UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

FACULTAD DE INGENIERIA

Inteligencia computacional para la generación de matrices origen-destino y la relocalización de paradas de autobuses

Enzo Fabbiani Pérez

Ingeniería en Computación
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Marzo de 2018

UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

FACULTAD DE INGENIERIA

Inteligencia computacional para la generación de matrices origen-destino y la relocalización de paradas de autobuses

Enzo Fabbiani Pérez

Informe de Proyecto de Grado presentado al Tribunal Evaluador como requisito de graduación de la carrera de Ingeniería en Computación

Directores:
Prof. Sergio Nesmachnow
Prof. Renzo Massobrio

Montevideo – Uruguay
Marzo de 2018

Fabbiani Pérez, Enzo

Inteligencia computacional para la generación de matrices origen-destino y la relocalización de paradas de autobuses / Enzo Fabbiani Pérez. - Montevideo: Universidad de la República, Facultad de Ingeniería, 2018.

XIX, 98 p.: il.; 29, 7cm.

Directores:

Sergio Nesmachnow

Renzo Massobrio

Tesis de Grado – Universidad de la República, Ingeniería en Computación, 2018.

Referencias bibliográficas: p. 59 – 63.

1. matrices OD, 2. algoritmos evolutivos, 3. optimización, 4. transporte urbano, 5. sistemas de transporte inteligente. I. Nesmachnow, Sergio, Massobrio, Renzo, . II. Universidad de la República, Programa de Grado en Ingeniería en Computación. III. Título.

Montevideo – Uruguay

Marzo de 2018

RESUMEN

Conocer la movilidad de los habitantes resulta fundamental para la planificación y diseño del sistema de transporte urbano con el fin de brindar un servicio de calidad cada vez más exigente. En este proyecto se estudia el problema de generación de matrices origen-destino y el problema de relocalización de paradas de autobuses. Ambos problemas están relacionados y tienen aspectos en común.

Para el primer problema se implementan dos algoritmos para procesar grandes volúmenes de información del transporte público específico para la ciudad de Montevideo, Uruguay. Estos algoritmos permiten generar la matriz origen-destino mediante la estimación de los destinos analizando la información de los boletos vendidos y la localización de los autobuses. El escenario utilizado corresponde a los datos del año 2015, alcanzando un volumen de 200 GB de datos. Se propone una implementación distribuida, obteniendo mejoras en *speedup* de hasta 17.10.

Para el segundo problema se implementa un algoritmo evolutivo multiobjetivo (más específicamente NSGA-II) para relocalizar las paradas de los autobuses, buscando mejorar la calidad de servicio, optimizando el tiempo de recorrido minimizando el costo. El algoritmo es evaluado con instancias realistas del problema, generadas a partir de la matriz origen-destino obtenida y las líneas y paradas de autobuses correspondientes al año 2015. Los resultados experimentales muestran que el algoritmo es eficiente en la búsqueda de soluciones de calidad, obteniendo mejoras de hasta 16.7 % y 33.9 % en tiempo y costo respectivamente frente a la situación correspondiente al año 2015.

Palabras claves:
matrices OD, algoritmos evolutivos, optimización, transporte urbano, sistemas de transporte inteligente.

ABSTRACT

Knowing the mobility of habitants is essential for planning and design of the urban transport system in order to provide an even more demanding quality service. This project studies the generation of origin-destination matrices problem and the bus stop relocation problem. Both problemas are related and have aspects in common.

For the first problem two algorithms are implemented for processing large volumes of public transport information specific for the city of Montevideo, Uruguay. These algorithms allow generating the origin-destination matrix by estimating the destinations analyzing the information of the tickets sales and the location of the buses. The scenario used corresponds to the data for the year 2015, reaching a volume of 200 GB of data. A distributed implementation is proposed, obtaining improvements in *speedup* of up to 17.10.

For the second problem, a multiobjective evolutionary algorithm (more specifically NSGA-II) is implemented to relocate bus stops, in order to improve quality of service, by optimizing travel time while minimizing cost. The algorithm is evaluated with realistic instances of the problem, generated from the origin-destination matrix obtained and the lines and bus stops for the year 2015. The experimental analysis show that the algorithm is efficient in searching for quality solutions, obtaining improvements of up to 16.7 % and 33.9 % in time and cost respectively compare to the current situation for the year 2015.

Keywords:
OD matrices, evolutionary algorithms, optimization, public transport, intelligent transportation system.

Lista de figuras

Lista de tablas

Lista de algoritmos

Tabla de contenidos

Capítulo 1

Introducción

La complejidad de las actividades desarrolladas a diario en las ciudades impone desafíos para la movilidad de sus habitantes. En ese contexto, los sistemas de transporte urbano juegan un papel fundamental en la vida de las grandes urbes y proveen soluciones para los desplazamientos de gran parte de la población [16]. Actualmente, los principales problemas relativos a la movilidad urbana en grandes ciudades se relacionan con la incapacidad de los sistemas de transporte público de satisfacer las necesidades de un número creciente de usuarios.

Las ciudades inteligentes (*smart cities*) proponen la incorporación de tecnologías de la información y la comunicación con el objetivo de mejorar la calidad y eficiencia de los servicios urbanos [7]. Ciudades como Tokyo, Londres o Nueva York son algunos ejemplos de ciudades que están avanzando hacia este paradigma. Apoyarse en tecnologías de la información en el proceso de diseño y planificación de sistemas de transporte urbano resulta indispensable para ofrecer un servicio de calidad acorde a los ciudadanos.

Los sistemas de transporte inteligente (*Intelligent Transportation Systems*, ITS) son un componente fundamental en las ciudades inteligentes. Los ITS permiten recolectar una gran cantidad de datos acerca del transporte y la movilidad en las ciudades [13]. En grandes urbes, los ITS generan enormes volúmenes de datos que pueden ser procesados con el fin de extraer valiosa información acerca de la movilidad de los ciudadanos. Esta información puede ser ofrecida tanto a los usuarios del sistema de transporte como a los planificadores y tomadores de decisiones.

En este contexto, la Intendencia de Montevideo (IM) introdujo en 2010 un plan de movilidad urbana para rediseñar y modernizar el transporte público en la ciudad [18]. En el marco de este plan se instrumentó el Sistema de Transporte Metropolitano (STM), con el objetivo de centralizar los diferentes componentes del sistema de transporte público. Uno de los primeros pasos fue la incorporación de geolocalización de los autobuses mediante la colocación de un Sistema de Posicionamiento Global (GPS) a cada vehículo y la introducción de una tarjeta inteligente para abonar los viajes (tarjeta STM). Estos dispositivos permiten recolectar un enorme volumen de datos sobre la ubicación de las unidades, las ventas de boletos, los trasbordos entre distintas líneas, entre otros. Actualmente, la IM recolecta datos enviados por las empresas de transporte pertenecientes al STM. Sin embargo, hasta el momento de plantear la investigación de este proyecto, los datos recolectados no eran utilizados con el fin de extraer información respecto a la movilidad de los pasajeros.

Este proyecto presenta una metodología para el procesamiento de datos provenientes de ITS con el fin de extraer información acerca de la movilidad de los pasajeros. En particular, se muestran los resultados correspondientes al procesamiento de los datos recolectados por el STM durante el año 2015 (aproximadamente 13 millones de viajes abonados con tarjetas STM). Específicamente, se describe un método para obtener matrices origen-destino (OD), que indican la cantidad de personas desplazándose entre cada par de puntos de la ciudad en un período determinado de tiempo. Además, se presenta un algoritmo evolutivo para relocalizar las paradas de autobuses y optimizar los tiempos de recorrido. Se reportan los resultados correspondientes a la ejecución del algoritmo en diferentes escenarios, generados a partir de la matriz OD y el diseño de las líneas y sus paradas en el mismo año.

El documento se encuentra estructurado del siguiente modo. El Capítulo 2 define los problemas de generación de matrices origen-destino y de relocalización de paradas de autobuses. Para éste último, se presenta la formulación matemática y su complejidad computacional. En el Capítulo 3 se introduce la principal técnica utilizada en la resolución del problema de relocalización de paradas de autobuses. En el Capítulo 4 se presenta un resumen de los principales trabajos relacionados. En el Capítulo 5 se presentan los algoritmos implementados para la resolución de los problemas de generación de matrices origen-destino y de relocalización de paradas de autobuses. En el Capítulo 6 se presenta la evaluación experimental realizada sobre los algoritmos implementa-

dos y se discuten los resultados obtenidos. Finalmente, el Capítulo 7 presenta las conclusiones del proyecto y las principales líneas de trabajo futuro.

Capítulo 2

Descripción del problema

El presente capítulo describe los problemas de optimización del transporte colectivo abordados en este proyecto. La Sección 2.1 presenta el problema de la generación de la matriz origen-destino, que aporta información fundamental para poder estudiar una mejora en la calidad de servicio del transporte colectivo. La Sección 2.2 presenta el problema de relocalización de paradas de autobuses, que consiste en buscar una configuración óptima en la localización de las paradas de autobuses de forma tal que se reduzcan los tiempos de recorrido, mejorando así la calidad de servicio.

2.1. El problema de la generación de la matriz origen-destino

Las matrices de demanda y las matrices OD suelen ser generadas a través de encuestas realizadas a los pasajeros, las cuales implican un proceso costoso en tiempo y recursos y que sólo brindan información parcial de la movilidad de la población. Este proyecto presenta una alternativa para estudiar la movilidad de la población explotando los datos generados por los ITS, incluyendo datos de geolocalización de los autobuses y de los boletos vendidos.

La metodología propuesta es específica para la generación de matrices de demanda y OD utilizando datos provenientes de los ITS. Teniendo en cuenta el gran volumen de datos a procesar, se propone la aplicación de técnicas de computación distribuída para reducir el tiempo de procesamiento. El enfoque propuesto puede ser fácilmente extendido a otros escenarios y conjuntos de datos.

El principal desafío al momento de generar matrices OD utilizando datos de la venta de boletos mediante tarjetas inteligentes (*smartcards*) consiste en que los pasajeros validan su tarjeta al ascender pero no al descender del autobús. Por esta razón, a pesar de que el origen de cada viaje es conocido con certeza, el destino es desconocido y es necesario estimarlo. Además, algunos pasajeros no utilizan tarjetas inteligentes al abonar sus boletos. En estos casos, existen registros de ventas que no proveen suficiente información cómo para poder estimar su destino.

Las compañías de autobuses que operan en Montevideo están obligadas a enviar la información correspondiente a la geolocalización de los autobúses y a las ventas de boletos a las autoridades de la ciudad. La red de autobuses de Montevideo es compleja; está conformada por unas 144 líneas diferentes y 4835 paradas [20]. En el marco de este proyecto, se considera el conjunto de datos correspondiente a las ventas de boletos y la geolocalización de los autobúses de todas las líneas para el total del año 2015, alcanzando un volumen de 200 GB de datos. La información de geolocalización de los autobuses contiene la posición de cada vehículo, medida con una frecuencia de entre 10 y 30 segundos. Cada registro de GPS de los autobuses contiene la siguiente información:

- *ID línea*, código que identifica la línea
- *ID viaje*, código que identifica en forma unívoca un viaje para una línea
- *latitud*
- *longitud*
- *velocidad* del autobús (en km/h)
- *timestamp*
- *ID parada*, si el GPS se encuentra cerca de una parada, se almacena el identificador de la misma

Los datos correspondientes a las ventas son aquellas realizadas con y sin tarjetas inteligentes. En el caso de las ventas realizadas con tarjetas inteligentes, se almacena un identificador único que permite identificar otros viajes abonados con la misma tarjeta. Por cuestiones de privacidad, este identificador se encuentra enmascarado por una función de *hash* para evitar obtener información sensible de los pasajeros que utilizan el servicio.

Existen dos modalidades diferentes para comprar el boleto: los boletos de 1 hora, permitiendo realizar un transbordo a otra línea dentro de una hora desde

que se compró el boleto; y los boletos de 2 horas, permitiendo realizar transbordos ilimitados dentro de las dos horas desde que el boleto fue comprado. Sin embargo, no es obligatorio utilizar la tarjeta STM para pagar el boleto, los pasajeros pueden pagarlo en efectivo, perdiendo el derecho a realizar transbordos.

Cada registro por venta realizada con o sin tarjeta inteligente contiene la siguiente información:

- *ID viaje*, igual al registro de GPS
- *ID viaje*, igual al registro de GPS
- *latitud*
- *longitud*
- *ID parada*, igual al registro de GPS
- *cantidad de pasajeros*, debido a que es posible comprar boletos para varios pasajeros
- *timestamp*

Si la venta es realizada con tarjeta inteligente se almacena también: *ID tarjeta inteligente*; que identifica la tarjeta STM, la *cantidad de transacciones* realizadas con la misma tarjeta STM y la *última transacción paga*. A partir de la comparación de estos datos es posible identificar si el viaje fue un transbordo. Si es un transbordo, la cantidad de transacciones aumentará mientras que la *última transacción paga* no.

2.2. El problema de relocalización de paradas de autobuses

En esta sección se presenta el problema de relocalización de paradas de autobuses, su formulación matemática y su complejidad computacional.

2.2.1. Introducción

Generar una configuración adecuada de paradas de autobuses es un proceso costoso debido a la dificultad en identificar aquellas que son innecesarias, como por ejemplo paradas que se encuentran muy juntas unas de otras. El problema de relocalización de paradas de autobuses consiste en buscar la configuración

óptima de paradas que respete tanto la calidad de servicio de los pasajeros como el costo de las empresas.

La reestructura de las paradas está restringida por varios factores: *i*) la capacidad máxima de pasajeros que pueden viajar simultáneamente en un autobús, *ii*) la distancia que un pasajero tolera caminar y *iii*) los costos por concepto de salario y combustible. La mejora en la calidad de servicio consiste en minimizar el tiempo de recorrido, teniendo en cuenta información del tiempo de arribo y espera en la parada y la duración del viaje. El costo de las empresas se calcula como la diferencia entre la ganancia por la cantidad de boletos vendidos y los gastos necesarios para hacer los recorridos. En el marco de este proyecto se asume que los únicos gastos son por concepto de salarios de los choferes y el combustible de los vehículos.

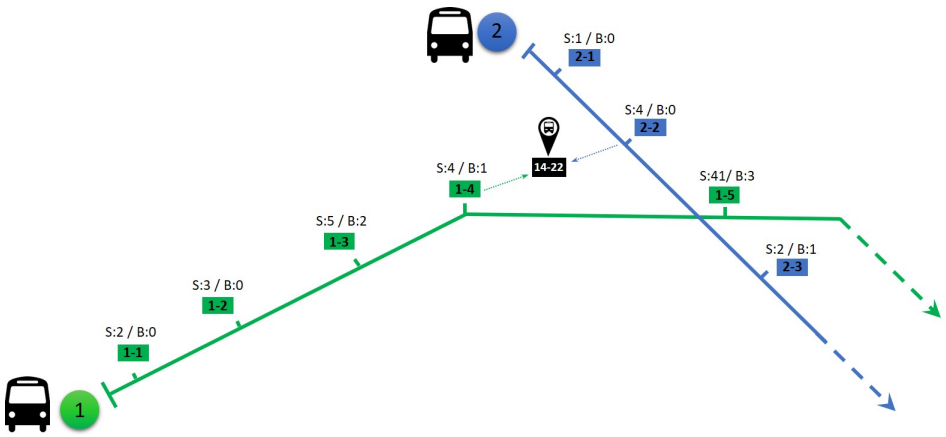

Figura 2.1: Caso de ejemplo del problema de relocalización de paradas de autobuses.

La Figura 2.1 muestra un ejemplo del problema para dos líneas con el mismo destino y que cuentan con una capacidad máxima de 10 pasajeros cada autobús. Para cada una de sus respectivas paradas se muestran los valores correspondientes a la cantidad de pasajeros que desean ascender y descender del mismo. En la parada 1-4 del autobús 1 (línea verde) desean ascender cuatro personas pero el espacio libre en ese instante es de tres asientos. Sin embargo, en la parada 2-2 del autobús 2 (línea azul) desean ascender cuatro personas y el espacio libre en ese instante es de nueve asientos. Por lo tanto, una posible opción es eliminar ambas paradas y crear una nueva en el punto medio de la distancia entre ambas de forma tal que ambas líneas la utilicen y que la persona faltante del autobús 1 ascienda pueda ascender en el autobús 2.

En el marco de este proyecto se utiliza información correspondiente a la ciudad de Montevideo, Uruguay en el año 2015. Todos los datos utilizados están parametrizados, otorgando así mayor flexibilidad a los algoritmos para que puedan ser instanciados fácilmente en otras realidades. En la Sección 5.2.2 se describen con más detalle los parámetros del problema y los datos de entrada a utilizar.

2.2.2. Formulación matemática

Considerándose los siguientes elementos:

- un conjunto (disjunto) de *zonas* $Z = \{z_1, \ldots, z_n\}$, en donde cada zona representa una zona geográfica, por ejemplo un barrio en la ciudad.
- un conjunto de posibles ubicaciones de paradas de autobuses, $S = \{s_1, \ldots, s_p\}$, en donde cada ubicación está definida por sus coordenadas geográficas $(lat_s, long_s)$ en la ciudad. Aquellas ubicaciones que se encuentran activas representan paradas de autobuses para una o varias líneas. Además, algunas ubicaciones son lugares específicos para ser potencialmente utilizados para la instalación de nuevas paradas de autobúses. Por ejemplo, una nueva parada de autobús puede ser definida en una nueva ubicación luego de eliminar paradas cercanas dentro de un área.
- un conjunto de rutas de transporte público, *líneas* $R = \{r_1, \ldots, r_m\}$. Cada línea atraviesa un conjunto de zonas, dado por la función $RZ : R \to Z$. Cada línea tiene una frecuencia fija en minutos, dada por la función $RF : R \to \mathbb{N}$ (por ejemplo, los autobuses que realizan la línea r_j llegan a cada parada con regularidad, cada $RF(r_j)$ minutos). Cada línea tiene una lista ordenada de paradas, dada por la función $RS : R \to S$ (el orden importa).
- una función de *demanda de pasajeros* $D : S \times Z \to N$, donde $D(s_i, z_j)$ indica la cantidad de pasajeros que ascienden al autobús en la parada s_i y viajan al destino z_j. La función de demanda de pasajeros es definida desde paradas a zonas debido a que los pasajeros validan el viaje (mediante tarjeta inteligente o pago en efectivo) cuando ascienden al autobus y no cuando descienden. Por esta razón, es necesario necesario estimar la parada exacta dentro de la zona de destino z_j donde cada pasajero desciende. La estrategia para estimar la parada donde descienden los

pasajeros forma parte de la generación de la matriz de origen-destino, explicada en la Sección 2.1.

- un conjunto de vehículos (*autobuses*) $B = \{b_1, \ldots, b_q\}$; cada autobús tiene una capacidad fija K, que representa el número máximo permitido de pasajeros a viajar en el autobús. Cada vehículo recorre el conjunto de paradas asociadas a una línea en un orden específico. El *tiempo de viaje* entre paradas consecutivas está relacionado con la distancia geográfica entre ellas (respetando el recorrido definido para cada línea), las condiciones de movilidad (tráfico, velocidad máxima, etc.) y la cantidad de pasajeros que ascienden y descienden del autobús en cada parada. El tiempo de viaje está dado por la función $TT : S \times S \rightarrow N$, donde $TT(s_i, s_j)$ es el tiempo requerido por el autobús en viajar desde la parada s_i a la siguiente parada s_j. Cuando un autobus b_h se detiene en la parada s_i, el modelo distingue dos casos: i) al menos un pasajero asciende al autobus b_h en la parada s_i; el tiempo de viaje sufre una demora $t_s + \sum_{k=1}^{k=n} D(s_i, z_k \times t_b)$, donde t_s es el tiempo demandado por el autobus en parar, y t_b es el tiempo necesario por un pasajero en ascender al autobus; ii) el autobus b_h se detiene en la parada s_i solo para permitir a los pasajeros que desciendan, el tiempo de viaje sufre una demora $t_s + t_a$, donde t_a es el tiempo (constante) demandado por los pasajeros en descender del autobus. El modelo asume que descender del autobus es significativamente más rápido que ascender.

  Con el fin de decidir si los pasajeres descienden del autobús en una parada s_i, el modelo utiliza una matriz OD estimada. La generación de esta matriz fue descrita en la Sección 2.1.
- una función de *cantidad de pasajeros* $C : B \times S \rightarrow \mathbb{N}$, donde $C(b_i, s_i)$ indica la cantidad de pasajeros que se encuentran en el vehículo b_i en la parada s_i. La cantidad de pasajeros en cada autobús es actualizada cada vez que un pasajero asciende o desciende del autobús en la parada. En ningún momento la cantidad de pasajeros puede superar la capacidad K.
- una función de *costos* asociados a las empresas de transporte, la cual calcula el costo necesario entre paradas de una línea y está dada por $BC : S \times S \rightarrow \mathbb{N}$, donde $CC(s_i, s_j) = (C_g + C_s) \times dist(s_i, s_j)$ es el costo necesario en viajar desde s_i a s_j, siendo C_g el costo de combustible y C_s el costo de salario que se utiliza para para recorrer 1 Km.

La formulación matemática del problema de relocalización de paradas de autobuses considera las siguientes variables:

- x_{ij} indica la parada i-ésima definida en la línea r_j.
- u_{ij}^k indica que la línea k conecta las zonas z_i y z_j en la ciudad.

Cuando una parada s_i es eliminada de la línea r_j, los pasajeros que viajan desde s_i a la zona z_k son distribuidos entre las paradas cercanas que permiten alcanzar el mismo destino (por ejemplo al ser eliminada la parada s_i de la línea r_l, los pasajeros se trasladan a la parada s_h la cual cumple que $s_h \in RZ(r_l)$ y $s_i \in RZ(r_l)$). Cuando hay más de una opción disponible, la demanda se distribuye en forma inversamente proporcional a la distancia que los pasajeros deben caminar para trasladarse a la nueva parada más cercana. En ese caso, los pasajeros necesitan trasladarse hacia la nueva parada, y el tiempo de caminata necesario está dado por la función $WT : S \times Z \to \mathbb{N}$, donde $WT(s_i, z_k)$ está definida como $WT(s_i, z_k) = \overline{ws} \times dist(s_i, s_h)$ donde $\overline{ws}$ es la velocidad promedio de caminata para los peatones y s_h es la parada más cercana de una línea r_j tal que $z_k \in RZ(r_j)$.

El problema a resolver corresponde a un problema de optimización multiobjetivo, ya que involucra la optimización simultánea de dos funciones. Los objetivos del problema consisten en minimizar el tiempo de recorrido de los autobúses, el cual impacta directamente en la calidad de servicio ofrecida a los pasajeros, dada por la función QT (Ecuación 2.1), maximizando en forma simultánea la ganancia de las empresas mediante reducción de costos, dada por la función CT (Ecuación 2.2). La ganancia está dada como la diferencia entre la cantidad de boletos vendidos y el costo necesario para brindar el servicio.

$$QT = \sum_{j=1}^{j=m} \sum_{i=1}^{i=\#RS(r_j)} (t_s + D(s_i) \times t_b) \; + \; TT(x_{ij}, x_{i-1j}) + \sum_{j=1}^{j=m} \sum_{i=1}^{i=\#RS(r_j)} + \sum_{k=1}^{k=\#Rz(r_j)} WT(s_i, z_k) \tag{2.1}$$

$$CT = \sum_{j=1}^{j=m} \sum_{i=1}^{i=\#RS(r_j)} (G \times D(x_{ij}) - CC(x_{ij}, x_{i-1j}) \tag{2.2}$$

2.2.3. Complejidad del problema

Suponiendo que en cada calle sólo puede haber una única parada de autobús, la cantidad de combinaciones posibles para la localización de una parada de autobús es tanta como la cantidad de calles de una ciudad. Además, cada línea cuenta con sus propias paradas las cuales deben al mismo tiempo ser compartidas entre otras líneas. Es por esto, que obtener una localización de las paradas de autobuses que sea eficiente para el sistema de transporte colectivo no es un problema sencillo. Suponiendo que se tiene un conjunto de K líneas y P paradas de autobuses, la cantidad de combinaciones posibles para localizarlas es K^P. Por lo tanto, su complejidad computacional es exponencial en la cantidad de paradas ($O(K^P)$).

