



UNIVERSIDAD DE CHILE  
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS  
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

MODELAMIENTO DE CAMBIOS EN LA DEMANDA DE TRANSPORTE PÚBLICO EN  
SANTIAGO DE CHILE USANDO TÉCNICAS DE APRENDIZAJE DE MÁQUINA E  
INTELIGENCIA ARTIFICIAL

MEMORIA PARA OPTAR AL TÍTULO DE  
INGENIERO/A CIVIL EN COMPUTACIÓN

SEBASTIÁN ALEJANDRO MONTEIRO PARADA

PROFESOR GUÍA:  
EDUARDO GRAELLS-GARRIDO

SANTIAGO DE CHILE

2025

# Capítulo 1

## Introducción

El sistema de transporte público en Santiago de Chile es un componente esencial para el funcionamiento de la ciudad. Cambios en su oferta, sean planificados o inesperados, pueden generar impactos significativos en la movilidad en los barrios cercanos, tanto a corto como a largo plazo.

Imagínese que usted es ingeniero de Red Metropolitana de Movilidad (el sistema de transporte público de Santiago), y le encomiendan la ardua tarea de agregar nuevos recorridos a la red. A priori es una tarea fácil si es que se tiene dinero infinito, pero al tener restricciones en términos de recursos, tiene que tomar la difícil decisión de priorizar ciertas áreas. Tomando eso en cuenta, usted crea un documento extenso con los nuevos recorridos y la frecuencia de ellos. Finalmente, su propuesta es aprobada y comienzan a circular nuevos buses y recorridos. Después de un par de meses le llegan los primeros reportes de demanda de la red. Para su sorpresa, nota que algunos de los recorridos que ya existían (anteriores a los que usted propuso), tuvieron una notable bajada de demanda, en otras palabras, es dinero perdido. Los buses ahora pasan mas vacíos y Red pierde millones de pesos en cargar a los vehículos eléctricos y pagándole a conductores que no recogen a casi ningún pasajero. Usted se encuentra en un dilema. Modificar la red genera cambios en otros recorridos, y pensar en cada cambio posible parece una tarea titánica. Usted se pregunta si hay alguna manera de generar datos de demanda sintéticos para analizar como cambia el resto de la red y su demanda ante sus cambios propuestos de manera mas sencilla.

Generar la demanda sintética de una red de transporte en base a cambios nuevos (una redistribución de la demanda) es el objetivo de esta memoria.

Una exploración bibliográfica sugiere que el campo de la predicción de la demanda usando técnicas de Inteligencia Artificial, ha crecido notablemente. Una exploración bibliográfica ayudada por el paper review de Torrepadula et al. [15], allanan el camino para entender cómo se ha abordado la predicción de la demanda de transporte público en distintas ciudades del mundo.

Torrepadula menciona que el problema de la demanda es de tipo pronóstico de series de tiempo. En ese sentido, se abren varias soluciones, como el uso de Redes Neuronales Recurrentes (RNN) o Long-Short Term Memory (LSTM), para analizar secuencias de datos temporales para predecir datos futuros. Podemos pensar en cómo, todos los días de semana, la curva de demanda sigue el mismo patrón, dos horarios peaks, uno a la hora de ida al trabajo, y otro a la hora de vuelta.

Una red neuronal especializada en analizar secuencias ordenadas es idónea para esta tarea. Estas redes neuronales son ampliamente usadas en el estado del arte para predecir el flujo de pasajeros en líneas de Metro en China que mas adelante se mencionarán. Eso es por el lado del análisis temporal, pero es importante notar la correlación espacial de la red, sobre todo si ocurren cambios en ella. Debido a la naturaleza de Red o **set de puntos conectados**, se nos viene a la cabeza inmediatamente usar Redes Neuronales de Grafos, gracias a su versatilidad en la forma de los inputs, a diferencia de los MLP o las CNN las cuales reciben vectores o grillas (como las imágenes).

Actualmente, algunos de los estudios que abordan esta problemática desde Chile lo hacen desde enfoques estadísticos y/o a nivel macro. Estos suelen analizar el antes y el después de una intervención, sin capacidad real de abstracción. Otros modelos tienen una orientación más predictiva, pero se encuentran desactualizados y no reflejan adecuadamente las dinámicas actuales del transporte urbano. También existen enfoques centrados en el transporte privado, que estudian cómo factores como la infraestructura, las tarifas o las políticas públicas afectan la movilidad general. Sin embargo, estos trabajos no se enfocan en cambios estructurales de la red de transporte público, sino que operan sobre la oferta ya existente.

Por otro lado, existe el sistema ADATRAP [13], desarrollado por la Universidad de Chile y el Instituto Sistemas Complejos de Ingeniería. ADATRAP es un software que analiza datos y permite planificar y crear estrategias para la priorización en la asignación de servicios públicos de transporte. El software toma en cuenta la distribución de la oferta para los usuarios del servicio en la Región Metropolitana. Adatrap será una fuente de datos importante para la predicción de la demanda.

Para finalizar, la solución propuesta en este proyecto se basa en el uso de técnicas de aprendizaje automático, modelando el sistema de transporte como un grafo en el que se representen recorridos, paradas y transbordos. Este modelo (sea uno discreto, como una MNL o una GNN) tendrá que aprender a predecir el comportamiento de los usuarios en función de múltiples factores, como la duración del viaje, el número de transbordos y el tiempo de espera. Estos modelos y sus resultados se compararán con datos reales de uso, para afinar el modelo y su precisión. Finalmente, se realizarán simulaciones de diferentes escenarios, como la introducción de nuevas líneas o la eliminación de recorridos, para observar cómo estos cambios afectan la demanda y la distribución de usuarios en la red obteniendo datos de la red y su uso modelado usando técnicas de ML y grafos.

## 1.1 Situación actual

Una gran motivación para este proyecto es la optimización en el uso de los recursos públicos. Modificar dinámicamente la frecuencia de los buses, crear nuevos recorridos o eliminar aquellos que han quedado obsoletos son decisiones que pueden tener un impacto significativo en la calidad del servicio y en la satisfacción de los usuarios. Sin embargo, estas decisiones deben basarse en datos precisos y actualizados sobre el uso del transporte público, así como en una comprensión profunda de cómo los cambios en la red afectan la demanda.

Siguiendo el trabajo de Torrepadula mas a fondo [15] se abren muchas soluciones y consideraciones: La primera , el sujeto de la predicción.

### 1.1.1 Sujeto de la predicción :

Diversos trabajos se enfocan tanto en:

1. Cantidad de personas en una parada en la ruta. Trabajos como el de Wei et. al [17] usan enfoques no lineales para estimar la demanda en algunas estaciones de metro.
2. Cantidad de personas en la ruta. El trabajo de Zhao [19] utiliza Prophet para estimar las personas en la ruta 320 de Zhengzhou, China
3. Cantidad de personas en un vehículo. Algunos trabajos lo predicen , como el de Wang et. al[9] con un Support Vector Machine mas un filtro de Kalman.
4. Cantidad de personas en un área. El trabajo de Wang et al [16] explora predicciones espacio temporales con un modelo llamado GALLAT (GrAphic preddiction with ALL ATtention), que modela la red como un grafo

Notar que cada enfoque o sujeto requiere un set de datos distintos, por ejemplo, para saber cuanta gente hay en un momento dado en un vehículo, debemos de usar cámaras o sensores, en cambio, para saber un estimado de gente en la ruta, podemos usar los datos de las validaciones de la tarjeta bip! de la ruta.

Por otro lado, todos los enfoques tienen el objetivo de predecir la demanda, pero cada uno de ellos tiene una metodología distinta de cómo hacerlo.

Un dato importante es el de como ADATRAP estima la cantidad de personas en la ruta[13]. ADATRAP tiene un algoritmo estimador de paradas de bajada de los usuarios. En sistemas como RED, el usuario solo valida en su subida al vehículo, por lo que no se sabe donde baja. La predicción se basa en el horario y ubicación de su subida al bus en la ida y en el horario y ubicación de la vuelta. Se entiende como ida y vuelta como el primer y el último viaje del día.

Una exploración inicial indica que el curso ideal sería contar cuanta gente usa cada ruta, por ello es que la cantidad de personas en la ruta puede ser un buen candidato.

### 1.1.2 Tipo de datos:

El tipo de datos es importante. Algunos ejemplos son:

1. Datos de validación de la tarjeta Bip! (que se puede usar para saber cuanta gente hay en una ruta, o en un área).
2. Datos de sensores (que se pueden usar para saber cuanta gente hay en un vehículo).
3. Datos de cámaras (que se pueden usar para saber cuanta gente hay en un vehículo, o en un área o un paradero).
4. GPS para el flujo de personas en un área.

Citando a Torrepadula [15], los datos de validación de la tarjeta , como la Bip! o sus equivalentes en otros países son los más utilizados, ya que son fáciles de obtener y tienen una buena cobertura geográfica. Sin embargo, también tienen limitaciones, como la falta de información sobre el origen y destino de los viajes. Los datos de sensores y cámaras son más precisos, pero son más difíciles de obtener y tienen una cobertura geográfica limitada. Los datos de GPS son muy precisos, pero también son difíciles de obtener y tienen una cobertura geográfica limitada.

Trabajos como los de Ye [18], Jian [3], Li [8] y Yang et.al utilizan datasets provenientes de tarjetas de validación con tecnología similar o idéntica a la de la tarjeta Bip!.

### 1.1.3 Factores

Los factores que afectan la demanda son diversos y pueden variar según el contexto. Algunos de los más relevantes son:

1. **Tarifas:** El costo del transporte público puede influir en la demanda, especialmente en áreas donde existen alternativas de transporte privado.
2. **Frecuencia:** La cantidad de buses o trenes disponibles en una ruta puede afectar la demanda, ya que una mayor frecuencia puede atraer a más usuarios.
3. **Tiempo de viaje:** La duración del trayecto es un factor clave en la decisión de utilizar el transporte público. Un tiempo de viaje más corto puede aumentar la demanda.
4. **Comodidad:** La calidad del servicio, como la limpieza, el confort y la seguridad, puede influir en la decisión de utilizar el transporte público.
5. **Accesibilidad:** La facilidad de acceso a las paradas o estaciones, así como la disponibilidad de servicios complementarios (como estacionamientos o bicicletas compartidas), puede afectar la demanda.
6. **Condiciones climáticas:** Factores como la lluvia, el frío o el calor extremo pueden influir en la decisión de utilizar el transporte público.
7. **Eventos especiales:** La realización de eventos masivos, como conciertos o ferias, puede generar picos de demanda en ciertas rutas.
8. **Fiestas y feriados:** La demanda de transporte público puede variar significativamente durante días festivos o feriados, lo que puede afectar la planificación de la oferta.
9. **Búsqueda Web** Los turistas, generalmente, se informan de las rutas y horarios de los buses en la web, por lo que el tráfico web puede ser un buen indicador de la demanda.

### 1.1.4 Modo de transporte

En distintos trabajos, se exploró la predicción de distintos métodos de transporte. Entre ellos están el Metro, buses, trenes y tranvías.

### 1.1.5 Técnicas de preprocesado de datos

Transformar los datos en una estructura de datos es un paso importante. GNNs requieren preprocesar los datos en matrices o grafos. Trabajos como los de Liu Et. Al[10] utilizan grafos representados por matrices del tipo (o,d), donde o es el origen y d es el destino de la persona. Predicciones hechas por ADATRAP [13] pueden ser utilizadas para llenar esta matriz de origen-destino. Otros enfoques, como el de Massobrio[11] modelan una red con nodos que representan las paradas de las rutas.

### 1.1.6 Técnicas de predicción

El área de interés de este trabajo son las soluciones que usan MNL o RNN y GNN/GCNN debido a la naturaleza de la creación de grafos y por el auge que Torrepadula menciona en su trabajo.

Algunas ventajas y desventajas de las mencionadas son:

- **RNN:** Ventajas: Captura correlaciones temporales, buena para series de tiempo multivariadas. Desventajas: No está diseñada para usarse con correlación espacial, es intensiva en recursos y tiene procesamiento paralelo limitado. Kang [5] explora una LSTM para predecir el volumen de personas en líneas de metro en China
- **GNN/GCNN:** Ventajas: Captura correlaciones espaciales. Desventajas: No captura correlaciones temporales, es intensiva en recursos, necesita la construcción del grafo. Li [7] es uno de los trabajos que explora estos métodos.
- **MNL :** Ventajas: Modelo interpretable, fácil de implementar. Rápido de entrenar. Tiene una capacidad media de generalización. Desventajas: No captura correlaciones espaciales ni temporales. Asume independencia de alternativas (que en este caso no afecta).

### 1.1.7 Actualmente en Chile.

Hoy en día, la red está enfrentando transformaciones importantes. La construcción e implementación de nuevas líneas de metro, como la Línea 7 y la futura Línea 8, tendrá un efecto profundo sobre el uso de ciertos recorridos de buses. Algunos servicios podrían volverse redundantes, mientras que otros —como los recorridos locales tipo [LETRA]-XX— podrían experimentar un aumento significativo en la demanda, al convertirse en alimentadores hacia las nuevas estaciones. Esta situación presenta una oportunidad para replantear frecuencias, redistribuir flotas y mejorar la eficiencia general del sistema.

El más destacado es ADATRAP, desarrollado por la Universidad de Chile y el Instituto Sistemas Complejos de Ingeniería. Este software permite analizar datos y planificar estrategias para la priorización en la asignación de servicios públicos de transporte. ADATRAP toma en cuenta la distribución de la oferta para los usuarios del servicio en la Región Metropolitana.

