



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

Resumen

Estimar el efecto de cambios de demanda en el transporte público de Santiago de Chile, debido a cambios de oferta es un trabajo difícil ya que cambios en frecuencias o de topología de la red de transporte provocan efectos en cadena complicados de predecir. Es por ello que se propone una solución en base a modelos de decisión discreta y redes neuronales de grafos para simular estos datos de demanda sintéticos, y así tomar decisiones con mejores datos.

Para ello, se utilizaron los datos de viajes proporcionados por Red Metropolitana de Movilidad, enriquecidos por ADATRAP para modelar la elección de ruta pasando por tres ejes (o componentes) fundamentales del trabajo, el modelado del mundo, en este caso, un grafo bipartito que simula a la red completa, un motor de utilidades, en este caso, el GNN/MNL y experimentos para comprobar y poner a prueba estos modelos.

El modelo MNL resultó ser interpretable y preciso, con un 90% de precisión. El hallazgo principal en cuanto a las características fue la preferencia de los usuarios a viajes sin transbordo aunque eso signifique un tiempo de viaje mayor.

El modelo GNN logró una leve mejoría. Se comparan GNN con capas de decisión discreta activadas y desactivadas. Debido a la leve mejoría del GNN, se opta por hacer experimentos con el MNL gracias a su velocidad e interpretabilidad.

Finalmente, se realizan experimentos para poner a prueba el modelo del MNL, en este caso, se cambian frecuencias de servicios y se suspenden otros, analizando localmente la redistribución de la demanda. Para terminar, se agrega la Línea 7 del Metro de Santiago pronta a inaugurarse, para analizar los cambios de la demanda. Se observa un aumento de carga general en el metro, especialmente en la L1; un aumento de la cantidad de transbordos, un aumento en la demanda de servicios alimentadores.

Se concluye que los experimentos evidencian fortalezas de la solución, como su capacidad de generar datos sintéticos, pero también debilidades, como la inmutabilidad de la elección del paradero inicial y final.

Tabla de Contenido

1	Introducción	1
2	Revisión de la Literatura	3
2.1	Revisión de Literatura y Contexto	3
2.1.1	Sujeto de la predicción :	3
2.1.2	Tipo de datos:	4
2.1.3	Factores	5
2.1.4	Modo de transporte	5
2.1.5	Técnicas de preprocesado de datos	5
2.1.6	Actualmente en Chile.	6
2.2	Representación de los Datos y las Técnicas de Predicción	7
2.2.1	Grafo	7
2.2.2	Algoritmos de Ruteo	10
2.2.3	MNL	10
2.2.4	GNN	11
2.2.5	Red Metropolitana de Movilidad	12
2.2.6	ADATRAP	12
3	Metodología	15
3.1	Objetivos Específicos a cumplir	15
3.2	Solución Propuesta	15
3.3	Evaluación	16

3.4	Plataforma de desarrollo técnica	17
3.4.1	Exploración del repositorio	17
3.4.2	Instalación	17
3.4.3	Notebooks de exploración	17
3.5	Parte 1: Exploración de Datos	18
3.5.1	Descarga y Exploración de datos	18
3.6	Parte 2: Creación de los Grafos	18
3.6.1	Grafo Agrupado:	18
3.6.2	Grafo Bipartito	20
3.6.3	Frecuencias de los servicios	21
3.6.4	Velocidades promedio de los servicios.	21
3.6.5	Nodos	22
3.6.6	Aristas	23
3.6.7	Algoritmo para la creación del Grafo Bipartito	24
3.7	Parte 3: Modelo de Regresión Lineal Simple	24
3.7.1	Dataset de entrenamiento	25
3.7.2	Algoritmo de entrenamiento:	25
3.8	Parte 4: MNL con Destino	27
3.8.1	Métricas de Entrenamiento	27
3.8.2	Dataset	28
3.8.3	Entrenamiento	31
3.9	Parte 5: GNN	31
3.9.1	Arquitectura de la Solución.	31
3.9.2	Datos	31
3.9.3	Embeddings Iniciales	32
3.9.4	Bipartite GNN	32
3.9.5	Normalización y Contexto	32

3.9.6	Features y Concatenación	32
3.9.7	MLP (Perceptron Multi Capa) Scorer	32
3.10	Parte 6: Experimentos	35
3.10.1	Caso 1: Disminución de la Oferta de un Servicio	35
3.10.2	Caso 2: Suspensión de un Servicio.	35
3.10.3	Caso 3: Agregar Línea 7	35
4	Resultados y Discusiones	38
4.1	Parte 1: Exploración de los Datos	38
4.1.1	Subidas a un paradero durante el día.	38
4.1.2	Uso de un servicio.	40
4.1.3	Métricas de Demanda	42
4.1.4	Zona 777	43
4.2	Parte 2: Grafos de la Red	44
4.2.1	Grafo Agrupado	44
4.2.2	Grafo Bipartito	44
4.3	Parte 3: Modelo de Regresión Lineal Simple	47
4.3.1	Estructura del Dataset	47
4.3.2	Resultados del Entrenamiento	47
4.4	Parte 4: MNL con destino	49
4.4.1	Resultados y coeficientes.	49
4.4.2	Primeros Transbordos y la Penalización Inicial	50
4.4.3	Entrenamiento Diario.	51
4.4.4	Entrenamiento Semanal completo.	51
4.5	Parte 5: GNN	53
4.5.1	Resultados	53
4.6	Parte 6: Experimentos	57
4.6.1	Experimento 1: Disminución de oferta de un servicio.	57

4.6.2	Experimento 2: Suspensión de un servicio.	62
4.6.3	Experimento 3: Agregar Línea 7	64
4.7	Limitaciones del Modelo	67
5	Conclusión	69

Índice de Tablas

3.1	Descripción de las columnas del dataset de decisiones para el modelo MNL con destino.	30
3.2	Estaciones y coordenadas. El nombre de las estaciones no es el oficial. Están en orden partiendo desde Brasil hasta Estoril. Entre paréntesis aparecen las combinaciones. "esq" abrevia "esquina"	36
4.1	Coeficientes obtenidos del modelo MNL (todos nulos).	49
4.2	Métricas de entrenamiento y validación del modelo MNL.	49
4.3	Coeficientes del modelo MNL entrenado para distintos días de la semana. La última fila muestra el promedio de cada columna. Este entrenamiento fue por cada día.	51
4.4	Resumen de datos y particionado	52
4.5	Historial de optimización (iteraciones)	52
4.6	Coeficientes del modelo MNL	52
4.7	Métricas de la mejor época de la MNL con Destino Semanal	53
4.8	Resumen del modelo	53
4.9	Historial de entrenamiento por época	54
4.10	Resumen del modelo	55
4.11	Historial de entrenamiento por época	55
4.12	Mejor modelo y evaluación en test	56
4.13	Itinerarios de las alternativas desde T-11-64-P0-30 hasta T-20-177-P0-20	61
4.14	Uso de Servicios (Etapas Reales)	65
4.15	Uso de Servicios (Etapas Sintéticas)	65

4.16	Servicios Alimentadores Principales de L7 (Transbordos)	66
4.17	Comparación de Carga de Metro (Total Estaciones-Pasajero Recorridas)	66

Índice de Ilustraciones

2.1	Esquema resumen del Grafo Bipartito	9
3.1	Arquitectura de la GNN	34
3.2	Esquema direccional propuesto para la Línea 7	36
4.1	Subidas en el paradero PJ394	39
4.2	Subidas en Tobalaba L1 y L4	40
4.3	Uso del servicio 507 de vuelta (Desde Grecia a ENEA)	41
4.4	Uso de la Línea 1 durante el día	41
4.5	Zonas tarifarias 777 en Santiago	43
4.6	Mapa en Plotly con zoom a un barrio de Cerro Navia	44
4.7	Esquema resumen del grafo bipartito	45
4.8	Resumen del dataset para el modelo MNL básico	47
4.9	Histograma de Pesos	50
4.10	Costes para ir desde PJ394 a PA300. En azul se muestran los costes reales y en naranja los cambios de oferta	58
4.11	Distribución de probabilidades para alternativas de viaje antes y después del cambio de oferta	58
4.12	Costes para ir desde PJ394 a PA433	59
4.13	Distribución de probabilidades para alternativas de viaje antes y después del cambio de oferta (Experimento 2)	59
4.14	Distribución de probabilidades para alternativas de viaje con un aumento del 1500% en el tiempo de espera del servicio 507	60

4.15	Distribución de probabilidades para alternativas de viaje con suspensión de la Línea 1	62
4.16	Costes para ir de San Pablo a Baquedano con suspensión de la Línea 1	63
4.17	Costos del paradero E-20-53-PO-115 a Tobalaba con suspensión de la Línea 1 . . .	63
4.18	Distribución de probabilidades para alternativas con suspensión de la Línea 1 . . .	64

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.

La planificación o intervenciones a la red de transporte enfrenta el desafío de predecir efectos colaterales en la red existente.

Crear un modelo de decisión en base a alternativas (oferta) ofrecidas al usuario para llegar a un destino dado un origen y hora, para luego con ese mismo modelo, generar una demanda sintética de una red de transporte en base a cambios nuevos (una redistribución de la demanda) es el objetivo de esta memoria.

Históricamente se han usado soluciones discretas como las MNL (Multinomial Logit) y recientemente modelos como GNN (Redes Neuronales de Grafos) para la predicción de la demanda.

Por el lado de la MNL, esta solución se enfoca en una ingeniería de características la cual propone variables de interés en el proceso de decisión de un usuario, tales como el tiempo de espera, la velocidad o que tanto acerca al destino el servicio de transporte público.

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**, usar Redes Neuronales de Grafos es buena idea, 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 [9], 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.

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 proceso de decisión 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.

Como objetivos específicos, se propone modelar la red completa como un grafo, entrenar a modelos predictores, para terminar con experimentos para poner a prueba a estos modelos.

Es por ello, que el objetivo de esta memoria se reduce a:

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.

La estructura de la memoria consiste en el Capítulo 2, en el que se hace una revisión de la literatura, datos, modelado de la red y modelos de predicción. En el Capítulo 3 se presenta una metodología basada en exploración de los datos y su modelado, datos y entrenamiento del MNL/GNN y termina con los experimentos. El Capítulo 4 presenta los resultados y conclusiones de la metodología aplicada con su respectiva discusión. El capítulo 5 presenta la conclusión de la memoria.

Capítulo 2

Revisión de la Literatura

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.

La siguiente sección se dividirá en dos, la primera, una revisión de literatura para entender el contexto y el estado del arte en el ámbito de la predicción de uso de transporte público, y la segunda, un marco teórico que explora las técnicas de predicción y las herramientas a utilizar en la memoria.

2.1 Revisión de Literatura y Contexto

Siguiendo el trabajo de Torrepadula más a fondo [11] se abren muchas soluciones y consideraciones: La primera, el sujeto de la predicción.

2.1.1 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 [13] usan enfoques no lineales para estimar la demanda en algunas estaciones de metro.
2. Cantidad de personas en la ruta. El trabajo de Zhao [15] 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[5] con un Support Vector Machine más un filtro de Kalman.

4. Cantidad de personas en un área. El trabajo de Wang et al [12] 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[9]. 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.

2.1.2 Tipo de datos:

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 [11], 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 [14], Jian [3], Li [4] y Yang et.al utilizan datasets provenientes de tarjetas de validación con tecnología similar o idéntica a la de la tarjeta Bip!.

2.1.3 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.

2.1.4 Modo de transporte

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

2.1.5 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[6] 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 [9] pueden ser utilizadas para llenar esta matriz de origen-destino. Otros enfoques, como el de Massobrio[7] modelan una red con nodos que representan las paradas de las rutas.

2.1.6 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 [9] 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 (el cambio de demanda) han sido objeto de análisis en trabajos previos. Un ejemplo representativo es el de Ramírez [8], 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 preveer 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.

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 periodos 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 [10], 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.

2.2 Representación de los Datos y las Técnicas de Predicción

En este trabajo se explorarán dos formas de predecir el uso del transporte público, estas son el MNL y las GNN. Pero antes de seguir ahondando en ellas, es importante establecer bases teóricas sobre como se harán estas predicciones.

2.2.1 Grafo

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). Es directo observar que se puede modelar un sistema complejo de caminos usando grafos, si es que además se agregan pesos a las aristas.

Considerar una red de transporte de la siguiente manera:

- Cada paradero P es un nodo V de este grafo. Por ejemplo, Estación Central es un paradero o PA433 es un paradero (el paradero justo fuera de la Facultad de Ciencias Físicas y Matemáticas).
- Cada arista E es una conexión entre dos paraderos, dada por un servicio. Se define un servicio como un recorrido de la red. Por ejemplo, la Línea 1 (L1) o la 506 son servicios.

En este sentido, se definen dos tipos de grafos que se utilizarán en la memoria, uno para exploración visual, el Grafo Agrupado, y otro para el uso de algoritmos de *ruteo* y las técnicas de aprendizaje que se mencionarán en un futuro, el Grafo Bipartito.

Grafo Agrupado

El primer grafo, denominado grafo agrupado, es un grafo necesario para la visualización de la información. No es útil para entrenar modelos ni para ejecutar algoritmos de ruteo, pero permite un mayor entendimiento de la estructura de la red.

Consiste en un grafo el cual los nodos son los paraderos y las aristas son las conexiones consecutivas de un servicio al recorrer los paraderos. Un servicio S tiene un set de paraderos P_k con k el índice del paradero en el orden en que los recorre.

Por ejemplo, se tienen dos servicios, S_1 y S_2 , ambos tienen un set de paraderos distintos P_k y Q_i , con P denotando los paraderos de S_1 y Q los de S_2 . Notar que un paradero V en el grafo (un nodo V) puede ser el paradero P_3 (el tercer paradero para S_1) y el paradero Q_4 (el cuarto paradero del recorrido S_2). Entonces, si es que el primer servicio recorre en un subset de paraderos en el mismo orden que el segundo servicio, la arista que une ambos paraderos tendrá la información de

que es recorrida por ambos servicios, en una sola arista. Es directo ver que este grafo no conviene para ejecutar algoritmos de ruteo, pues se pierde la información del peso de la arista, ya que S_1 puede demorarse más en recorrer la arista que S_2 .

Este grafo si conviene para visualizar la red, pues agrupa todos los servicios que recorren los mismos paraderos en el mismo orden en una sola arista. Cada nodo entonces tendrá la información de los servicios que paran en él y las aristas los servicios que conectan los nodos.

Grafo Bipartito.

Un grafo Bipartito en cambio se puede entender como un grafo de estado que captura las transiciones por las que pasa un usuario al tomar un transporte. Se entienden 4 cambios de fases.

- CAMINAR: Se permite caminar entre paraderos para combinar o hacer transbordo si es necesario.
- SUBIR: Subir al servicio.
- BAJAR: Bajarse del servicio.
- VIAJAR: Viajar a bordo del servicio.

Para cada cambio de fase, se asocia un coste, estos son:

- COSTE DE CAMINAR: Calculado como la distancia entre dos paraderos caminando a una velocidad dada promedio.
- COSTE DE ESPERA (SUBIR) : Calculado como el tiempo de espera esperado para subir al servicio.
- COSTE DE BAJAR (BAJAR): Es cero, pues bajar del servicio es un acto puntual.
- COSTE DE VIAJAR (VIAJAR) : Es el tiempo que tarda en recorrer un servicio entre dos paraderos.

Como los cambios de fase con costes, es directo entender que los cambios de fases definen los tipos de aristas que habrán en el grafo.

Pero, cada cambio de fase o arista debe de mostrar la transición de un estado al otro. Se definen los siguientes dos estados (nodos):

- EN PARADERO: Cuando se espera al servicio.
- EN SERVICIO: Cuando se está en el servicio.

Entonces, entre dos estados EN PARADERO pueden haber aristas CAMINAR y aristas VIAJAR. En cambio, entre un PARADERO y un SERVICIO, pueden haber aristas SUBIR o BAJAR. La direccionalidad de las aristas es importante. Lógicamente la arista BAJAR irá desde el SERVICIO hasta el PARADERO. En cambio, la arista CAMINAR es bidireccional. Una arista VIAJAR es de un solo sentido en el caso de los buses, pero bidireccional en el METRO.

El diagrama de la figura 4.7 muestra un esquema resumen del grafo Bipartito.

