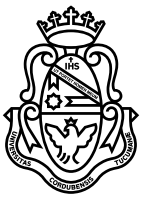
Trabajo especial de grado

Diseño de Rutas y Paradas Óptimas para el Transporte Público de Pasajeros

Autor: Nicolás Jares
Director: Damián Fernandez Ferreyra
Co-Director: Lisandro Armando Parente



Universidad Nacional de Córdoba
Facultad de Matemática, Astronomía y Física
Córdoba, 7 de Agosto de 2015

Abstract

In this work the problem of designing a new bus line for a given system of mass urban passenger transport, with the aim of improving the overall system efficiency, is studied. For that is considered the graph of the streets of a city and its existing lines. Some possible routes between some source-destination pairs are generated by a search strategy that combines the A* search algorithm and Deep First Search. A model of traffic affectation is used to determine the Wardrop user balance. The latter is written as an optimization problem of a concave function on a convex set, which is solved with the projected gradient method.

To make this work, real data from a sector of the city of Cordoba were used: some of its streets and existing lines at the date. In particular it sought to improve the efficiency of the system with respect to access to the offices of the Universidad Nacional de Córdoba.

Classification (Math. Subject Classification)

90B20 - Traffic problem.

90C90 - Operations research, mathematical programming. Applications of mathematical programming.

Key words

-Traffic assignment problems

-Non linear Optimization

Resumen

En este trabajo se estudia el problema de diseñar una nueva línea de colectivo para un sistema de transporte urbano masivo de pasajeros dado, con el objetivo de mejorar la eficiencia total del sistema. Para ello se considera el grafo de las calles de una ciudad y las líneas ya existentes. Se generan algunas rutas posibles entre algunos pares origen-destino mediante una estategia de búsqueda que combina el algoritmo de busqueda A* y el algoritmo de Búsqueda en Profundidad (Deep First Search). Se utiliza un modelo de afectación de tráfico para determinar el equilibrio de usuario de Wardrop. Este último se escribe como un problema de optimización de una función cóncava sobre un conjunto convexo, el cual es resuelto con el método de gradiente proyectado.

Para realizar el presente trabajo se utilizaron datos reales de un sector de la ciudad de Córdoba: algunas de sus calles y las líneas de colectivos existentes a la fecha. En particular se intentó mejorar la eficiencia del sistema con respecto al acceso a las dependencias de la Universidad Nacional de Córdoba.

Clasificación (Math. Subject Classification)

90B20 - Traffic problem.

90C90 - Operations research, mathematical programming. Applications of mathematical programming.

Palabras Claves

-Problemas de Asignación de Tráfico

-Optimización no lineal

Índice general

Introducción

El objetivo de este trabajo es encarar el problema de diseñar una nueva línea de colectivo que reduzca la deficiencia que existe en el sistema de transporte público de la ciudad de Córdoba con respecto al acceso a las dependencias de la Universidad Nacional de Córdoba. Esta deficiencia causa que desde algunos puntos de la ciudad no existan líneas de colectivos que permitan llegar a algunas de las dependencias de la UNC, haciendo necesario realizar transbordos.

Para encarar este problema, se eligió un sector de la ciudad que contiene a todas las dependencias. Se generó el grafo correspondiente a ese sector de la ciudad y se relevaron los recorridos de las líneas de colectivos existentes a la fecha de realización del trabajo. Se tuvieron en cuenta varios pares origen-destino que se consideraron convenientes para el problema y se encontraron algunas rutas posibles entre esos pares usando una estrategia combinada entre los algoritmos de búsqueda A* y de búsqueda en profundidad (Deep First Search). Se utilizó sobre el grafo un modelo de afectación de tráfico que determina flujos que satisfacen la demanda y las restricciones de la red y encuentran el equilibrio del usuario de Wardrop [7]. Para ello se eligieron de manera apropiada las demandas y las funciones de costos para ajustarlas al problema. Se esperó que los flujos óptimos que se encontraran definieran el recorrido de la nueva línea. Finalmente, se resolvió el modelo utilizando el método del gradiente proyectado.

En el Capítulo 1 se describirá con más detalle las motivaciones que llevaron a intentar resolver este problema, así como la formulación. Se mostrarán gráficos que muestran la conectividad actual del Sistema de Transporte Público de Pasajeros.

En el Capítulo 2 se describirá en detalle el modelo que se eligió para resolver este problema. Básicamente, el modelo está formulado como un problema de optimización de una función cóncava sobre un conjunto convexo de la forma:

$$\begin{aligned} \text{minimizar} \quad & T(\Delta^T h) \\ \text{sujeto a} \quad & \Gamma h = g \\ & h \geq 0, \end{aligned}$$

donde la función objetivo $T(v)$ es una función de costo sobre los flujos por arco, v es el vector de flujo por arco, Δ es una matriz que relaciona los flujos por arco con los flujos por ruta, h es el vector de flujos por ruta, Γ es una matriz que relaciona los

flujos por ruta con los flujos entre pares origen-destino y g es el vector de demandas entre pares origen-destino.

Luego, en el Capítulo 3 se explicará de qué manera fue obtenida la información necesaria para construir el modelo. Se describirá cómo se construyó el grafo que representa al sector de la ciudad de Córdoba que fue estudiado, y cómo se obtuvo la información de las líneas existentes. También se explicará cómo se determinaron los pares origen-destino con los que se construyó el modelo.

En el Capítulo 4 se explicará cómo fue construido el modelo, es decir, cómo a partir de la información obtenida se construyeron las matrices Δ y Γ. También se explicará cómo fue diseñada la función $T(v)$, para finalmente concluir con el planteo del problema de optimización mencionado.

Por último, en el Capítulo 5 se analizará la forma del problema de optimización obtenido, explicando algunos resultados de convergencia que garantizan que el resultado obtenido es un mínimo local del problema. Luego se explicará como se llegó a la solución final, para la que fue necesario resolver varios problemas de optimización, y se presentarán algunas conclusiones.

Capítulo 1

Descripción del Problema

1.1. Motivación

La motivación para encarar este problema fue la falta de disponibilidad de líneas de colectivo del Sistema de Transporte Urbano Masivo de Pasajeros (STUMP) que permitan el acceso desde algunos lugares de la ciudad de Córdoba a algunas de las dependencias de la UNC.

Esta situación se evidencia a partir de las figuras 1.1 a 1.3 obtenidas de [2].