Para instancias realistas de gran tamaño, los algoritmos exactos tradicionales no resultan útiles para hallar una relocalización en las paradas de transporte colectivo en tiempos de ejecución aceptables. La utilización de metaheurísticas permiten calcular soluciones de calidad aceptables en tiempos de ejecución razonables [32]. Las metaheurísticas son estrategias de alto nivel que combinan un conjunto de técnicas de intervención más simples (generalmente heurísticas) con el fin de obtener un procedimiento más potente. Los AE permiten obtener rápidas soluciones candidatas debido a la eficiencia en la exploración del espacio de soluciones.

Capítulo 3

Algoritmos evolutivos

El presente capítulo describe la metodología utilizada para la resolución del problema de relocalización de paradas de autobuses. En la Sección 3.1 se introducen los conceptos generales de los algoritmos evolutivos como técnicas para resolver problemas de búsqueda y optimización. En la Sección 3.2 se describen los algoritmos evolutivos para problemas multiobjetivo, en particular el utilizado para la resolución del problema de relocalización de paradas de autobuses.

3.1. Algoritmos evolutivos

En esta sección se presentan los algoritmos evolutivos como técnica para resolver problemas complejos de búsqueda y optimización, y se introducen los conceptos fundamentales de esta técnica, los cuales serán utilizados a lo largo del documento.

3.1.1. Introducción

Los algoritmos evolutivos (AE) son algoritmos de búsqueda basados en los mecanismos de la selección natural y la genética [15]. Los primeros trabajos sobre estos algoritmos comenzaron a partir de 1960 a cargo de John Holland y sus estudiantes en la Universidad de Michigan. La primera propuesta de algoritmo evolutivo fue presentada en 1975 en su libro *"Adaptation in Natural and Artificial Systems"*, introduciendo los conceptos de selección, cruzamiento y mutación, sentando las bases para la investigación en el área de la computación evolutiva [27].

Los AE se han popularizado en los últimos 30 años y han sido exitosamente aplicados a resolver problemas de optimización subyacentes a problemas complejos del mundo real en múltiples áreas de aplicación [29].

Un AE es una técnica iterativa (cada iteración se denomina generación) que mediante la aplicación de operadores estocásticos sobre un conjunto de individuos (población) busca obtener aquellos más acordes al problema que se busca resolver. Inicialmente, la población se genera a través de alguna técnica estocástica, que puede o no incluir información relativa al problema a resolver. Cada individuo en la población codifica una solución tentativa al problema y tiene asociado un valor de fitness, dado por una función de evaluación que determina qué tan apto es para resolver el problema. La búsqueda de aquellos individuos más aptos, es decir con mayor valor de fitness, se logra mediante la aplicación de operadores evolutivos como la selección de individuos según su fitness, la recombinación de partes de dos individuos y la mutación probabilística de su representación, conduciendo al AE hacia soluciones de mayor calidad. En el Algoritmo 1 se presenta el funcionamiento básico de un AE.

Algoritmo 1 Pseudocódigo de un algoritmo evolutivo.

```
inicializar (P(0))
generacion = 0
mientras no criterioParada() hacer
  evaluar P(generacion)
  padres = seleccion(P(generacion))
  hijos = operadoresEvolutivos(padres)
  nuevaPoblacion = reemplazo(hijos,P(generacion))
  P(generacion) = nuevaPoblacion
  generacion++
fin mientras
retornar mejor individuo encontrado
```

3.1.2. Representación de soluciones

Al momento de trabajar sobre un problema, un AE trabaja sobre una abstracción de las soluciones candidatas al problema denominadas cromosomas. Un cromosoma es un conjunto de genes que contiene información característica de una solución. Los diferentes valores posibles que pueden tomar los genes se denominan alelos. Un genotipo refiere al conjunto de cromosomas que definen

a un individuo y el fenotipo representa un punto del espacio de soluciones del problema [27]. Debido a que se suele utilizar un único cromosoma por individuo, los términos genotipo, cromosoma e individuo se suelen utilizar indistintamente.

Es necesario tener un mecanismo que permita transformar el genotipo en fenotipo y viceversa, es decir, transformar las soluciones del problema en individuos del AE. Este mecanismo es la representación que se le da a los individuos del problema a resolver y es un punto clave en el diseño de un AE, ya que la representación elegida debe ser capaz de modelar el problema adecuadamente.

Entre las representaciones más usuales en la literatura se encuentran la representación binaria y la representación entera. La representación binaria consiste en un vector de bits, donde cada alelo puede tomar el valor 0 ó 1 en codificación binaria. La representación entera consiste en un vector de enteros, donde cada alelo es un número entero.

3.1.3. Función de fitness

Todo individuo en un AE tiene asociado un valor de fitness que representa qué tan apto es para resolver el problema. El valor de fitness es calculado a partir de la función de fitness, la cual asigna un valor a cada individuo mediante la evaluación de su fenotipo. Por lo tanto, la función de fitness es fundamental en el proceso de guiar al AE hacia soluciones tentativas de mejor calidad, seleccionando aquellos individuos más aptos y transmitir así su información genética a sus descendientes. Debido al proceso de codificación en los AE, es posible que un determinado genotipo tenga un fenotipo asociado que no represente una solución factible del problema que se busca resolver. En estos casos, es posible aplicar una función correctiva que asigne un valor de fitness adecuado (a través de un esquema de penalización) que permita reducir su probabilidad de que continúen en el proceso evolutivo. También es posible aplicar un mecanismo de corrección a la codificación de forma tal que represente una solución válida.

3.1.4. Operadores evolutivos

En un AE, los individuos son sometidos a ciertas transformaciones denominadas operadores evolutivos. En su versión más elemental, estos operadores son los de selección, recombinación y mutación.

3.1.4.1. Selección

Este operador es el encargado de seleccionar a los individuos más aptos para continuar en el proceso evolutivo. Muchas técnicas diferentes han sido propuestas en la literatura. A continuación, se describen las más utilizadas.

- Selección elitista: garantiza la selección de los individuos más aptos en cada generación.
- Selección proporcional: asigna a cada individuo una probabilidad de ser seleccionado a partir del cociente entre su valor de fitness y la suma del fitness de todos los individuos de la población. Los individuos más aptos tienen mayor probabilidad de ser seleccionados.
- Selección por torneo: sortea un subconjunto de individuos de la población, que luego compiten entre ellos en base a su fitness para ser seleccionados. En general sólo se elige un individuo del subconjunto.

3.1.4.2. Recombinación

Este operador es el encargado de generar descendientes a partir de dos individuos, combinando características codificadas por ambos, en forma análoga al proceso de reproducción en la evolución natural. Diferentes operadores se utilizan dependiendo de la codificación de las soluciones y del problema a resolver. A modo de ejemplo, en la Figura 3.1 se presenta la recombinación de dos puntos o 2PX (*Two Point Crossover*) para una representación de enteros. El operador toma dos puntos al azar del cromosoma y crea dos descendientes tomando los genes entre los puntos de corte de un padre y el resto de los genes del otro padre.

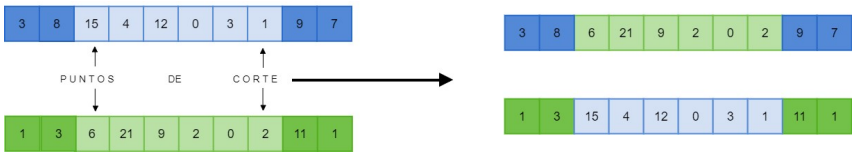

Figura 3.1: Operador de recombinación 2PX.

3.1.4.3. Mutación

Este operador es el encargado de modificar uno o varios genes del cromosoma, introduciendo diversidad en el proceso evolutivo, en forma análoga al

proceso de mutación en la evolución natural. De esta forma se pueden obtener soluciones que no podrían ser obtenidas si solamente se aplicara el proceso de recombinación. A modo de ejemplo, en la Figura 3.2 se presenta la mutación de inversión de bit (*Flip Bit Mutation*) para una representación binaria. Este operador cambia el valor a mutar por el valor opuesto binario.

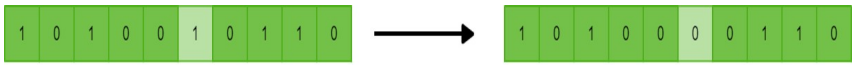

Figura 3.2: Operador de mutación de inversión de bit.

3.2. Algoritmos evolutivos multiobjetivo

En esta sección se introducen los algoritmos evolutivos multiobjetivo aplicados a problemas de optimización. Además, se presenta el algoritmo multiobjetivo utilizado para la resolución del problema de relocalización de paradas de autobuses en este proyecto.

3.2.1. Problemas de optimización multiobjetivo

Un problema de optimización multiobjetivo (*Multiobjective Optimization Problem*, MOP) involucra un conjunto de funciones objetivo a maximizar o minimizar, habitualmente en conflicto entre sí, con restricciones a satisfacer.

En su forma más general, un MOP puede ser formulado de acuerdo a la Ecuación 3.1 [8]. Una solución X perteneciente a R^n es un vector de n variables de decisión $X = (x_1, x_2, .., x_n)^T$.

$$\left.\begin{aligned} Minimizar/Maximizar\ f_m(X), &\quad m = 1, 2, ..., M; \\ sujeto\ a\ g_j(X) \geq 0, &\quad j = 1, 2, ..., J; \\ h_k(X) = 0, &\quad k = 1, 2, ..., K; \\ x_i^{(L)} \leq x_i \leq x_i^{(U)} &\quad i = 1, 2, ..., n. \end{aligned}\right\} \tag{3.1}$$

Para determinar qué soluciones son mejores que otras en problemas de optimización multiobjetivo se utiliza el concepto de dominancia. Una solución x^1 se dice que domina a otra solución x^2, si se cumplen simultáneamente las siguientes dos condiciones:

1. La solución x^1 no es peor que x^2 en todos los objetivos.
2. La solución x^1 es estrictamente mejor que x^2 en al menos un objetivo.

De esta forma es posible calcular el conjunto de soluciones de todo el espacio de soluciones que cumple que para toda solución una no domine a la otra. Este conjunto de soluciones no dominadas se conoce como conjunto óptimo de Pareto. La región de puntos definida por el conjunto óptimo de Pareto en el espacio de valores de las funciones objetivo se conoce como frente de Pareto. La Figura 3.3 muestra un ejemplo de un problema multiobjetivo con dos funciones a minimizar, con sus respectivas soluciones dominadas, soluciones no dominadas y su frente de Pareto.

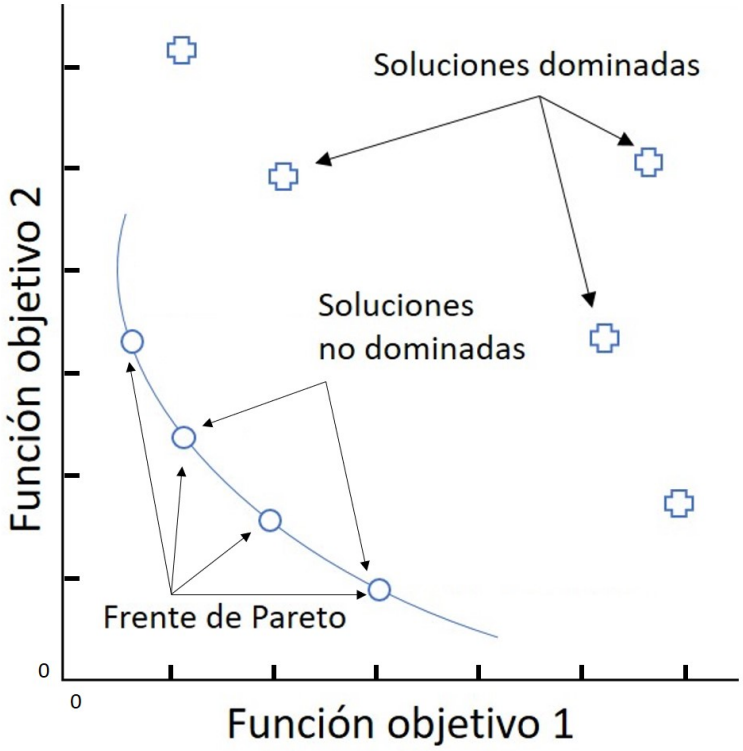

Figura 3.3: Frente de Pareto, soluciones dominadas y soluciones no dominadas para un problema de minimización de dos objetivos.

3.2.2. Algoritmos evolutivos multiobjetivo

Los algoritmos evolutivos para optimización multiobjetivo (*Multiobjective evolutionary algorithm*, MOEA) buscan cumplir con dos propósitos simultáneamente: obtener un conjunto de soluciones óptimas que estén lo más próximas al frente de Pareto (convergencia) y encontrar un conjunto de soluciones que sea lo suficientemente diverso abarcando todo el frente de Pareto [8]. La Figura 3.4 muestra gráficamente los propósitos a cumplirse simultáneamente. La convergencia se obtiene a partir del proceso evolutivo, mientras que la diversidad se obtiene mediante la aplicación de operadores específicos (p. ej. *crowding*, *sharing*).

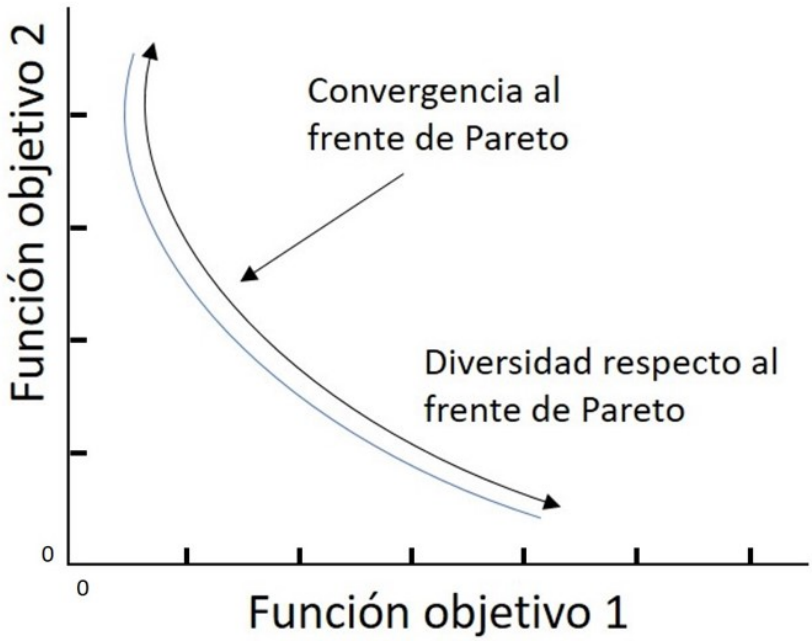

Figura 3.4: Objetivos a cumplirse en un MOEA en un problema de dos objetivos que se buscan minimizar.

3.2.3. El algoritmo NSGA-II

Uno de los algoritmos multiobjetivo más utilizados es el *Non-dominated Sorting Genetic Algorithm*, segunda versión (NSGA-II) [9], el cual ofrece una búsqueda evolutiva de acuerdo a las siguientes características: *i.* utiliza un ordenamiento no dominado, elitísta, que disminuye la complejidad asociada

al chequeo de dominancia, *ii.* utiliza un mecanismo explícito para mantener la diversidad (*crowding*) y *iii.* utiliza una función de fitness que considera la distancia de *crowding*. En el Algoritmo 2 se muestra un pseudocódigo de NSGA-II [6].

Algoritmo 2 Pseudocódigo del algoritmo NSGA-II.

inicializar ($P(0)$) tamaño M
$generacion = 0$
evaluar($P(0)$)
mientras no criterioParada() **hacer**
  $PH = Padres\ U\ Hijos$
  $FPND$ = generarFrentesNoDominados(PH)
  $nuevaPoblacion = 0$
  $i = 1$
  mientras $nuevaPoblacion + FPND <= M$ **hacer**
    distanciaCrowding($FPND(i)$)
    $nuevaPoblacion = nuevaPoblacion\ U\ FPND(i)$
    $i++$
  fin mientras
  ordenar frentes por distancia de crowding
  $nuevaPoblacion = nuevaPoblacion\ U$ seleccionElitista($FPND(i)$)
  $generacion++$
  P($generacion$) = cruzamientoymutacion($nuevaPoblacion$)
fin mientras
retornar mejor solución encontrada

Capítulo 4

Trabajos relacionados

Este capítulo presenta un resumen de los principales trabajos relacionados a los problemas abordados en este proyecto. La Sección 4.1 presenta trabajos que investigan el problema de generación de matrices OD. La Sección 4.2 presenta trabajos que investigan redes de tráfico, planificación de horarios y recorridos. Estos problemas tienen varios puntos en común con el presentado en este proyecto, ya que todos apuntan a mejorar la calidad de servicio. Además, la literatura consultada es amplia en el abordaje de éstos problemas, no así en el problema de relocalización de paradas de autobuses.

4.1. Problema de generación de la matriz origen-destino

Un resumen preciso y detallado sobre el uso de tarjetas inteligentes fue presentado por Pelletier et al. [33]. El trabajo abarca el *hardware* y *software* necesario para la implementación de soluciones de pago con tarjetas inteligente en sistemas de transporte colectivo, así como la privacidad y cuestiones jurídicas que se presentan al utilizar datos de las tarjetas inteligentes. Además, los autores identifican los principales usos para esos datos, incluyendo: la planificación a largo plazo, los ajustes del servicios e indicadores de desempeño de los sistemas de transporte. Finalmente, se presentan ejemplos de la utilización de datos de las tarjetas inteligentes en diferentes ámbitos.

Trépanier et al. [38] presenta un modelo para estimar los destinos de los pasajeros que utilizan tarjetas inteligentes para pagar el boleto, siguiendo un enfoque programático de base de datos. Dos hipotésis son consideradas, las

cuales se usan en general en muchos trabajos relacionados en la literatura: *i.* el origen de un nuevo viaje es el destino del viaje anterior, *ii.* al final del día, los pasajeros regresan al origen del primer viaje del día. Basados en estas suposiciones, los autores proponen un método para seguir la secuencia de viajes de cada usuario en el sistema. Los viajes cuyo armado de la secuencia no es posible (por ejemplo, un único viaje en el día para un usuario particular) son comparados con todos los otros viajes del mes para el mismo usuario, con el fin de encontrar viajes similares con destino conocido. La evaluación experimental fue realizada con información real de una autoridad de tránsito de la ciudad de Gatineau, Quebec. Dos conjuntos de datos fueron utilizados, con 378260 viajes de julio del 2003 y 771239 viajes de octubre del 2003. Los resultados que obtuvieron mostraron que es posible una estimación de los destinos de hasta un 66 % de los viajes. También detectaron que la mayoría de los viajes donde el destino no es posible hallar se dan durante las horas de baja demanda, donde se realizan viajes poco comunes. Considerando solo horas de alta demanda, el porcentaje de viajes a los que es posible estimar sus destinos asciende al 80 %. Sin embargo, la precisión de las estimaciones no pudo ser evaluada debido a la falta de una segunda fuente de datos (por ejemplo, el conteo automático de pasajeros) para la comparación.

Wang et al. [39] propone utilizar una metodología de secuencia de viajes para inferir los origenes y destinos de los pasajeros de los autobuses que utilizan tarjetas inteligentes y datos de Localización Automática de Vehículos (*Automatic Vehicle Location*, AVL) de Londres, Reino Unido. En la realidad estudiada, los autores necesitaron estimar tanto origen como destino de los viajes. Los orígenes fueron calculados con exactitud mediante la utilización del *timestamp* guardado en los registros de las tarjetas inteligentes de los AVL, para determinar la parada de cada viaje. Para estimar los destinos, utilizaron un enfoque similar al presentado por Trépanier et al. [38], siguiendo la secuencia de viajes cuando es posible para inferir los destinos. Los resultados fueron comparados con los datos de la encuesta de pasajeros de transporte de Londres, realizada cada cinco a siete años para cada ruta de autobús e incluyendo el número de personas que ascienden y descienden en cada parada de autobús. El análisis realizado demostró que el origen pudo ser estimado en más del 90 % de los viajes, mientras que tanto el origen como el destino pudieron ser estimados en el 57 % de los viajes. Cuando se contrasta con los datos de la encuesta, la diferencia en la estimación de los destinos se encuentra por debajo del 4 %.

Por último, se presentan dos aplicaciones prácticas. La primera consiste en el estudio de la variación de flujo/carga diaria para identificar las ubicaciones a lo largo del recorrido del autobús donde la demanda de pasajeros es alta, así como también ubicaciones sub-utilizadas. La segunda aplicación consiste en un análisis del tiempo de transbordo, evaluando el tiempo en que los usuarios deben esperar para hacer su transbordo, en función de la parada destino y de los datos del AVL.

Munizaga et al. [28] presentó un enfoque similar para estimar matrices OD en el sistema multimodal de transporte de Santiago, Chile, donde los pasajeros pueden utilizar sus tarjetas inteligentes para pagar boletos en el metro, autobuses y estaciones de autobús.

Diferentes estudios han aplicado estrategias de computación distribuida para procesar grandes volumenes de datos de tráfico, aunque muy pocos trabajos han abordado la estimación de las matrices OD o de demanda utilizand ocomputación distribuída.

Los primeros trabajos aplicaron computación distribuida para recopilar datos de tráfico. Sun [36] propuso un modelo cliente-servidor implementado en CORBA para recolectar información de tráfico en tiempo real y ser utilizado para la estimación de la demanda de origen y destino en forma dinámica. La solución propuesta incluye un cliente CORBA para extraer datos de una red de tráfico y un servidor CORBA para almacenar datos en un repositorio centralizado. Toda la información es procesada para luego ser utilizada en estrategias de asignación dinámica de tráfico (*Dynamic Traffic Assignment*) para la red de tráfico estudiada y en la estimación de matrices OD a partir de la detección de zonas densas aplicando un *framework* de optimización bi-nivel que incorpora información sobre el flujo del tránsito.

Toole et al. [37] propone combinar distintas fuentes de información (registros de llamadas de teléfonos celulares, censos y encuestas) para inferir matrices OD. Los autores combinan varios algoritmos existentes para generar matrices OD, asignar viajes a rutas específicas y para calcular métricas de calidad de uso de las rutas. Adémas, se presenta una aplicación web para visualizar en forma amigable la información obtenida. Los autores mencionan que los cálculos son realizados en paralelo, pero no se describe el modelo utilizado ni se reportan métricas de desempeño.

También mediante la recolección de datos de telefonía móvil, Mellegård [26] propuso una implementación con Hadoop para generar matrices OD mante-

niendo la privacidad de los usuarios. Sin embargo, el análisis experimental fue realizado sobre datos sintéticos (simulados) debido a las dificultades de obtener datos reales de los operadores de telefonía móvil. Además, no se informa de métricas de desempeño, por lo que las ventajas de aplicar Hadoop no están claras.

Huang et al. [17] propuso una metodología para calibración offline/online de sistemas de asignación de tráfico dinámico a través de cálculos distribuidos de gradientes. Se presenta un *framework* adaptativo para la descomposición de redes para el cálculo de métricas en paralelo de la red de tránsito y para simulaciones paralelas, con el fin de trabajar con redes de tráfico de gran escala con un gran número de pares OD y sensores.