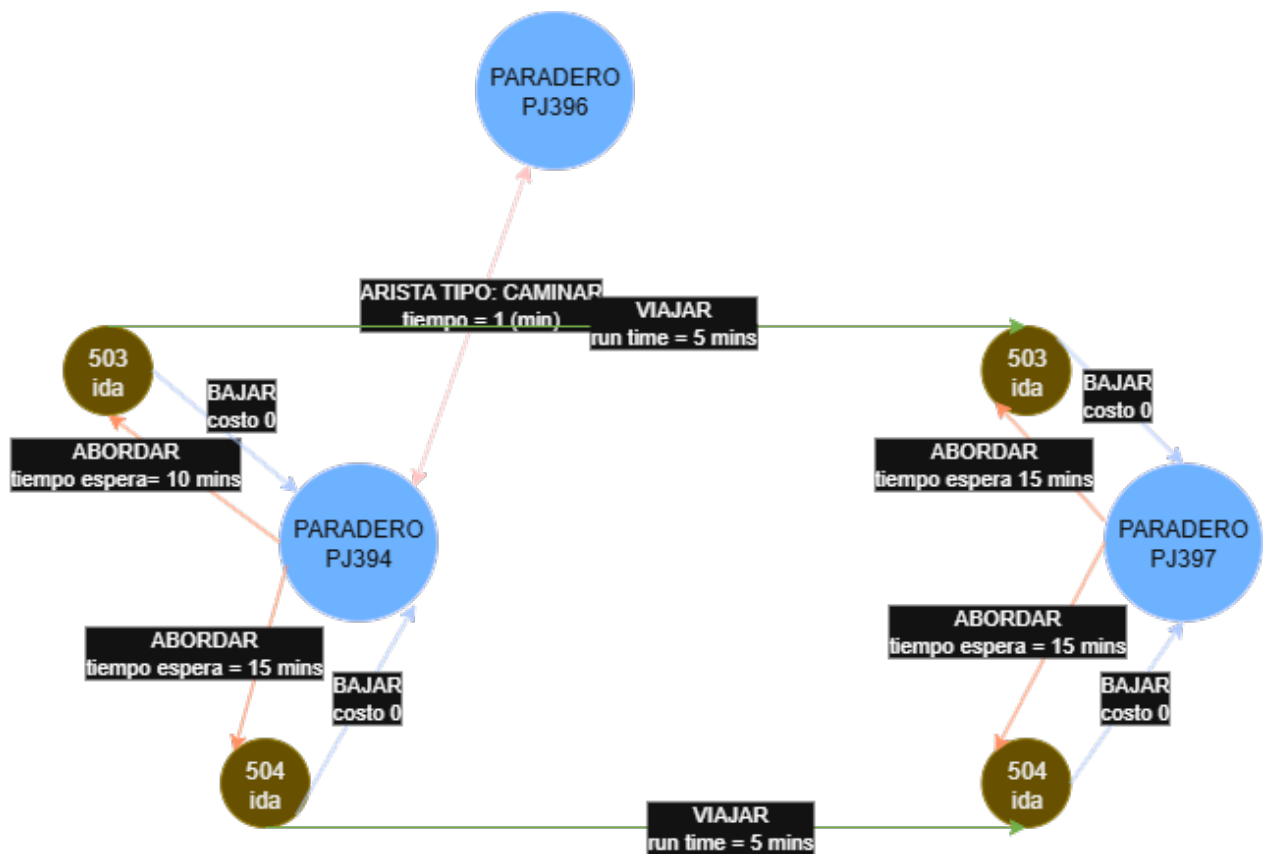


Figura 2.1: Esquema resumen del Grafo Bipartito

Notar como el grafo codifica cambios de estado en un viaje de un usuario. En este grafo, el usuario puede caminar, combinar y viajar.

Finalmente, un usuario si quiere ir de un paradero P al paradero Q, tendrá que:

- ESPERAR al servicio deseado (ABORDAR)
- VIAJAR hasta el destino o al paradero de bajada para el primer transbordo si lo amerita (N ARISTAS VIAJAR)
- BAJAR
- CAMINAR si es necesario o ESPERAR si es que el servicio deseado para en el mismo paradero de bajada
- Volver a VIAJAR ...
- BAJAR en el destino final.

Cada acción requiere pagar un coste.

2.2.2 Algoritmos de Ruteo

Para saber el camino óptimo de un usuario dado un paradero, se usará el Algoritmo de Dijkstra, debido a su simplicidad, y porque en muchas librerías ya está implementado y muy optimizado. Dado dos nodos, el algoritmo de Dijkstra buscará el camino con menos coste entre ambos nodos.

Gracias a que el grafo indirectamente penaliza los transbordos, el algoritmo de Dijkstra podrá encontrar caminos con sentido lógico, en vez de cambiar de recorrido en cada paradero, cosa que pasaría en el grafo agrupado, en el cual no se penaliza hacer transbordos.

2.2.3 MNL

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 N alternativas de servicio en un paradero, sean S_1, S_2, \dots, S_n en un paradero de origen P y un destino Q , el modelo MNL permite predecir la probabilidad de que el usuario elija cada una de las alternativas en base a variables propuestas como *determinantes* por el propio ingeniero. En este sentido, el ingeniero de software propone variables que él considera importantes para la toma de decisiones, pero no le da la importancia él mismo. El modelo será encargado de decir que variable es más importante que otra en el proceso de entrenamiento.

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 parte determinística (los atributos) y una parte aleatoria ϵ_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 β_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 β_1 es muy grande, el atributo x_1 es muy importante en la decisión del usuario. Si β_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 β_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.

Ventajas

- Fácil de interpretar, pues se obtienen valores para cada feature de los costes, permitiendo interpretar que considera más valioso el usuario en términos de utilidad.
- Más rápida de entrenar, pues requiere menos datos.
- Resistente a outliers. Al ser una función global la obtenida, servicios que tengan desvíos o datos corruptos no se verán afectados por esto.
- Fácil de crear escenarios ficticios de nuevos servicios pues no es necesario reentrenar.

Desventajas

- No captura correlación espacial .
- Depende fuertemente de las variables propuestas.

2.2.4 GNN

Una red neuronal de grafos (GNN) son redes neuronales especializadas para recibir como inputs grafos. A diferencia de redes neuronales como las LSTM o las convolucionales, las cuales reciben datos con una estructura más rígida (una secuencia o una grilla), las GNN reciben datos en grafos abstractos. Entonces, se puede pensar a las GNN como una abstracción general de un set de datos relacionados entre sí.

Una GNN aplicada al transporte público es una red neuronal capaz de aprender características espaciales. La idea es que la GNN prediga la primera elección de un viaje en el transporte público, dado un paradero inicial, un destino paradero y un bin/día. Para ello, se explora la solución de

una GNN Heterogénea que aproveche la riqueza de los tipos del grafo bipartito entre los nodos y las aristas.

Una GNN tiene embeddings o representaciones vectoriales, tanto en los nodos paradero como en los nodos servicio, que permite añadir riqueza y similitud entre paraderos.

Ventajas

- Captura correlación espacial
- más riqueza local gracias a los embeddings

Desventajas

- Más tiempo de entrenamiento y más datos necesarios.
- Menos resistente a outliers (embeddings de recorridos corruptos o desviados afectan localmente)
- Menos interpretabilidad debido a lo abstracto que son los vectores de embeddings.
- Requiere reentrenar para escenarios ficticios de servicios que no existen (ya que hay que obtener los embeddings)

2.2.5 Red Metropolitana de Movilidad

Red Metropolitana de Movilidad es el sistema de transporte público de Santiago, el cual opera mediante empresas como Metro S.A, el tren urbano Estación Central-Nos y el sistema de Buses anteriormente llamado Transantiago .

Su operación se basa en distintas unidades de negocio (UN) que trabajan en conjunto con un set de servicios acordados mediante licitaciones.

Los datos necesarios para crear el grafo y enriquecerlo con costes para entrenar están disponibles de manera pública en los planes de operaciones y en las matrices de viaje.

2.2.6 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 término 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, entonces se concluye que en la mañana el usuario se bajó cerca del paradero P' usando el servicio X, y que en la tarde el usuario se bajó cerca del paradero P en el servicio Y.

Tabla de viajes y etapas

Las tablas de viajes y etapas serán las 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 línea. 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). Estos se pueden separar en bins de 30 minutos cada uno. En resumen, 48 bins de tiempo. Se denominan en el lenguaje de ADATRAP como mediahora.

Servicios de las 4 etapas

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

Hay más 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.

Consolidado de recorridos

Para crear el grafo, lógicamente es necesario el trazado de todos los recorridos de RED. Para ello, se RED tiene en 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, se podría hacer lo siguiente:

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)

Zonas 777

Red delimita Santiago en zonas, llamadas Zonas777. Estas están accesibles tanto desde la tabla de etapas (es decir, la subida y bajada denota en que zona ocurrieron), como también en el consolidado en cada paradero. Además, RED entrega archivos SHAPE que pueden analizarse con GEOPANDAS para visualizar las zonas 777.

Con todo esto presente, se hacen las siguientes afirmaciones:

- La demanda de origen y destino es inamovible. Esto debido a que no se sabe donde vive exactamente cada usuario, ni donde trabaja o estudia.
- 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.

Capítulo 3

Metodología

En la presente sección, primero se darán a conocer los objetivos específicos para completar el objetivo general ya mencionado. Luego, se enunciará la solución escogida y finalmente la metodología para cumplir cada paso de la solución.

3.1 Objetivos Específicos a cumplir

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, 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. 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.

3.2 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 a la “Pila Tecnológica”, se usará Python como lenguaje de programación, bibliotecas como Tensorflow o Pytorch y sus derivados para crear redes. 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: Se creará un grafo agrupado y un grafo bipartito para la visualización y entrenamiento respectivamente.
2. MNL: Se implementará un modelo logit multinomial que aprenda a captar factores determinantes ajustando una función de probabilidad. Estos factores determinantes son factores cuantitativos tales como el tiempo de viaje, el numero de transbordos, el tipo de transporte usado y el tiempo de espera.
3. GNN + MNL 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! y patrones de movilidad. Para ello se utilizará un modelo con GNN + MNL. Una GNN procesará la estructura espacial del grafo y los datos de costes serán procesados por un modelo de decisión discreto.
4. Entrenamiento y ajuste del modelo: Utilizando datos históricos (validaciones Bip!, patrones de movilidad, datos censales), se ajustarán los parámetros del modelo de MNL y GNN para que el comportamiento simulado refleje lo más fielmente posible la realidad.
5. Ajustes a la oferta: Con el modelo calibrado, se introducirán 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.

3.3 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 MNL/GNN 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 MNL/GNN 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.
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.

Antes de seguir, es importante dejar en claro la plataforma técnica del proyecto.

3.4 Plataforma de desarrollo técnica

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, más adelante, se darán a conocer los paquetes y herramientas utilizadas.

3.4.1 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.

3.4.2 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. Más información sobre la instalación se encuentra en el README.md del repositorio mismo el cual está alojado en GitHub.

3.4.3 Notebooks de exploración

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

3.5 Parte 1: Exploración de Datos

3.5.1 Descarga y Exploración de datos

Los materiales del proyecto lógicamente serán los datos. Estos serán descargados desde la página web de RED.

Una exploración de estos datos es necesaria. Para ello, a modo de experimentación se obtendrán los siguientes gráficos e histogramas, con ayuda de Polars para manejar los datos de red en formato csv.

- Subidas a un paradero durante el día. Esto se obtendrá filtrando todos los viajes de la tabla de etapas que tengan paradero inicial en cualquier etapa el paradero requerido. Esto para todas las horas.
- Ocupación de un servicio durante el día. Esto se obtendrá sumando y restando las subidas y bajadas de los usuarios en cada paradero. Por ejemplo, si una entrada de la tabla de etapas es una subida y bajada de la 507 en un paradero dado, la hora de la bajada denota una resta de la cantidad de usuarios que están usando el servicio en cualquier vehículo en circulación.
- Zonas 777: Se crearán visualizaciones para ilustrar los límites de las zonas777 en Santiago.
- Métricas de Demanda: Se entregarán todas las subidas de cualquier etapa en cualquier paradero en cualquier bin. Esto se hará usando Polars filtrando adecuadamente el dataset.

3.6 Parte 2: Creación de los Grafos

Para crear los grafos Agrupados y Bipartito, se realizará lo siguiente:

3.6.1 Grafo Agrupado:

Aristas

En este 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 se tienen 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.

Si varios servicios paran en las mismas paradas consecutivas, podemos unir todos los recorridos en la misma arista. Es más 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.

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.

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. Cada servicio tiene una secuencia de nodos que visita en orden. Por ejemplo, la secuencia de paraderos que visita un recorrido X es $P[]$. Si el set de nodos es $V[]$, se puede 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 que ser igual a k .
4. 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 se puede 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.
5. 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. más 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 todos 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, se revisa el parámetro “siguiente_parada” que contiene la siguiente parada desde la que se está revisando (un puntero básicamente). Se crea 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. Se crea un nodo por cada paradero.
11. Se unen los nodos con las aristas.

Con ello, se puede crear un grafo interactivo con Gephi (software open source) que permite visualizar el grafo. Se utiliza el par lat, lon para generar un grafo configurado de manera visual con GeoLayout.

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

3.6.2 Grafo Bipartito

Un grafo más sofisticado es necesario para capturar la información de la demanda. Para ello, se creará un Grafo Bipartito. El grafo tendrá distintos tipos de aristas y nodos. Pero antes, es necesario establecer algunas fuentes de datos extra.

Notación y datos base

De ahora en adelante, se usará la siguiente notación para definir los elementos del grafo bipartito:

- **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:**

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

Esto nos permite definir lo siguiente:

3.6.3 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.

3.6.4 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>

3.6.5 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.

3.6.6 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.

3.6.7 Algoritmo para la creación del Grafo Bipartito

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 más cercanos. Se definió 10 paraderos a menos de 200 metros. Si un paradero no tiene ningún 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 todos los nodos de tipo servicio con aristas VIAJAR según el recorrido .

3.7 Parte 3: Modelo de Regresión Lineal Simple

Para comenzar, es interesante un modelo simple. Este no tendrá en cuenta hacia donde querrá ir el usuario. Simplemente tomará en cuenta cuánto tarda el bus.

3.7.1 Dataset de entrenamiento

La receta para construir el dataset de entrenamiento es el más 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 *is_chosen*, que es 1 si el servicio es el que tomó el usuario y 0 en caso contrario.
4. Entrenar el modelo.
5. Obtener Métricas

3.7.2 Algoritmo de entrenamiento:

- 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 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 :**

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

3.8 Parte 4: MNL con Destino

Un MNL con destino se refiere a incluir en los parámetros un coste llamado *coste restante* y *costo de viaje* dependiendo del destino final del usuario. Un ejemplo ilustrativo viene a continuación.

Suponer que para ir a un destino D desde un origen O tiene dos opciones. Un servicio S_1 que le deja directamente en el destino, con un coste de viaje asociado Cv_1 y un servicio S_2 que tiene un coste de viaje Cv_2 hasta el primer transbordo, para luego tener un costo de viaje de ese servicio de transbordo Cr_2 .

Si es que el tiempo de viaje de S_1 es menor y además deja directamente en su destino, es lógico que tomar este servicio es la decisión idónea u óptima. Ahora, si el costo de viaje de S_1 es mucho más alto, quizás convenga tomar un transbordo. Un ejemplo clásico de esto sería hacer transbordo al metro usando un bus alimentador para llegar al sistema subterráneo. A priori, dependiendo de la urgencia del usuario, deberá de elegir una de las dos alternativas. No todos los usuarios piensan igual. Algunos prefieren comodidad y no hacer transbordos, sobre todo si están con algo de tiempo de sobra. Otras personas confían más en servicios más rápidos que les obligan a hacer transbordo. Como no todo el mundo piensa igual, el MNL es muy útil para estos casos, ya que entrega una distribución de probabilidad sobre que servicio se va a tomar, sobre todo cuando las utilidades de ambos son parecidas. El objetivo de este modelo es descubrir que prefieren los usuarios, si viajes más directos con menos transbordos -pero más largos-, o viajes más rápidos pero con transbordos. Notar que los transbordos tienen tiempos de viajes más variables. Poca confianza en los headways de los buses de transbordo pueden inflar el tiempo de viaje real, ya que la variable de tiempo de espera suele tener más varianza que el tiempo de viaje. más transbordos implican más varianza en el tiempo de viaje total y por lo tanto menos confianza en el trayecto, o sea, menos comodidad.

Con esta reflexión, es directo darse cuenta que lo que se busca con este modelo es descubrir como se comparan el tiempo de viaje total v/s que tanto acerca el servicio inicial al destino.

3.8.1 Métricas de Entrenamiento

Esta sección aplicará tanto para la MNL como la próxima GNN. Las métricas de entrenamiento serán:

- NLL (Loss): Indica que tan bien calibradas las probabilidades. Penaliza fuertemente la sobreconfianza cuando se falla. *más bajo es mejor*.
- NLL Normalizado: Normaliza NLL dependiendo del tamaño del set de alternativas. *0 es perfecto, 1.0 es uniforme*.
- Acc (Accuracy TOP1): Indica la proporción de predicciones que acertaron en el primer rank de las probabilidades, es decir, si la elegida realmente fue la más alta probabilidad. *más alto es mejor*.
- AccNT (Accuracy Non Trivial): Precisión de decisiones no triviales (es decir, con set de alternativas mayor a 1). *más alto es mejor*. Esta métrica es más importante que Acc, debido a la gran proporción de decisiones con solo una posibilidad.
- MRR (Mean Reciprocal Rank): Valora que la elegida esté alta en el ranking a pesar de que no esté top 1. *más alto es mejor*.