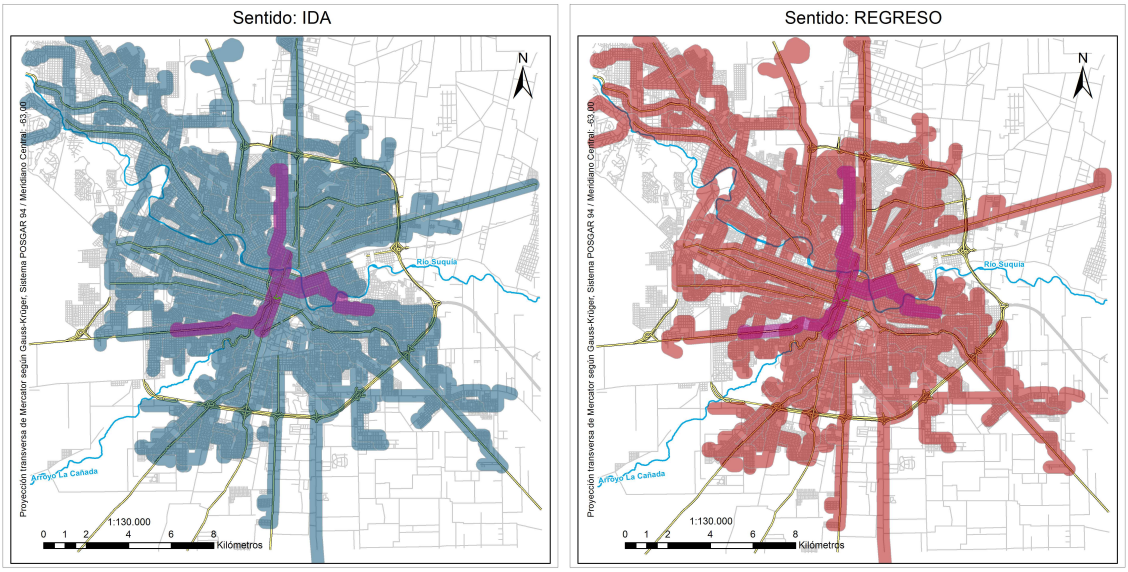


Figura 1.1: Conectividad con dependencias de la UNC en Área Central

En las imágenes se coloreó con azul y rojo las áreas servidas por el STUMP que poseen conectividad con las distintas dependencias de la UNC. Los colores azul y el rojo se corresponden con el sentido Ida y el sentido Regreso respectivamente. En

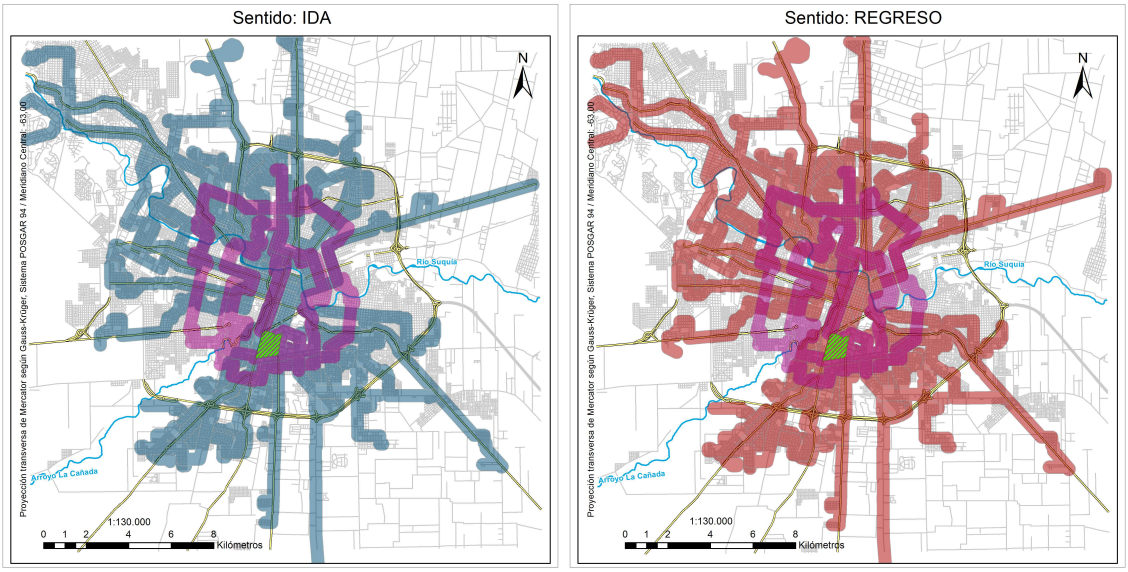


Figura 1.2: Conectividad con Ciudad Universitaria

magenta están coloreadas con el mismo criterio las zonas que poseen conectividad mediante las líneas anulares, de trolebuses y especiales. En verde se resaltan las dependencias de la UNC correspondientes. El criterio utilizado en la confección de estas imágenes para considerar un área como "servida" es que se encuentre a 300 m de una línea de colectivo.

Se sabe que en la ciudad de Córdoba el STUMP es predominantemente radial. Es decir, casi todas las líneas pasan por el centro de la ciudad, y existen lugares desde los cuales la única forma de ir a alguna dependencia de la UNC es yendo primero al centro de la ciudad para realizar un transbordo. En muchos de estos casos, el recorrido realizado está bastante alejado del camino más corto que se podría haber realizado.

En la figura 1.1, correspondiente a la conectividad con las dependencias de la UNC del área central, se observa que la conectividad es casi completa. No obstante se observan algunos "huecos" en la cobertura. Estas son deficiencias propias del sistema de transporte y no se intentará resolverlas en el presente trabajo. En la figura 1.2, correspondiente a la conectividad con Ciudad Universitaria se observa un nivel de cobertura menor al del área central, y se ven grandes zonas no conectadas al suroeste y al noreste de la ciudad. Por la ubicación de la Ciudad Universitaria con respecto al centro de la ciudad, y teniendo en cuenta la radialidad del STUMP es fácil explicar la ubicación de las áreas no servidas. Por último, en la figura 1.3, correspondiente al Hospital Nacional de Clínicas y la E.S.C.M.B., se observa una conectividad aún menor. Esta disminución se explica de manera análoga al del caso de Ciudad Universitaria, teniendo en cuenta que el Hospital y la E.S.C.M.B se