Massobrio et al. [25] presentó un análisis sobre la utilización de técnicas de computación distribuida para procesar datos de GPS de los autobuses. Se introduce un enfoque MapReduce para procesar datos históricos para el estudio de métricas para evaluar la calidad de servicio de los sistemas de transporte en Montevideo, Uruguay. Los autores utilizaron una estrategia para calcular la velocidad media de los autobuses y para identificar zonas del recorrido problemáticas, según el retraso y la desviación de los tiempos para llegar a cada parada. La implementación paralela escala adecuadamente cuando se procesan grandes volúmenes de datos.

4.2. Problema de relocalización de paradas de autobuses

Johar et al. [22] realizó una reseña exhaustiva sobre los trabajos realizados en el diseño y planificación de redes de tráfico. El trabajo propone una categorización de los problemas de la red de tráfico, separándolos en cuatro grandes bloques: establecer las rutas (*routing*), planificación (*scheduling*), combinación de ambas e integración y planificación de tránsito masivo. Para cada trabajo investigado, se reportan los objetivos a minimizar, las variables de decisión y la estructura de la red de tráfico. Se puede observar que en gran parte de los trabajos la optimización se realiza en el costo total del recorrido, teniendo en cuenta tanto al usuario como a las empresas que brindan el servicio. Los autores sugieren para la resolución de problemas en las redes de tráfico, la utilización de

algoritmos evolutivos por sobre los algoritmos clásicos de optimizacion debido a la complejidad y cantidad de variables de decisión involucradas.

Deb et al. [11] presentó un algoritmo evolutivo para planificar en forma óptima los horarios y recorridos de autobúses o trenes del transporte colectivo. La metodología se basa en mejorar la calidad de servicio minimizando el tiempo total, teniendo en cuenta tanto el tiempo de recorrido como el tiempo necesario para que los pasajeros arriben al punto de origen y asciendan al vehículo. La evaluación experimental fue realizada sobre un modelo basado generado por los autores bajo ciertas suposiciones. No se presentan métricas de calidad de los resultados del algoritmo evolutivo. Además, realizaron un análisis comparativo entre los enfoques clásicos para resolver el problema (por ejemplo, ramificación y poda) y la utilización de algoritmos evolutivos. Del mismo modo que Johar et al. [22], sugieren la utilización de éstos últimos para la planificación de los sistemas de transporte colectivo debido a la cantidad de variables de decisión involucradas y a la complejidad computacional necesaria para hallar una solución óptima.

Deb et al. [10] propuso un algoritmo evolutivo para la planificación de los horarios de un sistema de autobuses que permite el transbordo de pasajeros entre diferentes líneas. Los autores utilizan ciertas hipótesis que son usuales para este tipo de problemas en la literatura: los autobuses comienzan y terminan su recorrido en el tiempo estipulado y la capacidad de los vehículos es suficiente para acomodar a todos los pasajeros que desean ascender en las paradas de autobús. El objetivo propuesto es minimizar el tiempo de espera total, considerando el tiempo de arribo a la estación de autobús, el tiempo de llegada, el tiempo de espera hasta que arriba el autobús y el tiempo de transbordo. Se presentan los parámetros utilizados en el algoritmo evolutivo, aunque no se especifica si fue realizado un ajuste de los parámetros ni se reportan métricas de calidad de las soluciones. Para la evaluación experimental se utilizaron cuatro escenarios diferentes con 10 instancias cada uno. Se concluye que los algoritmos evolutivos son robustos ya que se obtienen soluciones de calidad para diferentes escenarios.

Król [23] presentó un algoritmo evolutivo para resolver el Diseño de la Red de Tránsito y el Problema del Ajuste de Frecuencias (*Transit Network Design and Frequency Setting Problem*, TNDFSP) utilizando como información una matriz OD y la cantidad de líneas con sus recorridos actuales. El análisis experimental fue realizado sobre la red de tránsito de Bielsko-Biala, Polonia,

la cual cuenta con 46 líneas que brindan servicio a aproximadamente 90 mil personas por día. La generación de la matriz OD fue a partir de datos obtenidos en octubre de 2014 por autoridades de la ciudad, teniendo en cuenta sólo el horario de 13:00 a 17:00 horas. El trabajo detalla brevemente la configuración paramétrica realizada, aunque no brinda detalles sobre cómo fue generada la matriz OD ni tampoco métricas de calidad de los resultados del algoritmo evolutivo.

Bielli et al. [1] propuso un algoritmo evolutivo para mejorar las redes de autobuses, a partir de una optimización en la cantidad de vehículos. El algoritmo utiliza una codificación de las soluciones en la cual cada gen es una línea de autobús con dos alelos que indican la frecuencia y si la línea se encuentra o no en esa red de autobuses. Para las operaciones de recombinación utilizaron la recombinación de dos puntos, mientras que para la mutación eligen al azar una línea del cromosoma y modificaron la frecuencia. Para la función de fitness utilizaron un conjunto de indicadores, como por ejemplo el tiempo promedio de viaje y de espera hasta que se asciende al autobús o la cantidad de vehículos y su capacidad. El análisis experimental fue realizado con cuatro redes de tránsito. Una de ellas es la red real de la ciudad de Parma, Italia, con 22 líneas y 459 paradas, mientras que las demás son escenarios definidos por los autores. Se presenta el ajuste paramétrico realizado, aunque no se reportan métricas de calidad de las soluciones del algoritmo evolutivo. Además, si bien no se reportan resultados de las evaluaciones, se menciona una mejoría del 90 % en los indicadores respecto a la red de partida original.

Xiong et al. [40] propuso la utilización de un algoritmo evolutivo para entrenar una red neuronal que permita obtener soluciones al problema de diseñar una red discreta de transporte (*Discrete transportation network design problem*, DTNDP), un problema NP-díficil. El algoritmo evolutivo, denotado algoritmo evolutivo acumulativo (*cumulative genetic algorithm*, CGA), detecta los individuos con mayor valor de fitness para ser utilizados en conjunto con la nueva generación en la reproducción. La red neuronal es entrenada a partir del CGA. La red cuenta con tres capas, con un total de 62 neuronas y 1261 interconexiones. Para entrenar a la red se utilizaron 2500 soluciones seleccionadas al azar, calculando sus costos con el algoritmo Floyd-Warshall. El analisis experimental fue realizado sobre escenarios definidos por los autores, con 20 modelos diferentes de la red de transporte y una matriz de demanda fija. No se reportan resultados del algoritmo evolutivo ni de la red neuronal, simplemente

se menciona la capacidad del algoritmo evolutivo de hallar soluciones óptimas las cuales luego son utilizadas para entrenar la red neuronal.

4.3. Resumen

La literatura consultada demuestra un gran interés en estudiar diferentes enfoques tanto para la generación de matrices OD como la mejora en la red de tránsito. Para la generación de matrices OD muchos trabajos utilizaron propuestas similares a la presentada en este proyecto, pero en ninguno de ellos se utiliza un enfoque distribuido para su generación a partir de datos de localización de GPS y de boletos vendidos. Para la relocalización de paradas de autobuses no se han encontrado trabajos previos. Sin embargo, muchos enfoques que buscan mejoras en el diseño y planificación de redes de tránsito utilizan algoritmos evolutivos debido a la cantidad de variables de decisión involucradas. En todos los casos, se busca optimizar el tiempo total de recorrido. El análisis de la literatura consultada permite concluir el interés en proponer soluciones basadas en algoritmos evolutivos. En particular, éste proyecto utiliza un enfoque multiobjetivo, debido a que la realidad es muy amplia como para ser modelada con un único objetivo.

Capítulo 5

Implementación

El presente capítulo describe los algoritmos desarrollados tanto para la generación de matrices origen y destino como para el problema de relocalización de paradas de autobuses. La Sección 5.1 presenta el entorno de desarrollo y las bibliotecas utilizadas para la implementación. La Sección 5.2 detalla las estrategias utilizadas y los algoritmos implementados tanto para la generación de la matriz origen y destino como para el algoritmo evolutivo que relocaliza las paradas de autobuses.

5.1. Entorno de desarrollo

La siguiente sección presenta el entorno de desarrollo utilizado para realizar la implementación de los algoritmos de generación de matrices OD y del algoritmo evolutivo para la relocalizacón de paradas de autobuses.

5.1.1. Introducción

Para generar la matriz OD se utilizó Python 2.7.5 además del sistema de información geográfica de código libre QGIS, en su versión 2.8.3 [35] para realizar las operaciones espaciales. El manejo de la arquitectura maestro-esclavo, la creación y distribución de tareas en los diferentes nodos y el manejo de la comunicación entre los nodos esclavos y el nodo maestro fue implementada utilizando el entorno de trabajo *dispy*, en su versión 4.7 [34]. Para la implementación del AE se utilizó Java en su versión 1.8 y la biblioteca ECJ en su versión 24 [24].

5.1.2. Bibliotecas de desarrollo

La siguiente sección presenta las bibliotecas utilizadas para la implementación de los algoritmos de generación de matrices origen y destino y del algoritmo evolutivo para el problema de relocalización de paradas de autobuses.

5.1.2.1. QGIS

QGIS es un Sistema de Información Geográfica (SIG) de código libre que provee una interfaz amigable para visualizar, gestionar y editar mapas y operaciones espaciales. Sus principales características son la visualización de mapas y datos en diferentes formatos, con una interfaz gráfica que ofrece una experiencia de usuario amigable. Además, permite realizar análisis espaciales incluyendo álgebra de mapas, análisis de terreno, modelos hidrólogicos, entre otros [2]. QGIS cuenta con una biblioteca para ejecutar código Python llamada PyQGIS, que permite generar *scripts* que automaticen tareas, generar reportes y realizar operaciones espaciales en aplicaciones personalizadas. La biblioteca PyQGIS fue utilizada para la realización de operaciones espaciales tales como la distancia entre dos puntos, utilizada en la generación de la matriz OD, descrita en la Sección 5.2.1.

5.1.2.2. Dispy

Dispy es un entorno de desarrollo en Python que permite ejecutar, de forma sencilla, tareas paralelas en ambientes distribuidos solucionando el manejo del balanceo de carga y recuperación ante fallos. Dispy brinda una interfaz que permite la creación de *clusters* y programación de tareas a ser ejecutadas en ellos. La creación de un *cluster* en Dispy consiste en encapsular fragmentos de cómputo (código y datos) a ser ejecutados en forma independiente, especificando opciones que controlan cómo las ejecuciones son realizadas [34]. Dispy fue utilizado para facilitar la implementación de la arquitectura distribuida mediante el enfoque maestro-esclavo utilizado en la generación de la matriz OD, descrita en la Sección 5.2.1.1.

5.1.2.3. ECJ

ECJ es una biblioteca de computación evolutiva de código libre desarrollada en Java en el Laboratorio de Computación Evolutiva de la Universidad George Mason, Estados Unidos [24]. ECJ fue diseñada para resolver problemas grandes y complejos, proveyendo herramientas para una gran variedad de algoritmos evolutivos, haciendo énfasis en algoritmos genéticos. La biblioteca permite la resolución de problemas multiobjetivo, con facilidades para implementaciones distribuidas como el modelo de islas y el maestro-esclavo. Además, brinda facilidades para implementar estrategias de evolución estacionarias, co-evolutivas y de evolución natural. Cuenta con una documentación completa y exhaustiva, soporte activo y liberaciones de nuevas versiones periódicamente. La biblioteca ECJ fue utilizada para la implementación del algoritmo evolutivo NSGA-II (descrito en la Sección 3.2.3) para la resolución del problema de relocalización de paradas de autobuses.

5.2. Implementación de los algoritmos propuestos

En esta sección se presentan los detalles de los algoritmos implementados para la generación de la matriz OD y para resolver el problema de relocalización de paradas de autobuses.

5.2.1. Generación de la matriz origen y destino

La estrategia utilizada para la estimación de matrices OD está basada en la reconstrucción de la secuencia de viajes de los pasajeros que utilizan tarjeta inteligente, mediante un enfoque de encadenamiento de viajes, similar al aplicado en trabajos previos [38, 39, 28]. El enfoque propuesto se basa en el procesamiento de cada viaje, obteniendo la parada origen y estimando la parada destino a partir de la información disponible. Se utiliza dos metodologías similares para la estimación de los destinos, dependiendo de si el viaje es o no un transbordo.

En los viajes con transbordo los pasajeros pagan el boleto al ascender al primer autobús con su tarjeta inteligente identificada por su *ID*. Luego, pueden realizar transbordo en uno o más autobuses dentro del tiempo permitido. Para

cada boleto vendido se almacena el *número de transacción* y la *última transacción paga*. Con esta información es posible detectar si el viaje corresponde a un transbordo (*número de transacción* es distinto a la *última transacción paga*) o a un nuevo viaje (*número de transacción* es igual a la *última transacción paga*).

Asumiendo que los pasajeros evitan caminar en exceso y que un pasajero termina su primer viaje en la parada más cercana a la parada donde asciende al autobús del segundo viaje, es posible estimar la parada destino del primer viaje buscando la parada más cercana (a la parada origen del segundo viaje) correspondiente a la primera línea del recorrido.

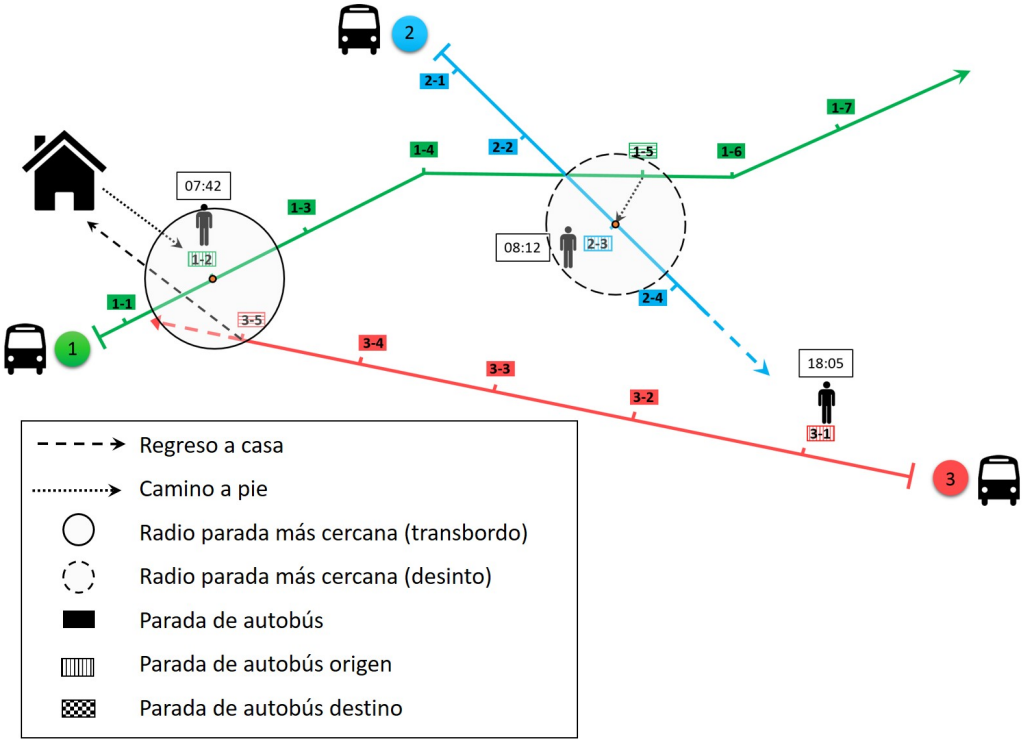

Figura 5.1: Ejemplo de funcionamiento del algoritmo de estimación de viajes con y sin transbordo.

La Figura 5.1 muestra un ejemplo del algoritmo de estimación de destinos. A las 07:42 el pasajero toma el autobús 1 (línea verde) en la parada 1-2. A las 08:12 el pasajero realiza un transbordo y toma el autobús 2 (línea azul) en la parada 2-3 sin pagar un nuevo boleto. Por lo tanto, se puede estimar que el pasajero descendió del autobús 1 en la parada 1-5, que es la parada más cercana

a la parada 2-3 donde el pasajero realizó el transbordo. Con respecto al destino del segundo viaje, por tratarse del último tramo del viaje, es considerado como un viaje sin transbordo, cuyo procesamiento se describe a continuación.

Para los viajes sin transbordo se consideran dos hipótesis. La primer hipótesis es que los pasajeros comienzan un nuevo viaje desde una parada cercana al destino del viaje anterior. La segunda hipótesis es que al final del día, los pasajeros retornan a una ubicación cercana a donde ascendieron al primer viaje del mismo día.

Con el fin de estimar los destinos para viajes sin transbordo es necesario reconstruir la secuencia de viajes hechos por cada pasajero en un mismo día. Un estudio previo realizado sobre el conjunto de ventas mostraron que la mejor opción es considerar cada día nuevo a partir de las 04:00, ya que en ese momento se registra la menor cantidad de ventas. Esto permite considerar pasajeros con diferentes patrones de viaje, tales como aquellos que se desplazan para trabajar durante el día y aquellos que trabajan por la noche.

La metodología para obtener los viajes con transbordo de una persona consiste en iterar sobre todos sus viajes realizados en un período de 24 horas (de 04:00 a 04:00 del día siguiente). Para cada nuevo viaje, se intenta estimar el punto de descenso buscando una parada ubicada en un rango predefinido desde la parada de ascenso del viaje posterior.

En el ejemplo de la Figura 5.1 el pasajero toma el autobús número 3 (línea roja) a las 18:05 para volver a su casa. Con el fin de estimar el destino de este viaje, se busca la parada más cercana del autobús número 3 ubicada dentro de un determinado rango de búsqueda desde la parada 1-2, que es el origen de un viaje anterior. En el ejemplo, la parada 3-5 es la única dentro de este rango, por lo que es elegida como la parada de descendo de este viaje. Cuando no existe ninguna parada dentro del rango, el procedimiento se repite con un radio mayor (dos veces el original) para buscar paradas.

La matriz OD representa la movilidad de los pasajeros entre distintas zonas independientemente del camino elegido para llevarlo a cabo. Por lo tanto, el algoritmo final fusiona los viajes que comienzan con un viaje con transbordo. Es decir, el origen del viaje es la parada donde asciende en el primer autobús y el destino del viaje, es la parada destino del último viaje realizado dentro del rango permitido, si fue posible obtenerla con el algoritmo para viajes sin transbordo.

5.2.1.1. Arquitectura de la solución

La capacidad del algoritmo secuencial para la estimación de la matriz OD está limitada al poder de cómputo de un único nodo. El procesamiento utilizando una infraestructura de alto desempeño basada en arquitecturas distribuidas puede alcanzar un gran nivel de eficacia y escalabilidad para resolver problemas complejos [14].

Pruebas iniciales indicaron que el tiempo necesario para procesar el volumen de datos correspondiente a un mes de venta de boletos y posiciones de los autobuses es de aproximadamente 18 días utilizando un algoritmo secuencial en una computadora de escritorio. Por lo tanto, se tomó la decisión de migrar a una implementación distribuida utilizando múltiples unidades de cómputo. La solución propuesta fue realizar un procesamiento en paralelo mediante el paradigma de arquitectura distribuida del tipo maestro-esclavo, utilizando el paradigma de bolsa de tareas (*Bag of Tasks*, BoT) [5]. Las ventas pueden ser fácilmente separadas por pasajero pudiendo de ésta forma ser procesadas de manera independiente por una única unidad de cómputo.

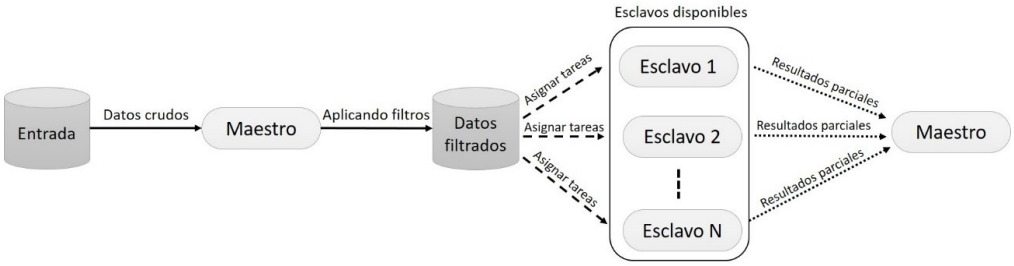

Figura 5.2: Arquitectura distribuida propuesta para el algoritmo de generación de la matriz OD.

La Figura 5.2 muestra la arquitectura propuesta. Inicialmente, el maestro obtiene los datos y descarta los registros que puedan estar corruptos, cómo aquellos que contengan valores de GPS inválidos o que la cantidad de transacciones no sea igual a la última transacción paga más el transbordo. Posteriormente, el algoritmo separa las ventas en bolsas de tareas y las envía a los diferentes nodos esclavos para que sean procesadas. Una vez que cada esclavo obtiene los resultados parciales como resultado de ejecutar el algoritmo para cada pasajero, el nodo maestro genera la matriz OD combinando los resultados parciales.

El nodo maestro es el encargado de crear el *cluster* y la asignación de los datos a los diferentes nodos esclavos para su procesamiento. Los procesos de creación del *cluster* y asignación de tareas es realizado con *dispy*. En el Algoritmo 3 se describe brevemente un pseudocódigo del procesamiento para la ejecución distribuida del algoritmo de generación de matrices OD.

Algoritmo 3 Pseudocódigo para la ejecución distribuida del algoritmo de generación de matrices OD.

```
cluster = crearCluster(algoritmo, dependencias)
tareas, resultadosParciales = []
para cada bolsa de tareas hacer
  tarea = cluster.submit(bolsa)
  tareas = tareas U tarea
fin para
para cada tarea en tareas hacer
  resultadoParcial = ejecutarTarea(tarea)
  resultadosParciales = resultadosParciales U resultadoParcial
fin para
retornar matriz generada a partir de resultados parciales
```

Los datos relativos a las líneas y sus paradas son replicados en cada nodo esclavo. Cada nodo utiliza su propio conjunto de archivos, con el fin de evitar problemas de concurrencia entre nodos durante el acceso.

5.2.2. Algoritmo evolutivo para el problema de relocalización de paradas de autobuses

En esta sección se describen las particularidades del AE implementado para resolver el problema de relocalización de paradas de autobuses.

5.2.2.1. Representación

Si bien ECJ ofrece implementaciones para las representaciones más usuales, la complejidad del problema deriva en que sea necesario definir una representación adaptada al mismo. Como se tiene un conjunto de líneas, una representación sencilla es tener una lista de líneas, donde cada línea tiene una lista de paradas ordenadas desde el origen hasta el destino. Un ejemplo de la representación puede verse en la Figura 5.3 para las líneas 116, 117 y 121. La

solución de ejemplo muestra la correspondencia entre un conjunto de líneas y sus paradas con su respectiva representación. Una restricción del problema es que las paradas origen y destino de cada línea deben permanecer invariables.

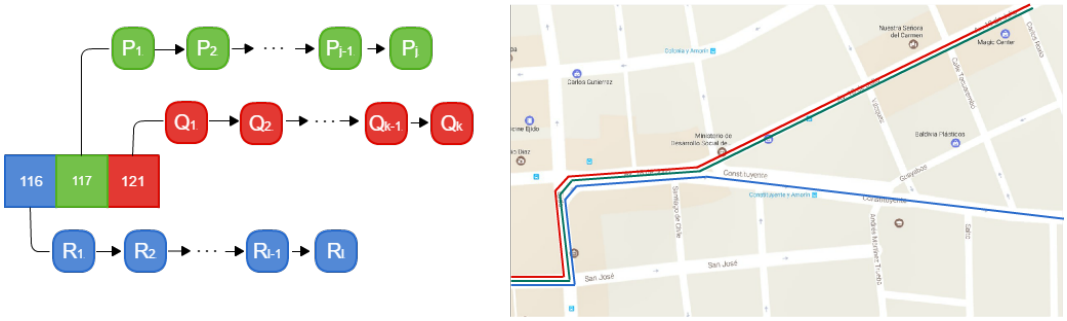

Figura 5.3: Ejemplo de representación de una solución.

