UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FíSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA

ESTIMACIÓN DE FLUJOS DE EVASIÓN VÍA ESCALAMIENTO DE MATRICES

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MATEMÁTICO

FAN WANG

PROFESOR GUÍA:
JOSÉ RAFAEL CORREA

MIEMBROS DE LA COMISIÓN:
HÉCTOR RAMÍREZ CABRERA
ANDREAS WIESE

Este trabajo ha sido parcialmente financiado por FONDEF IT16I10010 "Desarrollo de tecnologías para el control de la evasión en el transporte público"

SANTIAGO DE CHILE
2017

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO
POR: FAN WANG
FECHA: 2017
PROF. GUÍA: JOSÉ RAFAEL CORREA

ESTIMACIÓN DE FLUJOS DE EVASIÓN VÍA ESCALAMIENTO DE MATRICES

Actualmente el sistema de transporte público Transantiago presenta un alto índice de evasión que se resume en que uno de cada tres chilenos no paga su pasaje. Una parte esencial para estudiar la evasión es estimar correctamente los flujos de evasores que viajan a través de un par Origen-Destino. En esta memoria precisamente se enfoca en el problema de calcular los flujos de evasores basándose de la matriz de validaciones, el cual contiene datos de los flujos de pasajeros no evasores del Transantiago. El enfoque utilizado consiste en resolver un problema de escalamiento de matrices, el cual consiste básicamente en expandir la matriz de validaciones con el fin de que las sumas por filas y columnas de la matriz resultante coincidan con números previamente calculados O_i y D_j, que representan la cantidad de viajes totales por origen y destino. Como la evasión es un fenómeno que varía según la hora del día, es necesario también incorporar el efecto horario en la forma de calcular de los O_i y D_j, analizando los casos para el día completo, para la mañana y la tarde. Hay varios métodos que permiten encontrar una solución al problema del escalamiento de matrices: algoritmo de flujo, optimización convexa y algoritmo de Furness, todos detallados en extenso en este trabajo. Finalmente, se presenta una aplicación relacionada con los flujos de evasión, el cual consiste en identificar los mejores paraderos del Transantiago donde ubicar fiscalizadores para así combatir la evasión.

A los tres hermanos que vivieron para siempre

Agradecimientos

Primero que todo, agradezco a mi familia por darme la oportunidad de acceder a la educación superior y por apoyarme de forma incondicional durante todo este tiempo que estuve estudiando. A mi madre quien me motivó siempre a dar lo mejor de mí. A mi padre, de quien siempre estaré orgulloso.

Agradezco también a numerosos amigos que me han apoyado durante este proceso. A Ignacio Toro, quien lo conozco desde séptimo básico. A Abner, con quien compartimos infinitos almuerzos. A Christian Flores, con quien compartí los primeros años en la universidad y le deseo lo mejor ahora que está en Miami. A Gumi, quien lo conozco del colegio, un gran compañero a pesar de no estudiar en Unviersidad de Chile. Al grupo de amigos de la chacra que pasamos juntos miles de tardes jugando ajedrez. A los amigos del DIM con quienes compartí muy buenos momentos.

Debo agradecer también a Carlos Bonet y Bastián Bahamondes, compañeros de oficina, quienes me han apoyado siempre durante el Trabajo de Título. A mi profesor guía, José Correa por aceptar ser mi tutor, quien siempre tuvo una gran disposición para guiarme en este camino. A Andy Wiese, quien ayudó mucho con la lectura y discusión de los papers.

Tabla de Contenido

Introducción

Según los datos del departamento de Fiscalización del Ministerio de Transporte, durante el último trimestre del 2016 el índice de evasión del Transantiago alcanzó un nivel histórico de 34,6 % [6]. El problema de la evasión ha ocasionado un enorme déficit al sistema de transporte de buses que ha tenido que ser subsidiado por el gobierno con cerca de 700 millones de dólares al año [14]. Una parte esencial para estudiar la evasión es estimar correctamente los flujos de evasores que viajan a través de un par Origen-Destino. Concretamente, el problema consiste en dado una matriz de validaciones donde sus filas y columnas representan un origen y un destino y cada entrada representa la cantidad de viajes de pasajeros no evasores, lo que se quiere es expandir la matriz para así conocer la cantidad de pasajeros totales incluyendo la cantidad de evasores.

En esta memoria, el enfoque que se basa para expandir la matriz de validaciones es resolver un problema de *escalamiento de matrices*, el cual consiste básicamente en dado una matriz A no negativa y vectores O y D positivos, encontrar las matrices diagonales X e Y, tal que la suma por cada fila i de la matriz XAY coincida justamente con O_i y la suma de cada columna j con D_j. El problema de escalamiento de matrices es un problema importante que ha sido utilizado y estudiado por diversas comunidades científicas, empezando con el trabajo de Kruithof en el año 1937 sobre el tráfico de las líneas telefónicas [21], Deming y Stephan en transporte y predicción de viajes [11], Brown en teoría de probabilidades [5], Stone en economía [31], Wilkinson en análisis numérico [24], Friedlander [16] y Sinkhorn en estadística [30]. Ha sido una herramienta importante que ha sido aplicado en numerosas áreas tales como reconstrucción de imágenes [17], gestión de operaciones [26], gestión y control [3], teoría de la computación [22], machine learning [9] etc. Dado un sistema lineal $Ax = B$, se puede obtener una solución eficiente calculando $Y(XAY)^{-1}XB$, ya que XAY es más estable numéricamente que calculando directamente A^{-1}[32]. El problema de escalamiento de matrices también sirve para calcular matchings de tamaño máximo en grafos bipartitos [29]. En estadística, a menudo se ha utilizado para el ajuste proporcional iterativo [33]. Numerosos trabajos se han hecho para caracterizar el problema de escalamiento de matrices, resaltando el trabajo de Evans en 1970 [15], quien caracterizó el problema sobre las matrices positivas, y años más tarde en el 1989, G. Rothblum y Schneider dieron y demostraron la caracterización general [28]. Para mayor información, se puede revisar el artículo sobre escalamiento de matrices de Idel [18].

A lo largo de los años, se han desarrollado numerosos algoritmos para resolver el problema de escalamiento de matrices. Quizás el algoritmo más conocido y sencillo es el algoritmo de Furness [15], que data del 1960, el cual consiste en ajustar iterativamente las filas y las columnas hasta llegar a la solución. La convergencia de este método se probado para matrices

positivas y se conoce desde el año 2008 que su orden de complejidad es $\widetilde{O}(mh^2/\varepsilon^2)$ donde $h = ||O||_1$ y ε es el error [20]. Un algoritmo basado en el método de la elipsoide fue descubierto por Kalantari y Khachiyan, quienes obtuvieron un algoritmo $\widetilde{O}(n^4)$[19]. Balakrishnan obtuvo un algoritmo con complejidad $\widetilde{O}(n^6)$ usando métodos de punto interior [3]. Rote y Zachariasen trasformaron el problema de escalamiento de matrices a un problema de flujo y obtuvieron un algoritmo con complejidad $O(n^4(\log(n) + \log(\frac{h}{\varepsilon})))$ [27]. Linal y Samorodnitsky fueron los primeros en proponer un algoritmo fuertemente polinomial con complejidad $\widetilde{O}(n^7)$ [22]. Más recientemente, Allen-Zhu *et al.* publicaron un algoritmo basado en la optimización convexa, el cual mejora el algoritmo de Furness obteniendo $\widetilde{O}(mn^{2/3}h^{1/3}/\varepsilon^{2/3})$ [2]. Cohen *et al.* utilizando el método de newton y métodos de punto interior, obtuvieron algoritmos de complejidad $\widetilde{O}(m\log(\delta)\log^2(1/\varepsilon))$ [7], donde δ es la división entre el elemento más grande de la matriz versus el menor y m es la cantidad de elementos no ceros de la matriz.

El trabajo de esta memoria consiste en explicar en extenso cómo aplicar la teoría del escalamiento del matrices al problema de la evasión del Transantiago y también mostrar los resultados correspondientes utilizando los datos que se tienen a disposición. En este trabajo, los números O_i y D_j del problema de escalamiento de matrices representan la cantidad de viajes totales por origen y destino. Para calcular la cantidad de viajes por origen O_i se requieren de los datos de evasión recogido por la fiscalización incógnita del Transantiago. Calcular la cantidad de viajes por destino D_j es más complicado ya que no existen datos, por cual se sugiere estimar basándose en los O_i. Como la evasión es un fenómeno que varía según la hora del día, es necesario también incorporar el efecto horario en la forma de calcular de los O_i y D_j, analizando los casos para el día completo, para la mañana y la tarde. Según la forma de calcular de los O_i y D_j, se pueden obtener interesantes propiedades en la matriz resultante. Para resolver el problema de escalamiento de matrices, se utilizaron tres algoritmos que hay en la literatura: algoritmo de flujo, optimización convexa y algoritmo de Furness, todos detallados en extenso en esta memoria. Se analiza también la eficiencia práctica de los tres algoritmos ya mencionados.

Finalmente, se muestra una aplicación de los flujos de evasión ya obtenidos. El problema consiste en identificar los mejores paraderos del Transantiago donde ubicar fiscalizadores para así combatir la evasión. El modelo consiste en un problema de *optimización bi-nivel* en un grafo que representa la red de desplazamientos posibles para el pasajero. El operador del sistema, en la primera etapa, establece probabilidades de fiscalización en cada arco de tal forma que los costos de los evasores sean lo máximo posible. En la segunda etapa, los evasores, quienes se asumen que son seres racionales, eligen las rutas menos costosas para ellos dadas las probabilidades de fiscalización. El flujo de evasores son representados mediante *commodities*, que consisten en un par origen-destino y la cantidad de evasores que viajan a través de ese par.

Capítulo 1

Datos relevantes disponibles

Es en este capítulo se explica en detalle los datos relevantes que se tienen a disposición para este trabajo. Para estimar los flujos de evasión, es de especial importancia los datos de evasión por fiscalización incógnita y la matriz Origen-Destino de validaciones.

1.1. Datos de evasión

Los datos de evasión que se consideraron en este trabajo principalmente son del año 2016. Actualmente, hay dos bases de datos de fiscalización diferentes, lo cuales se explican a continuación:

1.1.1. Fiscalización en Paraderos

En el Transantiago existen fiscalizadores instalados en ciertos paraderos, quienes se suben a los buses y verifican si los pasajeros que van en el viaje han pagado su pasaje mediante el uso de la validación de la tarjeta de BIP (la forma de pago del Transantiago). Los fiscalizadores multan a los pasajeros que no pagan el pasaje. Durante cada fiscalización se anota la cantidad de pasajeros que se controlaron, la cantidad de personas que no pagó y la cantidad de multados.

	COMUNA_PUNTO_CONTROL	PUNTO_CONTROL	FECHA_CONTROL	BUSES_CONTROLADOS	PASAJEROS_CONTROLADOS	PASAJEROS_EVADEN
20411	QUILICURA	(PARADERO) AV. MANUEL A. MATTA ENTRE CALLE 1 Y 4 OR	24-10-2016	18	220	15
20412	SAN JOAQUIN	(PARADERO) AV. DEPARTAMENTAL ENTRE MAIHUE Y CANA	24-10-2016	9	156	15
20413	SAN MIGUEL	(PARADERO) GRAN AVENIDA ENTRE 2DA AVENIDA Y 3RA A	24-10-2016	7	115	23
20414	SAN MIGUEL	(PARADERO) GRAN AVENIDA ENTRE MIGUEL LEÓN PRADO	24-10-2016	21	395	71
20415	SANTIAGO	(PARADERO) AVENIDA SANTA ROSA ENTRE AV. STA. ISABE	25-10-2016	6	175	18
20416	SANTIAGO	(PARADERO) ALAMEDA ENTRE LORD COCHRANE Y NATANI	25-10-2016	20	408	38
20417	SANTIAGO	(PARADERO) ALAMEDA ENTRE SAN FRANCISCO Y SANTA R	25-10-2016	14	188	22
20418	SANTIAGO	(PARADERO) ALAMEDA ENTRE NAMUR Y DR. RAMÓN CORV	25-10-2016	13	274	22
20419	SANTIAGO	(PARADERO) ALAMEDA ENTRE NAMUR Y JOSÉ V. LASTARRI	25-10-2016	16	473	90
20420	ESTACION CENTRAL	(PARADERO) AVENIDA LOS PAJARITOS ENTRE VISTA HERM	25-10-2016	17	405	22
20421	ESTACION CENTRAL	(PARADERO) AVENIDA LOS PAJARITOS ENTRE ALAMEDA Y	25-10-2016	15	451	46
20422	ESTACION CENTRAL	(PARADERO) AVENIDA 5 DE ABRIL ENTRE LAS REJAS SUR Y (	25-10-2016	12	181	12
20423	INDEPENDENCIA	(PARADERO) AV. INDEPENDENCIA ENTRE AV. EINSTEIN Y S	25-10-2016	18	153	15
20424	INDEPENDENCIA	(PARADERO) AV. INDEPENDENCIA ENTRE PINTO Y GRAL DE	25-10-2016	10	99	11
20425	INDEPENDENCIA	(PARADERO) AV. INDEPENDENCIA ENTRE SANTOS DUMON	25-10-2016	18	190	12
20426	INDEPENDENCIA	INDEPENDENCIA 1420	25-10-2016	9	74	9
20427	NUNOA	(PARADERO) AVENIDA IRARRÁZAVAL ENTRE EMILIO VAISS	25-10-2016	14	389	20
20428	NUNOA	(PARADERO) AVENIDA GRECIA ENTRE ALC.J.MONCKEBERG	25-10-2016	9	208	13
20429	PROVIDENCIA	(PARADERO) AVENIDA PROVIDENCIA ENTRE AV. BUSTAMA	25-10-2016	25	704	45

Figura 1.1: Fiscalización en paraderos

1.1.2. Fiscalización Incógnita

A diferencia de la fiscalización en paraderos, estos datos son obtenidos mediante fiscalizadores incógnitos quienes se suben a los buses y miden la cantidad de pasajeros que se suben por cada puerta del bus y registran la cantidad de gente que validadan su tarjeta BIP.

