.fing.edu.uy/inco/cursos/simulacion*

[11] EOSimulator. *http://www.fing.edu.uy/inco/cursos/simulacion/eosim_html/index.html*

[12] R. Davies, R. O'Keefe. *Simulation Modelling with Pascal*, 1989.

[13]Distancias Manhattan y Euclidea.
http://www.dma.fi.upm.es/gregorio/JavaGC/DistLink/distaciaLink.htm

[14] G. Desaulniers y M. Hickman. *Public transit*, 2003.

[15] DESMO-J. *http://desmoj.sourceforge.net/home.html*

[16] Dev-C++. *http://www.bloodshed.net/devcpp.html*

[17] Dia. http://projects.gnome.org/dia

[18] DRACULA. *http://www.its.leeds.ac.uk/software/dracula*

[19] DSOL. *http://sk-3.tbm.tudelft.nl/simulation*

[20] EMME/2 INRO. *http://www.inro.ca/en/products/emme/index.php*

[21] ESRI. *http://www.esri.com*

[22] R. Ferández. *Modelling of bus-stop operations. http://www.etcproceedings.org/paper/modelling-of-bus-stop-operations*, 2000.

[23] R. Fernández. *Tecnologías intermedias de transporte público ¿Qué son, cuánto cuestan y qué capacidad ofrecen?*, 2000.

[24] R. Fernández, N. Tyler. *Study of Passenger-Bus-Traffic Interactions on Bus Stop Operations,* 2004.

[25] R. Fernández, C. Cortés, V. Burgos. *Modelación de pasajeros, buses y paraderos en microsimuladores de tráfico,* 2006.

[26] R. Fernández, Vanessa Burgos. *How public transport is being modelled in Microscopic Traffic Simulators?*, 2004.

[27] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns*, 1995.

[28] D.Gawenda, H. Martínez. *Interfaz para herramienta de planificación de recorridos para transporte público*, 2009.

[29] V. Guihaire, Jin-Kao Hao. *Transit Network Design And Scheduling: a Global Review,* 2008.

[30] S. Hall, J. Desyllas. *Bus Stops - How people actually use them and the implications for design,* 2006.

[31] Introducción a los Sistemas de Información Geográfica. *http://www.fing.edu.uy/inco/cursos/sig/ clases/Generalidades160810.pdf*

[32] JavaSim. *http://javasim.codehaus.org*

[33] Y. Lim, H. Kim. *Development of a dynamic traffic simulation model for transit network with heterogeneous passengers*, 2003.

[34] R. Liu, S. Clark, F. Montgomery, D. Watling. *Microscopic modelling of Traffic Management Measures for Guided Bus Operation*, 1998.

[35] MapWindow GIS. *http://www.mapwindow.org*

[36] A. Mauttone. *Optimización de Recorridos y Frecuencias en Sistemas de Transporte Público Urbano Colectivo*, 2005.

[37] A. Mauttone, M. Urquhart. *Una heurística basada en memoria para el problema del diseño de recorridos en transporte público urbano*, 2007.

[38] P. McGinley. *Simulation Software Survey*. *http://www.lionhrtpub.com/orms/surveys/Simulation/Simulation.html*, 2009.

[39] L. Montero, E.Codina, J. Barceló, P. Barceló. *Combining Macroscopic and Microscopic Approaches for Transportation Planning and Design of Road Networks.* http://www.aimsun.com/site/content/view/47/50, 2001.

[40] D. Morgan. *A Microscopic Simulation Laboratory for Advanced Public Transportation System Evaluation*, 2002.

[41] MSDN Library. *http://msdn.microsoft.com/es-uy/library/ms123401.aspx*

[42] R. Odoni, J. Rousseau, N. Wilson. *Models in urban and air transportation*, 1994.

[43] F. Ortega. *Topología Óptima en el Diseño de Redes Metropolitanas de Transporte Público,* 2007.

[44] PARAMICS. *http://www.paramics-online.com/public-transport.php*

[45] Patrones de diseño. *http://patronesdediseno.blogspot.com/2009/05/patron-strategy.html*

[46] Proyecto de Investigación PDT 48/02, Departamento de Investigación Operativa, Facultad de Ingeniería, Universidad de la República.

[47] H. Retamales, F. Moll, G. Olguín, G.González, J. Rodríguez, J. Díaz. *Modelo de simulación del sistema de transporte público de pasajeros de Gran Medoza. Indicadores de comportamiento*, 2008.

[48] Y. Sheffi. *Urban Transportation Networks*, 1984.

[49] SIGMUR. *El formato vectorial, Análisis espacial. http://www.um.es/geograf/sigmur/sigpdf/temario_8.pdf*

[50] SimKit. *http://diana.cs.nps.navy.mil/simkit*

[51] Simple DIrectMedia Layer. *http://www.libsdl.org*

[52] H. Spiess, M. Florian. *Optimal Strategies: A New assignment model for transit network*, 1988.

[53] M. Fowler. *UML Distilled*, 2003.

[54] M. Urquhart, H. Cancela. *Simulación del Transporte Colectivo Urbano*, 1999.

[55] VISSIM. *http://www.vissim.com*

A. Estudio de Simuladores

El objetivo de este anexo es realizar un estudio de los diferentes simuladores existentes con el propósito de poder observar las distintas funcionalidades que proveen y así detectar requerimientos de interés para nuestro proyecto, además de los ya especificados.

En [38] encontramos un estudio exhaustivo de varios simuladores, de propósito general y específicos para el transporte público, donde la mayoría son del primer tipo. En este anexo, agregamos descripciones de aquellos simuladores específicos para el transporte público que no se encuentran en la publicación de la revista, pero que creemos conveniente mencionarlos debido a su similitud con este proyecto.

AIMSUN (Advance Interactive Microscopic Simulation for Urban and non-urban Networks)

Esta herramienta modela rutas urbanas como rurales así como también autopistas, señales de tránsito, prioridades en los cruces, rotondas y desvíos. Está integrado al entorno de simulación GETRAM (Generic Environment for Traffic Analysis and Modelling), el cual está compuesto por un editor gráfico (TEDI), una base de datos de la red, una base de datos para los resultados y una API (Application Programming Interface) para permitirle al usuario que desarrolle una aplicación externa la cual podrá ser ejecutada durante la simulación.

Las entradas para este simulador son un conjunto de parámetros y un escenario el cual está compuesto por: las características de la red, el comportamiento del simulador, la demanda y el comportamiento de los vehículos.

Los vehículos correspondientes al transporte público son agrupados en una misma clase para así definir carriles con prioridad o carriles exclusivos. Los vehículos son generados siguiendo la frecuencia o la tabla de horarios. Esta herramienta no tiene un modelo para generar la demanda de los pasajeros. Por lo que el tiempo que los vehículos permanecen detenidos en las paradas sigue una distribución normal con media y desviación estándar dadas. Estos valores son ingresados para cada línea y para cada parada. Las paradas tienen un área de estacionamiento donde pueden aparcar los buses para subir y descender pasajeros. Si el área está llena, éstos deben esperar en la calle bloqueando el tránsito. Las paradas de buses son asignadas a ciertas líneas.

CORSIM (CORridor SIMulation)

Está hecho a partir de dos microsimuladores: NETSIM para áreas urbanas y FRESIM para autopistas. Corre en el entorno TSIS (Traffic software Integrated System) el cual incluye un procesador de entrada de datos TRAFED para construir la red y un programa de animación TRAFVU. También puede interactuar con una aplicación externa en tiempo de ejecución.

Como dato de entrada toma la red, la demanda y las señales de tránsito. También puede modelar carriles exclusivos para buses.

Los buses son generados según la frecuencia por línea definida por el usuario. Esta herramienta no tiene un modelo para generar la demanda de los pasajeros por lo que el tiempo que el bus se encuentra en las paradas debe ser definido antes de correr la simulación.

Este valor fluctúa siguiendo una distribución de probabilidades dada. Las paradas de buses son asignadas a ciertas líneas.

DRACULA (Dynamic Route Assignment Combining User Learning and MicrosimulAtion)

Es un microsimulador de tráfico urbano desarrollado por "Institute for Transport Studies, University of Leeds". Los datos de entrada para este simulador son: la demanda, la red, parámetros correspondientes a las señales de tránsito y parámetros correspondientes a la simulación. Esta herramienta permite modelar señales de tráfico, rotondas y prioridad en ciertos cruces.

La herramienta maneja 6 tipos de vehículos incluyendo los buses. Su comportamiento puede ser representado por los siguientes modelos: car-following, lane-changing y gap-acceptance. DRACULA también modela los carriles exclusivos para un cierto tipo de vehículo.

Los vehículos son generados según la frecuencia establecida. Éstos tienen capacidad ilimitada y los pasajeros abordan al primero que llegue a la parada, sin importar el destino. Las paradas de buses son asignadas a ciertas líneas. La demanda de pasajeros en cada parada es generada en base a un valor promedio el cual es por hora. El tiempo que demora el bus en la parada es calculado por la herramienta según ciertos parámetros que son asignados por el usuario de la aplicación. La herramienta asume que el tiempo que demoran los pasajeros en bajar del bus es nulo.

PARAMICS (PARAllel MICroscopic Simulation)

Esta herramienta modela el tráfico en zonas urbanas y rurales. Está compuesto por cuatro componentes:

- PARAMICS Modeller: provee una imagen de la red y de la demanda por medio de una GUI.
- PARAMICS Processor: corre la simulación y genera los datos de salida, usando los modelos para el comportamiento de los vehículos car-following, lane-changing o gap-acceptance.
- PARAMICS Analyser: lee y procesa los resultados.
- PARAMICS Programmer: es un API para poder hacer aplicaciones externas.

Los vehículos son generados según la frecuencia definida por el usuario para cada línea. Las paradas de buses son asignadas a ciertas líneas. El usuario puede definir el tamaño de la zona donde los buses pueden aparcar para subir y/o bajar pasajeros. El tiempo que demora el bus en la parada depende de los pasajeros que se bajen y de los pasajeros que se suban al bus.

VISSIM (Verkehr In Stadten - Simulation)

Este microsimulador modela tanto las rutas de zonas urbanas y rurales así como también autopistas. Puede interactuar con modelos mesoscópicos para obtener el modelo de asignación de tráfico.

VISSIM está compuesto por dos programas: un simulador de tráfico y un modelo de señales de tránsito. El primero, modela el comportamiento de los buses siguiendo los modelos: car-following o lane-changing. El modelo de señales de tránsito, almacena información de los

detectores de tráfico en la red y en base a ello decide el estado de las señales de tránsito en el próximo paso de la simulación.

Los datos de entrada son: la demanda (representada como matriz origen-destino), la red, los parámetros relacionados con las señales de tránsito y los parámetros relacionados con la simulación.

Los vehículos se generan mediante la frecuencia que definió el usuario. El tiempo de espera en las paradas, puede ser calculado por una distribución normal, por una distribución definida por el usuario o por un cálculo explícito. Si se elije esta última opción, la demanda de pasajeros por línea en cada parada (por hora), y el porcentaje de pasajeros que descienden del bus deben ser especificados. También se deben especificar, el tiempo promedio para subir y bajar del bus (este dato debe ser especificado para cada bus), la cantidad de puertas, y el tiempo extra en la parada. Este tiempo puede estar compuesto por el tiempo que le toma frenar en la parada, abrir y cerrar puertas, etc. También se puede modelar que los pasajeros suban y bajen por distintas puertas del bus paralelamente.

B. Estudio de bibliotecas de Simulación

Dada la necesidad de contar con un simulador que permita satisfacer ciertos requerimientos, fue necesario escoger una biblioteca de simulación. Esta tarea requirió tener que realizar un estudio de un conjunto de bibliotecas de simulación existentes y tomar la decisión de escoger una de ellas para el desarrollo del simulador. Esta decisión no fue al azar sino que tuvo un cierto criterio para su elección. En primer lugar, se decidió acotar el conjunto de bibliotecas a estudiar. Para eso, se tomaron solamente aquellas cuyo lenguaje de programación sea familiarizado por nosotros (particularmente C, C++ o Java). Luego, teniendo definidas estas bibliotecas se procedió a estudiarlas de forma particular para poder tomar la decisión de cual utilizar.

En el estudio de las bibliotecas de simulación nos focalizamos en:

- SimKit
- DSOL
- DESMO-J
- JavaSim
- EOSimulator

SimKit
La biblioteca SimKit está basada en java. Su modelado se realiza a partir de un modelo de grafo a eventos (Event Graph Model). Dado que no se tiene conocimiento de este tipo de modelado y el objetivo del proyecto no es estudiarlo, decidimos descartarlo como una posible biblioteca a estudiar.

DSOL
Es una biblioteca open source basada en Java. Soporta simulación a eventos discretos (distribuida e interacción a procesos) y también simulación continua. La misma se destaca por la posibilidad de crear animaciones en 2-D y 3-D, mostrar la salida en la Web y su interoperabilidad con sistemas de información externos. También posee muestreo de distribuciones continuas y discretas y varios generadores de números aleatorios. Los experimentos que se pueden realizar deben estar definidos mediante un archivo XML.

DESMO-J
Se trata de una biblioteca implementada en Java que soporta los enfoques de orientación a eventos e interacción de procesos. Particularmente maneja el enfoque de dos fases pero no el enfoque de tres fases. La misma no permite animaciones durante la simulación (las salidas son sólo reportes, histogramas, etc.). Cabe mencionar que los reportes son automáticamente generados en HTML o en XML. Por otro lado, posee muestreo de distribuciones basado en generadores de números aleatorios, cuenta con componentes de colas y también brinda clases abstractas como modelo, entidad, evento y procesos de simulación. Asimismo, brinda la posibilidad de manejar scheduler, listas de eventos y el reloj, todo encapsulado en una misma clase denominada “Experiment”. Posee una documentación de usuario completa que consiste en un tutorial y su API.

JavaSim
Como su nombre la describe, es una biblioteca implementada en Java, orientada a interacción de procesos. La misma no permite animaciones durante la simulación.

EOSimulator

Esta biblioteca es implementada por el Departamento de Investigación de Operaciones de la Facultad de Ingeniería. Está implementada en C++ y soporta el paradigma de orientación a eventos en dos y tres fases. EOSimulator está basado en dos bibliotecas de simulación a eventos discretos: PascalSim (1989) y DESMO-J (1999 -2005). Posee algunas características que ambas bibliotecas tienen en común así como también contiene otras características que son propias de cada una de ellas. Se dice que EOSimulator se describe como PascalSim pero a través de una perspectiva de DESMO-J. EOSimulator tiene la característica de separar lo que es el modelo, de aquellas estructuras necesarias para correr la simulación. Es decir, estos conceptos están encapsulados en dos clases diferentes:

- Model: define el sistema que estamos modelando.
- Experiment: proporciona las estructuras y operaciones necesarias para correr la simulación.

Esta biblioteca posee la capacidad de poder agregar generadores de números aleatorios así como también distribuciones. Cuenta con acumuladores de datos de salidas. Por otro lado, tiene la capacidad de generar una salida gráfica de la simulación en forma de animaciones icónicas en 2D en base a primitivas disponibles. Básicamente cuenta con primitivas de movimiento, el cual es fluido pero depende en cierta medida de los recursos de la máquina. Se puede manejar la velocidad de la simulación cambiando la relación entre el tiempo real y el tiempo de simulación, y también es posible importar imágenes para ser utilizadas como representación en la simulación. EOSimulator maneja histogramas (series de tiempo, observaciones y tiempos ponderados) los cuales se deben generar de forma manual.

C. Documentación técnica de igoR-tp

En esta sección se documentan todas las operaciones que ofrece el componente que maneja la información geográfica tanto para el módulo de construcción como para el módulo de manipulación. Esto incluye todas las funcionalidades que provee igoR-tp v1.0, así como también todas las agregadas y modificadas en la versión de igoR-tp v2.0.

Interfase del Componente Geográfico del módulo de Construcción

A continuación se nombran las funcionalidades que ofrece el componente geográfico del módulo de construcción acompañado de una breve descripción de cada funcionalidad.

Evento generado por el control para informar las acciones que ocurren mediante una serie de cadenas de caracteres.

public event MyControlEventHandler Informacion;

Retorna o asigna el valor que indica si se desea recibir la información mediante el método anterior.

public bool Info

Constructor del control.

public MWControl()

Agrega las capas de aristas, puntos demanda, centroides, conexiones y zonas al control del mapa, inicializa algunas propiedades de estas capas como tamaños de puntos y bordes, inicializa además el FSS (grafo que se utiliza para el cálculo del peso de las conexiones, caminos mínimos en capa de aristas entre dos de sus intersecciones).

public bool VerCapas()

Retorna a la vista anterior del mapa si no es la primera. Retorna la cantidad de vistas previas que hay almacenadas.

public int ZoomToPrevExtent()

Retorna a la vista siguiente del mapa si es que hay alguna almacenada. Retorna la cantidad de vistas siguientes que hay almacenadas.

public int ZoomToNextExtent()

Retorna una lista con los nombres de las capas cargadas en el mapa.

public List<string> GetListaCapasMapa()

Retorna una lista con los nombres de los campos de la capa nomCapa cargada en el mapa.

public List<string> GetNombreCamposCapa(string nomCapa)

Retorna una lista con los nombres de los campos de la capa que se encuentra en pathCapa. Para esto la capa es abierta y luego de consultar, cerrada.

public List<string> GetNombreCamposCapaNoCargada(string pathCapa)

Retorna TRUE si la capa que se encuentra en path es del tipo Punto, Punto M, o Punto Z.

public bool EsCapaPuntos(string path)

Retorna TRUE si la capa ubicada en path es del tipo Poli línea, Poli línea M o Poli línea Z.

public bool EsCapaPolilinea(string path)

Retorna TRUE si la capa que se encuentra en path es del tipo Polígono, Polígono M, Polígono Z.
public bool EsCapaPoligonos(string path)

Retorna o asigna el valor del Factor Conversor de la capa de aristas.
public double FactorConversor

Retorna o asigna el valor de la Velocidad Media.
public double VelocidadMedia

Retorna el valor del Factor Conversor de la capa que se encuentra en path.
public double GetFactorConversorDeCapa(string path)

Carga la capa que se encuentra en s como capa de ptos demanda la cual tiene como campo ID a nomCampoId.
public bool CargarCapaPtoDemanda(string s, string nomCampoId)

Asigna el color a la capa de ptos demanda.
public void SetearColorPtoDemandas(uint color)

Asigna el tamaño size a los puntos de la capa ptos demanda.
public void SetearSizePtoDemandas(float size)

Retorna el color de la capa ptos demanda.
public uint GetColorPtoDemandas()

Retorna el tamaño de los puntos de la capa ptos demanda.
public float GetSizePtoDemandas()

Asigna el estilo de puntos ep a la capa ptos demanda.
public void SetearEstiloPtoDemanda(ESTILO_PUNTO ep)

Retorna el estilo de puntos de la capa ptos demanda.
public int GetEstiloPtoDemandas()

Coloca la capa ptos demanda como visible o no visible dependiendo del valor visible.
public void SetearVisibilidadPtoDemandas(bool visible)

Retorna TRUE si la capa ptos demanda es visible FALSE en caso contrario.
public bool EsVisiblePtoDemandas()

Carga la capa que se encuentra en s como a capa de aristas. Si recalc es TRUE recalcula los campos toNodo y fromNodo de la capa.
public bool CargarCapaAristas(string s, bool recalc)

Carga la capa que se encuentra en s como a capa de aristas. Si recalc es TRUE recalcula los campos toNodo y fromNodo de la capa.

public bool CargarCapaAristas(string s, bool recalc, string campoFrom, string campoTo)

Asigna el color a la capa de aristas.
public void SetearColorAristas(uint color)

Retorna el color de la capa de aristas.
public uint GetColorAristas()

Asigna el grosor a la capa de aristas.
public void SetearGrosorAristas(float grosor)

Retorna el grosor de la capa de aristas.
public float GetGrosorAristas()

Asigna el estilo de linea el a la capa de aristas
public void SetearEstiloAristas(ESTILO_LINEA el)

Retorna el estilo de linea de la capa de aristas
public int GetEstiloAristas()

Coloca la capa de aristas como visible o no visible dependiendo del valor visible.
public void SetearVisibilidadAristas(bool visible)

Retorna TRUE si la capa de aristas es visible, FALSE en caso contrario.
public bool EsVisibleAristas()

Crea la capa de zonas en la ruta path.
public void CrearCapaZonas(string path)

Carga la capa en path como capa de zonas.
public void CargarCapaZonas(string path)

Asigna el color como color de relleno a la capa de zonas.
public void SetearColorRellenoZonas(uint color)

Retorna el color de relleno de la capa de zonas.
public uint GetColorRellenoZonas()

Asigna el valor transparencia a la capa de zonas.
public void SetearTransparenciaZonas(float transparencia)

Retorna el valor de transparencia de la capa de zonas.
public float GetTransparenciaZonas()

Asigna el estilo de polígonos ep a la capa de zonas.
public void SetearEstiloRellenoZonas(ESTILO_POLY ep)

Retorna el estilo de polígonos de la capa de zonas.
public int GetEstiloRellenoZonas()

Coloca la capa de zonas como visible o no visible dependiendo de visible.
public void SetearVisibilidadZonas(bool visible)

Crea la capa de conexiones en la ruta path.
public void CrearCapaConexiones(string path)

Carga la capa en path como capa de conexiones.
public void CargarCapaConexiones(string path)

Asigna el color a la capa de conexiones.
public void SetearColorConexiones(uint color)

Retorna el color de la capa de conexiones.
public uint GetColorConexiones()

Asigna el grosor de línea grosor a la capa de conexiones.
public void SetearGrosorConexiones(float grosor)

Retorna el grosor de línea de la capa de conexiones.
public float GetGrosorConexiones()

Asigna el estilo de línea a la capa de conexiones.
public void SetearEstiloConexiones(ESTILO_LINEA el)

Retorna el estilo de línea de la capa de conexiones.
public int GetEstiloConexiones()

Coloca la capa de conexiones como visible o no visible dependiendo de visible.
public void SetearVisibilidadConexiones(bool visible)

Retorna TRUE si la capa de conexiones es visible FALSE en caso contrario.
public bool EsVisibleConexiones()

Crea la capa de centroides en path.
public void CrearCapaCentroides(string path)

Carga la capa en path como capa de centroides.
public void CargarCapaCentroides(string path)

Asigna el color a la capa de centroides.
public void SetearColorCentroides(uint color)

Asigna el tamaño size a los puntos de la capa de centroides.
public void SetearSizeCentroides(float size)

Retorna el color de la capa de centroides.
public uint GetColorCentroides()

Retorna el tamaño de los puntos de la capa de centroides.
public float GetSizeCentroides()

Asigna el estilo de punto ep a la capa de centroides.
public void SetearEstiloCentroides(ESTILO_PUNTO ep)

Retorna el estilo de punto de la capa de centroides.
public int GetEstiloCentroides()

Coloca la capa de centroides como visible o no visible dependiendo de visible.
public void SetearVisibilidadCentroides(bool visible)

Retorna TRUE si la capa de centroides es visible FALSE en otro caso.
public bool EsVisibleCentroides()

Crea la capa de nodos viales en path.
public void CrearCapaNodoVial(string path)

Carga la capa que se encuentra en la dirección path como capa de nodos viales.
public void CargarCapaNodosViales(string path)

Retorna TRUE si la capa no contiene nodos viales, o FALSE en caso de que contenga al menos un nodo vial.
public bool esVaciaCapaNodoVial()

Asigna el color a la capa de nodos viales.
public void SetearColorNodoVial (uint color)

Asigna el tamaño size a los puntos de la capa de nodos viales.
public void SetearSizeNodoVial (float size)

Retorna el color de la capa de nodos viales.
public uint GetColorNodoVial ()

Retorna el tamaño de los puntos de la capa de nodos viales.
public float GetSizeNodoVial ()