- LL (Log Likelihood): Se usa solo en el MNL. Suma de $\log(p|elegida)$. *Cuanto más negativo y cercano a cero mejor.*
- LL_null: LL del modelo uniforme. Para medir ganancia sobre el azar.
- Pseudo- R^2 de McFadden: $1 - LL_model/LL_null$. Análogo a R^2 ; 0 a <1 (mayor es mejor). En elección discreta, ~0.2–0.4 suele considerarse muy bueno.

3.8.2 Dataset

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

Los pasos para generar el dataset fueron los siguientes:

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

En este punto, se tiene 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. Tomar el siguiente ejemplo. Una persona que quiere ir desde PJ394 a PA433 (Beauchef) a las 10 de la mañana un día laboral.

- En el paradero PJ394 se tienen las siguientes alternativas a las diez de la mañana un día LABORAL: 503, 504, 507, 517, 518 , B38.
- Convenientemente también para el 507, así que el costo restante es cero, pues después de bajarse en la parada óptima, ya se llegó al destino.
- Para los otros servicios, el costo restante es mayor que cero, ya que ninguno deja directamente en PA433. Entonces, se debe 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 modo general, 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 este caso, el AD se corre en sentido inverso, es decir, partimos del nodo destino y vamos hacia atrás. De esta forma, se obtiene el costo mínimo para llegar al destino desde cualquier otro nodo.

Entonces, para un paradero de destino, un bin y un día se obtiene 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 sobre coste.

Por ejemplo, se puede ejecutar el AD para PA433 y el costo restante para ir desde cada paradero hasta PA433 es el costo de viajar. Como se separaron 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, se obtiene 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 se ejecuta el AD para el origen y destino, se obtendrán 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 siguen al paradero actual .
- Por cada alternativa, se calcula el costo de esperar el servicio más 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 en un espacio de tiempo acotado fue algo a tener en cuenta. Al comienzo se obtenían tiempos de cómputo de días o semanas. 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 Dijkstra para muchas decisiones a la vez, y estos resultados se cacheaban.
- Se *cacheó* el perfil de cada servicio.

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

La tabla de decisiones resultante tiene las siguientes columnas:

Columna	Descripción
decision_id	ID de la decisión. Común entre alternativas de la misma persona
person_id	ID de la tarjeta
trip_id_real	ID de viaje real: [person_id]_[número_de_viaje]_[número_de_etapa]
servicio_usuario_alternativa	Servicio alternativa en código formato usuario
sentido	Sentido del servicio
wait_time	Tiempo de espera
optimal_alight_stop	Parada óptima de bajada del servicio alternativa. Es la bajada real observada cuando el servicio fue tomado.
cost_to_go	Coste restante entre la parada de bajada y el destino final
viajar_cost	Coste entre el nodo servicio inicial y final del servicio inicial tomado
total_cost	Coste total sumando todos los costes (wait, cost_to_go, viajar_cost)
chosen	Si la alternativa fue elegida o no (solo una fila con 1 por decision_id)
choice_set_size	Tamaño del set de alternativas
day_type	Día (LAB, SAB, DOM)
bin30_k	Bin de tiempo en el que se llegó al paradero
origin_p_k	Paradero de origen en código TS
dest_p_final_obs	Paradero de destino final real en código TS
srv_obs_ts	Servicio real tomado en código TS
srv_obs_usuario	Servicio real tomado en código formato usuario
is_corrupt	Si la fila está corrompida
is_initial_transfer	1 si Dijkstra decide caminar a otro paradero para tomar un servicio adecuado, 0 en caso contrario

Tabla 3.1: Descripción de las columnas del dataset de decisiones para el modelo MNL con destino.

Nota sobre el coste restante

El coste restante tiene un problema. Asume que después de la primera decisión las personas son deterministas y no eligen con distribución de probabilidad. Esto es intencional. Modelar todo el trayecto como una concatenación de decisiones probabilísticas complica el modelo, cosa que para el desarrollo de la memoria puede ser perjudicial, teniendo en cuenta que se desarrollará el GNN después. Una pequeña intuición que no se desarrollará en este trabajo indica que posiblemente el costo restante sería una distribución o variable aleatoria más que un valor fijo escalar. Esto tiene más sentido real. Una persona sabe que hacer transbordo aumenta su varianza en su tiempo de viaje debido a que debe de esperar otro servicio, que induce una incerteza temporal. Aunque en el coste restante está incluido el tiempo de espera, realmente el tiempo de espera *esperado* debería de ser un tiempo que tenga en cuenta todos los tiempos de espera del paradero que le puedan servir al usuario. Lo mismo con los tiempos de viaje del servicio que pueda tomar el usuario. Es inmediato notar como se complica el problema, pues ahora cada decisión subsecuente tiene una distribución.

3.8.3 Entrenamiento

Algoritmo de Entrenamiento

Para el entrenamiento, se considera lo siguiente:

- Variables base: cost_to_go, wait_time, viajar_cost.
 - Variables derivadas : is_initial_transfer (binaria) y first_walk_time= penalty (5 mins) si is_initial_transfer es True.
 - zero_onboard = 1 si viajar_cost es 0.
 - ASC_METRO si usa el metro.
 - intercepto. Constante de probabilidad.
1. Se dividió un 20% del dataset para pruebas (test).
 2. El modelo tiene la utilidad ya mencionada anteriormente $u = X\beta$. La probabilidad por alternativa es un softmax estable por decisión.
 3. Tiene una regularización L2 sobre β .
 4. Optimización. Minimiza NLL con L-BFGS-B.
 5. Gradiente Analítico.
 6. 300 iteraciones máximas por época. tol = 1e-7 y l2_reg= 1e-3.
 7. Métricas las ya anteriormente mencionadas.

3.9 Parte 5: GNN

3.9.1 Arquitectura de la Solución.

Para ello, se implementa una GNN con la siguiente arquitectura mostrada en la figura 3.1.

3.9.2 Datos

Para los datos, se usará el grafo Bipartito ya mencionado anteriormente. Además, se utilizará la misma tabla de decisiones para entrenar al MNL para reutilizar datos. Se añadió la variante a la tabla de decisiones infiriéndola desde el bin, día y paradero en que tomó el servicio el usuario. Esto para homogeneizar los datos con respecto al grafo bipartito.

Las aristas SUBIR, VIAJAR, BAJAR Y CAMINAR tienen tensores relacionados con los costes de transicionar de estado en el grafo bipartito.

- SUBIR: “wait_48x3” un tensor [servicio, tipo dia, bin30].
- VIAJAR: “run_48x3” un tensor [arista, day_type, bin30].
- BAJAR: “cost” un tensor de ceros (no penalizar bajar del servicio).
- CAMINAR: “run_scalar” un tensor de rango 0 (un escalar) que denota el tiempo de caminata calculando distancia euclidiana dividido por la velocidad.

3.9.3 Embeddings Iniciales

Los embeddings son atributos vectoriales aprendibles por la red locales a cada nodo. Hay embeddings para :

- Paraderos
- Servicios
- Destinos

Una representación vectorial de un nodo es útil pues permite reducir la dimensionalidad, establecer similitudes y aprender localmente características de los nodos. Notar que la MNL no tiene esta característica espacial. Esto hace a la GNN más completa. Se podrían hacer embeddings para las aristas, pero esto hace al modelo menos interpretable.

3.9.4 Bipartite GNN

Una GNN Bipartito tiene 4 capas de GraphConv (en un inicio se usó SageConv, pero SageConv daba resultados poco convincentes debido a que diluía los pesos de las aristas). GraphConv aplica una convolución que permite que cada nodo agregue información de sus vecinos, el *message passing*.

Se agrega un parámetro τ_e que es aprendible por relación. Esto para todas las aristas. Este τ_e modula la intensidad del paso de mensajes, lo que implica aristas más importantes que otras (en otras palabras, unos costes pueden ponderar más, básicamente lo que hace la MNL)

3.9.5 Normalización y Contexto

Se aplica una Normalización L2 para que algunos embeddings dominen por magnitud, mejore la convergencia, facilita las comparaciones entre embeddings y previene el overfitting.

3.9.6 Features y Concatenación

Las features de Dijkstra son añadidas opcionalmente como una capa extra. Estos features se concatenan con todas las características ya aprendidas mediante los embeddings. El vector final entonces tendrá dimensión 64, 64, 64 + DIMENSIONES DE Dijkstra (FEATURES).

3.9.7 MLP (Perceptron Multi Capa) Scorer

Toma el vector concatenado y retorna una probabilidad para cada alternativa del modelo, implementando la lógica de decisión del modelo, muy parecido al MNL. El scorer es dinámico, por lo que respeta las dimensiones del vector concatenado. El MLP tiene las siguientes fases:

- Capa de entrada (nn.Sequential)
- Scores de Utilidad: Retorna los scores de cada servicio.
- Masking de alternativas : Tamaño variable demanda padding.
- Softmax: Transforma Scores a probabilidades de la misma manera que el MNL.
- CrossEntropyLoss: Penaliza predicciones incorrectas.

Las métricas de evaluación serán las mismas que las del MNL para poder compararlos efectivamente.

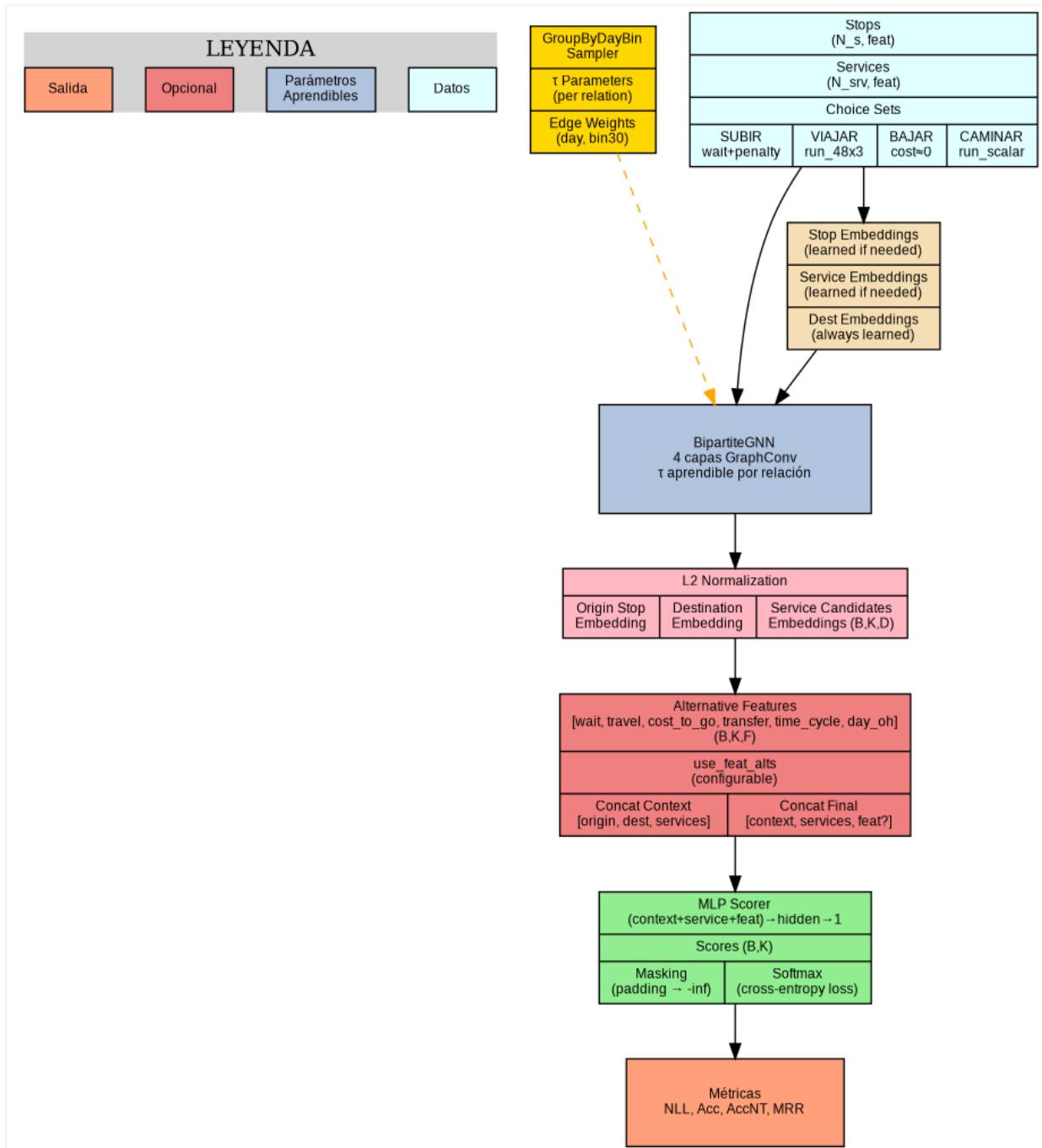


Figura 3.1: Arquitectura de la GNN

3.10 Parte 6: Experimentos

Para poner a prueba el modelo, un ejercicio interesante será exponerlo a cambios en la oferta .

Para ello, se ponen dos casos locales y uno global.

3.10.1 Caso 1: Disminución de la Oferta de un Servicio

Se disminuirá la oferta de un servicio modificando los tiempos de espera en el grafo. Con ello, ocurrirá una redistribución de la probabilidad en las alternativas que será analizada comparando las probabilidades *baseline* (basales) y las *contrafactuales* (el nuevo grafo). En específico, se ilustrarán dos casos, uno en el que dos servicios compiten (tienen probabilidades comparables) y uno de ellos verá su tiempo de espera modificado, mientras que el otro caso se enfrenta un servicio dominante (probabilidad muy alta), versus el resto, analizando la redistribución de la demanda en los transbordos. Esto se hará de manera local en un paradero y no en todo el grafo.

3.10.2 Caso 2: Suspensión de un Servicio.

Se suspenderá la L1, colocando un indicador booleano en sus aristas para que Dijkstra no permita subir al servicio, y se ejecutará el modelo de predicción . Se realizará la misma comparación mencionada en el caso anterior, y como esta redistribución sobrecargará otros servicios aledaños. Específicamente, en Providencia. Esto se hará de manera local, es decir, no se ejecutará el algoritmo de predicción sobre todo el día, ya que tardará demasiado.

3.10.3 Caso 3: Agregar Línea 7

Se obtendrá un trazado de la nueva Línea 7 pronta a construir. Los datos de esta nueva línea 7 siguen en la tabla 3.2. El trazado se obtuvo desde Metro y las coordenadas se aproximaron viendo google maps.

algoritmo:

- Eliminar las etapas intermedias, es decir, solo quedan las intenciones de viaje (paradero inicial, final, bin30, día)
- Por cada intención de viaje, ejecutar el predictor, y tomar el camino con mayor probabilidad de ser elegido .
- Expandir el camino en etapas.

Con esta tabla de demandas sintética, se comparan con la tabla de etapas original prestando atención específicamente a:

- Servicios alimentadores a la L7.
- Servicios que pierden demanda
- Servicios que ganan demanda.
- Cantidad de etapas promedio obtenida.

Capítulo 4

Resultados y Discusiones

En la siguiente sección se mostrarán resultados e inmediatamente una discusión de ellos. Esta parte está estructurada de la misma manera que la metodología para relacionarlas directamente de manera más sencilla.

4.1 Parte 1: Exploración de los Datos

En la siguiente sección se mostrarán los resultados de una exploración previa de los datos, fase importante para entender su estructura y trabajar de manera adecuada con ellos.

4.1.1 Subidas a un paradero durante el día.

Las subidas a un paradero durante el día se denotan simplemente contando las subidas en un paradero con código paradero igual al deseado en la tabla de viajes.

La figura 4.1 muestra la cantidad de subidas en el paradero PJ394 (Presidente José Joaquín Pérez esq. Las Lomas). Todos los servicios que pasan por este paradero van hacia el centro de Santiago o hacia el lado oriente de la ciudad. Los servicios que se detienen aquí son el 503,504,507,517,518 Y B38.

Se puede observar un pico a las siete de la mañana y una bajada consistente hasta la noche, sin experimentar otro máximo.

Notar además como no se registran subidas entre las 23:00 y las 4:30. Una explicación probable es porque esta zona de Cerro Navia no es segura. Es interesante ver la posible incidencia de la seguridad o comodidad en el proceso de elección de un servicio o una parada.