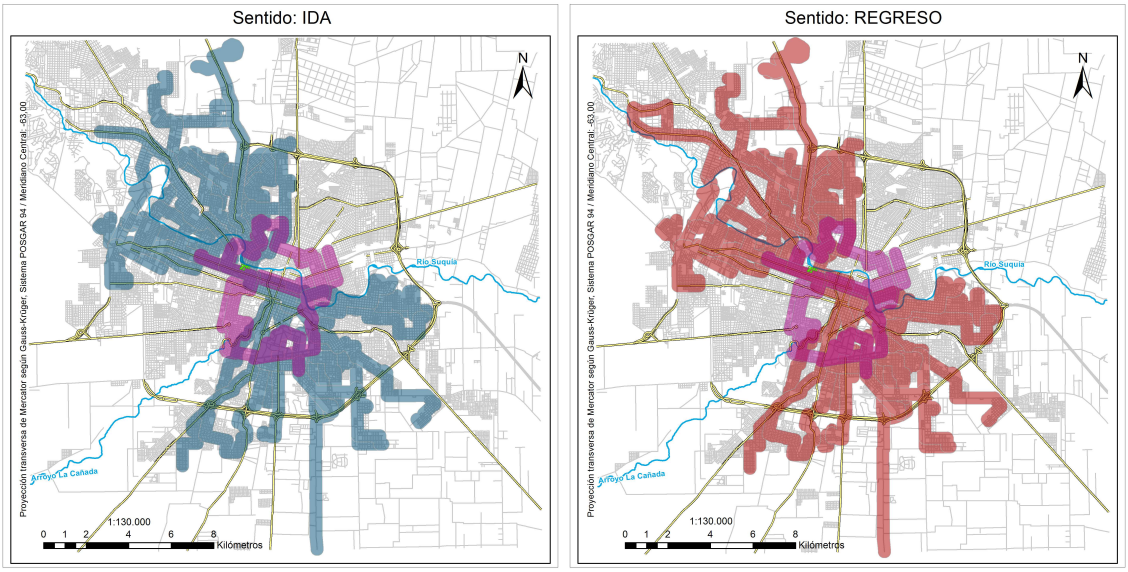


Figura 1.3: Conectividad a Hospital Nacional de Clínicas y E.S.C.M.B.

encuentran aún más lejos del centro de la ciudad.

En [2] no se estudia la conectividad con el Hospital Universitario de Maternidad y Neonatología ni con el Observatorio Astronómico de Córdoba (OAC). En la sección 4.3 se verá que la conectividad con este último es especialmente baja.

1.2. El problema

Siendo claras las deficiencias del STUMP respecto a la conectividad con las dependencias de la UNC, se intentará mejorar esta situación mediante el diseño de una nueva línea.

La nueva línea propuesta serán dos anillos, uno horario y otro anti-horario que pasarán por todas las dependencias de la UNC. La idea es que los eventuales pasajeros que se ven obligados a hacer un transbordo para llegar hasta alguna dependencia de la UNC no tengan que ir hasta el centro de la ciudad, sino que puedan hacer el transbordo en las proximidades de su destino y viajar en la nueva línea hasta el destino. De manera análoga, para el regreso se espera que con la nueva línea los pasajeros puedan viajar en la misma hasta la parada más cercana que los lleve de regreso a su casa.

Para llegar a una formulación matemática del problema se necesita clarificar que se desea mejorar (optimizar). Si se realizara la optimización sólo respecto al tiempo, eso "beneficiaría" a los viajes más largos, que son los que más contribución tendrían a una eventual función objetivo que considere los tiempos de viaje de todos los pasajeros.

Otra alternativa sería utilizar los tiempos pero normalizados en algún sentido. El sentido más natural en el que se podrían normalizar sería considerando la longitud de cada camino. Pero eso sería equivalente a usar la velocidad con la que cada pasajero llega a su destino como el parámetro a optimizar. Ahora bien, calcular una velocidad implica una división y esto trae complicaciones a la hora de formular problemas de optimización ya que se pierde la linealidad.

Para atacar estas dificultades se tendrá en cuenta que la nueva línea intenta mejorar el acceso a un conjunto de lugares (las dependencias de la UNC) que se encuentran dentro de un radio pequeño en comparación con la ciudad. Luego se puede considerar que la mejora en los tiempos se va a realizar sólo en la parte del viaje que esté incluida dentro de ese radio, tanto para los que viven lejos como para los que no tanto. En definitiva, se buscará optimizar los tiempos, pero sólo sobre la parte de las líneas que se encuentren dentro de una zona que contenga a todas las dependencias de la UNC, más algunas cuadras a la redonda.

En la figura 1.4 se ve cual es la zona que se consideró. En la sección 3.1 se explicará como fue construido ese grafo y porqué se decidió hacerlo de esa forma.

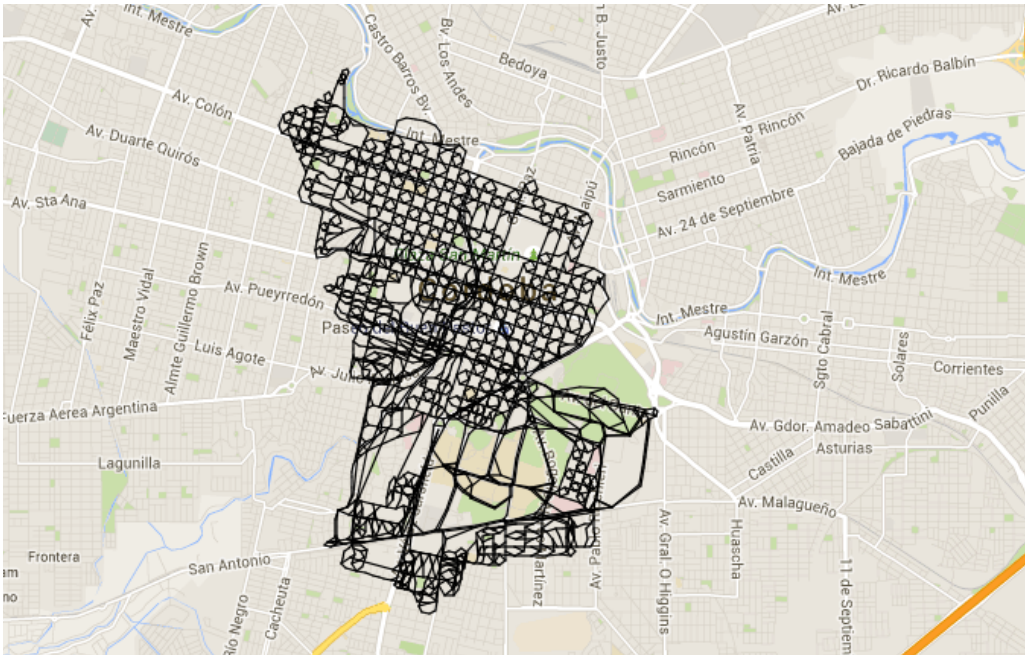


Figura 1.4: Grafo construido de la zona considerada

Capítulo 2

Descripción del Modelo

A continuación se darán las definiciones necesarias para plantear el modelo. Las definiciones de la sección 2.1 y el resultado expuesto en la sección 2.2 fueron obtenidos de [7].