Asigna el estilo de punto ep a la capa de nodos viales.
public void SetearEstiloNodoVial (ESTILO_PUNTO ep)

Retorna el estilo de punto de la capa de nodos viales.
public int GetEstiloNodoVial ()

Coloca la capa de nodos viales como visible o no visible dependiendo de visible.
public void SetearVisibilidadNodoVial (bool visible)

Retorna TRUE si la capa de nodos viales es visible FALSE en otro caso.
public bool EsVisibleNodoVial ()

Crea la capa de paradas en path.
public void CrearCapaParada(string path)

Carga la capa ubicada en path como capa de paradas.
public void CargarCapaParada (string path)

Asigna el color a la capa de paradas.
public void SetearColorParada (uint color)

Asigna el tamaño size a los puntos de la capa de paradas.
public void SetearSizeParada (float size)

Retorna el color de la capa de paradas.
public uint GetColorParada ()

Retorna el tamaño de los puntos de la capa de paradas.
public float GetSizeParada ()

Asigna el estilo de punto ep a la capa de paradas.
public void SetearEstiloParada (ESTILO_PUNTO ep)

Retorna el estilo de punto de la capa de paradas.
public int GetEstiloParada ()

Coloca la capa de paradas como visible o no visible dependiendo de visible.
public void SetearVisibilidadParada (bool visible)

Retorna TRUE si la capa de paradas es visible FALSE en otro caso.

public bool EsVisibleParada ()

Crea la capa de caminatas en la ruta path.
public void CrearCapaCaminatas (string path)

Carga la capa ubicada en path como capa de caminatas.
public void CargarCapaCaminatas (string path)

Asigna el color a la capa de caminatas.
public void SetearColorCaminatas (uint color)

Retorna el color de la capa de caminatas.
public uint GetColorCaminatas ()

Asigna el grosor de línea grosor a la capa de caminatas.
public void SetearGrosorCaminatas(float grosor)

Retorna el grosor de línea de la capa de caminatas.
public float GetGrosorCaminatas ()

Asigna el estilo de línea a la capa de caminatas.
public void SetearEstiloCaminatas (ESTILO_LINEA el)

Retorna el estilo de línea de la capa de caminatas.
public int GetEstiloCaminatas ()

Coloca la capa de caminatas como visible o no visible dependiendo de visible.
public void SetearVisibilidadCaminatas (bool visible)

Retorna TRUE si la capa de caminatas es visible FALSE en caso contrario.
public bool EsVisibleCaminatas ()

Setea el control en estado NONE.
public void None()

Setea el control en estado ZOOM_IN.
public void ZoomIn()

Setea el control en estado ZOOM_OUT.
public void ZoomOut()

Setea el control en estado PAN.
public void Pan()

Setea el control en estado NUEVA_ZONA.
public void NuevaZona()

Setea el control en estado NUEVA_CONEXION.
public void NuevaConexion()

Setea el control en estado NUEVA_CAMINATA_VIAL
public void NuevaCaminata()

Setea el control en estado ELIMINAR_ZONA.
public void EliminarZona()

Setea el control en estado ELIMINAR_CONEXION.
public void EliminarConexion()

Setea el control en estado ELIMINAR_CAMINATA_VIAL
public void EliminarCaminatas()

Setea el control en estado SETEAR_PARADA desde el cual se puede marcar o desmarcar una parada.
public void SetearParadas()

Retorna la matriz demanda dada la información de demanda dataDemanda
public double[,] CrearMatrizDemanda(Dictionary<int, Dictionary<int, double>> dataDemanda)

Construye la estructura FSS a partir de la capa de aristas, esta estructura es la que se utiliza para calcular el peso de las conexiones y para evaluar la conectividad de la red de aristas.
public void ConstruirFSS()

Remueve todas las capas del mapa e inicializa todas las variables.
public void ClearAll()

Deshace la última acción (que se puede deshacer) y retorna la cantidad de acciones para deshacer.
public int Undo()

Rehace la última acción deshecha y retorna la cantidad de acciones para rehacer.
public int Redo()

Limpia todas las acciones a rehacer y deshacer.
public void LimpiarHistorial()

Retorna los formatos soportados para las capas de fondo.
public string FormatoFondo()

Carga la capa en path como fondo.
public void CargarFondo(string path)

Remueve todas las capas de fondo del mapa.
public void RemoverFondo()

Asigna el color a la capa en el mapa de id nombre.
public void SetearColorCapa(string nombre, uint color)

Retorna el color de la capa del mapa de id nombre.
public uint GetColorCapa(string nombre)

Asigna el tamaño size a los puntos de la capa de puntos del mapa de id nombre.
public void SetearSizePtoCapa(string nombre, float size)

Asigna el tamaño grosor a las líneas de la capa de líneas del mapa de id nombre.
public void SetearGrosorLineaCapa(string nombre, float grosor)

Asigna el valor de transparencia a los polígonos de la capa de polígonos del mapa de id nombre.

public void SetearTransparenciaPoliCapa(string nombre, float transparencia)

Retorna el tamaño de los puntos de la capa de puntos del mapa de id nombre.
public float GetSizePtoCapa(string nombre)

Retorna el grosor de las líneas de la capa de líneas del mapa de id nombre.
public float GetGrosorLineaCapa(string nombre)

Retorna la transparencia de los polígonos de la capa de polígonos del mapa de id nombre.
public float GetTransparenciaPoliCapa(string nombre)

Asigna el estilo de punto ep a los puntos de la capa de puntos del mapa de id nombre.
public void SetearEstiloPuntoCapa(string nombre, ESTILO_PUNTO ep)

Asigna el estilo de línea a las líneas de la capa de líneas del mapa de id nombre.
public void SetearEstiloLineaCapa(string nombre, ESTILO_LINEA el)

Asigna el estilo de polígonos ep a los polígonos de la capa de polígonos de id nombre.
public void SetearEstiloPoligonoCapa(string nombre, ESTILO_POLY ep)

Retorna el estilo de los puntos de la capa de puntos del mapa de id nombre.
public int GetEstiloPuntoCapa(string nombre)

Retorna el estilo de las líneas de la capa de líneas del mapa de id nombre.
public int GetEstiloLineaCapa(string nombre)

Coloca como visible o no visible a la capa del mapa de id nombre dependiendo de visible.
public void SetearVisibilidadCapa(string nombre, bool visible)

Retorna TRUE si la capa del mapa de id nombre es visible FALSE en otro caso.
public bool EsVisibleCapa(string nombre)

Remueve la capa de fondo del mapa de id nomCapa.
public void RemoverFondo(string nomCapa)

Dibuja o borra las componentes conexas de la capa de aristas dependiendo del valor b.
public void SetVisibleCompConexas(bool b)

Dibuja o borra los caminos construidos para calcular el peso de las conexiones dependiendo del valor b.
public void SetVisibleSPT(bool b)

Dibuja o borra las marcas sobre los puntos de demanda no cubierta por zonas dependiendo del valor de b. Estas marcas aparecen cada vez que se intenta calcular la matriz de demanda.
public void SetVisibleDemCubierta(bool b)

Retorna TRUE si están dibujadas las componentes conexas, FALSE en otro caso.
public bool GetVisibleCompConexas()

Retorna TRUE si están dibujados los caminos relacionados con las conexiones.
public bool GetVisibleSPT()

Retorna TRUE si están dibujadas las marcas sobre los puntos de demanda no cubierta por zonas, FALSE en otro caso.
public bool GetVisibleDemCubierta()

Habilita o deshabilita el tooltip dependiendo de b.
public void SetToolTip(bool b)

Retorna TRUE si el tooltip esta habilitado, FALSE en otro caso.
public bool GetToolTip()

Registra el campo nomCampo de la capa del mapa de id nomCapa para que aparezca en el tooltip.
public void RegistrarCampoParaTT(string nomCapa, string nomCampo)

Desregistra el campo nomCampo de la capa del mapa de id nomCapa.
public void DesRegistrarCampoParaTT(string nomCapa, string nomCampo)

Retorna todos los campos de cada capa del mapa que pueden aparecer en el tooltip en un diccionario de clave id de la capa y valor lista de nombres de sus campos.
public Dictionary<string, List<string>> GetCamposCapasTT()
public bool CampoCapaTT(string nomCapa, string nomCampo)

Retorna una tabla con los valores de los campos de la capa del mapa de id nombre.
public DataTable ObtenerPropiedadesCapa(string nombre)

Crea una selección para la capa del mapa de id nombre si no tenia.
Marca en el mapa los shapes de la capa de id nombre indicados en índices.
public void MarcarSeleccion(string nombre, List<int> indices)

Borra las marcas en el mapa de la capa de id nombre.
public void BorrarSeleccion(string nombre)

Elimina la selección de la capa del mapa de id nombre.
public void EliminarSeleccion(string nombre)

Realiza zoom a la selección de la capa del mapa de id nombre.
public void zoomToSelectedShapes(string nombre)

Crea una nueva selección en el mapa para marcar centroides.
public void NuevaSeleccionOD()

Marca los centroides identificados en la lista ods.
public void MarcarOD(List<int> ods)

Borra todas las marcas de la selección de centroides.
public void BorrarSeleccionOD()

Elimina la selección de centroides.
public void EliminarSeleccionOD()

Interfase del Componente Geográfico del módulo de Manipulación

A continuación se nombran las funcionalidades que ofrece el componente geográfico del módulo de Manipulación acompañado de una breve descripción de cada funcionalidad.

Evento generado por el control para informar las acciones que ocurren mediante una serie de cadenas de caracteres.
public event MyControlEventHandler Informacion;

Retorna o asigna el valor que indica si se desea recibir la informacion mediante el método anterior.
public bool Info

Constructor del control.
public MWControl()

Retorna el nombre del recorrido activo.
public string Rcurrent

Retorna o asigna la cantidad de pasajeros de los buses.
public int CapacidadBuses

Setea el control en estado NONE.
public void None()

Setea el control en estado ZOOM_IN.
public void ZoomIn()

Setea el control en estado ZOOM_OUT.
public void ZoomOut()

Setea el control en estado PAN.
public void Pan()

Visualiza el mapa en su máxima extensión.
public void FullExtent()

Crea un recorrido en path con frecuencia w.
public void NuevoRecorrido(string path, float w)

Setea el control en estado NUEVO_TRAMO, en donde el usuario puede agregar tramos al recorrido activo.
public void NuevoTramo()

Setea el control en estado ELIMINAR_TRAMO, en donde el usuario puede eliminar tramos al recorrido activo.
public void EliminarTramo()

Finaliza la edición del recorrido activo.
public void FinEdicion()

Agrega las capas de aristas, puntos demanda, centroides, conexiones y zonas al control del mapa, inicializa algunas propiedades de estas capas como tamaños de puntos y bordes, inicializa además el FSS (grafo que se utiliza para el cálculo del peso de las conexiones, caminos mínimos en capa de aristas entre dos de sus intersecciones).
public bool VerCapas()

Retorna a la vista anterior del mapa si no es la primera. Retorna la cantidad de vistas previas que hay almacenadas.
public int ZoomToPrevExtent()

Retorna a la vista siguiente del mapa si es que hay alguna almacenada. Retorna la cantidad de vistas siguientes que hay almacenadas.
public int ZoomToNextExtent()

Retorna o asigna el valor de frecuencia mínima.
public double FrecMin

Retorna o asigna el valor de frecuencia máxima.
public double FrecMax

Retorna o asigna el valor de unidades por frecuencia para dibujar los recorridos en modo ponderado por frecuencia.
public int UnitFrec

Retorna una lista con los nombres de las capas cargadas en el mapa.
public List<string> GetListaCapasMapa()

Retorna una lista con los nombres de los campos de la capa nomCapa cargada en el mapa.
public List<string> GetNombreCamposCapa(string nomCapa)

Retorna una lista con los nombres de los campos de la capa que se encuentra en pathCapa. Para esto la capa es abierta y luego de consultar, cerrada.
public List<string> GetNombreCamposCapaNoCargada(string pathCapa)

Retorna TRUE si la capa que se encuentra en path es del tipo Punto, Punto M, o Punto Z.
public bool EsCapaPuntos(string path)

Retorna TRUE si la capa que se encuentra en path es del tipo Poli línea, Poli línea M, o Poli línea Z.
public bool EsCapaPolilinea(string path)

Retorna TRUE si la capa que se encuentra en path es del tipo Polígono, Polígono M, Polígono Z.
public bool EsCapaPoligonos(string path)

Carga la capa en path como capa de conexiones.
public void CargarCapaConexiones(string path)

Asigna el color a la capa de conexiones.
public void SetearColorConexiones(uint color)

Retorna el color de la capa de conexiones.
public uint GetColorConexiones()

Asigna el grosor de línea grosor a la capa de conexiones.
public void SetearGrosorConexiones(float grosor)

Retorna el grosor de línea de la capa de conexiones.
public float GetGrosorConexiones()

Asigna el estilo de línea a la capa de conexiones.
public void SetearEstiloConexiones(ESTILO_LINEA el)

Retorna el estilo de línea de la capa de conexiones.
public int GetEstiloConexiones()

Coloca la capa de conexiones como visible o no visible dependiendo de visible.
public void SetearVisibilidadConexiones(bool visible)

Retorna TRUE si la capa de conexiones es visible FALSE en caso contrario.
public bool EsVisibleConexiones()

Carga la capa en path como capa de centroides.
public void CargarCapaCentroides(string path)

Asigna el color a la capa de centroides.
public void SetearColorCentroides(uint color)

Asigna el tamaño size a los puntos de la capa de centroides.
public void SetearSizeCentroides(float size)

Retorna el color de la capa de centroides.
public uint GetColorCentroides()

Retorna el tamaño de los puntos de la capa de centroides.
public float GetSizeCentroides()

Asigna el estilo de punto ep a la capa de centroides.
public void SetearEstiloCentroides(ESTILO_PUNTO ep)

Retorna el estilo de punto de la capa de centroides.
public int GetEstiloCentroides()

Coloca la capa de centroides como visible o no visible dependiendo de visible.
public void SetearVisibilidadCentroides(bool visible)

Retorna TRUE si la capa de centroides es visible FALSE en otro caso.
public bool EsVisibleCentroides()

Carga la capa en path como capa de caminatas.
public void CargarCapaCaminatas (string path)

Asigna el color a la capa de caminatas.
public void SetearColorCaminatas (uint color)

Retorna el color de la capa de caminatas.
public uint GetColorCaminatas ()

Asigna el grosor de línea grosor a la capa de caminatas.
public void SetearGrosorCaminatas (float grosor)

Retorna el grosor de línea de la capa de caminatas.
public float GetGrosorCaminatas ()

Asigna el estilo de línea a la capa de caminatas.
public void SetearEstiloCaminatas (ESTILO_LINEA el)

Retorna el estilo de línea de la capa de caminatas.
public int GetEstiloCaminatas ()

Coloca la capa de caminatas como visible o no visible dependiendo de visible.
public void SetearVisibilidadCaminatas (bool visible)

Retorna TRUE si la capa de caminatas es visible FALSE en caso contrario.
public bool EsVisibleCaminatas ()

Carga la capa en path como capa de paradas.
public void CargarCapaParadas (string path)

Asigna el color a la capa de paradas.
public void SetearColorParadas (uint color)

Asigna el tamaño size a los puntos de la capa de paradas.
public void SetearSizeParadas (float size)

Retorna el color de la capa de paradas.
public uint GetColorParadas ()

Retorna el tamaño de los puntos de la capa de paradas.
public float GetSizeParadas ()

Asigna el estilo de punto ep a la capa de paradas.
public void SetearEstiloParadas (ESTILO_PUNTO ep)

Retorna el estilo de punto de la capa de paradas.
public int GetEstiloParadas ()

Coloca la capa de paradas como visible o no visible dependiendo de visible.
public void SetearVisibilidadParadas (bool visible)

Retorna TRUE si la capa de paradas es visible FALSE en otro caso.
public bool EsVisibleParadas()

Carga la capa en path como capa de nodos viales.
public void CargarCapaNodosViales (string path)

Asigna el color a la capa de nodos viales.
public void SetearColorNodosViales (uint color)

Asigna el tamaño size a los puntos de la capa de nodos viales.
public void SetearSizeNodosViales (float size)

Retorna el color de la capa de nodos viales.
public uint GetColorNodosViales ()

Retorna el tamaño de los puntos de la capa de nodos viales.
public float GetSizeNodosViales ()

Asigna el estilo de punto ep a la capa de nodos viales.
public void SetearEstiloNodosViales (ESTILO_PUNTO ep)

Retorna el estilo de punto de la capa de nodos viales.
public int GetEstiloNodosViales ()

Coloca la capa de nodos viales como visible o no visible dependiendo de visible.

public void SetearVisibilidadNodosViales (bool visible)

Retorna TRUE si la capa de nodos viales es visible FALSE en otro caso.
public bool EsVisibleNodosViales()

Crea una capa de recorridos en path con frecuencia frec.
public bool CrearCapaRecorrido(string path, double frec)

public void CargarCapaRecorrido(string path, uint c, double frec, bool visible,
bool dirigido, bool circular)

Elimina el recorrido identificado
public void EliminarCapaRecorrido(string n)

Retorna la información de los recorridos en un diccionario que tiene como clave el nombre del recorrido.
public Dictionary<string, Recorrido> GetRecorridos()

Retorna la información de los recorridos en una lista.
public List<Recorrido> GetListaRecorridos()

Asigna como recorrido activo al recorrido con id n.
public void SetearRecorridoActivo(string n)

Asigna el color al recorrido con id nombre.
public void SetearColorRecorrido(uint color, string nombre)

Asigna la frecuencia frec al recorrido con id nombre.
public void SetearFrecuenciaRecorrido(double frec, string nombre)

Retorna el valor de la frecuencia del recorrido con id nombre.
public double GetFrecuenciaRecorrido(string nombre)

Retorna el color del recorrido con id nombre.
public uint GetColorRecorrido(string nombre)

Retorna TRUE si el recorrido con id nombre es visible en el mapa.
public bool EsVisibleRecorrido(string nombre)

Crea un recorrido en path a partir de la lista de identificadores de centroides caminoCent y la frecuencia frec.
public void CrearRecorrido(string path, List<int> caminoCent, double frec)

Retorna la cantidad de recorridos de la solución abierta.
public int GetCantRecorridos()

Retorna la suma de los costos de las conexiones asociadas a los tramos del recorrido con id nomRec cargado en el mapa.
public double TiempoRecorrido(string nomRec)

Retorna en la estructura grafo el grafo que tiene por vértices a los centroides y por aristas a las conexiones.
public Grafo ConstruirGrafo()

Retorna los formatos soportados para las capas de fondo.

public string FormatoFondo()
Carga la capa en path como fondo.
public void CargarFondo(string path)

Remueve todas las capas de fondo del mapa.
public void RemoverFondo()

Asigna el color a la capa en el mapa de id nombre.
public void SetearColorCapa(string nombre, uint color)

Retorna el color de la capa del mapa de id nombre.
public uint GetColorCapa(string nombre)

Retorna el tamaño de los puntos de la capa de puntos del mapa de id nombre.
public float GetSizePtoCapa(string nombre)

Asigna el tamaño size a los puntos de la capa de puntos del mapa de id nombre.
public void SetearSizePtoCapa(string nombre, float size)

Asigna el tamaño grosor a las líneas de la capa de líneas del mapa de id nombre.
Retorna el estilo de los puntos de la capa de puntos del mapa de id nombre.
public ESTILO_PUNTO GetEstiloPuntoCapa(string nombre)

Asigna el estilo de punto ep a los puntos de la capa de puntos del mapa de id nombre.
public void SetearEstiloPuntoCapa(string nombre, ESTILO_PUNTO ep)
public void SetearGrosorLineaCapa(string nombre, float grosor)

Retorna el grosor de las líneas de la capa de líneas del mapa de id nombre.
public float GetGrosorLineaCapa(string nombre)

Asigna el estilo de línea a las líneas de la capa de líneas del mapa de id nombre.
public void SetearEstiloLineaCapa(string nombre, ESTILO_LINEA el)

Retorna el estilo de las líneas de la capa de líneas del mapa de id nombre.
public ESTILO_LINEA GetEstiloLineaCapa(string nombre)
public float GetTransparenciaPoliCapa(string nombre)

Asigna el valor de transparencia a los polígonos de la capa de polígonos del mapa de id nombre.
public void SetearTransparenciaPoliCapa(string nombre, float transparencia)

Retorna el estilo de los polígonos de la capa de polígonos del mapa de id nombre.
public ESTILO_POLY GetEstiloPoliCapa(string nombre)

Asigna el estilo de polígonos ep a los polígonos de la capa de polígonos de id nombre.
public void SetearEstiloPoligonoCapa(string nombre, ESTILO_POLY ep)

Remueve la capa de fondo del mapa de id nomCapa.
Coloca como visible o no visible a la capa del mapa de id nombre dependiendo de visible.
public void SetearVisibilidadCapa(string nombre, bool visible)

Retorna TRUE si la capa del mapa de id nombre es visible FALSE en otro caso.
public bool EsVisibleCapa(string nombre)

Retorna la transparencia de los polígonos de la capa de polígonos del mapa de id nombre.
public void RemoverFondo(string nomCapa)

Retorna una lista con los nombres de las capas de fondo cargadas en el mapa.
public List<string> NombresCapasFondo()

Retorna la cantidad de capas de fondo cargadas en el mapa.
public int GetCantFondos()

Remueve todos los recorridos del mapa e inicializa todas las variables globales correspondientes a los recorridos.
public void ClearRecorridos()

Remueve todas las capas del mapa e inicializa todas las variables.
public void ClearAll()

Deshace la última acción (que se puede deshacer) y retorna la cantidad de acciones para deshacer.
public int Undo()

Rehace la última acción deshecha y retorna la cantidad de acciones para rehacer.
public int Redo()

Limpia todas las acciones a rehacer y deshacer.
public void LimpiarHistorial()

Dibuja el recorrido de ide rec en el mapa.
public void DibujarRecorrido(string rec)

Retorna o asigna el valor del ancho de la línea. Este valor indica el ancho con que son dibujados los recorridos en modo normal.
public int LineaBasica

Retorna o asigna el valor que indica la separación entre los tramos de los recorridos asociados a una misma conexión al dibujar en modo normal.
public int SepBasico

Retorna o asigna el valor del ancho de la flecha. Este valor indica el ancho con que son dibujadas las flechas que indican el sentido de los recorridos.
public int MultVal

Retorna o asigna el valor del largo de la flecha. Este valor indica el largo con que son dibujadas las flechas que indican el sentido de los recorridos.
public int MultV

Retorna o asigna el valor que indica si las flechas que indican el sentido del recorrido se dibujaran llenas o no.
public bool FlechaFill

Retorna o asigna el valor que indica si al dibujar los recorridos en modo ponderado por frecuencia.
public bool PolyFill

Retorna o asigna el modo en que se están dibujando los reocrridos.
public int ModoDibuRec

Habilita o deshabilita el tooltip dependiendo de b.
public void SetToolTip(bool b)

Retorna TRUE si el tooltip está habilitado, FALSE en otro caso.
public bool GetToolTip()

Registra el campo nomCampo de la capa del mapa de id nomCapa para que aparezca en el tooltip.
public void RegistrarCampoParaTT(string nomCapa, string nomCampo)

Desregistra el campo nomCampo de la capa del mapa de id nomCapa.
public void DesRegistrarCampoParaTT(string nomCapa, string nomCampo)

Retorna todos los campos de cada capa del mapa que pueden aparecer en el tooltip en un diccionario de clave id de la capa y valor lista de nombres de sus campos.
public Dictionary<string, List<string>> GetCamposCapasTT()
public bool CampoCapaTT(string nomCapa, string nomCampo)

Retorna una tabla con los valores de los campos de la capa del mapa de id nombre.
public DataTable ObtenerPropiedadesCapa(string nombre)

Marca en el mapa los shapes de la capa de id nombre indicados en índices.
public void MarcarSeleccion(string nombre, List<int> indices)

Borra las marcas en el mapa de la capa de id nombre.
public void BorrarSeleccion(string nombre)

Elimina la selección de la capa del mapa de id nombre.
public void EliminarSeleccion(string nombre)

Realiza zoom a la selección de la capa del mapa de id nombre.
public void zoomToSelectedShapes(string nombre)

Crea una nueva selección en el mapa para marcar centroides.
public void NuevaSeleccionOD()

Marca los centroides identificados en la lista ods.
public void MarcarOD(List<int> ods)

Borra todas las marcas de la selección de centroides.
public void BorrarSeleccionOD()

Elimina la selección de centroides.
public void EliminarSeleccionOD()

Retorna o asigna las cargas de los tramos de los recorridos del mapa.
public List<List<double>> CargasRecorridos

Muestra el histograma de cargas del recorrido con id n.
public void MostrarCargasRecorrido(string n)

Calcula, dibuja y retorna todos los recorridos que sirven para ir desde el centroide o hasta el centroide d ya sea utilizando un recorrido o un par de recorridos mediante un punto de transbordo.
public DataTable DibujarDirectosTransbordos(int o, int d)

Borra los dibujos creados en el mapa por la función anterior.
public void FinDirectosTransbordos()

Dibuja los transbordos que utilizan los id de los centroides seleccionados como nodo de transbordo.
public void DibujarNodosTransbordosSeleccionados(List<int> seleccionados)

Retorna TRUE si el recorrido de id nomRec es dirigido, FALSE en otro caso.
public bool EsDirigido(string nomRec)

Retorna TRUE si el recorrido de id nomRec es circular, FALSE en otro caso.
public bool EsCircular(string nomRec)

Asigna el camino de id de centroides camino al recorrido nomRec.
public void SetCaminoCentroides(string nomRec, List<int> camino)

Torna el recorrido con id nomRec como dirigido o no dirigido dependiendo del valor dirigido.
public void SetDirigido(string nomRec, bool dirigido)

Retorna TRUE si el recorrido activo está finalizado, FALSE en otro caso.
public bool activoFinalizado()

D. Documentación técnica del Simulador

El objetivo de esta sección es brindarle al desarrollador toda la información necesaria para poder continuar con el desarrollo del simulador. Esto incluye la preparación del entorno de desarrollo utilizado y sus respectivos parámetros de configuración, así como también los pasos a seguir a la hora de modificar, extender o agregar alguna funcionalidad.