Adatrap [13] es un software que utiliza la información geotemporal referenciada (GPS) en buses de Transantiago, en conjunto con la información que entrega la tarjeta Bip!, con el objetivo

de estimar desempeño de transporte público, velocidades de traslado, hacinamiento, perfiles de carga, etc. Logra crear perfiles de velocidad por servicio y por tramo de ruta, perfiles de carga por servicio, matrices origen-destino, indicadores de calidad de servicio. El software está registrado a nombre de la Universidad de Chile y transferido mediante acuerdo de licencia a la Subsecretaría de Transportes. Se utiliza diariamente para tomar decisiones tales como la definición semanal de programas de operación, modificación de servicios y decisiones de infraestructura

Estos fenómenos han sido objeto de análisis en trabajos previos. Un ejemplo representativo es el de Ramírez [12], quien estudia el cambio espacial en la demanda de transporte público tras la apertura de una nueva línea de metro, empleando un enfoque estadístico. Si bien su análisis es útil para evaluar efectos pasados y prever algunos eventos futuros, su falta de generalización da una ventana de oportunidad.

Por otra parte, el trabajo de Camus [1] propone una simulación basada en agentes dentro de la red de transporte público. Sin embargo, dicho modelo considera la oferta como un elemento estático y no contempla escenarios en los que esta pueda ser modificada. Aun así, su enfoque representa un punto de partida interesante, ya que podría ser extendido para evaluar diferentes configuraciones de red.

También existe el modelo desarrollado para el Directorio de Transporte Público Metropolitano (DTPM) [2], mediante el software EMME de Bentley. Algunas características del modelo de demanda generado se basan en entradas como el diseño, la demanda y los datos operacionales. Luego, puede predecir y simular el impacto de cambios en la infraestructura para planear cambios. Este modelo de demanda fue creado con el plan de operación de 2020 (Marzo) y la demanda de 2019 (Agosto). Las franjas horarias (o períodos de análisis) son 3, Punta Mañana, Fuera de Punta Mañana y Punta Tarde. En el documento expuesto por la DTPM, el proceso de ajuste de matrices de viaje no contó con los aforos de la zona oriente, pero pudieron ser subsanados con datos anteriores, aunque se reconoce una posible subestimación de la demanda en esa zona. Al no ser de código abierto el software, no mucha más información se puede recabar.

Asimismo, existen modelos de demanda agregada, como el desarrollado por Méndez [14], que se apoyan en técnicas econométricas y estudian elasticidades en función de variables como tarifas o cantidad de servicios disponibles. Aunque valiosos, estos trabajos no abordan cambios estructurales en la red, sino que se enfocan en la oferta existente.

En resumen, los trabajos existentes suelen analizar contextos acotados no generalizables y con poca versatilidad. Otros no son de código abierto como el de EMME. Esta memoria intenta abstraer el problema y generalizar los casos, para que más que un estudio pueda ser usada como una herramienta para generar datos sintéticos y de toma de decisiones. Una red neuronal es capaz de hacer esto debido a su capacidad de abstracción espacial, en el caso de las GNN, y abstracción temporal, en el caso de las RNN.

## 1.2 Objetivos

### 1.2.1 Objetivo general

Diseñar e implementar un modelo que prediga demanda de transporte dado un escenario (definido como una configuración de red y su respectiva infraestructura urbana); y usar este modelo para predecir demanda en distintos escenarios para medir el impacto de intervenciones en el escenario actual.

### 1.2.2 Objetivos específicos

1. Disponer de datos actualizados sobre el uso de transporte publico, como frecuencias e itinerarios y los destinos/origenes de los usuarios, como también, a de ser posible, de flujos de transporte.
2. Modelar la red de transporte publico en un grafo o hipergrafo de ser necesario, que permita representar la topología de la red de transporte y las combinaciones de ellas.
3. Modelar un sistema de decisiones con un Modelo Logit Multinomial y obtener parámetros para una función de probabilidad para crear datos de demanda sintéticos.
4. Modelar la demanda en sus dos aspectos, espacial y temporal. Para ello, se utilizará un GNN para capturar la topología de la red y una RNN para capturar la temporalidad de los datos. Se espera que el modelo sea capaz de predecir la demanda en función de los factores anteriormente mencionados.
5. Cambiar la topología de la red y observar cómo cambia la demanda . Cambiar la topología involucrará cambios de infraestructura (agregar, quitar o modificar rutas existentes) como también cambios en la frecuencia de los buses.
6. Analizar los datos de la nueva demanda prestando atención al nuevo número de pasajeros transportados por cada línea.

### 1.2.3 Evaluación

Cada objetivo se verificaría de la siguiente manera:

1. Datos actualizados: Se espera contar con datos de validación de la tarjeta Bip! y registros de uso de suelo de Santiago.
2. Modelado de la red: Se espera contar con un modelo de la red de transporte público que permita representar recorridos, paradas y transbordos. Para ello, se compara con trabajos previos que han utilizado modelos similares de modelado de las redes.
3. Modelo de ML para predicción: Se espera contar con un modelo de aprendizaje automático que simule el comportamiento de los usuarios en función de múltiples factores. Este modelo se validará comparando sus predicciones con datos reales de uso de transporte público, como los proporcionados por la tarjeta Bip!.



4. Al modificar la red, se espera que el modelo de ML pueda predecir cambios en la demanda y la distribución de usuarios en la red. Esto se validará instanciando diferentes escenarios y comparando los resultados con datos reales de uso. (Por ejemplo, red pre/post línea 6)
5. Análisis de resultados: Se espera realizar un análisis exhaustivo de los resultados obtenidos a partir de la simulación, identificando patrones y tendencias que puedan informar futuras decisiones en la red de transporte.

## 1.3 Solución propuesta

La solución propuesta se basa en la creación de un sistema de simulación del transporte público que combine estructuras de grafos y técnicas de aprendizaje automático. El enfoque contempla los siguientes componentes:

0. En cuanto al tech stack, se usará Python como lenguaje de programación, bibliotecas como Tensorflow o Pytorch y sus derivados para crear las LSTM y GNN/CNN. NetworkX puede ser utilizado para trabajar con grafos y numpy, scipy y pandas para analizar y cargar los datos.
1. Modelado de la red como grafo: La red de transporte será representada como un grafo, donde los nodos corresponden a paradas o estaciones, y las aristas a tramos recorridos. Esta representación permitirá modelar recorridos compartidos (por ejemplo, buses distintos que recorren el mismo tramo), y considerar distintas características de cada servicio como atributos de las aristas: frecuencia, tiempo estimado, comodidad, etc. Los datos para esto se obtendrán de datos de RED y sus recorridos.

También se va a explorar la creación del hipergrafo peatonal si es que es necesario para mejorar las métricas del modelo, ya que no todas las estaciones combinan (por ejemplo, caminar dos cuadras para ir de un lugar a otro).

### 2. MNL

Se implementará un modelo logit multinomial que aprenda a captar factores decidores ajustando una función de probabilidad. Estos factores decidores son factores cuantitativos tales como el tiempo de viaje, el numero de transbordos, el tipo de transporte usado y demases.

3. GNN + RNN: Se implementará un modelo de aprendizaje automático para replicar la demanda de uso de transporte público en función de múltiples factores. Este modelo aprenderá a predecir el comportamiento de los usuarios en función de variables como la duración del viaje, el número de transbordos y el tiempo de espera. Se utilizarán técnicas de aprendizaje supervisado para ajustar los parámetros del modelo, utilizando datos históricos de validaciones Bip!, datos de uso de suelo y patrones de movilidad. Para ello se utilizará un modelo con GNN + RNN (por ejemplo, una LSTM) . Una GNN procesará la estructura espacial del grafo y la demanda histórica con una LSTM.

4. Entrenamiento y ajuste del modelo: Utilizando datos históricos (validaciones Bip!, patrones de movilidad, datos censales), se ajustarán los parámetros del modelo de ML para que el comportamiento simulado refleje lo más fielmente posible la realidad. Esto puede abordarse como un problema de optimización o incluso como un sistema de aprendizaje supervisado.
5. Ajustes a la oferta: Con el modelo calibrado, se podrán introducir cambios en la red (nuevas líneas, suspensión de servicios, variaciones de frecuencia) y observar cómo cambia la distribución de la demanda. Esto permitirá anticipar efectos como saturación de recorridos, desplazamiento de flujos o desuso de servicios.
6. Análisis de resultados: Finalmente, se realizará un análisis exhaustivo de los resultados obtenidos: se evaluarán métricas como tiempos promedio de viaje, número de transbordos, uso por línea y comparativas entre escenarios. El objetivo es que este análisis brinde insumos para decisiones estratégicas en la planificación del sistema de transporte.

En la figura 1.1 se presenta un diagrama de la solución propuesta, que ilustra los componentes y flujos de información del sistema.

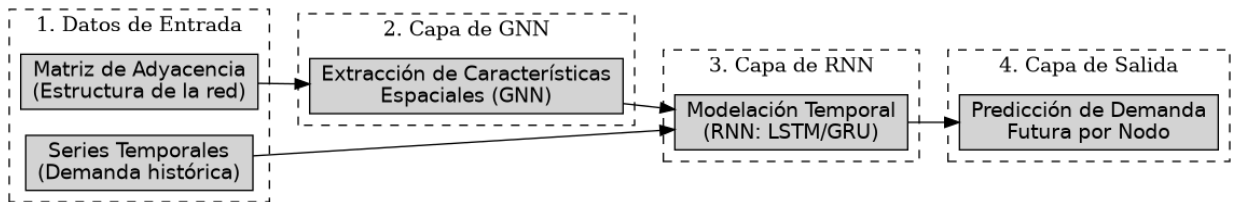


Figura 1.1: Diagrama de solución

## 1.4 Plan de trabajo

Debido al trabajo adelantado hecho en este informe, una reestructuración de la carta Gantt es necesaria para reflejar el progreso.

Tabla 1.1: Carta Gantt.

Tarea	Mes 1	Mes 2	Mes 3	Mes 4
Obtención de datos de Demanda y Recorridos	LISTO			
Análisis exploratorio de los datos de Demanda y Recorridos	LISTO			
Usando datos de recorridos, crear grafo		LISTO		
Validación visual del grafo	X__			
Limpieza de datos y/o correcciones post-analisis del grafo	_X__			
Obtención, limpieza y aplicación de datos de uso del suelo	__XX			
Crear modelo de GNN (Ambos enfoques)		XXXX		
Comparar resultados de demanda con los reales (validar modelo)			XX__	
Con la red hecha y el modelo de ML validado, experimentar con cambios en la oferta modificando la red			__XX	XX__
Analizar los cambios de la demanda y ajustar el modelo según resultados				__XX
Redactar memoria y preparar defensa.				XXXX

## Capítulo 2

# Plataforma técnica, datos y exploración de ellos.

En la siguiente sección se darán a conocer los datos mediante una exploración de ellos. Para ello, si se desea seguir el informe, se mostrarán alternativas para

## 2.1 Plataforma de desarrollo y tech stack

Debido a la mayor disponibilidad de paquetes y herramientas, y la familiaridad del lenguaje, se optó por usar Python como plataforma de desarrollo. A medida que se mencionarán los pasos seguidos, mas adelante, se darán a conocer los paquetes y herramientas utilizadas.

## 2.2 Exploración del repositorio

Para efectos de visualización y/o inspección de los datos, podemos clonar el repositorio ubicado en <https://github.com/Sebamon2/memoria-repo>. La plataforma del proyecto es en Python.

### 2.2.1 Instalación

Este apartado es solo para quienes estén interesados en interactuar con los grafos y mapas generados. No es estrictamente necesario para entender la memoria, pero puede ser usado como una herramienta de exploración. Mas información sobre la instalación se encuentra en el README.md del repositorio mismo el cual está alojado en GitHub.

## 2.2.2 Notebooks de exploración

La carpeta notebooks contiene todos los notebooks de jupyter para la exploración de los datos.

Para la próxima sección, puede ser interesante revisar el notebook llamado ‘data\_inspection.ipynb’.

## 2.3 Exploración de datos generados por ADATRAP

ADATRAP entrega datos de viajes y etapas. Los datos están públicos en el siguiente enlace: <https://www.dtpm.cl/index.php/documentos/matrices-de-viaje>. Cada viaje tiene n etapas, hasta 4 como máximo.

Cada viaje tiene un origen y un destino. El sistema de transportes capitalino no posee validación de la Bip! o sus derivados al termino de la etapa, por lo que la estimación de este parámetro fue realizada por el software ADATRAP. ADATRAP analiza los patrones de viaje de usuarios para detectar donde se sube y baja. Por ejemplo, si un usuario sube a las 7:00 AM en el servicio X en el paradero P, y se sube a las 19:00 en el servicio Y en el paradero P’, esto con cierta regularidad. Se concluye que en la mañana el usuario se bajo cerca del paradero P’ usando el sevicio X, y que en la tarde el usuario se bajó cerca del paradero P en el servicio Y.

### 2.3.1 Tabla de viajes y etapas

En nuestra solución, las tablas de viajes y etapas serán nuestras demandas históricas.

La tabla de viajes contiene la información de los viajes del usuario, registrando hasta 4 etapas o 3 combinaciones. Combinaciones en metro no cuentan, pues no se valida la tarjeta al cambiar de linea. Cada tabla de viajes o de etapas corresponde a un solo día de análisis. Las tablas de viajes y de etapas vienen generalmente en packs de una semana completa.

### Código TS y Código Usuario

Los servicios y paraderos se encuentran codificados en formato TS, esto es, un código interno usado por DTPM para identificar a los recorridos. La mayoría de los recorridos tiene un código TS que coincide con el de usuario. Por ejemplo, el servicio **T507 OOI** codifica al servicio 507 de ida (servicio en sentido ENEA- AV GRECIA). En algunas ocasiones no coincide, esto ocurre mayoritariamente en servicios locales con prefijo alfabético, casos como el servicio con código de usuario **J01** en código TS es en **T521**. Esta es la razón por la cual algunos recorridos nuevos tienen códigos de usuario que no siguen el numerado del usuario, ya que si lo siguieran, habrían colisiones de nombres.

Por otro lado, los códigos de paradero también poseen esta distinción. Ningún código de paradero de usuario coincide con su versión en TS. En el set de datos de tabla de viajes y de etapas ambos códigos, tanto el de paraderos como el de servicios vienen en código TS.

## **Paraderos subida y bajada**

Ambas en código TS, denotan, para las 4 posibles etapas, las subidas y bajadas del usuario. Máximo 8 (2 por cada etapa).