2.1. Definiciones preliminares

A lo largo de este trabajo se hablará mucho de "rutas" y "líneas". Pese a que el significado pueda quedar claro en función del contexto, se intentará utilizar la palabra "ruta" para referirse a las rutas en la teoría de grafos y "línea" para referirse a los distintos recorridos de los colectivos en la ciudad. Se evitará entonces referirse a los recorridos de los colectivos como "rutas" cuando esto pueda generar alguna confusión.

Sea $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$ un grafo dirigido, donde $\mathcal{N}$ es el conjunto de nodos y $\mathcal{A} \in \mathcal{N}\times\mathcal{N}$ es el conjunto de arcos o aristas. Con este grafo se representará una red de transporte, donde cada nodo será una ubicación posible dentro de la red (incluyendo los posibles orígenes y destinos) y cada arco representará un acceso directo entre dos nodos. Como el grafo es dirigido, los arcos tendrán orientación, y por lo tanto no será posible trasladarse entre los nodos de sus extremos de manera indistinta. Si el arco $a \in \mathcal{A}$ es $a = (n, m) \in \mathcal{N}\times\mathcal{N}$, esto significa que la dirección de a es desde n a m y diremos que n es el *nodo inicial* de a y que m es el *nodo final* de a. También diremos que los arcos a y b son *consecutivos* si el nodo inicial de b es igual al nodo final de a.

Definición 2.1.1 (Matriz Origen Destino) *Dado un grafo dirigido $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$ representando una red de transporte, se define la* Matriz Origen Destino *como una matriz $g \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$ donde la entrada g_{pq} representa la cantidad de usuarios de la red que tienen la intención de desplazarse desde el nodo p hacia el nodo q.*

Se define el conjunto $\mathcal{C}$ de pares origen-destino como el conjunto de pares ordenados $c_i = (p, q) \in \mathcal{N}\times\mathcal{N}$ tales que $g_{pq} \neq 0$.

Por simplicidad en la notación, se escribirá la Matriz Origen Destino como un vector $g \in \mathbb{R}^{|\mathcal{C}|}$ donde la entrada $g_i = g_{pq}$ representa la cantidad de usuarios con origen p y destino q si $c_i = (p, q) \in \mathcal{C}$.

Se dirá que g representa una demanda de la red dada por $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$.

Definición 2.1.2 (Ruta) *Sea $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$ un grafo dirigido representando una red de transporte y g una demanda de esa red. Dado un par $(p, q) \in \mathcal{C}$ diremos que una colección ordenada de arcos $\{a_1, a_2, a_3, \ldots, a_k\} \subset \mathcal{A}$ es una* ruta *para ese par si:*

(i) el nodo inicial de a_1 es p

(ii) el nodo final de a_k es q

(iii) los nodos a_i y a_{i+1} son consecutivos para $i = 1, 2, \ldots, k-1$

Se define el conjunto $\mathcal{R}_{pq}$ como el conjunto de todas las rutas para el par (p, q). Análogamente al caso de g, se escribirá $\mathcal{R}_i$ como el conjunto de rutas para el par $c_i = (p, q)$.

Se dirá que una ruta es simple *si no contiene ciclos.*

Definición 2.1.3 (Matriz de incidencia arco-ruta) *Sea $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$ un grafo dirigido representando una red de transporte, g una demanda de esa red y $\mathcal{R}_{pq}$ las rutas para el par $(p, q) \in \mathcal{C}$. Se define la* Matriz de incidencia arco-ruta *como el arreglo*[1] *$\Delta = (\delta_{pqra})$ tal que $\forall (p, q) \in \mathcal{C}$, $\forall r \in \mathcal{R}_{pq}$, $\forall a \in \mathcal{A}$:*

$$\delta_{pqra} = \begin{cases} 1 & \text{si } a \in r \\ 0 & \text{en cualquier otro caso} \end{cases}$$

Definición 2.1.4 (Matriz de incidencia arco-nodo) *Sea $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$ un grafo dirigido representando una red de transporte. Se define la* Matriz de incidencia arco-nodo *como la matriz $A = (a_{ib}) \in \mathbb{R}^{|\mathcal{N}| \times |\mathcal{A}|}$ tal que:*

$$a_{ib} = \begin{cases} 1 & \text{si el nodo } i \text{ es nodo inicial del arco } b \\ -1 & \text{si el nodo } i \text{ es nodo final del arco } b \\ 0 & \text{en todo otro caso} \end{cases}$$

Definición 2.1.5 (Afectación de tráfico factible) *Sea $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$ un grafo dirigido representando una red de transporte, g una demanda de esa red y $\mathcal{R}_{pq}$ las rutas para el par $(p, q) \in \mathcal{C}$. Se definen los vectores de* flujo por ruta *y* flujo por arco *como*

[1] Pese a la definición de Δ con cuatro índices, más adelante será reindexada y quedará claro porque se la llama matriz

$h = (h_{pqr})_{(p,q)\in\mathcal{C},\, r\in\mathcal{R}_{pq}}$ *y* $v = (v_a)_{a\in\mathcal{A}}$ *respectivamente. Diremos que cualquiera de ellos representa una* afectación de tráfico.

Dados v *y* h *vectores de flujo por arco y flujo por ruta respectivamente, diremos que representan la misma afectación de tráfico si se cumple*[2] $v_a = \sum_{r\in\mathcal{R}_a} h_{pqr}$ *donde* $\mathcal{R}_a = \{r \in \bigcup_{(p,q)\in\mathcal{C}} \mathcal{R}_{pq} \mid a \in r\}$.

Luego, dada una afectación de tráfico, se puede obtener sin problemas el vector de flujo por arco dado el vector de flujo por ruta.

Por último, dada una afectación de tráfico diremos que es una afectación de tráfico factible *si las componentes de los vectores de flujo que la representan guardan volúmenes de viaje que satisfacen la demanda de la red. Esto es, si sucede* $\sum_{r\in\mathcal{R}_{pq}} h_{pqr} = g_{pq}, \forall (p,q) \in \mathcal{C}$.

De ahora en más se dirá afectación de tráfico para referirse a una afectación de tráfico factible.