D.1. Preparación del entorno de desarrollo

El entorno de desarrollo que utilizamos para implementar el simulador es Dev-C++. Por este motivo es necesaria la instalación del mismo. La biblioteca de simulación que utilizamos es EOSimulator y esta debe estar correctamente instalada para hacer las configuraciones necesarias en el proyecto. Los pasos a seguir son los siguientes:

1. Instalar Dev-C++. Este entorno de desarrollo se puede descargar de la página http://www.bloodshed.net/devcpp.html .

2. Descargar la biblioteca de simulación EOSimulator. La misma se puede descargar de la página:
   http://www.fing.edu.uy/inco/cursos/simulacion/eosim_html/index.html.

3. Ejecutar el archivo bajo el directorio eosim1.1.1/eosim1.1.1.dev y compilar la biblioteca haciendo clic en Ejecutar->Reconstruir Todo

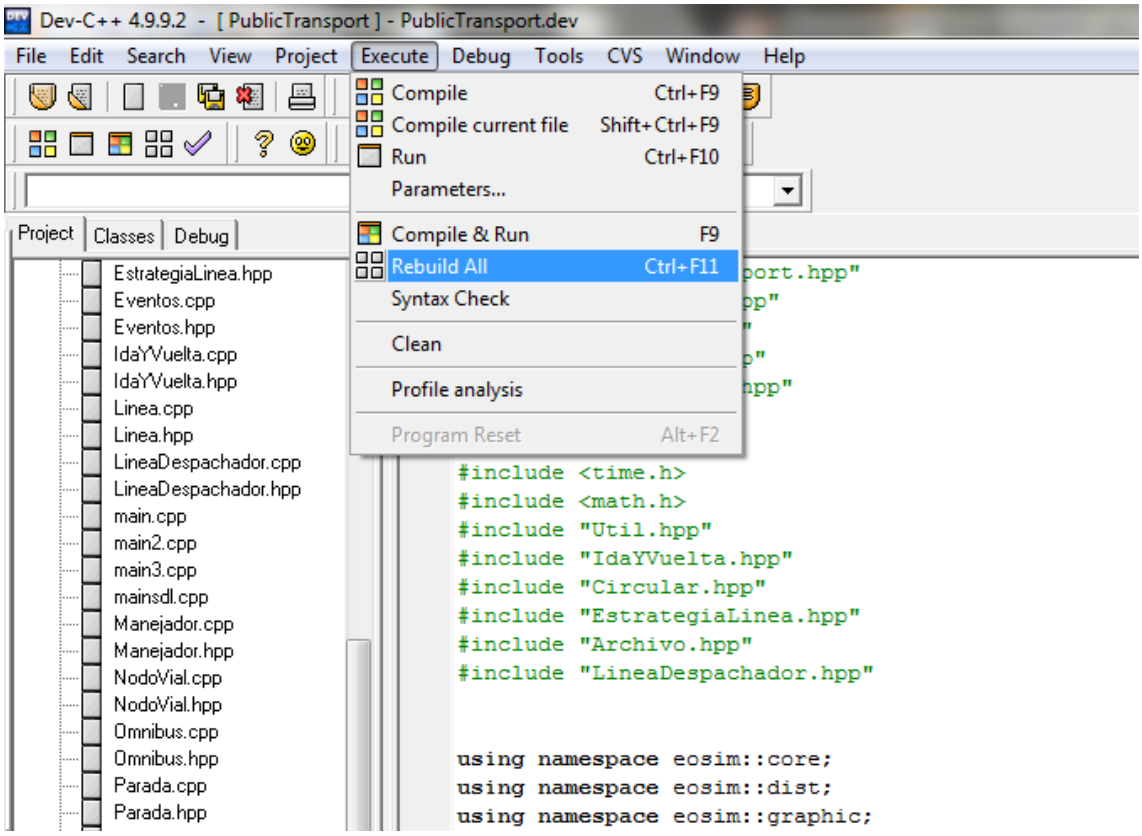


Figura D.1.1

Cumpliendo con estos pasos estamos en condiciones de abrir el proyecto asociado al simulador y configurar los parámetros para utilizar EOSimulator como librería de simualción.

D.2. Parametros de configuración del proyecto

Ya que Dev-C++ es el entorno de desarrollo utilizado, la información del proyecto se almacena en un archivo denominado PublicTransport.dev. Este archivo contiene toda la información del proyecto, los archivos que forman parte del mismo y las configuraciones necesarias para correrlo.

En primer lugar es importante chequear que los parámetros de configuración del proyecto estén correctamente configurados. Esto puede variar, y es necesario volverlo a configurar, si se cambia el directorio bajo el cual se encuentra el proyecto del simulador. Los pasos a seguir son los siguientes:

1. Abrir el proyecto del simulador haciendo clic sobre el archivo PublicTransport.dev
2. Ir a las opciones del proyecto haciendo clic sobre el menú Proyecto->Opciones de proyecto

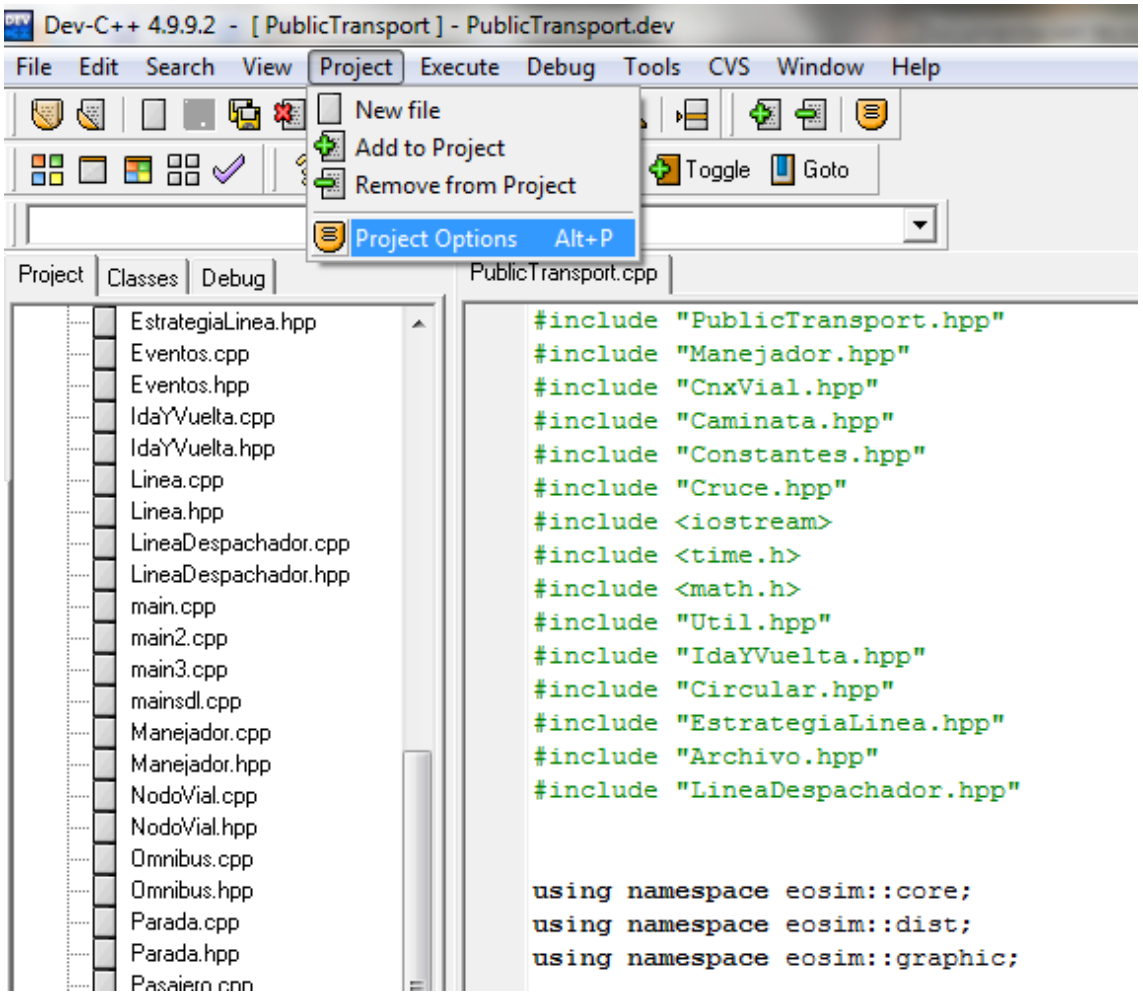


Figura D.1.2

3. Presionar la pestaña “Directorios”. Allí, es necesario configurar correctamente directorios de librerías y directorios de include. Pueden ser escritos de manera relativa o absoluta. Las siguientes imágenes ilustran las configuraciones de los directorios.

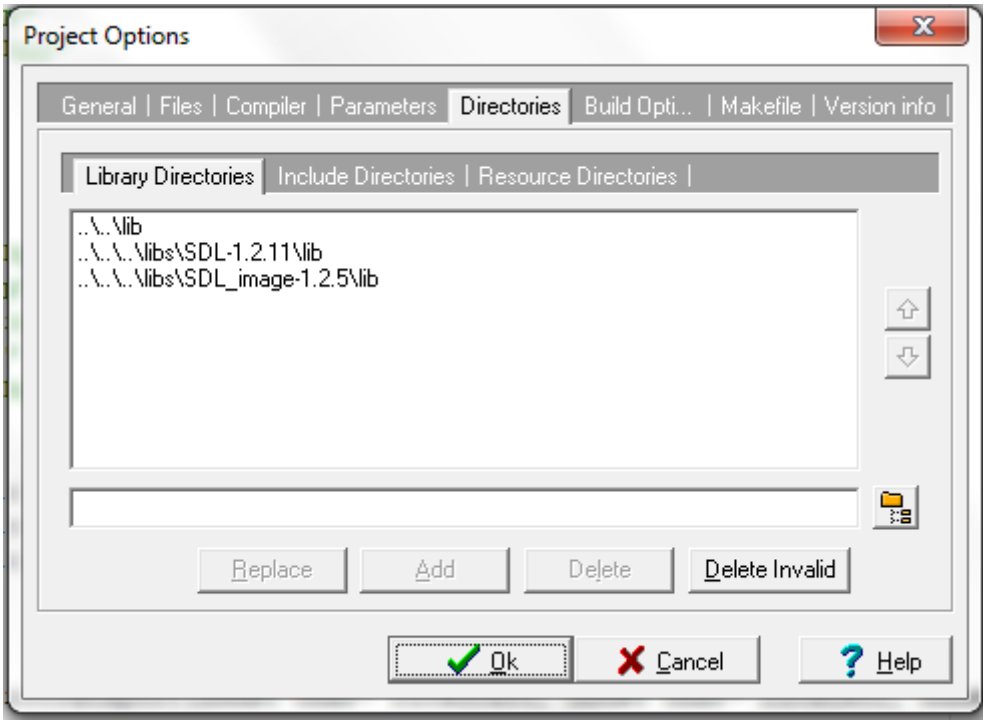


Figura D.1.3

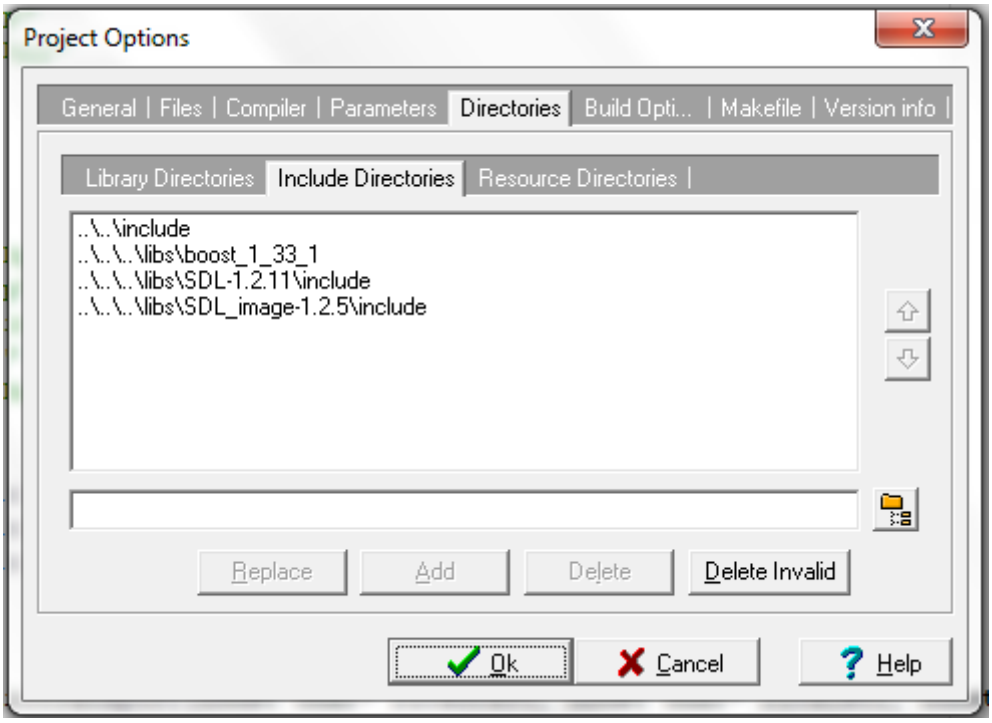


Figura D.1.4

Con esto estamos agregando includes y librerías de EOSimulator, y también manejo de interfaz grafica que éste provee. Las librerías e includes propias del simulador se encuentran bajo el directorio “EOSimulator\eosim1.1.1”.

Por otro lado, el soporte de animaciones iconográficas en 2D que tiene EOSimulator está basado en Boost C++ Libraries, SDL, SDL_image. Este es el motivo de sus inclusiones, y las mismas se pueden encontrar bajo el directorio “EOSimulator\libs” Figura D.1.2.

4. Presionar la pestaña “Parametros”. Allí es necesario agregar parámetros para el compilador y para el linker. En el sector de “Compilador” debe estar la siguiente línea:

```
-Wall -Wextra -ansi -pedantic -O1 -Dmain=SDL_main
```

Mientras que en el sector “Linker” deben estar las siguientes líneas:

..\lib\eosim1_1_1.a
..\..\..\libs\agg23\lib\libagg23.a
..\..\..\libs\freetype-2.1.10\objs\freetype2110.a
..\..\..\libs\SDL-1.2.11\lib\libSDL.dll.a
..\..\..\libs\SDL-1.2.11\lib\libSDLmain.a
..\..\..\libs\SDL_image-1.2.5\lib\libSDL_image.a
-lmingw32 –lSDLmain –lSDL –lSDL_image –liberty

Las primeras 6 líneas son links a librerías gráficas necesarias para EOSimulator. Se observa que están escritas de forma relativa al proyecto, pero pueden escribirse de forma absoluta. La primera línea se encuentra bajo el directorio “EOSimulator\eosim1.1.1” mientras que las 5 restantes se pueden encontrar bajo el directorio “EOSimulator\libs”. La última línea corresponde a parámetros necesarios para utilizar funcionalidades de SDL de forma nativa. La Figura D.1.5 ilustra este punto.

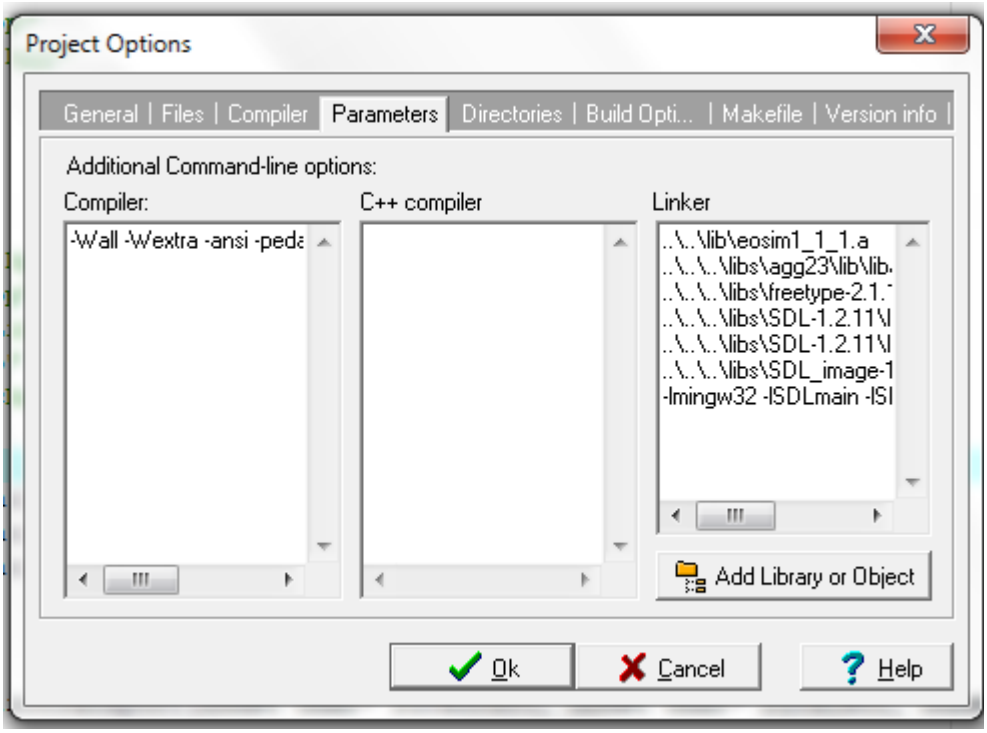


Figura D.1.5

5. Por último es necesario configurar las opciones del compilador incluyendo las librerías nativas de SDL y SDL_image para el manejo de la interfaz gráfica y así pode utilizar ciertas funcionalidades que la parte gráfica de EOSimulator no las provee. Para eso, se debe hacer clic en el menú Herramientas->Opciones del compilador e ir a la pestaña “Directorios”.

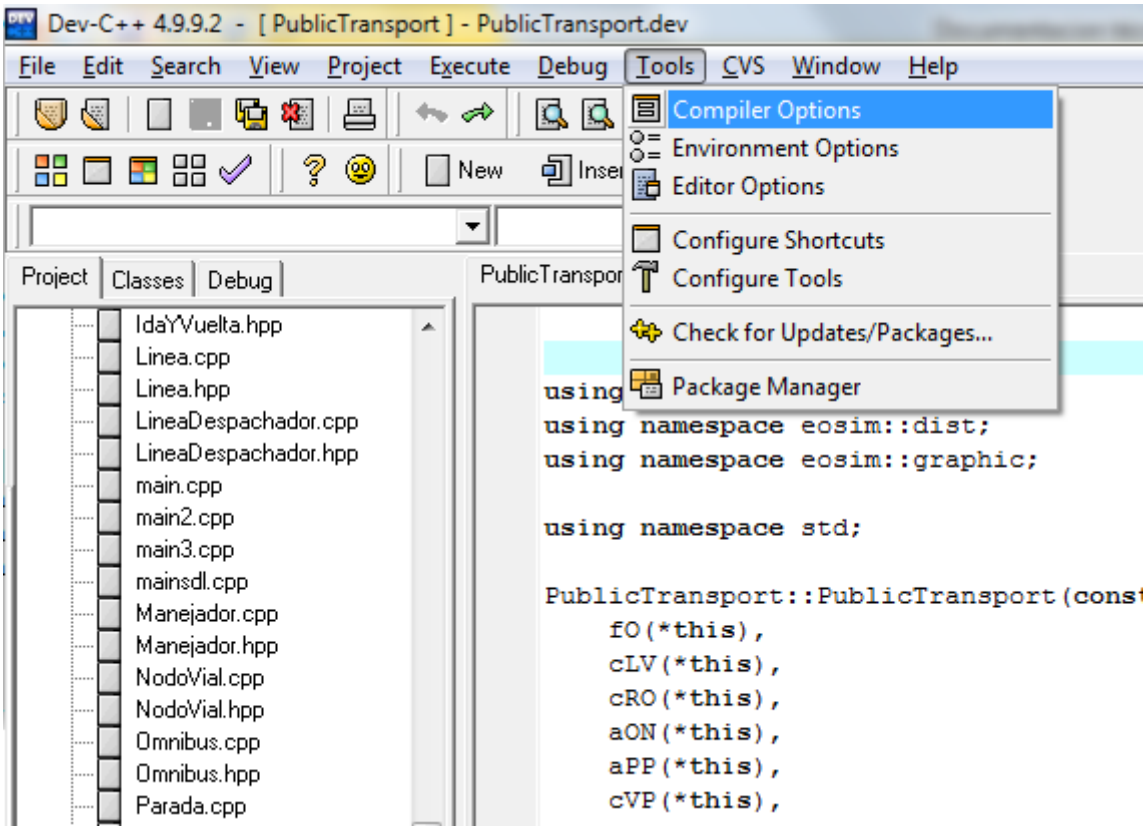


Figura D.1.6

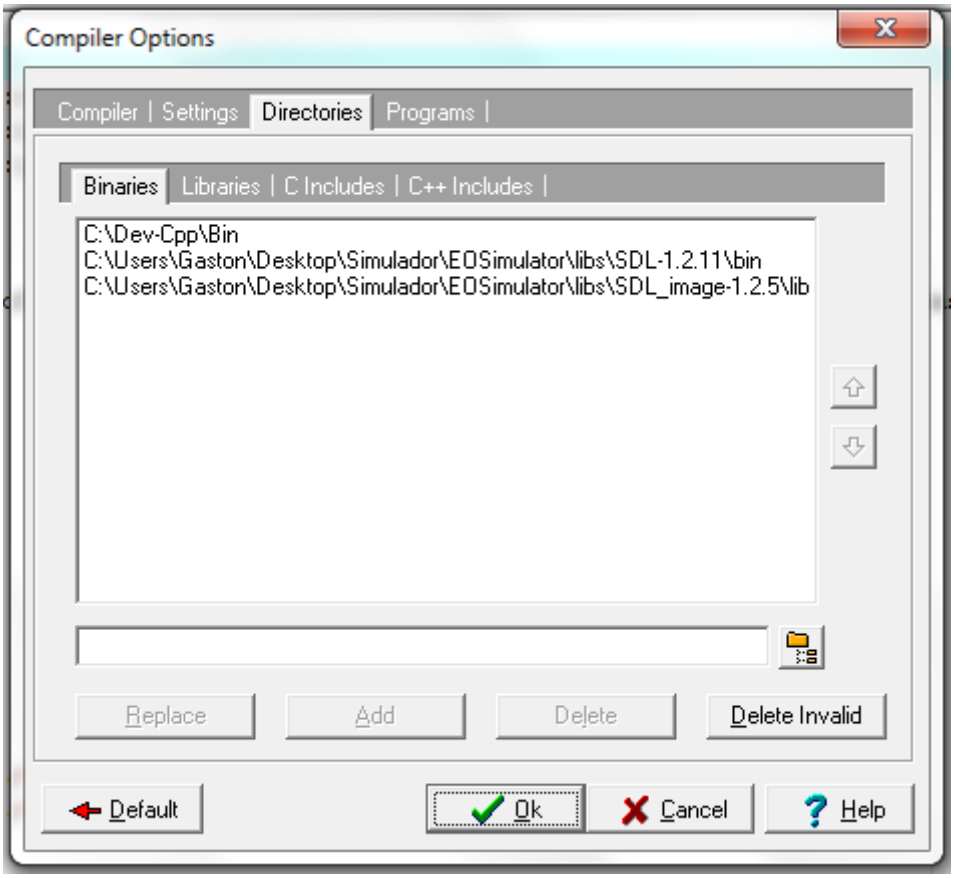


Figura D.1.7

Dentro de esta opción se debe agregar la ruta donde se encuentra el SDL (EOSimulator\libs\SDL-1.2.11\bin) y también la ruta donde se encuentra el SDL_image (EOSimulator\libs\SDL_image-1.2.5\lib).