## **Horas de subida y bajada**

Estimados con la velocidad promedio de los buses y los itinerarios, cada etapa tiene un horario de subida y bajada. Máximo 8 (2 por cada etapa).

## **Servicios de las 4 etapas**

En formato TS. Servicio de cada etapa. Máximo 4 (1 por cada etapa).

Hay mas columnas, pero para el análisis posterior no son de relevancia.

La tabla de etapas contiene la misma información pero de manera disgregada, es decir, cada fila es una etapa.

### **2.3.2 Consolidado de recorridos**

Para crear el grafo, lógicamente es necesario el trazado de todos los recorridos de RED. Para ello, se descargó desde su página web el trazado activo hasta ahora. Este archivo contiene en sus columnas:

1. Los códigos de los servicios y paraderos en TS y en formato usuario.
2. El nombre del paradero.
3. Excepciones del paradero.
4. Las posiciones X e Y del paradero.(UTGSM)

Cada fila contiene una parada de un trazado de un servicio.

Con esta información, podemos hacer dos cosas.

1. Crear el grafo de la red (sin aún añadir información de la demanda).
2. Crear un diccionario de TS a Usuario de los paraderos.

Algo importante a notar es la fecha de esta tabla de recorridos. Es válida desde el 31/05/2025 hasta a fin de año (al momento de hacer este informe)

## 2.4 Exploración de datos

Usando toda la información disponible de momento, podemos generar algunos histogramas interesantes para familiarizarnos con las varias formas de acceder y manipular los datos. La figura 2.1 muestra las subidas de un paradero PJ394 (José Joaquín Pérez con Las Lomas en Cerro Navia)

### 2.4.1 Subidas a un paradero durante el día.

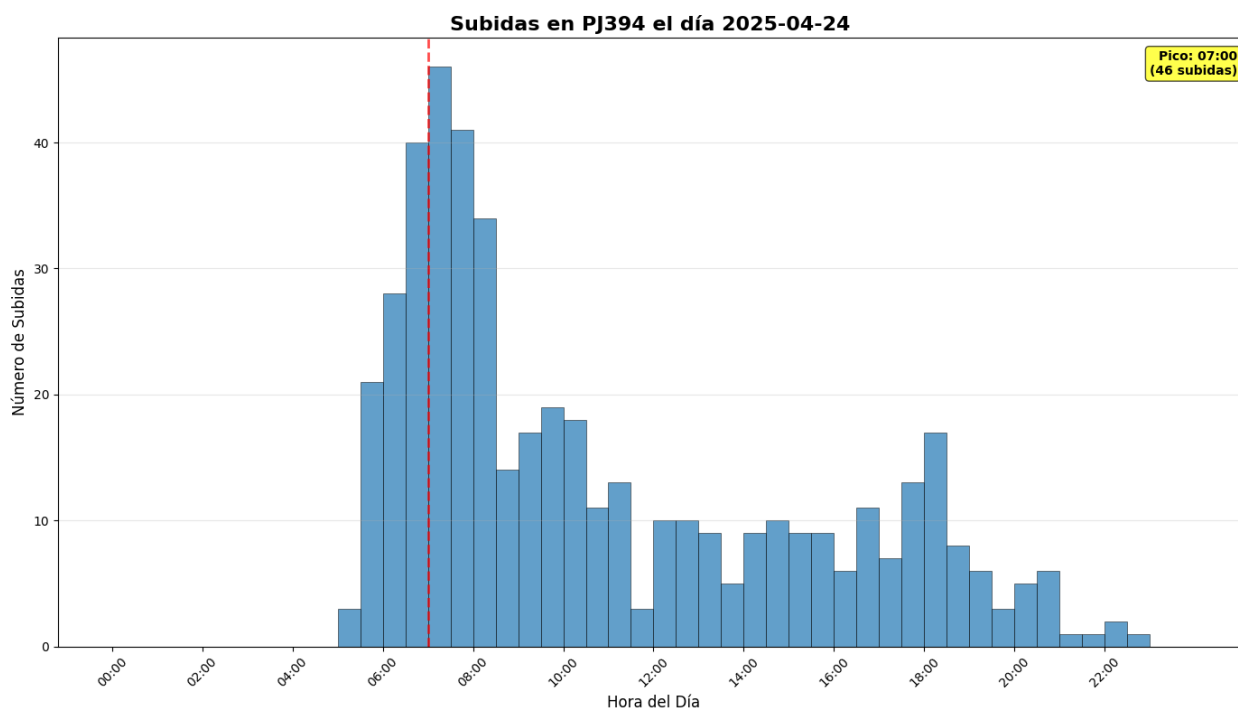


Figura 2.1: Subidas en el paradero PJ394

Podemos hacer el mismo análisis para paradas del Metro de Santiago, por ejemplo, analizar la estación de Metro Tobalaba.



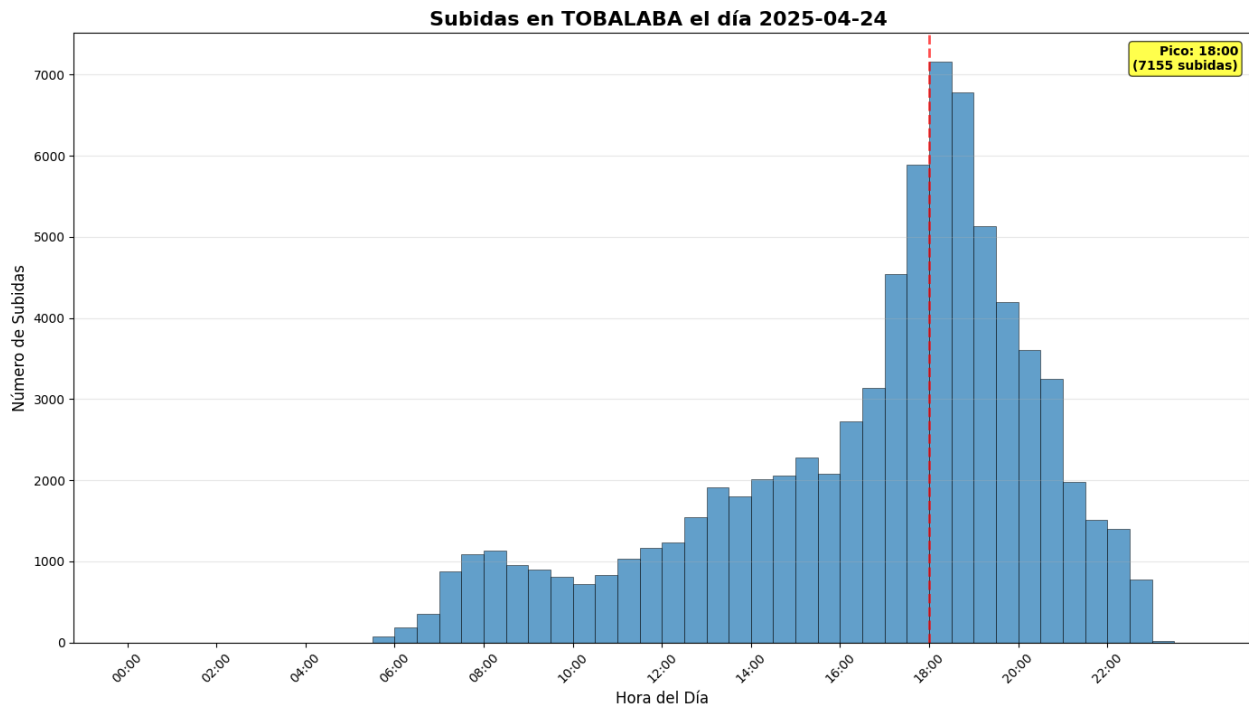


Figura 2.2: Subidas en Tobalaba L1 y L4

Podemos darnos cuenta claramente de la distribución de la hora peak en el Metro Tobalaba a las 18:00 horas. Algo importante se nos muestra en el grafico anterior. Tenemos que tratar a las paradas de buses igual que a las de Metro, es decir, como un Hub de servicios que pasan por ahí. Alguien puede marcar su pasaje en los torniquetes de la línea 1 y dirigirse automáticamente a la línea 4.

## 2.4.2 Uso de un servicio.

Una métrica clave a comparar cuando se realicen cambios en la oferta del transporte, es el uso de un servicio. Una hipótesis razonable es que si quito un servicio dado, servicios aledaños van a ver su demanda subir. Ejemplos tangibles de ello es cuando la línea 1 colapsa por eventos fortuitos. Servicios de superficie que circulan por el eje Alameda-Providencia se ven saturados. El siguiente gráfico muestra el uso del servicio T507 00R.

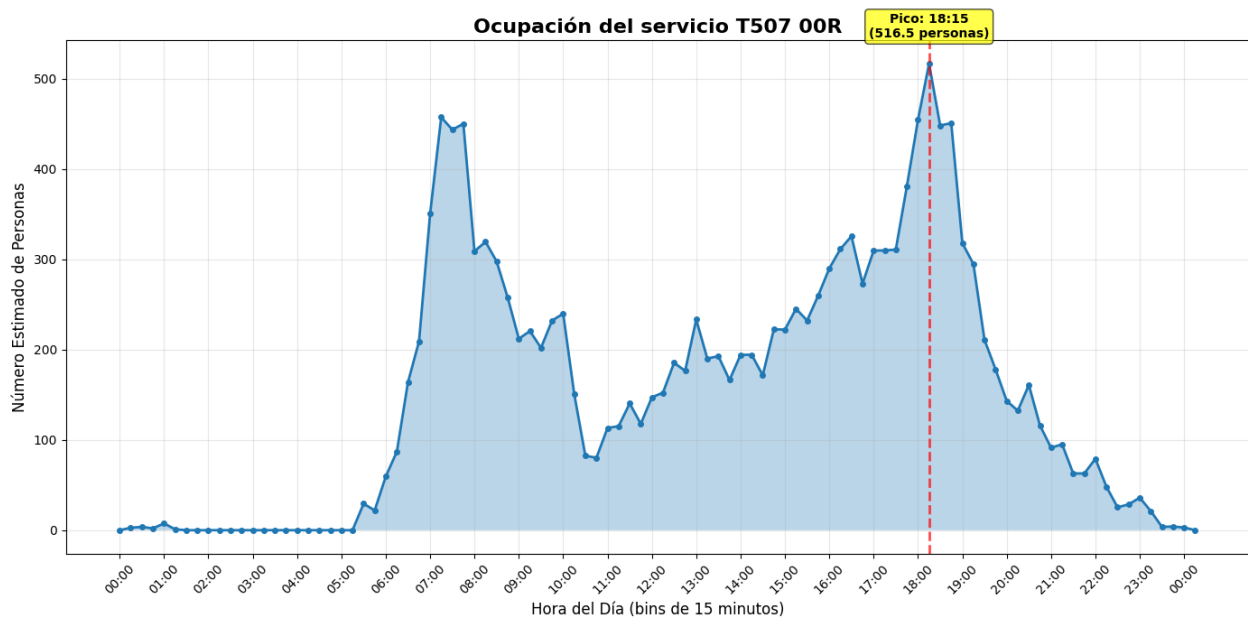


Figura 2.3: Uso del servicio 507 de vuelta (Desde Grecia a ENEA)

Algunos viajes no tenían hora de bajada (eran nulls). Cuando esto pasaba, se asumía que la persona se bajaba 30 minutos después de subirse. Es un valor arbitrario, pero razonable.

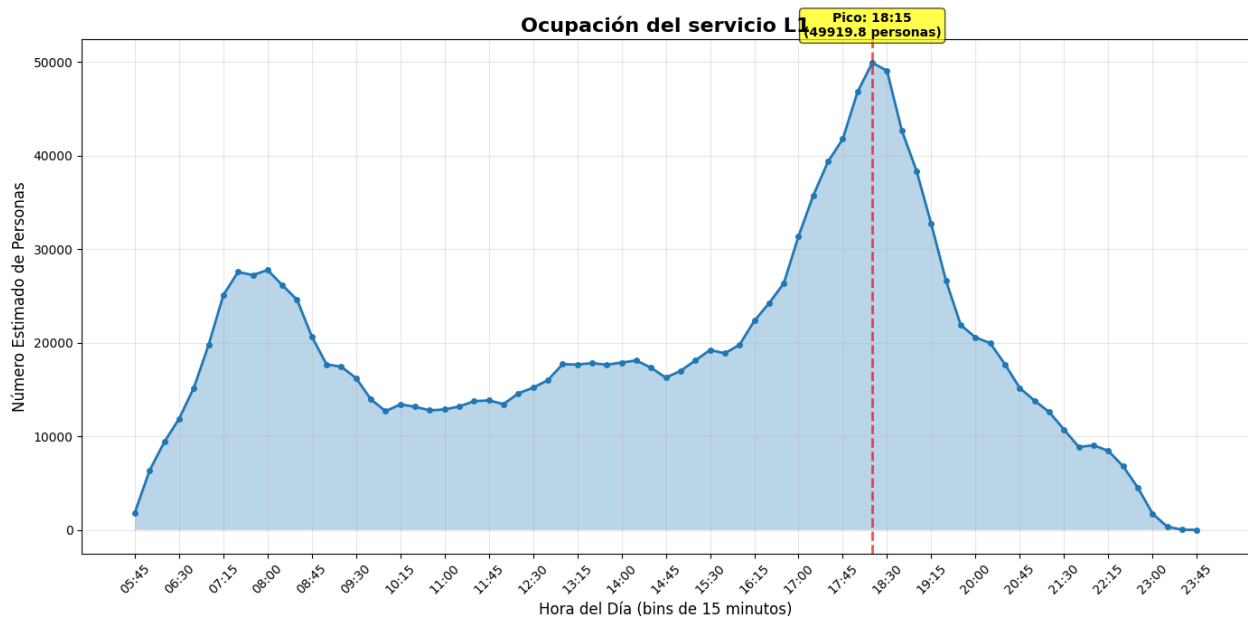


Figura 2.4: Uso de la Línea 1 durante el día

La figura 2.4 nos muestra algo interesante. El uso de la Línea 1 no es simétrico en el tiempo como el de la 507.