Definición 2.1.6 (Afectación de equilibrio) *Sea* $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$ *un grafo dirigido representando una red de transporte,* g *una demanda de esa red y* $\mathcal{R}_{pq}$ *las rutas para el par* $(p,q) \in \mathcal{C}$. *Diremos que el vector de flujo por ruta* $h^* = (h^*_{pqr})$ *o de flujo por arco* $v^* = (v^*_a)$ *representa una* afectación de equilibrio *si representa una afectación de tráfico que además satisface alguna noción de equilibrio.*

Definición 2.1.7 (Costo de arcos y rutas) *Sea* $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$ *un grafo dirigido representando una red de transporte,* g *una demanda de esa red y* $\mathcal{R}_{pq}$ *las rutas para el par* $(p,q) \in \mathcal{C}$. *El* costo *de recorrer un arco* $a \in \mathcal{A}$ *se definirá mediante una función* $t_a(v_a)$.

Para definir el costo de recorrer una ruta se supondrá aditividad de los costos por ruta. *Esto es, dado* $(p,q) \in \mathcal{C}$, *y* $r \in \mathcal{R}_{pq}$, *el costo de recorrer la ruta será* $c_{pqr} = \sum_{a\in r} t_a(v_a)$.

Se considerará un modelo donde los costos por arco son independientes de los flujos en otros arcos, lo que se conoce como *separabilidad de los costos por arco.*

2.2. El equilibrio de Wardrop

Para definir la noción de equilibrio que se utilizará se considerará que el costo de un arco o una ruta es equivalente al tiempo que se necesita para recorrer el arco o la ruta según corresponda.

[2]Hay un abuso de notación en la forma en la que está escrita esta sumatoria ya que no queda claro el papel que juegan p y q. Por el momento fueron escritos para guardar coherencia con la reciente definición de h. En realidad p y q quedan completamente definidos para cada r. Más adelante h será reindexado con esta idea.

Definición 2.2.1 (Equilibrio del usuario de Wardrop) *Sea $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$ un grafo dirigido representando una red de transporte, g una demanda de esa red y $\mathcal{R}_{pq}$ las rutas para el par $(p,q) \in \mathcal{C}$. Dado $(p,q) \in \mathcal{C}$, llamaremos π_{pq} al costo mínimo con el que se puede ir desde p hasta q. Dada una afectación de tráfico representada por un vector de flujo por ruta $h = (h_{pqr})$, diremos que h satisface el* equilibrio del usuario de Wardrop *si:*

$$\begin{cases} h_{pqr} > 0 \implies c_{pqr} = \pi_{pq}, \ \forall r \in \mathcal{R}_{pq} \\ h_{pqr} = 0 \implies c_{pqr} \geq \pi_{pq}, \ \forall r \in \mathcal{R}_{pq} \end{cases} \tag{2.1}$$

para todo par $(p,q) \in \mathcal{C}$.

Un par (h^*, π^*) que satisface (2.1) se llama solución de equilibrio, h^* se llama afectación de equilibrio y π^* costos de equilibrio.

Todo lo definido hasta ahora puede ser resumido en un único conjunto de ecuaciones que represente al equilibrio del usuario de Wardrop. Dicho conjunto de ecuaciones es:

$$h_{pqr}(c_{pqr} - \pi_{pq}) = 0, \ \forall r \in \mathcal{R}_{pq}, \forall (p,q) \in \mathcal{C} \tag{2.2a}$$

$$c_{pqr} - \pi_{pq} \geq 0, \ \forall r \in \mathcal{R}_{pq}, \forall (p,q) \in \mathcal{C} \tag{2.2b}$$

$$\sum_{r \in \mathcal{R}_{pq}} h_{pqr} = g_{pq}, \ \forall (p,q) \in \mathcal{C} \tag{2.2c}$$

$$h_{pqr} \geq 0, \ \forall r \in \mathcal{R}_{pq}, \forall (p,q) \in \mathcal{C} \tag{2.2d}$$

$$\pi_{pq} \geq 0, \ \forall (p,q) \in \mathcal{C} \tag{2.2e}$$

En esta formulación no se incluyen restricciones sobre las variables de flujo que impliquen que éstas sean enteras. Por ese motivo esta formulación en realidad define una relajación continua de las condiciones de Wardrop originales. Esto se hace por cuestiones de simplicidad y porque dicha aproximación es lo suficientemente buena para redes reales [7].

Con estas definiciones y la notación usada es posible introducir la formulación del problema de optimización. Como se dijo, el modelo que se utilizará es un modelo de afectación de tráfico que pertenece al conjunto de problemas denominados Problemas de Asignación de Tráfico (Traffic Assignment Problem, TAP).

En este caso particular, el problema consiste en determinar flujos (por arco o por ruta) que representen una afectación de equilibrio, donde la noción de equilibrio que se considerará será la del equilibrio del usuario de Wardrop.

Las condiciones (2.2) que representan el equilibrio del usuario de Wardrop no asumen características particulares de las funciones de costo más que la no negatividad. Sin embargo es posible sacrificar la generalidad de estas condiciones agregando hipótesis para que sea posible contar con un problema de optimización cuyas condiciones de optimalidad sean (2.2). De esta manera se podrá encarar el problema de

encontrar esos flujos que representen una afectación de equilibrio como un problema de optimización. Esto permitirá poder utilizar todas las herramientas matemáticas desarrolladas en el campo de la optimización para analizar las soluciones de equilibrio y desarrollar métodos numéricos que las calculen.

Según se explica en [7], la formulación que se utilizará fue primero publicada por Prager en [5], sin embargo la versión más popular se adjudica a Dafermos y Sparrow [3].

Las hipótesis adicionales que se mencionaron son bastante naturales para una red de transporte urbana y se presentan en el enunciado del siguiente teorema. La demostración de este teorema se dará en la sección 5.2.

Teorema 2.2.2 *Sea $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$ un grafo dirigido representando una red de transporte, g una demanda de esa red y $t_a(v_a), a \in \mathcal{A}$ las funciones de costo por arco. Si se cumple que la red es:*

- *fuertemente conectada,*[3]
- *con demandas positivas,*
- *con funciones de costo por arco positivas y continuas,*

entonces las condiciones de optimalidad de primer orden del siguiente problema:

$$\text{minimizar} \quad T(v) = \sum_{a \in \mathcal{A}} \int_0^{v_a} t_a(s)ds \tag{2.3a}$$

$$\text{sujeto a} \quad \sum_{r \in \mathcal{R}_{pq}} h_{pqr} = g_{pq} \qquad \forall (p,q) \in \mathcal{C} \tag{2.3b}$$

$$h_{pqr} \geq 0 \qquad \forall r \in \mathcal{R}_{pq},\ \forall (p,q) \in \mathcal{C} \tag{2.3c}$$

$$\sum_{(p,q) \in \mathcal{C}} \sum_{r \in \mathcal{R}_{pq}} \delta_{pqra} h_{pqr} = v_a \qquad \forall a \in \mathcal{A} \tag{2.3d}$$

son equivalentes a las condiciones de equilibrio del usuario (2.2).