Luego de completado estos 5 puntos logramos configurar el proyecto y ya estamos en condiciones de trabajar con el simulador.

D.3. Estructura

Con respecto a la implementación tomamos la convención de tener un archivo header y un archivo fuente por clase.

A continuación detallamos clases claves dentro del simulador que hay que tener en cuenta a la hora de querer extender funcionalidades. Cabe recordar que cada clase se compone de un archivo .cpp y un archivo .hpp.

- Archivo: Esta clase maneja todo lo referente a archivos, incluyendo lectura y escritura de los mismos.
- Constantes (solo .hpp): Este archivo header contiene todas las constantes utilizadas dentro del simulador. En caso de querer agregar nuevas constantes o querer modificar algunas existentes, se deberá acceder a este archivo.
- Estrategia: Corresponde a la clase abstracta que representa la estrategia de un pasajero. Se debe definir una subclase de la misma para crear nuevas estrategias.
- Eventos: Contiene todos los eventos del simulador. Nuevos eventos deben definirse en este archivo.
- Util: Allí se definen todo tipo de métodos extras que se quiera tener.
- PublicTransport: Corresponde al modelo del simulador.

D.4. Generación de reporte

El simulador provee una funcionalidad de generación de un reporte de la simulación en cualquier momento de la misma y cuando finaliza su ejecución. Los datos que se reportan están detallados en la sección 4.8. Para poder extender esta funcionalidad aconsejamos respetar ciertas decisiones de diseño.

En primer lugar, el método para generar los reportes se encuentra definido en el modelo. Es decir en “PublicTransport”. Por tal motivo, el código asociado a la generación de reportes se encuentra definido en el archivo PublicTransport.cpp, más específicamente, bajo el método:

```
void PublicTransport::generarReporte(){…}
```

Actualmente el reporte generado contiene información sobre las distintas líneas de ómnibus y sobre las diferentes paradas. Por decisión de diseño, optamos que tanto la clase Lineas como la clase Paradas sean las encargadas de obtener su propia información. Por tal motivo, en caso de querer obtener información de alguna otra clase, se deberá definir un método `generarReporte():String` que retorne la

información asociada a la misma. A modo de ejemplo, si se quiere obtener la información de los Pasajeros, se deberá definir en dicha clase un método `generarReporte():String` que retorne toda la información referente al Pasajero.

D.5. Nueva funcionalidad en la interfaz

Como se mencionó en el punto 4.8, la salida gráfica cuenta con un sector donde se despliega la simulación, un sector donde se muestran datos en tiempo de simulación de forma automática y un sector de referencias donde se muestran las distintas funcionalidades aplicadas a la simulación. En caso de tener la necesidad de agregar una nueva funcionalidad, este nuevo sector deberá ser modificado para desplegar a que tecla se asociará.
A todo esto, si se desea agregar una nueva funcionalidad se deberá:

1. Agregar la referencia en la interfaz gráfica: Las referencias se despliegan mediante el siguiente método ubicado en "PublicTransport":

   ```
   void PublicTransport::displayAgregarReferencias(){...}
   ```

   En este método se encuentra centralizado el despliegue de referencias. Para agregar un nuevo texto es necesario crear un TextSprite (clase de EOSimulator para desplegar texto) de la siguiente manera:

   ```
   //Agregamos el texto en pantalla  diciendo "K – nueva funcion", en
   la posición 500 en el eje Y.
   eosim::graphic::TextSprite* spr = new
   eosim::graphic::TextSprite(this->disp, "K - nueva funcion",
   REF_TEXT_SIZE, POSICION_X_REFERENCIAS,500);

     //Seteamos el texto como visible
   spr->setVisible(true);

   //Agregamos a una lista que contiene todas las referencias
   this->referencias.push_back(spr);
   ```

2. Asociar la funcionalidad a una tecla: El hecho de presionar una determinada tecla lo maneja el evento

   ```
   void TiempoSimulacion::eventRoutine() {…}
   ```

   El mismo se encuentra ubicado en "Eventos.cpp" y se debe agregar un nuevo "case" del "switch" dentro del método. A modo de ejemplo si se quiere asociar la tecla "K" a la función "nueva funcion" se deberá agregar las siguientes líneas de código:

   ```
   case SDLK_k:
   t.displayNuevaFuncion();
         break;
   ```

D.6. Agregar eventos nuevos

Suponemos que deseamos agregar un nuevo evento al simulador al cual denominaremos “NuevoEventoSimulador”. Los pasos que se deben seguir son los siguientes:

1. Declaración del evento en el archivo header Eventos.hpp. En primer lugar se debe definir una constante que identifique al evento, siempre y cuando se trate de un Evento B (para los Eventos C, este paso no es necesario). Esto puede ser cualquier texto que no esté previamente definido. A modo de ejemplo agregar:

```
const std::string EV_NES = "NUEVO_EVENTO_SIMULADOR";
```

Luego, es necesario declarar la clase asociada al evento así como también la rutina que lo atienda. La implementación del método asociado al mismo se hará posteriormente. Continuando con el ejemplo, se deben agregar las siguientes líneas de código en caso de tratarse de un Evento B:

```
class NuevoEventoSimulador: public eosim::core::BEvent {
public:
 NuevoEventoSimulador(eosim::core::Model& model);
 ~NuevoEventoSimulador();
 void eventRoutine(eosim::core::Entity* who);
};
```

En caso de tratarse de un Evento C, agregar las siguientes líneas de código:

```
class NuevoEventoSimulador: public eosim::core::CEvent {
public:
 NuevoEventoSimulador(eosim::core::Model& model);
 ~NuevoEventoSimulador();
 void eventRoutine();
};
```

2. Implementación de la rutina asociada al evento en el archivo Eventos.cpp. Aquí se implementará la rutina del evento. Tomando el ejemplo original, el código para el caso de un Evento B es el siguiente:

```
NuevoEventoSimulador:: NuevoEventoSimulador (eosim::core::Model&
model): BEvent(EV_NES, model) {
 <código asociado al constructor del evento>
}
```

```
NuevoEventoSimulador::~ NuevoEventoSimulador () {
 <código asociado al destructor del evento>
}

void NuevoEventoSimulador::eventRoutine(eosim::core::Entity* who)
{
 <código asociada a la rutina asociada al evento>
}
```

De tratarse de un Evento C, el código es el siguiente:

```
NuevoEventoSimulador:: NuevoEventoSimulador (eosim::core::Model&
model): CEvent(model) {
 <código asociado al constructor del evento>
}

NuevoEventoSimulador::~ NuevoEventoSimulador () {
 <código asociado al destructor del evento>
}

void NuevoEventoSimulador::eventRoutine() {
 <código asociada a la rutina asociada al evento>
}
```

3. Declaración del evento en el modelo. Agregar en el archivo header PublicTransport.hpp un atributo privado que haga referencia al nuevo evento. Para ello, y siguiendo con el ejemplo inicial, es necesario agregar la siguiente línea de código:

```
NuevoEventoSimulador nES;
```

4. Inicialización y registro de eventos. Estos dos pasos se deben realizar en el archivo PublicTransport.cpp. Para inicializar el nuevo evento es necesario agregar la creación del mismo en el constructor del modelo. A modo de ejemplo, el código que realiza esto es el siguiente:

```
PublicTransport::PublicTransport(const char* rutaGrafo, const
char* rutaLsol, const char* rutaDemanda, const char* rutaLineas):
    fO(*this),
    cLV(*this),
    cRO(*this),
    aON(*this),
    aPP(*this),
    cVP(*this),
    bPO(*this),
    lPC(*this),
    tSP(*this),
    tS(*this),
    sPO(*this),
    nSE(*this), //Inicializacion del Nuevo evento
```

```
    rG(*this, disp),
    disp(1300,700),
    histTiemposEsperaP("Tiempos de espera de pasajeros"),
    histTiemposEsperaO("Tiempos de espera de omnibus"){
 <constructor del modelo>
}
```

Para registrar el evento, agregar en el "init" del modelo el registro del mismo. A modo de ejemplo, agregar al método `void PublicTransport::init()`, la siguiente línea de código para el caso de los Eventos B:

```
registerBEvent(&nES);
```

Y la siguiente línea de código para los eventos C:

```
registerCEvent(&nES);
```

El orden en que son registrados los eventos, es importante. Primero deben registrarse todos los eventos B y por último deben registrarse los eventos C.

D.7. Agregar nueva regla operativa

Para agregar una nueva regla operativa para los ómnibus, de modo que sirva para modelar un determinado comportamiento del mismo, es necesario crear una clase que extienda de la clase "Regla Operativa" y sobrecargar el método que tiene asociado (aplicarRegla()).

Si deseamos agregar una nueva regla operativa cuyo nombre es "ReglaNueva" que modele la espera de un tiempo determinado de un ómnibus en una parada, entonces debemos seguir los siguientes pasos:

1. Crear un archivo de nombre ReglaNueva.hpp, declarar la nueva clase de nombre "ReglaNueva" que extienda de la clase "ReglaOperativa" y declarar el método necesario para implementar la nueva regla. Este va a ser el encargado de generar el nuevo comportamiento del ómnibus. En este ejemplo, el método sería hacer esperar al ómnibus un tiempo determinado en la parada.

```
class ReglaNueva: public ReglaOperativa{

public:
    ReglaNueva();
    ~ReglaNueva();
     void  aplicarRegla(Omnibus* o);
};
```

2. Crear un archivo de nombre ReglaNueva.cpp e implementar el método que modele el nuevo comportamiento del ómnibus. A modo de ejemplo:

```
Void ReglaNueva::aplicarRegla(Omnibus* o);{
 …
 <código asociado a este método que modela la nueva regla
operativa >
 …
}
```

3. Esta nueva regla operativa ya queda accesible para ser utilizada. Basta con asociarla a los ómnibus que se desee.

D.8. Agregar nueva estrategia

Para agregar una nueva estrategia para los pasajeros es necesario extender de la clase "Estrategia" y sobrecargar dos métodos.

Supongamos que deseamos agregar una nueva estrategia para los pasajeros cuyo nombre es "EstrategiaNueva", entonces los pasos a seguir son los siguientes:

1. Crear un archivo de nombre EstrategiaNueva.hpp, declarar la nueva clase de nombre "EstrategiaNueva" que extienda de la clase "Estrategia" y declarar los dos métodos necesarios para implementar la nueva estrategia. A modo de ejemplo:

```
class EstrategiaNueva: public Estrategia{
       public:
    EstrategiaNueva();
    ~EstrategiaNueva();
     Linea* elegirLineaDesdeOrigen(list<Linea*> l, Pasajero* p);
     double calcularTiempoMinimoDesdeOrigen(Linea* l, Pasajero* p);
};
```

2. Crear un archivo de nombre EstrategiaNueva.cpp e implementar los dos métodos. A modo de ejemplo:

```
Linea* EstrategiaNueva::elegirLineaDesdeOrigen(list<Linea*> l, Pasajero*
p){
 …
 <código asociado a este método>
 …
}

double EstrategiaNueva::calcularTiempoMinimoDesdeOrigen(Linea* l,
Pasajero* p){
 …
 <código asociado a este método>
 …}
```

4. Esta nueva estrategia ya queda accesible para ser utilizada. Basta con asociarla a los pasajeros que se desee.

E. Manual de usuario del Simulador

El objetivo de ésta sección es poder brindarle al usuario del simulador una guía de cómo utilizarlo, detallando como utilizar cada una de la funcionalidades que provee y también describiendo los pasos a seguir para agregar líneas de ómnibus.

E.1.Funcionalidades

Como se observa en la figura E.1.13, la interfaz gráfica se divide en 3 sectores bien diferenciados:

1) Sector de simulación
2) Sector de datos
3) Sector de referencias

Sector simulación:

Este sector corresponde a la visualización de la simulación. Muestra la red vial, la ubicación de las paradas, la ubicación de los centroides, los ómnibus, los pasajeros y el comportamiento de los mismos a lo largo de la simulación. Este comportamiento incluye el movimiento de los ómnibus a lo largo de la red vial, la espera de los mismos en las diferentes paradas para que los pasajeros puedan bajar y/o subir del mismo, la generación de pasajeros en los centroides y su caminata a la parada para esperar el ómnibus, el descenso de los mismos del ómnibus y la caminata al centroide destino.

Sector datos:

En este sector se encuentran datos referentes a las líneas los cuales son actualizados en tiempo de simulación de forma automática (Figura E.1.1).

Hay dos variables que son medidas, “Promedio Tiempo en Viaje” y “Promedio Tiempo en Parada”. La primera, hace referencia al promedio por línea de ómnibus del tiempo que le lleva completar el viaje, sin contemplar el tiempo que demora en las paradas para subir y bajar pasajeros. Si se trata de una línea de Ida y Vuelta, tampoco contempla el tiempo de espera para comenzar el recorrido de vuelta. El segundo hace referencia al promedio por línea de ómnibus del tiempo que consume esperando por la subida y bajada de pasajeros en las paradas a lo largo de su viaje. Ambas mediciones son actualizadas al final del recorrido completo de la línea (en el caso de un línea de Ida y Vuelta el recorrido completo es el recorrido de ida más el de vuelta).

Linea	Promedio Tiempo de viaje	Promedio Tiempo en paradas
Linea1	3250.8	49.0
Linea2	3619.0	27.0
Linea3	4072.4	42.0
Linea4	3224.4	14.0
Linea5	4695.3	112.8
Linea6	3147.3	32.0
Linea7	2703.3	21.0
Linea8	2392.3	55.0
Linea9	3403.1	18.0
Linea10	3596.7	26.8
Linea11	3660.1	102.4
Linea12	2177.5	4.0
Linea13	2105.9	22.8

Figura E.1.1

Sector referencias:

Este sector muestra las diferentes acciones que se pueden realizar sobre la simulación utilizando el teclado para llevarlas a cabo (Figura E.1.2).

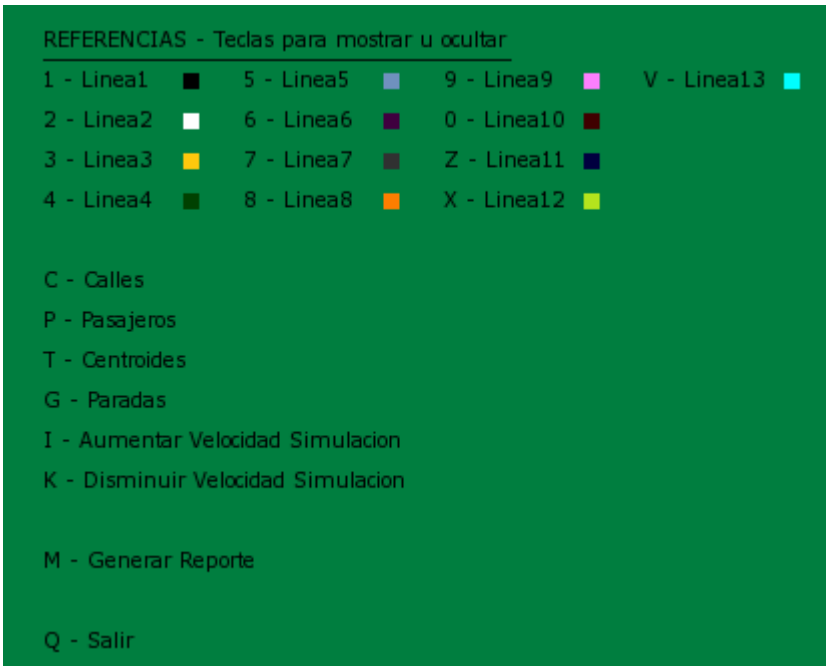


Figura E.1.2

En este manual pasaremos a detallar este sector:

- Líneas de ómnibus: Se muestran las líneas de ómnibus existentes y se las asocia a un número/letra y también a un color. El número/letra hace referencia a la tecla que hay que presionar para ocultar o mostrar todos los ómnibus de la línea en cuestión. El color refleja cómo se visualiza los ómnibus de dicha línea durante la simulación. A modo de ejemplo: Al presionar la tecla “1” se muestra/oculta Linea1, al presionar la tecla “2” se muestra/oculta Linea2 y así sucesivamente.

- C - Calles: Al presionar la tecla “C”, se muestra la red vial completa o se muestra parte de la misma. Esta parte de la red vial corresponde a aquellas calles que forman parte de algún recorrido de alguna línea.

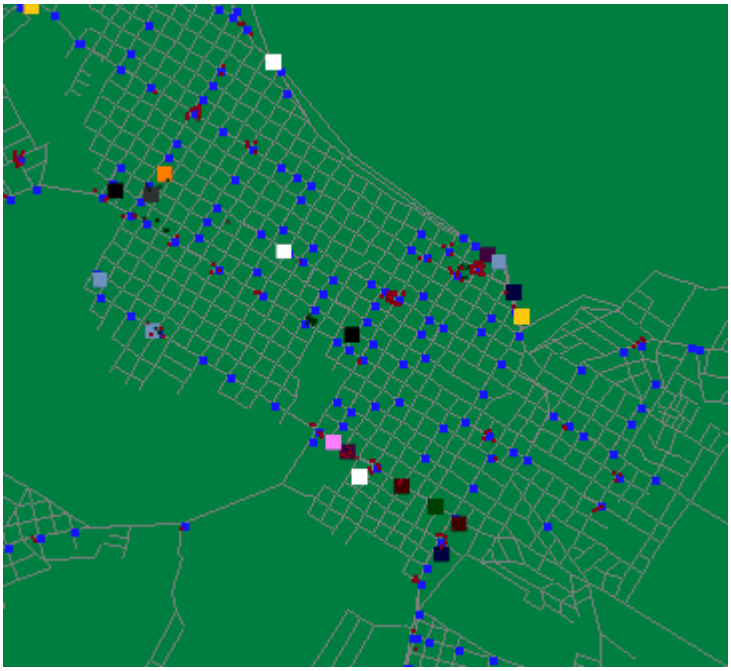


Figura E.1.3

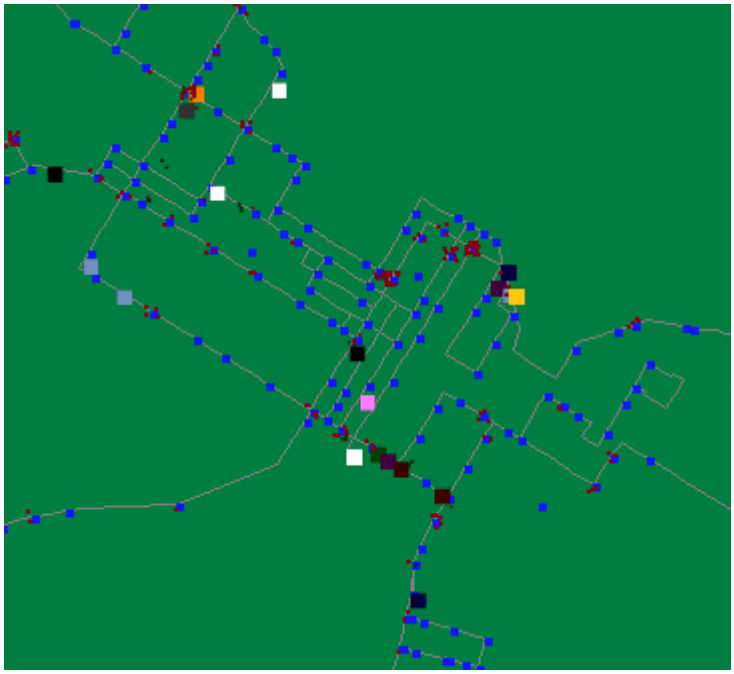


Figura E.1.4

- P- Pasajeros: Al presionar la tecla “P”, se muestran u ocultan los pasajeros del sistema. Se visualizan en color rojo cuando son creados, mientras realizan la caminata a la parada origen y mientras esperan en la misma. Se visualizan en verde, cuando descienden del ómnibus y realizan la caminata al centroide destino.

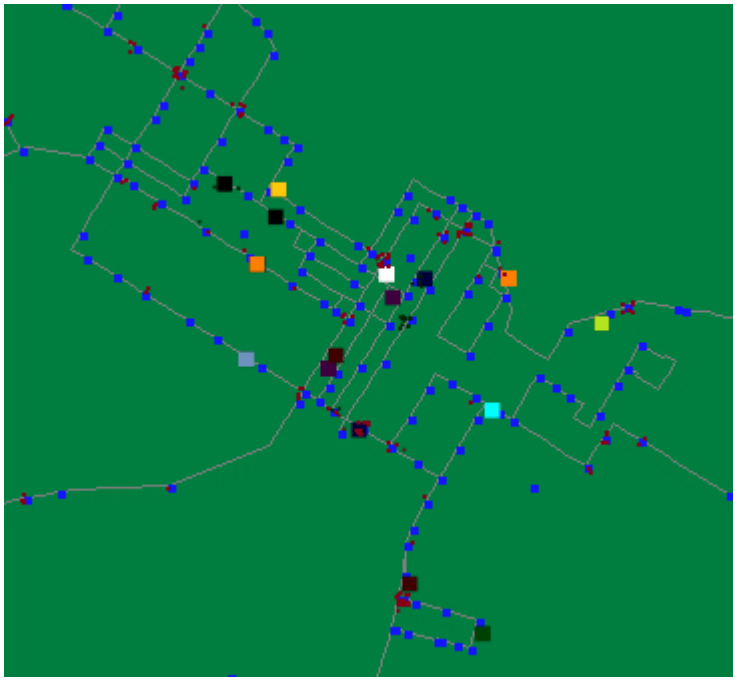


Figura E.1.5

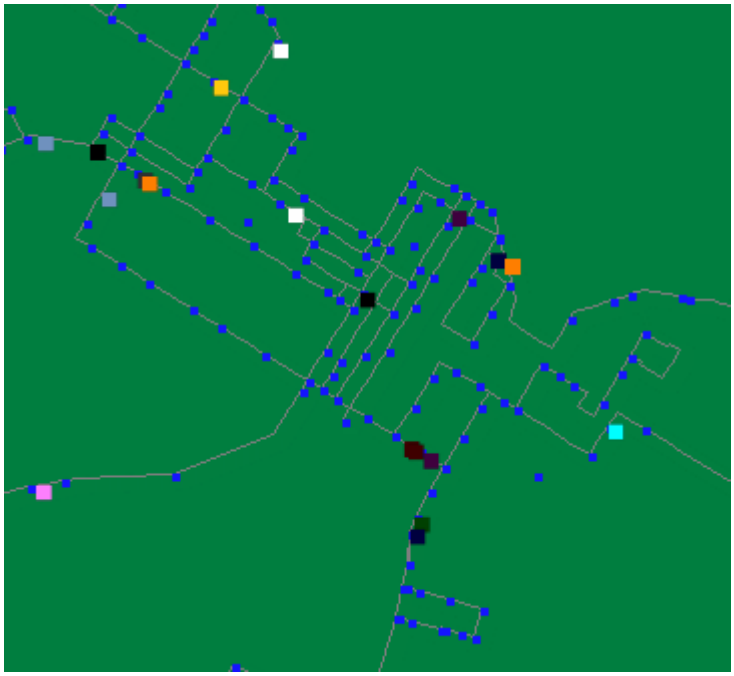


Figura E.1.6

- T - Centroides: Al presionar la tecla “T”, se muestran o se ocultan todos los centroides del sistema y las caminatas que salen de los mismos.