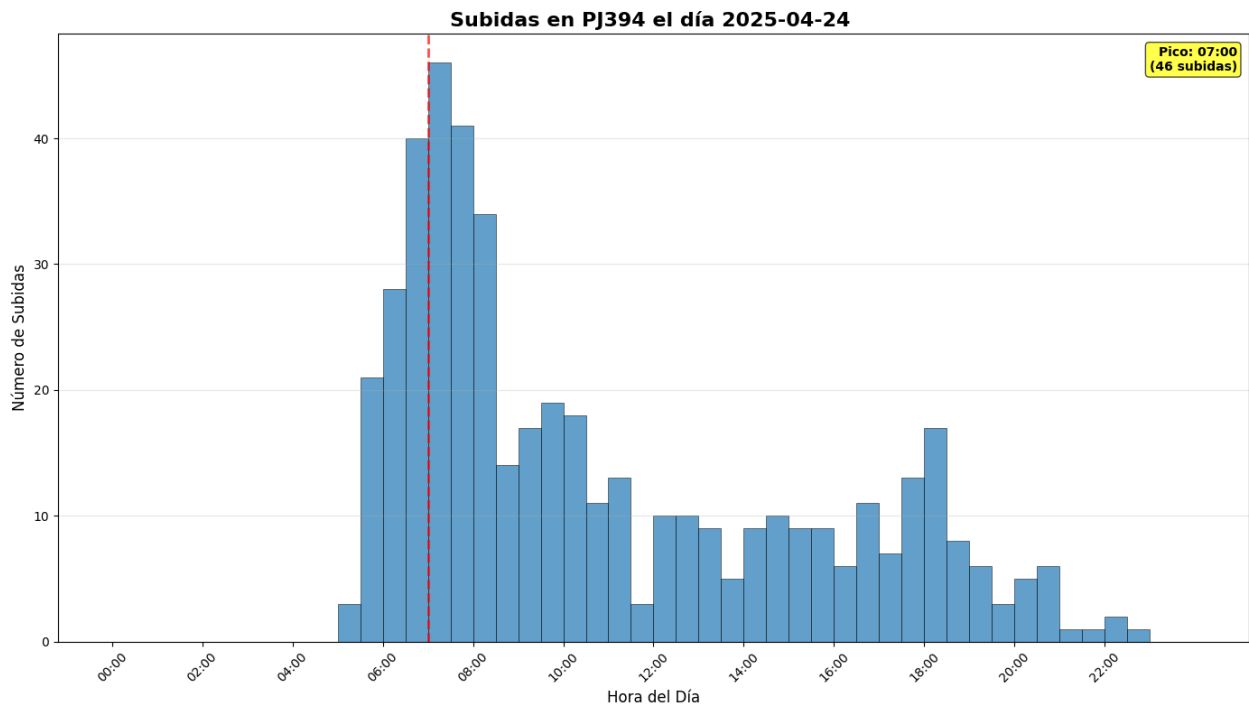


Figura 4.1: Subidas en el paradero PJ394

El mismo análisis se puede hacer para estaciones de Metro, en este caso, la figura 4.2 muestra las subidas en Tobalaba, tanto de Línea 1 y Línea 4. Notar que al no saber a priori cual de las dos líneas toma el usuario, ya que el torniquete es global para la estación, la única forma de saber en que línea efectivamente uso es infiriéndola sobre la estación de bajada, lo cual ya es una predicción hecha por ADATRAP.

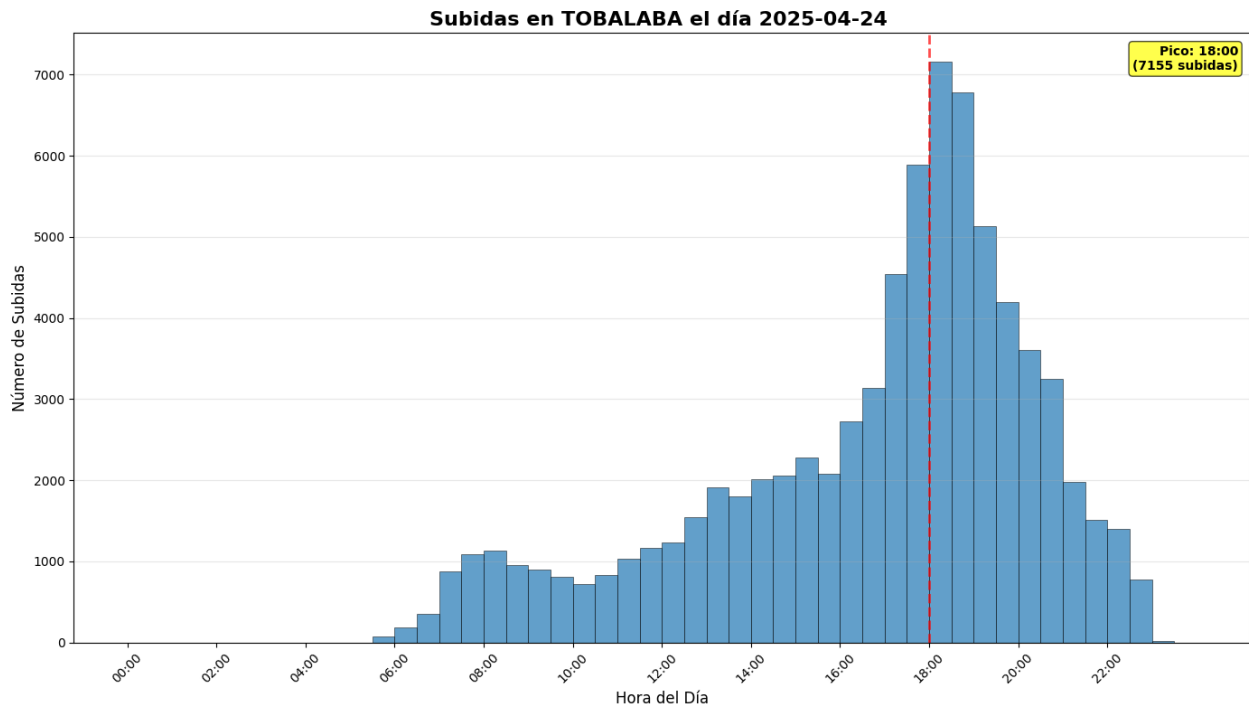


Figura 4.2: Subidas en Tobalaba L1 y L4

Darse cuenta claramente de la distribución de la hora peak en el Metro Tobalaba a las 18:00 horas. Esta estación es caracterizada por una alta afluencia de usuarios en hora peak. Solo con este dato, se puede extrapolar cuanto tiempo tardaría un usuario en tomar el metro al llegar a la estación, si es que quiere ir a Tobalaba L4. Esto se puede hacer viendo la frecuencia y la capacidad máxima de los trenes. Esta información es clave para planear una estación con más espacio y con mejores andenes.

4.1.2 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 se quita 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.

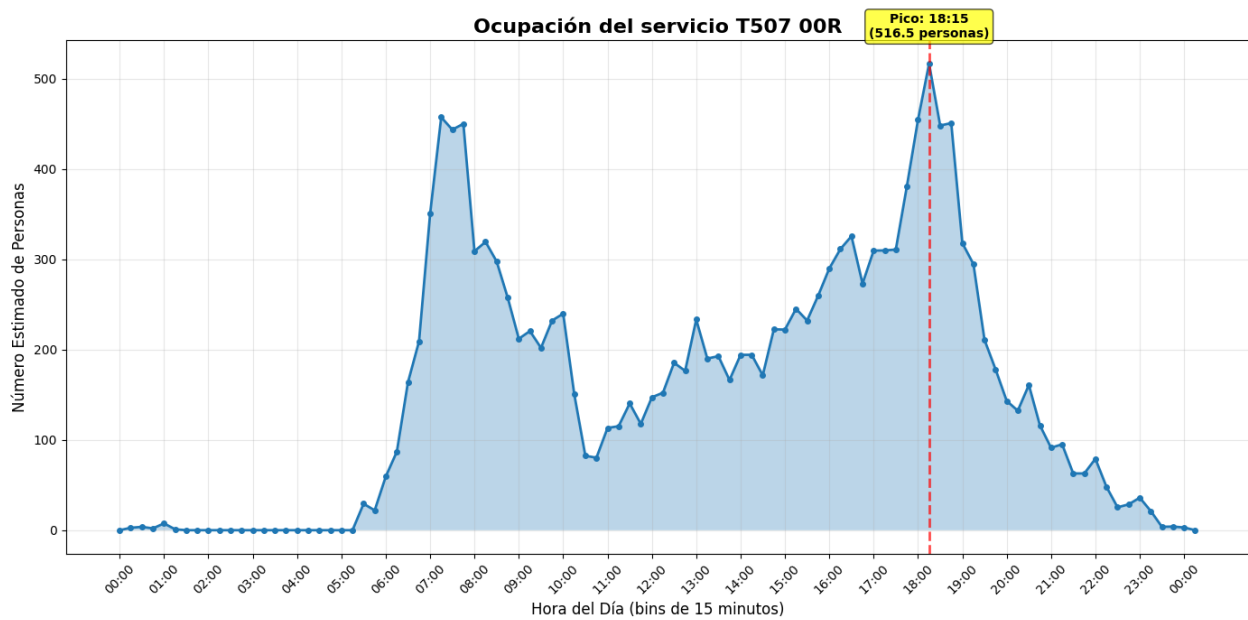


Figura 4.3: Uso del servicio 507 de vuelta (Desde Grecia a ENEA)

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.

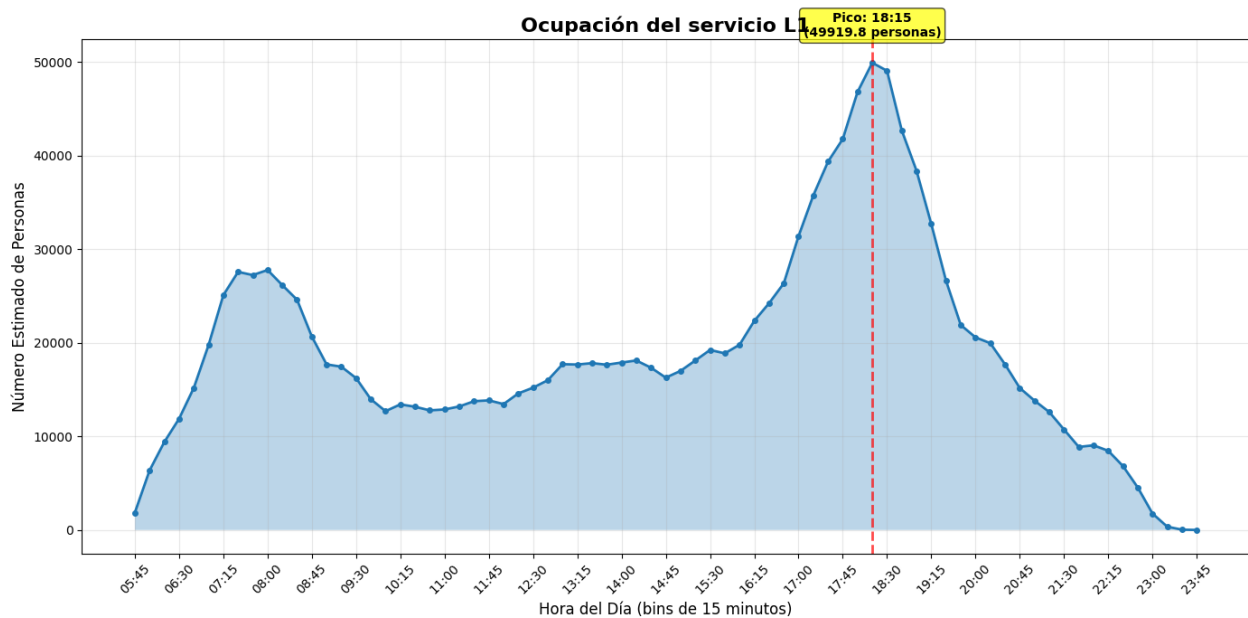


Figura 4.4: Uso de la Línea 1 durante el día

La figura 4.4 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 más 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 memoria, pero queda como propuesto.

4.1.3 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, se puede obtener la demanda de todas las tuplas P,S,T . La idea es escoger una ventana de tiempo Δt y establecer una distribución acumulada que determine la demanda entre ambos tiempos. 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 4.1: Salida del Programa

```
El paradero PJ394 en formato TS es: T-11-64-PO-30
Buscando demanda en T-11-64-PO-30 para T507 00I entre 08:00:00 y 10:00:00 ...
Procesando etapa 1 ...
Demanda en T-11-64-PO-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-PO-30 para T507 00I: 20
```