2.3. El modelo

Teniendo enunciado el teorema 2.2.2, se mostrará cómo se reformuló el problema (2.3) para que sea más sencillo escribir posteriormente los códigos en Octave para resolverlo y se explicará cómo esta formulación permitirá diseñar la nueva ruta.

[3] Quiere decir que es posible llegar a cualquier nodo desde cualquier otro nodo respetando el sentido de circulación del grafo dirigido.

En primer lugar, se reordenó a la matriz origen-destino como un vector de componentes no nulas como fue descrito en la definición de la misma. Con este primer paso tenemos que todos los pares (p, q) que se refieren a pares origen-destino en (2.3) se convierten en un solo índice. Se hará mención indistinta del par (p, q) como $c_i \in \mathcal{C}$ o simplemente i.

En segundo lugar, para reescribir de una manera más cómoda la restricción (2.3b) se indexará de otra manera el vector h. A priori se tiene que el vector h tiene dos índices, sin embargo no se podría decir que es una matriz pues el segundo índice es $r \in \mathcal{R}_i$ y la cantidad de rutas para cada par origen-destino puede ser distinta. Si llamamos $r_i = |\mathcal{R}_i|$ y $\bar{r} = \sum_{i \in \mathcal{C}} r_i$ se puede pensar a h como un elemento de $\mathbb{R}^{\bar{r}}$, donde las primeras r_1 componentes son los flujos de las rutas $r \in \mathcal{R}_1$, las siguientes r_2 componentes son los flujos de las rutas $r \in \mathcal{R}_2$ y así sucesivamente. De ahora en adelante se considerará a h como un vector con la estructura ya mencionada. Asimismo se hará referencia a las rutas como si estuvieran indexadas de la misma manera que h.

En este punto es conveniente introducir la matriz Γ:

Definición 2.3.1 (Matriz Γ) *Sea $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$ un grafo dirigido representando una red de transporte, g una demanda de esa red y $\mathcal{R}_i$ las rutas para el par $i \in \mathcal{C}$. Se define la* Matriz Γ *como una matriz en $\mathbb{R}^{|\mathcal{C}| \times \bar{r}}$ tal que:*

$$\Gamma_{ij} = \begin{cases} 1 & \textit{si la ruta } j \textit{ pertenece al conjunto } \mathcal{R}_i \\ 0 & \textit{en todo otro caso} \end{cases}$$

Con esta nueva indexación de h y la defínción de Γ se puede reescribir la restricción (2.3b) sencillamente como $\Gamma h = g$.

Emulando estas consideraciones sobre los índices p, q y r para la matriz Δ, se la escribirá como una matriz en $\mathbb{R}^{\bar{r} \times |\mathcal{A}|}$. Sus elementos serán entonces de la forma:

$$\delta_{ij} = \begin{cases} 1 & \text{si la ruta } i \text{ contiene la arista } j \\ 0 & \text{en todo otro caso} \end{cases}$$

Con esta nueva forma de escribir Δ la restricción (2.3d) se puede escribir sencillamente como $\Delta^T h = v$.

Por último, si se reemplaza la restricción (2.3d), escrita como $\Delta^T h = v$ en la función objetivo (2.3a) se tiene que se puede pensar lo que originalmente era un problema que minimizaba sobre v como un problema que minimiza sobre h. Luego el problema (2.3) es equivalente al siguiente problema:

$$\begin{aligned} \text{minimizar} \quad & T(\Delta^T h) \\ \text{sujeto a} \quad & \Gamma h = g \\ & h \geq 0, \end{aligned} \tag{2.4}$$

y esta será la formulación que se utilizará para diseñar el recorrido de la nueva ruta. Para ello, el grafo que se considerará estará compuesto por el sector de la ciudad elegido y las líneas de colectivo cuyos recorridos estén contenidos en ese sector. La determinación de la demanda (los pares origen destino) se explicará en la sección 3.3.

Con esa demanda se averiguará primero sobre un sub-grafo, formado sólo por los recorridos de colectivo actual, cuál es el valor de la función objetivo en el equilibrio del usuario de Wardrop. En segundo lugar, se averiguará sobre todo el grafo (ciudad y líneas existentes) cuál es la distribución de flujos que satisface la demanda. En este caso, se definirá una función de costo que penalice que un individuo se desplace "solo" por el grafo de la ciudad. Esta penalización está inspirada en el siguiente razonamiento: En la situación actual, como se vió en la sección 1.1, algunos usuarios se ven obligados a ir hasta el centro antes de hacer un transbordo que los lleve a su destino. La nueva ruta debe ofrecer un transbordo que sea más directo. Al resolver el problema del equilibrio del usuario de Wardrop sobre un grafo compuesto por los recorridos más las calles de la ciudad, se espera que las rutas encontradas contengan algunos arcos de las rutas de colectivos, y otros arcos de las calles de la ciudad. En algún sentido, sería equivalente a que alguien en lugar de hacer el transbordo en el centro, lo haga en una parada anterior. Ahora bien, en esa situación, el equilibrio del usuario de Wardrop dictará que cada usuario busque individualmente cuál es el mejor camino, pero lo que se busca es diseñar una ruta, por eso se penalizará que los usuarios se desplacen "solos" para forzar que los flujos que se encuentren sobre el grafo de las calles represente el camino en el que irá la nueva línea.

Capítulo 3

Obtención de Datos

Teniendo definido el modelo (2.4), es necesario construir las matrices Δ y Γ, definir la función $T(v)$ y elegir los pares origen-destino y sus demandas para obtener g. En este capítulo se explicará cómo se obtuvo la información real de la ciudad de Córdoba y su sistema de transporte para construir las matrices Δ y Γ y qué criterios se utilizaron para elegir los pares origen-destino y construir g.

3.1. El grafo de la ciudad

En primer lugar, fue necesario contar con el grafo del sector de la ciudad sobre el que interesa considerar el problema.