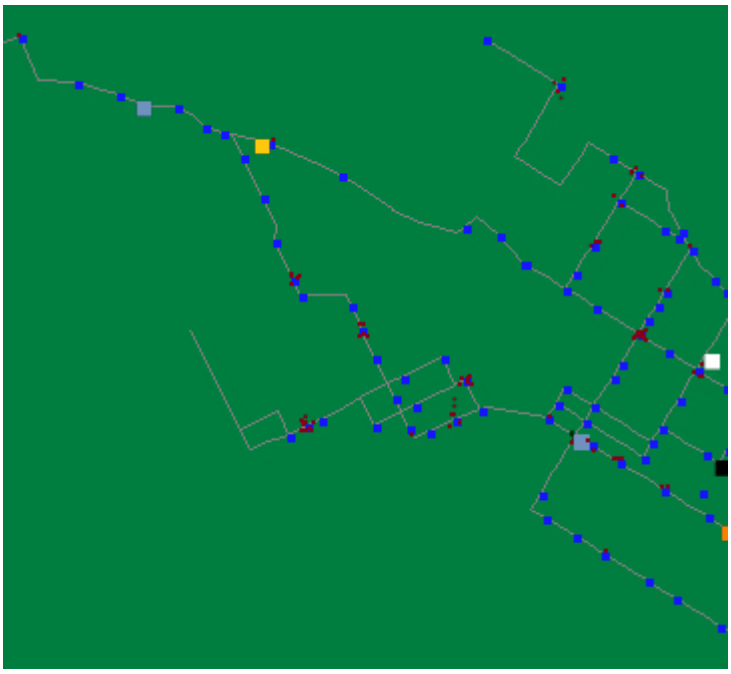


Figura E.1.7

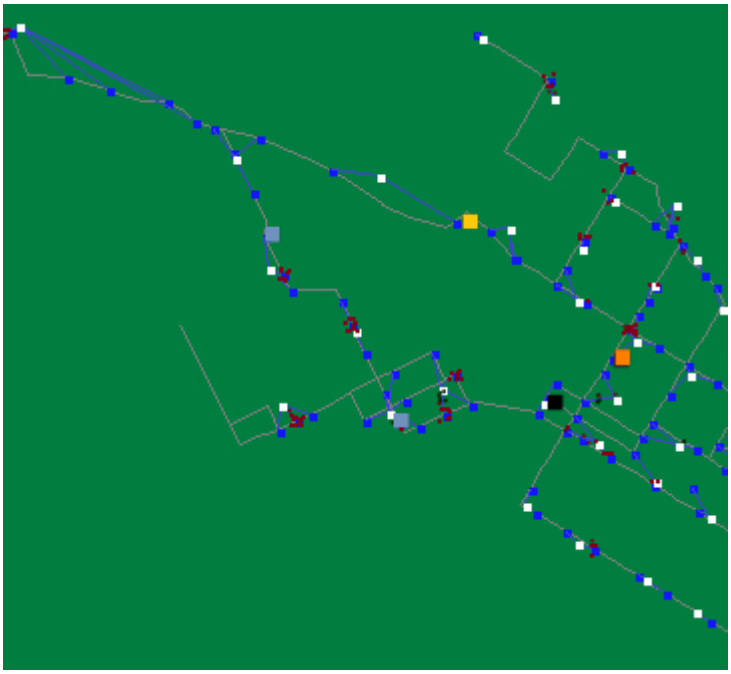


Figura E.1.8

- G - Paradas: Al presionar la tecla “G”, se muestran o se ocultan todas las paradas del sistema.

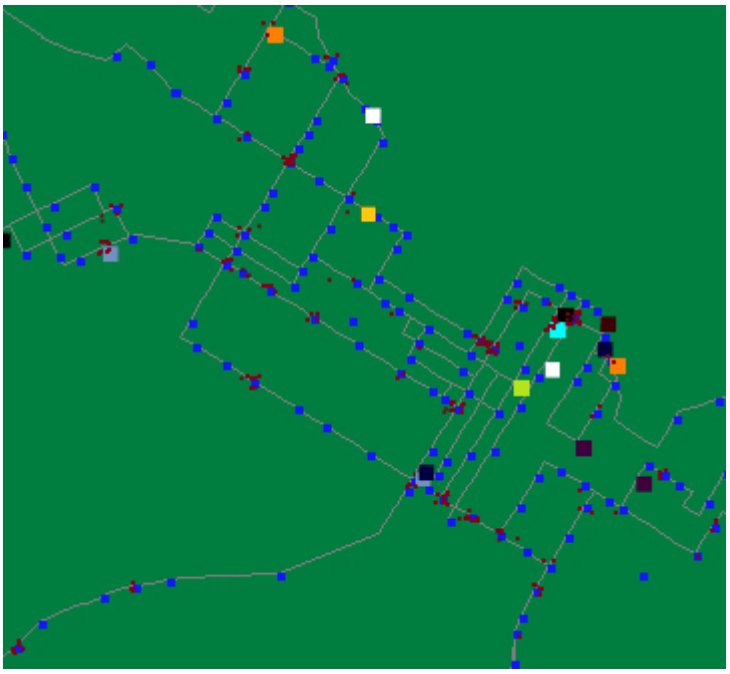


Figura E.1.9

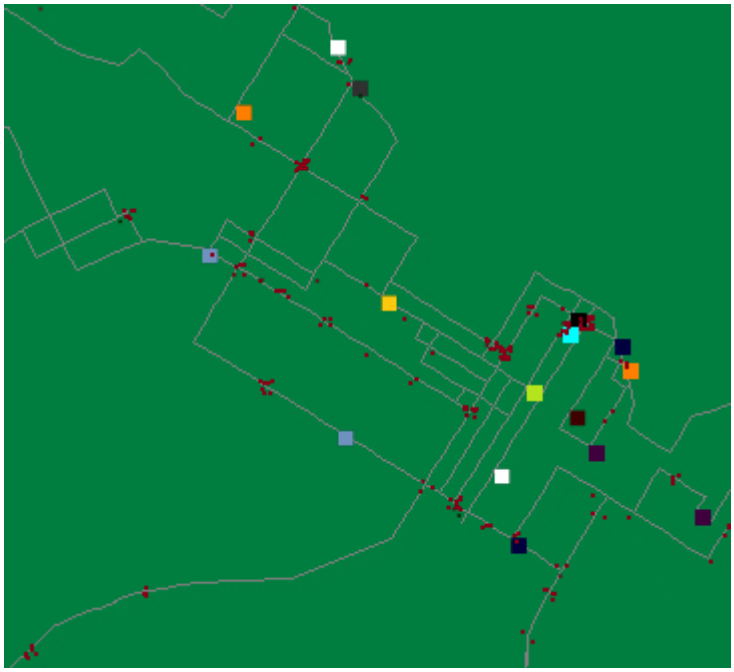


Figura E.1.10

- I - Aumentar Velocidad Simulación: Al presionar la tecla “I”, aumenta la velocidad de la simulación. Este aumento no es infinito sino que está acotado por una constante que determina la velocidad máxima permitida.

- K - Disminuir Velocidad Simulación: Al presionar la tecla “K”, disminuye la velocidad de la simulación. Esta disminución no es infinita sino que está acotada por una constante que determina la velocidad minima permitida.

- M - Generar Reporte: Al presionar la tecla “M”, se genera un reporte del estado actual de la simulación. Se despliega fecha y hora y tiempo de simulación en el que fue generado dicho reporte. El reporte es generado en la carpeta “reportes”, ubicada en la raíz del simulador. El nombre del mismo es identificado por la fecha y la hora en el que fue realizado.

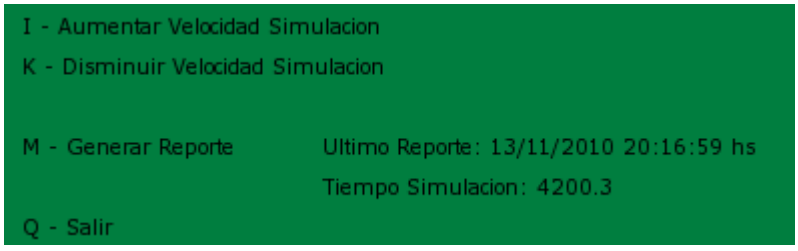


Figura E.1.11

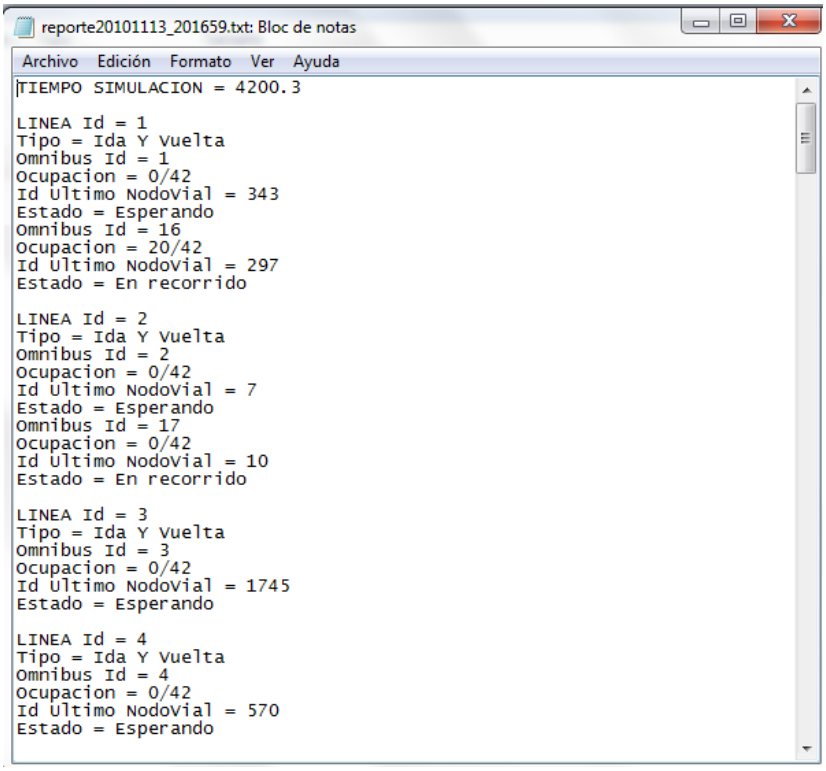


Figura E.1.12

- Q - Salir: Al presionar la tecla “Q”, se aborta la simulación y finaliza la aplicación. Una vez que finaliza la simulación, es decir, se cumple el tiempo estipulado para la misma, ésta queda pausada a la espera de que se presione “Q” para salir y finalizar la aplicación.

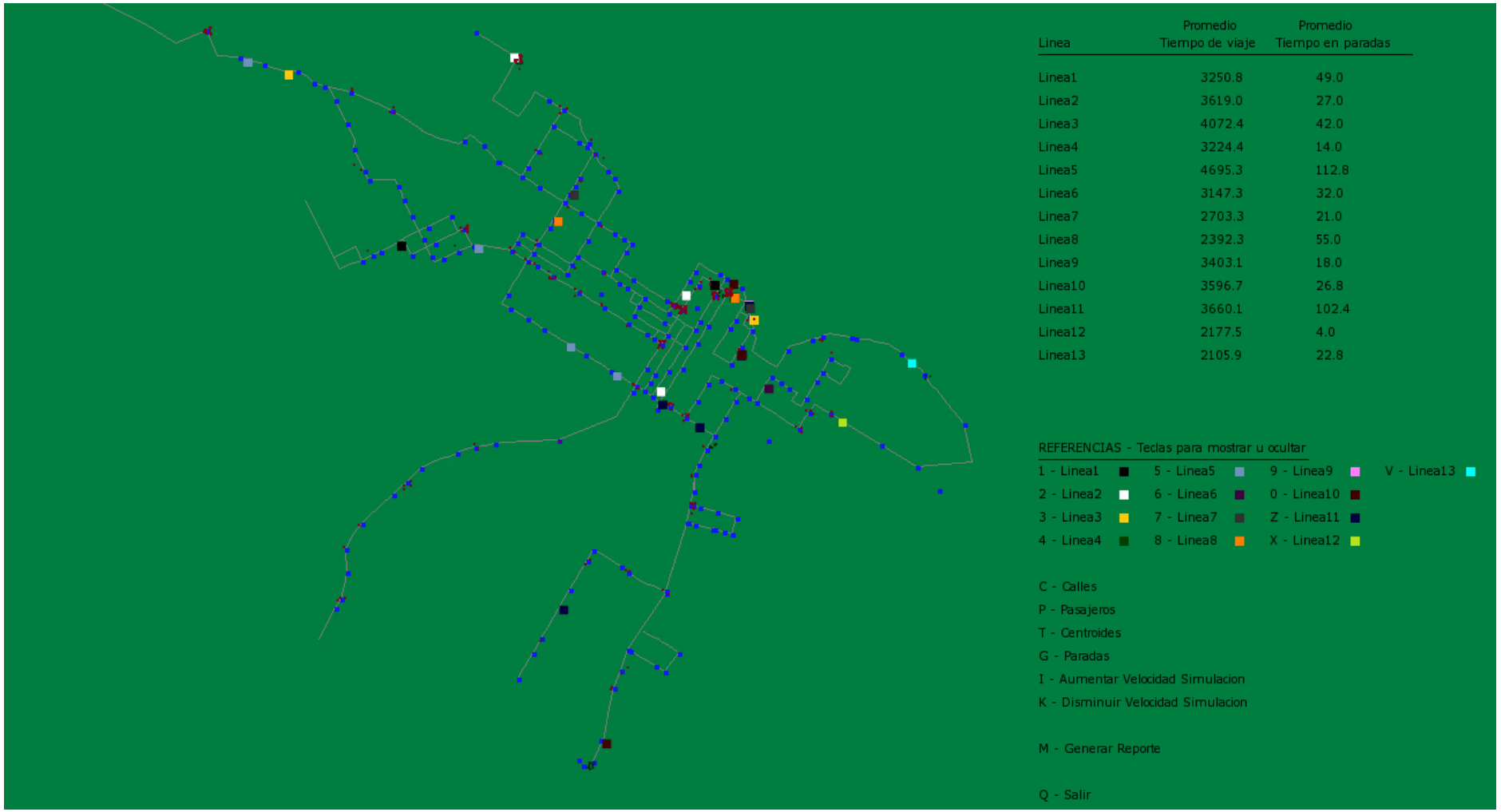

Linea | Promedio Tiempo de viaje | Promedio Tiempo en paradas
Linea1 3250.8 49.0
Linea2 3619.0 27.0
Linea3 4072.4 42.0
Linea4 3224.4 14.0
Linea5 4695.3 112.8
Linea6 3147.3 32.0
Linea7 2703.3 21.0
Linea8 2392.3 55.0
Linea9 3403.1 18.0
Linea10 3596.7 26.8
Linea11 3660.1 102.4
Linea12 2177.5 4.0
Linea13 2105.9 22.8
REFERENCIAS - Teclas para mostrar u ocultar
1 - Linea1
2 - Linea2
3 - Linea3
4 - Linea4
5 - Linea5
6 - Linea6
7 - Linea7
8 - Linea8
9 - Linea9
0 - Linea10
Z - Linea11
X - Linea12
V - Linea13
C - Calles
P - Pasajeros
T - Centroides
G - Paradas
I - Aumentar Velocidad Simulacion
K - Disminuir Velocidad Simulacion
M - Generar Reporte
Q - Salir

Figura E.1.13

E.2.Nuevas líneas de ómnibus

Para agregar una nueva línea de ómnibus se deben seguir los siguientes pasos:

1. Generar en igoR-tp el recorrido (si se trata de una línea circular) o los recorridos (si se trata de una línea de ida y vuelta).

2. Se debe actualizar el archivo de entrada denominado líneas.txt (sección 4.9) agregando una nueva entrada que contenga la información de la nueva línea. En el mismo se debe especificar el tipo de línea, que recorrido o recorridos la conforman, la frecuencia de la línea y si se trata de una línea de ida y vuelta, se debe especificar el tiempo necesario para pasar del recorrido de ida al de vuelta.

3. Finalmente, se deben agregar 2 nuevas imágenes en la carpeta “images” ubicada en la raíz del simulador. A modo de ejemplo, si contamos con 11 líneas y deseamos agregar una nueva, las imágenes que debemos agregar son “omnibusVacio12.jpg” y “ominbusLleno12.jpg” que corresponden a la representación iconográfica de los ómnibus de esa nueva línea cuando no va lleno y cuando va lleno en su capacidad respectivamente. Estas imágenes deben ser 8x8 px y en formato jpg.

F. Actas

En este anexo se adjuntan las actas tomadas durante las reuniones que mantuvimos con los tutores.

Acta

Fecha: 27/04/2009

Hora: 17:00 – 17:45

Participantes:

- Marita
- Patricia
- Gastón

Temas tratados:

1. Descripción del proyecto

Marita nos contó de qué se trata el proyecto y de cómo deberíamos encararlo. Se mencionaron las posibles etapas del mismo que nosotros deberíamos definir y luego validarlas con los tutores. En este momento estaríamos en la etapa de autoestudio del problema TNDP y de la Simulación a Eventos Discretos aplicada al problema de modelado de sistemas de transporte público, en particular a la interacción de usuarios con un conjunto de líneas de buses. Se destacó la problemática de modelar la conducta de los pasajeros del caso de estudio de la ciudad de Rivera ya que no se cuenta con los datos de la realidad.

Nos recomendó ir analizando las librerías de simulación existentes (EOSimulator, SimPP, etc) la cual utilizaremos para realizar nuestro proyecto.

También nos aconsejó que si queremos terminar en diciembre sería bueno dedicarle 4 horas por día.

Una característica muy importante del modelo es que debe ser genérico.

2. Documentación final

Se comentó a grandes rasgos los puntos que debería tener el informe final y las ventajas de ir documentando las tareas realizadas desde el comienzo del proyecto, como por ejemplo anotar los detalles de las referencias y en el caso de que el material esté en la web, anotar el link y guardar el material. Marita nos enviará el formato para escritura de artículos.

3. Reuniones

Las reuniones se definieron cada 15 días, los lunes a las 17 horas. Se deberá mandar siempre mail para confirmar o suspender la reunión. Luego de cada una de ellas, se deberá mandar el acta correspondiente.

Tareas:

1. Patricia y Gastón
    a. Definir un cronograma y enviarlo por mail a los tutores.
    b. Buscar material del problema TNDP
    c. Experimentar con los simuladores que ya descargaron y además seguirán buscando otros.
    d. Estudiar las librerías de simulación existentes.
    e. Enviar el acta de la reunión del día de la fecha.

2. Marita nos mandará:
    a. Formato para escritura de artículos.
    b. Bibliografía sobre TNDP y algún otro material que considere relevante.

Próxima reunión: 11/05/2009

Acta

Fecha: 11/05/2009

Hora: 9:30 – 10:20

Participantes:

- Marita
- Patricia
- Gastón

Temas tratados:

1. Validación de cronograma

Marita está de acuerdo con el cronograma propuesto y sugirió que mientras estemos en la etapa de implementación, vayamos estudiando y analizando el caso de estudio de la ciudad de Rivera.

2. Fuentes de Proyectos: igoR-tp y SimPP

En caso de que no se encuentren en la biblioteca del Inco, se le enviará mail a los tutores para que nos faciliten este material.

3. Modelo de Proceso de Desarrollo

No hay obligación de documentar esto pero algunos tribunales se fijan en eso y es un plus si lo documentamos.

4. Detalles sobre el comportamiento de pasajeros y líneas de ómnibus

Los pasajeros van a tener asignados un conjunto de líneas de ómnibus y se determinará cuál tomar, por ejemplo, de acuerdo al destino que va, cuál le conviene más.

Si un pasajero está en la parada y vienen dos buses que le sirven se plantea la problemática de cuál tomar. Esto podría estar determinado por azar, por tiempo de llegada a destino, el que geográficamente le quede más cerca al pasajero, el bus que tenga menos gente aglomerada para subir, etc.

Si un pasajero está esperando el ómnibus y este no para o si va muy lleno, el pasajero podría esperar otro ómnibus o irse. Estas hipótesis se definirán cuando se especifiquen los requerimientos así como también el tiempo de espera máximo del pasajero.

Es muy importante que se modele el comportamiento de los pasajeros como entidades individuales y en particular, considerar la existencia de pasajeros heterogéneos (jóvenes, ancianos, discapacitados, etc) y si el ómnibus permite que se pueda subir con sillas de ruedas o no.

El modelo será unimodal, sólo se modelarán los buses y no otro tipo de vehículos.

Los tiempos entre paradas estarán determinados por una distribución la que estará definida por factores como: calidad de la calle, velocidad permitida, etc.

Es fundamental que se modele el trasbordo. Se deberá estudiar cómo se definen los trasbordos y que algoritmos hay para esto. En el caso de Rivera la ciudad es como un polígono, todas las líneas van al centro. En la actualidad si se quiere ir de un barrio a otro, el pasajero tendría que hacer trasbordo y esto no es posible.

También se deberá modelar que el bus cumpla con sus horarios. Por ejemplo, si va adelantado deberá ir más lento o esperar en alguna parada y si va atrasado, no parará en algunas paradas.

Se deberá buscar un conjunto de características que definan lo que es una línea de ómnibus.

No se deberá manejar la asignación de choferes a los buses. Pero si se deberá tener en cuenta: la existencia de choferes con distintas características, cada cierto tiempo un determinado ómnibus va a mantenimiento, cambio de choferes, etc.

En especial, se deberá enfatizar en las características de choferes y buses que afectan el comportamiento del pasajero.

El módulo a desarrollar debe comunicarse con la interfaz igoR-tp, y de ésta se tomará la matriz origen – destino, y el recorrido de las líneas. Sobre este tema se profundizará más adelante. Marita nos mandará mail sobre esto.