FECHA	SERVICIO	PATENTE	LUGAR INICIO	HORA INICIO	HORA	MINUTOS	INGRESAN	NO VALIDAN	PARADERO	ZONA777	COMUNA
12-01-2016	501	CJRZ81	NUEVA BILBAO - VITAL APO	7:00	7	0	18	3	PC1223	512	LAS CONDES
12-01-2016	501	CJRZ81	NUEVA BILBAO - VITAL APO	7:00	7	6	1	0	PC458	227	LAS CONDES
12-01-2016	501	CJRZ81	NUEVA BILBAO - VITAL APO	7:00	7	9	6	3	PC128	227	LAS CONDES
12-01-2016	501	CJRZ81	NUEVA BILBAO - VITAL APO	7:00	7	11	1	0	PC484	756	LAS CONDES
12-01-2016	501	CJRZ81	NUEVA BILBAO - VITAL APO	7:00	7	14	1	0	PC485	755	LAS CONDES
12-01-2016	501	CJRZ81	NUEVA BILBAO - VITAL APO	7:00	7	17	0	0	PC488	186	PROVIDENCIA
12-01-2016	501	CJRZ81	NUEVA BILBAO - VITAL APO	7:00	7	18	3	0	PC488	186	PROVIDENCIA
12-01-2016	501	CJRZ81	NUEVA BILBAO - VITAL APO	7:00	7	21	0	0	PC491	186	PROVIDENCIA
12-01-2016	501	CJRZ81	NUEVA BILBAO - VITAL APO	7:00	7	23	4	0	PC492	186	PROVIDENCIA
12-01-2016	501	CJRZ81	NUEVA BILBAO - VITAL APO	7:00	7	25	1	0	PC494	184	PROVIDENCIA
12-01-2016	501	CJRZ81	NUEVA BILBAO - VITAL APO	7:00	7	28	3	0	PC495	183	PROVIDENCIA
12-01-2016	501	CJRZ81	NUEVA BILBAO - VITAL APO	7:00	7	34	0	0	PC501	182	PROVIDENCIA

Figura 1.2: Fiscalización incógnita

Cabe mencionar que un problema de estos datos es que los fiscalizadores incógnitos solo registran el lugar de inicio del recorrido pero no anotan el paradero donde se produce cada medición. Más adelante, se explicará cómo poder solucionar éste problema.

1.2. Matrices origen-destino

Las matrices de origen-destino (OD) de validaciones son datos obtenidos a través de las bases de datos de la tarjeta BIP y de los GPS de los buses. Las matrices OD muestran el origen y el destino del viaje y un número representativo de la cantidad de validaciones BIP registradas. Este número se calcula como el promedio de validaciones BIP que se registraron durante 5 días laborales en el trayecto. Las matrices se clasifican en dos niveles de agregación:

zonas 777 y comunas. Las zonas 777 son delimitaciones geográficas de la ciudad de Santiago y constan de aproximadamente 800 zonas. En total son 34 comunas que cubre Transantiago. Además, la matriz divide su información en 48 intervalos de 30 minutos cada uno. Las matrices de origen destino son públicas y se pueden descargar de la página del DPTM (Directorio de Transporte Público Metropolitano) [12]. Las matricies están hechas con la tecnología ADATRAP, desarrollado por la Universidad de Chile [1].

A	B	C	D	E	F	G	H	I	J	K	L
diseno777su	diseno777ba	mediahora	viaje_labora	viajes_adult	viajes_estud	viajes_1_eta	viajes_2_eta	viajes_3_eta	viajes_4_eta	viajes_5oma	viajes_usan_
0	0	6:00:00	0,568912	0,568912	0	0,568912	0	0	0	0	0
0	0	6:30:00	1,457268	1,457268	0	1,457268	0	0	0	0	0
0	0	7:00:00	1,943024	1,457268	0,485756	1,943024	0	0	0	0	0
0	0	7:30:00	0,728634	0,728634	0	0,728634	0	0	0	0	0
0	0	8:00:00	1,457268	1,21439	0,242878	1,457268	0	0	0	0	0
0	0	8:30:00	0,234192	0,234192	0	0,234192	0	0	0	0	0
0	0	9:00:00	0,702576	0,234192	0,468384	0,702576	0	0	0	0	0
0	0	9:30:00	1,743994	1,743994	0	1,494852	0,249142	0	0	0	0
0	0	10:30:00	0,996568	0,747426	0,249142	0,996568	0	0	0	0	0
0	0	11:00:00	1,24571	1,24571	0	1,24571	0	0	0	0	0
0	0	11:30:00	0,996568	0,996568	0	0,747426	0	0,249142	0	0	0,249142
0	0	12:00:00	1,24571	0,996568	0,249142	0,996568	0	0,249142	0	0	0,249142
0	0	12:30:00	0,540812	0,540812	0	0,540812	0	0	0	0	0
0	0	13:00:00	1,35203	0,811218	0,540812	1,35203	0	0	0	0	0
0	0	13:30:00	0,811218	0,270406	0,540812	0,811218	0	0	0	0	0
0	0	14:00:00	2,7536	2,47824	0,27536	2,7536	0	0	0	0	0
0	0	14:30:00	1,10144	0,55072	0,55072	1,10144	0	0	0	0	0
0	0	15:00:00	1,65216	1,10144	0,55072	1,65216	0	0	0	0	0

Figura 1.3: Matriz de viajes del año 2016 por zonas

Las primeras dos columnas indican la zona origen y la zona destino. Las otras columnas indican la cantidad de viajes que han habido en la categoría especificada.

1.3. Consolidado de recorridos y paraderos

El consolidado de recorridos contiene información sobre todos los recorridos del Transantiago. Además existe información sobre las frecuencias, velocidades y distancias de los recorridos.

El consolidado de paraderos contiene información sobre todos los paraderos del Transantiago. Existen aproximadamente 11 mil paraderos de buses en el Transantiago y todos se indentifican mediante un código usuario y un código del Transantiago.

Los consolidados son datos públicos y también se pueden descargar en la página del DPTM [13].

Capítulo 2

Escalamiento de matrices

En esta memoria, la herramienta fundamental para calcular los flujos de evasores es el escalamiento de matrices. En este capítulo se desarrolla la teoría del escalamiento de matrices y se muestran tres algoritmos que existen en la literatura para su resolución: el algoritmo de Furness, el algoritmo de flujo y un algoritmo basado en la optimización convexa. El algoritmo de Furness fue el primer intento para resolver el problema de escalamiento de matrices y data del año 1960. El algoritmo de flujo y el algoritmo de optimización convexa son algoritmos más recientes y ocupan técnicas como el clásico problema de flujo de costo mínimo y la optimización de una función convexa, respectivamente. En el capítulo 3 veremos como aplicar esta teoría al problema de la evasión.

2.1. Definición

Dado una matriz A de dimensiones $n \times n$ [1] no negativa, el problema de **escalamiento de matrices** consiste en *escalar* la matriz A de tal forma que la suma por filas y por columnas dé los números deseados. Más formalmente, se puede definir como:

Definición 2.1 *Dado una matriz A no negativa de dimensiones $n \times n$ y vectores O, $D \in \mathbb{R}^n$ estrictamente positivos, se dice que la matriz A es (O,D)-escalable si existen matrices diagonales no negativos X,$Y \in \mathbb{R}^{n \times n}$ de tal forma que $B = XAY$ es una (O,D)- matriz, ésto es $B\mathbb{1} = O$ y $\mathbb{1}^T B = D$.*

Donde $\mathbb{1}$ es el vector de solo unos. Notar que usando la definición anterior, para que exista el escalamiento es necesario que se cumpla $\sum_{i=1}^{n} O_i = \sum_{j=1}^{n} D_j \stackrel{\text{def}}{=} h$.

Notar que la multiplicación XAY da una matriz resultante $\alpha_i \beta_j A_{ij}$ (por ser X e Y matrices diagonales) donde los α_i son valores de la diagonal de X y los β_j son los valores de la diagonal de Y. Por lo tanto el problema de escalamiento de matrices se reduce en encontrar los escalares α y β tal que

[1] La misma teoría funciona para matrices no cuadradas.

$$\sum_{j=1}^{n} A_{ij}\alpha_i\beta_j = O_i \quad \sum_{i=1}^{n} A_{ij}\alpha_i\beta_j = D_j$$

La matriz resultante $B = XAY$ se dice que es el (O, D)-escalamiento o simplemente, escalamiento de A.

Observación Algunas propiedades que se pueden deducir de las definiciones

- Las matriz B es no negativa puesto que los factores de balanceo son positivos
- Si $A_{ij} = 0$ entonces $B_{ij} = 0$
- Se puede dar el caso de $B_{ij} - A_{ij} < 0$

Definición 2.2 *Dado una matriz A no negativa de dimensiones $n \times n$, vectores O, $D \in \mathbb{R}^n$ estrictamente positivos y $\varepsilon > 0$, se dice que la matriz A es asintóticamente (O,D)-escalable (o simplemente ε-escalable) si existen matrices diagonales no negativos X,$Y \in \mathbb{R}^{n\times n}$ de tal forma que $B = XAY$ cumple con $\|B\mathbb{1} - O\| < \varepsilon$ y $\|\mathbb{1}^T B - D\| < \varepsilon$.*

Si bien el concepto exacto del escalamiento de matrices es interesante, la versión asintótica es incluso más ya que se ajusta mejor con el hecho de resolver el problema computacionalmente ya que muchas vez no se quiere encontrar la solución exacta sino que basta con un error ε.

Ejemplo La matriz $A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ con los vectores $O = D = (1, 1)$ no es escalable ya que escribiendo la matriz $\begin{pmatrix} \alpha_1\beta_1 & \alpha_1\beta_2 \\ 0 & \alpha_2\beta_2 \end{pmatrix}$ e imponiendo en ella las restricciones, se llega a que necesariamente uno de los factores de escalamiento debe ser 0 lo cual es una contradicción. Sin embargo, la matriz A sí es asintóticamente escalable ya que basta tomar $\begin{pmatrix} 1 & \varepsilon \\ 0 & 1 \end{pmatrix}$ el cual cumple las restricciones con un error ε y con los factores de escalamiento $\alpha_1 = \beta_1 = 1$, $\alpha_2 = 1/\varepsilon$ y $\beta_2 = \varepsilon$.

También los factores de escalamiento pueden ser irracionales siendo que las matrices y vectores tengan solo números racionales: Sea $A = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$ con los vectores $O = D = (1, 1)$, se puede comprobar que:

$$\begin{pmatrix} (\sqrt{2}+1)^{-1} & 0 \\ 0 & (\sqrt{2}+2)^{-1} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} \sqrt{2} & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 2-\sqrt{2} & \sqrt{2}-1 \\ \sqrt{2}-1 & 2-\sqrt{2} \end{pmatrix}$$

2.2. Resultados conocidos

Como se ha visto anteriormente, el problema de escalamiento de matrices no siempre tiene solución. A. W. Evans (1970) estudió el problema de escalamiento para matrices que no tienen ceros [15]. Es el primer gran resultado sobre esta materia.

Teorema 2.3 *Si la matriz A y los vectores O y D contienen solo números positivos entonces el problema de escalamiento de matrices tiene solución y ésta es única.*

Entonces la pregunta que faltaba por responder es qué pasaba en el caso de matrices con ceros. No fue hasta el año 1989, que G. Rothblum y Schneider dieron y demostraron la caracterización general [28].

Teorema 2.4 *Las siguientes proposiciones son equivalentes:*

- *La matriz A es (O,D)-escalable.*
- *Existe una matriz B tal que supp(B)=supp(A) y cumple las restricciones de las sumas por columnas y filas.*
- *Para cada $I \times J \subseteq [n] \times [n]$ para lo cual se cumple $A_{I^cJ} = 0$ se tiene que $\sum_{i\in I} O_i \geq \sum_{j\in J} D_j$ y la igualdad se cumple si y solo si $A_{IJ^c} = 0$.*

Donde el $supp(A)$ son los elementos no ceros de la matriz A.

También los mismo autores demostraron una versión para las matrices asintóticamente escalables.

Teorema 2.5 *Las siguientes proposiciones son equivalentes:*

- *La matriz A es asintóticamente(O,D)-escalable.*
- *Existe una matriz B tal que $supp(B) \subseteq supp(A)$ y cumple las restricciones de las sumas por columnas y filas.*
- *Para cada $I \times J \subseteq [n] \times [n]$ para lo cual se cumple $A_{I^cJ} = 0$ se tiene que $\sum_{i\in I} O_i \geq \sum_{j\in J} D_j$.*

Además se logró probar la unicidad de la solución para todas las matrices escalables.

Teorema 2.6 *Si la matriz A es (O,D)- escalable, entonces su escalamiento B=XAY es único.*

Cabe destacar que la matriz B es única pero no las matrices X, Y ya que basta multiplicar la matriz X por una constante y dividir la matriz Y por la misma constante para generar la misma matriz B.

Ejemplo El mismo ejemplo anterior, la matriz $A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ con los vectores $O = D = (1, 1)$ es ε-escalable según este torema ya que tomando $I = \{1\}$ y $J = \{1\}$ se puede verificar que $A_{I^cJ} = A_{2,1} = 0$ y que se cumple $\sum_{i\in I} O_i = \sum_{j\in J} D_j = 1$. Sin embargo no es escalable ya

que se cumple la igualdad pero $A_{IJ^c} = 1 \neq 0$.

El Teorema 2.5 da una forma fácil de verificar si una matriz A es asintóticamente escalable. Basta verificar si el siguiente sistema

$$\sum_{j=1}^{n} x_{ij} = O_i \quad \sum_{i=1}^{n} x_{ij} = D_j,$$

tiene solución sujeto a que: $x_{ij} \geq 0$ si $A_{ij} > 0$ y $x_{ij} = 0$ si $A_{ij} = 0$. Todas las restricciones son lineales por lo tanto se puede usar programación lineal para verificar si hay solución.

2.3. Algoritmo de Furness

El algoritmo de Furness (o también conocido como RAS) es un algoritmo muy simple del año 1960 y es el primer intento de encontrar una solución al problema de escalamiento de matrices [10].

ALGORITMO 1 FURNESS

Input: Una matriz $A \in \mathbb{R}^{n \times n}$ no negativa; vectores O, $D \in \mathbb{R}^n$ estrictamente positivos; K número de iteraciones;

1: $\alpha_i \leftarrow 1$ para todo $i = 1, 2, \ldots, n$
2: **for** $k = 0$ **TO** K **do**
3: $\quad \beta_j \leftarrow D_j / \sum_{i=1}^{n} A_{ij}\alpha_i$ para todo $j = 1, 2, \ldots, n$
4: $\quad \alpha_i \leftarrow O_i / \sum_{j=1}^{n} A_{ij}\beta_j$ para todo $i = 1, 2, \ldots, n$
5: **end for**
6: $\hat{A_{ij}} \leftarrow A_{ij}\alpha_i\beta_j$ para todo $i, j = 1, 2, \ldots, n$
7: **return** $\hat{A}$

El algoritmo de Furness consiste básicamente en cada iteración ajustar los α_i para que la matriz $A_{ij}\alpha_i\beta_j$ cumpla las sumas por filas y luego ajustar β_j para que $A_{ij}\alpha_i\beta_j$ cumpla las sumas por columnas y así sucesivamente.

No es claro a priori que haciendo $K \to \infty$ el método de Furness converge a una solución válida. Sin embargo, El mismo Evans (1970) logró probar que para las matrices positivas, el método converge a la solución (siempre existe por teorema 3.3) [15].