5.2.2.2. Datos de entrada

El algoritmo utiliza datos de la realidad del problema. Estos datos son utilizados como entrada del algoritmo y proveen información sobre los costos y ganancias de los viajes, tiempos requeridos, velocidades y otros valores que se encuentran en un archivo de parámetros para hacer más flexible al algoritmo y que pueda ser fácilmente aplicable a otras realidades. Estos parámetros se describen en la Tabla 5.1.

Además, se genera un conjunto de archivos con información necesaria para cada línea los cuales son utilizados por el AE: *i*) linea _pasajeros, el cual contiene, para todas las paradas de esa línea, la cantidad de pasajeros que ascienden y descienden en cada parada (datos tomados a partir de la matriz OD); *ii*) línea _paradas, el cual contiene información de las paradas como su identificador y sus coordenadas geográficas.

5.2.2.3. Población inicial

Para la generación de la población inicial se utiliza la realidad actual para el año 2015 en Montevideo, datos otorgados por la IM. Es decir, se utilizan las líneas y paradas habilitadas en ese año, información pública accesible desde el catálogo de Datos Abiertos [4, 3]. Además, la población es inicializada a partir de la matriz OD obtenida (cuya generación fue descrita en la Sección

Parámetro	Valor	Descripción
RutaArchivos	Ruta absoluta a la ubicación de los archivos de entrada	Ruta absoluta en donde se encuentran los archivos de entrada
ArchivoLineas	líneas	Nombre del archivo que contiene las líneas del problema
CantidadMaximaPasajeros	57	Cantidad máxima de pasajeros que pueden viajar simultáneamente en un autobús
CantidadParadas	6150	Cantidad de paradas total del problema
NuevaDistanciaMaxima	500	Distancia máxima que se puede desplazar una parada en metros
CostoCombustible	5	El costo en $U de la nafta necesaria para realizar 1 km
CostoSalario	100	El costo en $U del salario del conductor para realizar 1 km
StopTime	5	El tiempo necesario del autobús para parar en una parada (en segundos)
BoardTime	5	El tiempo necesario para que el pasajero ascienda al autobús (en segundos)
AlightTime	3	El tiempo necesario para que el pasajero descienda del autobús (en segundos)
WalkingSpeed	5	La velocidad promedio que demora una persona en caminar (en km/h)
Distancias	Haversine	Distancia utilizada por el AE entre paradas y el tiempo que le lleva recorrerlas. Haversine: utiliza la fórmula haversine para hallar la distancia. ITS: toma los datos reales del GPS de los autobúses para Montevideo 2015

Tabla 5.1: Parámetros utilizados por el AE.

2.1) con el fin de contar con información real sobre los destinos de los usuarios. El propio algoritmo es el encargado luego de modificar la matriz debido a la relocalización de las paradas.

Con el fin de otorgar mayor diversidad en la generación de la población inicial se realizan variaciones en los individuos. Estas variaciones son realizadas sobre un subconjunto de los individuos, modificando los pasajeros que ascienden y descienden de las paradas y la ubicación de las mismas. La diversidad en las paradas está dada por una redistribución de los pasajeros que ascienden y descienden hacia las paradas inmediatamente anterior y siguiente a la elegida. La ubicación de las paradas seleccionadas se modifica, reubicándolas a una distancia variable (máximo 500 metros) desde la ubicación actual. Para la reubicación no se consideran las paradas origen y destino por ser características de las líneas. En el Algoritmo 4, se muestra un pseudocódigo del procedimiento de generación de diversidad.

Algoritmo 4 Generación de diversidad en la población inicial.

```
para cada individuo de la población incial hacer
  para cada línea del individuo hacer
    n = seleccionarParadaAgregarDiversidad()
    redist = cantidadGenteRedistribuir()
    moverGente(n − 1,redist/2)
    moverGente(n + 1,redist/2)
    m = seleccionarParadaAgregarDiversidad()
    dist = obtenerDistanciaReubicar()
    moverParada(m,dist)
  fin para
fin para
```

5.2.2.4. Operadores evolutivos

Los operadores evolutivos son un componente fundamental del proceso evolutivo ya que son los encargados de guiar al AE hacia soluciones de calidad a medida que pasan las generaciones. A continuación, se presentan los operadores evolutivos utilizados en el marco de este proyecto.

Selección. Se utilizó el operador de selección por torneo, ya que ofreció una adecuada presión selectiva sobre pruebas iniciales realizadas en un escenario reducido.

Recombinación. Para recombinar los individuos se utilizó el operador de cruzamiento de un punto (SPX) para cada línea de las soluciones a mutar. Las soluciones que violen las restricciones del problema (por ejemplo, que superen la capacidad del autobús) son eliminadas. La Figura 3.1 muestra un ejemplo de recombinación para una instancia del problema con tres líneas y sus respectivas paradas.

Mutación. Se utilizó un operador de mutación específico para el problema a resolver. Para cada línea de la solución se elige probabilísticamente una parada a mutar. Luego, se elige una acción a realizar que puede ser dejarla invariable, eliminarla del recorrido o desplazarla una distancia en metros (distancia máxima parametrizable). En el caso que la parada sea eliminada, los pasajeros que ascienden y descienden en ésta parada son reubicados en forma equitativa

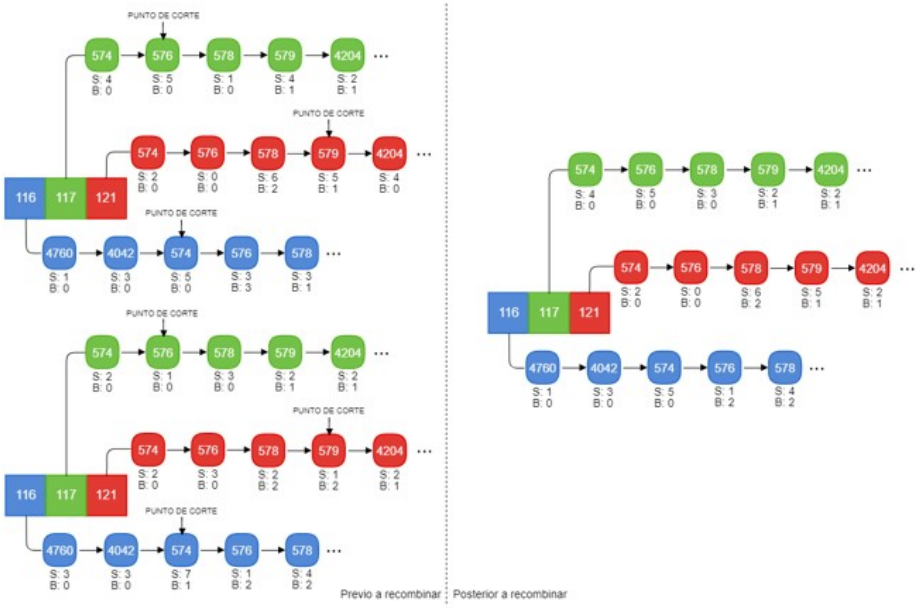

Figura 5.4: Ejemplo de recombinación de una solución.

hacia las paradas inmediatamente siguiente y anterior. En el caso que la parada sea desplazada, los pasajeros que ascienden y descienden en ésta parada son reubicados en forma equitativa entre la parada desplazada y las paradas inmediatamente anterior y siguiente.

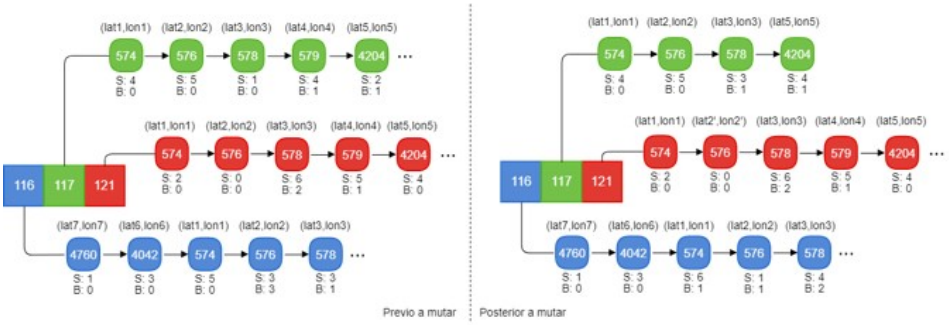

Figura 5.5: Ejemplo de mutación de una solución.

La Figura 5.5 muestra un ejemplo de mutación para una instancia del problema con tres líneas y sus respectivas paradas. Se puede ver, para cada línea, una acción de las posibles. La parada 4042 de la línea 116 se mantiene invariable, la parada 579 de la línea 117 se elimina y la parada 576 de la línea 121 es desplazada a una nueva ubicación.

Función correctiva. Durante el proceso de evolución algunos de los individuos pueden tener en su genoma líneas con paradas que son poco frecuentes para los pasajeros. Por este motivo se implementó una función correctiva de búsqueda local que, si existen dos paradas consecutivas en donde no asciendan pasajeros, se juntan en una única parada en el punto medio de la distancia entre ambas. Las paradas de origen y destino son excluídas en este proceso de búsqueda debido a que son características de las líneas.

Capítulo 6

Evaluación experimental

El presente capítulo presenta la evaluación experimental de los algoritmos implementados para resolver los problemas de generación de la matriz OD y de relocalización de paradas de autobuses. La Sección 6.1 describe la metodología utilizada durante la evaluación experimental del algoritmo para la generación de la matriz OD. La Sección 6.2 describe la metodología utilizada durante la evaluación experimental del AE implementado para el problema de relocalización de paradas de autobuses.

6.1. El problema de generación de matrices origen-destino

La presente sección describe la evaluación experimental realizada para la generación de la matriz OD. La Sección 6.1.1 presenta la instancia utilizada en la evaluación experimental y las métricas utilizadas para medir la eficiencia computacional y para evaluar los resultados obtenidos. La Sección 6.1.2 presenta los resultados experimentales obtenidos en la generación de la matriz OD para la instancia del problema estudiada.

6.1.1. Ambiente de ejecución, instancia del problema y métricas

En esta sección se presenta el ambiente de ejecución, la instancia del problema abordada y las métricas utilizadas en la evaluación experimental para la generación de la matriz OD.

Ambiente de ejecución. El análisis experimental fue realizado en un procesador AMD Opteron 6172 Magny Cours con 24 núcleos de 2.26 GHz, 72 GB de RAM, con sistema operativo CentOS Linux 5.2 del Cluster FING, la infraestructura de cómputo de alto desempeño de la Facultad de Ingeniería, Universidad de la República, Uruguay [30]. Las ejecuciones fueron realizadas en forma exclusiva para evitar que otros procesos puedan interferir en los resultados.

Instancia del problema. Para el análisis experimental fue utilizado el conjunto completo de datos de la ciudad de Montevideo del año 2015, que incluye información de ventas de boletos y la geolocalización de los autobuses; superando el medio millón de tarjetas inteligentes y los 13 millones de viajes.

Métricas. Para evaluar la eficiencia computacional del algoritmo distribuido se evalúan el *tiempo de ejecución* y el *speedup*. El *speedup* se define como la relación entre el tiempo de ejecución para un algoritmo utilizando un único procesador T_1 y el tiempo de ejecución utilizando m procesadores T_m. Esta relación se presenta en la Ecuación 6.1.

$$S_m = \frac{T_1}{T_m} \tag{6.1}$$

Entre los posibles usos que se le puede dar a una matriz OD se encuentran las métricas de perfiles de carga y los mapas de calor. El perfil de carga permite ver cuántos pasajeros se encuentran en el autobús a lo largo de un viaje, mientras que un mapa de calor es una herramienta que permite visualizar características específicas sobre un mapa.

6.1.2. Resultados obtenidos

En la arquitectura propuesta es necesario definir el tamaño correcto del BoT que le será asignado a cada nodo esclavo, con el fin de optimizar el balanceo de carga y la comunicación entre los nodos esclavos y el nodo maestro. Diferentes ejecuciones fueron realizadas variando el tamaño de la BoT y la cantidad de procesadores (*cores*). Para cada combinación se realizaron cinco ejecuciones independientes, a excepción de la ejecución con un único procesador debido al tiempo requerido para finalizarse. Los resultados obtenidos al ejecutar el algoritmo de generación de la matriz de demanda se muestran en la Tabla

6.1, donde se indica la cantidad de procesadores (*#cores*) y el tamaño del BoT (*#BoT*) utilizados en cada ejecución. Luego, para cada combinación de éstos, se presenta el valor promedio, la desviación estándar y el mejor valor alcanzado de tiempo de ejecución y el valor de *speedup*. El tiempo de ejecución está expresado en minutos y los resultados corresponden a cinco ejecuciones independientes del algoritmo.

#cores	*#BoT*	*viajes sin transbordo*		*viajes con transbordo*	
		tiempo		tiempo	
		promedio±σ (mejor)	speedup	promedio±σ (mejor)	speedup
1	1	25920	-	30240	-
16	5000	2092.1±3.4 (2089.6)	12.4	2648.9±3.2 (2645.5)	11.4
16	10000	2372.4±1.8 (2371.1)	10.9	3068.8±3.5 (3063.2)	9.9
24	5000	1582.7±2.4 (1579.4)	16.4	2371.1±2.5 (2368.1)	12.8
24	10000	1858.2±2.1 (1855.9)	14.0	2617.9±3.3 (2614.3)	11.6

Tabla 6.1: Resultados de tiempo de ejecución y *speedup* para la generación de la matriz de demanda al variar la cantidad de *cores* y el tamaño de la BoT.

Los resultados obtenidos sugieren que utilizar un enfoque distribuido mejora significativamente la eficiencia en el procesamiento de los datos necesarios para la generación de la matriz OD. En cuanto al *speedup* obtenido, se alcanzó una mejora significativa de hasta 12.8 y 16.4 para viajes con y sin transbordo respectivamente utilizando un BoT de 5000 viajes y 24 nodos de compúto. Estos resultados confirman que la arquitectura maestro-esclavo propuesta permite mejorar el tiempo de ejecución de las tareas necesarias para la generación de la matriz de demanda, aprovechando múltiples unidades de cómputo.

Se puede observar que el tamaño del BoT tiene un impacto significativo en el tiempo de ejecución del algoritmo. Los tiempos se redujeron cuando se utilizó el tamaño más pequeño para el BoT (5000). Futuros experimentos podrían realizarse para validar si utilizar un tamaño de BoT aún menor implica una mayor eficiencia y determinar el valor en el cual la comunicación entre los nodos esclavo y el nodo maestro se vuelve más costosa derivando en una degradación del tiempo de ejecución.

La Tabla 6.2 muestra los resultados obtenidos al ejecutar el algoritmo para la generación de la matriz OD. Los resultados sugieren que el enfoque distribuido mejora significativamente la eficiencia en el procesamiento de datos necesarios para la generación de la matriz OD. En cuanto al *speedup* alcanzado, se obtuvo una mejora significativa de hasta 17.1 utilizando también un BoT de 5000 viajes y ejecutándose en 24 nodos.

$\#cores$	$\#BoT$	tiempo$\pm\sigma$ (mejor)	speedup
1	1	63972	-
16	5000	5070.3±2.5 (5656.2)	12.58
16	10000	5553.1±2.4 (5525.9)	11.19
24	5000	3730.2±2.1 (3728.6)	17.10
24	10000	4433.4±2.0 (4401.5)	14.39

Tabla 6.2: Resultados de tiempo de ejecución y análisis de resultados para la generación de la matriz OD al variar la cantidad de *cores* y el tamaño de la BoT.

El algoritmo de generación de la matriz OD obtuvo un total de 35755578 viajes de todas las líneas para viajes abonados con tarjetas STM con y sin transbordo.

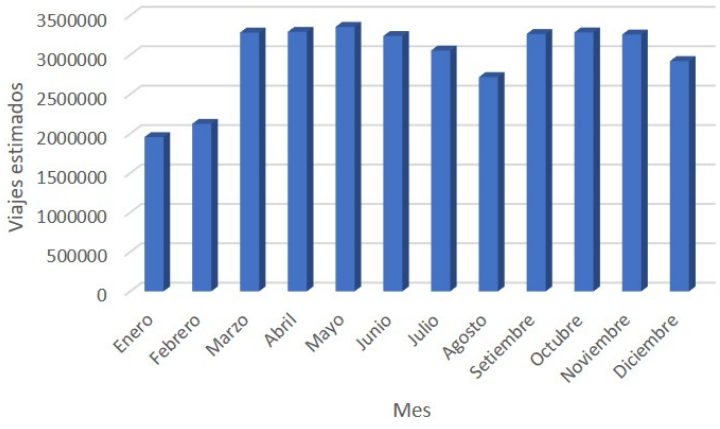

Figura 6.1: Cantidad de viajes obtenidos por el algoritmo de generación de la matriz OD.

La Figura 6.1 muestra la cantidad de viajes obtenidos por el algoritmo de generación de la matriz OD. Se puede observar un aumento considerable

a partir del mes de marzo posiblemente atribuible al fin de las vacaciones y comienzo de las actividades educativas y laborales. También se aprecia un leve descenso de las ventas en los meses de agosto y diciembre, manteniéndose a un nivel prácticamente constante para el resto de los meses.

Mediante la utilización de los perfiles de carga se pueden identificar zonas de un determinado recorrido que estén sub-utilizadas. A modo de ejemplo, la Figura 6.2 muestra la cantidad de personas que viajan en la línea 121 (código de variante 2399) entre las 17:00 y las 19:00 horas (hora pico) para la semana del 16 al 20 de marzo. En color naranja está representada la capacidad máxima de una línea, que equivale a 32 pasajeros sentados y 25 parados. En color azul está representada la capacidad de pasajeros a bordo del autobús a lo largo del recorrido. La frecuencia promedio es de un autobús cada cinco minutos [19].

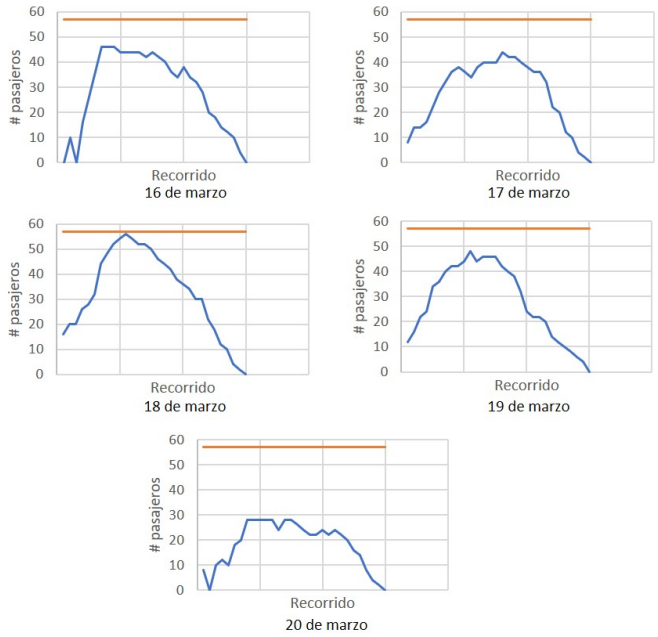

Figura 6.2: Perfil de carga de la línea 121 entre las 17:00 y las 19:00 hs.

En la Figura 6.3 se muestra el perfil de carga para la misma línea entre las 13:00 y las 17:00 horas para la semana antes mencionada. Los resultados sugieren que esta línea no se encuentra sobre-utilizada, ya que si bien para la hora pico el autobús se encuentra próximo a su capacidad máxima, en ninguna de las franjas horarias el autobús se encuentra por encima de su capacidad. Esto puede deberse a que tiene un trayecto relativamente corto, ya que so-

lamente utiliza 30 paradas y recorre únicamente los barrios Centro, Cordón, Tres Cruces, Punta Carretas y Pocitos.

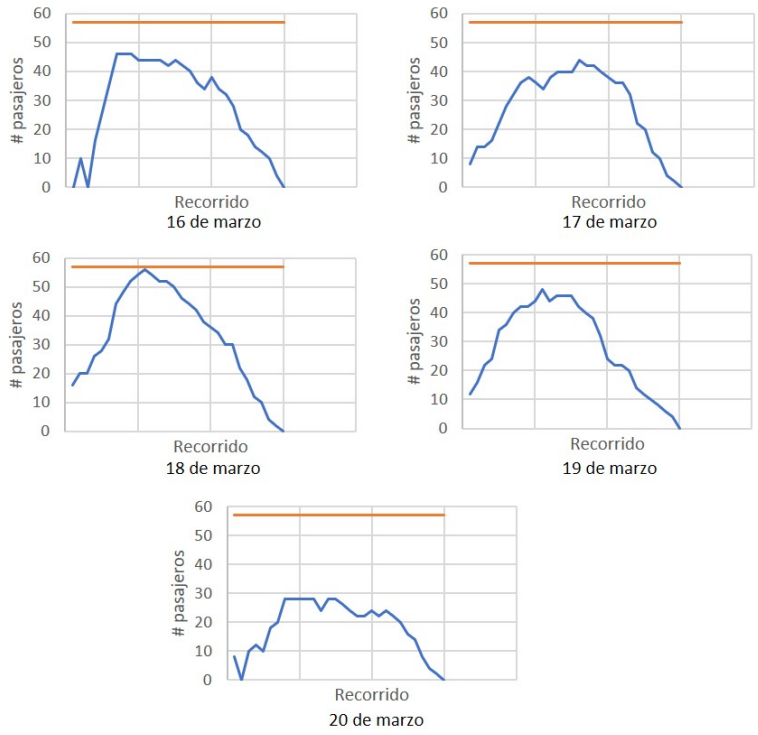

Figura 6.3: Perfil de carga de la línea 121 de 13:00 a 17:00 hs.

En la Figura 6.4 se muestra el perfil de carga de la línea 300 (código de variante 2065) para las franjas horarias de 13:00 a 17:00 y de 17:00 a 19:00 del 11 de mayo de 2015 respectivamente. En el Apéndice A se encuentra detallada la información completa para el resto de la semana. Esta línea tiene un trayecto más largo a la presentada previamente, con 74 paradas y transitando por los barrios Sur, Palermo, Parque Rodó, Cordón, Tres Cruces, Parque Batlle, La Blanqueada, La Unión, Curva de Maroñas, Jardines del Hipódromo, Piedras Blancas, Manga y Puntas de Manga. La frecuencia de esta línea promedio es de un autobús cada 10 minutos [19].

Se puede observar que la línea 300 se encuentra sub-utilizada en los horarios de 13:00 a 19:00 horas, con una mayor demanda en la hora pico, patrón que se repite para el resto de la semana. A diferencia de la línea 121, la línea 300 recorre más del doble de barrios y tiene una frecuencia reducida a la mitad, lo que deriva en un exceso de pasajeros.

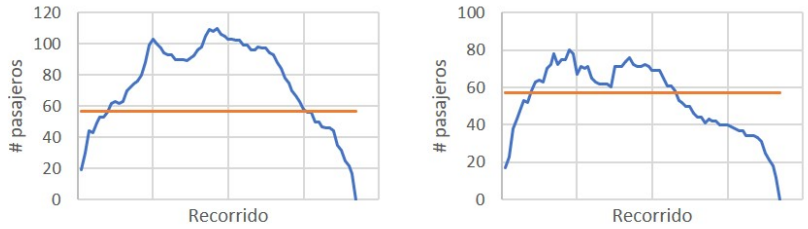

Figura 6.4: Perfil de carga de la línea 300 de 13:00 a 17:00 y de 17:00 a 19:00 hs el 11 de mayo.