Tareas:

1. Patricia y Gastón
    a. Buscar las fuentes de los proyectos igoR-tp y SimPP
    b. Definir biblioteca de simulación
    c. Resumir la información leída hasta el momento y mandar los links de los simuladores encontrados
    d. Mandar mail confirmando la próxima reunión
    e. Enviar el acta de la reunión del día de la fecha

2. Marita nos mandará:
    a. Material sobre el modelo de asignación
    b. Los capítulos de los dos Handbooks de OR: el capítulo de Urban and Air Transportation del Volumen de OR in the Public Sector Odoni, Rousseau y Wilson), y el Public Transit de Desaulniers y Hickman, solo la parte de TNDP y modelos de asignación

Próxima reunión: 25/05/2009 Hora: 17:00

Acta

Fecha: 26/05/2009

Hora: 17:15 – 18:20

Participantes:

- Marita
- Patricia
- Gastón

Temas tratados:

1. Comunicación con IgoR-tp

Se puede definir en base al modelo y a la librería que se va a utilizar una comunicación con la interfaz generada por IgoR-tp. Tal vez se podría realizar através de manejo de archivos.

Es recomendable hacer todo en forma modular, es decir, pensar como solucionar la optimización, la simulación, la interfaz, la comunicación, etc.

2. Visualización

La vista 3D supera el alcance del proyecto, por lo que se propone realizar algún grafico "simple". Por ejemplo que se pueda visualizar entre otras cosas trasbordos y consultas. La salida gráfica es importante pero por ahora tenemos que dejarlo un poco de lado. Se especificará más adelante.

3. Libros y Revistas

Se recomendó leer un libro que al comienzo tiene una tabla con las características que tienen cada una de las librerías de simulación.

También se hablo de ORMS que contiene información al día de distintos lenguajes de simulación y características relevantes de cada uno. Marita lo pidió para nosotros en la biblioteca del Inco. Debemos entregarlo en menos de una semana.

Tomando toda esta información debemos buscar que características debería tener el lenguaje de simulación para poder realizar el modelo de comportamiento de los pasajeros. A modo de ejemplo preguntarse: ¿Como serían tratados los pasajeros? ¿Entidades distintas? ¿Tiene el lenguaje la capacidad de hacerlo?

Además, se pidió en la biblioteca del Inco el libro "Promodel" el cual es interesante darle una leída porque tiene conceptos importantes.

4. Librerías de simulación

Marita nos comento que simPP no es conveniente ya que depende mucho de otras librerías. Se hablo de las librerías que habíamos estudiado. Debemos profundizar en este tema al menos una semana mas, tomando como información la documentación brindada por los libros y revistas comentados en el punto anterior.

Parece ser que EOSimulator va a ser la librería a utilizar, de todas formas es importante realizar el ejercicio anterior.

Hablamos de JavaSim. Marita no la conoce mucho, pero se menciono que no estaría mal que usáramos esta dado que Java es el lenguaje con el cual mejor nos llevamos. De todas formas no es el único punto importante a la hora de elegir la librería de simulación.

5. Fuentes IgoR-tp

En este momento no se encuentra en la biblioteca del Inco,

Marita o Manuel Martínez (integrante del proyecto IgoR-tp) nos puede facilitar una copia de la última versión de IgoR-tp.

6. Especificación de Requerimientos

Empezaremos con esta tarea luego de que venga Antonio.

Tareas:

1. Patricia y Gastón
    a. Buscar las fuentes de los proyecto IgoR-tp
    b. Estudiar una semana mas librerías de simulación
    c. Mandar características de las librerías que nosotros consideramos que hay que tener en cuenta a la hora de la elección
    d. Mandar mail confirmando la próxima reunión
    e. Enviar el acta de la reunión del día de la fecha

2. Marita nos mandará:
    a. Material de un libro (que no recuerda el nombre) en el cual al comienzo contiene una tabla con características de librerías de simulación.

Próxima reunión: Dentro de dos semanas. A definirse por mail.

Acta

Fecha: 03/08/2009

Hora: 17:30 – 19:00

Participantes:

- Marita
- Antonio
- Patricia
- Gastón

Temas tratados:

1. Situación actual

Le contamos a Antonio lo que habíamos hecho hasta el momento (estudio sobre: el problema a resolver, librerías, simuladores, reuniones con Marita) y la elección de la librería de simulación EOSimulator.

2. Cronograma

Presentamos una nueva versión del cronograma. Les pareció que estaba bien y Marita sugirió que el caso de estudio podría llevar más de un mes. De todos modos, se deberá actualizar el cronograma para que contemple la modificación de igoR-tp.

3. IgoR-tp

Actualmente, la red vial está representada por un grafo que se obtiene uniendo los centroides de las distintas zonas. Los buses no circulan por esas aristas del grafo, sino que circulan por calles. Por lo que se tiene la necesidad de incluir a la representación actual de la red vial, la información de las calles (p.j.: el sentido) y de los cruces así como también información sobre las paradas de ómnibus. Por esta razón, se agregará un módulo nuevo a igoR-to que le permitirá al usuario construir la red vial de tres formas distintas. Más precisamente, se le dará la opción al usuario de elegir el detalle del grafo que representará la red vial con los siguientes tipos de nodos:

- Centroides
- Centroides y cruces de calles
- Centroides, cruces de calles y paradas de ómnibus.

El primer ítem ya está implementado. Empezaremos implementando el segundo ítem ya que el tercero es más complicado porque requiere una capa de paradas. Para realizar esta tarea, nos sugirieron elaborar los casos de uso, como por ejemplo: dar de alta una zona. También es necesario pensar como conectar centroides de zona a la red vial.

En un principio, el peso de las aristas hace referencia al tiempo.

El grafo define la red vial, y sobre la red vial se definen las líneas de ómnibus. Éstas últimas son definidas en el módulo de manipulación. En este momento es bastante rudimentaria la visualización de recorrido de buses. La frecuencia de los buses también se define en el módulo de manipulación.

De igoR-tp vamos a tomar el grafo y la matriz de origen – destino. También tenemos que ir pensando que información va a necesitar el modelo de simulación y de donde la podemos obtener, por ejemplo: del módulo de construcción o del de manipulación (líneas de ómnibus, frecuencias).

En este momento, se cuenta con una representación del grafo como se mencionó anteriormente y con un algoritmo para evaluar. Pero nosotros debemos pensar el problema de forma más general, como si contáramos con varias representaciones de la red vial y con varios algoritmos para evaluar y la idea es que el simulador sea una opción más a la hora de evaluar, con la particularidad de la salida gráfica del simulador.

Antonio realizó una demo sobre igoR-tp.

Antonio nos dio una versión de igoR-tp, el caso de estudio de Rivera y una digital del libro con material sobre representación de redes de transporte urbano.

Tareas:

1. Patricia y Gastón
    a. Estudiar el Módulo de Construcción de IgoR-tp. Pensar como armar el grafo según lo hablado en la reunión.

b. Mandar mail confirmando la próxima reunión
c. Enviar el acta de la reunión del día de la fecha

Próxima reunión: La próxima semana, a definirse por mail.

Acta

Fecha: 12/08/2009

Hora: 17:30 – 18:10

Participantes:

- Antonio
- Patricia
- Gastón

Temas tratados:

1. Construcción de cruces, paradas y calles

La representación de los cruces se van a realizar con un simple punto que representa la intersección de las calles y nos se va a ser tan detallista en este tema (en la versión en pdf del libro "Urban Transportation Network" de Yosef Sheffi que nos mando Antonio muestra otra forma de representarlo con 4 puntos). De momento la capa de red vial no tiene información alguna, esta sirve como referencia para la construcción de las zonas.

La idea es que a partir de esta capa nosotros generemos automáticamente los cruces (puntos) así como también las conexiones entre los mismos (líneas), que representaría las calles, de acuerdo a la red vial. Asumiremos que las calles son de doble sentido.

Es responsabilidad del usuario construir esta red vial de acuerdo a sus necesidades.

Para la construcción de las paradas se determinó que en función de la capa de cruces de calles, el usuario pueda marcar cuales son paradas y cuales no, por ejemplo haciendo clic sobre el cruce en cuestión y determinando si es parada o no.

2. Conexión entre zonas

Esto no lo tenemos que realizar pero si debemos dar la posibilidad de crear arcos de cada centroide a paradas.

3. Información de capas

Antonio nos mostró una forma de poder ver los atributos que tiene una capa determinada (se puede observar a través de una carpeta temporal que se crea al correr el programa igoR-Tp). Los archivos dbf contienen dicha información y pueden ser abiertos con excel. Además hay una herramienta, MapWindow, que no solo permite ver los atributos de las distintas capas sino que también permite observar su representación. También es conveniente observar la documentación técnica del igoR-Tp para poder ver las operaciones que se pueden hacer sobre las distintas capas.

4. Trasbordos

El tema de los trasbordos es interno de los algoritmos, no lo tenemos que modelar en esta parte de construcción.

Tareas:

- Patricia y Gastón
  - Estudiar el Módulo de Construcción de IgoR-tp. Implementar la nueva representación del grafo.
  - Mandar mail confirmando la próxima reunión
  - Enviar el acta de la reunión del día de la fecha

Próxima reunión: A definirse por mail.

Acta

Fecha: 10/09/2009

Participantes:

- Antonio
- Patricia
- Gastón

Temas tratados:

1) Validación de la capa de Nodos Viales

Se discutieron las dos formas que se implementaron para generar la capa de nodos viales. En ambos casos, si el nodo a agregar ya pertenece a la capa, entonces no se agrega, y si no pertenece, se agrega. La diferencia está en cómo determinar si el nodo pertenece a la capa o no. En el primer caso, queda determinado por la intersección de la capa de nodos viales con un buffer del punto más una cierta tolerancia. Estaría bueno que la tolerancia se ingrese por parámetro y no que sea un valor fijo como está hecho ahora.

La otra opción para verificar si un punto de la capa de red vial, pertenece o no a la capa de nodos viales es comprobar si el valor de "fromNodo" (o "toNodo", según el punto que se quiera agregar), es igual al valor de "ID_NODOVIAL" de la capa de nodos viales.

Se concluyó que la mejor opción es la segunda ya que brinda más exactitud.

2) Asignación de paradas a nodos viales

Se debe diferenciar, de alguna manera (por ejemplo con otro color), las paradas de los nodos viales. Posiblemente tengamos que ir generando otra capa.

No hay requisitos sobre la interfaz gráfica para esta funcionalidad pero una opción podría ser un botón que de la posibilidad de comenzar a marcar/desmarcar las paradas.

3) Grafo

Se debe mirar el grafo que se arma en igoR-tp. Este tiene nodos y arcos (puede tener atributos, no estructurados) y sirve como entrada para los diferentes algoritmos y en particular servirá como entrada para nuestro simulador. Deberíamos reunir toda la información de los distintos shapefiles que se van construyendo y "dbf" en dicho grafo.

4) Cambios al menú

Antonio sugirió tener un solo ítem en el menú principal que contenga las funcionalidades de zonas, conexiones y nodos viales en vez de tenerlos como ítems separados en el menú principal como está actualmente.

Tareas:

- Patricia y Gastón
  - Implementar la asignación de paradas a la capa de nodos viales.
  - Estudiar el grafo de igoR-tp.
  - Organizar "Zonas", "Conexiones", "Nodos Viales" en un mismo ítem de la barra principal de menú, ya que son elementos de la Red Vial.
  - Enviar el acta de la reunión del día de la fecha

Próxima reunión: Jueves 17 a las 17:30.

Acta

Fecha: 01/10/2009

Participantes:

- Marita
- Antonio

- Patricia
- Gastón

Temas tratados:

1) Capa de Red Vial

Se eliminó el ingreso del parámetro "velocidad media del bus" ya que sólo se utilizaba en la capa de conexiones la cual se quitó del sistema. Pero este parámetro debe permanecer, ya que ahora los buses van a circular por la capa de red vial y el costo de los arcos es el tiempo que lleva en recorrer la distancia entre los nodos, necesitando así el valor de la velocidad media del bus para su cálculo.

Se discutió la complejidad de considerar el sentido de las calles de la red vial. Además tenemos que revisar cómo están creados estos parámetros para detectar futuros errores.

Una primera opción que se habló para resolver estos inconvenientes (campo costo en capa de red vial y sentido de las calles) fue hacer la capa de red vial en una capa diferente, de forma de hacer una simplificación al problema ignorando los "sentidos" de la capa y resolviendo el costo de la misma en unidades de tiempo.

Finalmente se llegó a la conclusión de que toda la información se almacene en la capa de red vial original y al momento de crear un nuevo proyecto se debe agregar a esa misma capa el atributo de costo en unidad de tiempo (minutos).

2) Cambio de nombre

Se propuso cambiar el nombre de la capa denominada "Conexión peatonal" correspondiente a las conexiones centroide-parada por "Conexión de accesos" o simplemente "Accesos".

3) Centroides

Se explico la razón de la existencia de los cetroides. En una zona determinada se dice que la demanda esta distribuida uniformemente, por dicho motivo es que se toma el centroide (baricentro) como punto de concentración de demanda. El costo de caminata (conexión centroide-parada) son promedios de lo que camina la persona en la zona. También se asume que tomar los datos tan finos por parada en particular no es gran medida ya que por ejemplo, es importante saber cuanto caminó la persona hasta llegar a la parada.

4) Levantar proyectos con formato viejo

En este momento se puede levantar proyectos con el formato viejo pero estos tienen la particularidad de que al levantarlos se pierden las conexiones. Estas conexiones no existen más ya que son sustituidas por las calles de la red vial. Se aconseja que al levantar esos proyectos viejos se alerte de que faltan las conexiones.

5) Distancia Euclidea y Manhattan

En este momento el tipo de cálculo de la distancia es seteado una única vez al momento de crear el proyecto. Se considera muy bueno poder cambiar esta opción para que pueda ser modificada en cualquier momento ya que podrían haber lugares en que las manzanas y las

calles son considerablemente distintas por lo que requerirían un calculo de distancia de tipo diferente. Por ejemplo, las personas en el centro no caminan a la misma velocidad que en el barrio Pocitos.

6) Uniformizar datos

Se debe uniformizar todos los datos, es decir que las velocidades siempre se midan en Kilómetros por hora, los tiempos en minutos y las distancias en metros. El manual de usuario del igoR-tp contiene toda esta información.

7) Deshacer - Rehacer

Al menos la función de Deshacer debe estar. Están funcionalidades se deberán implementar para las operaciones simples. Se considera vital en toda interfaz del estilo de igoR-tp que presente esta opción. Se aconsejó que esta funcionalidad tuviera como mínimo un paso.

8) Repositorio

Se va a solicitar un repositorio para que podamos versionar y respaldar el proyecto.

Tareas:

- Patricia y Gastón
  - Finalizar implementación del módulo de construcción de igoR-tp.
  - Recordar a Antonio y Marita de pedir prórroga.
  - Enviar el acta de la reunión del día de la fecha

Próxima reunión: Jueves 15 de Octubre a las 17:30.

Acta

Fecha: 15/10/2009

Participantes

- Antonio
- Patricia
- Gastón

Temas tratados:

1) Módulo de Construcción

Se validó con Antonio ciertas consideraciones tomadas sobre el módulo de construcción. En primer lugar se decidió que al cambiar la velocidad de la flota en cualquier momento, se recalcule y se actualice el costo de toda la capa de red vial.

También se extendió la funcionalidad del Deshacer-Rehacer más específicamente en lo que tiene que ver con el color, estilo y tamaño de las diferentes capas. Finalmente se consideró correcto que cuando se desea desactivar una parada que es parte de un acceso, se notifique que no es posible realizar dicho suceso mediante el mensaje correspondiente.

Nos dimos cuenta que la herramienta "Notas del mapa" no funciona por lo que debemos ver si en la versión de Mayo de igoR-tp está funcionando y solucionar dicha herramienta.

También se mencionó que al realizar algún cambio en algún proyecto de construcción no es necesario actualizar ningún dato en el proyecto de manipulación creado a partir del anterior pero si es sumamente importante notificar algo del estilo "Todos los proyectos de manipulación creados a partir de éste proyecto de construcción tendrá valores de costos desactualizados"

Además en el menú "Opciones" deberíamos sacar la primera opción, ya que ésta no aplicaría más y dejar las demás tal cual está.

Con respecto a la precisión de los minutos se llegó a la conclusión de tener 2 o 3 cifras decimales de precisión dado que con una velocidad media de la flota de 40km/h llegamos a valores de minutos de 0.0 y 0.1. Aunque esta velocidad en la práctica llega a aproximadamente 22km/h, es aconsejable tener una mayor precisión.

Finalmente se dijo que es una tarea importante documentar las consideraciones tomadas, así como también todos los cambios realizados.

2) Módulo de Manipulación

La única modificación que se le ha hecho al módulo de manipulación es el cambio de nombres de las etiquetas de la interfaz gráfica. Más específicamente, se cambió "Centroides" por "Nodos Viales" y "Conexiones" por "Red Vial".

Se habló que debemos levantar 3 capas más de las que ya están (nodos viales y red vial), que serían necesarias para construir el grafo. Éstas son: la capa de Paradas, la capa de Accesos y la capa de Centroides. Se debe corroborar que no se puedan definir tramos de recorridos sobre la capa de Accesos.

3) Algoritmos

Dado los cambios realizados no podemos ejecutar los algoritmos. Lo que debemos hacer en primer lugar es fijarnos si realmente se encuentran físicamente dichos algoritmos. La lista donde aparecen en el módulo de manipulación es generada a través de un XML que debería contener la información de la ubicación de los mismos.

4) Estructura del grafo

Dada la estructura del grafo, llegamos a la conclusión de agregar tanto a los nodos como a los arcos un campo más que denominamos "tipo". Éste grafo va a ser el que vamos a utilizar para generar el archivo de texto que va a servir de entrada para nuestro simulador.

5) Recorridos y Frecuencias

Sobre este punto debemos fijarnos en el manual del igoR-tp. Posiblemente los recorridos y las frecuencias estén en carpetas diferentes a las que contienen los datos del caso (grafo y matriz).

6) Cambios y documentación

Debemos tratar de ver todos los cambios conceptuales realizados sobre el igoR-tp, "convencernos" de los mismos y tratar de plasmarlos escribiéndolos correctamente.

Tareas:

- Patricia y Gastón
  - Cambiar pequeños detalles para dar por finalizado el módulo de construcción
  - Finalizar implementación del módulo de manipulación de igoR-tp
  - Enviar el acta de la reunión del día de la fecha

Próxima reunión: Jueves 29 de Octubre a las 17:30.

Acta

Fecha: 29/10/2009

Participantes

- Antonio
- Patricia
- Gastón

Temas tratados:

1) IgoR-tp

   Se cierra esta etapa. Luego de que terminemos de probar y lleguemos a una versión estable, le tenemos que mandar los dos módulos a Antonio. Estimamos que sería para la semana que viene.

   En diciembre tienen un congreso y quieren presentar la herramienta igoR-tp así que estaría bueno que estuviera pronto para esa fecha.

   Hemos encontrado varios errores (que ya solucionamos) en la versión de mayo de igoR-tp. Antonio sugirió pasarle la lista de errores a Manuel porque a nosotros nos podría parecer que hay un error y sin embargo no serlo.

   En ninguna de las dos versiones que tenemos (versión del cd y mayo 09) de igoR-tp andan las notas del mapa y no sabemos cómo arreglarlo.

2) Simulador

Antonio nos sugirió ir documentando como generamos la primera versión de los requisitos. Sobre todo, incluir referencias ya que hay cosas particulares que interesa conocer la fuente. Le comentamos que algunos puntos surgieron en las reuniones con Marita, otros los sacamos de la información leída de internet y otros se nos ocurrieron a nosotros.

Tenemos que hacer un diagrama de estructura estática para definir las clases con sus atributos y las relaciones entre estas clases.

A continuación se incluye la primera versión de los requisitos con los comentarios que se realizaron sobre el mismo.

Pasajeros:

1. Los pasajeros van a tener asignados un conjunto de líneas de ómnibus y se determinará cuál tomar, por ejemplo, de acuerdo al destino que va, cuál le conviene más.
   Esto seguro lo tenemos que modelar. Capaz el conjunto de líneas no esté determinado a priori. Lo calculamos nosotros y según a donde va, cuál le sirve. En principio, al pasajero le serviría un conjunto de líneas.

2. Un pasajero está en la parada y vienen dos buses que le sirven. ¿Cuál tomar?
   - Azar
   - Tiempo de llegada a destino
   - Geográficamente le quede más cerca al pasajero
   - El bus que tenga menos gente aglomerada para subir
   - etc

   La decisión a tomar se modelará como una estrategia asociada al pasajero, o sea, el pasajero le pregunta a la estrategia que hacer.

3. Un pasajero está esperando el ómnibus y este no para o va muy lleno:
   - Espera otro ómnibus
   - Se va

   En un principio discutimos que contemplar que el pasajero se fuera no sería un buen criterio. Por ejemplo, hay líneas que tienen una frecuencia alta (ej.: dos minutos) y pasan 3 ómnibus llenos y al 4 el pasajero se sube. Pero el pasajero se podría tomar un taxi por lo que se iría de la parada.

   Concluimos que "Se va" sería parte de la estrategia.

   Información como cuanto tiempo espera, cuantos buses deja pasar, etc, se codifica con la estrategia.

   Debemos hacerlo lo más genérico posible.

4. Comportamiento de los pasajeros como entidades individuales

Si. Debemos modelar esto.

5. Tiempo máximo de espera del pasajero

Está relacionado con el punto 3.

6. Existencia de pasajeros heterogéneos:
    - Jóvenes, ancianos, discapacitados, etc

Los pasajeros tendrían atributos y sus tiempos dependen de los atributos. Más precisamente, cada objeto pasajero tiene un conjunto de atributos los cuales tienen nombre y valor.

7. ¿Qué pasa si hay mucha congestión de pasajeros en una parada?

Si contemplamos este punto, deberíamos setearle un atributo a las paradas.

Una posibilidad es si hay más de n personas se va a la parada más cercana.

Asumimos que la parada tiene un dato que es la capacidad. No necesariamente se lo setearemos desde igoR-tp.

8. Los pasajeros esperan el ómnibus en:
    - La parada misma (el punto)
    - Un buffer de la parada. En este caso: ¿Cómo se distribuyen los pasajeros?

En la parada misma (el punto).

9. ¿Interesa diferenciar entre aquellos pasajeros que van parados y los que van sentados?

Sí.

Ómnibus:

1. Modelo unimodal: sólo se modelarán los buses y no otro tipo de vehículos.

Si. La limitante de modelar otro tipo de vehículos es que todo el transporte público transitaría por las calles. Si quisiéramos agregar el metro, que no transita por las calles, no podríamos. En este caso sólo se toma en cuenta una red sino habría que modelar varios modos.

2. Tiempos entre paradas determinados por distribución, definida por factores como:
    - Calidad de la calle
    - Velocidad permitida
    - etc.

No consideraremos las opciones presentadas ya que el tiempo queda fijado en igoR-tp según la distancia y la velocidad media de la flota. Lo que si podríamos modelar es la velocidad del bus según si la calle está despejada o no. Se definiría una función F(v,b) donde v es la velocidad media de la flota y b es la cantidad de ómnibus que hay entre punto y punto. Entonces F(v,b) = tiempo + g(b) , donde "tiempo" corresponde al costo de ir de un nodo vial a otro y g(b) es un valor en función de los buses.

3. Modelar trasbordo.
    - Cómo se definen los trasbordos?
    - Qué algoritmos hay?
    - *Comentario: En el caso de Rivera la ciudad es como un polígono, todas las líneas van al centro. En la actualidad si se quiere ir de un barrio a otro, el pasajero tendría que hacer trasbordo y esto no es posible*

   El trasbordo lo modelaríamos como una concatenación de estrategias.

   También se aclaró lo siguiente: cuando el pasajero llega al centroide destino, el pasajero se destruye.

4. Modelar que el bus cumpla con sus horarios
    - Si va adelantado debe ir más lento o esperar en alguna parada
    - Si va atrasado, no parar en algunas paradas.

   Con esta pregunta surgió que se podría hacer algo que centralice el despacho de los ómnibus. El horario de salida y el recorrido va a tener que estar centralizado en algún lado.

   En igoR-tp tenemos recorridos y frecuencias. Las frecuencias indican cada cuanto tiempo sale el bus pero no a que hora.