Teorema 2.7 *Si la matriz A es positiva, entonces el método de Furness converge y su límite es el escalamiento de A.*

Ejemplo Si se toma la matriz $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ con los vectores $O = D = (1, 1)$, se puede ver que claramente no es escalable y tampoco ε-escalable. El algoritmo de Furness simplemente

diverge ya que si se ajusta las filas resulta la matriz $\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$ y luego si se ajusta las columnas da $\begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$ y así sucesivamente sin llegar a converger.

Cada iteración del algoritmo de Furness tiene complejidad $O(m)$, puesto que se trabaja solo con los números no ceros de la matriz. En el año 2008, B. Kalantari, I. Lari, F. Ricca, y B. Simeone logró probar el siguiente teorema sobre el cálculo de la complejidad del Algoritmo de Furness [20].

Teorema 2.8 *Si la matriz A es asintóticamente escalable, entonces el algoritmo de Furness produce un ε-escalamiento de la matriz en $\widetilde{O}(mh^2/\varepsilon^2)$* [2] *.*

Notar que este orden de complejidad no es polinomial ya que depende del factor h. La entrada tiene tamaño $m\log(\max(A_{ij}))$.

2.3.1. Algoritmos alternativos

Existen numerosas versiones de algoritmos que difieren del algoritmo de Furness en su forma de escalar de las matrices. A continuación se nombra dos de los más conocidos: **Factor promedio** y método de **Detroit**.

El algoritmo factor promedio consiste en cada instancia promediar la suma de las diferencias entre la suma por filas y las sumas por columnas.

$$\hat{A}_{ij}^{(1)} = \frac{1}{2}\left(\frac{O_i}{\sum_{j=1}^{n} A_{ij}} + \frac{D_j}{\sum_{i=1}^{n} A_{ij}}\right) A_{ij}$$

$$\hat{A}_{ij}^{(n+1)} = \frac{1}{2}\left(\frac{O_i}{\sum_{j=1}^{n} \hat{A}_{ij}^{(n)}} + \frac{D_j}{\sum_{i=1}^{n} \hat{A}_{ij}^{(n)}}\right) \hat{A}_{ij}^{(n)} \text{ para } n \geq 1.$$

Evans (1970) logró probar el siguiente teorema sobre el factor promedio [15]

Teorema 2.9 *Si la matriz A es (O,D) escalable, el algoritmo del factor promedio no siempre converge al escalamiento de la matriz A.*

Ésto hace que el factor del promedio no sea un buen algoritmo para encontrar la solución al escalamiento de matrices.

El método de Detroit tiene una forma un tanto más complicada

[2] $\widetilde{O}$ ignora los factores logarítmicos en n y en $1/\varepsilon$.

$$\hat{A}_{ij}^{(1)} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}}{h}\left(\frac{O_i}{\sum_{j=1}^{n} A_{ij}}\frac{D_j}{\sum_{i=1}^{n} A_{ij}}\right)A_{ij}$$

$$\hat{A}_{ij}^{(n+1)} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} \hat{A}_{ij}^{(n)}}{h}\left(\frac{O_i}{\sum_{j=1}^{n} \hat{A}_{ij}^{(n)}}\frac{D_j}{\sum_{i=1}^{n} \hat{A}_{ij}^{(n)}}\right)\hat{A}_{ij}^{(n)} \text{ para } n \geq 1,$$

Donde $h = \sum_{i=1}^{n} O_i = \sum_{j=1}^{n} D_j$.

La convergencia del algoritmo de Detroit no es claro hasta hoy en día pero hay casos donde el algoritmo de Detroit es más ineficiente que el algoritmo de Furness. Basta tomar una matriz A donde ya estén ajustada las sumas por filas pero las sumas por columnas no, entonces se puede ver que el método de Detroit produce la misma secuencia de matrices que el método de Furness pero en cada iteración realiza más cálculos.

2.4. Algoritmo de flujo

El problema de escalamiento de matrices entero se puede formular como encontrar los α_i y los β_j tal que

$$\sum_{j=1}^{n}[A_{ij}\alpha_i\beta_j] = O_i \quad \sum_{i=1}^{n}[A_{ij}\alpha_i\beta_j] = D_j \ ,$$

donde $[\cdot]$ es la función redondeo hacia abajo. Es necesario que los vectores O y D tengan solo números enteros.

Gunter Rote y Martin Zachariasen lograron demostrar que el problema del escalamiento de matrices entero es equivalente a resolver el siguiente problema [27]

$$\begin{aligned}
\min_{x} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=1}^{x_{ij}} \log\frac{t}{A_{ij}} \\
\text{sujeto a} \quad & \sum_{j=1}^{n} x_{ij} = O_i, \text{ para } i = 1,\ldots,n. \\
& \sum_{i=1}^{n} x_{ij} = D_j, \text{ para } j = 1,\ldots,n. \\
& x_{ij} \geq 0, \ x_{ij} \text{ entero} \\
& x_{ij} = 0, \text{ si } A_{ij} = 0
\end{aligned} \tag{2.1}$$

El problema anterior es un problema de flujo sobre un grafo bipartito $G(A)$ donde hay n nodos de origen R_i y n nodos de destino S_j tal que el valor x_{ij} es valor del flujo entre R_i y

S_j. Si $x_{ij} = 0$ significa que no hay arco entre R_i y S_j. La función de costos para un arco con $A_{ij} \neq 0$ está dada por

$$f_{ij} = \sum_{t=1}^{x_{ij}} \log \frac{t}{A_{ij}}$$

El cual es una función linear por pedazos y convexa.

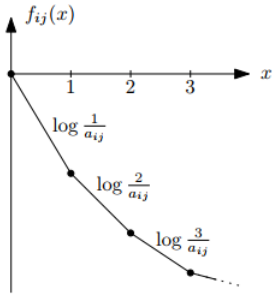


Figura 2.1: Función f_{ij}

Una función linear por pedazos convexa se puede modelar mediante una secuencia de funciones lineales. Basta considerar arcos paralelos entre R_i y S_j de capacidad 1 y cada uno con costo $c_{ij}^t = \log \frac{t}{A_{ij}}$ para $t = 1, 2, \ldots . t_{\max}$. Donde un posible valor para el $t_{\max}$ es el valor h. Como el flujo de R_i a S_j es de algún valor integral x, entonces el flujo se distribuirá entre los x arcos menos costosos entre R_i a S_j y por lo tanto el costo final es justamente f_{ij}.

Por lo tanto, el problema anterior es un problema de flujo clásico y se puede resolver mediante programación lineal. En particular se sabe que existe una solución integral ya que los costos son números enteros.

$$\begin{aligned}
\min_x \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=1}^{t_{\max}} x_{ijt} \log \frac{t}{A_{ij}} \\
\text{sujeto a} \quad & \sum_{j=1}^{n}\sum_{t=1}^{t_{\max}} x_{ijt} = O_i, \quad \text{para } i = 1, \ldots, n. \\
& \sum_{i=1}^{n}\sum_{t=1}^{t_{\max}} x_{ijt} = D_j, \text{ para } j = 1, \ldots, n. \\
& x_{ijt} \leq 1 \\
& x_{ijt} \geq 0, \text{ si } A_{ij} > 0 \\
& x_{ijt} = 0, \text{ si } A_{ij} = 0
\end{aligned} \tag{2.2}$$

Ahora para resolver el problema original de escalamiento de matrices, solo basta multiplicar la matriz A y los O_i y D_j por una constante $F = \frac{n}{\varepsilon}$ y redondear hacia abajo para

obtener los nuevos $\hat{O}_i$ y $\hat{D}_j$. Luego, usar las matrices y vectores escalados para resolver el problema de flujo y obtener una solución al problema de escalamiento de matrices integral. Si a la matriz resultante se le multiplica cada entrada por $1/F = \frac{\varepsilon}{n}$ se obtiene una solución al problema original. Entonces se obtiene el siguiente resultado

Teorema 2.10 *Si la matriz A es ε-asintóticamente escalable, entonces usando el problema de flujo se puede revoler el problema de escalamiento de matrices con error ε.*

La complejidad total del algoritmo es la siguiente

Teorema 2.11 *Si la matriz A es asintóticamente escalable, entonces el algoritmo de flujo produce un ε-escalamiento de la matriz en $O(n^4(log(n) + log(\frac{h}{\varepsilon})))$.*

Lo cual quiere decir que el algoritmo de flujo es un algoritmo polinomial.

El pseudocódigo es el siguiente

ALGORITMO 2 Algoritmo de flujo

Input: Una matriz $A \in \mathbb{R}^{n\times n}$ no negativa; vectores O, $D \in \mathbb{R}^n$ estrictamente positivos; ε el error;

1: $\hat{A} \leftarrow [\frac{n}{\varepsilon}A]$
2: $\hat{O} \leftarrow [\frac{n}{\varepsilon}O]$
3: $\hat{D} \leftarrow [\frac{n}{\varepsilon}D]$
4: $t_{\max} \leftarrow$ el número más grande de los vectores O y D
5: Resolver el problema de flujo 2.2 con $\hat{A}, \hat{O}, \hat{D}$ y las variables x_{ijt}
6: $X_{ij} \leftarrow \sum_t x_{ijt}$ para todo $i, j = 1, 2, \ldots, n$
7: $A' \leftarrow \frac{\varepsilon}{n}X$
8: **return** A'

2.5. Optimización convexa

El problema de escalamiento de matrices también se puede plantear como un problema de optimización de la siguiente función convexa

$$f(x) = \sum_i^n r_i \log(\sum_{j=1}^n A_{ij} e^{x_j}) - c^T x. \tag{2.3}$$

Proposición 2.12 *La función f(x) es convexa y cumple con las siguiente propiedades*

- $\nabla_j f(x) = \sum_{i=1}^n \frac{O_i A_{ij}}{<A_i, e^x>} e^{x_j} - D^T x$
- *Si* $||\nabla f(x)||^2_{D^{-1}} < \varepsilon$ *entonces* $(\frac{O_i A_{ij}}{<A_i, e^x>} e^{x_j})_{ij}$ *es un ε-(O,D) escalamiento de la matriz A.*
- *Si A es escalable entonces existe un x^* tal que minimiza $f(x)$ y $\nabla f(x) = 0$*

- *Si A es asintóticamente escalable entonces* $\text{ínf}_x\{f(x)\} > -\infty$
- *A no es asintóticamente escalable si y solo si* $\text{ínf}_x\{f(x)\} = -\infty$

Donde $||v||_w^2 = \sum_{i=1}^n w_i v_i^2$ es una norma si el vector w es positivo. Notar por la forma de la solución $(\frac{O_i A_{ij}}{<A_i, e^x>} e^{x_j})_{ij}$, el gradiente de $f(x)$ no es más que una medida de cuán cerca está la solución actual x de cumplir las sumas por columnas.

Allen-Zhu, Li, Oliveira y Wigderson (2017) [2] descubrieron varios algoritmos iterativos para minimizar $f(x)$ de forma eficiente, basado en métodos de primer orden y segundo orden. Las ideas básicas de estos algoritmos es encontrar *buenas* direcciones de descenso para así minimizar la norma del gradiente. En esta memoria, el algoritmo que se estudia de los autores se llama scaling0, el cual tiene la siguiente propiedad

Teorema 2.13 *Si la matriz A es asintóticamente escalable, se puede resolver el problema de escalamiento de matrices usando scaling0 con un error ε en $\widetilde{O}(mn^{2/3}h^{1/3}/\varepsilon^{2/3})$.*

Si se compara este resultado con el teorema 2.8, este algoritmo converge más rápido que el algoritmo de Furness.

En esta sección, sin perder generalidad, se puede asumir que la matriz A cumple la condición de la suma por filas pero no el de las columnas. Ésto es debido a que siempre se puede multiplicar la filas de la matriz original por una constante para que se ajusten a la condición. Además se puede asumir también que la máxima entrada de cada es el valor 1 ya que basta normalizar cada fila de la matriz por el valor máximo.

El algoritmo de scaling0 consta de varias sub-rutinas que las llamaremos $\text{Grad}^N(x)$, $\text{Grad}^\infty(x)$ y Linear Coupling. Se explicará en qué consiste la primera subrutina que se llama $\text{Grad}^N(x)$, el cual consiste en un especie de método del gradiente. Para ello es necesario dar algunas definiciones.

Definición 2.14 *Se definen las coordenadas* $\Lambda^s \overset{def}{=} \{j \in [n] : \nabla_j \in [-D_j, D_j]\}$ *y* $\Lambda^l \overset{def}{=} \{j \in [n] : \nabla_j > D_j\}$.

Definición 2.15 *Dado $\delta \in \mathbb{R}^n$ con $||\delta||_\infty \leq 1/2$ se define las funciones Q^+ y Q^-*

- $Q^+(x,\delta) \overset{def}{=} \sum_{j\in\Lambda^s}(-\nabla_j \cdot \delta - \frac{4}{3}D_j \cdot \delta^2) + \sum_{j\in\Lambda^l} -\frac{7}{3}\nabla_j \cdot \delta_j$.
- $Q^-(x,\delta) \overset{def}{=} \sum_{j\in\Lambda^s}(-\nabla_j \cdot \delta - \frac{4}{3}D_j \cdot \delta^2) + \sum_{j\in\Lambda^l} -\frac{1}{2}\nabla_j \cdot \delta_j$.