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

Listing 4.2: 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 4.3: 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
```

Las sospechas sobre como se guarda el servicio en estaciones de metro fueron ciertas. Al marcar la Bip! en Tobalaba, se marca automaticamente como L4 , nunca como L1.

4.1.4 Zona 777

La ciudad de Santiago esta dividida en zonas tarifarias. Las zonas 777 son el nombre que poseen. más abajo la figura 4.5 muestra las zonas 777 de Santiago.

Mapa Interactivo de Zonas 777

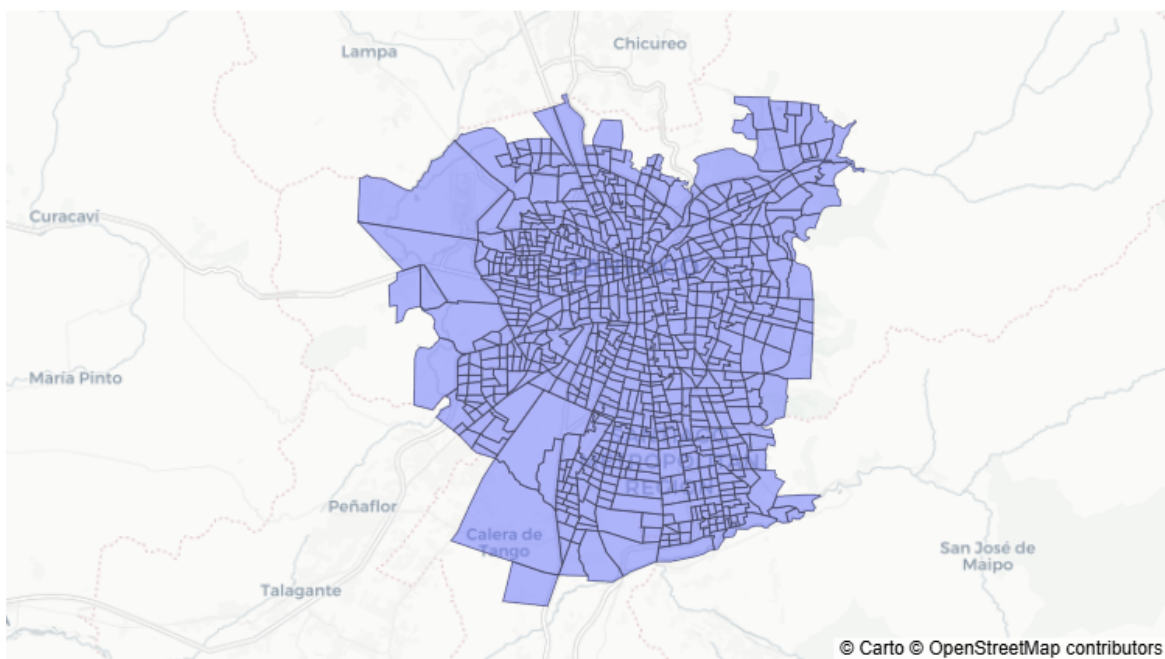


Figura 4.5: Zonas tarifarias 777 en Santiago

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.

4.2 Parte 2: Grafos de la Red

4.2.1 Grafo Agrupado

La creación del grafo agrupado dió como resultados un grafo con las siguientes características:

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

El mapa está ubicado en /notebooks/mapa_con_zonas.html. Si se desea observar el grafo con Gephi, es necesario descargar el software, instalarlo, cargar el grafo (ubicado en `data/graphs/grafos.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.

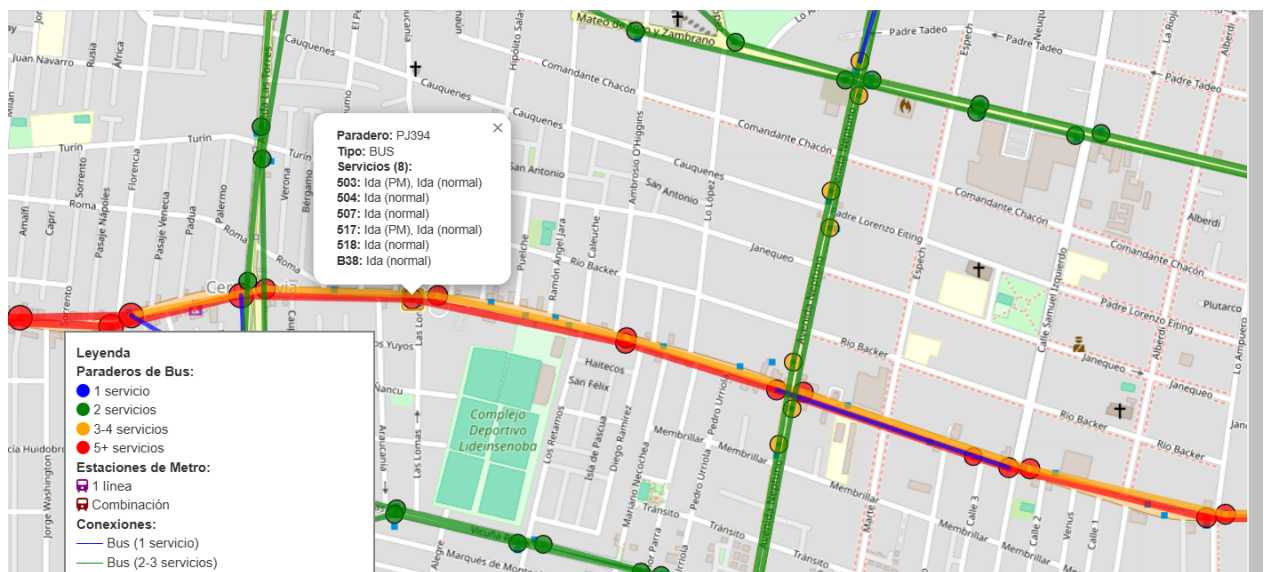


Figura 4.6: Mapa en Plotly con zoom a un barrio de Cerro Navia

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

4.2.2 Grafo Bipartito

A modo de recordatorio, se mostrará un esquema del grafo bipartito.

En la figura 4.7 se muestra un ejemplo del grafo de estado en la misma zona que se mostró en la figura 4.6.

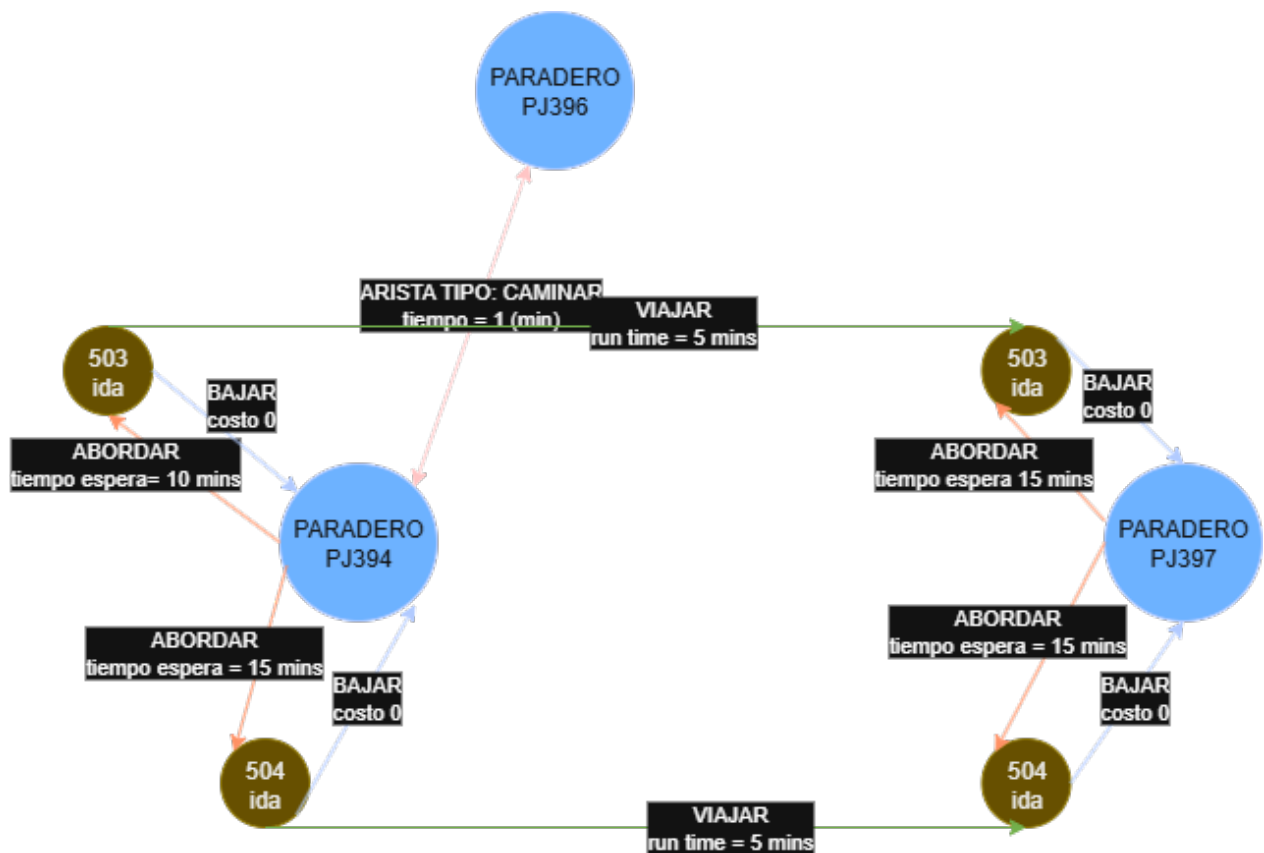


Figura 4.7: Esquema resumen del grafo bipartito

Si se audita el grafo para sanear errores, se obtiene lo siguiente:

=== Estado del grafo bipartito ===

Nodos totales : 60679

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

Aristas totales : 217305

Aristas por tipo :

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

Se obtiene un grafo muy útil. Por ejemplo, ya con este grafo con pesos se puede correr un algoritmo de Dijkstra para encontrar la ruta más corta entre dos paraderos. Notar que esta ruta

más 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 se tiene 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 .

4.3 Parte 3: Modelo de Regresión Lineal Simple

4.3.1 Estructura del Dataset

El modelo se entrenó con 8 millones de decisiones de los 7 días de la semana. Al expandirlo en alternativas, se obtienen 47 millones de decisiones. La figura 4.8 muestra el resumen de los datos de entrenamiento del modelo.

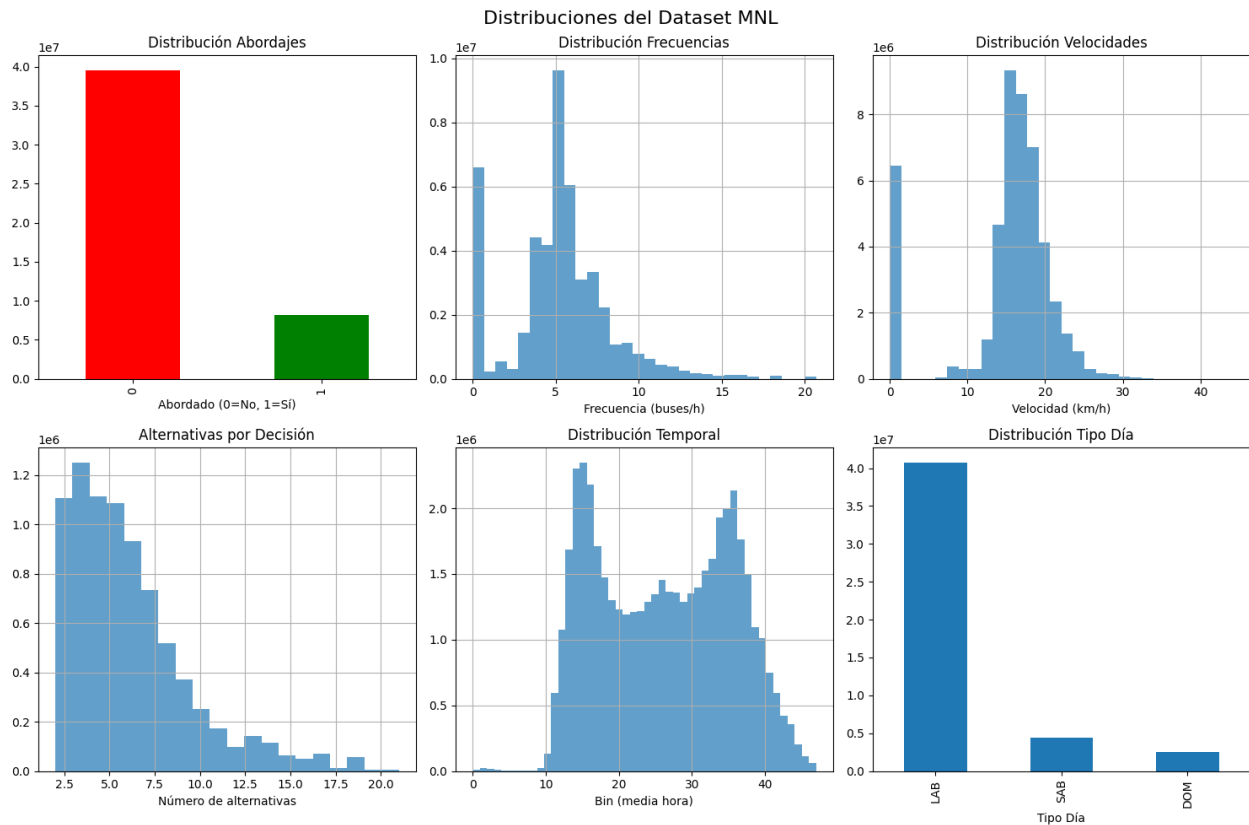


Figura 4.8: Resumen del dataset para el modelo MNL básico

Notar que en promedio hay 5 decisiones por abordaje. También, se ve 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.

4.3.2 Resultados del Entrenamiento

Los resultados del entrenamiento 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

β_0 (intercept): -1.497097
 β_1 (wait_time): -0.211864
 β_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.

4.4 Parte 4: MNL con destino

Recordar ahora que el MNL tiene más parámetros a ajustar. Estos son:

- Tiempo de espera
- Tiempo de viaje
- Coste restante

4.4.1 Resultados y coeficientes.

El entrenamiento dio los siguientes resultados.

	intercept	wait_time	viajar_cost	cost_to_go	zero_onboard	ASC_metro
Coeficiente	0.0	0.0	0.0	0.0	0.0	0.0

Tabla 4.1: Coeficientes obtenidos del modelo MNL (todos nulos).

	loglik	loglik_null	pseudo- R^2 (McFadden)	Top-1 Accuracy	# Decisiones	# Alternativas
Entrenamiento	NaN	-178245.70	NaN	0.435	151279	701136
Validación	NaN	-44693.39	NaN	0.433	37791	175648

Tabla 4.2: Métricas de entrenamiento y validación del modelo MNL.

Como se observa en las tablas, todos los coeficientes resultaron nulos y las métricas de log-likelihood y pseudo- R^2 no son numéricas (NaN), lo que indica que el modelo no logró aprender una relación significativa entre las variables y la elección observada. Sin embargo, la *top-1 accuracy* se mantiene en torno al 43%, lo que sugiere que, pese a la falta de ajuste en los coeficientes, el modelo logra predecir la alternativa elegida en una proporción considerable de los casos, posiblemente debido a la estructura de los datos o a la presencia de alternativas dominantes.

Es decir, hay un problema. La respuesta a esto está en la figura 4.9.

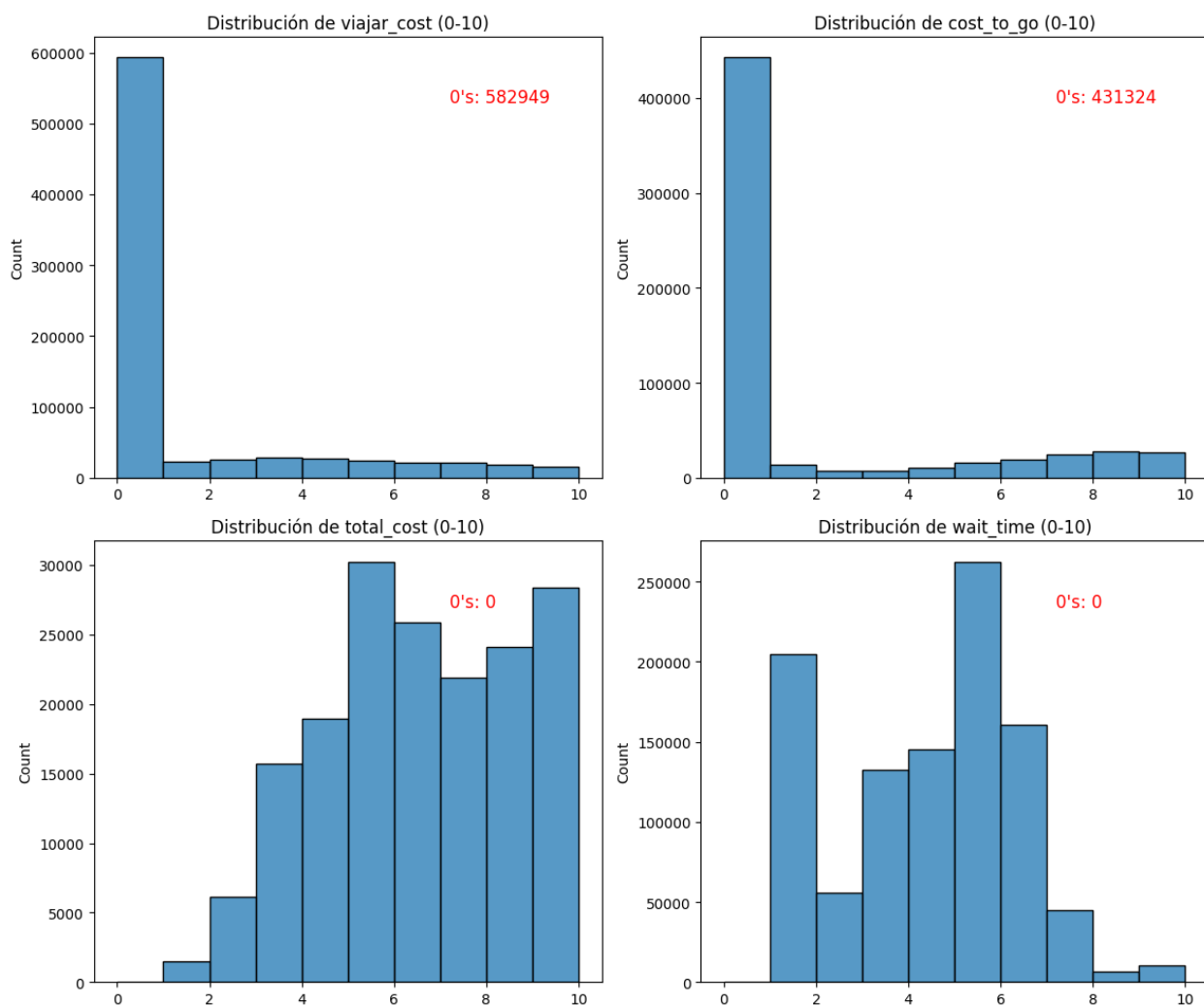


Figura 4.9: Histograma de Pesos

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 más 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.

4.4.2 Primeros Transbordos y la Penalización Inicial

Una solución elegida fue penalizar a las alternativas que requieran hacer transbordo inicial (es decir, cambiarse de paradero sin siquiera tomar el primer servicio), con un tiempo extra que se pueda configurar, esto por dos razones:

- Es más 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 más complicado, consume más memoria, y además era necesario de ya entrenar para avanzar con la memoria. Esta idea se descartó.

Se decide con aplicar una penalización de 5 minutos al tiempo inicial para evitar valores nulos. Esta es una heurística razonable.

4.4.3 Entrenamiento Diario.

Primero, para agilizar el entrenamiento, se analizó día por día. El día miércoles no estaba disponible en red. Entrenar semanalmente nos permite identificar cambios en los parámetros dependiendo del día. Ver la tabla 4.3 a modo resumen de los parámetros obtenidos.

En esta sección no se mostrarán las métricas diarias, pues consumirían mucho espacio, pero en la sección siguiente, se mostrarán las métricas de desempeño semanal. La precisión promedio de todos los días fue del 92% tanto en el split de validación como en el de entrenamiento.

Día	intercept	wait_time	viajar_cost	cost_to_go	first_walk_min	is_initial_transfer	zero_onboard	ASC_metro
lunes	5.81×10^{-13}	-0.75	0.06	-5.41	-0.10	-0.19	2.02	1.70×10^{-13}
martes	5.79×10^{-13}	-0.99	0.03	-5.63	-0.07	-0.15	2.13	1.75×10^{-13}
jueves	4.95×10^{-13}	-0.83	-0.01	-5.59	-0.07	-0.15	2.07	1.50×10^{-13}
viernes	5.94×10^{-13}	-0.96	-0.04	-5.64	-0.10	-0.20	2.13	1.79×10^{-13}
sábado	7.64×10^{-13}	-1.09	-0.16	-5.36	-0.09	-0.19	2.28	2.38×10^{-13}
domingo	7.59×10^{-13}	-0.50	-0.23	-5.11	-0.08	-0.15	2.28	2.16×10^{-13}
Promedio	6.29×10^{-13}	-0.85	-0.06	-5.46	-0.09	-0.17	2.15	1.88×10^{-13}

Tabla 4.3: Coeficientes del modelo MNL entrenado para distintos días de la semana. La última fila muestra el promedio de cada columna. Este entrenamiento fue por cada día.

4.4.4 Entrenamiento Semanal completo.

Para tener una vista general, se opta por hacer un entrenamiento general de la semana completa, tomando un split del 20% de los datos totales. La tabla 4.4 muestra la cantidad de decisiones. Con ello, se obtienen las siguientes métricas (4.5) y parámetros (4.6).

Split de Datos	Cantidad
Datos cargados (tuplas)	(5 609 970, 25)
Después de limpieza (tuplas)	(4 620 455, 25)
Train (filas)	3 696 362
Val (filas)	924 093
Decisiones (train)	814 793

Tabla 4.4: Resumen de datos y particionado

Iter	NLL	$\ \text{grad}\ $	Acc	AccNT	MRR	NLLn	R^2
pre	942 424.1203	7.395e+05	–	–	–	–	–
001	514 839.8027	2.153e+05	0.762	0.680	0.861	0.400	0.454
002	435 046.4608	1.410e+05	0.778	0.702	0.872	0.338	0.538
003	333 427.0366	6.856e+04	0.918	0.890	0.953	0.259	0.644
004	284 922.6559	3.608e+04	0.916	0.887	0.951	0.223	0.695
005	259 999.6626	1.618e+04	0.917	0.889	0.952	0.205	0.720
006	250 318.8548	8.243e+03	0.919	0.891	0.953	0.199	0.730
007	247 499.3992	4.732e+03	0.919	0.892	0.953	0.198	0.733
008	246 881.7824	6.201e+02	0.921	0.893	0.954	0.198	0.733
009	246 841.4710	2.107e+02	0.921	0.893	0.954	0.198	0.733
010	246 833.2999	9.714e+01	0.921	0.893	0.954	0.198	0.733
011	246 831.8212	1.026e+01	0.921	0.893	0.954	0.198	0.733
012	246 831.8212	1.026e+01	0.921	0.893	0.954	0.198	0.733

Tabla 4.5: Historial de optimización (iteraciones)

Parámetro	Valor
intercept	6.10933635e-13
wait_time	-9.01719464e-01
viajar_cost	8.10947587e-02
cost_to_go	-5.61161490e+00
first_walk_min	-9.16079751e-02
is_initial_transfer	-1.83215950e-01
zero_onboard	2.22291522e+00
ASC_metro	1.85152930e-13

Tabla 4.6: Coeficientes del modelo MNL

Notar como afecta más a la utilidad tener un *cost_to_go* alto que un tiempo de viaje alto. Con esto se puede concluir que los usuarios prefieren alternativas que le acerquen lo más que puedan al destino, inclusive pagando más coste de viaje que otro servicio alimentador.

Se obtienen 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 se mira desde un punto de vista de comodidad, un coste restante menor indica que el viaje tiene menos transbordos probablemente y es más directo. Entonces, un coste restante menor es más atractivo. Para tener un costo restante

Métrica	Train	Val
NLL	0.3029	0.3078
Acc	0.922	0.921
AccNT	0.895	0.893
MRR	0.954	0.954
NLLn	0.196	0.198
R^2	0.738	0.733

Tabla 4.7: Métricas de la mejor época de la MNL con Destino Semanal

menor, es necesario viajar más tiempo en el primer servicio.

No se logran observar diferencias sustanciales entre los días de semana y fines de semana. Un análisis usando más semanas debe de ser imperativo para extraer conclusiones en cuanto a este tema.

4.5 Parte 5: GNN

4.5.1 Resultados

A continuación se presentan los resultados de la GNN en sus dos modos, el primero para cuando se agrega a los embeddings los features de Dijkstra, y posteriormente cuando se omiten estos features.

GNN con Features de Dijkstra

Elemento	Valor
Embedding destinos	10 687 (dim = 128)