Para decidir cómo son los arcos y los nodos del grafo con respecto al trazo real de la ciudad se hicieron algunas consideraciones. Sobre este grafo se van a buscar luego todas las rutas posibles entre los distintos pares origen-destino. Cada una de estas rutas debería representar una manera que tiene un individuo de ir desde su origen hacia su destino. Cada una de esas maneras está compuesta por algunos tramos realizados a pie y otros realizados en colectivo. Ahora bien, los individuos no recorren la ciudad de la misma manera si lo hacen a pie o en colectivo. Es decir, el colectivo debe respetar los sentidos de las calles y los giros permitidos en las distintas esquinas, mientras que el peatón puede recorrer la ciudad de manera prácticamente arbitraria.

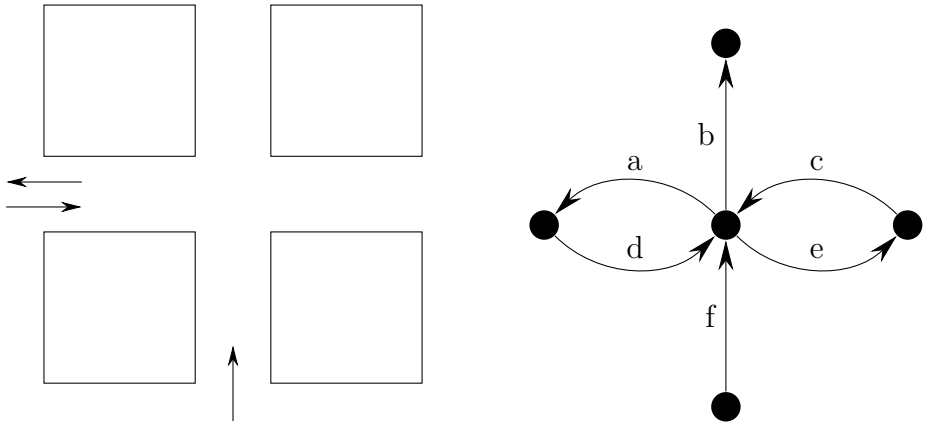
Para simplificar esta situación, el grafo sólo representará las calles, sus sentidos y las prohibiciones de ciertos giros desde el punto de vista de cómo son recorridas por los colectivos.

Esta simplicación hace que se dejen de considerar las partes del camino que son recorridas a pie. En la sección 3.3 se explicará como se lidió con este problema.

Habiendo clarificado que va a representar el grafo, se buscó la forma de construirlo de manera tal que describa los aspectos de la ciudad que se desean considerar.

El enfoque intuitivo sobre cómo construir un grafo que represente las calles de una

ciudad sería asignarle a cada esquina un nodo y a cada cuadra una arco. Eligiendo apropiadamente la orientación de los arcos, se puede lograr representar las calles y sus sentidos. Sin embargo este enfoque no contempla la situación que se da, por ejemplo, en algunas esquinas de calles doble mano en las que esta prohbido doblar a la izquierda. En la figura 3.1(a) se da un ejemplo de esta situación.



(a) Diagrama de la ciudad (b) Grafo representativo

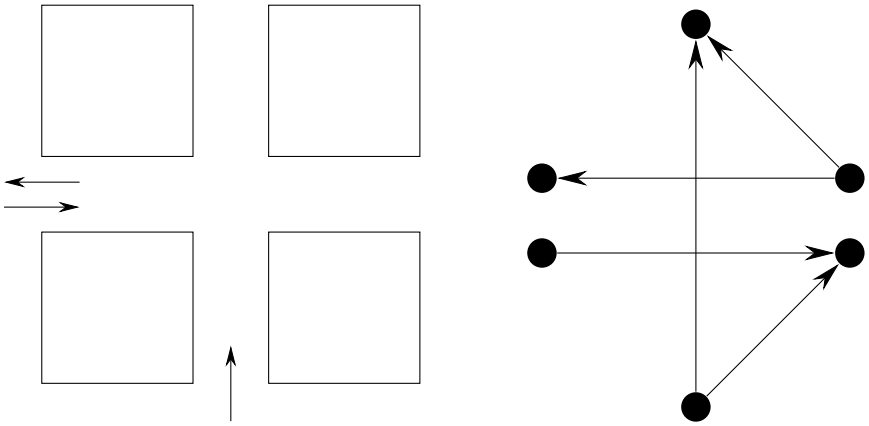
Figura 3.1: Problema de usar el enfoque intuitivo

Como se observa en 3.1(b), no existe ninguna restricción para que una ruta contenga por ejemplo los arcos d y b o incluso los arcos d y a, ambos en ese orden. Pero si se observa 3.1(a) se ve que recorrer esos arcos en ese orden equivale a girar a la izquierda en una calle doble mano o a girar en U, respectivamente. Ambas son maniobras que no están permitidas.

Para salvar estos inconvenientes se decidió asignar un nodo a cada sentido de circulación de cada cuadra, y un arco a cada "forma directa" que exista de ir entre dos cuadras dadas. Esto se explica mejor con el ejemplo de la figura 3.2. Allí se observa como la estrategia propuesta para representar la ciudad respeta tanto los sentidos de las calles, como los giros permitidos en esa esquina.

Una ventaja de este enfoque para representar la ciudad es que las paradas de los colectivos van a estar ubicadas precisamente en los nodos del grafo. De hecho, se pueden pensar todas las paradas de colectivos que haya en una cuadra como una sola parada, y en el grafo serán representadas por un único nodo.

Teniendo claro cuál era la forma que se deseaba que tuviera el grafo, se necesitaba encontrar alguna forma de generarlo. Se encontró una herramienta de Google llamada My Maps que permite colocar sobre los mapas de Google elementos llamados "puntos de referencia", "línea" y "ruta". Más aún, se encontró que Google brindaba la posibilidad de exportar esos elementos colocados sobre el mapa en un archivo en formato `kml`. Se vió que este formato no era más que una variedad del formato `xml` y que dentro de los archivos `kml` se encontraban las coordenadas de los



(a) Diagrama de la ciudad (b) Grafo representativo

Figura 3.2: Estrategia propuesta para construir el grafo

elementos colocados sobre el mapa.