En la Figura 6.5 se muestra el perfil de carga de la línea 582 (código de variante 219) para las franjas horarias de 13:00 a 17:00 y de 17:00 a 19:00 horas del 11 de mayo respectivamente. En el Apéndice A se encuentra detallada la información completa para el resto de la semana. Esta línea tiene también un trayecto amplio, con 69 paradas y transitando por los barrios Punta Carretas, Parque Rodó, Centro, Aguada, Prado, Sayago y Peñarol. La frecuencia promedio es de un autobús cada 10 minutos [19]. Se puede observar que esta línea también se encuentra sobre-utilizada en ambos horarios, con una mayor demanda en el horario de 13:00 a 17:00 horas, patrón que se repite para el resto de la semana.

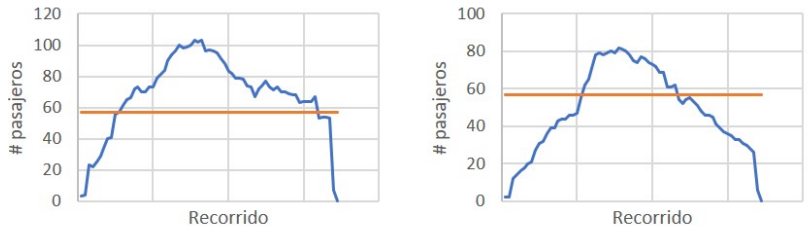

Figura 6.5: Perfil de carga de la línea 582 de 13:00 a 17:00 y de 17:00 a 19:00 hs el 11 de mayo.

El análisis realizado no pretende ser exhaustivo sino mostrar brevemente los patrones que se pueden detectar mediante la visualización de los perfiles de carga obtenidos a partir de la matriz OD. En un trabajo futuro, se puede aplicar esta métrica a líneas y horarios que se crean convenientes para analizar la demanda.

La Figura 6.6 y Figura 6.7 muestran un mapa de calor para los barrios de Centro y Ciudad Vieja respectivamente entre las 17:00 y las 19:00 horas para

el mes de marzo para aquellas paradas de autobuses que son orígenes de los recorridos. Se puede observar que la mayor aglomeración de pasajeros se da en puntos donde se concentran gran cantidad de oficinas o puestos de trabajo.

En la Figura 6.6, se puede observar que la cantidad de pasajeros es superior en las paradas próximas a Plaza Independencia respecto a otras. De forma similar, en la Figura 6.7 se aprecia cómo la afluencia de pasajeros es mayor cuanto más próximo se está de la Plaza Independencia, descendiendo a medida que se va en sentido opuesto. Además, se puede apreciar cómo en calles donde la cantidad de líneas es superior la cantidad de pasajeros también aumenta debido a la cantidad de pasajeros que retornan a sus hogares luego de la jornada de trabajo, como por ejemplo en las paradas de la Avenida 18 de Julio.

Figura 6.6: Mapa de calor del barrio Centro.

También se puede destacar que en las paradas por las que pasan líneas que tienen como destinos barrios que no son la Ciudad Vieja (como en las calles Cerrito y Buenos Aires), la cantidad de pasajeros que toman un autobús es mucho mayor con respecto a paradas con líneas con destino Ciudad Vieja (como la calle 25 de mayo). Esto puede deberse a que en este horario finaliza la jornada laboral y muchos trabajadores retornan a sus hogares.

Figura 6.7: Mapa de calor del barrio Ciudad Vieja.

6.2. El problema de relocalización de paradas de autobuses

La presente sección describe la evaluación experimental llevada a cabo para el AE implementado para resolver el problema de relocalización de paradas de autobuses. La Sección 6.2.1 describe el proceso de generación de instancias del problema. La Sección 6.2.2 describe la metodología utilizada para la evaluación experimental. La Sección 6.2.3 muestra los resultados experimentales obtenidos mediante la ejecución del AE en las instancias de prueba.

6.2.1. Generación de instancias del problema

Con el objetivo de realizar la evaluación experimental del AE es necesario definir instancias realistas del problema. A continuación, se describe el proceso de generación de las instancias y cuáles fueron las utilizadas durante la evaluación experimental.

6.2.1.1. Proceso de generación de instancias

Para la generación de instancias se utilizó información obtenida de diferentes fuentes. La información de destinos, recorridos y paradas de los autobuses es pública y se puede acceder desde el Catálogo de Datos Abiertos [4, 3]. La matriz OD utilizada fue generada en el marco de este proyecto como se describe en la Sección 5.2.1. Se consideran tres franjas horarias para la generación de las instancias de prueba: horario de la mañana, de 07:00 a 13:00 horas, el horario de la tarde, de 13:00 a 19:00 horas y el horario de la noche, de 19:00 a 07:00 horas. Al momento de generar las instancias para la evaluación experimental aún no se contaba con información real sobre el tiempo necesario para que los autobuses se trasladen entre paradas ni tampoco la velocidad promedio, trabajo de otro proyecto de grado, por lo que se utilizó la formula Haversine [21] para el cálculo de las distancias entre las paradas. El AE es lo suficientemente robusto para incluir esta información cuando esté disponible.

6.2.1.2. Instancias de prueba

Para la evaluación experimental del AE que resuelve el problema de relocalización de paradas de autobuses se generaron un total de 30 instancias que se describen a continuación:

- 11 instancias chicas: formadas por 10 líneas con origenes en barrios característicos de Montevideo y diferentes barrios de destino. La matriz OD corresponde al mes de enero.
- 9 instancias medianas: formadas por entre 30 y 60 líneas tomando conjuntos de barrios contiguos de Montevideo como origen y hacia diferentes barrios de destino. La matriz OD corresponde al mes de mayo.
- 10 instancias grandes: formadas por más de 100 líneas. La matriz OD corresponde al mes de setiembre.

Las instancias se encuentran publicadas en la página web del proyecto [12]. El Apéndice B muestra en mayor detalle las instancias definidas.

6.2.2. Metodología del análisis experimental

En esta sección se presentan las características generales utilizadas en la evaluación experimental. Se describe el ambiente de ejecución y las métricas utilizadas, así como también particularidades propias de los MOEA.

6.2.2.1. Ambiente de ejecución

La evaluación experimental del algoritmo NSGA-II fue realizada utilizando una computadora de escritorio con sistema operativo Ubuntu 14.04, la cual cuenta un procesador de dos núcleos *Core* i5-4210U 1.70 GHz y 6 GB de memoria RAM disponible.

6.2.2.2. Ejecuciones independientes

Debido a la naturaleza estocástica de los AE, para diferentes ejecuciones sobre la misma instancia del problema se pueden obtener diferentes resultados. Con el fin de obtener resultados significativos, se realizaron 30 ejecuciones independientes para cada instancia.

6.2.2.3. Configuración paramétrica

Para la configuración paramétrica se utilizaron tres instancias chicas del problema, diferentes a las utilizadas en la evaluación experimental para evitar sesgo. Los parámetros a ajustar fueron el tamaño de la población (valores candidatos 50, 100 y 150), probabilidad de recombinación (valores candidatos 0.6, 0.75 y 0.9) y probabilidad de mutación (valores candidatos 0.001, 0.01 y 0.05). Para cada combinación de parámetros se realizaron 30 ejecuciones independientes de 5000 generaciones cada una. Para la elección de la mejor configuración se tiene utiliza el hipervolumen relativo (definido en la Sección 6.2.2.4) debido a que tiene en cuenta tanto la dispersión de las soluciones halladas como la cercanía al frente de Pareto.

En la Tabla C.1 y la Tabla C.2 del Apéndice C se muestran los resultados obtenidos, los cuales sugieren que utilizar un tamaño de población de 100, probabilidad de recombinación de 0.9 y de mutación de 0.05, permite encontrar las mejores soluciones para las instancias del problema evaluadas.

6.2.2.4. Métricas multiobjetivo

Diferentes métricas han sido propuestas en la literatura para poder evaluar la calidad de los resultados obtenidos mediante los AE, teniendo en cuenta tanto la convergencia hacia el frente de Pareto como la extensión del frente de soluciones hallado [31]. A continuación, se definen las métricas utilizadas en la evaluación experimental.

- Número de puntos no dominados. Evalúa la cantidad de soluciones distintas no dominadas encontradas por el algoritmo (ND).
- Distancia generacional. Evalúa la distancia promedio entre las soluciones no dominadas encontradas por el algoritmo y el frente de Pareto real (Ecuación 6.2). Por lo tanto, valores pequeños indican una mayor proximidad al frente de Pareto real (DG).

$$DG = \frac{1}{ND}\left(\sum_{i=1}^{ND}(d_i^{FP})^k\right)^{\frac{1}{k}} \tag{6.2}$$

- Spacing. Evalúa la distribución de soluciones no dominadas en el frente de Pareto calculado (Ecuación 6.3).

$$spacing = \sqrt{\frac{1}{ND-1}\sum_{i=1}^{ND}(\bar{d} - d_i)^2} \tag{6.3}$$

- Spread. Evalúa la distribución de soluciones no dominadas en el frente de Pareto calculado, incluyendo la distancia hacia los extremos del frente de Pareto real de forma de obtener un valor más exacto de la distribución (Ecuación 6.4). Valores más cercanos a cero indican una mejor distribución de las soluciones no dominadas.

$$spread = \frac{\sum_{h=1}^{k} d_h^e + \sum_{i=1}^{ND}(\bar{d} - d_i)^2}{\sum_{h=1}^{k} d_h^e + ND \times \bar{d}} \tag{6.4}$$

- Hipervolumen relativo. Evalúa el cociente entre el volumen (en el espacio de las funciones objetivo) cubierto por el frente de Pareto calculado por el algoritmo y el volumen cubierto por el frente de Pareto real. Por lo tanto, el valor ideal del hipervolumen relativo es 1 (HR).

En la Ecuación 6.2, d_i^{FP} es la distancia entre la *i-ésima* solución en el frente de Pareto calculado y el punto más cercano en el frente de Pareto real. En las Ecuaciones 6.3 y 6.4, d_i es la distancia entre la solución *i-ésima* del frente de Pareto calculado y su vecino más cercano (la solución *j-ésima*), mientras que $\bar{d}$ es el promedio de todos los d_i. En la Ecuación 6.4, d_h^e es la distancia entre el punto extremo del frente de Pareto real (considerando la *h-ésima* función objetivo) y el punto más cercano en el frente de Pareto calculado. Debido a que algunas métricas utilizan el frente de Pareto real y éste es desconocido para las

instancias del problema a resolver, se construye una aproximación combinando de los puntos no dominados obtenidos en las 30 ejecuciones independientes realizadas.

El número de puntos no dominados y la distancia generacional evalúan la calidad de los resultados obtenidos utilizando un MOEA, en términos de convergencia al frente de Pareto. Por otro lado, las métricas de spacing y spread evalúan la distribución de las soluciones no dominadas, midiendo la capacidad de muestrear el frente de Pareto del problema a resolver.

6.2.3. Resultados obtenidos

En esta sección se presentan los resultados obtenidos en la evaluación experimental del algoritmo NSGA-II implementado para la resolución del problema de relocalización de paradas de autobuses. Al igual que para la configuración paramétrica, se realizaron 30 ejecuciones independientes para cada una de las instancias definidas en la Sección 6.2.1.2.

La Tabla 6.3 muestra los resultados obtenidos por el algoritmo NSGA-II agrupados por familia de instancias para las métricas definidas en la Sección 6.2.2.4. Se reportan los valores promedio, la desviación estándar y el mejor valor alcanzado entre paréntesis. Los resultados detallados por instancia se encuentran en el Apéndice C en las Tablas C.3-C.5.

	DG	spacing	spread	HR
chicas	0.14±0.10 (0.0)	0.12±0.05 (0.02)	1.02±0.13 (0.80)	0.9±0.1 (1.0)
medianas	0.14±0.04 (0.01)	0.17±0.05 (0.02)	0.86±0.06 (0.64)	1.0±0.1 (1.0)
grandes	0.20±0.04 (0.05)	0.27±0.17 (0.02)	0.86±0.07 (0.73)	1.0±0.1 (1.0)

Tabla 6.3: Métricas multiobjetivo para el algoritmo NSGA-II.

Se puede observar que los valores de distancia generacional son sumamente bajos, lo que sugiere una buena convergencia al frente de Pareto real. Los valores de spread alcanzados también son bajos, lo que evidencia una buena distribución de las soluciones no dominadas halladas. Los valores de hipervolumen relativo alcanzados confirman la buena convergencia y diversidad de las soluciones encontradas.

La Tabla 6.4 muestra los valores numéricos alcanzados por las funciones objetivo para cada instancia de prueba número 1 (Apéndice B) junto a los va-

lores para la situación real del año 2015. El tiempo de recorrido está expresado en segundos y la ganancia está expresada en pesos uruguayos.

instancia	*valores reales en 2015*		*valores mejorados por el AE*		% mejora	
	tiempo	ganancia	tiempo	ganancia	tiempo	ganancia
chica	253650.87	303611.63	211286.08	459443.96	16.70	33.92
mediana	1390141.97	550285.98	1314699.66	666476.22	5.43	17.43
grande	2474535.49	1029411.29	2411451.29	1226279.14	2.55	16.05

Tabla 6.4: Resultados comparativos entre la situación actual y la solución encontrada por el AE.

Se puede observar que el AE es capaz de mejorar el tiempo en hasta un 16.70 % y la ganancia en hasta un 33.92 % en los mejores casos. Se aprecia que para una instancia chica, es decir con pocas líneas a modificar, el porcentaje de mejora es sensiblemente mayor. Esto se puede explicar en que, a mayor cantidad de líneas, una modificación en una parada tiene un mayor impacto en el resto de los recorridos. En un trabajo futuro se podrían evaluar otros aspectos a tener en cuenta en el modelo para mejorar estos valores.

Capítulo 7

Conclusiones y trabajo futuro

El presente capítulo presenta las conclusiones del trabajo realizado en este proyecto y las principales líneas de trabajo futuro.

7.1. Conclusiones

En este proyecto se presentan los problemas de generación de matrices OD y de relocalización de paradas de autobuses. Se estudiaron ambos problemas y se relevaron los principales trabajos relacionados en la literatura.

Para el problema de generación de matrices OD se propuso un algoritmo eficiente que utiliza información de ventas de boletos realizadas con tarjetas inteligentes. La metodología propuesta se basa en estimar los destinos de los viajes, debido a que los pasajeros no validan la tarjeta inteligente al descender del autobús. Se propusieron dos estrategias diferentes discriminando si el viaje es o no un transbordo. La evaluación experimental se realizó utilizando datos reales correspondientes al Sistema de Transporte Metropolitano de la ciudad de Montevideo, Uruguay. En términos de eficiencia, los resultados experimentales muestran que la arquitectura distribuida propuesta permite obtener una mejora en el tiempo de ejecución 370 minutos, respecto a un algoritmo secuencial que resuelve el mismo problema, con un tiempo de 56120 minutos y alcanzó un *speedup* de 17.1 cuando se utiliza 24 nodos en el mejor caso.

Para el segundo problema se propuso un algoritmo evolutivo multiobjetivo, que propone minimizar el tiempo de recorrido en los viajes y maximizar la ganancia de las empresas. El algoritmo propuesto se basa en la relocalización

de paradas de autobuses bajo ciertos criterios, como por ejemplo la fusión de dos paradas en una sola en el punto medio entre ambas.

El análisis experimental fue realizado sobre un conjunto de 30 instancias realistas del problema, generadas a partir de la matriz OD estimada aplicando la metodología desarrollada en el proyecto e información asociada a las líneas y sus paradas correspondientes. Los resultados obtenidos muestran que el AE implementado resulta eficiente para obtener soluciones de calidad, siendo capaz de muestriar correctamente el espacio de soluciones del problema. Se obtuvieron mejoras de hasta 16.7 % en el tiempo de recorrido y una reducción del costo de hasta un 33.9 % para las instancias propuestas respecto a la realidad actual.

En el marco de este proyecto se presentó un artículo científico en el congreso internacional *"High Performance Computing Latin America"*, celebrado del 29 de agosto al 1° de setiembre de 2016 en Ciudad de Mexico, Mexico y publicado en *Springer CCSIS Series*. El artículo titulado *"Distributed Big Data analysis for mobility estimation in Intelligent Transportation System"* presenta los algoritmos implementados para la generación de matrices OD y su evaluación experimental con los datos de ventas realizadas con tarjetas STM en el año 2015 en Montevideo, Uruguay. El artículo se adjunta en el Anexo A.

7.2. Trabajo futuro

A continuación, se detallan las principales líneas de trabajo futuro que surgen a partir del trabajo realizado en el marco de este proyecto.

Para el problema de generación de la matriz OD se identifican varias líneas de trabajo futuro. En primer lugar, se propone profundizar en el análisis de los datos que ofrece la matriz OD obtenida en este proyecto con el fin de obtener resultados que puedan ser de utilidad para brindar un mejor servicio. Las métricas que fueron utilizadas en este proyecto (definidas en la Sección 6.1.1) pueden ser de utilidad. En segundo lugar, resulta interesante explorar otros métodos para la obtención de la matriz OD con el fin de validar la fiabilidad del algoritmo propuesto. En tercer lugar, se debería estudiar la posibilidad de incorporar técnicas de aprendizaje automático para mejorar las técnicas de estimación de los destinos, por ejemplo identificando destinos recurrentes de un pasajero.

Para el problema de relocalización de paradas de autobuses se identifican

varias líneas de trabajo futuro. En primer lugar, la incorporación de otras fuentes de información que permitan un mayor conocimiento de la realidad y así permitir al algoritmo evolutivo obtener soluciones de mayor calidad. Por ejemplo, se podría utilizar la velocidad real de los vehículos obtenida a partir de la localización GPS, la cual no pudo ser incorporada al momento de implementar el algoritmo por no contar aún con estos datos. También se puede considerar la incorporación de datos del tráfico, permitiendo detectar tramos entre paradas innecesarios y así modificar el recorrido. En segundo lugar, se debería abordar el problema utilizando otras técnicas con el fin de comparar los resultados por el MOEA propuesto en este proyecto.

Referencias bibliográficas

[1] Bielli, M., Carambia, M., and Carotenuto, P. Genetic algorithms in bus network optimization. *Transportation Research Part C: Emerging Technologies 10* (2002), 19–34.

[2] Bosch, A. Qué es Quantum GIS y por qué utilizarlo. Disponible en: `http://pleiadesic.com/es/que-es-quantum-gis-y-por-que-utilizarlo/`, 2014. [Online; último acceso: Julio 2017].

[3] Catálogo de Datos Abiertos. Transporte colectivo: paradas y puntos de control. Disponible en: `https://catalogodatos.gub.uy/dataset/transporte-colectivo-paradas-y-puntos-de-control`, 2013. [Online; último acceso: Julio 2016].

[4] Catálogo de Datos Abiertos. Líneas de ómnibus, origen y destino. Disponible en: `https://catalogodatos.gub.uy/dataset/lineas-omnibus-origen-y-destino`, 2014. [Online; último acceso: Julio 2016].

[5] Cirne, W., Brasileiro, F., Sauvé, J., Andrade, N., Paranhos, D., Santos-Neto, E., and Medeiros, R. Grid computing for bag of tasks applications. *Proceedings of the 3rd IFIP Conference on E-Commerce, E-Business and EGovernment* (2003).

[6] Coello, C. A. C., Lamont, G. B., and Veldhuizen, D. A. V. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2007.

[7] Deakin, M., and Waer, H. A. From intelligent to smart cities. *Intelligent Buildings International 3* (2011), 133–139.

[8] Deb, K. Multi-objective optimization using evolutionary algorithms: An introduction. *Springer* (2011), 2–3.

[9] Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II. *Springer* (2000), 849–858.

[10] Deb, K., and Chakroborty, P. Time scheduling of transit systems with transfer considerations using genetic algorithms. *Evolutionary Computation 6* (1998), 1–24.

[11] Deb, K., Chakroborty, P., and Subrahmanyam, P. S. Optimal scheduling of urban transit systems using genetic algorithms. *Journal of Transportation Engineering 121* (1995), 544–553.

[12] Fabbiani, E. Instancias de prueba del problema de optimización del transporte colectivo. Disponible en: `https://www.fing.edu.uy/inco/grupos/cecal/hpc/IOTU/files/instancias.rar`, 2017. [Online; último acceso: Julio 2017].

[13] Figueiredo, L., Jesus, I., Tenreiro Machado, J., Rui Ferreira, J., and Carvalho, J. Towards the Development of Intelligent Transportation Systems. *IEEE Intelligent Transportation Systems* (2001), 1206 – 1211.

[14] Foster, I. Designing and building parallel programs. Disponible en: `https://pdfs.semanticscholar.org/09ed/7308fdfb0b640077328aa4fd10ce429f511a.pdf`, 2003. [Online; último acceso: Julio 2017].

[15] Goldberg, D. *Genetic algorithms in search, optimization and Machine Learning.* Addison-Wesley Pub. Co, 1989.

[16] Grava, S. Urban transportation systems: choices for communities. *McGraw-Hill* (2002).

[17] Huang, E., Antoniou, C., Lopes, J., Wen, Y., and Ben-Akiva, M. Accelerated on-line calibration of dynamic traffic assignment using distributed stochastic gradient approximation. In *13th International IEEE Conference on Intelligent Transportation Systems* (2010), pp. 1166–1171.

[18] Intendencia de Montevideo. Plan de movilidad urbana: hacia un sistema de movilidad accesible, democrático y eficiente. Disponible en: http://www.montevideo.gub.uy/sites/default/files/plan_de_movilidad.pdf, 2010. [Online; último acceso: Julio 2017].

[19] Intendencia de Montevideo. Horarios de ómnibus. Disponible en: http://www.montevideo.gub.uy/horariosSTM/, 2017. [Online; último acceso: Julio 2017].

[20] Intendencia de Montevideo. STM: Sistema de Transporte Metropolitano. Disponible en: http://www.montevideo.gub.uy/transito-y-transporte/stm-sistema-de-transporte-metropolitano/el-sistema, 2017. [Online; último acceso: Julio 2017].

[21] J. Mendoza y Rios. Recherches sur les principaux Problemes de l'Astronomie Nautique. *Proceedings of the Royal Society* (1796).

[22] Johar, A., Jain, S. S., and Garg, P. K. Transit network design and scheduling using genetic algorithm - a review. *An International Journal of Optimization and Control: Theories & Applications 6* (2015), 9–22.

[23] Król, A. *Application of the Genetic Algorithm for Optimization of the Public Transportation Lines.* Springer International Publishing, Cham, 2017, pp. 135–146.

[24] Luke, S. ECJ: A java-based evolutionary computation research system. Disponible en: https://cs.gmu.edu/~eclab/projects/ecj/, 2016. [Online; último acceso: Julio 2017].

[25] Massobrio, R., Pias, A., Vázquez, N., and Nesmachnow, S. Map-Reduce for Processing GPS Data from Public Transport in Montevideo, Uruguay. In *2nd Argentinian Symposium on Big Data (AGRANDA)* (2016).

[26] Mellegård, E. Obtaining origin/destination-matrices from cellular network data. Disponible en: "http://publications.lib.chalmers.se/records/fulltext/154702.pdf", 2011. [Online; último acceso: Setiembre 2017].

[27] Mitchel, M. *An introduction to genetic algorithms.* The MIT Press, 1996.

[28] Munizaga, M. A., and Palma, C. Estimation of a disaggregate multimodal public transport origin–destination matrix from passive smartcard data from Santiago, Chile. *Transportation Research Part C: Emerging Technologies 24* (2012), 9–18.

[29] Nesmachnow, S. An overview of metaheuristics: accurate and efficient methods for optimisation. *Int. J. Metaheuristics 3* (1991), 320–347.

[30] Nesmachnow, S. Computación científica de alto desempeño en la Facultad de Ingeniería, Universidad de la República. *Revista de la Asociación de Ingenieros del Uruguay 61* (2010), 12–15.