Finalmente, se define $\text{Grad}^N(x) = \arg\min_{y\in\{y_1,y_2\}} f(y)$ donde

$$y_1 = x + \arg\max_{\delta\in\Omega^+_{N,x}} \text{ y } \quad \Omega^+_{N,x} = \{\delta \geq 0 | ||x+\delta||_\infty \leq N \wedge ||\delta||_\infty \leq 1/2\}$$

$$y_2 = x + \arg\max_{\delta\in\Omega^-_{N,x}} \text{ y } \quad \Omega^-_{N,x} = \{\delta \leq 0 | ||x+\delta||_\infty \leq N \wedge ||\delta||_\infty \leq 1/2\}$$

La subrutina $\text{Grad}^N(x)$ consiste básicamente en maximizar las funciones Q^+ y Q^- en los conjuntos $\Omega^+_{N,x}$ y $\Omega^-_{N,x}$, lo cual relativamente sencillo ya que se trata de funciones cuadráticas. Notar que el problema de maximizar se puede trabajar separadamente según cada coordenada $j \in [n]$. El siguiente algoritmo entrega el resultado $\text{Grad}^N(x)$

ALGORITMO 3 Grad^N

Input: Una matriz $A \in \mathbb{R}^{n \times n}$ no negativa; $N \geq 1$ diámetro; $x \in \mathbb{R}^n$ el vector de entrada con $||x||_\infty \leq N$

for $j = 0$ **TO** n **do**
  if $-D_j \leq \nabla_j f(x) \leq D_j$ **then**
    $z \leftarrow \frac{3\nabla_j f(x)}{8D_j}$
    if $0 \leq z \leq \min\{0{,}5, N - x_j\}$ **then**
      $\delta_j \leftarrow z$
    else
      $v \leftarrow \min\{0{,}5, N - x_j\}$
      $a_1 \leftarrow \nabla_j f(x)v - (\frac{8}{3}D_j v)$
      $a_2 \leftarrow 0$
      if $a_1 \geq a_2$ **then**
        $\delta_j \leftarrow v$
      else
        $\delta_j \leftarrow 0$
      end if
    end if
    $\delta_j \leftarrow 0$
  end if
end for
for $j = 0$ **TO** n **do**
  if $-D_j \leq \nabla_j f(x) \leq D_j$ **then**
    $z \leftarrow \frac{3\nabla_j f(x)}{8D_j}$
    if $0 \leq z \leq \min\{-0{,}5, -N - x_j\}$ **then**
      $\delta'_j \leftarrow z$
    else
      $v \leftarrow \min\{-0{,}5, -N - x_j\}$
      $a_1 \leftarrow \nabla_j f(x)v - (\frac{8}{3}D_j v)$
      $a_2 \leftarrow 0$
      if $a_1 \geq a_2$ **then**
        $\delta'_j \leftarrow v$
      else
        $\delta'_j \leftarrow 0$
      end if
    end if
    $\delta'_j \leftarrow \min\{-0{,}5, -N - x_j\}$
  end if
end for
$y_1 \leftarrow x + \delta$
$y_2 \leftarrow x + \delta'$
if $f(y_1) \geq f(y_2)$ **then**
  return y_2
else
  return y_1
end if

ALGORITMO 4 Grad$^{\infty}$

Input: Una matriz $A \in \mathbb{R}^{n \times n}$ no negativa; $x \in \mathbb{R}^n$ el vector de entrada

for $j = 0$ **TO** n **do**
  if $-D_j \leq \nabla_j f(x) \leq D_j$ **then**
    $z \leftarrow \frac{3\nabla_j f(x)}{8D_j}$
    if $0 \leq z \leq 0{,}5$ **then**
      $\delta_j \leftarrow z$
    else
      $v \leftarrow 0{,}5$
      $a_1 \leftarrow \nabla_j f(x)v - (\frac{8}{3}D_j v)$
      $a_2 \leftarrow 0$
      if $a_1 \geq a_2$ **then**
        $\delta_j \leftarrow v$
      else
        $\delta_j \leftarrow 0$
      end if
    end if
    $\delta_j \leftarrow 0$
  end if
end for
for $j = 0$ **TO** n **do**
  if $-D_j \leq \nabla_j f(x) \leq D_j$ **then**
    $z \leftarrow \frac{3\nabla_j f(x)}{8D_j}$
    if $0 \leq z \leq -0{,}5$ **then**
      $\delta'_j \leftarrow z$
    else
      $v \leftarrow -0{,}5$
      $a_1 \leftarrow \nabla_j f(x)v - (\frac{8}{3}D_j v)$
      $a_2 \leftarrow 0$
      if $a_1 \geq a_2$ **then**
        $\delta'_j \leftarrow v$
      else
        $\delta'_j \leftarrow 0$
      end if
    end if
    $\delta'_j \leftarrow -0{,}5$
  end if
end for
$y_1 \leftarrow x + \delta$
$y_2 \leftarrow x + \delta'$
if $f(y_1) \geq f(y_2)$ **then**
  return y_2
else
  return y_1
end if

El algoritmo Grad$^\infty$ es muy parecido a Grad, solo que se ignora el término N.

La siguiente subrutina se llama Linear Coupling y consiste en una combinación linear $x_{k+1} = t_k z_k + (1 - t_k)y_k$ para algún parámetro t_k y en la siguiente etapa realiza dos actualizaciones a z_k y y_k usando Grad^N y Mirr^N, donde Mirr^N se define como

Definición 2.16 *Dado una z con $||z||_\infty \leq N$ y un vector $v \in \mathbb{R}^n$, se define*

$$Mirr^N(z, v) = arg\ min_{||z'||_\infty \leq N}\{< v, z' > + \frac{1}{2}||z' - z||^2\}$$

ALGORITMO 5 LC

Input: Una matriz $A \in \mathbb{R}^{n\times n}$ no negativa; $N \geq 1$ diámetro; T número de iteraciones; $y_0 \in \mathbb{R}^n$ que satisface $||y_0||_\infty \leq 15N$

$z_0 \leftarrow 0$ y $t_0 \leftarrow \frac{1}{32}N$
for $k = 0$ **TO** $T - 1$ **do**
    $t_k \leftarrow$ la única raíz positiva de la ecuación $\frac{t_k^2}{t_{k-1}^2} + t_k + 1$
    $x_k \leftarrow t_k z_k + (1 - t_k)\, y_k$
    $y_k \leftarrow \mathbf{Grad}^{15n}(x_{k+1})$
    Sea $\nabla^s \in \mathbb{R}^n$ donde $\nabla^s \leftarrow \min\{\nabla_j f(x_{k+1}), 1\}$
    $z_{k+1} \leftarrow \mathbf{Mirr}^N(z_k, \alpha_k \nabla^s)$ donde $\alpha = \frac{3}{64T_k}$
end for
return y_T

El algoritmo final se llama Scaling0, el cual devuelve un valor z que minimiza el gradiente de la función f, el cual indica que es una solución al problema de escalamiento de matrices.

ALGORITMO 6 Scaling0

Input: Una matriz $A \in \mathbb{R}^{n\times n}$ no negativa; $N \geq 1$ diámetro; T número de iteraciones

$z_0 \leftarrow 0$
for $k = 0$ **TO** $\log N$ **do**
    $z_0 \leftarrow \mathbf{LC}(A, N, O(N), z_0)$
end for
$z_1 \leftarrow \mathbf{LC}(A, N, T, z_0)$
for $k = 1$ **TO** T **do**
    $z_{k+1} \leftarrow \mathbf{Grad}^\infty(z_k)$
end for
$z = \arg\min_{z\in\{z_1, z_2, \ldots, z_T\}} \{||\nabla f(z)||^2\}$
return (z)

Capítulo 3

Expansión por origen y destino

La matriz A a escalar es la matriz de validaciones vista ya en el capítulo de los datos 1.2. Como se ha visto en el capítulo anterior, es necesario contar con los vectores O y D, los cuales representan los viajes totales por origen y destino. En este capítulo se va a mostrar las fórmulas para calcular los vectores O y D y el significado que ellas tienen. Para este objetivo, es fundamental el uso de los datos de evasión por incógnito 1.1.2.

3.1. Georreferenciación

Uno de los defectos de los datos de fiscalización por incógnitos, es que los fiscalizadores no registran el paradero donde se realiza cada medición. Muchas veces ésto es debido a que el fiscalizador no dispone de suficiente tiempo para anotar esa información ya que también debe estar preocupado de la subida de los pasajeros al bus. Sin tener el paradero donde se realiza cada medición, no se puede a priori asignar ese dato de la evasión a alguna de las comunas o zonas de Santiago. Por lo tanto es necesario hacer ante que todo una **georreferenciación de los datos** de fiscalización.

Los fiscalizadores solo hacen una breve descripción del lugar donde inician el recorrido, ya sea nombrando las calles o las esquinas más cercanas al paradero. Para obtener el código del paradero inicial, fue necesario entrar a la página web de Transantiago y buscar la línea de bus señalado y buscar el paradero por donde pasa esa línea de bus que mejor calze con la descripción de las calles. Cabe destacar que una línea de bus por lo general hacen el recorrido de ida y de vuelta. Los paraderos que recorre el bus en el viaje de ida por lo general no coinciden con los del viaje de vuelta. Por lo tanto también es importante en esta etapa reconocer si el recorrido de que hizo el fiscalizador fue de ida o de vuelta. También fue necesario incluir la ayuda de Google Maps en el proceso, el cual permite visualizar mejor el entorno de las calles de Santiago. En total fue necesario hacer una búsqueda cercana a 600 paraderos, tomando en cuenta solo los datos de fiscalización del 2016. La mayoría de los paraderos georreferenciados fueron paraderos terminales, o sea, donde la línea de bus finaliza o inicia su recorrido. Esta primera etapa fue totalmente manual.

La segunda etapa es georreferenciar el resto de los paraderos. Para ello fue clave disponer de un archivo de perfiles el cual recauda una gran cantidad información sobre los recorridos de líneas de buses del Transantiago que se hicieron en un día laboral en particular (14 de marzo del 2016). Por ejemplo, aparece todas las horas que pasó el bus 506 en cada uno de los paraderos que les correspondía. A continuación se detalla a grandes rasgos los pasos que se hicieron para georreferenciar todos los paraderos restantes

- Por cada línea de bus que aparece en los datos de fiscalización, se busca esa misma línea en el archivo de los perfiles pero de tal forma que las horas inicio de ambos recorridos sean lo más cercana posible. Se busca así que hayan similitudes en el tiempo de inicio del recorrido.
- Analizar la diferencias de tiempo entre cada medición hecha por el fiscalizador en esa línea y compararlo con los tiempos que transcurre entre cada par de paraderos de la línea que tuvo la hora inicio más cercano del archivo de perfiles.
- Si la diferencias de tiempo que aparece entre cada par de mediciones en los datos de fiscalización es superior significa que el fiscalizador no tomó datos en algunos paraderos.
- Etiquetar con el paradero que mejor se ajusta con los tiempos.

Obviamente es posible que haya errores en la georreferenciación puesto que la datos de perfiles son de un día en particular y los datos de fiscalización son de muchos días. Sin embargo, equivocarse en un paradero no es muy significativo ya que como se verá después, el análisis posterior se hace sobre zonas o comunas, o sea, en un niveles más agregados.

3.2. Expansión día completo

Comúnmente se suele hablar de la evasión de cada zona o comuna, lo cual da la sensación de que la evasión es un fenómeno localizado, propio de cada zona o comuna. Sin embargo, la evasión debe entenderse más bien como un fenómeno **Origen-Destino**, es decir, la evasión cambia según el origen y el destino del viaje. Hacer una análisis de la evasión dado origen y destino agrega mayor complejidad al modelo pero a la vez permite entender mejor el panorama de la evasión de la ciudad. De ahora en adelante, se introduce el concepto de la evasión entre un par origen y destino (i, j).

Siguiendo en esta línea, surge de modo natural la pregunta siguiente **¿cómo calcular la evasión de los viajes que empiezan en la zona i y terminan en la zona j?** No existen datos que resuelvan este tema directamente. El objetivo de esta sección es plantear una metodología que permita calcular la evasión de un viaje que parte en un origen y que termine en un destino determinados.

Los supuestos con respecto a la evasión son

1. La evasión se explica principalmente por las zonas de origen y destino.
2. Gran parte de la evasión es evasión dura, es decir, evaden siempre.
3. La evasión en una zona determinada es similar entre las distintas líneas buses que pasan por ella.

También es necesario introducir algo de notación

- t_{ij} = validaciones bip hechas entre la zona i y la zona j
- e_i= índice de evasión origen de la zona i
- $\hat{t}_{ij}$ = viajes totales entre la zona i y la zona j
- $\hat{e}_i = \hat{t}_{ij} - t_{ij}$ = evasión posterior a la expansión de la zona i

Notar que los $\hat{t}_{ij}$ son desconocidos y es el valor que hay que estimar para los todos los pares de (i, j) existentes.

El modelo propuesto para encontrar los $\hat{t}_{ij}$ es precisamente usar el problema de **escalamiento de matrices**, detallado en el capítulo 2. La matriz a escalar es la matriz de validaciones $T = (t_{ij})$ donde las filas y columnas representan los origenes y destinos, y cada entrada t_{ij} representa la cantidad de validaciones del día completo. Los origenes y destinos pueden ser zonas y comunas pero por simplicidad se hablará más de zonas.

El vector O_i representa la cantidad de viajes con origen en la zona i y tiene la siguiente forma

$$O_i = \sum_{j=1}^{n} \frac{t_{ij}}{1 - e_i}$$

La fórmula anterior asume que los datos de evasión de fiscalización, son en realidad evasión de origen ya que mide la cantidad de evasores que se sube en cada paradero.

Siguiendo la misma lógica, el vector D_j debe representar la cantidad de viajes que tienen como destino en la zona j. Este valor no es intuitivo estimar ya que no existen datos sobre la cantidad de evasores que se bajan en cada zona. Ésto es debido a que simplemente que la gente al bajarse de los buses, no deja registro del lugar de bajada. La fórmula que se propone para el día complemento es la siguiente

$$D_j = \sum_{i=1}^{n} \delta \frac{t_{ij}}{1 - e_j} \tag{3.1}$$

El factor $\delta = \frac{\sum_i O_i}{\sum_j \sum_i \frac{t_{ij}}{1-e_j}}$. Es para así se cumpla la condición $\sum_{i=1}^{n} O_i = \sum_{j=1}^{n} D_j = h$.

Notar que las matriz T es no negativa y los vectores O y D son positivos según estas definiciones. Finalmente, la matriz $\hat{T} = \left(\hat{t}_{ij}\right)$ representa el escalamiento de la matriz T.

$$\hat{T}_{ij} = T_{ij}\alpha_i\beta_j$$

$$\sum_{j=1}^{n} \hat{T}_{ij} = O_i \quad \sum_{i=1}^{n} \hat{T}_{ij} = D_j$$

El escalamiento $\hat{T}$ con la fórmulas anteriores presentan algunas interesantes propiedades

Proposición 3.1 *La evasión de origen producto del escalamiento* $\hat{e}_i = \frac{\sum_{j=1}^{n}\left(\hat{t}_{ij}-t_{ij}\right)}{\sum_{j=1}^{n}\hat{t}_{ij}}$ *coincide con la evasión* e_i*. Además, se cumple que para la evasión de destino* $\hat{e}_j = \frac{\sum_{i=1}^{n}\left(\hat{t}_{ij}-t_{ij}\right)}{\sum_{i=1}^{n}\hat{t}_{ij}} = \frac{\delta-1+e_j}{\delta}$.

Demostración. Usando el hecho que $O_i = \sum_{j=1}^{n}\frac{t_{ij}}{1-e_i}$

$$\begin{aligned}
\hat{e}_i &= \frac{\sum_{j=1}^{n}\left(\hat{t}_{ij}-t_{ij}\right)}{\sum_{j=1}^{n}\hat{t}_{ij}} \\
&= 1 - \frac{\sum_{j=1}^{n} t_{ij}}{\sum_{j=1}^{n}\hat{t}_{ij}} \\
&= 1 - \frac{\sum_{j=1}^{n} t_{ij}}{O_i} \\
&= 1 - 1 + e_i \\
&= e_i
\end{aligned}$$

Usando el hecho que $D_j = \sum_{i=1}^{n}\delta\frac{t_{ij}}{1-e_j}$

$$\begin{aligned}
\hat{e}_j &= \frac{\sum_{i=1}^{n}\left(\hat{t}_{ij}-t_{ij}\right)}{\sum_{i=1}^{n}\hat{t}_{ij}} \\
&= 1 - \frac{\sum_{i=1}^{n} t_{ij}}{\sum_{j=1}^{n}\hat{t}_{ij}} \\
&= 1 - \frac{\sum_{i=1}^{n} t_{ij}}{D_j} \\
&= 1 - \frac{1-e_j}{\delta} \\
&= \frac{\delta-1+e_j}{\delta}
\end{aligned}$$

□

Esta propiedad indica que el escalamiento mantiene los índices de evasión por origen de cada zona.