Scorer dinámico	393 \rightarrow 128 \rightarrow 1
Parámetros entrenables	\approx 1 717 765

Tabla 4.8: Resumen del modelo

Época	Train NLL	Train Acc	Train AccNT	Train MRR	Val NLL	Val Acc	Val AccNT	Val MRR
01	0.1166	0.948	0.870	0.970	0.0924	0.960	0.900	0.977
02	0.0839	0.956	0.890	0.975	0.0856	0.961	0.901	0.977
03	0.0806	0.956	0.891	0.975	0.0852	0.961	0.902	0.978
04	0.0800	0.957	0.892	0.975	0.0838	0.961	0.903	0.978
05	0.0799	0.957	0.891	0.975	0.0833	0.961	0.902	0.978
06	0.0793	0.957	0.892	0.975	0.0842	0.961	0.902	0.978
07	0.0793	0.957	0.891	0.975	0.0829	0.961	0.902	0.978
08	0.0790	0.957	0.892	0.975	0.0831	0.961	0.903	0.978
09	0.0788	0.957	0.891	0.975	0.0830	0.961	0.903	0.978
10	0.0787	0.957	0.891	0.975	0.0822	0.961	0.902	0.978
11	0.0786	0.957	0.891	0.975	0.0812	0.961	0.902	0.978
12	0.0785	0.957	0.892	0.975	0.0822	0.962	0.903	0.978
13	0.0784	0.957	0.892	0.975	0.0816	0.961	0.902	0.978
14	0.0786	0.957	0.892	0.975	0.0815	0.961	0.903	0.978
15	0.0788	0.957	0.891	0.975	0.0826	0.961	0.903	0.978
16	0.0786	0.957	0.891	0.975	0.0825	0.962	0.903	0.978
17	0.0784	0.957	0.892	0.975	0.0810	0.961	0.903	0.978
18	0.0783	0.957	0.892	0.975	0.0811	0.962	0.903	0.978
19	0.0783	0.957	0.892	0.975	0.0813	0.961	0.903	0.978
20	0.0780	0.957	0.892	0.976	0.0817	0.962	0.903	0.978

Tabla 4.9: Historial de entrenamiento por época

Con ello se obtiene un modelo en que en su mejor época tiene unas métricas de Precisión del 96.2% y Precisión No Trivial del 90.3%.

GNN sin Features de Dijkstra.

Elemento	Valor
Embedding destinos	10687 (dim = 128)
Scorer dinámico	384 \rightarrow 128 \rightarrow 1
Parámetros entrenables	\approx 1,716,613

Tabla 4.10: Resumen del modelo

Epoch	Train NLL	Train Acc	Train AccNT	Train MRR	Val NLL	Val Acc	Val AccNT	Val MRR
01	0.5828	0.737	0.342	0.827	0.5810	0.742	0.352	0.831
02	0.5724	0.740	0.348	0.829	0.5717	0.745	0.359	0.833
03	0.5672	0.741	0.351	0.830	0.5652	0.746	0.362	0.834
04	0.5640	0.743	0.356	0.831	0.5594	0.752	0.377	0.838
05	0.5497	0.748	0.367	0.835	0.5453	0.756	0.388	0.842
06	0.5361	0.751	0.376	0.838	0.5371	0.759	0.394	0.844
07	0.5275	0.752	0.379	0.839	0.5258	0.761	0.400	0.846
08	0.5239	0.754	0.382	0.841	0.5333	0.762	0.401	0.847
09	0.5211	0.754	0.384	0.841	0.5221	0.764	0.408	0.849
10	0.5190	0.755	0.385	0.842	0.5272	0.764	0.408	0.849
11	0.5178	0.755	0.387	0.842	0.5273	0.763	0.405	0.848
12	0.5162	0.756	0.388	0.842	0.5222	0.765	0.410	0.849
13	0.5152	0.756	0.389	0.843	0.5234	0.764	0.408	0.849
14	0.5150	0.757	0.390	0.843	0.5238	0.765	0.410	0.849
15	0.5142	0.757	0.390	0.843	0.5170	0.766	0.412	0.850
16	0.5134	0.757	0.391	0.844	0.5196	0.767	0.414	0.851
17	0.5115	0.758	0.392	0.844	0.5193	0.766	0.412	0.850
18	0.5121	0.758	0.393	0.844	0.5152	0.767	0.414	0.851
19	0.5118	0.758	0.392	0.844	0.5163	0.767	0.415	0.851
20	0.5109	0.758	0.394	0.844	0.5117	0.767	0.415	0.851

Tabla 4.11: Historial de entrenamiento por época

Discusión de los Resultados

- Claramente se observa una diferencia entre colocar una capa de Dijkstra versus no colocarla. Sin los datos de Dijkstra, el modelo no consigue buenos resultados.
- Comparado con el MNL, el cual tuvo una precisión NT del 90% (-1%) y una precisión del 93% (-3%), hay una leve mejora aportada por los embeddings. Se concluye que

Mejor época (validación)	20 (Val NLL: 0.5117)
Test NLL	0.5169
Test Acc	0.766
Test AccNT	0.415
Test MRR	0.850

Tabla 4.12: Mejor modelo y evaluación en test

4.6 Parte 6: Experimentos

Se realizaron experimentos para mostrar redistribución de demanda. Para ello, se usarán los siguientes coeficientes obtenidos de un día viernes. No se usó la GNN debido a que tomaba más tiempo y que los resultados obtenidos eran muy parecidos a los de la MNL cuando tenía la capa de Dijkstra activada. Como ambos tienen resultados parecidos, la MNL es más interpretable y además tarda menos tiempo en generar resultados, se decide por usar la MNL como motor de utilidad.

Coeficiente	Valor
intercept	5.94×10^{-13}
wait_time	-0.96
viajar_cost	-0.04
cost_to_go	-5.64
first_walk_min	-0.10
is_initial_transfer	-0.20
zero_onboard	2.13
ASC_metro	1.79×10^{-13}

Estos valores reflejan la importancia relativa de cada atributo en la elección de alternativas de viaje según el modelo MNL entrenado. Para el predictor solo se usarán los coeficientes wait_time, viajar_cost, cost_to_go y first_walk_min.

4.6.1 Experimento 1: Disminución de oferta de un servicio.

Se tiene un paradero P y Q conectados por un set de servicios {S} para un bin b. En el *baseline* (la oferta real) se obtiene una distribución de probabilidad dada. Si se modifica la oferta de uno de los servicios, por ejemplo, aumentando el doble el tiempo de espera (disminuyendo la cantidad de buses que operan el servicio) se obtiene una comparación entre las distribuciones de probabilidades para antes y después del cambio de oferta. Se muestran dos ejemplos ilustrativos.

Ejemplo 1: Ir desde PJ394 a PA300

Ambos paraderos tienen de servicios disponibles que dejan directo en el destino, el 503 y el 517. Entonces, el costo restante o *cost_to_go* es 0, ya que dejan directamente en el destino del usuario. Ver figura 4.10 que ilustra los tiempos de cada servicio del paradero. El experimento consiste en aumentar al doble el tiempo de espera del 517.

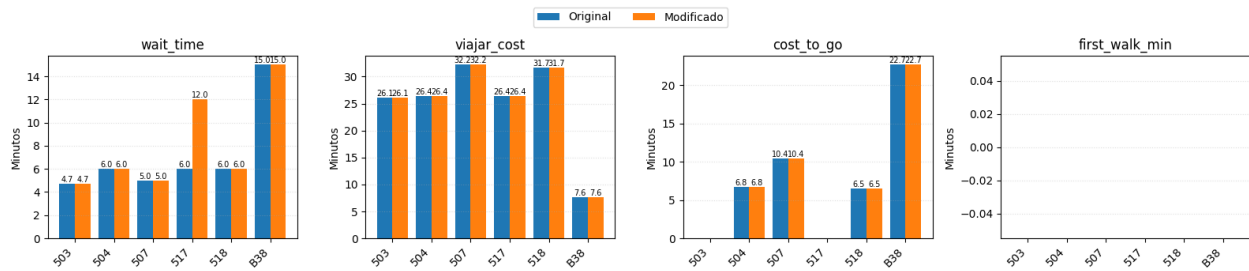


Figura 4.10: Costes para ir desde PJ394 a PA300. En azul se muestran los costes reales y en naranja los cambios de oferta

Si se ejecuta el predictor, se obtiene una redistribución de probabilidades como la mostrada en la figura 4.11.

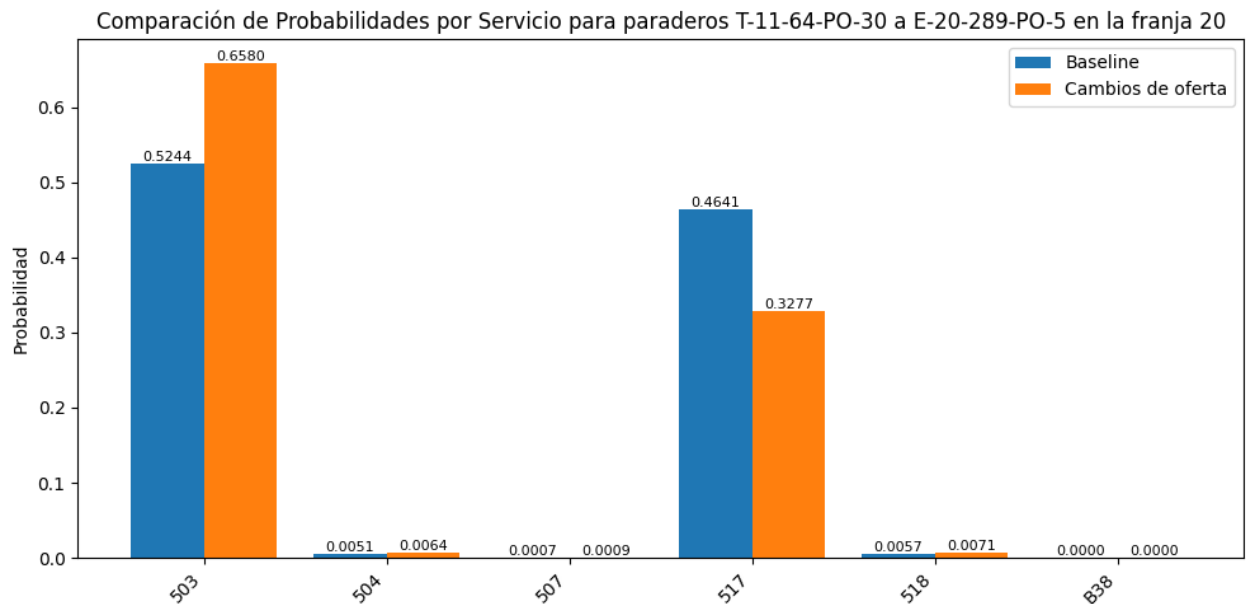


Figura 4.11: Distribución de probabilidades para alternativas de viaje antes y después del cambio de oferta

Notar como el servicio 503 pierde probabilidad y el 517 la gana. Pero no es una transferencia directa. Los otros servicios igual ganan un poco de atractivo al perderlo el 503.

Ejemplo 2: Ir desde PJ394 a PA433

Este ejemplo es distinto. A diferencia del anterior, efectivamente solo un servicio llega directamente al destino, el 507. El resto entonces, tiene un costo restante mayor que cero. Ver figura 4.12 que ilustra los tiempos de cada servicio del paradero.

Si ejecutamos el MNL, obtenemos una redistribución de probabilidades como la mostrada en la figura 4.13.

Notar que no cambia mucho la probabilidad del servicio 507. A pesar de que su tiempo de

espera se duplica, sigue siendo la mejor alternativa.

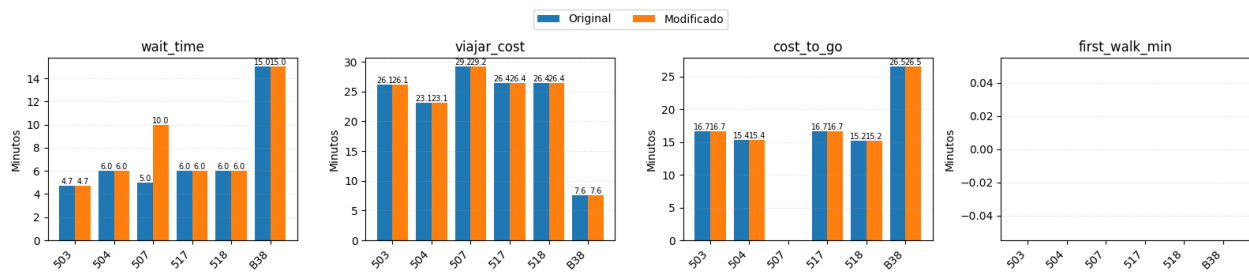


Figura 4.12: Costes para ir desde PJ394 a PA433

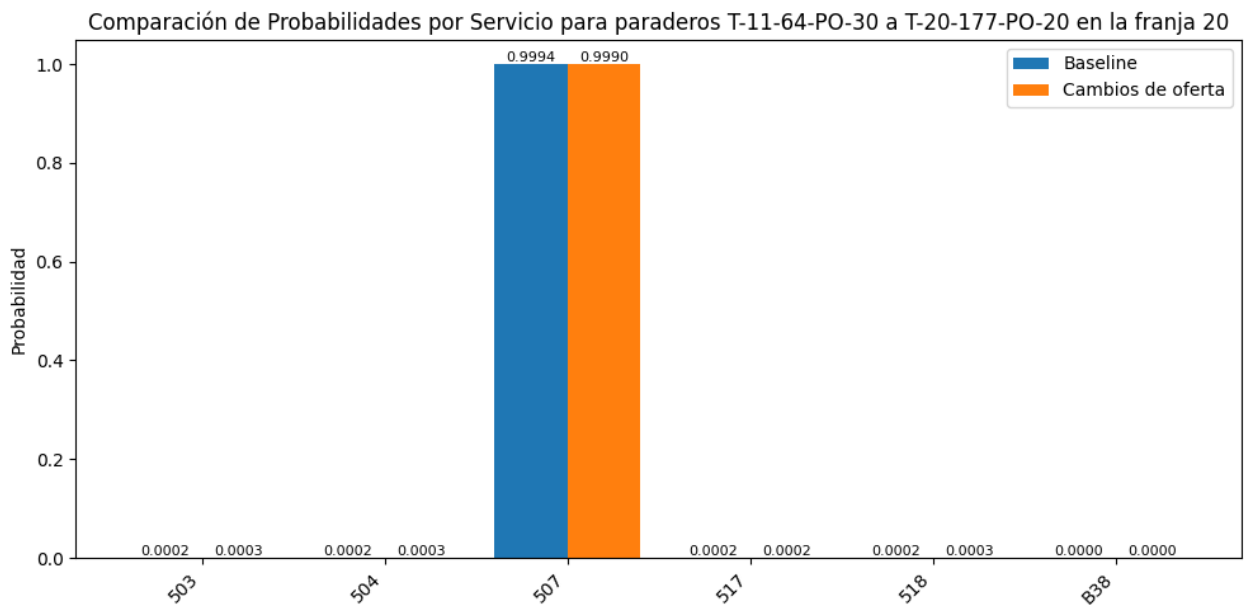


Figura 4.13: Distribución de probabilidades para alternativas de viaje antes y después del cambio de oferta (Experimento 2)

Se observa que:

- Dos servicios *compiten* cuando tienen costes de viaje y restantes parecidos. Es decir, dos servicios que llevan directamente al destino o tienen trayectos parecidos. Un ejemplo de esto es el 503 y el 517. El coeficiente del costo restante evidencia este comportamiento. Viajes sin transbordos son altamente atractivos para los usuarios.
- Cuando hay servicios que compiten, se obtiene una redistribución de la demanda más notoria como fue el caso del 503 vs el 517. En el caso del 507, al no tener ningún servicio que compita directamente, un mayor tiempo de espera no influye en lo atractivo del servicio.

Se puede seguir aumentando el tiempo de espera, hasta un 1500% más grande que el original (esto haría que el tiempo de espera pase a 100 minutos). Se obtiene una redistribución de probabilidades como la que sigue en la figura 4.14.

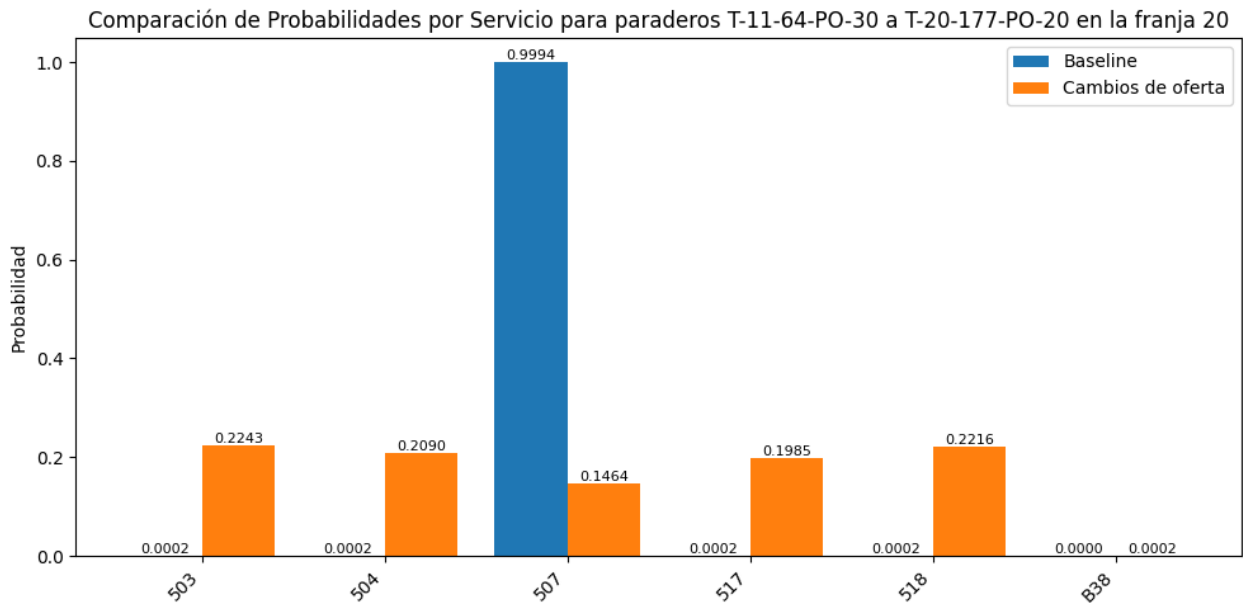


Figura 4.14: Distribución de probabilidades para alternativas de viaje con un aumento del 1500% en el tiempo de espera del servicio 507

Esto causará un efecto dominó que cambiará los transbordos siguientes. Por un efecto de simplicidad, el siguiente paso de decisión será determinístico. Cuando un usuario se baje en un paradero dado para hacer transbordo, se tomará el siguiente servicio de manera segura, y no con probabilidades. (Si no, sería una cadena de probabilidades condicionales que complicaría mucho el problema).

Para cada alternativa, no solo aumentará la demanda del servicio dado, si no que su transbordo aumentará también de demanda. En el caso de ir de PJ394 a PA433, los servicios que aumentaron su demanda alimentarán a los siguientes servicios en su transbordo. Para ello, se verán los caminos de cada servicio obtenidos por Dijkstra. La tabla 4.13 muestra los caminos que toma cada alternativa junto con la diferencia de probabilidad entre el *baseline* y el cambio de oferta. Un análisis indica que los servicios que aumentaron su demanda propagaran este aumento de demanda a los transbordos, en este caso, fijarse en 503, 504, 517 y 518. Estos recorridos dejan a usuarios en L2, por lo que es sensato concluir que un aumento de tiempo de espera en 507 provoca un aumento de demanda de L2 sujeto a que las personas se suban a PJ394.

Una vez se bajen en Parque Ohiggins, un efecto importante ocurre en el paradero aledaño a la estación de Metro. El AD predice que se tomará el 506, pero realmente es el servicio más probable a ser tomado, no necesariamente todos lo tomarán. Esta nueva demanda redistribuida se repartirá en los servicios que pasan por este paradero y que llevan a Beauchef, en este caso, el 506, 506v, 506e y el 507. Ver la tabla 4.13 para visualizar los trayectos y su cambio en su probabilidad.