Igualmente, no se tomó en cuenta los casos en los que las personas validan en torniquetes de la línea 1 y combinan inmediatamente. Es necesario mas cuidado en casos del metro.

Una posible métrica interesante, sería obtener el porcentaje de uso de un servicio en un sentido con respecto a la cantidad de vehículos que tiene circulando el servicio en un período de tiempo. Esto permitiría ajustar la oferta de manera dinámica. Para ello habría que estimar la cantidad de personas máxima que cae dentro de un vehículo típico del servicio. Este análisis no se hará en esta fase del informe, pero queda propuesto para el siguiente semestre.

## 2.5 Creación del grafo agrupado

La Matriz de adyacencia puede ser construida al mirar la consolidación de recorridos. Por lo tanto, crear el grafo de la red es el paso mas crucial para poder crear esta matriz de adyacencia, que representa la estructura de la red.

Un grafo  $G(E,V)$  es un conjunto de aristas( $E$ ) y vertices( $V$ ). Estos pueden ser dirigidos (los vértices tienen dirección bloqueada) o no (ambas direcciones posibles).

### 2.5.1 Aristas

En nuestro caso, las aristas  $E$  son las conexiones entre dos paraderos en un recorrido. Por ejemplo, una arista conecta la estación Los Héroes con Moneda. Una arista, por lo tanto, debe de guardar, al menos, los servicios que la recorren. En este caso, sería la Línea 1 en ambas direcciones, por lo que aquí tenemos dos opciones, o tener dos aristas para ambas direcciones o una arista sin direcciones.

Otro caso, son las aristas que unen paradas de servicios en superficie. Una arista va a representar la conexión entre dos paraderos consecutivos mediante un servicio.

Podemos dibujar las aristas de dos formas:

1. Cada arista representa solo la conexión dada entre dos paraderos consecutivos recorridos por un servicio. Es mas complicado computacionalmente y hará que el grafo tenga mas aristas, pero es mas completo y permite guardar mas información. Por ejemplo, si un servicio  $X$  e  $Y$  tienen las mismas paradas consecutivas, pero el recorrido  $Y$  pasa por calles distintas al  $X$  entre las paradas, es evidente que el tiempo que le toma a ambos servicios recorrer la arista es distinto, pues la geografía es distinta (a pesar de que la topología sea la misma en el grafo).
2. Si varios servicios paran en las mismas paradas consecutivas, podemos unir todos los recorridos en la misma arista. Es mas simple computacionalmente, pero datos como la distancia o tiempo que toma al servicio recorrer la arista (el peso de la arista) no podría ser el mismo.

En este documento se explora la segunda forma de hacerlo, pero probablemente se tenga que hacer de la primera forma.

Notar que los vértices además de guardar la distancia o tiempo promedio que recorre el servicio correspondiente, guardan el sentido. Lo que no guardan, es la geografía del recorrido. Esa información está implícita en la distancia o tiempo que le toma al servicio recorrer la arista.

### 2.5.2 Vértices

Los vértices  $V$  son las paradas. Cada parada tiene un par coordenado (lat, lon) que la posiciona en el grafo. Una parada se identifica con el código TS del paradero. Una parada contiene 1 o más servicios.

### 2.5.3 Algoritmo para crear el grafo agrupado

Una primera aproximación para crear el grafo, consistirá en agrupar a todas las conexiones de dos paraderos consecutivos en una arista en común. Es decir:

1. El servicio  $X$  tiene una secuencia de paraderos  $P_k$ , con  $k$  el número de paradero en el recorrido.  $P_0$  es el paradero inicial y  $P_N$  es el paradero final del recorrido.
2. Los paraderos se configuran en nodos  $V$ . Cada nodo  $V$  tiene como llave su código de usuario  $C$ , una lista de servicios  $S[]$  y un par coordenado (lat, lon) para ubicarlo geográficamente.
3. La lista de servicios de un paradero depende de la hora. En esta versión del grafo no se implementará esto, pero en futuras versiones, es necesario para identificar paraderos con recorridos no invariantes temporalmente.
4. Cada servicio tiene una secuencia de nodos que visita en orden. Digamos que la secuencia de paraderos que visita un recorrido  $X$  es  $P[]$ . Si el set de nodos es  $V[]$ , podemos hacer una biyección entre  $P_k$  y  $V_i$ . Siendo  $k$  el  $k$ -ésimo paradero en orden e  $i$  el  $i$ -ésimo paradero de toda la red. Obviamente  $i$  no tiene por qué ser igual a  $k$ .
5. Si hay dos servicios,  $X$  e  $Y$ , que tienen secuencias de paraderos  $P_k$  y  $Q_k$  y tienen dos paraderos consecutivos que coinciden, es decir,  $P_k = Q_i$  y  $P_{k+1} = Q_{i+1}$ , luego podemos decir que desde  $P_k=Q_i=V_l$  a  $P_{k+1}=Q_{i+1}=V_m$  habrá una arista en esa dirección, con  $m$  y  $l$  no necesariamente consecutivos.
6. Esta arista direccionada desde  $V_l$  a  $V_m$  tendrá como información que los servicios  $X$  e  $Y$  pasan por ella.

Siguiendo estas reglas, se crea el grafo con el siguiente pseudocódigo:

1. Se obtienen todos los servicios únicos en el dataframe polars (Se eligió Polars en vez de pandas gracias a su rapidez para cargar archivos .csv grandes. Mas información sobre polars en el siguiente enlace: <https://pola.rs/>).
2. Se crea un diccionario con la información Código Usuario, Variante (PM o Normal), Sentido Servicio (Ida o Regreso).

3. Por cada servicio, se filtran del dataframe todas las filas que corresponden al servicio.
4. Se ordena el dataframe viendo la columna “orden\_circ”. Esta es la columna que denota el orden de circulación del servicio por los paraderos.
5. Por cada fila (paradero) del dataframe, se crea o actualiza un diccionario que corresponde al paradero, con llave código paradero, con los siguientes datos:

- llave(codigo paradero)
- lat
- lon
- servicios
- nombre (Por ejemplo, José Joaquín Pérez esq Las Lomas)
- nombre completo (código del paradero + nombre del paradero)
- tipo (BUS o Metro)

6. Por cada fila del dataframe, revisamos el parámetro “siguiente\_parada” que contiene la siguiente parada desde la que estamos revisando (un puntero básicamente). Creamos una arista  $E_1$  en un diccionario que une ambas paradas con la siguiente información:

- conexion\_id (llave formada por el par codigo\_paradero\_origen, codigo\_paradero\_siguiente)
- servicios
- nodo\_origen
- nodo\_destino
- tipo (Bus o Metro)

Notar que al hacer esto por todos los servicios, se van a agregar a cada arista los servicios que recorren ambos nodos en el mismo orden.

7. Se realiza el mismo procedimiento para el Metro, pero las aristas son bidireccionales (es decir, por cada conexión, se hace una simétrica pero en sentido inverso).
8. Con NetworkX se crea un grafo dirigido con DiGraph().
9. Se convierten los sets de servicios a listas para que GraphML la pueda procesar.
10. Creamos un nodo por cada paradero.
11. Unimos los nodos con las aristas.

Con ello, podemos crear un grafo interactivo con Gephi (software open source) que nos permite visualizar el grafo. Podemos utilizar el par lat, lon para generar un grafo configurado de manera visual con GeoLayout.

De la misma forma, podemos crear un mapa interactivo con toda la red usando Plotly en python.

Con ello, se crearon:

- 11890 paraderos de bus
- 126 estaciones de metro
- 15465 conexiones de bus
- 272 conexiones de metro
- 15737 conexiones totales

Al final de este informe se agregó en formato PDF el grafo, pero si se quiere ver de manera interactiva, el notebook de jupyter llamado 'visualization.ipynb' tiene todos los pasos necesarios para generar el grafo. En el mismo notebook se muestra el mapa de Santiago con toda la red usando Plotly. Si se desea observar el grafo con Gephi, es necesario descargar el software, instalarlo, cargar el grafo (ubicado en data/graphs/grafa.graphml) y en layout seleccionar Geo Layout y colocar la escala en 1E6 (10 a la 6). Si no se encuentra la opción, es necesario instalar el plugin en el mismo software desde el menú del mismo nombre.

En el grafo mostrado al final del informe, las aristas y vértices azules son las designadas a buses. Las rojas son las del metro. Para una próxima versión, será necesario agregar el metro tren

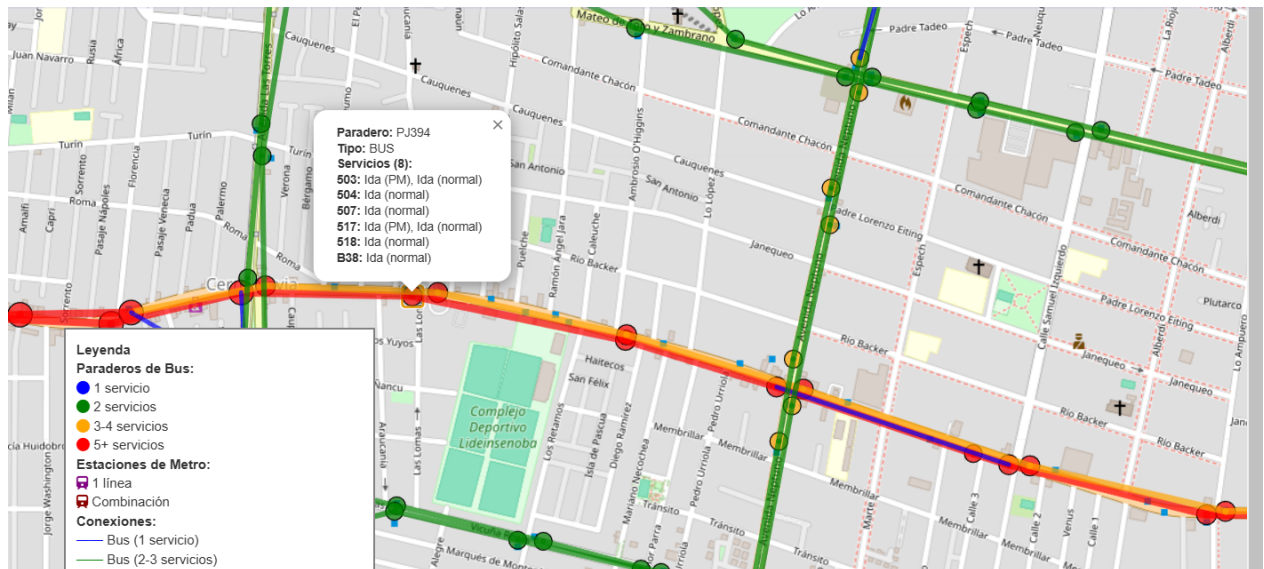


Figura 2.5: Mapa en Plotly con zoom a un barrio de Cerro Navia

Este algoritmo nos permite visualizar el grafo completo, pero carece de funcionalidad para agregarle información de la oferta.

## 2.6 Métricas de Demanda

La idea de predecir la demanda conlleva saber exactamente la demanda de un par paradero, servicio, hora.

Sea  $P$  el paradero,  $S$  el servicio,  $T$  el espacio de tiempo y  $D$  la demanda, debemos de hacer una función  $D(P,S,T)$  la cual retorna la demanda de un paradero en función del servicio y la hora.

Haciendo esto, podemos obtener la demanda de todas las tuplas P,S,T. La idea es escoger una ventana de tiempo  $\Delta t$  y establecer una distribución acumulada que determine la demanda entre ambos tiempos. En el notebook de jupyter llamado demand\_getter.ipynb se muestran ejemplos de demandas de varios paraderos. Por ejemplo, al ejecutar la función en el paradero **PJ394** con  $T_{ini} = 8:00$  y  $T_{fin} = 10:00$ , con el servicio **T507** obtenemos:

#### Listing 2.1: Salida del Programa

```
El paradero PJ394 en formato TS es: T-11-64-P0-30
Buscando demanda en T-11-64-P0-30 para T507 00I entre 08:00:00 y 10:00:00 ...
Procesando etapa 1 ...
Demanda en T-11-64-P0-30 para T507 00I en etapa 1: 20 viajes
Procesando etapa 2 ...
Procesando etapa 3 ...
Procesando etapa 4 ...
Total de viajes en T-11-64-P0-30 para T507 00I: 20
```

Para Tobalaba L4 entre las 17:00 y las 18:00

#### Listing 2.2: Salida del Programa

```
No se encontró el paradero en formato TS.
O es un paradero de metro, o no existe el paradero en la base de datos.
Buscando demanda en TOBALABA para L4 entre 17:00:00 y 18:00:00 ...
Procesando etapa 1 ...
Demanda en TOBALABA para L4 en etapa 1: 9226 viajes
Procesando etapa 2 ...
Demanda en TOBALABA para L4 en etapa 2: 1191 viajes
Procesando etapa 3 ...
Demanda en TOBALABA para L4 en etapa 3: 16 viajes
Procesando etapa 4 ...
Demanda en TOBALABA para L4 en etapa 4: 3 viajes
Total de viajes en TOBALABA para L4: 10436
```

Algo curioso ocurre para Tobalaba L1

### Listing 2.3: Salida del Programa

```
No se encontró el paradero en formato TS.
0 es un paradero de metro, o no existe el paradero en la base de datos.
Buscando demanda en TOBALABA para L1 entre 17:00:00 y 18:00:00 ...
Procesando etapa 1 ...
Procesando etapa 2 ...
Procesando etapa 3 ...
Procesando etapa 4 ...
Total de viajes en TOBALABA para L1: 0
```

Nuestras sospechas sobre como se guarda el servicio en estaciones de metro fue cierto. Al marcar la Bip! en Tobalaba, se marca automaticamente como L4 , nunca como L1. Este problema hay que resolverlo prontamente.

Esta data por cada una de las tuplas es la información de entranamiento que tendra la GNN para predecir la demanda.