[31] Nesmachnow, S. Parallel evolutionary algorithms for scheduling on heterogeneous computing and grid events. Disponible en: `https://www.fing.edu.uy/~sergion/PhDThesis-Nesmachnow.pdf`, 2010. [Online; último acceso: Julio 2017].

[32] Nesmachnow, S. Using metaheuristics as soft computing techniques for efficient optimization. *An Encyclopedia of Information Science and Technology, Tercera Edición* (2015), 7390–7399.

[33] Pelletier, M.-P., Trépanier, M., and Morency, C. Smart card data use in public transit: A literature review. *Transportation Research Part C: Emerging Technologies 19*, 4 (2011), 557–568.

[34] Pemmasani, G. dispy: Distributed and parallel computing with/for python. Disponible en: `http://dispy.sourceforge.net`, 2016. [Online; último acceso: Mayo 2017].

[35] QGIS Development Team. Qgis geographic information system. Disponible en: `http://qgis.osgeo.org`, 2009. [Online; último acceso: Mayo 2017].

[36] Sun, C. Dynamic Origin/Destination Estimation Using True Section Densities. *California PATH Research Report* (2000).

[37] Toole, J. L., Colak, S., Sturt, B., Alexander, L. P., Evsukoff, A., and González, M. C. The path most traveled: Travel demand

estimation using big data resources. *Transportation Research Part C: Emerging Technologies 58, Part B* (2015), 162–177.

[38] Trépanier, M., Tranchant, N., and Chapleau, R. Individual trip destination estimation in a transit smart card automated fare collection system. *Journal of Intelligent Transportation Systems 11*, 1 (2007), 1–14.

[39] Wang, W., Attanucci, J., and Wilson, N. Bus passenger origin-destination estimation and related analyses using automated data collection systems. *Journal of Public Transportation 14*, 4 (2011), 131–150.

[40] Xiong, Y., and Schneider, J. B. Transportation nework design using a cumulative genetic algorithm and neural network. *Transportation Research Record 1364* (1993).

APÉNDICES

Apéndice A

Resultados obtenidos para el problema de generación de la matriz OD

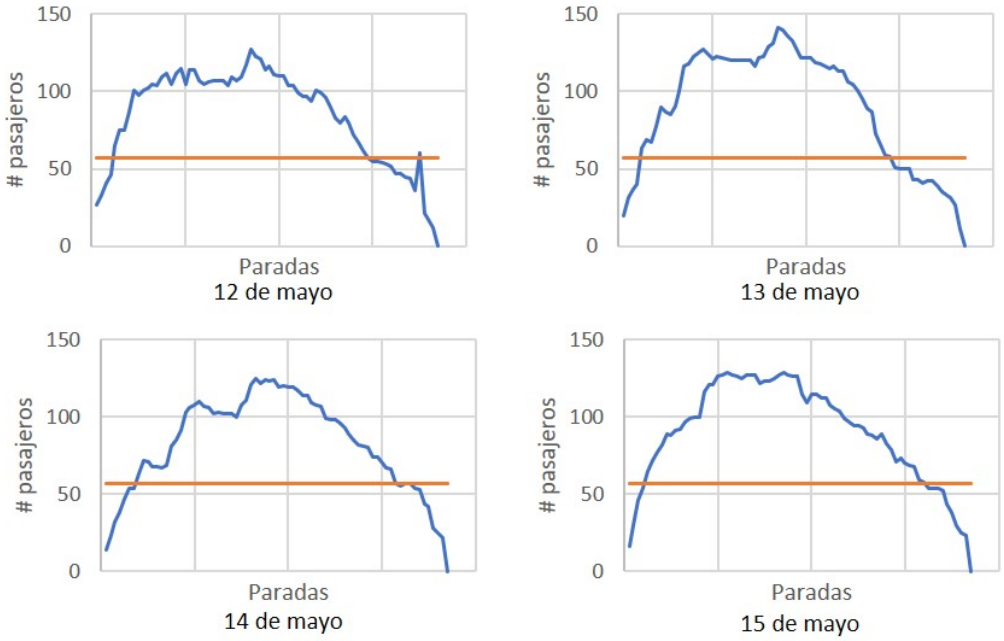

Figura A.1: Perfil de carga de la línea 300 de 13:00 a 17:00 hs.

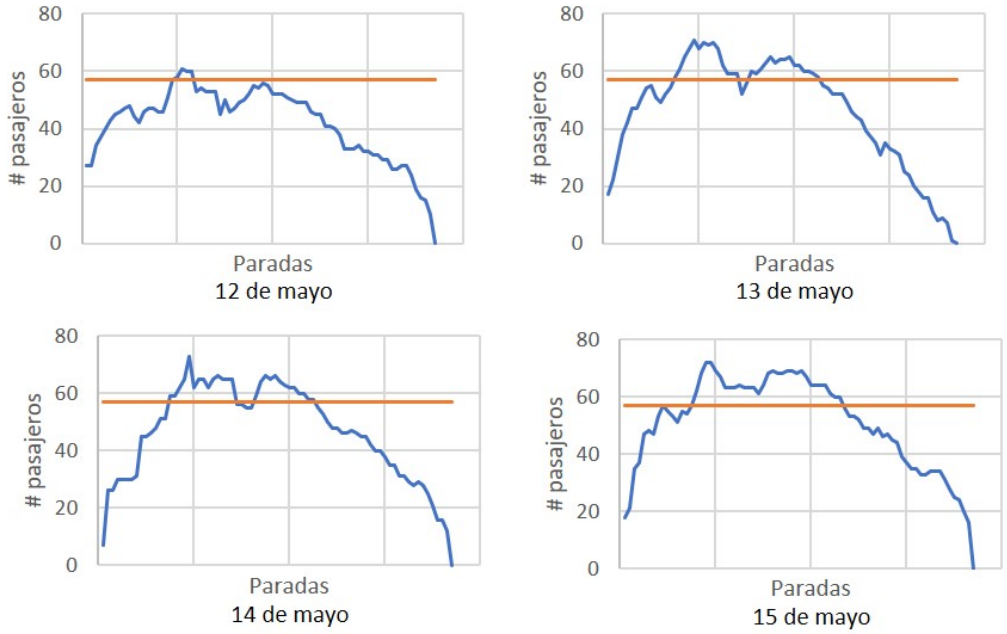

Figura A.2: Perfil de carga de la línea 300 de 17:00 a 19:00 hs.

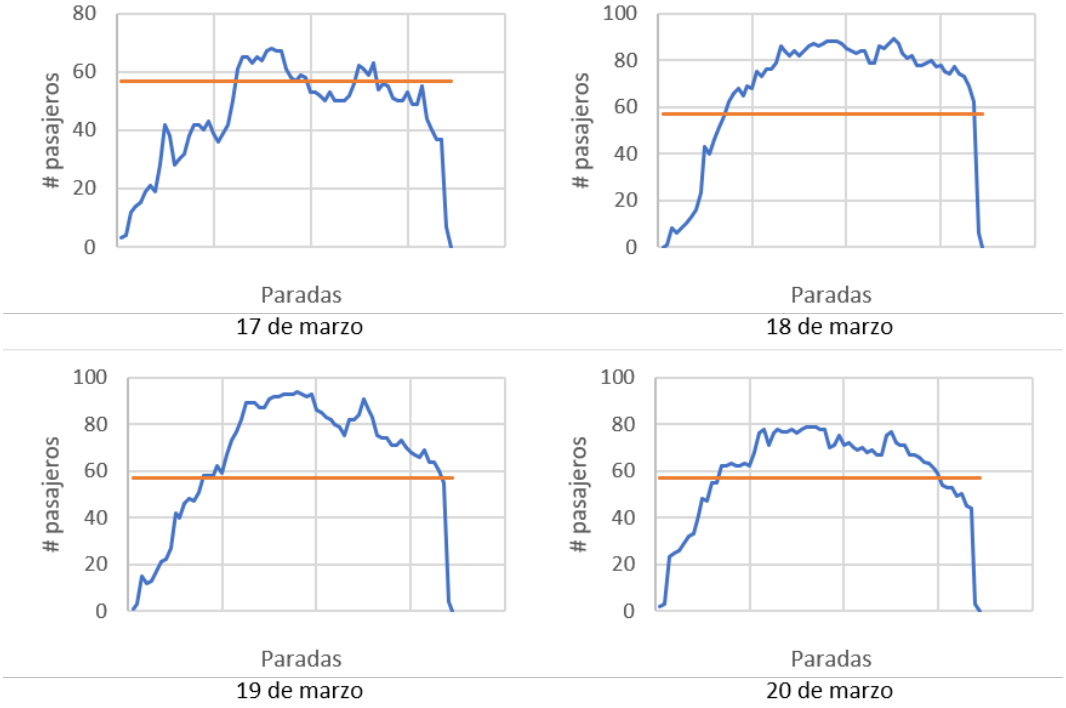

Figura A.3: Perfil de carga de la línea 582 de 13:00 a 17:00 hs.

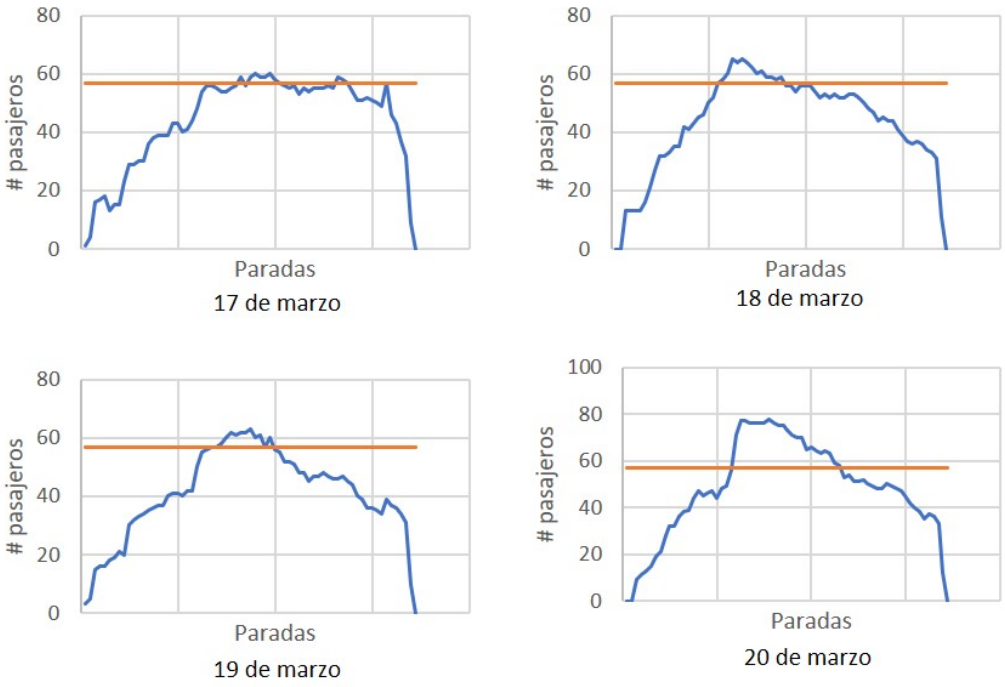

Figura A.4: Perfil de carga de la línea 582 de 17:00 a 19:00 hs.

Apéndice B

Instancias de prueba del problema de relocalización de paradas de autobuses

Instancias chicas

1. Líneas con origen en el barrio Parqué Rodo. La franja horaria es en el horario de 13:00 a 19:00 hs. Líneas 104, 117, 145, 157, 192, 300, 405, 407, 522, 582.
2. Líneas con origen en el barrio Centro. La franja horaria es en el horario de 13:00 a 19:00 hs. Líneas 60, 100, 102, 104, 105, 110, 111, 112, 113, 115.
3. Líneas con origen en el barrio Centro. La franja horaria es en el horario de 07:00 a 13:00 hs. Líneas 116, 117, 121, 126, 127, 141, 142, 144, 147, 148.
4. Líneas con origen en el barrio Punta Carretas. La franja horaria es en el horario de 07:00 a 13:00 hs. Líneas 17, 76, 117, 121, 191, 192, 199, 328, 329, 582.
5. Líneas con origen en el barrio Carrasco. La franja horaria es en el horario de 19:00 a 07:00 hs. Líneas 2, 21, 60, 64, 105, 140, 173, 370, 407, 546.
6. Líneas con origen en el barrio Prado. La franja horaria es en el horario de 07:00 a 13:00 hs. Líneas 147, 148, 149, 151, 185, 186, 522, 526, 538, 582.
7. Líneas con origen en el barrio Puntas de Manga. La franja horaria es en el horario de 19:00 a 07:00 hs. Líneas 71, 110, 149, 169, 192, 300, 456,

505, L2, L13.

8. Líneas con origen en el barrio Malvín. La franja horaria es en el horario de 13:00 a 19:00 hs. Líneas 2, 60, 104, 111, 113, 142, 173, 402, 427, 526.
9. Líneas con origen en el barrio Cerro. La franja horaria es en el horario de 13:00 a 19:00 hs. Líneas 17, 76, 124, 125, 126, 133, 137, 195, 306, 370.
10. Líneas con origen en el barrio Colón. La franja horaria es en el horario de 19:00 a 07:00 hs. Líneas 2, 145, 147, 148, 174, 329, 526, D5, G, G1.
11. Líneas con origen en el barrio Punta de Rieles. La franja horaria es en el horario de 19:00 a 07:00 hs. Líneas 4, 100, 103, 105, 155, 316, 404, 405, D8, L30.

Instancias medianas

1. Líneas con origen en los barrios Sur, Palermo, Cordón y Parqué Rodo. La franja horaria es en el horario de 19:00 a 07:00 hs. Líneas 14, 17, 21, 60, 62, 64, 79, 104, 116, 117, 121, 128, 137, 140, 142, 145, 149, 157, 161, 164, 174, 175, 180, 187, 188, 182, 183, 192, 199, 300, 370, 405, 407, 409, 427, 505, 522, 582, CA1, D1, D10, D11, G.
2. Líneas con origen en los barrios Ciudad vieja, Centro y Cordón. La franja horaria es en el horario de 13:00 a 19:00 hs. Líneas 14, 17, 60, 77, 79, 100, 102, 103, 104, 105, 106, 109, 110, 111, 112, 113, 115, 116, 117, 121, 124, 125, 126, 127, 128, 133, 137, 141, 144, 147, 148, 150, 155, 156, 158, 161, 164, 169, 175, 199, 370, 396, 402, 409, 456, 505, 524, 538, D2, D3, D5, G, D8.
3. Líneas con origen en los barrios Punta Carretas, Pocitos y Buceo. La franja horaria es en el horario de 07:00 a 13:00 hs. Líneas 14, 17, 60, 62, 71, 76, 104, 116, 117, 121, 128, 140, 141, 142, 143, 144, 145, 149, 163, 173, 181, 182, 183, 185, 186, 191, 192, 199, 316, 328, 329, 405, 427, 494, 495, 522, 526, 582, D1, D11, G.
4. Líneas con origen en los barrios Carrasco, Malvin y Punta Gorda. La franja horaria es en el horario de 07:00 a 13:00 hs. Líneas 2, 21, 60, 62, 64, 77, 104, 105, 109, 111, 112, 113, 140, 141, 142, 144, 173, 306, 370, 402, 407, 427, 526, 546, D1, D9, D10, D11, G, L21.
5. Líneas con origen en los barrios Prado, Paso Molino, La Teja y Belvedere. La franja horaria es en el horario de 19:00 a 07:00 hs. Líneas 17, 76, 125, 126, 127, 128, 133, 135, 137, 147, 148, 149, 151, 157, 163, 181, 183, 185,

186, 195, 306, 370, 409, 427, 495, 522, 524, 526, 538, 546, 582, G, L3, L14, L24.

6. Líneas con origen en los barrios Manga, Toledo Chico, Piedras Blancas y Bella Italia. La franja horaria es en el horario de 19:00 a 07:00 hs. Líneas 4, 71, 100, 102, 103, 105, 106, 110, 115, 149, 155, 158, 169, 175, 192, 300, 306, 316, 328, 330, 396, 404, 405, 456, 505, D8, L2, L13, L22, L30.
7. Líneas con origen en los barrios Cerro, Cerro Norte, Santa Catalina, Casabó, La Paloma y Tomkinson. La franja horaria es en el horario de 07:00 a 13:00 hs. Líneas 17, 76, 124, 125, 126, 128, 133, 137, 163, 185, 186, 195, 306, 370, 495, D2, L1, L4, L6, L7, L12, L15, L16, L17, L18, L19, L23, L26, L37, L63.
8. Líneas con origen en los barrios Colón, Sayago, Peñarol, Lezica y Nuevo París. La franja horaria es en el horario de 19:00 a 07:00 hs. Líneas 2, 127, 128, 135, 145, 147, 148, 150, 151, 157, 174, 195, 329, 405, 409, 427, 494, 495, 522, 526, 582, D3, D5, G, G1, G11, L3, L7, L14, L29.
9. Líneas con origen en los barrios Parque Battle, La Blanqueada y Unión. La franja horaria es en el horario de 19:00 a 07:00 hs. Líneas 2, 4, 14, 60, 62, 71, 76, 77, 100, 102, 103, 105, 106, 109, 110, 111, 112, 113, 115, 121, 140, 141, 142, 143, 144, 157, 163, 174, 181, 182, 183, 185, 186, 187, 191, 192, 300, 316, 328, 329, 330, 404, 405, 407, 494, 495, 526, 538, 546, G.

Instancias grandes

1. La franja horaria es en el horario de 07:00 a 13:00 hs. Líneas 103, 104, 106, 109, 110, 111, 112, 115, 116, 117, 121, 124, 127, 128, 133, 135, 137, 142, 143, 145, 147, 150, 151, 155, 158, 161, 163, 164, 169, 174, 175, 181, 182, 183, 185, 186, 187, 188, 191, 192, 195, 199, 21, 300, 316, 328, 330, 370, 396, 402, 404, 407, 409, 456, 494, 495, 505, 522, 524, 538, 546, 60, 62, 64, 71, 76, 77, 79, CA1, D11, D2, D3, D5, D8, D9, G, G1, G10, G11, G3, G4, G6, G8, L12, L14, L15, L16, L19, L2, L20, L22, L24, L26, L27, L28, L29, L3, L30, L35, L37, L4, L41, L5, L6, L63, L7, L94, PA.
2. La franja horaria es en el horario de 13:00 a 19:00 hs. Líneas 100, 102, 104, 105, 106, 109, 111, 112, 113, 115, 116, 117, 121, 124, 125, 126, 127, 128, 133, 135, 14, 141, 142, 143, 145, 147, 148, 151, 155, 156, 157, 158, 161, 163, 164, 169, 17, 173, 174, 180, 181, 182, 183, 185, 186, 187, 188, 191, 192, 195, 199, 2, 21, 300, 306, 316, 328, 329, 330, 370, 396, 402, 404,

407, 409, 456, 494, 495, 522, 524, 526, 538, 546, 582, 60, 62, 64, 71, 76, 77, CA1, D10, D2, D3, D5, D8, D9, G, G1, G11, G3, G4, G8, L1, L13, L14, L15, L16, L17, L18, L19, L2, L20, L21, L22, L23, L26, L28, L29, L3, L30, L35, L37, L4, L5, L6, L63, L7, L9, L94, M-A, M-B, PA.

3. La franja horaria es en el horario de 19:00 a 07:00 hs. Líneas 100, 102, 103, 105, 106, 109, 110, 111, 112, 113, 115, 116, 117, 121, 124, 125, 126, 128, 133, 137, 14, 141, 144, 145, 147, 148, 149, 150, 151, 155, 156, 157, 161, 163, 164, 169, 17, 173, 174, 175, 180, 181, 182, 183, 185, 186, 187, 191, 192, 195, 199, 2, 21, 300, 306, 329, 370, 396, 4, 402, 404, 405, 407, 409, 427, 456, 494, 495,505, 522, 524, 526, 538, 582, 60, 62, 64, 71, 76, 79, CA1, D1, D10, D2, D3, D5, D8, D9, G1, G11, G4, G6, G8, L13, L14, L16, L17, L18, L2, L20, L21, L22, L26, L27, L28, L29, L3, L30, L37, L4, L41, L5, L6, L63, L7, L9, M-B, PA, PB.
4. La franja horaria es en el horario de 07:00 a 13:00 hs. Líneas 100, 102, 103, 104, 105, 106, 109, 110, 111, 112, 113, 115, 116, 117, 121, 124, 125, 126, 127, 128, 133, 135, 137, 14, 141, 142, 143, 144, 145, 147, 148, 149, 150, 151, 155, 156, 157, 158, 161, 163, 164, 169, 17, 173, 174, 175, 180, 181, 182, 183, 185, 186, 187, 188, 191, 192, 195, 199, 2, 21, 300, 306, 316, 328, 329, 330, 370, 396, 4, 402, 404, 405, 407, 409, 427, 456, 494, 495, 505, 522, 524, 526, 538, 546, 582, 60, 62, 64, 71, 76, 77, 79, CA1, D1, D10, D11, D2, D3, D5, D8, D9, G, G1, G10, G11, G3, G4, G6, G8, L1, L12, L13, L14, L15, L16, L17, L18, L19, L2, L20, L21, L22, L23, L24, L26, L28, L29, L3, L30, L35, L37, L4, L41, L5, L6, L63, L7, L9, L94, M-A, M-B, PA, PB.
5. La franja horaria es en el horario de 13:00 a 19:00 hs. Líneas 100, 102, 103, 104, 105, 106, 109, 110, 112, 113, 115, 116, 117, 121, 124, 125, 126, 127, 128, 133, 135, 137, 14, 140, 141, 142, 143, 144, 145, 147, 148, 149, 150, 151, 155, 156, 157, 158, 161, 163, 164, 169, 17, 173, 174, 175, 180, 181, 183, 185, 186, 187, 188, 191, 192, 195, 199, 2, 21, 300, 306, 316, 328, 330, 370, 396, 4, 402, 404, 405, 407, 409, 427, 456, 494, 495, 505, 522, 526, 538, 582, 60, 62, 64, 71, 76, 77, 79, CA1, D1, D10, D11, D2, D3, D5, D8, D9, G, G1, G10, G11, G3, G4, G6, G8, L1, L12, L14, L15, L16, L17, L18, L19, L2, L20, L21, L22, L23, L24, L27, L28, L29, L3, L30, L35, L37, L4, L41, L5, L6, L63, L7, L9, L94, M-A, M-B, PA, PB.
6. La franja horaria es en el horario de 19:00 a 07:00 hs. Líneas 100, 102, 103, 104, 105, 106, 109, 110, 111, 112, 113, 115, 117, 121, 125, 127, 128,

133, 135, 137, 141, 143, 144, 145, 147, 148, 149, 150, 155, 156, 157, 158, 161, 163, 164, 169, 17, 173, 175, 180, 181, 182, 185, 186, 188, 191, 192, 195, 199, 2, 21, 300, 306, 316, 328, 329, 370, 396, 4, 402, 404, 405, 409, 427, 456, 494, 495, 505, 522, 524, 538, 546, 60, 62, 64, 71, 77, 79, D1, D10, D11, D2, D3, D5, D8, D9, G, G1, G10, G11, G4, G8, L1, L12, L13, L14, L15, L16, L17, L18, L19, L2, L20, L21, L22, L23, L24, L26, L27, L28, L29, L3, L30, L35, L37, L4, L5, L6, L7, M-A, PA, PB.