3.3. Expansión mañana-tarde

En la sección anterior se trabajó y se obtuvo resultados con los datos de un día laboral completo. Sin embargo, también es interesante incorporar el factor horario en el análisis puesto que los flujos de evasión pueden variar según el tramo horario. En esta sección, se consideró trabajar con dos tramos horarios

- **Horario mañana**: 06:00- 10:00
- **Horario tarde**: 17:00- 21:00

Los tramos fueron elegidos de tal forma que tuvieran flujos parecidos de viajes. Cada uno abarca cerca de 1 millón de viajes (validaciones).

La notación básicamente es la misma sino que ahora se hace la distinción entre la mañana y la tarde.

- t_{ij}^d = validaciones bip hechas entre la zona i y la zona durante la mañana j
- t_{ij}^t = validaciones bip hechas entre la zona i y la zona durante la tarde j
- e_i^d= índice de evasión origen de la zona durante la mañana i
- e_i^t= índice de evasión origen de la zona durante la tarde i
- $\hat{t}_{ij}^d$ = viajes totales durante la mañana entre la zona i y la zona j
- $\hat{t}_{ij}^t$ = viajes totales durante la tarde entre la zona i y la zona j

Las fórmulas para los vectores O_i^d y O_i^t son similares a los ya ocupados en la expansión del día completo.

$$O_i^d = \sum_{j=1}^{n} \frac{t_{ij}^d}{1 - e_i^d}$$

$$O_i^t = \sum_{j=1}^{n} \frac{t_{ij}^t}{1 - e_i^t}$$

Para la cantidad de viajes que llegan a una comuna en la mañana y en la tarde D_j^d y D_j^t son difíciles de estimar puesto que no se tiene información sobre la cantidad de evasores que llegan a un destino dado. Además, al estar divididos por tramos horarios, no es válido usar las mismas fórmulas para el todo día. Una estimación posible es usar las siguientes fórmulas para los vectores D_j^d y D_j^t

$$D_j^d = \sum_{i=1}^{n} \delta^d \frac{t_{ij}^d}{1 - e_j^t}$$

$$D_j^t = \sum_{i=1}^{n} \delta^t \frac{t_{ij}^t}{1 - e_j^d}$$

Donde $\delta^d = \frac{\sum_i O_i^d}{\sum_i \sum_j \frac{t_{ij}^d}{1-e_j^t}}$ y $\delta^t = \frac{\sum_i O_i^t}{\sum_i \sum_j \frac{t_{ij}^t}{1-e_j^d}}$. Notar que se usan los índices de evasión de la tarde para estimar los flujos de los destinos de la mañana y viceversa. Ésto genera una especie de simetría en los índices de evasión, resumido en la siguiente proposición

Proposición 3.2 *Se tiene las siguientes igualdades*

- *Los índices de evasión por origen se mantienen* $\hat{e}_i^d = e_i^d$ *y* $\hat{e}_i^t = e_i^t$.
- *Se cumple que* $\hat{e}_j^d = \frac{\delta^d - 1 + e_j^t}{\delta^d}$ *y* $\hat{e}_j^t = \frac{\delta^t - 1 + e_j^d}{\delta^t}$.

Demostración. Es similar a la demostración 3.2 □

3.4. Fórmulas alternativas

Como se ha visto a lo largo de todo el capítulo, estimar la cantidad de viajes que llegan a un zona o comuna D_j no es trivial y depende mucho de los supuestos que uno haga. Durante el trabajo, también se consideraron algunas fórmulas alternativas para estimar el vector D_j.

El primero consiste en usar la evasión de la zona o comuna i en cada par OD

$$D_j = \sum_{i=1}^{n} \frac{t_{ij}}{1 - e_i}$$

Ésta fórmula no es muy promisoria ya que tiene el siguiente inconveniente

Proposición 3.3 *El problema de escalamiento de matrices tiene solución explícita y además pares OD con el mismo origen tienen el mismo índice de evasión.*

Demostración. Basta tomar $\alpha_i = \frac{1}{1-e_i}$ y $\beta_j = 1$ y ver que $\hat{T}_{ij} = T_{ij}\alpha_i\beta_j$ cumple las restricciones de sumas por filas y por columnas.

$$\sum_{j=1}^{n} \hat{T}_{ij} = \sum_{j=1}^{n} \frac{T_{ij}}{1 - e_i} = O_i$$

$$\sum_{i=1}^{n} \hat{T}_{ij} = \sum_{i=1}^{n} \frac{T_{ij}}{1 - e_i} = D_j$$

Como a cada término T_{ij} se le multiplica solo por el factor $\alpha_i = e_i$, entonces pares OD con el mismo origen tienen el mismo índice de evasión. □

Una fórmula distinta es promediar la evasión de origen con la evasión de destino.

$$D_j = \sum_{i=1}^{n} \frac{t_{ij}}{1 - \frac{1}{2}(e_i + e_j)} \tag{3.2}$$

En la expansión mañana tarde, se ocuparon índices de evasión de la tarde en la mañana y viceversa. Ocupando la fórmula anterior, se podría trabajar ambos tramos horarios de forma independiente, sin depender uno del otro. Sin embargo, el análisis del escalamiento producto de esta fórmula es mucho más complicado.

Siguiendo en la misma línea, una opción interesante es ponderar por los flujos en vez de usar el promedio

$$D_j = \sum_{i=1}^{n} \frac{t_{ij}}{1 - \left(\frac{\sum_{j=1}^{n} t_{ij}}{\sum_{i=1}^{n} t_{ij} + \sum_{j=1}^{n} t_{ij}} e_i + \frac{\sum_{i=1}^{n} t_{ij}}{\sum_{i=1}^{n} t_{ij} + \sum_{j=1}^{n} t_{ij}} e_j \right)} \tag{3.3}$$

Se puede ver que los ponderadores suman 1. La idea de la fórmula es dar un peso a cada porcentaje de evasión acorde según el flujo que sale de ella.

Capítulo 4

Resultados

En este capítulo se va mostrar los principales resultados obtenidos de la expansión del día completo y de la expansión mañana y tarde, detallados en el capítulo anterior. También se analiza la eficiencia de los 3 algoritmos presentados para resolver el problema del escalamiento de matrices. Al final del capítulo, se muestra los resultados utilizando fórmulas alternativas para la evasión destino D_j.

4.1. Eficiencia de algoritmos

La resolución del problema de escalamiento de matrices se hizo a través de 3 algoritmos: **Algoritmo de Furness**, **Algoritmo de flujo** y **Optimización convexa**. Los métodos anteriores se detallaron en el capítulo 2. Como la matriz de validaciones existe tanto para las zonas y comunas, se testearon la eficiencia de los algoritmos en 6 instancias: comunas día, comunas mañana, comunas tarde, zonas día, zonas mañana, zonas tarde. En total son 34 comunas y 792 zonas. El error impuesto fue $\varepsilon = 10^{-7}$. Recordando, la complejidad de Furness es $\widetilde{O}(mh^2/\varepsilon^2)$, la de algoritmo de flujo es $O(n^4(\log(n)+\log(\frac{h}{\varepsilon})))$ y la de optimización convexa es $\widetilde{O}(mn^{2/3}h^{1/3}/\varepsilon^{2/3})$.

Tabla 4.1: Tiempos de resolución por comunas

Algoritmo	Comunas día	Comunas mañana	Comunas tarde
Furness	1 seg	<1 seg	<1 seg
Flujo	$30seg$	5 seg	6 seg
Optimización convexa	1 seg	<1 seg	<1 seg

Tabla 4.2: Tiempos de resolución por zonas

Algoritmo	Zonas día	Zonas mañana	Zonas tarde
Furness	30 seg	5 seg	5 seg
Flujo	∞	∞	∞
Optimización convexa	45 seg	8 seg	7 seg

Además se incluyó una instancia extra-grande, el cual consiste en una matriz random de dimensiones 10000×10000. donde cada entrada es un número en el intervalo abierto $(5000, 10000)$

Tabla 4.3: Instancia extra-grande

Algoritmo	Instancia extra-grande
Furness	12 min y 52 seg
Flujo	∞
Optimización convexa	10 min y 23 seg

Todas las pruebas se hicieron en un computador con procesador Intel i5 y 6 gigas de RAM.

Cabe destacar que el algoritmo de flujo para instancias grandes gasta enormes recursos para su resolución, por tener que crear variables auxiliares para convertir en programación lineal, por lo cual aparece con tiempo ∞ en algunos casos.

Se puede apreciar que claramente el algoritmo de Furness y el algoritmo de optimización convexa tienen tiempo de resolución mucho más cortos y muy parecidos entre sí. Claramente son los algoritmos más eficientes. Para valores de h más grandes, especialmente en la instancia extra-grande, se tiene que la optimización convexa es más rápido, lo cual concuerda con la complejidad teórica.

4.2. Evasión día completo

Una vez que se haya georreferenciado completamente los datos de fiscalización, ya se puede calcular la cantidad de evasores y la cantidad total de pasajeros que se suben al bus en cada paradero. La evasión de cada comuna simplemente es la cantidad de evasores que se suben en cada uno de los paraderos que pertenecen a la comuna divido por el total de pasajeros.

En la siguiente tabla se puede apreciar las comunas con mayor y menor evasión. Los datos de evasión es sobre el día completo.

Tabla 4.4: Tabla con las comunas con mayor índice de evasión

Comuna	Índice de evasión
La Pintana	55.6 %
San Ramón	50.6 %
El Bosque	46.5 %
La Granja	46.3 %
San Bernardo	44.8 %
Puente Alto	44.4 %
Renca	43 %
Quilicura	41.7 %
Huechuraba	40.9 %
Recoleta	37.5 %

Tabla 4.5: Tabla con las comunas con menor índice de evasión

Comuna	Índice de evasión
La Condes	17 %
Providencia	18.4 %
La Reina	18.7 %
Ñuñoa	20.9 %
Lo Barnechea	21.8 %
Macul	23.2 %
Santiago	24.8 %
Lo Prado	26.3 %
Peñalolén	26.4 %
Vitacura	26.4 %

El mapa térmico de la evasión en las 34 comunas de Santiago se muestra a continuación. El color rojo indica mayor evasión y el color amarillo lo contrario. Se puede apreciar que las comunas del sur de santiago presentan mayor evasión que las comunas del sector del norte.

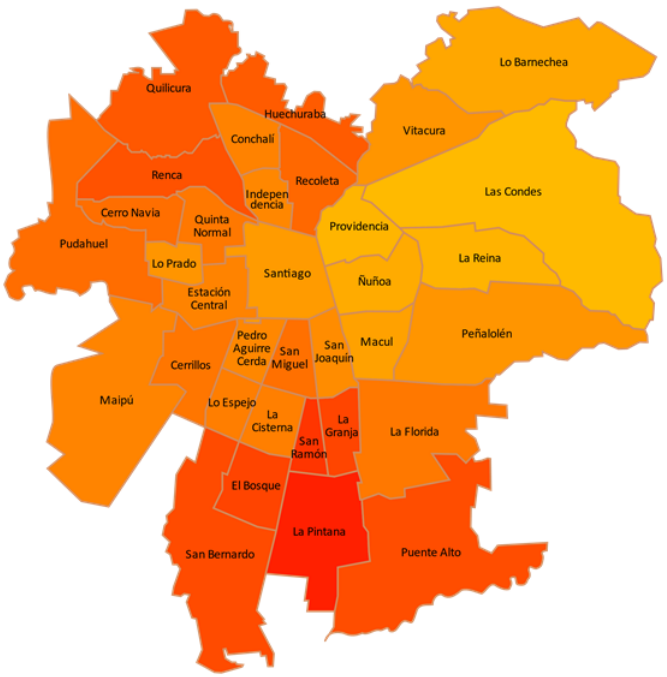


Figura 4.1: Mapa térmico de la evasión de Santiago

El análisis por zonas es ligeramente diferente que el de las comunas. Esto es debido al alto número de zonas que existen (unos 800) y baja cantidad de datos de evasión en un año entero, lo cual se traduce en que hay zonas con muy pocos datos, lo cual hace poco confiable el análisis. Por ejemplo: si los datos en la zona 0 indican que se subió solo una persona y ésta validó, la evasión total de la zona sería del 0 %, lo cual claramente es por falta de datos. Una manera que se hizo para solucionar éste problema fue establecer arbitrariamente que en cada zona debe haber fiscalizado a 30 personas como mínimo. Si ésto no ocurre, se reemplaza el porcentaje de la evasión de la zona por el de la comuna al que pertenece la zona.

Un procedimiento alternativo a lo anterior es considerar la evasión histórica de cada zona. Por ejemplo, si una zona tiene poco datos de fiscalización en el último año 2016, entonces se considera los datos de evasión del año 2015 y así calcular su índice de evasión con los datos del 2015 y 2016. Si todavía no alcanza el número de datos disponibles, se podría repetir el proceso para los años previos y así hasta cumplir con la cantidad de datos requerido.

A continuación se puede apreciar el índice de evasión de las primeras 5 zonas de un total de 792 zonas.

Tabla 4.6: Tabla de evasión por zonas

zona	Índice de evasión
0	9.5 %
1	29.3 %
2	38 %
3	15.6 %
4	29.4 %
5	12.5 %

4.3. Resultados expansión día completo

El escalamiento del día completo se hizo a nivel de zonas, y como cada zona pertenece a una comuna, se puede traducir con facilidad de zonas a comunas. La fórmula ocupada para la evasión destino D_j es la de 3.1. Los números a nivel globales de la expansión se muestran a continuación

Tabla 4.7: Viajes totales día completo

Validaciones	Viajes totales	Evasión global
2.5 millones	3.7 millones	32.5 %

El número total de validaciones se consideran viajes solo Bus y viajes que combinan Bus y Metro. El número de viajes totales es la suma del vector O. La evasión global es del 32,5 % lo cual concuerda los datos últimos datos de evasión [6]. Cabe destacar que la matriz resultante de la expansión contiene alguros pares OD que disminuyen ligeramente su cantidad viajes, lo cual genera evasión negativa. Para evitar estos casos, se puede imponer que esos pares tienen evasión cero. Cerca del 2 % de los pares OD, que tiene cantidad de viajes positivos, tienen este tipo de comportamiento, lo cual no es significativo.

En la siguiente tabla se puede apreciar los pares OD con mayor evasión neta.