## 2.7 Grafo de estado no agrupado.

Un grafo mas sofisticado es necesario para capturar la información de la demanda. Para ello, se creará un grafo no agrupado de estado. El grafo tendrá distintos tipos de aristas y nodos. Pero antes, es necesario establecer algunas fuentes de datos extra.

Para la siguiente sección, se es necesario tener un diccionario en común.

### 2.7.1 Notación y datos base

- **Paraderos:**  $P, Q, \dots$
- **Servicios:**  $S$  (ej: 507), con **sentido**  $d \in \{\text{Ida}, \text{Ret}\}$ .
- **Tipo de día:**  $D \in \{\text{LAB}, \text{SAB}, \text{DOM}\}$ .
- **Tiempo discreto:** 48 bins de media hora  $b \in \{0, \dots, 47\}$ .
- **Frecuencia** (buses/h):  $f_{S,d}(D, b)$ .
- **Headway** (min entre buses):  $H_{S,d}(D, b) = \frac{60}{f_{S,d}(D, b)}$ .
- **Aristas VIAJAR:** tramo  $(u \rightarrow v, S, d)$  con:
  - **distancia**  $L_e$  (m),
  - **velocidad**  $v_e(D, b)$  (km/h),
  - **tiempo a bordo:**

$$\tau_e(D, b) = \frac{L_e/1000}{v_e(D, b)} \cdot 60 \quad [\text{min}]$$



### 2.7.2 Zona 777

La ciudad de Santiago esta dividida en zonas tarifarias. Las zonas 777 son el nombre que poseen. Mas abajo se muestra la figura 2.6 que muestra las zonas 777 de Santiago.

Mapa Interactivo de Zonas 777

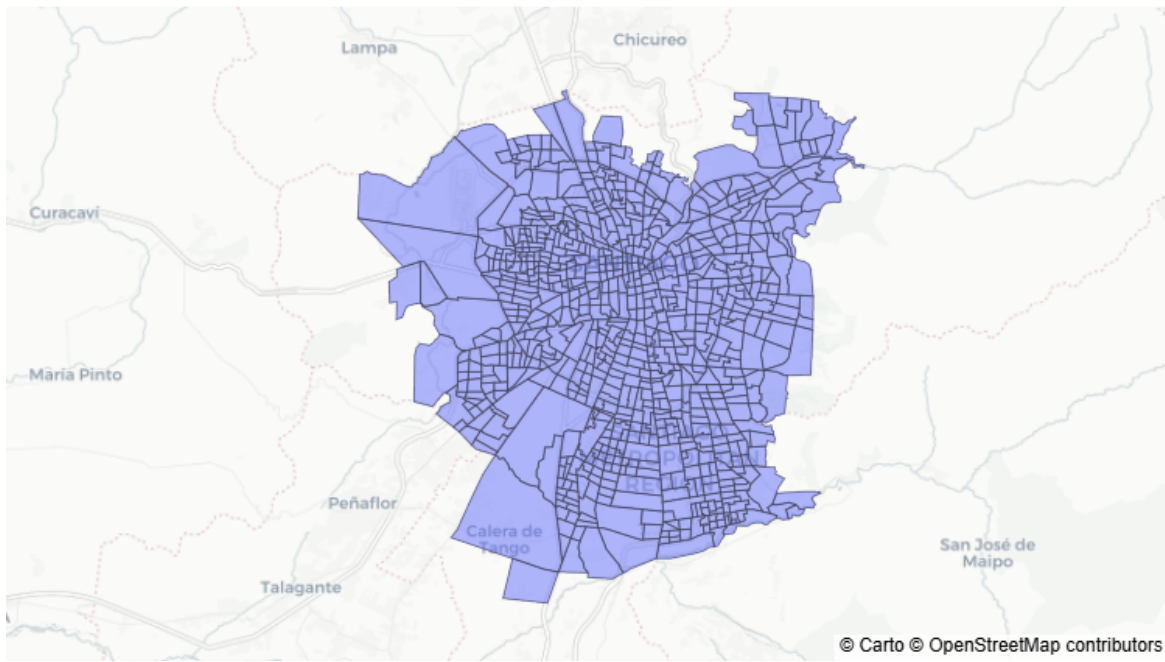


Figura 2.6: Zonas tarifarias 777 en Santiago

En el repositorio de GitHub en `main_notebook.ipynb` se encuentra un mapa interactivo con las zonas 777 de Santiago. Igualmente el mapa de la red tiene dibujadas las zonas.

Esto nos permite definir lo siguiente:

- La demanda de origen y destino es inamovible. Esto debido a que no sabemos donde vive exactamente cada usuario, ni donde trabaja o estudia. Por lo tanto, se asume que la demanda de origen y destino es inamovible.
- La demanda de transición (las bajadas y subidas interetapas) pueden cambiar si es que la oferta cambia, es decir, si es que el usuario decide que es mejor hacer transbordo en un paradero P en la zona777 z1 en vez del paradero Q en la zona777 z2. Esto es perfectamente posible. La idea de la redistribución de la demanda propuesta en esta memoria, es mover el trayecto de una persona con dos puntos fijos, el origen y el destino final.

### 2.7.3 Frecuencias de los servicios

Para cada tupla servicio, sentido, variante es necesario definir la frecuencia de esta tupla para cada bin, es decir, la función:

$$f(S, V, d, b)$$

Esta función  $f$  retorna la frecuencia de la tupla  $(S, V, d)$  en el bin temporal  $b$ . Red provee de tablas de frecuencias para todos los servicios TS (es decir, los buses). Esta frecuencia es por hora. Es decir, buses/hora. Si para una tupla  $(S, V, d)$  la frecuencia es 6 buses/hora, esto significa que cada 10 minutos pasa un bus (si asumimos equipartición, que es lo que se asumirá desde ahora hasta que se diga lo contrario). El tiempo está dividido en bins de 30 minutos de ahora en adelante, por lo tanto, tendremos 48 bins en un día, desde el 0 hasta el 47.

#### 2.7.4 Velocidades promedio de los servicios.

Para cada tupla servicio, sentido, variante es necesario definir la velocidad promedio de esta tupla para cada bin  $b$ , es decir, la función:  $v_e(S, V, d, D, b)$

Esta función  $g$  retorna la velocidad promedio de la tupla  $(S, V, d)$  en el bin temporal  $b$ . Notar que al ser promedio, para una tupla, es la misma por todo el recorrido, es decir, no influye la arista por la que circula el servicio.

Ambas tablas (de frecuencias y velocidades) las provee red en su plan de operaciones. Revisar acá: <https://www.dtpm.cl/index.php/programa-de-operacion>

#### 2.7.5 Nodos

Los tipos de nodos que tendrá el grafo son:

##### Paraderos

Cada paradero es un nodo. Cada nodo tiene la siguiente información:

- Código de paradero (TS y Usuario)
- Latitud y longitud (WGS84)
- Nombre del paradero
- Tipo (BUS o Metro)
- Servicios que pasan por el paradero (lista) en cualquier bin  $b$  y día  $D$ .
- Zona 777

##### Servicios

Cada paradero tiene un conjunto de servicios que pasan por él. Por lo tanto, se define un nodo servicio por cada paradero y servicio que pasa por él. Estos nodos nos permiten cerrar la transición entre estar en un paradero y subirse a un servicio.

### 2.7.6 Aristas

Los tipos de aristas que tendrá el grafo son:

#### VIAJAR

Aristas que corren entre nodos *Servicio*. Representan la conexión dirigida entre dos paradas consecutivas de un servicio. Estas aristas tienen la siguiente información:

- Nodo origen (Servicio en paradero P)
- Nodo destino (Servicio en paradero Q)
- Servicio
- Sentido
- Variante (PM o Normal)
- Distancia (m)
- Velocidad promedio (km/h) por bin y tipo de día (en un diccionario)
- Tiempo a bordo (min) por bin y tipo de día (en un diccionario)
- Tipo (BUS o Metro)

Notamos que las aristas VIAJAR tienen peso, el cual es el tiempo a bordo. Estas aristas son temporalmente dependientes.

#### CAMINAR

Aristas que corren entre nodos *Paradero*. Representan la conexión no dirigida entre dos paraderos cercanos. Estas aristas tienen la siguiente información:

- Nodo origen (Paradero P)
- Nodo destino (Paradero Q)
- Distancia (m)
- Tiempo estimado (min). Son iguales para ambas direcciones y son atemporales. Dependen de la velocidad promedio del usuario.

#### SUBIR

Aristas que corren entre nodos *Paradero* y *Servicio*. Representan la acción de subirse a un servicio en un paradero. Estas aristas tienen la siguiente información:

- Nodo origen (Paradero P)
- Nodo destino (Servicio en paradero P)
- Servicio
- Sentido
- Variante (PM o Normal)
- Tiempo de espera (min) por bin y tipo de día (en un diccionario)
- Tipo (BUS o Metro)

## BAJAR

Aristas que corren entre nodos *Servicio* y *Paradero*. Representan la acción de bajarse de un servicio en un paradero. Estas aristas tienen la siguiente información:

- Nodo origen (Servicio en paradero P)
- Nodo destino (Paradero P)
- Servicio
- Sentido
- Variante (PM o Normal)
- Tipo (BUS o Metro)

Estas aristas no tienen coste alguno. Bajarse es inmediato.

Todas las distancias son euclidianas en una geometría geodésica (WGS84). Representan una línea recta en una geodésica desde el punto de inicio al final. No tiene en cuenta la topología de la urbe.

En la figura 2.7 se muestra un ejemplo del grafo no agrupado.

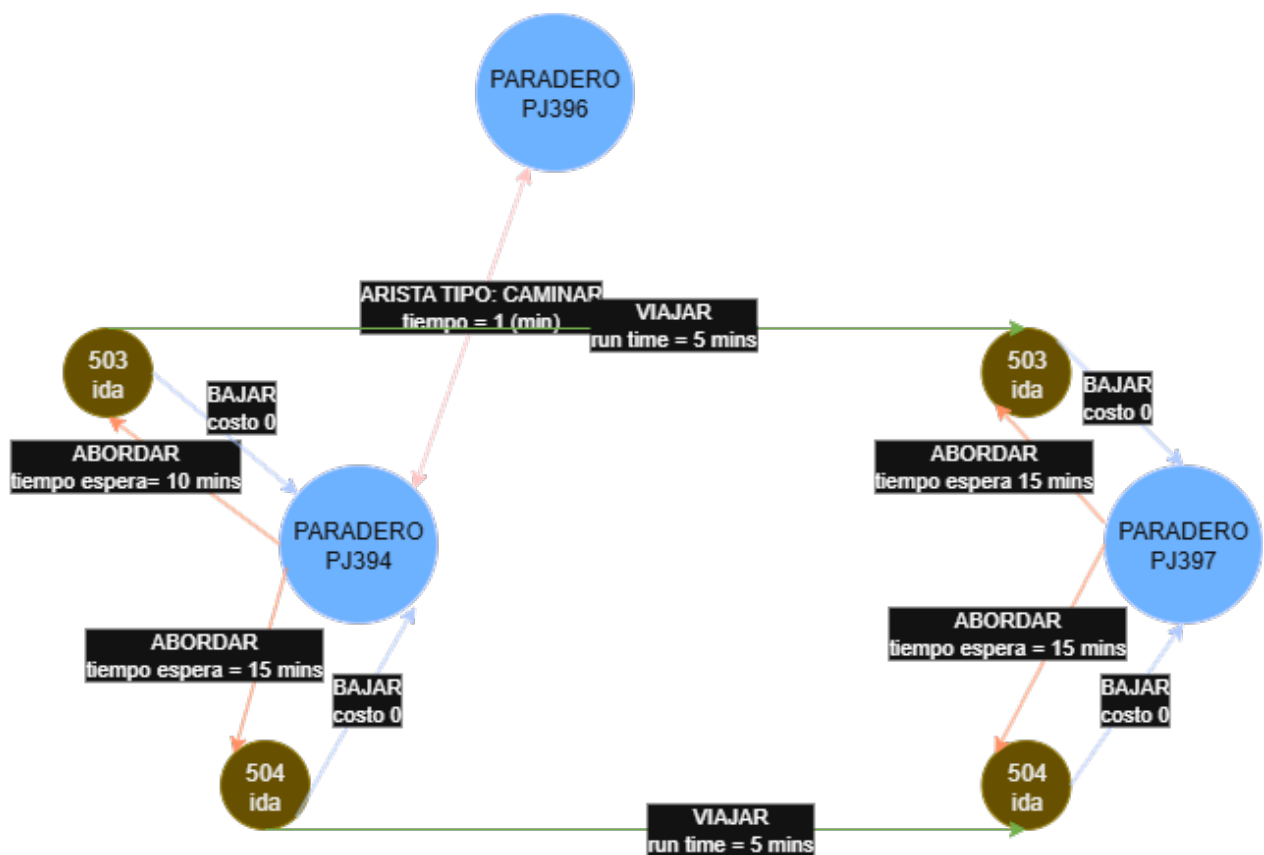


Figura 2.7: Esquema resumen del grafo de Estado

### 2.7.7 Algoritmo para el grafo de Estado

Hacer el grafo de estado es fácil teniendo el grafo agrupado. Un algoritmo recorre cada nodo de tipo paradero y crea :

1. Aristas de tipo CAMINAR entre los k nodos PARADERO mas cercanos. Se definió 10 paraderos a menos de 200 metros. Si un paradero no tiene ningun paradero a menos de 200 metros no tendrá aristas CAMINAR entrantes ni salientes.
2. Nodos de tipo servicio por cada servicio que pase.
3. Aristas de tipo SUBIR a cada nodo SERVICIO con los pesos en un diccionario.
4. Aristas de tipo BAJAR desde cada nodo SERVICIO al PARADERO.

Luego, conecta todas los nodos de tipo servicio con aristas VIAJAR según el recorrido .