Alternativa (Porcentaje en comparación al *baseline*)	Itinerario
B38 (+ 0% de probabilidades)	<p>Inicia en paradero T-11-64-PO-30</p> <p>Subir al servicio B38 (Ida) en T-11-64-PO-30</p> <p>Bajar en T-8-64-PO-30</p> <p>Subir al servicio 507 (Ida) en T-8-64-PO-30</p> <p>Bajar en T-20-177-PO-20</p>
503 (+ 22% de probabilidades)	<p>Inicia en paradero T-11-64-PO-30</p> <p>Subir al servicio 503 (Ida) en T-11-64-PO-30</p> <p>Bajar en E-20-289-PO-5</p> <p>Caminar de E-20-289-PO-5 a METRO_CAL Y CANTO</p> <p>Subir al servicio L2 (Metro) en METRO_CAL Y CANTO</p> <p>Bajar en METRO_PARQUE OHIGGINS</p> <p>Caminar de METRO_PARQUE OHIGGINS a E-20-189-OP-40</p> <p>Subir al servicio 506 (Ret) en E-20-189-OP-40</p> <p>Bajar en T-20-177-OP-8</p> <p>Caminar de T-20-177-OP-8 a T-20-177-PO-20</p>
504 (+ 20% de probabilidades)	<p>Inicia en paradero T-11-64-PO-30</p> <p>Subir al servicio 504 (Ida) en T-11-64-PO-30</p> <p>Bajar en T-20-188-NS-10</p> <p>Caminar de T-20-188-NS-10 a METRO_SANTA ANA</p> <p>Subir al servicio L2 (Metro) en METRO_SANTA ANA</p> <p>Bajar en METRO_PARQUE OHIGGINS</p> <p>Caminar de METRO_PARQUE OHIGGINS a E-20-189-OP-40</p> <p>Subir al servicio 506 (Ret) en E-20-189-OP-40</p> <p>Bajar en T-20-177-OP-8</p> <p>Caminar de T-20-177-OP-8 a T-20-177-PO-20</p>
517 (+ 19% de probabilidades)	<p>Inicia en paradero T-11-64-PO-30</p> <p>Subir al servicio 517 (Ida) en T-11-64-PO-30</p> <p>Bajar en E-20-289-PO-5</p> <p>Caminar de E-20-289-PO-5 a METRO_CAL Y CANTO</p> <p>Subir al servicio L2 (Metro) en METRO_CAL Y CANTO</p> <p>Bajar en METRO_PARQUE OHIGGINS</p> <p>Caminar de METRO_PARQUE OHIGGINS a E-20-189-OP-40</p> <p>Subir al servicio 506 (Ret) en E-20-189-OP-40</p> <p>Bajar en T-20-177-OP-8</p> <p>Caminar de T-20-177-OP-8 a T-20-177-PO-20</p>
518 (+ 22% de probabilidades)	<p>Inicia en paradero T-11-64-PO-30</p> <p>Subir al servicio 518 (Ida) en T-11-64-PO-30</p> <p>Bajar en T-20-203-NS-20</p> <p>Caminar de T-20-203-NS-20 a METRO_SANTA ANA</p> <p>Subir al servicio L2 (Metro) en METRO_SANTA ANA</p> <p>Bajar en METRO_PARQUE OHIGGINS</p> <p>Caminar de METRO_PARQUE OHIGGINS a E-20-189-OP-40</p> <p>Subir al servicio 506 (Ret) en E-20-189-OP-40</p> <p>Bajar en T-20-177-OP-8</p> <p>Caminar de T-20-177-OP-8 a T-20-177-PO-20</p>
507 (- 86% de probabilidades)	<p>Inicia en paradero T-11-64-PO-30</p> <p>Subir al servicio 507 (Ida) en T-11-64-PO-30</p> <p>Bajar en T-20-177-PO-20</p>

Tabla 4.13: Itinerarios de las alternativas desde T-11-64-PO-30 hasta T-20-177-PO-20

Notar que si hay 100 personas que quieren ir a Beauchef en un día, las 100 tomarían el 507 en el caso base. En el caso modificado, aumentaríamos la demanda del día en 80 para L2 y para 506. Esto es el efecto dominó del que se comentó al comienzo del informe que se debería de analizar.

Cuantificar los cambios de demanda en cuando hay cambios de oferta se vuelven interesantes cuando se prueban situaciones más realistas. Podemos por ejemplo, cortar la línea 1. Esto es lo que se hará en el siguiente ejemplo.

4.6.2 Experimento 2: Suspensión de un servicio.

Para el siguiente experimento, se colocará un indicador booleano que desactive todas las aristas SUBIR, BAJAR Y VIAJAR de la L1. Haciendo esto, tomemos el caso de alguien que quiera ir de San Pablo a Baquedano.

Las figuras 4.15 y 4.16 muestran las probabilidades y costes para cada alternativa. Notar como las probabilidades de usar la L1 bajan y las de la L5 suben. Ojo que estas probabilidades están condicionadas a que el usuario haya decidido ir a Baquedano.

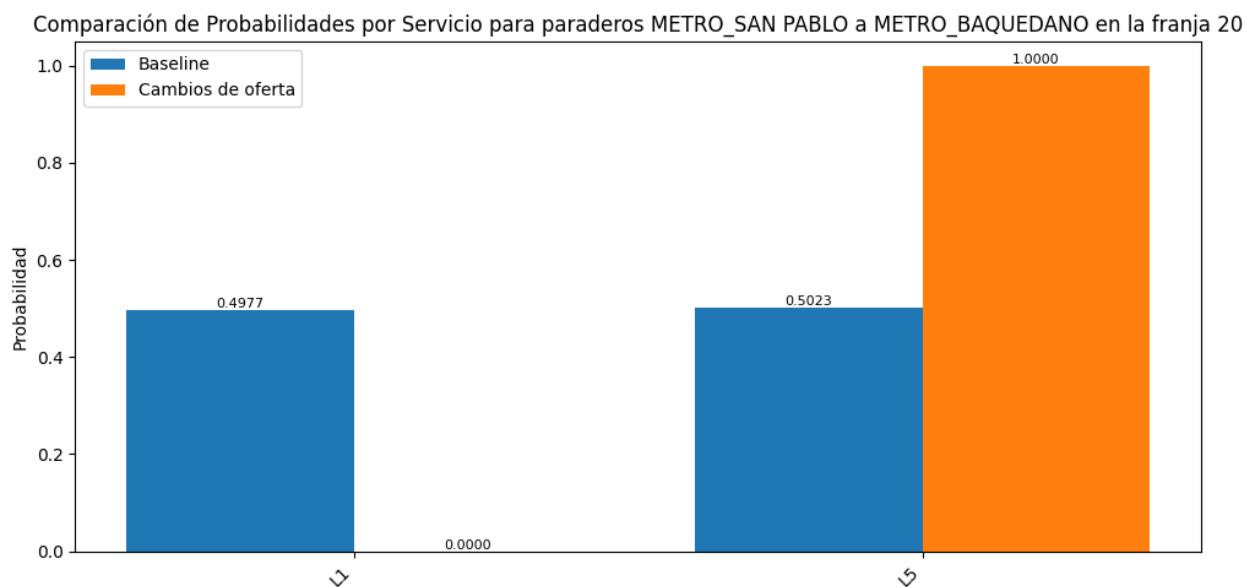


Figura 4.15: Distribución de probabilidades para alternativas de viaje con suspensión de la Línea 1

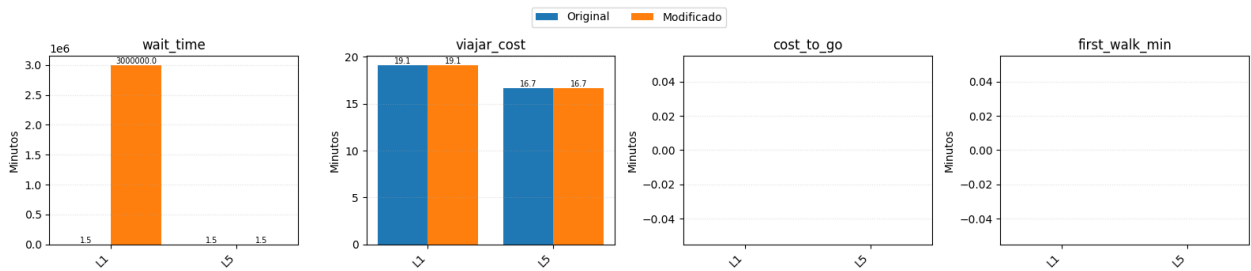


Figura 4.16: Costes para ir de San Pablo a Baquedano con suspensión de la Línea 1

Algo más interesante pasa cuando se quiere ir a una estación de L1 que no combine con L5, es decir, ambos servicios no compiten. Por ejemplo, ir de San Pablo a Tobalaba.

Viaje en alternativa: L5

- **Inicio:** Paradero METRO_SAN PABLO
 - Subir al servicio **L5** (sentido Metro) en METRO_SAN PABLO
 - Bajar en METRO_BAQUEDANO
 - Caminar desde METRO_BAQUEDANO hasta E-20-53-PO-115
 - Subir al servicio **503** (sentido Ida) en E-20-53-PO-115
 - Bajar en E-14-170-NS-5
 - Caminar desde E-14-170-NS-5 hasta METRO_TOBALABA
- **Fin del camino**

El algoritmo que se tiene, por construcción tomará el camino con el coste más bajo para ir desde Baquedano hacia Tobalaba si es que el Metro está desactivado. Se puede hacer este análisis corriendo el algoritmo del MNL desde el paradero E-20-53-PO-115 (el más cercano a Baquedano que tiene servicios que dejan cerca, según el enrutador). Las figuras 4.17 y 4.18 muestran los costes y probabilidades para cada alternativa. Notar como el servicio 503 es el más atractivo, ya que es el que tiene el menor coste total. Una suspensión de la L1 entre Baquedano y Tobalaba sugiere que todo el volumen de pasajeros que usualmente toma este tramo se redistribuirá en los servicios en superficie con las probabilidades de más abajo.

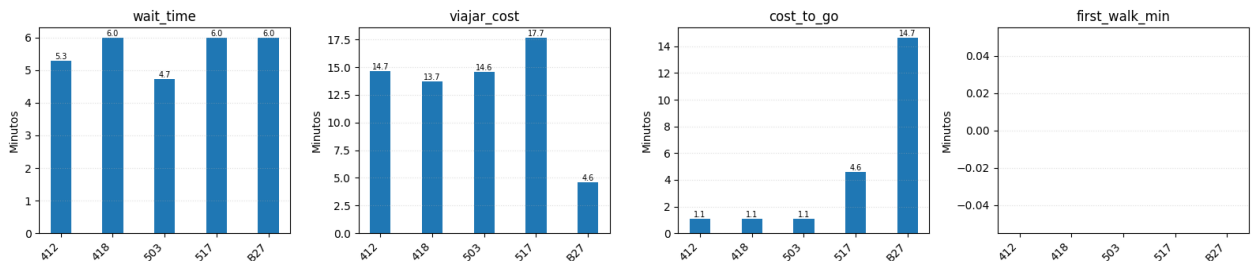


Figura 4.17: Costos del paradero E-20-53-PO-115 a Tobalaba con suspensión de la Línea 1

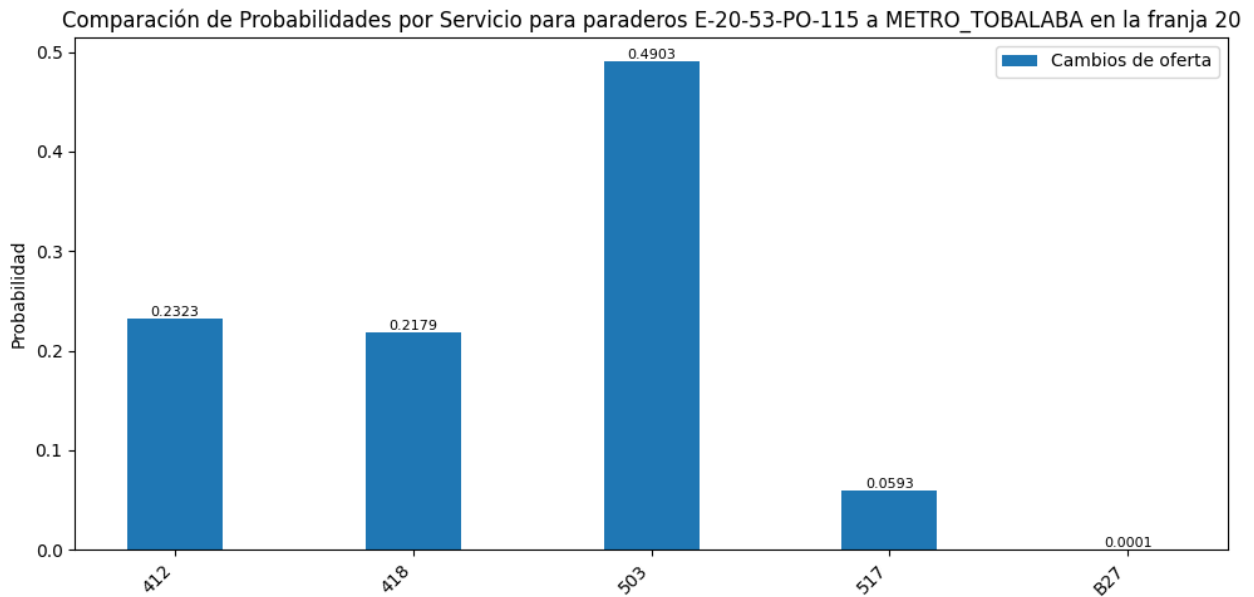


Figura 4.18: Distribución de probabilidades para alternativas con suspensión de la Línea 1

Al ver esta redistribución, la primera medida a tomar sería reforzar con recorridos como el 412, 418 y 503 pues tienen un `cost_to_go` bajo (es decir, acercan más a Tobalaba), en cambio, no reforzar servicios con recorridos similares al 517 y B27, pues por probabilidad, no deberían de ser elegidos para llegar a Tobalaba.

Este análisis solo funciona para las personas que quieran tomar el metro en Baquedano. Un análisis más profundo debería de considerar a todas las personas que quieran tomar algún tramo de la Línea 1 y se encuentren con un corte de servicio. Por ejemplo, una distribución de usuarios desde Baquedano a Tobalaba por cada estación intermedia tomaría distintos servicios que le dejen en Tobalaba dependiendo del origen y sus estrategias de elección.

4.6.3 Experimento 3: Agregar Línea 7

Para el siguiente experimento, se agregó la L7 y se obtuvieron las siguientes métricas:

Número de Etapas promedio

Nº Etapas Promedio (Real): 1.73 Nº Etapas Promedio (Sintético): 2.15

Hay un aumento de etapas. Esto debido a la inclusión de un servicio nuevo. La L7 actúa como un servicio intermedio interesante para los usuarios.

Uso de servicios (Líneas de Metro)

Se procede a comparar el uso antes y después de la L7. Sintético se refiere a este último.

Servicio	Etapas (Real)
L1	880 685
L5	624 255
L2	435 869
L4	371 041
L3	365 217
L6	173 456
L4A	71 110
T506 00I	10 168
T506 00R	9859
T556 00R	9715

Tabla 4.14: Uso de Servicios (Etapas Reales)

Servicio	Etapas (Sintético)
L1	1 173 755
L5	791 909
L2	557 354
L3	492 883
L4	480 813
L6	193 936
L4A	90 910
506	56 032
406	48 794
L7	40 516

Tabla 4.15: Uso de Servicios (Etapas Sintéticas)

Se observa un aumento sustancial en el uso del Metro en general. Esto puede ser causa de los siguientes puntos:

- L7 disminuye el uso del servicio de buses. Estos usuarios antes llegaban directamente a su destino usando una micro. Ahora, usando la L7, inyectan demanda directamente al servicio subterráneo, haciendo que la demanda nueva ganada por L7 se esparza por toda la red.
- Las líneas de metro comienzan a alimentar la L7, ya que esta está bien conectada con el resto de la red, haciendo más apetecibles los viajes en metro en general.

Uso de la L7 como primera etapa

No se registraron viajes con L7 en primera etapa, esto por la restricción del paradero inicial y el coste de transbordo inicial. Se ahondará más en esta decisión en la discusión final.

Servicios Alimentadores de L7

Los servicios alimentadores se definen como los que son transbordo directo a L7.

Servicio Alimentador (Etapa k-1)	N° de Viajes a L7 (Etapa k)
L1	5203
L5	4053
406	3311
L2	2533
426	1814
L3	1656
405	1617
508	1465
C01	1186
J01	1142

Tabla 4.16: Servicios Alimentadores Principales de L7 (Transbordos)

Notar como servicios que comparten recorrido con la L7 (por ejemplo, el 508 que recorre junto a la L7 desde Mapocho con Huelén hasta Salvador) sirven como alimentadores. Esto debido a que la L7 tiene estaciones más espaciadas que la L1, haciendo que es recomendable tomar un recorrido que acerque a caminar.

Servicios alimentadores como el J01 son del tipo que recogen a usuarios los cuales no viven cerca del trayecto de la L1. Por ejemplo, J01 recorre toda la avenida Neptuno desde Carrascal hasta General Bonilla. La mayoría de los usuarios tomaba J01 para acercarse a Metro San Pablo. Ahora, la usarán para acercarse a Metro Cerro Navia. Es esperable notar un aumento de demanda en J01.

Comparación de Carga Total (Número de Aristas VIAJAR recorridas por servicio)

Línea	Total Estaciones (Real)	Total Estaciones (Sintético)	Diferencia
L6	948 250	984 713	36 463
L4A	371 047	407 921	36 874
L4	3 585 554	3 633 894	48 340
L2	3 928 697	4 020 191	91 494
L5	6 095 884	6 228 347	132 463
L1	7 546 426	7 725 747	179 321
L3	2 464 663	2 734 873	270 210
L7	–	383 674	383 674

Tabla 4.17: Comparación de Carga de Metro (Total Estaciones-Pasajero Recorridas)

Hubo un aumento de carga en toda la red por lo anteriormente dicho. La L7 ganó viajes, pero no tanto como se esperaba.