7. La franja horaria es en el horario de 07:00 a 13:00 hs. Líneas 100, 102, 103, 105, 106, 109, 112, 115, 116, 117, 121, 125, 127, 128, 133, 135, 14, 142, 143, 144, 145, 148, 149, 151, 155, 156, 157, 158, 161, 163, 164, 169, 173, 174, 175, 181, 182, 183, 185, 186, 187, 188, 191, 199, 2, 306, 316, 328, 329, 330, 396, 4, 402, 404, 405, 407, 409, 427, 456, 494, 505, 524, 538, 546, 582, 62, 64, 71, 76, 77, 79, D1, D3, D5, D8, G, G10, G11, G3, G6, G8, L1, L12, L15, L16, L17, L19, L2, L20, L21, L22, L23, L24, L26, L28, L29, L35, L37, L4, L5, L6, L63, L7, L9, L94, M-A, M-B, PA.
8. La franja horaria es en el horario de 13:00 a 19:00 hs. Líneas 100, 102, 103, 104, 105, 106, 109, 110, 111, 112, 115, 116, 117, 121, 125, 126, 127, 128, 133, 135, 137, 14, 140, 141, 142, 143, 144, 147, 148, 149, 150, 151, 156, 157, 158, 161, 163, 164, 169, 17, 173, 175, 180, 181, 182, 183, 185, 187, 188, 191, 195, 199, 2, 21, 300, 306, 316, 328, 329, 330, 370, 396, 402, 404, 405, 407, 409, 427, 456, 494, 495, 505, 524, 538, 546, 582, 60, 62, 64, 76, 77, 79, CA1, D1, D10, D11, D2, D3, D5, D8, D9, G, G1, G10, G11, G3, G4, G6, G8, L1, L12, L14, L15, L16, L18, L19, L20, L21, L22, L23, L24, L26, L27, L28, L29, L3, L30, L35, L37, L4, L41, L5, L6, L63, L7, L9, L94, M-B, PA.
9. La franja horaria es en el horario de 19:00 a 07:00 hs. Líneas 100, 102, 103, 104, 105, 106, 109, 110, 111, 112, 113, 115, 116, 117, 121, 124, 125, 126, 127, 128, 133, 135, 137, 14, 140, 141, 142, 143, 144, 145, 147, 148, 149, 150, 151, 155, 156, 157, 158, 161, 163, 164, 169, 17, 173, 174, 175, 180, 181, 182, 185, 186, 187, 188, 191, 192, 195, 199, 2, 21, 300, 306, 316, 328, 329, 330, 370, 396, 4, 402, 404, 405, 407, 409, 427, 456, 495, 505, 522, 524, 526, 538, 582, 60, 62, 64, 71, 76, 77, 79, CA1, D1, D10, D11, D2, D3, D5, D8, D9, G, G1, G10, G11, G3, G4, G6, G8, L1, L12, L13, L14, L15, L16, L17, L18, L19, L2, L20, L21, L22, L23, L24, L26, L27, L28, L29, L3, L35, L37, L4, L41, L5, L6, L63, L7, L9, L94, M-A, M-B, PA, PB.

10. La franja horaria es en el horario de 07:00 a 13:00 hs. Líneas 100, 103, 104, 106, 109, 110, 111, 113, 117, 125, 126, 127, 128, 133, 135, 137, 140, 142, 143, 144, 145, 147, 148, 149, 151, 155, 156, 157, 158, 161, 163, 169, 17, 174, 175, 180, 181, 186, 187, 188, 191, 192, 199, 2, 21, 300, 306, 316, 329, 330, 370, 396, 4, 402, 404, 405, 407, 409, 456, 494, 495, 505, 522, 524, 526, 538, 546, 582, 60, 64, 71, 77, 79, D1, D10, D11, D5, D8, D9, G1, G10, G11, G3, G4, G8, L1, L12, L13, L14, L15, L16, L17, L18, L19, L2, L20, L23, L26, L27, L3, L35, L37, L4, L5, L63, L7, L9, L94, M-A, M-B, PA.

Apéndice C

Resultados obtenidos para el problema de relocalización de paradas de autobuses

Configuración	$\#P$	p_r	p_c	Configuración	$\#P$	p_r	p_c
#1	50	0.60	0.01	#15	100	0.75	0.01
#2	50	0.60	0.05	#16	100	0.90	0.01
#3	50	0.60	0.001	#17	100	0.90	0.05
#4	50	0.75	0.05	#18	100	0.90	0.001
#5	50	0.75	0.001	#19	150	0.60	0.01
#6	50	0.75	0.01	#20	150	0.60	0.05
#7	50	0.90	0.01	#21	150	0.60	0.001
#8	50	0.90	0.05	#22	150	0.75	0.05
#9	50	0.90	0.001	#23	150	0.75	0.001
#10	100	0.60	0.01	#24	150	0.75	0.01
#11	100	0.60	0.05	#25	150	0.90	0.01
#12	100	0.60	0.001	#26	150	0.90	0.05
#13	100	0.75	0.05	#27	150	0.90	0.001
#14	100	0.75	0.001				

Tabla C.1: Configuración de parámetros utilizados.

Parámetros	*Instancia 1*	*Instancia 2*	*Instancia 3*	Parámetros	*Instancia 1*	*Instancia 2*	*Instancia 3*
#1	0.85±0.08 (1.0)	0.79±0.10 (1.0)	0.90±0.08 (1.0)	#15	0.75±0.09 (1.0)	0.80±0.08 (1.0)	0.95±0.06 (1.0)
#2	0.80±0.11 (1.0)	0.50±0.28 (1.0)	0.85±0.10 (1.0)	#16	0.82±0.11 (1.0)	0.70±0.12 (1.0)	1.00±0.32 (1.0)
#3	0.68±0.11 (1.0)	0.70±0.13 (1.0)	0.72±0.10 (1.0)	#17	1.00±0.27 (1.0)	1.00±0.17 (1.0)	0.63±0.18 (1.0)
#4	0.87±0.10 (1.0)	0.68±0.12 (1.0)	1.00±0.22 (1.0)	#18	0.77±0.08 (1.0)	0.85±0.07 (1.0)	1.00±0.10 (1.0)
#5	0.78±0.11 (1.0)	0.66±0.15 (1.0)	0.71±0.11 (1.0)	#19	0.65±0.11 (1.0)	0.74±0.11 (1.0)	0.81±0.13 (1.0)
#6	0.85±0.11 (1.0)	0.88±0.09 (1.0)	0.96±0.11 (1.0)	#20	0.79±0.10 (1.0)	0.93±0.10 (1.0)	0.86±0.10 (1.0)
#7	0.89±0.14 (1.0)	0.73±0.10 (1.0)	0.72±0.17 (1.0)	#21	0.88±0.08 (1.0)	0.87±0.08 (1.0)	0.84±0.07 (1.0)
#8	0.79±0.16 (1.0)	0.75±0.12 (1.0)	0.74±0.10 (1.0)	#22	0.84±0.06 (1.0)	0.83±0.10 (1.0)	0.75±0.12 (1.0)
#9	0.80±0.09 (1.0)	0.66±0.140 (1.0)	0.79±0.12 (1.0)	#23	0.93±0.67 (1.0)	0.93±0.10 (1.0)	0.80±0.13 (1.0)
#10	0.73±0.10 (1.0)	0.74±0.10 (1.0)	0.86±0.13 (1.0)	#24	0.81±0.07 (1.0)	0.79±0.08 (1.0)	0.85±0.09 (1.0)
#11	0.70±0.10 (1.0)	1.00±0.08 (1.0)	0.77±0.10 (1.0)	#25	0.68±0.14 (1.0)	0.83±0.15 (1.0)	0.63±0.18 (1.0)
#12	0.78±0.11 (1.0)	0.91±0.07 (1.0)	0.98±0.07 (1.0)	#26	0.55±0.13 (1.0)	0.44±0.15 (1.0)	0.30±0.18 (1.0)
#13	0.87±0.08 (1.0)	0.89±0.05 (1.0)	0.75±0.14 (1.0)	#27	0.74±0.08 (1.0)	0.81±0.08 (1.0)	0.94±0.08 (1.0)
#14	0.89±0.12 (1.0)	0.85±0.08 (1.0)	0.74±0.11 (1.0)				

Tabla C.2: Resultados obtenidos para el hipervolumen relativo en la configuración paramétrica.

	DG	spacing	spread	HR
ch#1	0.66±0.53 (0.0)	0.09±0.05 (0.01)	0.97±0.14 (0.72)	1.0±0.2 (1.0)
ch#2	0.02±0.01 (0.0)	0.01±0.01 (0.0)	0.71±0.08 (0.62)	1.0±0.2 (1.0)
ch#3	0.66±0.53 (0.0)	0.09±0.05 (0.01)	0.97±0.14 (0.72)	1.0±0.2 (1.0)
ch#4	0.02±0.01 (0.0)	0.04±0.10 (0.0)	1.02±0.17 (0.75)	0.9±0.1 (1.0)
ch#5	0.01±0.01 (0.0)	0.04±0.05 (0.0)	1.04±0.13 (0.83)	0.8±0.1 (1.0)
ch#6	0.01±0.0 (0.0)	0.05±0.04 (0.0)	1.03±0.16 (0.80)	0.9±0.1 (1.0)
ch#7	0.01±0.0 (0.0)	0.03±0.02 (0.0)	0.99±0.08 (0.84)	0.8±0.1 (1.0)
ch#8	0.02±0.01 (0.01)	0.05±0.10 (0.0)	1.11±0.13 (0.94)	0.8±0.1 (1.0)
ch#9	0.02±0.01 (0.01)	0.04±0.01 (0.0)	1.06±0.12 (0.86)	0.8±0.2 (1.0)
ch#10	0.02±0.01 (0.0)	0.03±0.02 (0.0)	1.0±0.11 (0.84)	1.0±0.2 (1.0)
ch#11	0.07±0.04 (0.0)	0.81±0.07 (0.0)	1.31±0.18 (0.85)	1.0±0.2 (1.0)

Tabla C.3: Métricas multiobjetivo para el AE implementado sobre instancias chicas.

	DG	spacing	spread	HR
me#1	0.08±0.02 (0.03)	0.07±0.05 (0.03)	0.86±0.07 (0.74)	1.0±0.1 (1.0)
me#2	0.07±0.01 (0.03)	0.05±0.01 (0.03)	0.82±0.05 (0.71)	1.0±0.1 (1.0)
me#3	0.06±0.02 (0.02)	0.83±0.06 (0.02)	0.83±0.06 (0.02)	1.0±0.1 (1.0)
me#4	0.05±0.01 (0.03)	0.04±0.02 (0.02)	0.88±0.05 (0.74)	1.0±0.1 (1.0)
me#5	0.07±0.02 (0.02)	0.06±0.03 (0.03)	0.88±0.06 (0.71)	1.0±0.1 (1.0)
me#6	0.76±0.2 (0.0)	1.10±0.20 (0.0)	0.88±0.06 (0.71)	1.0±0.1 (1.0)
me#7	0.06±0.02 (0.01)	0.07±0.04 (0.02)	0.91±0.07 (0.73)	1.0±0.1 (1.0)
me#8	0.06±0.01 (0.03)	0.07±0.04 (0.02)	0.88±0.07 (0.75)	1.0±0.1 (1.0)
me#9	0.07±0.02 (0.03)	0.05±0.02 (0.02)	0.81±0.06 (0.67)	1.0±0.1 (1.0)

Tabla C.4: Métricas multiobjetivo para el AE implementado sobre instancias medianas.

	DG	spacing	spread	HR
gr#1	0.08±0.02 (0.03)	0.05±0.02 (0.02)	0.83±0.07 (0.70)	1.0±0.1 (1.0)
gr#2	0.09±0.02 (0.02)	0.06±0.01 (0.03)	0.86±0.07 (0.68)	1.0±0.1 (1.0)
gr#3	0.08±0.02 (0.03)	0.07±0.03 (0.02)	0.91±0.08 (0.76)	1.0±0.1 (1.0)
gr#4	0.12±0.02 (0.05)	0.08±0.03 (0.03)	0.90±0.06 (0.80)	1.0±0.1 (1.0)
gr#5	0.08±0.02 (0.02)	0.07±0.03 (0.02)	0.74±0.09 (0.74)	1.0±0.1 (1.0)
gr#6	0.09±0.01 (0.05)	0.06±0.02 (0.02)	0.89±0.07 (0.72)	1.0±0.1 (1.0)
gr#7	1.19±0.28 (0.28)	2.12±1.50 (0.0)	0.89±0.07 (0.72)	1.0±0.3 (1.0)
gr#8	0.08±0.02 (0.02)	0.07±0.02 (0.02)	0.90±0.05 (0.75)	1.0±0.1 (1.0)
gr#9	0.09±0.02 (0.02)	0.07±0.02 (0.03)	0.90±0.06 (0.77)	1.0±0.1 (1.0)
gr#10	0.08±0.02 (0.01)	0.05±0.02 (0.02)	0.82±0.07 (0.65)	1.0±0.1 (1.0)

Tabla C.5: Métricas multiobjetivo para el AE implementado sobre instancias grandes.

ANEXOS

Anexo A

Publicaciones

En este anexo se encuentra la publicación *"Distributed Big Data analysis for mobility estimation in Intelligent Transportation System"*. El artículo fue presentado en el congreso internacional *"High Performance Computing Latin America"*, celebrado del 29 de agosto al 1° de setiembre de 2016 en Ciudad de Mexico, Mexico. El artículo se encuentra publicado en *Communications in Computer and Information Science*, volumen 697, Springer, Cham, páginas 146-160.

Distributed Big Data analysis for mobility estimation in Intelligent Transportation Systems

Enzo Fabbiani[1], Pablo Vidal[2], Renzo Massobrio[1], and Sergio Nesmachnow[1]

[1] Universidad de la República, Herrera y Reissig 565, Montevideo 11300, Uruguay
{enzo.fabbiani, renzom, sergion}@fing.edu.uy
[2] CONICET–Universidad Nacional de la Patagonia Austral
Acceso Norte, Ruta N° 3, 9011, Caleta Olivia, Argentina
pjvidal@uaco.unpa.edu.ar

Abstract. This article describes the application of distributed computing techniques for the analysis of big data information from Intelligent Transportation Systems. Extracting useful mobility information from large volumes of data is crucial to improve decision-making processes in smart cities. We study the problem of estimating demand and origin-destination matrices based on ticket sales and location of buses in the city. We introduce a framework for mobility analysis in smart cities, including two algorithms for the efficient processing of large mobility data from the public transportation in Montevideo, Uruguay. Parallel versions are proposed for distributed memory (e.g., cluster, grid, cloud) infrastructures and a cluster implementation is presented. The experimental analysis performed using realistic datasets demonstrates that significant speedup values, up to 16.41, are obtained.

Keywords: Distributed computing; Big data; mobility analysis; Intelligent Transportation Systems

1 Introduction

Nowadays, many complex activities are developed in modern cities, which impose serious challenges to the mobility of citizens [7]. The main mobility issues in dense urban areas are related to public transport systems that are not capable of fulfilling the growing demand for transportation. In order to implement innovative solutions that address this issue, it is necessary to have access to updated information about the mobility of the citizens [2]. In most cities, the information available from public administrations is scarce and outdated, due to the lack of financial and human resources assigned to gathering and managing mobility data. In other cases, data are gathered but are not used for improving the mobility or optimizing public/vehicular transportation. In this scenario, developing improved decision-making processes related to urban mobility becomes mandatory. New smart city technologies are very helpful to offer high quality solutions for this kind of mobility problems.

The paradigm of smart cities proposes taking advantage of information and communication technologies to improve the quality and efficiency of urban services [4]. Intelligent Transportation Systems (ITS) are a key component of smart cities. ITS are defined as those systems integrating synergistic technologies, computational intelligence, and engineering concepts to develop and improve transportation. They are aimed at providing innovative services for transport and traffic management, with the main goals of improving transportation safety and mobility, and also enhancing productivity [18]. ITS allow gathering several data regarding transportation and mobility in the cities [5]. In big urban areas, they generate huge volumes of data that can be processed to extract valuable information about the mobility of citizens.

This article presents a framework to study mobility in the context of smart cities. In order to solve mobility and urban transportation optimization problems it is imperative to understand the mobility patterns of citizens and the demand distribution for public transport. This information is often represented using matrices: *i*) origin-destination (OD) matrices, indicate the amount of people moving from each point in the city in a given period of time [1]; *ii*) demand matrices, measure the number of bus tickets sold between each location in the city. These matrices are often built using data from surveys performed in-situ to passengers and drivers. However, surveys offer only a partial vision of the mobility patterns in the city, are expensive, and need to be performed regularly to get updated information. A different approach to build demand and OD matrices is based on processing real data from ITS, including tickets sold with and without smart cards, GPS data from buses, etc.

We introduce a specific methodology for generating OD and demand matrices using data from ITS to study mobility in smart cities. Taking into account the large computing time demanded when dealing with huge volumes of data, we propose applying distributed computing techniques to speed up the processing. A data-parallel approach is devised to process large datasets on distributed memory parallel architectures (e.g., cluster, grid, and cloud systems). Two specific algorithms are presented, based on real data from the ITS in Montevideo, Uruguay. The proposed approach can easily be extended to any other ITS-related information and scenarios.

The main contributions of the research reported in this article are: *i*) we introduce a methodology for OD matrix estimation using smart card information; *ii*) we design and implement specific algorithms for processing big data using distributed computing techniques; and *iii*) we report an experimental analysis which demonstrates that significant speedup values are obtained, allowing the efficient processing of large datasets.

The article is organized as follows. Section 2 introduces the problem of Big Data analysis for ITS. The proposal of applying distributed computing techniques to accelerate the processing of large volumes of ITS data is described in Section 3. The experimental evaluation of the proposed algorithms is reported in Section 5. Finally, Section 6 presents the conclusions of the research and formulates the main lines for future work.

2 Big Data analysis for Intelligent Transportation Systems

This section describes the problem of Big Data analysis for building demand and origin-destination matrices. A review of related work is also presented.

2.1 Problem description

The main challenge faced when generating demand and OD matrices using data from tickets sales is that passengers validate their smart cards when they board but not when they alight the bus. Therefore, while the origin of each trip is known with certainty, it is necessary to estimate the destination. Furthermore, some passengers do not use smart cards to pay for their ticket. Therefore, there are sale records which do not provide information to be used to track several trips made by a single passenger. Specific big data processing algorithms must be designed and implemented for each case.

In this article we focus on generating demand and OD matrices for the city of Montevideo, Uruguay. The city government in Montevideo introduced in 2010 an urban mobility plan to redesign and modernize urban transport in the city [9]. Under this plan, the Metropolitan Transport System (Sistema de Transporte Metropolitano, STM) was created, with the goal of integrating the different components of the public transportation system together. One of the first improvements in STM was to include GPS devices on buses and allow passengers to pay for tickets using a smart card (STM card). Additionally, the complex system of fares was simplified to allow only two different type of tickets: i) "one hour" tickets, allowing up to 1 transfer within an hour of taking the first bus; ii) "two hours" tickets, allowing unlimited transfers within 2 hours from the moment the ticket is purchased. However, it is not compulsory to use the STM card to buy bus tickets. Passengers may pay with cash directly to the driver. In this case, the ticket is only valid for that trip and no transfers are allowed.

The bus companies that operate in Montevideo are obliged to send bus location and ticket sale data daily to the city authorities. The bus network in Montevideo is quite complex, consisting of 1383 bus lines and 4718 bus stops. In this article we consider the complete dataset of ticket sales and bus location for 2015, comprising nearly 200 GB of data. Bus location data contain information about the position of each bus, sampled every 10 to 30 seconds. Each location record holds the following information:

- *lineID*, the unique bus line identifier;
- *tripID*, the unique trip identifier for each single trip for a given lineID;
- *latitude* and *longitude*,
- *vehicle speed*;
- *timestamp* of the location; and
- *stopID*, the identifier for the nearest bus stop to the current bus location.

Ticket sales data contain information about sales made with and without STM cards. Each sale record has the following fields:

– *tripID*, as in location data;
– *latitude* and *longitude*,
– *stopID*, as in location data;
– *number of passengers*, since it is possible to buy tickets for multiple passengers at once; and
– *timestamp* of the sale.

Additionally, tickets payed with STM cards have the following fields: unique STM card identifier (cardID) which is hashed for privacy purposes, number of transactions made with that STM card (transactionID), and the last payed transaction (payedID). This allows identifying when a passenger transfers between buses, since transactionID will increment while payedID will remain unchanged. The number of transfers is equal to transactionID−payedID.

2.2 Related work

Many articles in the related literature have proposed applying statistical analysis for estimating OD matrices and computing several other relevant statistics for ITS. Some approaches applying parallel and distributed computing techniques have also been proposed recently. A review of the main related works is presented next.

A detailed literature review on the use of smart cards in ITS was presented by Pelletier et al. [14]. The review covers the hardware and software needed for the deployment of smart card payment solutions in urban transportation systems as well as the privacy and legal issues that arise when dealing with smart card data. Additionally, the authors identify the main uses for these data, including: long-term planning, service adjustments and performance indicators of the transportation systems. Finally, examples of smart card data usage around the world are reviewed.

Trépanier et al. [20] presented a model to estimate the destination for passengers boarding buses with smart cards, following a database programming approach. Two hypotheses are considered, which are commonly used in many related works: *i*) the origin of a new trip is the destination of the previous one; *ii*) at the end of the day users return to the origin of their first trip of the day. Based on these assumptions, the authors propose a method to follow the chain of trips of each user in the system. Those trips for which chaining is not possible (e.g., only one trip in the day for a particular user) are compared with all other trips of the month for the same user, in order to find similar trips with known destination. The experimental evaluation was conducted using real information from a transit authority in Gatineau, Quebec. Two datasets were used, with 378,260 trips from July 2003 and 771,239 trips from October 2003. Results showed that a destination estimation was possible for 66% of the trips. Most trips where destination could not be estimated take place during off-peak hours, where more atypical and non-regular trips are performed. Considering only peak hours, the percentage of trips with their destination estimated improves to 80%. However, the estimation accuracy could not be assessed due to lack of a second source of data (e.g., automatic passenger count) for comparison.

Wang et al. [21] proposed using a trip-chaining method to infer bus passenger OD from smart card transactions and Automatic Vehicle Location (AVL) data from London, United Kingdom. In the studied scenario, authors needed to estimate both origin and destination of trips. Origins are accurately estimated by searching for the timestamp of each smart card transaction in the AVL records to determine the bus stop of each trip. To estimate destinations, the authors used a similar methodology to that presented by Trépanier et al. [20], chaining trips when possible to infer destinations. Results were compared against passenger survey data from Transport for London, performed every five to seven years for each bus route and including the number of people boarding and alighting at each bus stop. The analysis show that origins can be estimated for more than 90% of the trips while origin and destinations can be estimated for 57% of all trips. When compared to the survey data, the difference on the estimated destinations were below 4%. Finally, two practical applications of the results are presented. The first one consists of studying the daily load/flow variation to identify locations along each bus route where passenger load is high, as well as underutilized route segments. The second application consists of an transfer time analysis, evaluating the average time that users need to wait for transferring between buses, based on the alighting stop and the AVL data.

Munizaga et al. [12] presented a similar approach to estimate OD matrices in the multimodal transportation system of Santiago, Chile, where passengers can use their smart cards to pay for tickets at metros, buses or bus stations.

Several proposals have applied distributed computing approaches to process large volumes of traffic data, but few works have dealt with the estimation of demand or OD matrices.

Pioneering works on this topic applied distributing computing to gather traffic data. Sun [17] proposed a client-server model developed in CORBA for collecting traffic counts in real time, to be used for dynamic origin/destination demand estimation. The proposed solution included a CORBA client to extract data from the traffic network, and a CORBA server for storing data in a centralized repository. All the information is processed to be later used in Dynamic Traffic Assignment strategies for the traffic network studied, for the estimation of dynamic OD matrices applying a bi-level optimization framework.

Toole et al. [19] propose combining data from many sources (call records from mobile phones, census, and surveys) to infer OD matrices. The authors combine several existing algorithms to generate OD matrices, assign trips to specific routes, and to compute quality metrics on road usage. Furthermore, a web application is introduced to give simple visualizations of the computed information. The authors mention that computations are performed in parallel, but no parallel model is described and no performance metrics are reported.