Tabla 4.8: Tabla con las OD de comunas con mayor evasión neta

Origen	Destino	Evasión neta
Maipú	Santiago	15.000
Santiago	Maipú	13.500
Puente Alto	La Florida	12.600
Puente Alto	Santiago	12.200
San ramón	La Pintana	11.700
Pudahuel	Santiago	11.600
La Florida	Puente Alto	10.600
La Pintana	San Ramón	9.600
Santiago	La Pintana	9.500
Santiago	Pudahuel	9.500

Una parte interesante de la expansión es poder estudiar los flujos de evasores hacia las comunas que poseen gran cantidad de habitantes. En los siguientes mapas térmicos se pueden apreciar la evasión de los viajes que tiene como destino a Santiago y a Las Condes.

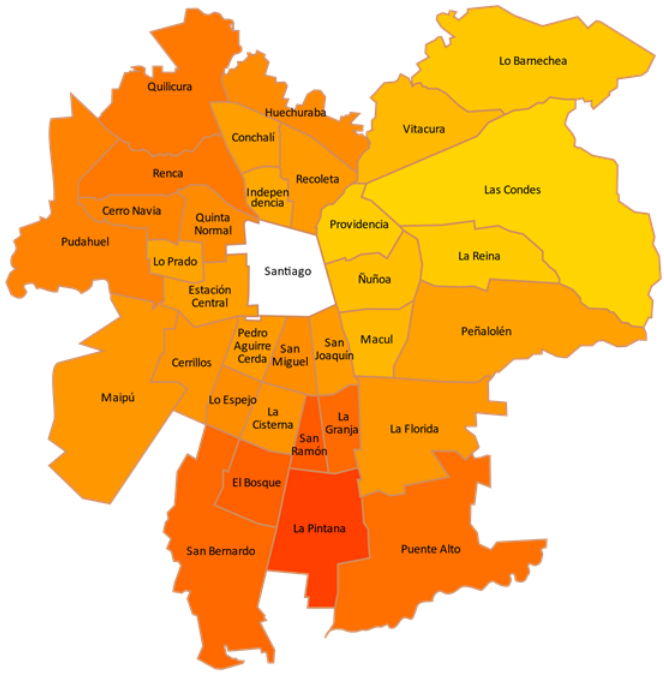


Figura 4.2: Mapa térmico de la evasión hacia Santiago como destino

Figura 4.3: Mapa térmico de la evasión hacia Las Condes como destino

Las comunas Las Condes y Santiago aparecen destacados en blanco.

Se puede apreciar que la evasión hacia Las condes es mucho menor que la evasión de los viajes hacia Santiago. Ésto es esperable ya que la evasión origen en Las Condes es menor que la evasión en Santiago.

4.4. Evasión mañana tarde

Al limitar por tramos horarios, la cantidad de datos disponibles es menor, lo cual genera dificultades para calcular índices de evasión por zonas. Para solucionar este inconveniente, se eligió trabajar con comunas, en un nivel más agregado que por zonas.

Los mapas térmicos de la evasión de las comunas en la mañana y en la tarde

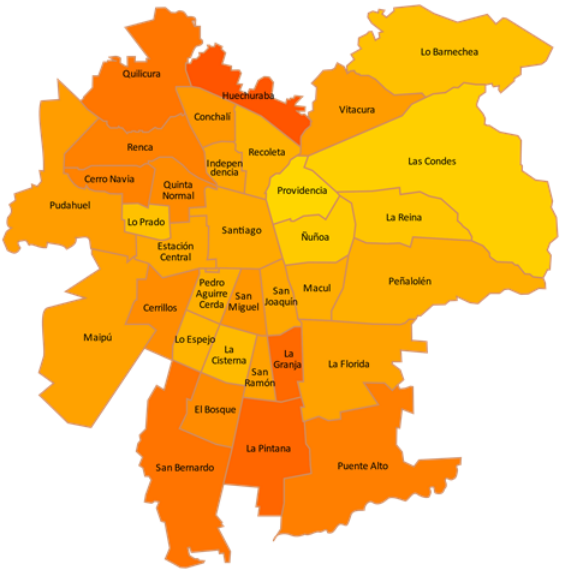


Figura 4.4: Mapa térmico de la evasión de la mañana

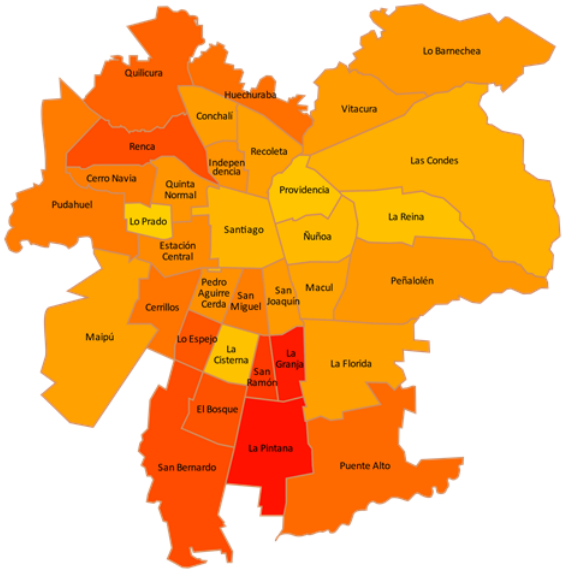


Figura 4.5: Mapa térmico de la evasión de la tarde

Se puede apreciar que en la tarde presenta colores más rojizos, lo cual quiere decir que el **índice de evasión de la tarde es mayor que el de mañana**.

Hay comunas que presentan notorias diferencias en su índice de evasión mañana con respecto al de la tarde. A continuación una lista de comunas que presentan las mayores diferencias.

Tabla 4.9: Tabla con las comunas que disminuyen

Comunas	Evasión mañana	Evasión tarde
Huechuraba	48 % %	40 %
Lo Prado	17 % %	13 %
Santiago	24 %	21 %

Tabla 4.10: Tabla con las comunas que aumentan

Comunas	Evasión mañana	Evasión tarde
San Ramón	22 %	56 %
Lo Espejo	21 %	48 %
La Pintana	43 %	67 %

4.5. Resultados expansión mañana tarde

Como se ha mencionado anteriormente, se hizo dos expansiones: uno en la mañana y uno en la tarde. Las fórmulas usadas fueron explicadas en la sección 3.3. En ninguno de los casos, se generaron pares OD con expansión negativa.

Es interesante estudiar el comportamiento de la evasión de los viajes hacia y desde Santiago y Providencia en los dos tramos horarios considerados, por ser comunas de bastante flujo en las horas peak.

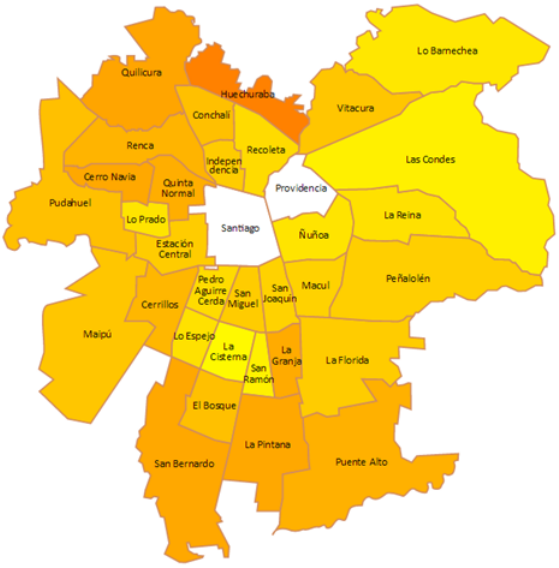


Figura 4.6: Mapa térmico de la evasión hacia Santiago y Providencia como destino mañana

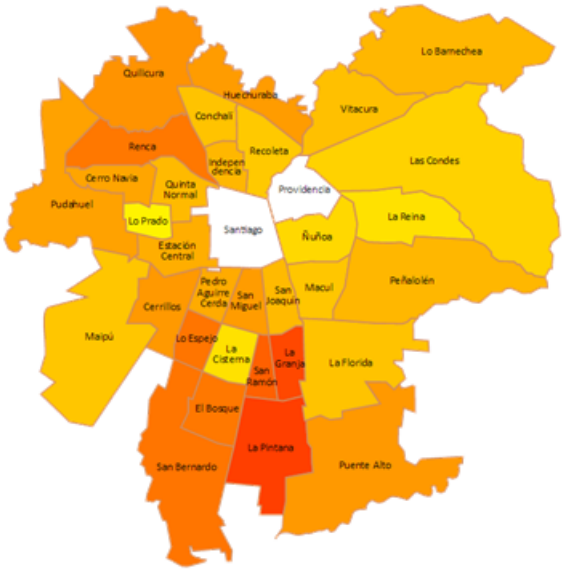


Figura 4.7: Mapa térmico de la evasión hacia Santiago y Providencia como destino tarde

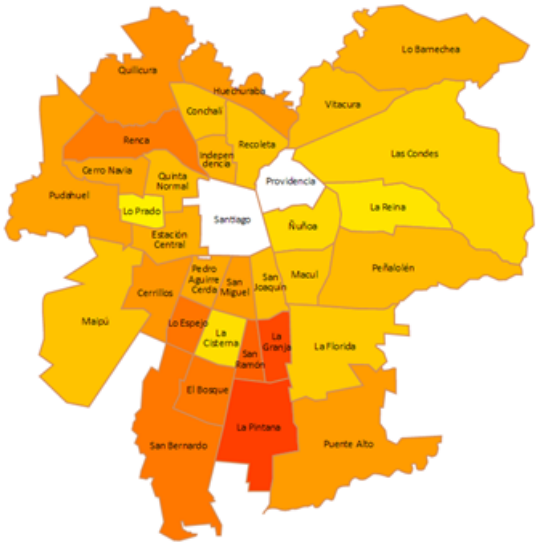


Figura 4.8: Mapa térmico de la evasión desde Santiago y Providencia como origen mañana

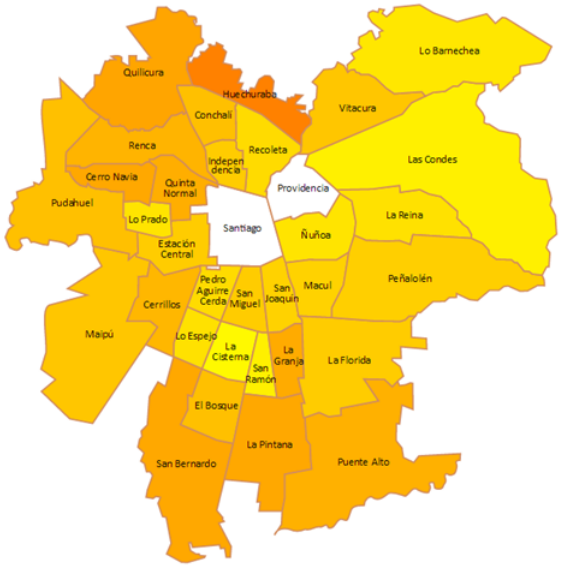


Figura 4.9: Mapa térmico de la evasión desde Santiago y Providencia como origen tarde

Los pares de OD de comunas que presentan mayor evasión neta durante la mañana y la tarde.

Tabla 4.11: Tabla con las comunas con mayor evasión neta mañana

Origen	Destino	Evasión mañana
Puente Alto	Puente Alto	12.600
Maipú	Santiago	6.700
Maipú	Maipú	6.400
La Pintana	La Pintana	6.100
Puente Alto	Santiago	5.200

Tabla 4.12: Tabla con las comunas con mayor evasión neta tarde

Origen	Destino	Evasión tarde
Puente Alto	Puente Alto	14.000
Maipú	Maipú	9.800
Santiago	Maipú	5.700
Quilicura	Quilicura	5.500
La Pintana	La Pintana	4.900

4.6. Fórmulas alternativas de evasión destino

A continuación se muestra el comportamiento de la evasión de destino según las fórmulas alternativas detalladas en la sección 3.4. Se muestra el comportamiento para la mañana y la

tarde.

En la mañana

Figura 4.10: Mapa evasión destino mañana para $D_j = \sum_{i=1}^{n} \delta \frac{t_{ij}}{1-\frac{1}{2}(e_i+e_j)}$

Figura 4.11: Mapa evasión destino mañana para $D_j = \sum_{i=1}^{n} \delta \frac{t_{ij}}{1-\left(\frac{\sum_{j=1}^{n} t_{ij}}{\sum_{i=1}^{n} t_{ij}+\sum_{j=1}^{n} t_{ij}} e_i + \frac{\sum_{i=1}^{n} t_{ij}}{\sum_{i=1}^{n} t_{ij}+\sum_{j=1}^{n} t_{ij}} e_j\right)}$

En la tarde

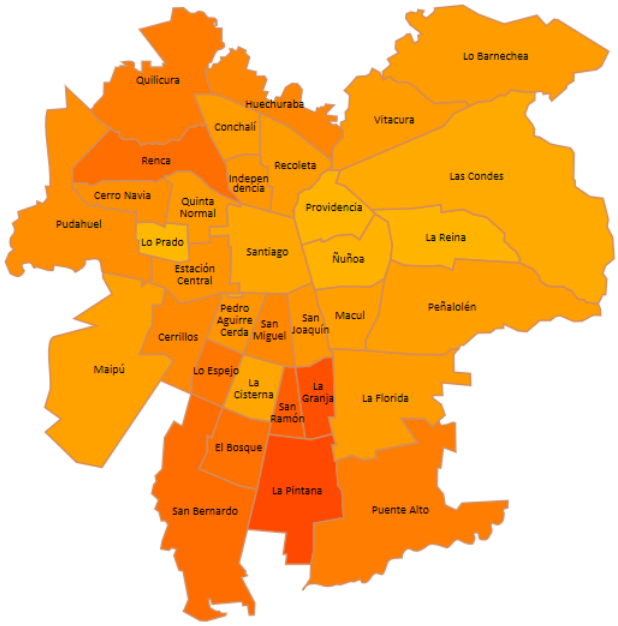


Figura 4.12: Mapa evasión destino tarde para $D_j = \sum_{i=1}^{n} \delta \frac{t_{ij}}{1-\frac{1}{2}(e_i+e_j)}$

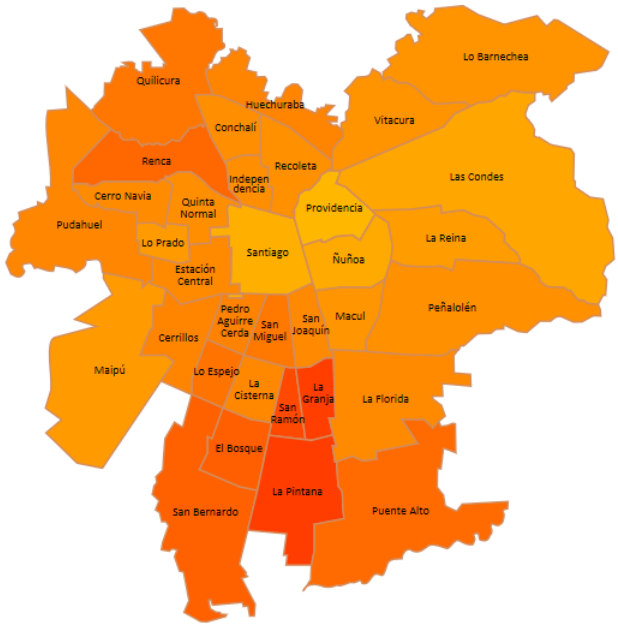


Figura 4.13: Mapa evasión destino tarde para $D_j = \sum_{i=1}^{n} \delta \frac{t_{ij}}{1-\left(\frac{\sum_{j=1}^{n} t_{ij}}{\sum_{i=1}^{n} t_{ij}+\sum_{j=1}^{n} t_{ij}} e_i + \frac{\sum_{i=1}^{n} t_{ij}}{\sum_{i=1}^{n} t_{ij}+\sum_{j=1}^{n} t_{ij}} e_j\right)}$

Se puede apreciar que en la mañana es donde la fórmula del factor promedio se diferencia más de la fórmula por ponderación por flujos. En la tarde las diferencias son menos notorias. Notar que Las Condes y comunas cercanas a ella tienen menor evasión destino en la fórmula de ponderación por flujos en ambos horarios.