Si auditamos el grafo para sanear errores, obtenemos lo siguiente:

=== Estado del grafo de servicio-aware ===

Nodos totales : 60679

- Paraderos : 11588
- A bordo (Servicio) : 49091

Aristas totales : 217305

Aristas por tipo :

- CAMINAR : 70826
- SUBIR : 49091
- VIAJAR : 48297
- BAJAR : 49091

— Problemas detectados (conteos) —

caminar\_missing\_reverse : 0 caminar\_bad\_length\_m : 0 caminar\_bad\_runtime\_format : 0  
subir\_missing\_wait\_series : 0 subir\_bad\_wait\_series : 0 subir\_orphan\_pairs : 0 bajar\_bad\_cost  
: 0 viajar\_missing\_run\_series : 0 viajar\_bad\_run\_series : 0 viajar\_service\_mismatch : 0  
bnodes\_without\_viajar : 10

Muestras (si existen):

bnodes\_without\_viajar:

```
('b', 'E-5-42-OP-5', '107c', 'Ida', 'normal')
('b', 'L-12-24-15-NS', '101', 'Ida', 'normal')
('b', 'T-4-24-PO-15', '101', 'Ret', 'normal')
('b', 'E-7-53-PO-50', 'I04', 'Ida', 'normal')
('b', 'L-33-95-10-SN', 'H14', 'Ret', 'normal')
```

Obtenemos un grafo muy útil. Por ejemplo, ya con este grafo con pesos podemos correr un algoritmo de Dijkstra para encontrar la ruta mas corta entre dos paraderos. Notar que esta ruta mas corta es teniendo en cuenta que todos los pesos “pesan” lo mismo, es decir, da lo mismo recorrer 15 minutos caminando, que en bus o metro, ni que un minuto de espera vale lo mismo que un minuto a bordo. Esto es lo que tenemos que descubrir viendo los parámetros, en este caso, del MNL.

Este grafo tiene toda la información de la OFERTA de transporte. Junto con las tablas de etapas y viajes tenemos la DEMANDA.

Recordar que el objetivo es tener un grafo de estado artificial con OFERTA ARTIFICIAL y obtener, en base a tablas de etapas y viajes reales, DEMANDA ARTIFICIAL al tener modelos de elección y de grafos que las generen .

# Capítulo 3

## MNL

### 3.1 Marco Teórico

El modelo MNL (Multinomial Logit Model) es un modelo de elección discreta que se utiliza para predecir la probabilidad de que un individuo elija una alternativa dentro de un set de ellas.

Por ejemplo, si un usuario tiene 3 *alternativas* de servicio, sean S1, S2, Y S3 en un paradero de origen P y un destino Q, el modelo MNL nos permite predecir la probabilidad de que el usuario elija cada una de las alternativas en base a variables propuestas como decidoras por el propio ingeniero.

Algunas variables propuestas pueden ser:

- Tiempo de viaje
- Tiempo de caminata
- Número de transbordos (o indirectamente el tiempo de caminata)
- Tipo de transporte (Bus o Metro)

Notar que todas estas variables son *atributos* de la alternativa.

Una Utilidad  $U_n$  se define como la suma de la la parte determinística (los atributos) y una parte aleatoria  $\epsilon_n$  que captura la incertidumbre del modelo.

$$U_n = V_n + \epsilon_n$$

$V_n$  se construye como una función predictora lineal de los atributos ponderados por coeficientes  $\beta_i$  que representan la importancia de cada atributo en la decisión del usuario. Es decir:

$$V_n = \beta_1 x_{1n} + \beta_2 x_{2n} + \dots + \beta_k x_{kn} = \sum_{i=1}^k \beta_i x_{in}$$

Cada beta es un peso que pondera la importancia del atributo  $x_i$  en la decisión del usuario. Por ejemplo, si  $\beta_1$  es muy grande, el atributo  $x_1$  es muy importante en la decisión del usuario. Si  $\beta_2$  es negativo, el atributo  $x_2$  tiene un efecto negativo en la decisión del usuario.

Ejemplos de atributos  $x_i$  pueden ser el tiempo de viaje, el tiempo de caminata, el coste que queda después de tomar el servicio y bajarse en el paradero óptimo, etc.

Estos atributos son los mencionados anteriormente, los cuales definen un vector el cual pondera con un producto lineal los atributos con los coeficientes  $\beta_i$ . Esto genera una probabilidad de que la alternativa sea elegida, dada la fórmula:

$$P_{ni} = \frac{e^{V_{ni}}}{\sum_j e^{V_{nj}}}$$

En el ámbito de predicción de demanda en transporte público, el modelo MNL se es conveniente pues permite incorporar múltiples factores que influyen la decisión final.

## 3.2 Modelo de Regresión Lineal simple

### 3.2.1 Dataset de entrenamiento

Un Modelo de regresión lineal básico fue el primero en tener en cuenta. Este modelo solo tenía en cuenta el primer abordaje, es decir, el primer servicio que tomaba el usuario en su viaje. No importaba si el usuario hacía transbordos o no y hacia donde fuera.

La receta para construir el dataset de entrenamiento es el mas complicado. El *pipeline* es el siguiente:

1. Por cada fila en la tabla de viajes, obtener el paradero de origen, el servicio tomado y la hora.
2. Expandir esta tabla de decisiones con las alternativas posibles disponibles para el usuario en ese paradero y bin. Esta expansión se hizo de la siguiente forma:
  - Obtener todas las aristas SUBIR que salgan del paradero en cuestión y que tengan tiempos de espera no infinitos.
  - Estas aristas proveen los servicios que el usuario puede tomar.
  - Extraer de cada alternativa la velocidad promedio y el tiempo de espera. Estos son atributos relevantes para el modelo.
3. Crear la variable dependiente, que es 1 si el servicio es el que tomó el usuario y 0 en caso contrario.
4. Entrenar el modelo.



### 3.2.2 Algoritmo de entrenamiento:

Realmente, este no es un MNL, sino una regresión lineal.

- Se eliminan valores o filas corruptas (se eliminan las decisiones mal formadas).
- Se construyen las características. Estas son:
  - **Tiempo de espera** (wait\_time):

$$\text{wait\_time} = 0.5 \times \left( \frac{30}{\text{freq}_h} \right)$$

- **Velocidad** (speed\_kmh): velocidad promedio del servicio en el bin y tipo de día correspondiente (más atractivo).
- Se limpian infinitos o nulos.
- Se construyen las matrices  $X$  e  $y$  con las características y la variable dependiente.
- Se ajusta un modelo binario con `sklearn.linear_model.LogisticRegression`:

$$P(\text{abordar} = 1 \mid \text{alternativa}) = \sigma(\beta_0 + \beta_1 \text{wait\_time} + \beta_2 \text{speed\_kmh})$$

donde  $\sigma(z) = \frac{1}{1+e^{-z}}$ .

- Se reconstruye la utilidad:

$$U = \beta_0 + \beta_1 \text{wait\_time} + \beta_2 \text{speed\_kmh}$$

- Se agrupa por decision\_id y se aplica softmax estable:

$$P_i = \frac{\exp(U_i - \max U_{\text{set}})}{\sum_j \exp(U_j - \max U_{\text{set}})}$$

(Esto simula un MNL post hoc).

- **Top-1 accuracy**: porcentaje de decisiones donde la alternativa con mayor  $P_i$  coincide con  $\text{is\_abordado} = 1$ .
- **Log-likelihood**: suma de  $\log(P_i)$  de la alternativa elegida.
- **Modelo nulo**: probabilidad uniforme  $1/K_d \Rightarrow \text{loglik}_{\text{null}} = -\sum_d \log(K_d)$ .
- **McFadden pseudo- $R^2$** :

$$R^2 = 1 - \frac{\text{LL}}{\text{LL}_{\text{null}}}$$

### 3.2.3 Resultados

El modelo se entrenó con 8 millones de decisiones de los 7 días de la semana. Al expandirlo en alternativas, obtenemos 47 millones de decisiones. La figura 3.1 muestra el resumen de los datos de entrenamiento del modelo.

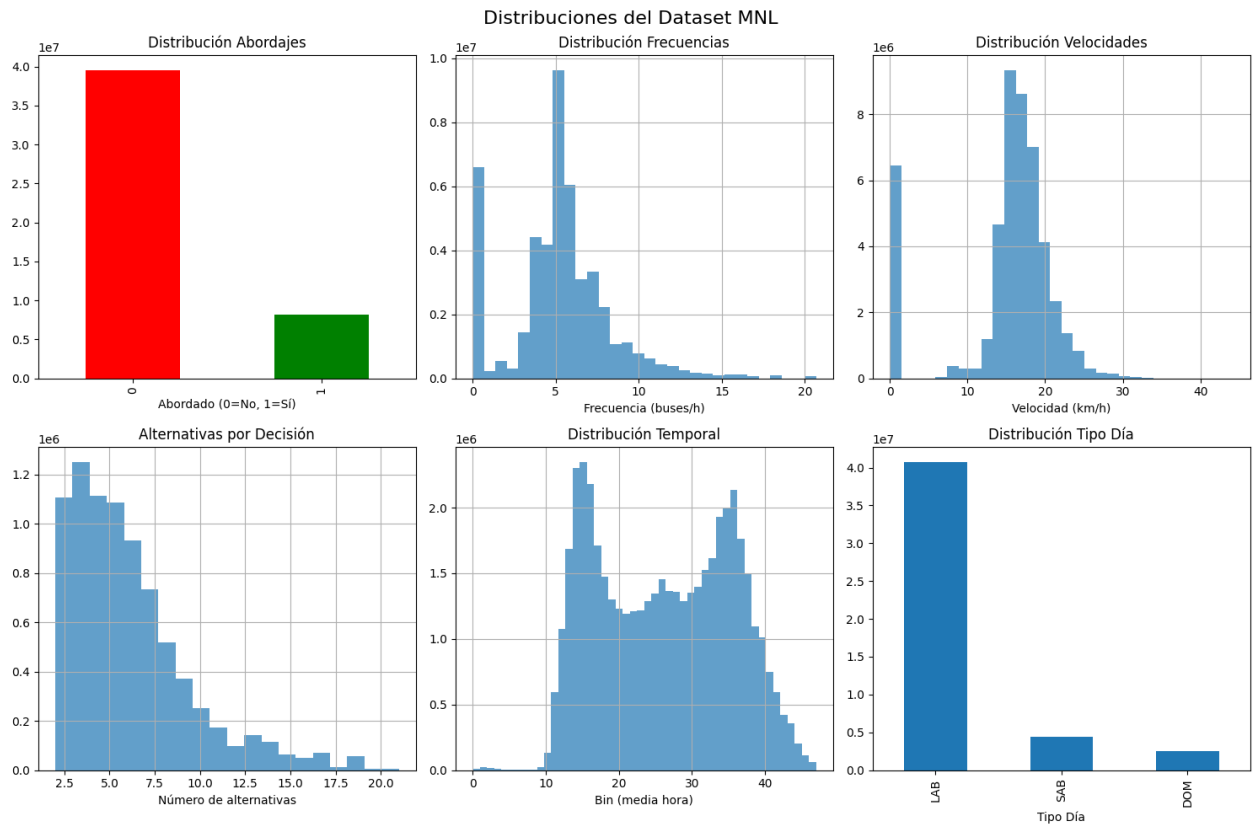


Figura 3.1: Resumen del dataset para el modelo MNL básico

Nos damos cuenta que en promedio hay 5 decisiones por abordaje. También, vemos claramente la distribución de bins (notar las horas punta), las frecuencias (notar las frecuencias 0 que denotan servicios inactivos) y velocidades como una gaussiana centrada en 16 km/h con outliers en 0, mostrando a los servicios inactivos.

Los resultados se muestran a continuación:

## DATOS PARA ENTRENAMIENTO

Filas iniciales: 47,667,498  
Decisiones válidas: 8,163,936 de 8,163,936

## FEATURES

Filas eliminadas por NaN: 6,454,427  
wait\_time: min=0.73, mean=2.95, max=82.50  
speed\_kmh: min=6.95, mean=17.31, max=44.21  
Matriz X: (41213071, 2)  
Vector y: (41213071,)  
Ratio de abordaje: 0.194

## RESULTADOS MODELO:

### COEFICIENTES

$\beta_0$  (intercept): -1.497097  
 $\beta_1$  (wait\_time): -0.211864  
 $\beta_2$  (speed\_kmh): 0.038037

## EVALUACIÓN MNL POR CONJUNTO

Top-1 accuracy (por decisión): 0.3500  
Log-Likelihood: -11,455,639  
McFadden pseudo- $R^2$ : 0.0361

## INTERPRETACIÓN ECONÓMICA

Trade-off: 1 min espera  $\approx$  5.57 km/h velocidad

### ELASTICIDADES (en medias):

Espera: -0.5043 (% cambio prob por % cambio espera)  
Velocidad: 0.5306 (% cambio prob por % cambio velocidad)

### SENSIBILIDAD:

+1 min espera  $\rightarrow$  -19.09% cambio en odds  
+1 km/h velocidad  $\rightarrow$  +3.88% cambio en odds

El modelo, para ser simple, sorprendentemente tiene una accuracy mejor que el azar, teniendo en cuenta que en promedio hay 5 decisiones. Se decide seguir explorando el MNL pero ahora teniendo en cuenta el destino de la persona. Esto debería de subir considerablemente las métricas.

## 3.3 MNL con destino

### 3.3.1 Dataset

Para el modelo MNL con destino, se utilizó la tabla de etapas, ya que ya venía disgregada y nos permitía manejar con mayor facilidad los datos.

Los pasos para generar el dataset fueron los siguientes:

1. De la tabla de etapas, obtenemos todos los viajes que tienen bajada registrada gracias a ADATRAP.
2. Por cada fila obtenemos el paradero de origen, el servicio tomado, el bin, el paradero de bajada observado, el tipo de día y el tipo de servicio. Agregamos el destino final para cada etapa gracias a agrupar las etapas con el mismo ID.

En este punto, tenemos un dataset muy parecido al anterior, pero con la información del destino. Lógicamente, la información del destino enriquece la cantidad de atributos observables que introduciremos al MNL. Entre ellos, el costo restante.