   Podríamos tener dos formas para modelar esto: una basada en frecuencia y otra basada en tabla de horarios. La primera es más sencilla pero la segunda ofrece mayor exactitud. Sería bueno que el simulador permitiera las dos cosas.

   Con respecto al punto 4, el simulador debería saber si un ómnibus va adelantado o atrasado. Se resolvió modelar este punto con operaciones genéricas y que tengan un comportamiento por defecto. Que los buses tengan una operación que permita modificar la velocidad y que indique si el bus para o no en una determinada parada. También le podríamos agregar un tiempo extra de permanencia en la parada, que por defecto este tiempo tendría el valor 0.

5. No manejar asignación de choferes a buses. Pero tener en cuenta:
    - Choferes con distintas características (que afectan al pasajero)
    - Cada cierto tiempo un determinado ómnibus va a mantenimiento
    - Cambio de choferes
    - etc

   En un día típico se considera que el ómnibus no va a mantenimiento. Modelar esto excedería el horizonte horario que queremos modelar.

   Se podría agregar un tiempo por razones operativas para prever, el cuál va a ser chequeado por el simulador. De todos modos, esto se definirá en reuniones posteriores.

6. Permitir subir con sillas de ruedas o no

   Se sugirió tener tipos de ómnibus compatibles con tales pasajeros. Por este motivo, se definirán tipos de buses, tipos de pasajeros y la compatibilidad entre éstos. Este requisito es de prioridad baja.

7. Contemplar el hecho de que el chofer cobre, suban con boleteras o pases, etc.
   Sí modelaremos esto, y le setearemos un valor por defecto a estos atributos.

8. Cantidad de puertas del ómnibus. Descienden por la puerta de atrás y por la puerta de adelante o solo por la puerta de atrás?
   Se deberá tener los siguientes atributos para el ómnibus: cantidad de puertas por las que pueden bajar y cantidad de puertas por las que pueden subir.

9. ¿Cuántos ómnibus pueden atracar en una misma parada?
   - ¿Cómo actuar cuando llega un ómnibus y los lugares para atracar están todos ocupados?
     - Ignora la parada en caso de que no haya nadie para bajar, sino espera hasta conseguir lugar para atracar
     - Espera siempre a que haya un lugar disponible

   Este punto no tiene sentido ya que no manejamos distancia, sólo tiempo.

10. ¿Influye la posición donde atraca en el tiempo de subida de los pasajeros?
    No hay información para eso por lo que no se modelará.

11. ¿Se va a considerar el clima? Ej: Si hay lluvia torrencial, la velocidad de los ómnibus disminuye
    No se modelará este punto.

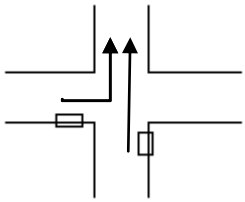
Líneas:

1. Si las paradas se modelan en un cruce, ¿esa parada puede ser accedida por cualquier ómnibus que venga en cualquiera de las dos calles que se intersectan?
   Si modelaremos eso.

2. ¿Semáforos?
   - De no haber:
     - ¿Se toma la política de dejar pasar al de la derecha?
     - ¿Hay avenidas que tienen mayor preferencia que cualquier calle que la intersecte?

   Esto no lo vamos a modelar.

   A raíz de esta pregunta, Antonio nos comentó un problema que surge pero que queda afuera del alcance del proyecto. Las dos paradas que se muestran en el dibujo, nosotros la representamos como un mismo punto al crearlas en el módulo de construcción de igoR-tp. Entonces el pasajero puede tomar la línea de la izquierda o la línea sur para ir a su destino. Esto no es muy realista ya que la persona se encontraría en una parada o en otra debido a la localización geográfica de las mismas.



3. ¿Cuántos carriles hay por calle?
4. El espacio que tienen los ómnibus para atracar en las paradas está separado de la calle misma
5. ¿Considerar cambios de tramos de líneas dinámicamente? Ej: hubo un accidente en una calle que es parte de una línea, corte de calles por espectáculos

No modelaremos ni el punto 3 ni el 4 ni el 5.

Resultados:

1. ¿Que se espera obtener como resultados?
   - Tiempo de espera de los pasajeros
     Si
   - Congestionamiento de las paradas
     No, mejor sería la cantidad media de personas esperando en las paradas (largo medio de la cola).
   - Cantidad de pasajeros por día
     No, ya que este valor es la suma de la matriz demanda.
   - Cantidad de pasajeros por viaje
     Más interesante es tener la ocupación media del bus.
   - Líneas con retraso
     Si se cuenta con la tabla de horarios, esto se podría calcular.

     Incluso se puede calcular el retraso total.
   - Cantidad de ómnibus que ignoran determinadas paradas
     Si es que consideramos eso, se podría calcular.
   - etc

Visualización:

1. La vista 3D supera el alcance del proyecto. Se propone realizar algún gráfico "simple". Por ejemplo que se pueda visualizar entre otras cosas trasbordos y consultas.

Visualización en 2D es suficiente.

Tareas:

- Patricia y Gastón
    - Seguir probando igoR-tp y luego que esté estable mandárselo a Antonio.
    - Hacer la segunda versión de los requerimientos del simulador.
    - Realizar la estructura de entidad estática del simulador.
    - Enviar el acta de la reunión del día de la fecha

Próxima reunión: Jueves 12 de Noviembre a las 17:30.

Acta

Fecha: 05/11/2009

Participantes

- Antonio
- Patricia
- Gastón

Temas tratados:

1) Requisitos del Simulador

Se le preguntó a Antonio cómo teníamos que especificar los requisitos, y nos sugirió que lo hagamos igual a como lo hace el libro del curso de Simulación o sino como se hizo en el caso de estudio que se propuso este año para la tarea de dicho curso, pero sería mejor que siguiéramos como modelo el libro.

2) Interfaz de datos de entrada

Algunos datos de entrada del simulador los vamos a obtener de igoR-tp pero otros le vamos a tener que pedir al usuario que los ingrese. Para pasarle estos últimos utilizaremos un archivo y en caso de que sea muy complicado el ingreso de éstos, se creará una interfaz para resolverlo.

3) Modelo de estructura estática

Discutimos la primera versión del modelo de estructura estática.

Definimos recorrido y línea. Un recorrido es una secuencia de calles de una línea. Una línea tiene una frecuencia y pude contener uno o dos recorridos, o sea, un recorrido de ida y otro de vuelta o un recorrido circular.

En el modelo tenemos que cambiar el concepto línea por recorrido y agregar la clase línea que se va a especializar en "Ida y Vuelta" o en "Circular", donde cada una de estas va a estar asociada a los recorridos asignándole roles a las asociaciones. En caso de la clase "Ida y Vuelta" va a tener asociado un recorrido de ida y otro de vuelta y la clase "Circular" va a tener asociado un solo recorrido.

El pasajero va a tener asociado un recorrido.

El ómnibus va a estar asociado a un recorrido.

Se discutió el concepto de trasbordo y como se definen las estrategias. En principio, sacamos el concepto de trasbordo y dejamos modelado las estrategias como está actualmente en el modelo. Antonio nos va a mandar un artículo que habla sobre las estrategias.

Se discutió si el ómnibus debería tener estrategias (por ejemplo, para ver si para o no en una determinada parada o si espera un tiempo extra en la parada) y se determinó que le tenemos que agregar "Reglas operativas" que son un conjunto de operaciones que siempre se invocan para ver que hace el bus.

El grafo tiene que estar modelado en el diagrama de estructura estática.

4) Salida gráfica

Como salida gráfica se quiere ver el movimiento de los ómnibus en el grafo. Éste se tiene que mostrar en pantalla por lo que se necesitan las coordenadas del los puntos (paradas, nodos viales, centroides) para poder ubicarlos en la misma. Por este motivo se tiene que modificar el grafo creado en el módulo de manipulación para incluir estos datos.

Si la ciudad es muy grande, se puede tomar alguna opción para que se vea cierta parte de la misma. Se comentó la opción de mostrar un cuadrante de una cantidad de metros reales, los cuales pueden ser paramétrico para darle mayor precisión y una mejor visualización de la simulación al usuario. Asimismo dar la chance de que se pueda cambiar el cuadrante durante la simulación. Antonio sugirió hacer una prueba de metros-pixels.

Tareas:

- Patricia y Gastón
    - Agregar al grafo generado en el módulo de manipulación las coordenadas de los puntos.
    - Hacer la segunda versión de los requerimientos del simulador.
    - Modificar la estructura de entidad estática del simulador.
    - Enviar el acta de la reunión del día de la fecha
- Antonio
    - Mandarnos el artículo que habla sobre estrategias.

Próxima reunión: Jueves 12 de Noviembre a las 17:30.

Acta

Fecha: 18/11/2009

Participantes

- Antonio
- Patricia
- Gastón

Temas tratados:

1) Comentarios sobre los requerimientos

Antonio realizó diversos comentarios sobre la primera versión de los requerimientos, los cuales nos enviará vía mail. También acotó como detalle que los ómnibus en realidad circulan sobre las aristas y no sobre los nodos viales. Nuestra intensión fue decir que los ómnibus circulan de nodo vial en nodo vial por lo que corregiremos la redacción para que sea más entendible.

2) Estrategias de los pasajeros

Este punto junto a las reglas operativas de los ómnibus son las partes cruciales, esenciales y de mayor importancia del simulador. Los pasajeros tendrán asociada una estrategia genérica la cual será instanciada mediante una específica. Éstas abarcan el comportamiento del pasajero desde que nace en un determinado centroide hasta que muere en su centroide destino. En el artículo que nos mandó Antonio (Optimal strategies: A new assignment model for transit networks) se describe un ejemplo claro de lo que sería una estrategia específica que un pasajero puede tomar. También hay un algoritmo que dice como hace un pasajero para ir de una parada origen a una parada destino, teniendo un conjunto de líneas como opciones para tomar. El mismo asume que los pasajeros son seres racionales y que tratan de minimizar los tiempos de viaje y espera.

Definir una estrategia general que contemple todo lo que habíamos hablado en las reuniones anteriores, es algo muy complejo de realizar por lo que la idea sería, utilizando el patrón Strategy ir definiendo y refinando estrategias más específicas. Se habló también de un posible modelado de estrategias que sería que cada una tuviera un conjunto de "Etapas" (mas de una si es que se trata de un trasbordo) y cada una de estas etapas tendría como dato: la parada origen, un conjunto de líneas y para cada una de ellas, cual es las paradas destino si es que tomara dicha línea.

3) Velocidad de los ómnibus

Se decidió que la velocidad de los ómnibus sea fija y no dependa de la cantidad de ómnibus que hay entre nodo y nodo debido a que sino, el pasajero no puede decidir a priori que línea tomar.

4) Reglas Operativas

Debemos pensarlo como las estrategias de los pasajeros, pero con la diferencia que cuando el ómnibus está en una parada, sólo tiene una opción para seguir (que la marca la regla operativa) mientras que los pasajeros en una determinada parada tienen varias opciones posibles a seguir. Las reglas operativas van a indicar si un ómnibus para o no en una parada y en caso de parar, el tiempo extra de permanencia en la parada. Si el ómnibus va adelantado, esperaría un tiempo extra en la parada y si va atrasado, no pararía en algunas paradas. Este caso se puede contemplar cuando contamos con la tabla de horarios. Sino, no tenemos manera de saber si un ómnibus va adelantado o atrasado.

Se mencionó también que no se van a manejar expresos y que los ómnibus deberían seguir siempre el mismo recorrido o cambiar de recorrido para que sea más realista.

5) Fechas

Se habló de las cosas que nos restan hacer: Documentación de igoR-tp, ayudas, posibles agregados al módulo de manipulación y errores que se puedan encontrar.

También se hizo referencia a que las definiciones que estamos realizando ahora tienen que estar muy bien documentadas ya que son muy importantes, incluso más importantes que la propia salida gráfica.

Tareas:

- Patricia y Gastón
  - Modificar los requerimientos del simulador
  - Modificar el modelo de estructura estática del simulador
- Antonio
  - Mandarnos los comentarios realizados sobre los requerimientos del simulador
  - Probar igoR-tp

Próxima reunión: Jueves 26 de Noviembre a las 17:30.

Acta

Fecha: 26/11/2009

Participantes

- Antonio
- Patricia
- Gastón

Temas tratados:

1) Comentarios sobre los requerimientos

Se discutió como modelar las reglas operativas. Si debíamos modelar el comportamiento del ómnibus cuando va atrasado o adelantado en una sola regla operativa específica o si debíamos modelar el comportamiento cuando el ómnibus va atrasado en una regla operativa específica y cuando el ómnibus va adelantado en otra regla operativa específica. Antonio dijo que estos dos comportamientos son disjuntos por lo que deberíamos modelar estos comportamientos por separado. En caso de que el ómnibus no tenga asociada una regla operativa, actúa normal. O sea, para en una parada si hay alguien para bajarse y/o alguien para subir en esa parada.

El usuario debe elegir por parámetro de entrada que reglas operativas seguir dentro de las reglas operativas existentes.

También deberemos mostrar cuáles son las estrategias del pasajero para que el usuario elija cuál seguir. En este caso, sólo vamos a tener la estrategia proporcionada por el algoritmo.

2) Diseño

Quedó finalizada la etapa de análisis y ahora comenzamos con el diseño del simulador. Tenemos que hacer un diagrama de clases. Antonio dijo que no vale la pena hacer los casos de uso y que realizar o no el diagrama de colaboración, queda a nuestra evaluación. Debemos hacer este último sólo en el caso que nos sirva.

Tareas:

- Patricia y Gastón
  - Enviar el modelo de dominio
  - Diseñar el simulador. Hacer diagrama de clases.
- Antonio
  - Probar igoR-tp

Próxima reunión: La semana que viene a coordinar por mail.

Acta

Fecha: 04/12/2009

Participantes:

- Marita
- Antonio
- Patricia
- Gastón

Temas tratados:

1) igoR-tp

Antonio hizo pruebas en el módulo de construcción y anduvo bien. No probó el módulo de manipulación

2) Preguntas realizadas por mail

¿En qué inciden el conjunto atributos de los ómnibus y de los pasajeros?

Se debería chequear que para cada atributo que tenga el pasajero, el ómnibus al cual subirá, contenga un atributo con el mismo nombre y mismo valor. De lo contrario no lo tomará.

¿Los arribos de los pasajeros a las paradas están determinados por una distribución o esos datos son dados?

La generación de pasajeros debe estar centralizada en algún lado. Los arribos pueden darse de 3 formas distintas: Primero, a través del tiempo fijo entre 2 arribos. Segundo, por intermedio de una distribución. Y por último, mediante una lista con la hora exacta de cada arribo. Nosotros deberíamos tratar de contemplar la mayor cantidad de casos posibles.

3) Diseño

En cuanto al diseño se cuestionó si la subida y/o bajada de pasajeros era de forma individual o a través de una lista de pasajeros. Se decidió que sobre este tema nosotros propongamos nuestra propia hipótesis, pero si o si debemos controlar que no suban y bajen pasajeros al mismo tiempo por una misma puerta. Se acotó que sería muy útil realizar diagramas de comunicación para operaciones complejas o claves.

4) Histogramas y Colas

Tanto las colas de espera como los histogramas no van en el diseño. Las colas son cosas que están implícitas en el diseño, mientras que los histogramas son cosas auxiliares. Debemos tener pensado desde antes de comenzar a implementar, la recolección de datos.

Tareas:

- Patricia y Gastón
  - Comenzar la implementación del simulador.
  - Enviar el acta de la reunión del día de la fecha.

Próxima reunión: Dentro de dos semanas a coordinar por mail.

Acta

Fecha: 17/09/2009

Participantes:

- Antonio
- Patricia
- Gastón

Temas tratados:

1) Interfaz gráfica

Se recomendó cambiar el nombre del menú que contiene las funcionalidades de zonas, conexiones y nodos viales (actualmente "Elementos Viales") por "Elementos de la Red"

2) Costo centroide-parada

Se determino que el costo asociado a las conexiones entre los centroides y las paradas es el "tiempo de acceso". Este tiempo está determinado por la velocidad de caminata de los usuarios y por la distancia entre ambos nodos. A su vez esta distancia puede ser considerada de dos formas distintas, distancia Euclidea o distancia de Manhattan. Tanto la forma de considerar la distancia como la velocidad de caminata de las personas deben ser parámetros que tienen que estar reflejados en la interfaz gráfica para que el usuario pueda ingresar y seleccionar.

Cabe mencionar que debemos controlar que al ingresar la conexiones entre centroides y paradas, estos deben estar dentro de una misma zona. De lo contrario desplegar el mensaje de error correspondiente.

3) Grafo

Debemos tener en cuenta que el grafo está formado por nodos y arcos. Cada nodo puede ser de tipo "vial", "parada", "centroide" o alguna combinación de ellos. A su vez los arcos pueden ser de tipo "viaje" o "caminata". Es importante tener esto en cuenta a la hora de generar el archivo txt que sirve de entrada para los algoritmos y en especial para el simulador que construiremos posteriormente. Respecto a los algoritmos existentes, es responsabilidad del usuario implementarlos nuevamente para que soporte el nuevo grafo.

Por ultimo cabe destacar que no se tendrá mas en cuenta las conexiones entre centroides, por lo que debemos sacar esa opción y determinar en que parte de la aplicación repercute.

4) Documentación

Al cerrar esta modificación de igoR-tp debemos empezar con la documentación de la herramienta, focalizándonos en la parte técnica de la misma. Dejaremos el manual de usuario para mas adelante.

Tareas:

- Patricia y Gastón
    - Implementar la conexión entre centroides y paradas
    - Eliminar las conexiones entre centroides.
    - Implementar el grafo de igoR-tp.
    - Enviar el acta de la reunión del día de la fecha

Próxima reunión: Jueves 1 de Octubre a las 17:30.

Acta

Fecha: 25/02/2010

Participantes:

- Antonio
- Gastón

Temas tratados:

1. Documentación igoR-tp

Se comenzó con la documentación de igoR-tp. La misma debe focalizase más que nada en describir los cambios realizados en la aplicación y no entrar muy en detalle en lo que es igoR-tp. No preocuparse por describir conceptos tales como "centroide", "nodos viales", "recorridos", "líneas" en esta parte ya que deberían ser explicados previamente en una introducción al tema.

Es importante que el documento se redacte para que pueda ser entendido por alguien que tenga un conocimiento similar al nuestro (tener en cuenta que no necesariamente tiene porque saber sobre transporte público urbano). Una vez realizada la primera versión del documento, enviársela a Antonio y Marita.

2. Simulador

Comenzamos con la implementación del simulador siguiendo el diagrama de clases del modelo de diseño realizado por nosotros. Más allá de implementar todos los componentes, todas las clases y sus asociaciones, preferimos implementar las funcionalidades de forma incremental. En este momento no se están tomando los datos del igoR-tp sino que se crean (mediante un juego de datos estático) nodos viales (paradas y cruces), conexiones, se modela que los ómnibus sigan un determinado recorrido, que los pasajeros arriben a una determinada parada, que éstos se suban al ómnibus que le sirve y que se bajen en la parada que corresponda. Está implementada la parte gráfica que muestra esto, teniendo como mayor inconveniente mostrar el movimiento del ómnibus de un nodo vial a otro.

Antonio mencionó que es importantísimo seguir desde el principio el diagrama de clases del modelo de diseño que hicimos. También se habló que lo más importante de todo no es tener una excelente salida gráfica del simulador (EOSimulator es "limitado" en cuanto a las cosas que se pueden hacer), sino que lo más importante es tener un diseño de la solución bien claro, bien especificado y bien implementado.

Algo importante que debe contar la interfaz es la posibilidad de apagar y prender líneas de ómnibus para poder ver y comparar con detalle las que realmente se desean. Debemos documentar más específicamente el alcance de la parte gráfica del simulador.

Antonio nos va a mandar ejemplos de códigos de simuladores que nos sirva de ayuda a la hora de implementar la interfaz. Sebastián tiene ejemplos de años anteriores.

Tareas:

- Patricia y Gastón
    - Seguir con la documentación de igoR-tp.
    - Seguir con la implementación del simulador.
    - Enviar el acta de la reunión del día de la fecha.
- Antonio
    - Pedir a Sebastián ejemplos de simuladores hechos con EOSimulator que tengan salida gráfica.

Próxima reunión: A coordinar por mail.

Acta

Fecha: 25/03/2010

Participantes:

- Marita
- Antonio
- Patricia
- Gastón

Temas tratados:

1. Tiempos

Marita y Antonio avisarán que no vamos a terminar ahora sino que estimamos finalizar dentro de un mes. En éste momento estamos implementando funcionalidades del simulador y documentando. Nos sugirieron armar una lista de las actividades que faltan de cara a la finalización del proyecto.

2. Simulador

- Subida y bajada de pasajeros: ¿Cómo elije el pasajero por cuál puerta bajar y por cuál subir?
  Se recomendó resolverlo utilizando sentido común. No contamos con elementos que especifiquen distancias entre pasajeros y puertas de ómnibus. Se resolvió que haya una única puerta para subir (la cual también se permitirá bajar) y varias puertas para bajar. Se resolvió que la puerta de bajada para los pasajeros esté determinada por una probabilidad uniforme.

- Una vez que un pasajero elije un ómnibus para tomar, no cambiará de opción por más que mientras espera para subir venga otro que le sirva también. Cualquier otra alternativa podría ser implementado en un futuro como otra estrategia.
- Alcance de los gráficos: Se descartó dividir la red vial en cuadrículas y hacer zoom en cada cuadrícula ya que no se trata de un proyecto focalizado en interfaz gráfica. Sí es necesario que aparezcan los ómnibus con los números de líneas y con la cantidad de pasajeros que lleva. Que muestre los parámetros importantes para el modelo de simulación y que al final muestre los histogramas. Que el usuario pueda elegir que visualizar. Por ejemplo, que elija mostrar o no tiempos de espera en paradas o cuantas personas hay en cada parada. Se puede mostrar en el mapa una lista de referencias a estas funcionalidades. Además la funcionalidad "Activar y Desactivar" líneas de buses es aconsejable que sea similar al ejemplo de Rivera que envió Antonio por mail.

3. Informe

En el informe y en la presentación hacer hincapié en el simulador.

En la introducción debemos poner los conocimientos que usamos para hacer el proyecto y hablar del trabajo que hicimos.

Tareas:

- Patricia y Gastón
  - Seguir con la documentación de igoR-tp, ayuda y documentación técnica.
  - Seguir con la implementación del simulador.
  - Enviar el acta de la reunión del día de la fecha.
- Antonio
  - Avisar que no terminaremos ahora.

Próxima reunión: Jueves 8 a las 8am en la cual hablaremos de las pruebas.

Acta

Fecha: 08/04/2010

Participantes:

- Antonio
- Patricia
- Gastón

Temas tratados:

1. Informe

Se debe agregar a la documentación una sección de "especificación de Eventos", más específicamente mostrar un pseudocódigo de cada evento y también deberíamos tener un diagrama de actividad de las entidades. Si los pseudocódigos no abarcan muchas páginas ponerlo dentro del capítulo "Simulación".
La especificación de requerimientos del simulador debe explicar los requerimientos "finales" a los que se llegaron, para explicar como hicimos para llegar a esos requerimientos debemos referir a las actas donde se habló sobre éste tema. Tenemos que tener las actas en anexos.
Actualmente en el desarrollo del informe contamos con un capítulo de "Introducción", uno de "igoR-tp", uno de "Simulador" y otro de "Caso de Prueba", cada uno con subcapítulos.
Debemos agregar un nuevo capítulo denominado "Marco Teórico" que hable de todas las herramientas de análisis (Modelos de asignación, etc.). Debe ser más específico que la Introducción, ésta última debe ser una introducción a la temática, describir los aportes del proyecto, etc. El marco teórico es más específico. En estos momentos estos dos capítulos los tenemos contemplados dentro de la introducción por lo que debemos separarlos.