Viendo todas estas comparaciones, se observan ciertas desviaciones por lo que se espera de esta línea y su desempeño. Esto era, una baja en la demanda de L1. Razones para esto tienen que ver con la solución propuesta y como introduce artefactos. El coste del transbordo inicial y la restricción de los puntos extremos fijos (el origen y el destino) que hacían como anclas del camino final indujeron desviaciones. En una situación real, una persona podría cambiar su paradero de viaje inicial para llegar a L7 y tomarla como etapa 1. Este comportamiento es difícil de replicar, pues no se sabe donde vive cada persona.

De todas maneras, un efecto interesante descubierto, y no predicho en las hipótesis, es como la L7 le quita demanda a servicios de buses. Esta redistribución inyecta más usuarios a la red de Metro.

También se nota un aumento de usos de servicios alimentadores, tal como se mencionó en la introducción y en la literatura.

4.7 Limitaciones del Modelo

Este modelo tiene varias limitaciones que no serán resueltas en esta memoria, pero vale la pena discutir. Por cada apartado se menciona entre paréntesis

Transbordos determinísticos

Los transbordos o viajes con más de una etapa fueron tratados de manera determinista en sus etapas posteriores a la inicial, esto quiere decir que después de bajarse, el costo restante es definido de manera estricta. Una solución interesante puede ser concatenar varios MNL para cada etapa, pero esto complica mucho el problema. Notar que este enfoque habría hecho el valor *cost_to_go* no determinado, si no que una distribución o valor esperado. En esta memoria el *cost_to_go* es el mínimo dado que el usuario se baja en el paradero óptimo y elija el servicio óptimo. Es una simplificación fuerte, pero que funciona en gran parte de las decisiones (notar el 92% de precisión obtenido).

Correlación Espacial (Para el MNL)

Tal como se mencionó al inicio de la memoria, un usuario puede decidir en base a clusters o paraderos cercanos que le proveen una mejor oferta. El caso de preferir paraderos con más servicios competitivos que otros puede ser un factor importante, tanto, que el usuario puede preferir a caminar más para tener un tiempo de espera más corto o confiable. Efectos como los de preferir el Metro por sobre otros modos de viaje debido a su fiabilidad y su alta frecuencia no son modelados.

Elección de Paradero Inicial

Esta limitación viene más por el lado de los datos. Lógicamente no se sabe donde vive la gente, solo su paradero de inicio del viaje y el del final de éste. Por ello, el viaje ya viene condicionado a que se eligió un paradero determinado desde el comienzo. Esto causa que nuevos servicios agregados nunca tengan demanda en la etapa 1, pues ningún viaje comienza en paraderos que recorren, a no ser que sean buses, caso que no se exploró en esta memoria.

Coste de la Primera Transferencia

El modelo penaliza de la misma manera caminar diez minutos reales para elegir el paradero, que caminar un minuto. Esto causa que la elección de la Línea 7 sea más acotada, pues los usuarios comienzan su viaje en el paradero real que eligieron, y cambiar a la Línea 7, aunque quede a un par de metros, sea más costoso que tomar un servicio y combinar más adelante. Esto contamina la elección real más probable.

Comodidad

Un factor importante no modelado. La comodidad puede verse afectada dinámicamente según la hora y el momento del recorrido. Un servicio que va lleno es menos cómodo que uno que va vacío. Modelar el llenado de los buses requeriría saber exactamente el tamaño de los buses, el tipo de flota que tiene cada servicio y que tipo de buses saca cada servicio dependiendo del bin.

Velocidad promedio v/s Velocidad por arista

Los datos entregados por red nos muestran datos de velocidad promedio en todo el recorrido. Claramente hay trazos del recorrido más lentos que otros, probablemente los más lentos son en áreas céntricas mientras que los rápidos son en áreas suburbanas. Esto puede afectar localmente en viajes cortos, tomando costos de viaje más pequeños que los reales. Este efecto se puede amortiguar en viajes cortos cuando los servicios que compiten en las alternativas comparten el eje de circulación, pero por ejemplo un servicio que no usa avenidas puede verse perjudicado ante uno que alcanza velocidades mayores.

Tarifas

No se tuvo en cuenta el coste del viaje. Esto afecta claramente el uso del servicio. Personas pueden preferir evitar el metro pues es más costoso.

Capítulo 5

Conclusión

Para concluir, es importante recordar el Problema y el Objetivo general de esta memoria:

Partiendo desde el problema

Predecir cambios en la demanda implica saber múltiples decisiones que no se pueden localizar, pues cambios de oferta pueden afectar de muchas maneras, casi como un efecto dominó

Se tiene como objetivo:

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.

Desde ese contexto y objetivo, se abordan dos métodos. Uno del MNL y otro el MNL/GNN.

Para ambos enfoques, la solución se enfocó en predecir una alternativa a usar. Esto, en base a su origen, destino (inamovibles) y el día.

El MNL permitió un análisis cuantitativo interpretable acerca de las variables que el autor de la memoria consideró importantes. Estos son, el tiempo de viaje, el coste restante de viaje al transbordar y el tiempo de espera. En este ámbito, se observó algo interesante. Las personas prefieren viajar más tiempo si eso significa minimizar el coste restante, la variable que más pesaba al seleccionar una alternativa. En otras palabras, las personas evitan hacer transbordos.

El GNN presenta mejores resultados para predecir, en parte gracias a sus embeddings y su correlación espacial, aunque no se usó en los experimentos, debido a su lentitud. De todas maneras, la precisión del MNL es suficientemente parecida (uno por ciento de diferencia) como para poder generalizar sin preocupaciones.

Los experimentos muestran tanto las fortalezas y las debilidades de la solución expuesta.

Primero, no se observó una clara diferencia entre los coeficientes para distintos días. Un análisis de más semanas es necesario para ver una tendencia. Por otro lado, dos servicios compiten (tienen probabilidades parecidas) en general cuando tienen costes restantes parecidos. Servicios

no compiten cuando uno tiene coste restante muy bajo y otro muy alto. Para que estos servicios se vuelvan competidores entre si, el coste de espera del dominante debe de aumentar bastante para que valga la pena hacer el transbordo del otro servicio. Esto resume la preferencia para minimizar el coste restante y su dominancia sobre los atributos necesarios para una decisión.

La redistribución de demanda es difícil de analizar de manera local. Esto debido a la determinancia de las decisiones después de seleccionar el primer transbordo. Un análisis local sobre la redistribución revela como hay un aumento de demanda en servicios que combinan con los servicios elegidos.

De manera global, la redistribución de demanda en el caso de la Línea 7 mostró resultados no esperados. Por un lado, el aumento del uso del metro en general, un hallazgo interesante. Esto gracias a la captación de demanda de buses que hacían el recorrido largo. Esto se puede ver reflejado en el aumento de etapas en promedio. Además, se lograron observar los alimentadores y como se relacionan con la L7. Estos eran los buses que se esperaba que aumentaran su demanda al llevar a personas hacia la L7. El aumento de carga en la L1 es un hallazgo fascinante. Se puede explicar por las restricciones del coste inicial de transferencia y como también las líneas del metro ahora son más llamativas para el usuario.

Las limitaciones de este trabajo pasan por las condiciones de borde de los viajes (origen y destino fijos), el coste de transbordo inicial y por las predicciones de bajada de ADATRAP. Estas decisiones, algunas causadas por los datos y otras por errores en la planeación de la solución, deben de ser tomadas en cuenta para interpretar los resultados.

Para finalizar, se concluye que se logró modelar la decisión de las personas, y estudiar la redistribución de la demanda causada por cambios en la oferta, sujeta a decisiones en la solución que pueden condicionar el análisis posible sobre estos resultados.

Para trabajo futuro, sería interesante desacoplar a los usuarios de los paraderos iniciales y finales, para establecer zonas o cuadrantes en donde tienen la libertad de elegir el paradero inicial al que ir. Esto solucionaría el problema encontrado en las etapas 1 de la L7 (que eran inexistentes) y se podría eliminar la simplificación del coste inicial.

Bibliografía

- [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] 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.
- [5] 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>.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] SmartcitySantiagoChile 2025. ADATRAP: Herramienta para análisis de datos masivos de transporte público.
- [10] Soto, F.J.M. 2023. *Estimación y análisis de modelos de demanda agregada para el transporte público en santiago de chile*. Universidad de Chile.
- [11] 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>.
- [12] 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.
- [13] 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>.
- [14] 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.
- [15] 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.

Anexos