#### Costo Restante

El costo restante es la medida en tiempo que nos da al bajarnos en el paradero óptimo y los transbordos que le preceden. Un ejemplo del día a día que el autor de la memoria pasa todos los días:

- En el paradero PJ394 tengo las siguientes alternativas a las diez de la mañana un día LABORAL: 503, 504, 507, 517, 518 , B38.
- Mi destino lógicamente es Beauchef. Elegiremos el paradero mas cercano a Beauchef, el PA433.
- Por suerte, ahí también para la 507, así que el costo restante es cero, pues después de bajarme en la parada óptima, ya llegué a mi destino.
- Para los otros servicios, el costo restante es mayor que cero, ya que ninguno me deja directamente en PA433. Entonces, tenemos que calcular el costo restante desde el paradero de bajada óptimo.

**Dijkstra Inverso** Para calcular el paradero óptimo y el costo restante al bajarse en ese paradero es importante la noción del Algoritmo de Dijkstra (AD). A groso modo, el AD es un algoritmo que funciona de la siguiente manera:

- Se parte en un nodo origen y se le asigna un costo 0.
- Se exploran todos los nodos vecinos y se les asigna un costo igual al peso de la arista que los conecta con el nodo origen.

- Se marca el nodo origen como visitado.
- Se selecciona el nodo no visitado con el costo más bajo y se repite el proceso hasta que todos los nodos hayan sido visitados o se haya alcanzado el nodo destino.

En nuestro caso, el AD se corre en sentido inverso, es decir, partimos del nodo destino y vamos hacia atrás. De esta forma, obtenemos el costo mínimo para llegar al destino desde cualquier otro nodo.

Entonces, para un paradero de destino, un bin y un día obtenemos una lista enorme de costos restantes para cada paradero de la red. Notar que es costoso ejecutar este algoritmo en un grafo tan grande, así que hay que ejecutar estrategias para evitar el sobrecoste.

Por ejemplo, podemos correr el AD para PA433 y el costo restante para ir desde cada paradero hasta PA433 es el costo de viajar. Como separamos los servicios (es decir, el grafo no es agrupado), cada camino es una combinación de aristas. Lo bueno de este enfoque, es que penaliza fuertemente los transbordos, haciendo más realistas los caminos.

Entonces, obtenemos un camino  $C$  que tiene de extremos dos nodos PARADERO y una cantidad par de aristas SUBIR + BAJAR y un número arbitrario de aristas VIAJAR visitadas. Esto es, el costo restante tiene el costo de los transbordos, esperas, tiempo a bordo y todo lo incluido.

Si yo ejecuto AD para mi origen y destino, obtendré paraderos de bajada óptimos para cada alternativa. Estos son , el destino para el 507, y paraderos de transbordo para los otros.

Siguiendo con el pipeline...

3. Se expande esta tabla de decisiones con las alternativas posibles disponibles para el usuario en ese paradero y bin. Esta expansión se hizo de la siguiente forma:

- Por cada decisión, se obtienen todos los servicios disponibles.
- Por cada alternativa, incluida la elegida, se extraen desde el grafo de estado el tiempo de espera en ese paradero y bin.
- Por cada alternativa que NO sea la decisión, se ejecuta el AD Inverso desde el paradero de destino FINAL . El paradero óptimo es el que minimiza el costo restante dentro del perfil de paraderos del servicio, siendo el perfil de paraderos una lista de paraderos que le preceden al paradero actual .
- Por cada alternativa, se calcula el costo de esperar el servicio mas el costo de viajar al paradero óptimo (o el de bajada real en el caso del usado) y el costo restante ya obtenido por Dijkstra.
- Se agregan todos estos atributos a la fila por cada alternativa.

Ejecutar este código a primeras veces fue un dolor de cabeza. Era extremadamente lento en las primeras iteraciones (miles de años, literalmente). Muchas optimizaciones fueron hechas, que nos permitió aplicar muchas técnicas de caché y de cursos teóricos de la carrera.

- Se agruparon todos los viajes que iban al mismo destino. Con ello, se calculaba solo una vez el algoritmo de dijsktra para muchas decisiones a la vez, y estos resultados se cacheaban.

Al ver el código en ejecución, notamos algo interesante. Al comenzar computando el AD para el grupo mas grande, inmediatamente comenzó a construir el grafo inverso para el paradero METRO TOBALABA, y es que gran cantidad de los viajes tenían como destino final este paradero.

- Se cacheo el perfil de cada servicio.

Con ello, una tabla de etapas de un día (300K decisiones) se podía procesar en 2 horas solamente.

### 3.3.2 Entrenamiento

El entrenamiento dio los siguientes resultados (#TODO: AHONDAR EN COMO FUE EL ENTRENAMIENTO)

```
Eval Train: {'loglik': nan, 'loglik_null': -178245.6990444304, 'pseudo_r2_mcfadden': nan,
'top1_acc': 0.435096741781741, 'n_decisions': 151279, 'n_alternatives': 701136} Eval Val
: {'loglik': nan, 'loglik_null': -44693.38604960674, 'pseudo_r2_mcfadden': nan, 'top1_acc':
0.4328279219920087, 'n_decisions': 37791, 'n_alternatives': 175648}
```

Esto nos da los coeficientes :

```
{ "col_order": [ "intercept", "wait_time", "viajar_cost", "cost_to_go", "zero_onboard",
"ASC_metro" ], "coef": [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ] }
```

Es decir, fallamos. La respuesta a esto está en la figura 3.2.

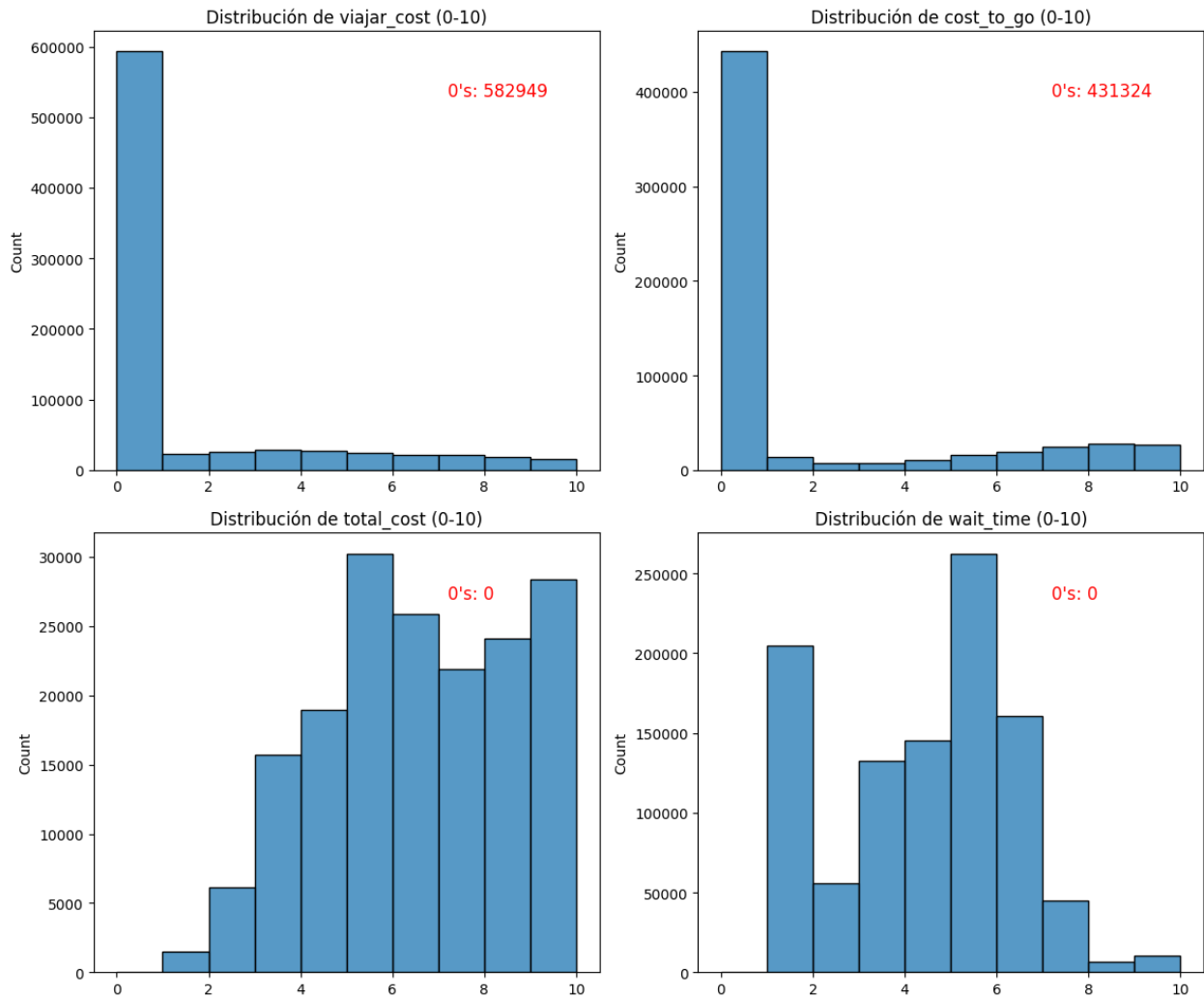


Figura 3.2: Histograma de Pesos

La cantidad de 0's en *viajar\_cost* y en *cost\_to\_go* tienen distintas razones.

Para *viajar\_cost*, el coste de viajar es 0 cuando el usuario llega a un paradero, y una de las alternativas decide en hacer transbordo a otro paradero, ya que la ruta mas corta comienza en ese paradero.

Para *cost\_to\_go* 0, es cuando es necesario solo una etapa para completar el viaje. Esto no es problema que sea 0.

### 3.3.3 Primeros Transbordos y la Penalización Inicial

Una solución elegida fue penalizar a las alternativas que requieran hacer transbordo, con un tiempo extra que se pueda configurar, esto por dos razones:

- Es mas simple y directo.

- No es necesario que Dijkstra retorne el camino completo para verificar hacia donde fue.

Una idea interesante podría haber sido guardar el camino completo hecho por dijkstra (y no solo el peso) y cuando hayan primeros transbordos, guardar el coste de caminar al paradero óptimo, pero esto es mas complicado, consume mas memoria, y además era necesario de ya entrenar apara avanzar con la memoria. Esta idea se descartó.

En primera instancia, se decide entrenar con una penalización de 5 minutos a las alternativas que requieran hacer transbordo desde el paradero inicial. Entonces, obtenemos los siguientes resultados:

A continuación se presentan los resultados del entrenamiento del modelo MNL con destino, organizados en tablas para mayor claridad.

	Filas	Decisiones
Datos cargados	1,049,845	–
Después de limpieza	876,784	–
Train	702,054	151,279
Val	174,730	37,791

Tabla 3.1: Resumen de datos y partición

Iteración	nll	$ \mathcal{V} $	valR <sup>2</sup>	acc	Tiempo (s)
pre	178,528	$1.40 \times 10^5$	–	–	–
1	96,483	$4.17 \times 10^4$	0.456	0.761	–
2	80,694	$2.66 \times 10^4$	0.544	0.778	–
3	62,125	$1.29 \times 10^4$	0.649	0.919	–
4	53,277	$6.65 \times 10^3$	0.700	0.919	–
5	49,019	$2.95 \times 10^3$	0.725	0.919	–
6	47,437	$1.45 \times 10^3$	0.735	0.920	–
7	46,988	$6.96 \times 10^2$	0.738	0.920	–
8	46,907	$9.63 \times 10^1$	0.739	0.921	–
9	46,903	$3.93 \times 10^1$	0.739	0.921	–
10	46,901	$1.59 \times 10^1$	0.739	0.921	–
11	46,901	6.74	0.739	0.921	–
12	46,901	6.74	0.739	0.921	–

Tabla 3.2: Progreso del entrenamiento



Coeficiente	Valor
intercept	$5.51 \times 10^{-13}$
wait_time	-0.787
viajar_cost	-0.039
cost_to_go	-5.36
first_walk_min	-0.088
is_initial_transfer	-0.177
zero_onboard	1.979
ASC_metro	$1.60 \times 10^{-13}$

Tabla 3.3: Coeficientes del modelo

	Train	Val
Log-likelihood	-46,901	-11,593
Log-likelihood nulo	-178,528	-44,411
Pseudo- $R^2$ McFadden	0.737	0.739
Top-1 accuracy	0.922	0.921
Decisiones	151,279	37,791
Alternativas	702,054	174,730

Tabla 3.4: Evaluación del modelo

Éxito	True
Iteraciones	12
NLL final	46,901.33
Tiempo total (s)	717.3

Tabla 3.5: Resumen de entrenamiento

Si revisamos las variables y sus pesos, notamos algo extraño.

- wait\_time negativo: Bien
- viajar\_cost negativo: Bien (pero no tan negativo, ojo con eso)
- cost\_to\_go: altamente negativo, afecta a la utilidad
- zero\_onboard: positivo, pues hay colinealidad entre el y variable de transbordo.
- asc\_metro: Nada, muy cercano a 0.