Also using mobile phone data, Mellegård [11] proposed a Hadoop implementation to generate OD matrices while keeping users' privacy. However, the experimental analysis is done on synthetic data due to the difficulties on getting real data from mobile operators. Furthermore, no performance metrics are reported, so the advantages of the Hadoop implementation are not clear.

Huang et al. [8] proposed a methodology for offline/online calibration of Dynamic Traffic Assignment systems via distributed gradient calculations. An adaptive network decomposition framework is introduced for parallel computation of traffic network metrics and for parallel simulation, in order to accelerate the computations. Parallel OD demand estimation is proposed as a line for future work, in order to deal with large-scale traffic networks with huge number of OD pairs and sensors.

Our recent work [10] presented a preliminary analysis on using distributed computing techniques to process GPS data from buses. We introduced a Map-Reduce approach for processing historical data to study relevant metrics to assess the quality-of-service of the transportation systems in Montevideo, Uruguay. We used the strategy to compute the average speed of buses and to identify troublesome locations, according to the delay and deviation of the times to reach each bus stop. The parallel implementation scaled properly when processing large volumes of input data.

In the present article, we extend our preliminary approach [10] to solve a more complex and computing intensive problem: the estimation of demand and OD matrices for public transportation. Up to our knowledge, this approach has not been previously proposed in the related literature.

3 Mobility estimation from ITS smart card data

This section presents the proposed methodology for estimating demand and OD matrices taking into account the two kind of transfer trips in Montevideo (explained in Section 2). We introduce two models for estimation: the first one for one-way transfer trips and the second one for multiple trips. We present a description of our sequential algorithm for estimating demand and OD matrices. Finally, the parallel implementation with all its components is described.

3.1 Models for demand and OD estimation

The model used for estimating OD matrices is based on reconstructing the trip sequence for passengers that use a smart card, following a similar approach to that applied in the related literature [20, 21, 12]. We assume that each smart card corresponds to a single passenger, so we use the terms *card* and *user* in an indistinct manner. The proposed approach is based on processing each trip, retrieving the bus stop where the trip started, and identifying/estimating the stop where the passenger alighted the bus from the information available.

We identify two different ways of estimating destinations from the data used: *transfer trips* and *direct trips*. The main details for each case are presented next.

Transfer trips. In a transfer trip, passengers pay for their ticket when boarding the first bus by using a smart card identified by its cardID. Later, they can take one or more buses within the time limits permitted by the ticket, as explained in Section 2.1. For each ticket sold, the number of transactions (transactionID)

made with that cardID and the last payed transaction (payedID) are recorded. This allows detecting whether a smart card record corresponds to a new trip (payedID is equal to transactionID) or to a transfer between buses (transactionID is higher than payedID). We assume that passengers avoid excessive walking in transfers; we consider that a passenger finishes its first leg at the nearest bus stop to the bus stop where he boards the second leg, and so on. The boarding bus stop for the second leg is recorded in the system, thus we estimate the alighting point from the first bus by looking for the closest bus stop corresponding to that line.

A transfer example is presented in Figure 1. At 07:42 the passenger takes bus number 1 (green line) at bus stop 12. At this point, we have no information about the destination of the passenger. However, at 08:12 the passenger boards bus number 2 (blue line) by using a transfer, without paying a new ticket. Therefore, we can confidently estimate that the passenger alighted from bus number 1 at bus stop number 1-5 which is the closest bus stop to bus stop 23, where the passenger does the transfer. We have no information about the destination of the second trip. This issue is addressed with the direct trip estimation, described next.

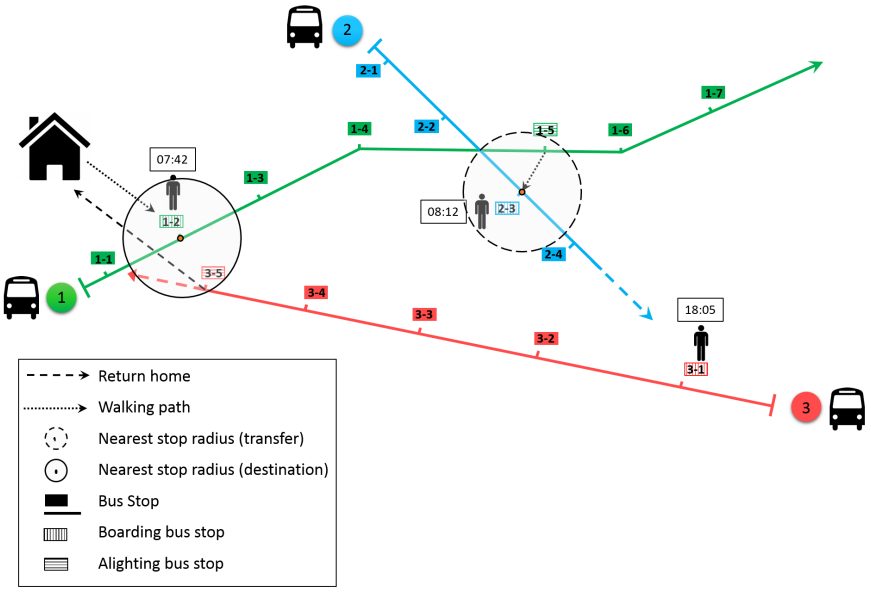

Fig. 1. Demand and OD estimation for transfer and direct trips.

Direct trips. We consider direct trips as those that have no bus transfers. We also consider the last leg of a trip with one or more transfers as a direct trip. In both cases, the difficulty lies in accurately estimating a destination point for these trips.

To estimate the destination points we consider two assumptions, which are commonly used in the related literature: *i*) passengers start a new trip at a bus stop which is close to the destination of their previous trip; *ii*) at the end of the day, passengers return to the bus stop where they boarded the first trip on the same day.

In order to estimate destinations it is necessary to chain the trips made by each passenger on a single day. A preliminary study performed on the sales dataset showed that the best option is to consider each day starting at 04:00, since the lowest number of tickets are sold at that time. This allows considering passengers with different travel patterns, such as those who commute to work during the day and those who work at night.

The model for chaining direct trips of a specific passenger works as follows. We iterate through all the trips done in a 24 hour period (from 04:00 to 04:00 on the following day). For each new trip, we try to estimate the alighting point by looking for a bus stop located in a predefined range from the boarding bus stop of the previous trip.

In the example shown in Fig. 1, the passenger takes bus number 3 (red line) at 18:05, to return home. In order to estimate the destination of this trip, we look for the closest bus stop of bus number 3 located within a given search radius from stop number 12, which is the origin of the previous trip. In the example, stop number 35 is the only stop within that radius, so it is chosen as the alighting point for the trip. When no bus stop is found on that radius, the procedure is repeated using a larger radius (two times the original one) to search for bus stops. If no bus stop is found using the larger radius, the origin of the trip is recorded, in order to report the number of unassigned destinations.

3.2 Algorithm for demand and OD estimation

We propose a specific methodology for reconstructing the trip sequence for passengers, by estimating the destination points from the information available. The algorithm for estimating trip destinations is described in the flowchart in Fig. 2.

Three phases are identified in the proposed algorithm, which are relevant for building the estimated demand and OD matrices:

Pre-processing. The pre-processing phase prepares the data, filtering those records with incoherent information and classifying records by month per passenger. The algorithm receives as input an unstructured dataset containing raw GPS positions and ticket sales data. Initially, the algorithm discards those sales records that have invalid GPS coordinates; which are not processed for the demand and OD matrices estimation. A sale record has an invalid location when its coordinates are not within the route of the bus corresponding to the sale, with a tolerance of 50 meters.

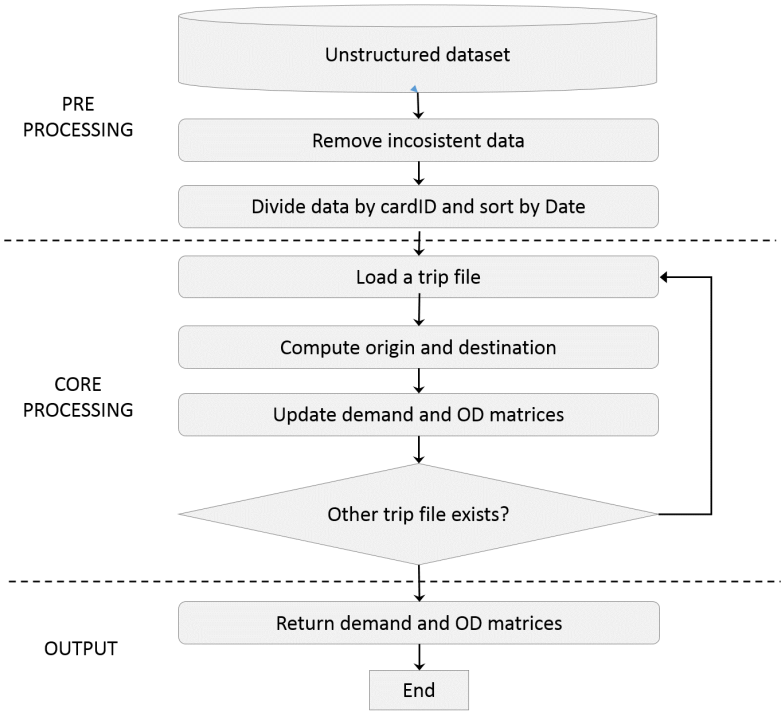

Fig. 2. Flowchart of the algorithm for estimating trip destinations.

Finally, trip records with consistent location information are separated into different files, according to their cardID and then ordered according to their date field. This allows processing the trips of each passenger independently.

Core processing. In this phase the sales data are processed in order to generate demand and OD matrices. Data are iteratively processed: for each passenger, trips are analyzed considering 24 hour periods starting and finishing at 04.00. First, the origin of the trip is recorded. In order to estimate the destination, the models defined in Section 3.1 are applied, depending on whether the trip corresponds to a transfer or to a direct trip. Once the origin and the destination are computed, the corresponding values are updated in the demand and OD matrices. The process is repeated until all trip records are processed. In our study, we consider a distance of 500m for the search radius used when estimating destination of direct trips, as described in Section 3.1.

Output After all records are computed the demand and OD matrices are returned.

4 A parallel algorithm for demand and OD matrices estimation

The capability of a traditional sequential algorithm for the estimation of demand and OD matrices is limited by the computational capacity of a single computing element (*node*). High performance computing architectures based on distributed parallel processing principles can achieve a large computational efficiency as well as high scalability for solving complex problems [6].

In this section, we present the parallel model for processing a dataset consisting of many trip files and estimating the demand and OD matrices using a parallel/distributed system.

4.1 Parallel Model

Processing large volumes of data is needed to accurately estimate demand and OD matrices from ticket sales and bus location information. Initial experiments on a reduced portion of the dataset suggest that processing only one month of sales data demands over 18 days of computational time, when using a sequential algorithm in a regular desktop computer (Core i5 x2, 6 GB of RAM, Ubuntu 14.04). Therefore, we propose to run the estimation algorithms in parallel, making use of several computing units. The basic idea of the proposed parallel algorithm is to apply a data-parallel approach, dividing the dataset of sales and GPS records in chunks, following the Bag-of-Tasks (BoT) paradigm [3]. In our case, the BoT corresponds to a set of user trip files. Since each set of trip files are independent, they can be assigned to different compute nodes (slaves) for processing. Using a master-slave architecture seems appropriate since the slave processes do not need to share information with each other.

Fig. 3 shows the proposed master-slave model for processing trip data. Initially, the master collects the data to be processed and filters inconsistent records. Next, the master sends the BoT to each slave in the slave pool in order to perform the assigned computation task. Then, each slave node runs the designated estimation procedure. Finally, the master receives the partial results and store them to join and create the final demand and OD matrices.

Two variants of the proposed algorithm were implemented, one for each of the two different estimation procedures presented in Section 3.1. Both variants follow the same general approach which is specified in Section 4.2.

4.2 Implementation details

The implemented algorithms were designed using Python 2.7.5. The cross-platform open-source geographic information system QGIS [16] was used to manage geographic information corresponding to bus location and bus stops data.

We applied `dispy` [15] for creating and distributing parallel tasks among several computer nodes. `dispy` is a Python framework that allows executing parallel

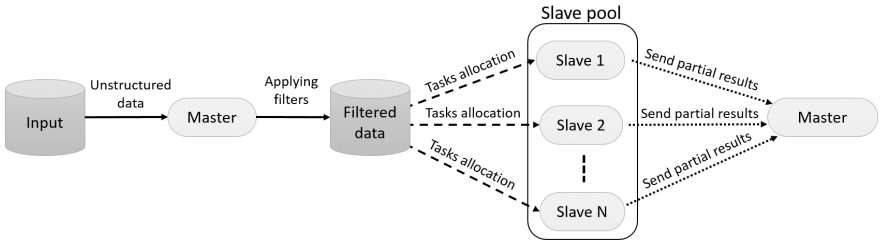

Fig. 3. Diagram of the proposed parallel model for demand and OD matrices estimation.

processes, supporting many different distributed computing infrastructures. The main features of the framework include tasks distribution, load balancing, and fault recovery. The `dispy` framework provides an API for the user to define clusters and schedule jobs to execute on those clusters. Creating a cluster in `dispy` consists of packaging computation fragments (code and data) and specifying parameters that control how the computations are executed, such as which nodes can execute each computation. Listing 1 presents the main methods used with `dispy` to create jobs and assign them to slave nodes in order to estimate demand and OD matrices.

Listing 1 `dispy` script for job creation and distribution.

```
# function 'estimationOD'
def estimationOD(tripFile):
    #code for estimating origin-destination
    return pairsOD
#main
if __name__ == '__main__':
    # modules imported, are not available in job computations
    import dispy
    cluster = dispy.JobCluster(estimationOD,depends=[settings,
                               data, visual,...],...more parameters)
    jobs = []
    #Create tasks related with each trip file
    for tripFile in os.listdir(settings.path_to_trip_dataset):
        job = cluster.submit(tripFile)
        jobs.append(job)
        for job in jobs: #Execute jobs into the nodes availables
            job()# waits for job to finish and returns results
            pairsOD = job.result
    for pairs in pairsOD:
        #OD matrix is reconstructed with partial results
        #stored in variable 'pairsOD'
```

A list of parameters are needed to set a `dispy` cluster. First of all, the program to execute in each node must be indicated (in our case, the O-D matrix estimation procedure). In addition, the list of nodes available to execute the jobs, and a list of dependencies needed for computation must be specified (in our application the only dependency is the availability of the QGIS software).

Once a cluster is created, jobs can be scheduled to execute at a certain node. `dispy` will execute the job on an available processor in the defined cluster. When a submitted job is called, it returns that job's execution result, waiting until the job is finished if it has not finished yet. After a job is finished the information about the pairs origin-destination found is used to build the OD matrices.

Each slave keeps track of the index of the last file or line processed. Therefore, in case of a system failure it is possible to resume the execution from the last processed record, without the need of starting the process from the beginning.

In our approach, the master creates a set of BoT where each task corresponds to all the trip records of a single passenger. Then, each BoT is distributed using `dispy` across the different slaves to execute the estimation algorithms. It is important to choose the amount of passengers' trip records to assign to each slave in order to optimize the execution time, avoiding costly communications between the slaves and the master. This parameter is configured in the experimental analysis presented in Section 5. Finally, the master node distributes tasks to slaves on demand, and obtains the results computed by each slave to gather them to return the final solution

5 Experimental analysis

This section describes the experimental analysis performed to assess the efficiency of the parallel algorithms developed. The computational platform used and the problem instances are detailed. Finally, the computational efficiency results are reported and commented.

5.1 Computational platform

The experimental analysis was performed on an AMD Opteron 6172 Magny Cours processors with 24 cores at 2.26 GHz, 72 GB RAM, with CentOS Linux 5.2 operating system from Cluster FING, the high performance computing infrastructure at Facultad de Ingeniería, Universidad de la República, Uruguay [13].

5.2 Problem instance and metrics

Problem instance. For the experimental analysis, the complete dataset corresponding to the ITS in Montevideo for January 2015 was processed, including ticket sales and bus location data. This dataset holds the mobility information for over half a million smart cards (corresponding to more than 13 million trips).

Computational efficiency metrics. In order to evaluate the computational efficiency of the proposed parallel algorithm we evaluate the *execution time* and the *speedup*. If we denote T_m the execution time for an algorithm using m processors, then the speedup is the ratio between the (larger) execution time on one processor T_1 and the (smaller) execution time on m processors T_m. This ratio value is presented in Eq. 1

$$S_m = \frac{T_1}{T_m} \tag{1}$$

5.3 Results and discussion

In the proposed master-slave parallel model it is necessary to define the size of the BoT assigned to each slave to compute, in order to have an appropriate load balance and avoid excessive communication between the slaves and the master. Several executions were performed varying the size of the BoT as well as the number of cores used. The experimental results are reported on Table 1. The number of cores (#cores) and the size of the BoT (#BoT) used in each experiment are indicated. Then, for each combination of these values, the best (i.e., minimum), average, and standard deviation of execution time and speedup values are reported for both direct and transfer trips. Execution times are reported in minutes and the results correspond to 5 independent executions of the algorithm using each configuration of #cores and #BoT.

#cores	*#BoT*	*direct trips*		*transfer trips*	
		avg. time$\pm\sigma$ (best)	speedup	avg. time$\pm\sigma$ (best)	speedup
1	1	25920	-	30240	-
16	5000	2092.1±3.4 (2089.6)	12.40	2648.9±3.2 (2645.5)	11.43
16	10000	2372.4±1.8 (2371.1)	10.92	3068.8±3.5 (3063.2)	9.87
24	5000	1582.7±2.4 (1579.4)	16.41	2371.1±2.5 (2368.1)	12.76
24	10000	1858.2±2.1 (1855.9)	13.96	2617.9±3.3 (2614.3)	11.56

Table 1. Execution time results and performance analysis.

The experimental results obtained suggest that the parallel approach is an appropriate strategy for significantly improving the efficiency of the data processing for demand and O-D matrices estimation. Promising speedup values were obtained, up to **16.41** for the direct trips processing and using a BoT of 5000 trips and executing in 24 nodes. These results confirm that the proposed master/slave parallel model allows improving the execution time of the computational tasks by taking advantage of multiple computing nodes.

Furthermore, the computational efficiency results indicate that the size of the BoT (i.e., the amount of passengers' trip data given to each slave to process at once) has a significant impact on the overall execution time of the algorithm. Execution times were reduced when using the smallest size for the BoT (5000). Further experiments should be performed to assess if using a smaller size for the BoT is still more efficient, and to determine the tradeoff value before the communications between the slaves and the master become more expensive and have a negative impact on the execution time.

Using 24 cores and tasks with the trip data corresponding to 5000 passengers, the proposed strategy allows improving in up to 54.4% the efficiency when compared to using 12 cores and a BoT size of 5000, and up to 57.9% against a sequential algorithm running on a single computing node. This efficiency allows processing the full information of GPS and trip data for one year (more than 130 GB) in 33 days, a significant improvement over the 468 days demanded by a sequential algorithm.

6 Conclusions and future work

In this work, we proposed and implemented an efficient estimation method for obtaining demand and origin-destination matrices from real smartcard data of bus ticket sales. The proposed procedure demands estimating the alighting stops, since passengers do not validate the smartcard when getting off the bus. Two different approaches are proposed depending on whether the trip record correspond to a direct trip or to a bus transfer. For the estimations we considered similar assumptions to other works in the related literature.

Due to the large volume of data to be processed, we designed and implemented a parallel version of the estimation algorithm, following the master/slave parallel model. When compared to a sequential algorithm, the proposed parallel model reduces execution time from 56120 to 3954 minutes and achieves speed up values of 16.41 when using 24 cores in the best case.

We identify three main lines of future work: i) validate the computed demand and OD matrices using other sources of data, such as surveys; ii) incorporate machine learning techniques to infer destinations with high accuracy, for example by identifying recurrent destinations of a single passenger; iii) take advantage of the computed mobility data to address optimization problem that arise in most modern ITS, such as bus route design, bus stop location, bus timetabling, etc.

References

1. Bell, M.: The estimation of an origin-destination matrix from traffic counts. Transportation Science 17(2), 198–217 (1983)
2. Chen, C., Ma, J., Susilo, Y., Liu, Y., Wang, M.: The promises of big data and small data for travel behavior (aka human mobility) analysis. Transportation Research Part C: Emerging Technologies 68, 285–299 (2016)

3. Cirne, W., Brasileiro, F., Sauvé, J., Andrade, N., Paranhos, D., Santos-Neto, E.: Grid computing for bag of tasks applications. In: Proceedings of the 3^{rd} IFIP Conference on E-Commerce, E-Business and EGovernment (2003)
4. Deakin, M., Waer, H.: From Intelligent to Smart Cities. Taylor & Francis (2012)
5. Figueiredo, L., Jesus, I., Machado, J.T., Ferreira, J., de Carvalho, J.M.: Towards the development of intelligent transportation systems. In: Intelligent transportation systems. vol. 88, pp. 1206–1211 (2001)
6. Foster, I.: Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1995)
7. Grava, S.: Urban Transportation Systems. McGraw-Hill Education (2002)
8. Huang, E., Antoniou, C., Lopes, J., Wen, Y., Ben-Akiva, M.: Accelerated on-line calibration of dynamic traffic assignment using distributed stochastic gradient approximation. In: 13^{th} International IEEE Conference on Intelligent Transportation Systems. pp. 1166–1171 (2010)
9. Intendencia de Montevideo: Plan de movilidad urbana: hacia un sistema de movilidad accesible, democrático y eficiente (2010)
10. Massobrio, R., Pias, A., Vázquez, N., Nesmachnow, S.: Map-Reduce for Processing GPS Data from Public Transport in Montevideo, Uruguay. In: 2^{nd} Argentinian Symposium on Big Data (AGRANDA) (2016)
11. Mellegård, E.: Obtaining Origin/Destination-matrices from cellular network data. Master's thesis (2011)
12. Munizaga, M.A., Palma, C.: Estimation of a disaggregate multimodal public transport origin–destination matrix from passive smartcard data from santiago, chile. Transportation Research Part C: Emerging Technologies 24, 9–18 (2012)
13. Nesmachnow, S.: Computación científica de alto desempeño en la Facultad de Ingeniería, Universidad de la República. Revista de la Asociación de Ingenieros del Uruguay 61, 12–15 (2010)
14. Pelletier, M.P., Trépanier, M., Morency, C.: Smart card data use in public transit: A literature review. Transportation Research Part C: Emerging Technologies 19(4), 557–568 (2011)
15. Pemmasani, G.: dispy: Distributed and parallel computing with/for python. http://dispy.sourceforge.net/, [Online; accessed July 2016]
16. QGIS Development Team: QGIS Geographic Information System. Open Source Geospatial Foundation (2009), http://qgis.osgeo.org, [Online; accessed July 2016]
17. Sun, C.: Dynamic origin/destination estimation using true section densities. Tech. Rep. UCB-ITS-PRR-2000-5, University of California, Berkeley
18. Sussman, J.: Perspectives on Intelligent Transportation Systems (ITS). Springer Science + Business Media (2005)
19. Toole, J.L., Colak, S., Sturt, B., Alexander, L.P., Evsukoff, A., González, M.C.: The path most traveled: Travel demand estimation using big data resources. Transportation Research Part C: Emerging Technologies 58, Part B, 162–177 (2015)
20. Trépanier, M., Tranchant, N., Chapleau, R.: Individual trip destination estimation in a transit smart card automated fare collection system. Journal of Intelligent Transportation Systems 11(1), 1–14 (2007)
21. Wang, W., Attanucci, J., Wilson, N.: Bus passenger origin-destination estimation and related analyses using automated data collection systems. Journal of Public Transportation 14(4), 131–150 (2011)