Capítulo 5

Aplicación

En este capítulo se mostrará una aplicación de la estimación de flujos de evasión hecha en el capítulo anterior. El problema que se quiere resolver consiste en identificar los mejores paraderos del Transantiago donde colocar fiscalizadores. Para ello, se verá que los flujos de evasión juegan un papel central en la realización del modelo.

5.1. Modelo de control aleatorio

El modelo utilizado se basa principalmente, con ligera modificaciones, en un paper publicado por Correa, Harks, Kreuzen, Matuschke [8]. El modelo consiste en un grafo dirigido $G = (V, E)$ con una función de costos $c : E \to \mathbb{Z}_+$ en sus arcos, el cual tiene relación con el costo de tiempo de viajar en un arco, traducido en costo monetario. El operador del sistema, para prevenir la evasión, puede establecer probabilidades de control $p_e \in [0, 1]$ en cada arco. El presupuesto del operador es limitado, lo cual se traduce en la restricción $\sum_{p_e \in E} p_e \leq B$, donde $B \geq 0$ indica el presupuesto disponible.

Evasión. Los evasores son modelan mediante un conjunto de commodities K. Cada commodity i consiste en un par origen-destino (s_i, t_i) donde s_i, t_i son nodos del grafo. Además, en cada commodite i hay una demanda $d_i \geq 0$, el cual representan la cantidad de evasores que viajan en ese commodity.

Un evasor elige una ruta antes del viaje y termina el viaje independientemente si se encuentra con un fiscalizador. Si un evasor es fiscalizado, entonces debe pagar una multa $F \geq 0$.

Dado un camino P, el costo de tiempo de viajar en ese camino es representado por

$$c\left(P\right) = \sum_{c_e \in E} c_e$$

La probabilidad de que en un camino P no es fiscalizado es

$$\pi(P) = \prod_{e \in P} (1 - p_e)$$

Dados las probalidades p_e, el costo esperado de viajar en a través de camino P para un evasor i en en este modelo es

$$f_{p,i}(P) = c(P) + (1 - \pi(P)) \cdot F$$

Los evasores del commodity i se asumen que son seres racionales y eligen el camino P de tal que forma el costo esperado $f_{p,i}(P)$ sea el menor entre todos los s_i-t_i caminos.

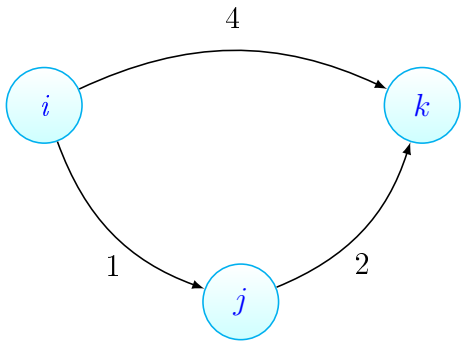
El operador. El problema del operador consiste en fijar probabilidades en los arcos de tal forma que el costo $f_{p,i}(P)$ de los evasores sea lo máximo. Notar que la multa y el precio de pasaje son fijos y no pueden ser cambiados según el operador.

El término $(1 - \pi(P))$ se puede linealizar por $\sum_{p_e \in E} p_e$, con lo cual el problema resultante es solo un problema de camino de costo mínimo. Usando el dual del programa lineal del problema de camino mínimo, se puede derivar en la siguiente relajación

$$\begin{array}{lll}
\text{[LP]} \quad \text{max} & \displaystyle\sum_{i \in K} d_i (y_i(t_i) - y_i(s_i)) & \\
\text{sujeto a} & \displaystyle\sum_{e \in E} p_e \leq B & \\
& y_i(v) - y_i(w) \leq c_e + F p_e & \forall i \in K, e = (v, w) \in E \\
& p_e \in [0, 1] & \forall e \in E
\end{array}$$

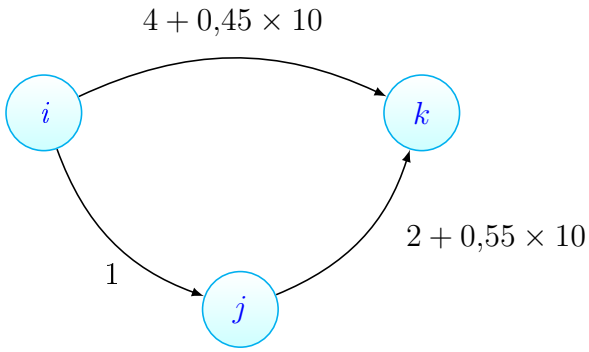
Este LP es el problema central que se usará para encontrar las probabilidades p_e y por lo tanto definir la estrategia de fiscalización.

Ejemplo Sea el siguiente grafo G



Si un evasor quiere viajar desde el nodo i a k tiene dos opciones: el camino P_1 que es viajar del nodo i a j y luego de j a k o el camino P_2 que es viajar usando el arco (i, k). Si no hubiera fiscalización, el evasor elige el camino P_2 por ser el menos costoso.

La multa es de 10 y una solución óptima para el operador del sistema (quien quiere maximizar los costos para el evasor) es asignar probalidad 0.55 al arco (j, k), 0.45 al arco (i, k) y 0 al arco (i, j).



Entonces los costos de los caminos son

- El camino $P1$ tiene costo $4 + 0{,}45 \times 10 = 8{,}5$
- El camino $P2$ tiene costo $3 + 0{,}55 \times 10 = 8{,}5$

Por lo tanto, el evasor es indiferente entre los dos caminos y por lo tanto es una solución óptima.

5.2. Construcción del Grafo

Para poder utilizar el modelo descrito en la sección anterior, es necesario construir un grafo que represente los recorridos en buses del Transantiago que se hacen diariamente. Se pensó en un grafo en donde sus nodos representan paraderos y los arcos solo representan las calles entre paraderos. Ésto quiere decir que las diferentes líneas de buses que recorren el mismo par de paraderos se representan solo con un arco. En la primera etapa, se agregan arcos de caminata entre nodos consecutivos que representan la opción de caminar de parte del pasajero de un paradero a otro.

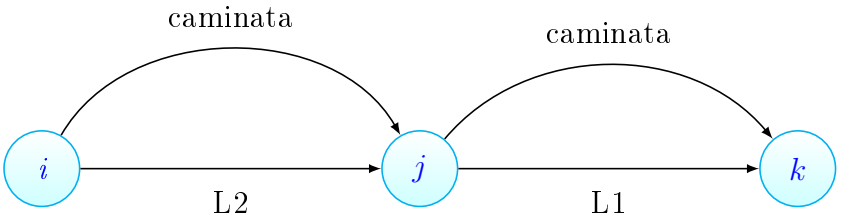


Figura 5.1: Grafo compacto

Finalmente, también se determina un radio de 150 metros en cada nodo del grafo y se agregan todos los arcos de caminata posibles entre el nodo en cuestión y los nodos que pertenecen al radio. Ésto modela el efecto de que el evasor puede caminar de un paradero a otro siempre cuando que no esté más de 150 metros de distancia.

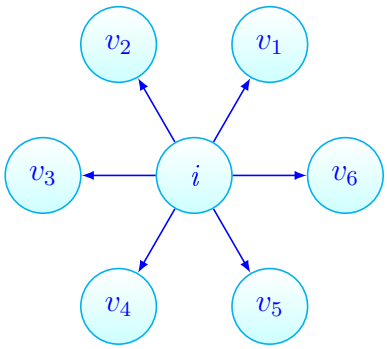


Figura 5.2: arcos de caminata en el radio de 150 metros

El grafo resultante contiene en total unos 11 mil nodos y 50 mil arcos.

5.3. Programación lineal

Una vez que se haya creado el grafo, se procede a resolver el problema de programación lineal. En la versión compacta del grafo, el programa lineal tiene la siguiente forma

$$
\begin{aligned}
\text{[LP]} \quad & \max \quad && \sum_{i \in K} d_i \left(y_i(t_i) - y_i(s_i) \right) && \\
& \text{sujeto a} \quad && \sum_{e \in E} p_e \leq B && \\
& && y_i(v) - y_i(w) \leq c_e + F p_e && \forall i \in K, e = (v,w) \text{ normal} \\
& && y_i(v) - y_i(w) \leq 20 c_e && \forall i \in K, e = (v,w) \text{ caminata} \\
& && p_e \in [0,1] && \forall e \in E
\end{aligned}
$$

El costo de los arcos que no son de caminata es simplemente la distancia de euclidiana entre par de nodos. En la siguiente sección se hablará de cómo trasformar este costo a dinero (pesos).

El factor de $20c_e$ es el costo de los arcos de caminata, el cual es 20 veces más que viajar en bus. Juntando con la restricción de arcos normales, se impone el hecho de que el costo esperado en cada arco no puede ser más que 20 veces que el costo de viajar en ese arco mediante bus.

Una parte importante del modelo es elegir a los commodities y asignar el flujo de evasores que les corresponde. Gran parte del trabajo ya fue hecho en el capítulo 3, con lo cual ya es

posible disponer de la matriz OD de los flujos de evasión. Sin embargo, se debe mencionar que los nodos del grafo son paraderos, pero la expansión de matrices fue a nivel de zonas y comunas, por lo se hace necesario asociar cada zona a un paradero. Para simplificar, para cada par Origen-Destino con zona i y zona j se asocia a un único paradero de la zona i y a uno de la zona j. Para elegir a los paraderos, se aplicó el siguiente algoritmo:

ALGORITMO 7 Commodity

Input: La matriz OD A con el flujo de evasión, el grafo G;

$c_{i,j} = 0$ para todo $i, j = 1, 2, \ldots, n$
for (i, j) **do**
  for paraderos de la zona i y j **do**
    $l \leftarrow$ el costo del camino de costo mínimo entre los paraderos
    if $l > c_{i,j}$ **then**
      $c_{i,j} \leftarrow l$
      $d_{i,j} \leftarrow A_{i,j}$
      $p_{i,j} \leftarrow$ paraderos
    end if
  end for
end for
return p, d

El vector p contiene todos los paraderos que representan a la zona i y a la zona j, el vector d la demanda de evasores de ese commodity.

La idea del algoritmo es elegir el camino de costo máximo entre todos los caminos de costo mínimo posibles. Ésto asegura que los paraderos representantes de cada par O-D estén a una distancia considerable y además existe al menos un camino en el grafo que los une.

5.4. Resultados

Se trabajó con un grafo reducido a tres comunas de Santiago: Santiago, Maipú y Estación Central. En total el grafo reducido a tres comunas contiene cerca de 1800 nodos y unos 8 mil arcos.

Para el programa lineal, se utilizó un presupuesto $B = 4$, una multa $F = 65000$ pesos. Para transformar los distancia de los arcos a dinero se utilizó la velocidad media de un bus que es de 17 kilómetros por hora y el costo de 0.132 euros por minuto de espera. El cambio de euros a pesos fue de 757 pesos por euro.

Se resolvió el programa lineal para dos tramos horarios:

- **Horario mañana**: 06:00- 10:00
- **Horario tarde**: 17:00- 21:00

De un total de 10 mil commodities se eligió trabajar con cerca de 400 commodities que tenían mayor flujo tanto para la mañana como para la tarde.

Una visualización parcial de los resultados usando Google Maps se muestran a continuación

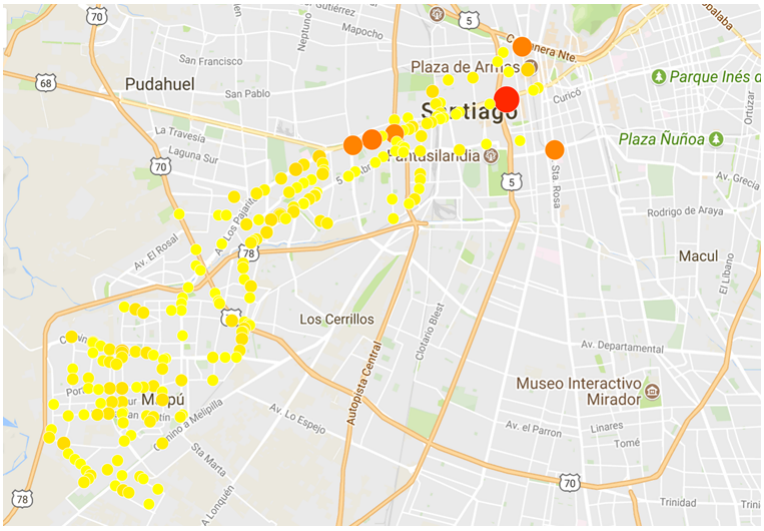


Figura 5.3: Lugares de fiscalización de Santiago en la mañana

Los puntos que aparecen en el mapa indican los paraderos que tienen probabilidad de mayor que 0. El color rojo indica mayor probabilidad y el color amarillo una menor probabilidad.