El estudio de las librerías de simulación debe estar en un anexo.

2. Caso de Prueba

Para llevarlo a cabo necesitaríamos datos de la red vial y de las paradas. Con respecto a la red vial deberíamos dibujar cada zona (84 zonas) de la división zonal. Seguramente éste dato se lo tenga como un mapa impreso. Sobre las paradas, se cuenta con un shape de las paradas, deberíamos cargarlo e ir seleccionando como parada a aquel nodo vial que coincida aproximadamente con paradas del shape cargado.
Las caminatas de centroides a paradas se deben considerar todas y las caminatas entre paradas no las consideraremos.

En cuanto a las Líneas de ómnibus, se cuenta con los shapes (13 capas).
La información de los ómnibus que contamos es la frecuencia de los mismos. Debido a esto es que para éste caso no se va a considerar las reglas operativas por no contar con una tabla de horarios. De todas maneras las reglas operativas no se deben quitar del diseño del simulador ya que el mismo está construido para un uso lo más genérico posible.

Para el caso de los pasajeros y sus arribos vamos a contar con una matriz origen-destino con las tazas de arribos.

Se cuenta con cierta información respecto a los tiempos de viajes y tiempos de espera que deberían dar aproximadamente, por lo que una vez realizada la simulación y obtenidos los resultados, se podrían validar.

Tareas:

- Patricia y Gastón
    - Seguir con la documentación del informe.
    - Seguir con la implementación del simulador.
    - Enviar lo que vamos realizando del informe, al menos como esta organizado (títulos y subtítulos)
    - Enviar el acta de la reunión del día de la fecha.

Próxima reunión: Jueves 22 a las 8am.

Acta

Fecha: 22/04/2010

Participantes:

- Antonio
- Patricia
- Gastón

Temas tratados:

1. Datos para el Caso de Prueba

Le comentamos a Antonio el inconveniente que tuvimos al digitalizar con el módulo de construcción de igoR-tp las zonas de Rivera ya que esa capa no tiene la misma proyección que la capa de calles. Nos explicó que la matriz OD es de 84x84 y se corresponde con las 84 zonas, por eso se necesitan esas zonas que se encuentran en el archivo "zonas_rivera.dwg". La solución más apropiada sería reproyectar la capa de zonas con las herramientas adecuadas. Ya que esa solución es muy compleja, Antonio nos pasó un programa para que visualicemos el archivo "rivera_zonas.dwg" y podamos digitalizar las zonas con igoR-tp.

La velocidad media del bus la debemos sacar de la tesis de Maestría de Antonio. Se resolvió que el tipo de distancia será Manhattan ya que Rivera tiene manzanas cuadradas. La velocidad media del peatón la tenemos que setear nosotros.

2. Simulador

<u>Recorridos:</u>

Se discutieron los conceptos de "recorrido de ida y vuelta" y "recorrido circular", más precisamente como deberíamos modelarlos ya que surgieron dudas.

Recorrido de ida y vuelta: Está modelado con dos listas (una para el recorrido de ida y otra para el recorrido de vuelta) donde cada una está compuesta por una secuencia de nodos viales. Cuando el bus visita al último nodo del recorrido de ida, luego pasa al primer nodo del recorrido de vuelta. El tiempo que demora en cambiar de recorrido (de ida a vuelta o viceversa), debe ser ingresado por parámetro y será un atributo de las líneas.

Despacho de ómnibus:

Si un ómnibus tiene que pasar a cierta hora pero no pasa, entonces sacamos otro del despacho. Se asume que tenemos la cantidad de ómnibus que necesitamos. Es muy importante llevar la cuenta de cuantos está circulando a la vez.

Hay ciertas variabilidades que se dan en los tiempos de viaje, o sea no siempre demora lo mismo. En este momento, los ómnibus demoran lo mismo en efectuar sus recorridos porque la velocidad del bus es constante. Lo más acorde con la realidad, es que ese tiempo sea producto del resultado de una distribución. Se discutió cuál distribución usar si una exponencial negativa o una normal, en ambos casos con media igual a la velocidad media del bus. Utilizaremos la distribución normal ya que se adecúa mejor a este caso. Con esta distribución contemplamos los efectos aleatorios que tienen sobre el tráfico de los ómnibus.

3. Informe

Los puntos que se encuentran en el informe 1, 2, 3,… son capítulos y Antonio nos sugirió resaltar más los títulos.

Debemos agregar al informe "Marco Conceptual" en donde definiremos conceptos y metodologías utilizadas en el proyecto de forma genérica, o sea sin mencionar al proyecto en sí. A modo de ejemplo, algunos conceptos son: grafo, SIG, simulación a eventos discretos, las capas, etc.

Le preguntamos a Antonio si debíamos dejar la información de Contexto del trabajo que sacamos de la propuesta formal y nos dijo que sí. Y también le consultamos sobre un punto de la propuesta formal que decía que el proyecto abarcaba estudiar la congestión del tráfico debido a la presencia de buses y nos contestó que está bien que lo hayamos quitado.

Debemos poner secciones dentro de los títulos (bajo el capítulo 2 poner secciones 2.1, 2.2, etc).

Sacamos la sección Análisis ya que este fue muy detallado y lo podemos tomar como parte del diseño. Tenemos que dividir el diseño en estático (DCD) y dinámico (diagramas de actividades de simulación). Debemos poner en el marco conceptual que utilizamos para realizar el diseño, y porque utilizamos esas herramientas.

Sólo se debe aclarar del DCD las cosas que no son obvias.

Poner el seudocódigo como está en el libro. En el seudocódigo podemos hacer referencia a los diagramas estáticos o dinámicos. No es necesario indicar la referencia ya que se asume de esa manera.

En la ventanita de presentación (splash) hay un texto que dice "Proyecto de Grado 2008", ¿Cómo deberíamos modificarlo? En el módulo principal de igoR-tp, en el menú Ayuda/ Acerca del programa están los datos del proyecto anterior, ¿cómo deberíamos modificarlo? Antonio nos confirma estas dos preguntas ya que hay una manera para numerar esta nueva versión e incluir los créditos.

La documentación técnica para desarrolladores va en el anexo. Tenemos que seguir el estilo del documento de la versión anterior, plasmando los cambios que realizamos.

También se deberá realizar la documentación técnica del simulador.

Debemos agregar un anexo: "Gestión del proyecto" donde pondremos el cronograma y las actas.

Debemos eliminar el capítulo Anexo que estaba pensado para contener todos los anexos, ya que cada anexo es un capítulo. Se acostumbra numerarlos A1, A2,…. O A, B, C,….

Debemos cambiar el término "Librerías" por "Bibliotecas" del anexo.

También debemos agregar un anexo sobre el estudio de los simuladores de transporte público el cuál fue realizado para detectar los requerimientos del simulador.

Las referencias a otras lecturas las podemos numerar secuencialmente o por capítulo.

Un error frecuente es "En la siguiente figura…." En vez de esto debemos nombrar a lo que se esté haciendo referencia. En este caso nombrar la figura.

Tareas:

- Patricia y Gastón
    - Seguir con la documentación del informe.
    - Seguir con la implementación del simulador.
    - Enviar informe.
    - Enviar el acta de la reunión del día de la fecha.
- Antonio
    - Confirmar como numerar la nueva versión incluyendo los créditos.

Próxima reunión: Jueves 6 a las 8am.

Acta

Fecha: 06/05/2010

Participantes:

- Antonio
- Patricia
- Gastón

Temas tratados:

1. Simulador

Se comentó que la velocidad media del peatón se seteó en 4km/h Y la del ómnibus en 13,6km/h. Antonio nos comentó que estaban bien esos valores. También se preguntó sobre como deberíamos manejar los tiempos en la simulación. Antonio nos dijo que había un parámetro que deberíamos manejar que mide la relación existente entre el tiempo real y el tiempo de simulación. Además, durante la simulación deberíamos poder aumentar y disminuir la velocidad de la misma.

El tiempo que demora en ir un ómnibus de un nodo a otro está determinado por una distribución normal. La media de la misma debe ser el tiempo que se toma como dato del igoR-tp y no debe ser la velocidad media del ómnibus como estaba escrito en el acta del día 22/04/2010 (corregir éste detalle). Además la desviación estándar debería ser de 1/5 (fijarse en el libro de Simulación).

2. Datos para el Caso de Prueba

Al momento de preparar los datos para el caso de prueba de Rivera nos encontramos con ciertos inconvenientes:

- No hay paradas en algunas zonas: Las zonas no cubren toda la ciudad y eso causa problemas ya que quedan paradas afuera de zonas. Le mostramos a Antonio un ejemplo en el que sucedía esto (zona 15). Dijo que éste problema se da porque los datos vienen de modelos diferentes. El problema debería resolverse en el módulo de construcción. Se llegó a la conclusión de que la solución es agregar más paradas en aquellas zonas en las que faltan. La mejor forma de realizarlo es mientras se construyen los recorridos ya que de ésta manera nos aseguramos que todos los recorridos contengan al menos una parada en cada zona.
- Recorridos y capa de calles: Solo se pudieron construir 3 recorridos debido a inconsistencias encontradas en la capa de calles. Le mostramos a Antonio un ejemplo en el que se da esto y nos dijo que no cree que se dé en la versión de Mayo. Las inconsistencias que encontramos se dan en los atributos "from" y "to" de la capa de calles. Nos recomendó que le preguntemos a Manuel sobre éste problema. Éste punto es considerado con máxima prioridad.
- Recorridos circulares: Antonio nos mencionó que debemos adaptar los datos reales al modelo nuevo. Esto significa, tratar de arreglar el recorrido "a ojo" de forma de que quede bien.
- No existe calle para el recorrido: La solución para éste problema es terminar el recorrido antes.

Es importante documentar todas las dudas, las decisiones tomadas y los problemas con los que nos encontramos. Describir todas las alternativas y soluciones tomadas.

También es importante tener en cuenta las limitaciones de los datos, la representación del modelo y tiempos del proyecto. Son causas de las decisiones tomadas.

Tareas:

- Patricia y Gastón
    - Seguir con la documentación del informe.
    - Seguir con la implementación del simulador.
    - Arreglar los problemas relacionados con los datos.
    - Enviar el acta de la reunión del día de la fecha.

Próxima reunión: Jueves 20 a las 8am.

Acta

Fecha: 15/05/2010

Participantes:

- Antonio
- Marita
- Patricia
- Gastón

Temas tratados:

1. Simulador y Documentación

La unidad en la que se encuentran las tasas de arribo en la matriz de demanda es en 1/minutos.

En la salida gráfica del simulador, donde se encuentran las referencias de las líneas, poner "Teclas para mostrar u ocultar".

Se mostraron los reportes y se explicó que se le pueden agregar más información muy fácilmente. Se acordó que está bien que se guarden los reportes en la carpeta del simulador y que cuando a este no se le pasan los parámetros, los toma de la raíz del simulador.

Agregar a las referencias que si el bus tiene un punto rojo es porque está lleno.

Al ser tan complejo mostrar los histogramas en la pantalla, se acordó que estaba bien exportarlos a un ".txt".

El histograma de perfil de cargas que se muestra en igoR-tp no es relevante, por lo que el simulador debe generar un archivo con el formato correcto pero vacío para enviárselo a igoR-tp.

Con otro caso de estudio hay que tener cuidado en como adaptar la vista en la pantalla. Lo mismo ocurre con las líneas. Por ejemplo, si en vez de 12 hay 50 líneas el cuadro de las referencias se va de la pantalla.

Documentar que si se agrega una línea más, hay que agregar otro color para identificarlo con la línea.

Nos sugirieron inventarnos un ejemplo más pequeño para el manual del desarrollador. Por ejemplo: Ciudad Estrella. Aparte es bueno siempre probarlo con otro caso más.

Cuando tengamos todo andando hay que validar con los datos reales. Podemos sacar dos datos de la tesis de maestría de Antonio: cantidad de buses que se precisan para completar el recorrido y tiempo total de viaje de todos los usuarios.

Tiempo total de viaje:

$$\sum_{i}^{n} \sum_{j}^{n} d_{ij} + t_{ij}$$

Probar con y sin caminata. En la tesis de Antonio los tiempos de caminata se consideran como 0. Lo mismo para los tiempos de subida y de bajada del ómnibus.

Después de validar, tenemos que realizar los experimentos, por ejemplo: si aumenta la frecuencia en un 10%, ¿cómo impacta en el sistema? Esto sería otro capítulo del informe.

El informe debe reflejar las etapas de la construcción de un simulador.

Tenemos que dejar documentados los errores.

Tenemos que poner todo lo que nosotros hicimos relativo a igoR-tp, que modelo de red utiliza, porqué implementamos otra red, etc. Tenemos que poner lo estrictamente necesario y hacer referencias a la documentación anterior.

También tenemos que decir que está organizado en capas (centroides, nodos viales, etc).

También documentar que los datos de entrada se corresponden a otros modelos y no al que implementamos nosotros. Por ejemplo, los datos de los centroides no están obtenidos para que soporten caminatas.

La introducción del informe dejarla para más adelante.

En el marco conceptual debemos definir todos los conceptos del área de aplicación de: Modelado de transporte, GIS, Simulación a eventos discretos.

Tareas:

- Patricia y Gastón
    - Seguir con la documentación del informe.
    - Modificar la implementación con los nuevos cambios. (unidades en tasa de arribos de pasajeros y archivo cargas.txt)
    - Enviar el acta de la reunión del día de la fecha.

Próxima reunión: A coordinar por mail.

Acta

Fecha: 02/09/2010

Participantes:

- Antonio
- Marita
- Patricia
- Gastón

Temas tratados:

1. Validación del Simulador

Los valores obtenidos tras una corrida del simulador son consistentes.

Por lo general, los ómnibus asignados a las líneas de Rivera no tienen problemas de capacidad. Este es otro factor a la hora de validar.

La creación de pasajeros está dada por la matriz de demanda y una exponencial negativa.

El tiempo de pasar de un recorrido de ida a vuelta o viceversa en este momento es 0 pero ese valor es configurable. Esto debe quedar documentado.

Otro dato a validar es el tiempo de viaje de los pasajeros que debería dar en el orden de los 400.

De los valores que observemos, se deben documentar los mínimos, máximos, intervalos y promedios.

Se discutió la estrategia implementada. La misma se divide en 2 etapas. La primera se da en el momento del arribo del pasajero al centroide origen. En ese instante éste debe tomar la decisión de cuál parada elegir para ir a tomar el ómnibus. Para eso, elige aquella parada en la que pase la línea que le lleve menos tiempo en llevarlo a su destino (esto incluye el tiempo de caminata de la parada final al centroide destino). La segunda etapa de la estrategia se da una vez que el pasajero se encuentra en la parada esperando por el ómnibus. Éste tomará el primero que le sirva para llegar a su destino. En caso de que en el mismo instante de tiempo llegue más de un ómnibus que le sirva, tomará el que le lleve menos tiempo. Una vez que el pasajero elije el ómnibus para tomar, no cambiará de opción por más que mientras espera para subir venga otro que le sirva también. El tiempo mínimo de viaje es determinado por la sumatoria de los tiempos fijos entre nodos viales que forman el recorrido desde la parada origen a la destino, más el tiempo de caminata de las paradas a los centroides correspondientes.

El tiempo de simulación lo asigna el usuario. Podría ser un parámetro más que es ingresado en igoR-tp.

Luego de validar el tiempo de viaje de los pasajeros, tenemos que armar el plan de pruebas y un análisis de los datos. Al realizar esta tarea nos tenemos que plantear el horizonte de pruebas (tiempo), que datos vamos a recabar, con que confianza, para luego sacar conclusiones.

Una de las formas de validar el simulador es la salida visual.

2. Informe

En la sección Implementación del Simulador, poner algún detalle pero a nivel macro. Aquí debemos escribir sobre la arquitectura, sobre que librerías usamos, y la manipulación que se hizo para acoplar las mismas. También debemos nombrar las tecnologías usadas. Se debe escribir sobre: EOSimulator, la biblioteca gráfica, los módulos de igoR-tp, y sobre cómo se comunican.

La documentación técnica no puede ir en el informe principal.

Se deberá recurrir a las lecturas de SIG para expresar adecuadamente el proceso de mapeo de coordenadas.

Marita nos enviará el nombre completo del proyecto PDT-48-02 de modo que podamos hacer referencia al mismo.

En la construcción del caso de estudio, para explicar cómo se realizó la zonificación de Rivera, se deberá indicar que se trabajó manualmente en base a una copia impresa del archivo .cad. Esas zonas son el resultado de la zonificación para la tesis de maestría de Antonio.

En el caso de la capa de paradas que tuvimos que agregar paradas, indicar que tuvimos que completar la información ya que estaba incompleta.

Para la sección "Descripción del caso de estudio", extraer la información de la tesis de maestría de Antonio pero escribirlo con nuestras palabras y hacer referencia a la fuente.

Agregar la línea Puerto seco 5 bocas b. Esta tiene el mismo recorrido que la que está definida con el mismo nombre pero tiene que tener distinto sentido. Lleva la misma frecuencia 60 min.

Poner como objetivo del caso de estudio que es para validar el simulador. Y pensar que pruebas vamos a poner en el caso de estudio de manera que quede acorde con el objetivo del mismo.

Tareas:

- Patricia y Gastón
    - Seguir con la documentación del informe.
    - Agregar una nueva línea Puerto Seco 5 bocas b.
    - Modificar el archivo de configuración del algoritmo para que soporte el parámetro tiempo de simulación.
    - Modificar la creación de los pasajeros en el simulador y tiempo de simulación pasado como parámetro
    - Enviar el acta de la reunión del día de la fecha.
- Marita
    - Marita nos enviará el nombre del proyecto PDT.

Próxima reunión: A coordinar por mail.

Acta

Fecha: 08/10/2010

Participantes:

- Antonio
- Marita
- Patricia
- Gastón

Temas tratados:

1. Informe

Debemos armar los capítulos 1 y 2.

En la sección descripción de los resultados poner la interpretación de los resultados.

Una posible razón para que los resultados obtenidos con el simulador y el algoritmo GRASP sean diferentes es que las capas de paradas y de red vial son distintas. Los recorridos en la tesis de Antonio fueron definidos sobre las conexiones entre los centroides.

Cambiar título en la sección 6.4 por "Validación".

Se deberá realizar un análisis de sensibilidad para demostrar que el simulador es una herramienta útil. Acá deberemos plantearnos qué pasaría si por ejemplo subimos la frecuencia y ver qué pasaría con los tiempos de espera. Si se aumenta un 20% la flota, cómo impactaría en los tiempos.

Sería bueno tener el informe completo para noviembre y defender en diciembre.

Tareas:

- Patricia y Gastón
    - Seguir con la documentación del informe.
    - Enviar el acta de la reunión del día de la fecha.
- Marita
    - Marita nos enviará el nombre del proyecto PDT.

Próxima reunión: A coordinar por mail.

Acta

Fecha: 16/11/2010

Participantes:

- Antonio
- Patricia
- Gastón

Temas tratados:

1. Informe

Se validaron los resultados del caso de estudio.

Se decidió que realizáramos otro experimento variando la matriz de demanda. Por ejemplo, estudiar qué pasa si aumenta cada uno de los elementos de la matriz de demanda en un determinado porcentaje. Mostrar que el simulador se comporta acorde, por ejemplo, al aumentar la demanda, tiene que subir la flota de buses utilizada. De los experimentos interesa ver si se llenan los buses, si queda gente afuera, etc. Variar en no más de 10% la

demanda. Antonio nos recomendó ver el ejercicio de la tarea del caso de estudio de los semáforos, que varía en un 5% la tasa de arribo para un año y luego varía otro 5% la tasa para el año siguiente.

Los experimentos sirven para mostrar la utilidad del simulador.

En el anexo estudio de las bibliotecas de simulación, poner la información más importante haciendo referencia a la tablita que sacamos de la revista.

Ver si amerita una sección sobre la comunicación entre igoR-tp y el simulador o si poner esta información en el caso de estudio.

La defensa sería en diciembre.

Tareas:

- Patricia y Gastón
    - Seguir con la documentación del informe.
    - Enviar el acta de la reunión del día de la fecha.
- Marita
    - Marita nos enviará el nombre del proyecto PDT.

Próxima reunión: A coordinar por mail.

Acta

Fecha: 1/12/2010

Participantes:

- Antonio
- Marita
- Patricia
- Gastón

Temas tratados:

1. Informe

A continuación se detallan los comentarios de los tutores sobre el punto 1.2 del informe v2.5. El modelo de estructura estática de un sistema de transporte público, es construido con igoR-tp. Mientras que el modelo dinámico se construye con el simulador. Se debe cambiar el título del punto 1.2.

Las herramientas construidas permiten modelar estos aspectos de los sistemas para ser usado en la planificación del transporte público.

En este punto se debe adaptar lo que está escrito antes con lo que está escrito después que son los objetivos. Para unir el punto 1.2 con los objetivos, Marita sugirió poner al final del punto 1.2: "Para implementar los modelos se necesitan ciertas herramientas de software."

En el punto "2.1.1 Red vial" debemos dejar los dos tipos de representaciones de cruces de calles. Poner la representación que usamos nosotros en la parte de igoR-tp.

En el marco teórico debemos poner los conceptos orientados a nuestro proyecto.

Motivación para extender igoR-tp:

1) fue un requerimiento del usuario

2) se extiende para ser usado en la planificación estratégica y táctica. Usar el modelo en otro tipo de planificación del que se venía usando. Incorporar detalles del comportamiento del pasajero que no se tenían en cuenta. El proyecto marco de este proyecto considera otras cosas. Poner q otras cosas considera.

En la motivación de porque extender igoR-tp, dejar los comentarios mostrados a los tutores sobre la espera de los pasajeros en las paradas, pero no asociarlos con el simulador. Con otros algoritmos también se puede estudiar ese tipo de cosas.

Antonio nos propuso un diagrama de actividades para que quede coherente con los eventos. En la figura 4.4.1 poner en un diagrama los nombres de las clases y en otro las clases sueltas con atributos y métodos.

Aclarar lo de la interfaz gráfica. En particular, enfatizar que se usa la salida gráfica de EOSimulator no siendo la salida grafica uno de los requerimientos fundamentales del proyecto.

Comentarios sobre el capítuo 6.

Para cada sección del informe poner una introducción de un párrafo. Siempre es bueno resumir de lo que se detalla en esa sección de manera genérica, para introducir y facilitar al lector de lo que se va a hablar en esa sección.

En el caso de uso, debemos explicar que las zonas fueron sacadas de la tesis y lo de la matriz de demanda. Cambiar el término teórico por real.

El algoritmo GRASP y TNDP no tienen mucho sentido con nuestro proyecto.

Cambiar el término teórico y poner "asignación". Aclarar que refiere al modelo de asignación Baaj y Mahmassani utilizado para evaluar la solución de Rivera.

Debemos numerar los histogramas.

Comentarios sobre el capítulo 7

Debemos sacar el resumen.

Tenemos que agregar más conclusiones. Por ejemplo: experiencia con los datos, experiencia con las herramientas de desarrollo.

En trabajos futuros, sobre los cambios sugeridos al modelo de red, aclarar que no siempre más detalle es bueno. Poner q ventajas y desventajas tiene agregar más nivel de detalle. Por ejemplo, representación de los cruces de calle. El nivel de detalle depende de los objetivos. Algunos niveles de detalle, podrían ser favorables y otros no.

Poner como anexo el Manual de usuario del simulador.

En el punto 1.5, poner que el manual de usuario de igor-tp se encuentra en el help de la aplicación.

Cerramos los cambios pedidos por los tutores y después mandamos informe

La defensa del proyecto sería en febrero pero terminamos en diciembre.

Tareas:

- Patricia y Gastón
    - Seguir con la documentación del informe.
    - Enviar el acta de la reunión del día de la fecha.
- Marita
    - Marita nos enviará el nombre del proyecto PDT.

Próxima reunión: A coordinar por mail.