Si corremos el mismo entrenamiento para otros días, por ejemplo, para el día Jueves de la misma semana, obtenemos:

	Filas	Decisiones
Datos cargados	1,063,814	–
Después de limpieza	885,991	–
Train	708,649	154,113
Val	177,342	38,444

Tabla 3.6: Resumen de datos y partición

Iteración	nll	$ \nabla $	valR <sup>2</sup>	acc	Tiempo (s)
pre	180,068	$1.42 \times 10^5$	–	–	–
1	96,929	$4.20 \times 10^4$	0.461	0.769	–
2	81,202	$2.70 \times 10^4$	0.548	0.785	–
3	62,311	$1.32 \times 10^4$	0.654	0.924	–
4	53,052	$6.87 \times 10^3$	0.706	0.924	–
5	48,344	$3.08 \times 10^3$	0.733	0.925	–
6	46,522	$1.53 \times 10^3$	0.743	0.927	–
7	46,025	$9.46 \times 10^2$	0.746	0.928	–
8	45,919	$1.17 \times 10^2$	0.747	0.928	–
9	45,914	$4.21 \times 10^1$	0.747	0.928	–
10	45,912	$1.80 \times 10^1$	0.747	0.928	–
11	45,912	5.41	0.747	0.928	–
12	45,912	5.41	0.747	0.928	–

Tabla 3.7: Progreso del entrenamiento

Coefficiente	Valor
intercept	$5.47 \times 10^{-13}$
wait_time	–1.003
viajar_cost	0.128
cost_to_go	–5.50
first_walk_min	–0.125
is_initial_transfer	–0.249
zero_onboard	2.179
ASC_metro	$1.63 \times 10^{-13}$

Tabla 3.8: Coeficientes del modelo

	Train	Val
Log-likelihood	-45,912	-11,437
Log-likelihood nulo	-180,068	-45,159
Pseudo- $R^2$ McFadden	0.745	0.747
Top-1 accuracy	0.930	0.928
Decisiones	154,113	38,444
Alternativas	708,649	177,342

Tabla 3.9: Evaluación del modelo

Éxito	True
Iteraciones	12
NLL final	45,912.15
Tiempo total (s)	913.5

Tabla 3.10: Resumen de entrenamiento

Obtenemos constantes positivas en el coste de viajar. Una colinealidad entre el coste restante (cost to go) y el tiempo de viajar puede ser una señal de esto. Si lo miramos desde un punto de vista de comodidad, un coste restante menor indica que el viaje tiene menos transbordos probablemente y es mas directo. Entonces, un coste restante menor es mas atractivo. Para tener un costo restante menor, es necesario viajar mas tiempo en el primer servicio.

Además, el intercepto no es necesario. Esto porque al calcular la probabilidad (de manera relativa) los interceptos se cancelan entre ellos ya que son todos iguales para todos los casos. Por lo tanto, un nuevo trainer es necesario.

# Capítulo 4

## GNN

Para comenzar a hablar de las GNN, es pertinente aclarar conceptos de redes neuronales y como nos ayudarán en el futuro a solucionar el problema.

### 4.1 Redes Neuronales

Las redes neuronales pueden ser pensadas como una colección de nodos o neuronas conectados por aristas o axones, los cuales tienen pesos. Estos nodos o neuronas se agrupan según su función en capas, las cuales pueden estar encargadas de recibir la información, procesarla o dar la respuesta a la pregunta. Por ejemplo, un clasificador binario tiene una capa de output con sola una neurona. Si la neurona se activa o no depende de la clasificación final.

El ajuste de los pesos se hace en el entrenamiento, proceso en el cual la red “aprende” en base a fallar una y otra vez en base a datos etiquetados y a funciones de pérdida que el algoritmo debe de minimizar. La forma que tiene la red de ajustar los pesos depende la tasa de aprendizaje (learning rate) o la función usada para optimizar o converger a la red (optimizer).

### 4.2 Redes Neuronales de Grafos

Las redes neuronales tradicionales (MLP, CNN, RNN/LSTM) están preparadas para recibir datos en forma de vectores, rejillas (grafos en forma de grilla, como imágenes) o secuencias temporales (grafos dirigidos en solo una dirección). Es decir, podemos pensar en una GNN como una abstracción o generalización de muchos tipos de redes neuronales. Técnicamente, una GNN puede recibir una imagen, pues una imagen es un grafo cuadriculado.

Se nos vienen a la cabeza miles de estructuras de datos que pueden aprovechar a las GNN para realizar predicciones. Redes sociales, moléculas, redes de transporte, entre otras.

### 4.2.1 ¿Cómo funcionan las GNN?

La idea de las GNN es la siguiente:

- Cada nodo tiene un vector de características inicial (atributos).
- Los nodos “envían mensajes” a sus vecinos.
- Cada nodo actualiza su representación basado en los mensajes recibidos.
- Este proceso se repite varias veces (capas de la GNN).

En el caso del transporte público, tenemos:

- Cada parada tiene su información propia (ubicación, servicios, tipo, información relevante del entorno).
- En la primera iteración, el nodo recibe información de los nodos que están conectados directamente a él.
- En la segunda, a dos pasos del nodo.
- Así sucesivamente.

### 4.2.2 Algunos ejemplos

#### STGCN (Spacio Temporal Graph Convolutional Network)

Trabajo realizado por Jin [4] uso una combinación de redes convolucionales para la predicción temporal y una Cluster-GCNN para la predicción espacial. El set de datos fue de Metro de Shanghai.

#### GNN for Robust Public Transit Demand Prediction usando PGCN

Un trabajo interesante de Li [6] abordó la partición de la ciudad de Sidney en zonas de acuerdo a su código postal (nodos) y agregó información del uso del suelo a los nodos (Densidad residencial, Oficinas, Industrias, Zonas Comerciales, etc). Por el lado de las GNN, utilizó una PGCN (Probabilistic Graph Convolutional Network) para predecir la demanda entre los pares (O,D). Esta red se compone de correlaciones espacio temporales mas un módulo de aproximación bayesiana para cuantificar la incertidumbre.

## 4.3 Reconstrucción de Demanda sin contexto temporal.

El reconstrucción de demanda, se refiere al proceso en el cual, dado un tiempo  $T$ , datos históricos  $X(T)$  y un grafo  $G(T)$  con características espaciales concretas, predecir la demanda en ese instante de tiempo. Ese instante puede ser igualmente una ventana de tiempo corta (5 minutos), o un período (hora punta entre las 17:30 y las 19:30). La versatilidad de este enfoque esta en la obtención de una abstracción espacial, que nos permitirá, a priori, cambiar la topología sustancialmente.

Algunas características que podrían tener los nodos, son:

- Servicios.
- Ubicación.
- Tipo (metro o bus).
- Si es combinación.
- Comuna.
- Frecuencia de los recorridos.
- Uso del suelo cercano.
- Paraderos cercanos a una distancia X.
- Tiempo actual del grafo (la demanda no es invariante temporalmente)

Las aristas tendrían...

- Tiempo promedio en recorrerla.
- Distancia entre nodos.
- Servicios que la recorren.
- Tipo (bus o metro)

Con ello, las aristas tendrían PESO, algo importante para detectar que camino utilizarían los usuarios.

Debido a que no tenemos el contexto temporal como input al modelo, es importante entrenar con muchos datos de distintos días y horas. Además, datos del uso de suelo pueden ser claves al detectar por qué una parada es mas usada que otra. Datos de densidad poblacional también lo pueden ser.

#### **4.3.1 Flujo de trabajo**

- Intentar obtener la mayor cantidad de datos de uso de suelo posibles para enriquecer la información de los nodos.
- Agregar datos de tiempo/distancia a las aristas.
- Entrenar al modelo en instantes de tiempo dados.
- Comprobar la eficacia del modelo comparándola con configuraciones topológicas ya existentes.
- Probar a predecir en configuraciones topológicas artificiales en las que se agregan servicios.
- Probar a predecir en configuraciones topológicas artificiales en las que se quitan servicios.

### **4.4 GNN con contexto temporal.**

Otro enfoque a tomar será el de predecir la demanda del TP usando datos históricos de tiempos anteriores como input al modelo, para predecir el tiempo siguiente. Varios trabajos nombrados

en este informe hicieron eso. La diferencia será en que en el tiempo siguiente se cambiará la topología.

Luego, el input del modelo sería una secuencia de demandas para un paradero  $P$  y servicio  $S$ , con  $D(t_k, P, S)$  una secuencia de demandas temporales. Una red neuronal apta para captar correlaciones temporales y series de tiempo tomará esta información y predecirá la demanda para el mismo paradero, servicio pero para un tiempo futuro.

#### **4.4.1 Flujo de trabajo**

- Crear atributos de demanda en cada nodo
- Validar modelo con datos históricos
- Con el modelo validado, agregar servicios.
- Quitar servicios.

Podemos pensar en agregar a los servicios nuevos con demanda vacía o con una demanda base tomando en cuenta servicios parecidos con paraderos parecidos. Luego, dejar evolucionar el modelo en el tiempo.

### **4.5 Validación del modelo.**

#### **4.5.1 Reconstrucción de Demanda sin contexto temporal**

Al momento de escribir este informe, (Finales de Junio/Inicios de Julio de 2025) se han agregado servicios nuevos los cuales aun no registran demanda. Se puede ver mas información de ello en <https://www.red.cl/red-comunica/ajuste-en-malla-de-recorridos-de-buses-dos-nuevos-servicios-y-cuatro-extensiones-comienzan-a-funcionar-este-sabado-05-de-julio/> .

Esta es una oportunidad perfecta para agregar el trazado de los recorridos nuevos, calcular la demanda esperada, y cuando red saque los datos de demanda de los nuevos recorridos (las tablas de viaje), podremos contrastar. Otra opción es viajar a atrás en el tiempo, analizar una tabla de datos antigua (anterior a la agregación de nuevos recorridos) y compararla con la actual (con los recorridos ya circulando).

#### **4.5.2 GNN con contexto temporal**

La validación de este caso es mas sencilla. Simplemente podemos comparar datos históricos en tiempos posteriores. (Por ejemplo, en base a datos históricos del día miércoles, predecir el día Jueves a las 5 de la tarde). Naturalmente, tenemos los datos de ese día jueves, por lo que solo es necesario comparar.

En ambos casos, al ser valores discretos numéricos muy variados, comparar con variables estadísticas globales como el promedio o la desviación estándar será razonable para establecer la demanda en paraderos.



# Bibliografía

## Anexos

- [1] Cayul, L.H.C. 2017. *Desarrollo y aplicación de modelo de simulación basada en agentes a gran escala para la ciudad de santiago*. Universidad de Chile.
- [2] Dirección de Transporte Público Metropolitano 2024. Modelos de demanda. <https://dtpm.cl/index.php/documentos/modelos-de-demanda>.
- [3] Jiang, Q. 2022. GMM clustering based on WOA optimization and space-time coupled urban rail traffic flow prediction by CEEMD-SE-BiGRU-AM. *Mobile Information Systems*. 2022, 1 (2022), 7846630.
- [4] Jin, S., Jing, C., Wang, Y. and Lv, X. 2022. SPATIOTEMPORAL GRAPH CONVOLUTIONAL NEURAL NETWORKS FOR METRO FLOW PREDICTION. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. XLIII-B4-2022, (Jun. 2022), 403–409. DOI:<https://doi.org/10.5194/isprs-archives-XLIII-B4-2022-403-2022>.
- [5] Kang, L., Liu, H., Chai, M. and Lv, J. 2020. A LSTM-based passenger volume forecasting method for urban railway systems. *Robotics and rehabilitation intelligence: First international conference, ICRRI 2020, fushun, china, september 9–11, 2020, proceedings, part i 1* (2020), 368–380.
- [6] Li, C., Bai, L., Liu, W., Yao, L. and Waller, S.T. 2022. Graph neural network for robust public transit demand prediction. *IEEE Transactions on Intelligent Transportation Systems*. 23, 5 (2022), 4086–4098. DOI:<https://doi.org/10.1109/TITS.2020.3041234>.
- [7] Li, L., Xu, J., Ng, S.T., Zhang, J., Zhou, S. and Yang, Y. 2020. Attention-based graph neural network enabled method to predict short-term metro passenger flow. *2020 5th international conference on universal village (UV)* (2020), 1–6.
- [8] Li, W., Zhou, M., Dong, H., Wu, X. and Zhang, Q. 2021. Forecast of passenger flow of urban rail transit based on the DNNC model. *2021 33rd chinese control and decision conference (CCDC)* (2021), 4615–4620.
- [9] Li, Y., Zhang, J., Wang, J. and Wang, Y. 2022. Deep learning-based short-term traffic flow prediction considering spatial-temporal correlation. (2022). DOI:<https://doi.org/https://doi.org/10.1049/itr2.12018>.
- [10] Liu, L., Chen, J., Wu, H., Zhen, J., Li, G. and Lin, L. 2020. Physical-virtual collaboration modeling for intra-and inter-station metro ridership prediction. *IEEE Transactions on Intelligent Transportation Systems*. 23, 4 (2020), 3377–3391.
- [11] Massobrio, R. and Nesmachnow, S. 2020. Urban mobility data analysis for public transportation systems: A case study in montevideo, uruguay. *Applied Sciences*. 10, 16 (2020), 5400.
- [12] Ramírez, Á.E.T. 2020. *Análisis espacial de los impactos en la demanda de transporte público producto de una nueva línea de metro utilizando datos masivos*. Universidad de Concepción.
- [13] SmartcitySantiagoChile 2025. ADATRAP: Herramienta para análisis de datos masivos de transporte público.
- [14] Soto, F.J.M. 2023. *Estimación y análisis de modelos de demanda agregada para el transporte público en santiago de chile*. Universidad de Chile.

- [15] Torrepadula Franca, R. di et al. 2024. Machine learning for public transportation demand prediction: A systematic literature review. *Engineering Applications of Artificial Intelligence*. (2024). DOI:<https://doi.org/https://doi.org/10.1016/j.engappai.2024.109166>.
- [16] Wang, Y., Yin, H., Chen, T., Liu, C., Wang, B., Wo, T. and Xu, J. 2021. Passenger mobility prediction via representation learning for dynamic directed and weighted graphs. *ACM Transactions on Intelligent Systems and Technology (TIST)*. 13, 1 (2021), 1–25.
- [17] Wei, J., Cheng, Y., Chen, K., Wang, M., Ma, C. and Hu, X. 2022. Nonlinear model-based subway station-level peak-hour ridership estimation approach in the context of peak deviation. (2022). DOI:<https://doi.org/https://doi.org/10.1177/03611981221075624>.
- [18] Ye, J., Xu, Z. and Gou, X. 2022. An adaptive grey-markov model based on parameters self-optimization with application to passenger flow volume prediction. *Expert Systems with Applications*. 202, (2022), 117302.
- [19] Zhao, X., Guan, H., Sun, H. and Lu, J. 2022. A prophet-based passenger flow prediction model on IC card data. *2021 6th international conference on intelligent transportation engineering (ICITE 2021)* (Singapore, 2022), 1082–1092.