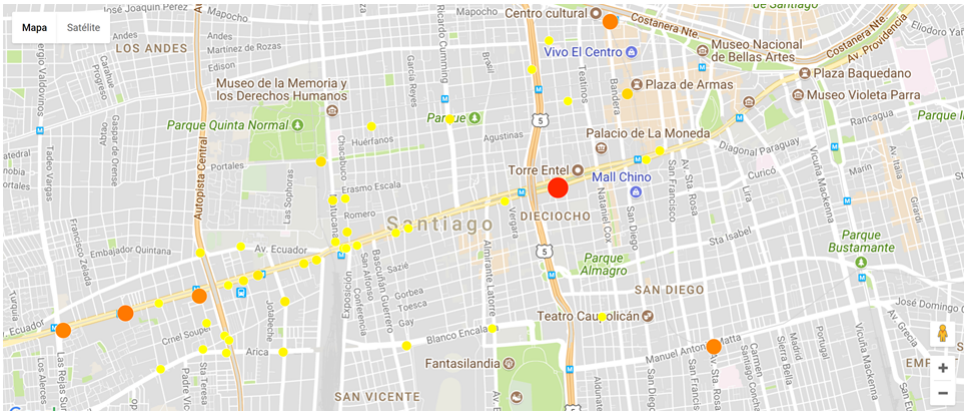


Figura 5.4: Alrededores de Parque O'Higgins mañana

La visualización de los resultados para la tarde

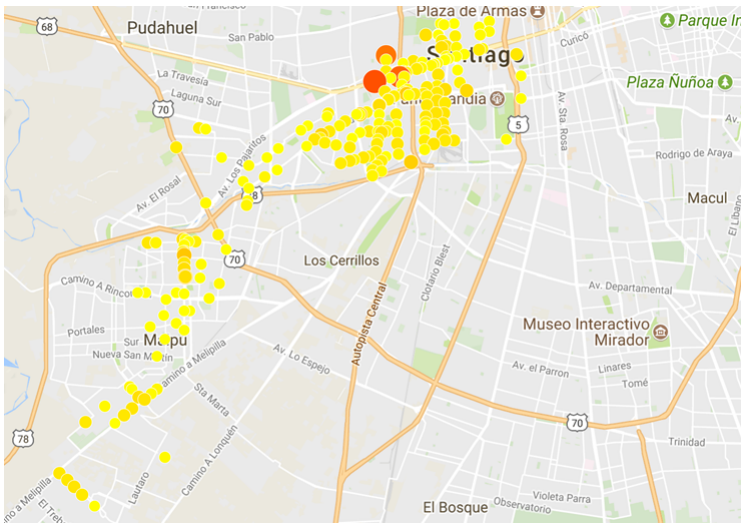


Figura 5.5: Lugares de fiscalización de Santiago en la tarde

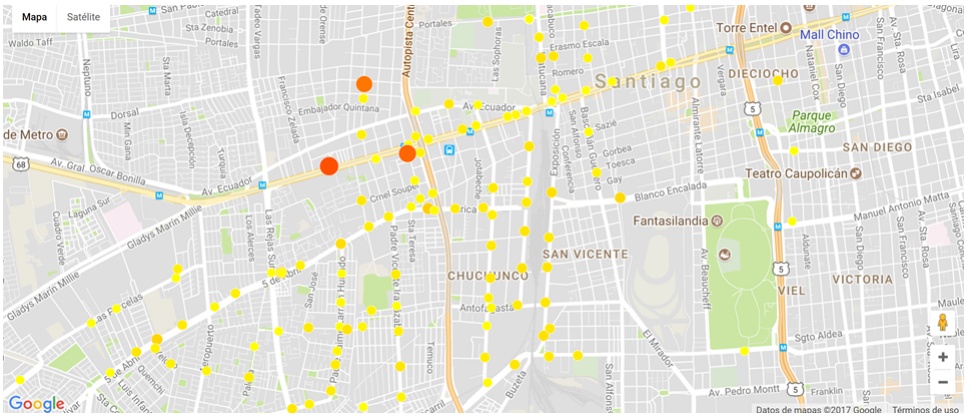


Figura 5.6: Alrededores de Parque O'Higgins en la tarde

5.5. Fiscalización histórica

Los resultados de la sección anterior se pueden comparar con la fiscalización histórica hecha por el área de fiscalización del Ministerio de Transporte, hecha con los datos explicados en la sección 1.1.1. La diferencia es que estos datos no están divididos por tramos horarios.

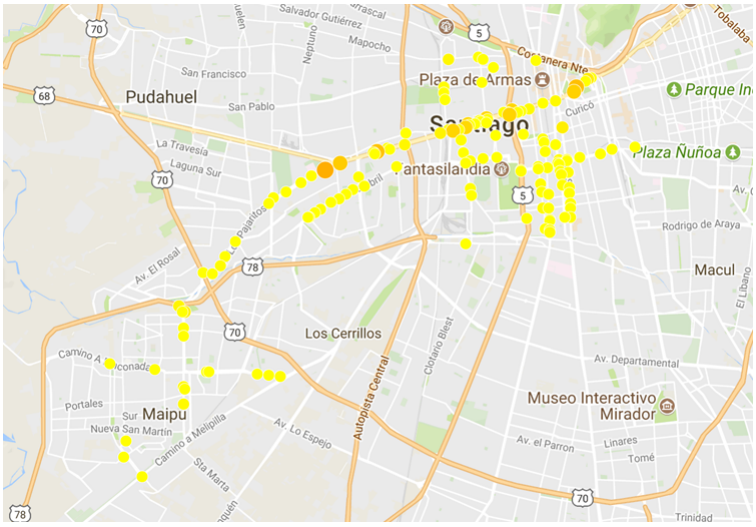


Figura 5.7: Fiscalización histórica

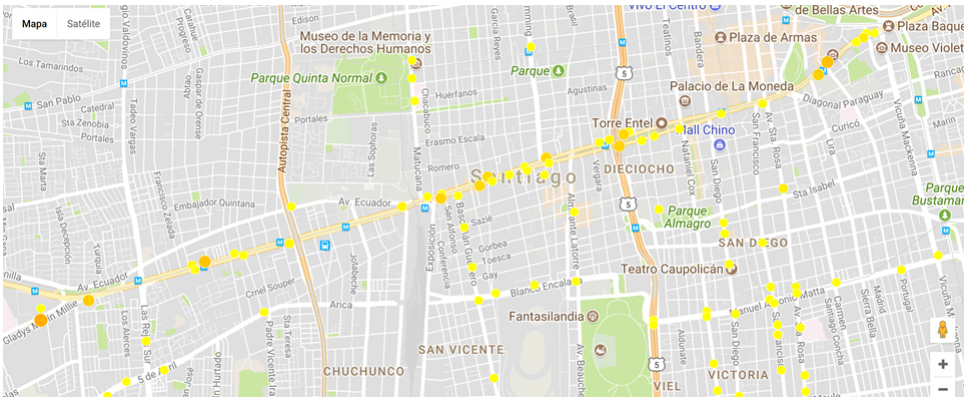


Figura 5.8: Alrededores de Parque O'Higgins en la tarde

Los colores más rojizos indican que fueron los lugares más controlados por los fiscalizadores y viceversa. Las figuras obtenidas por el modelo tiene colores más rojizos, en cambio la fiscalización histórica tiene obtiene colores más amarillentos. Ésto es esperable ya que que el modelo identifica lugares más críticos para fiscalizar. En cambio, la fiscalización histórica fue hecha de manera más uniforme, donde no hubo muchos lugares que fueron muy fiscalizados.

Capítulo 6

Trabajo Futuro

Una de las razones más importantes de que el programa lineal fueran hecho solo en 3 comunas fue la gran cantidad de restricciones y variables que son necesarios crear para resolver el programa lineal. El número de restricciones están dadas por la cantidad de commodities por la cantidad de arcos del grafo. Las variables se dividen en dos clases: las variables llamadas potenciales y las variables que representan las probabilidades en cada arco. Por cada commodity es necesario crear una variable potencial por cada nodo del grafo, con lo cual el número de variable potenciales es el número de nodos por el número de commodities. Por ejemplo, en el grafo reducido a tres comunas, el número de restricciones da $400 \times 8000 = 3200000$ y el número de variables potenciales son $400 \times 1800 = 720000$. Como se puede ver tanto el número de restricciones y variables es bastante alto.

Los tiempos de resolución por tramos horarios se puede apreciar en la siguiente tabla

Tabla 6.1: Tiempos de resolución

Horario	commodities	Tiempo
Mañana	400	10 horas
Tarde	400	2 horas

Las pruebas se hicieron en un computador con procesador Intel i7 y 16 gigas de RAM.

Por lo tanto resolver el programa lineal para todas las comunas de Santiago resulta casi imposible por lo cual es necesario probar con nuevas fórmulas en el futuro. Un método posible para resolver programas lineales de gran tamaño es el **Método de Benders** [4]. Al momento de la escritura de esta memoria, todavía está en proceso de discusión sobre cómo aplicar Benders.

A lo largo de esta memoria se ha podido constatar que el problema de la evasión del Transantiago se puede analizar y formalizar desde el punto de vista matemático. Es esperable que en un futuro mucha de las técnicas planteadas puedan ser mejoradas y afinadas. Por ejemplo, la expansión de matrices hecha no discrimina los viajes hechas en varias etapas, lo

cual es relevante porque existen pasajeros que podrían evadir en una primera etapa y pagar en la segunda etapa del viaje. La georreferenciación también se puede mejorar analizando más datos de tiempos de viaje de líneas de buses. Los costos del grafo que están basadas en distancias euclidianas también pueden ser mejoradas teniendo datos más certeros.

Por último vale cade mencionar que Transantiago está constantemente mejorando su trazado de recorridos, agregando líneas de buses que no existían antes y eliminando líneas obsoletas. Un proceso importante se viene para el Transantigo en el año 2018, el cual es la licitación de las nuevos operadores del Transantiago [25]. Con lo cual es esperable que la malla de recorridos cambien drásticamente. Además en las bases de la licitación se van incluir más medidas anti-evasión como la instalación de torniquetes en los buses. También para finales del año 2017 se espera que esté operativa la línea 6 de metro y para el 2018 la línea 3 [23]. Por todo lo anterior es esperable que el panorama de la evasión en Santiago cambien en los próximos años. Es esperable que la metodología aquí planteada sea una ayuda para afrontar el problema de la evasión del Transantiago.

Capítulo 7

Bibliografía

[1] Software ADATRAP. http://www.uchile.cl/noticias/119440/u-de-chile-desarrolla-software-que-beneficiara-al-transporte-publico. [Online; accessed 28-September-2017].

[2] Zeyuan Allen-Zhu, Yuanzhi Li, Rafael Oliveira, and Avi Wigderson. Much faster algorithms for matrix scaling. *arXiv preprint arXiv:1704.02315*, 2017.

[3] Hamsa Balakrishnan, Inseok Hwang, and Claire J Tomlin. Polynomial approximation algorithms for belief matrix maintenance in identity management. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 5, pages 4874–4879. IEEE, 2004.

[4] Jacques F Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252, 1962.

[5] David T Brown. A note on approximations to discrete probability distributions. *Information and Control*, 2(4):386–392, 1959.

[6] Fiscalización Transporte Chile. Indice de Evasión de Pago de Tarifa en Transantiago. http://www.fiscalizacion.cl/indice-de-evasion-de-pago-de-tarifa-en-transantiago/, 2017-08-01. [Online; accessed 19-July-2017].

[7] Michael B Cohen, Aleksander Madry, Dimitris Tsipras, and Adrian Vladu. Matrix scaling and balancing via box constrained newton's method and interior point methods. *arXiv preprint arXiv:1704.02310*, 2017.

[8] José Correa, Tobias Harks, Vincent JC Kreuzen, and Jannik Matuschke. Fare evasion in transit networks. *Operations Research*, 65(1):165–183, 2017.

[9] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300, 2013.

[10] Juan de Dios Ortuzar, Luis G Willumsen, et al. *Modelling transport*. Wiley New Jersey, 1994.

[11] W Edwards Deming and Frederick F Stephan. On a least squares adjustment of a sampled frequency table when the expected marginal totals are known. *The Annals of Mathematical Statistics*, 11(4):427–444, 1940.

[12] DPTM. Matrices de viajes. `https://www.dtpm.cl/index.php/2013-04-29-20-33-57/matrices-de-viaje`. [Online; accessed 19-July-2017].

[13] DPTM. Programas de operación. `https://www.dtpm.cl/index.php/2013-04-24-14-10-40/2013-04-26-17-44-02`. [Online; accessed 19-July-2017].

[14] Emol. Transantiago recibiría este ano US 715 millones, el mayor subsidio de su historia. `http://www.emol.com/noticias/Nacional/2017/05/17/858682/Transantiago-recibiria-este-ano-US-715-millones-el-mayor-subsidio-de-su-historia.html`, 2017-05-17. [Online; accessed 19-July-2017].

[15] Andrew W Evans. Some properties of trip distribution methods. *Transportation Research*, 4(1):19–36, 1970.

[16] D_ Friedlander. A technique for estimating a contingency table, given the marginal totals and some supplementary data. *Journal of the Royal Statistical Society. Series A (General)*, pages 412–420, 1961.

[17] Gabor T Herman and Arnold Lent. Iterative reconstruction algorithms. *Computers in biology and medicine*, 6(4):273–294, 1976.

[18] Martin Idel. A review of matrix scaling and sinkhorn's normal form for matrices and positive maps. *arXiv preprint arXiv:1609.06349*, 2016.

[19] Bahman Kalantari and Leonid Khachiyan. On the complexity of nonnegative-matrix scaling. *Linear Algebra and its applications*, 240:87–103, 1996.

[20] Bahman Kalantari, Isabella Lari, Federica Ricca, and Bruno Simeone. On the complexity of general matrix scaling and entropy minimization via the ras algorithm. *Mathematical Programming*, 112(2):371–401, 2008.

[21] J Kruithof. Telefoonverkeersrekening. *De Ingenieur*, 52:E15–E25, 1937.

[22] Nathan Linial, Alex Samorodnitsky, and Avi Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 644–652. ACM, 1998.

[23] Metro. Línea 3 y 6 de metro. `https://www.metro.cl/minisitio/linea-3-y-6/`. [Online; accessed 19-July-2017].

[24] Y Nesterov. Introductory lectures on convex programming: a basic course, volume i, 2004.

[25] Publimetro. Así será Transantiago el 2018. https://www.publimetro.cl/cl/noticias/2017/06/05/asi-sera-transantiago-desde-2018-ministerio-ingreso-contralor html, 2017-06-05. [Online; accessed 19-July-2017].

[26] TES Raghavan. On pairs of multidimensional matrices. *Linear Algebra and its Applications*, 62:263–268, 1984.

[27] Günter Rote and Martin Zachariasen. Matrix scaling by network flow. In *Symposium on Discrete Algorithms: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, volume 7, pages 848–854, 2007.

[28] Uriel G Rothblum and Hans Schneider. Scalings of matrices which have prespecified row sums and column sums via optimization. *Linear Algebra and its Applications*, 114:737–764, 1989.

[29] Michael H Schneider and Stavros A Zenios. A comparative study of algorithms for matrix balancing. *Operations research*, 38(3):439–455, 1990.

[30] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879, 1964.

[31] Richard Stone. *Multiple classifications in social accounting*. University of Cambridge, Department of Applied Economics, 1964.

[32] James Hardy Wilkinson. *Rounding errors in algebraic processes*. Courier Corporation, 1994.

[33] G Udny Yule. On the methods of measuring association between two attributes. *Journal of the Royal Statistical Society*, 75(6):579–652, 1912.