Aprovechando esta herramienta, se construyó con My Maps el grafo con la estructura como el de la figura 3.2(b). Como se dijo en la sección 1.2, sólo se necesitaba definir el grafo para una porción de la ciudad que incluyera a las dependencias de la UNC más algunas cuadras a la redonda. Se comenzó marcando las distintas dependencias de la UNC y los arcos correspondientes a una vecindad de hasta 5 cuadras de distancia de cada una. A continuación se definió una "cápsula" que contuviese a todas esas vecindades, y por último se rellenaron todos los arcos contenidos dentro de esa trayectoria. Para la construcción de la cápsula mencionada se tuvo como criterio que en la frontera del grafo no hubiera nodos fuente (que fuera nodo inicial de todos los arcos que lo tuvieran como nodo extremo) ni sumidero (que fuera nodo final de todos los arcos que lo tuvieran como nodo extremo). Esto se decidió así para satisfacer la hipótesis del teorema 2.2.2 de red fuertemente conectada.

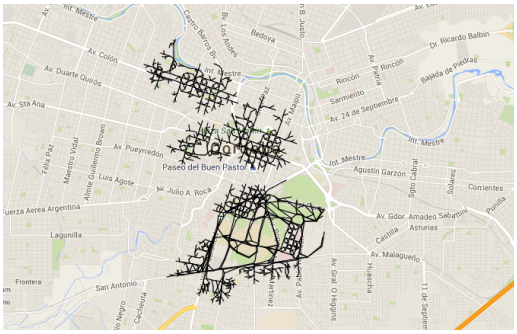
Se eligió utilizar el elemento *línea* para crear uno por uno los arcos. Se notó que, en el archivo `kml` que se podía exportar, el orden en el que aparecían los extremos de cada línea era el orden en el que habían sido definidos. Por esta razón se tuvo especial cuidado en definir cada línea respetando el sentido del arco que representaba.

Además de lo tedioso de la tarea de dibujar uno por uno todos los arcos del grafo, My Maps tiene la restricción de un máximo de 10 capas por mapa. Esto fue un problema pues se habían ido utilizando capas nuevas para nuevos conjuntos de arcos en función de la ubicación de éstos. Tener múltiples capas permitía poder etiquetarlos por color más fácilmente. Este coloreado se realizó para disminuir la posibilidad de pasar por alto algún arco en zonas del mapa donde había particularmente muchos de ellos.

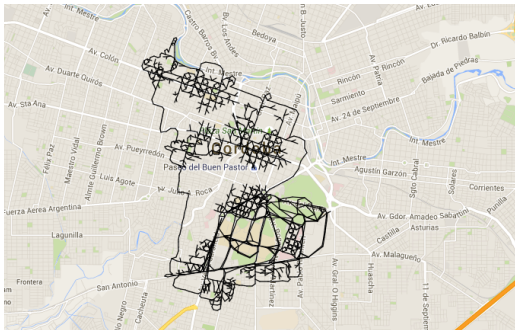
Esta dificultad se sorteó de la siguiente manera: Cada vez que se llegaba a las 10 capas, se exportaba el grafo parcial en un archivo `kml`, se lo editaba para que todos

los elementos pertenecieran a una única capa y se lo importaba en un mapa nuevo. Curiosamente si bien no había un límite de elementos por capa mientras se editaba el mapa, si había un límite de 2000 elementos para ser importados a un mapa por archivo, independientemente del número de capas que hubiera (siempre que fuera una cantidad menor o igual a 10). El grafo terminado tiene más de 3000 arcos, con lo que en las últimas etapas de confección del grafo esto fue otro problema y se tuvo que importar el grafo en dos archivos cada vez.

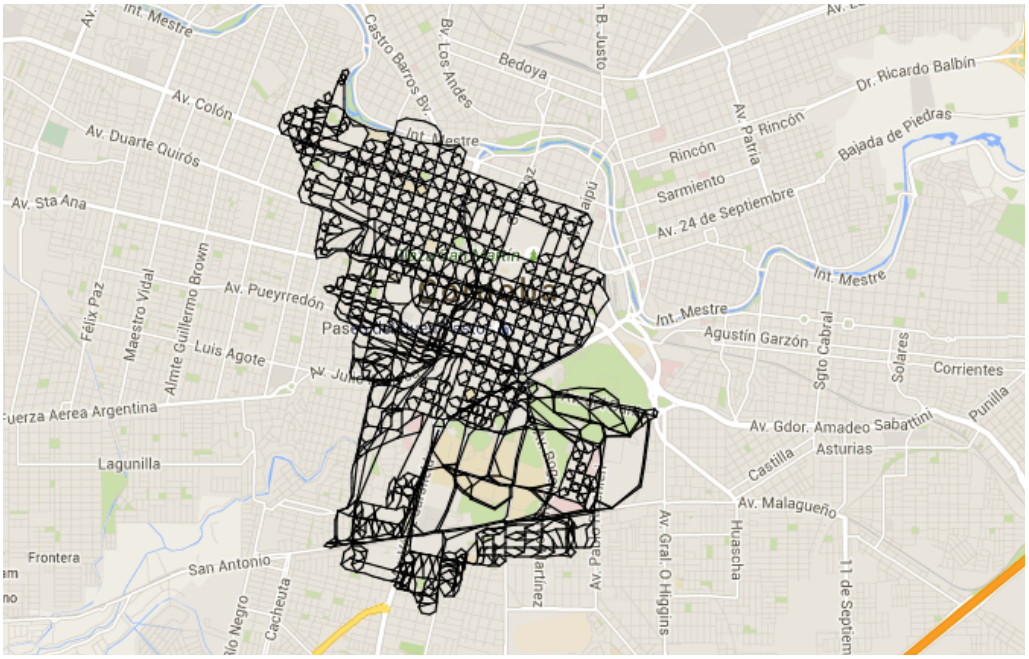
En las figuras 3.3(a) a 3.3(c) se ven el sector original elegido y las distintas etapas de construcción del grafo, hasta la versión final.



(a) Alrededores de nodos de la UNC



(b) Cápsula de las vecindades



(c) Grafo terminado

Figura 3.3: Etapas de construcción del grafo

3.2. Las líneas de colectivo existentes

Teniendo definido el grafo que representa la porción de la ciudad que se va a estudiar, se necesitó describir las líneas de colectivos existentes.

Naturalmente, sólo interesan las partes de cada línea que están incluidas dentro de la zona que se va a estudiar. De todas formas, como la zona incluye al centro de la ciudad, prácticamente todas las líneas existentes tenían alguna parte incluida dentro del grafo. No obstante, había algunas excepciones como las líneas barriales, que no fueron cargados. Tampoco se cargaron los recorridos anulares por no estar completamente incluidos en la zona elegida.

Se decidió que lo único que interesaba de cada línea eran las paradas que tuvieran dentro del grafo y el orden en el que eran recorridas. Esto se pensó así pues sólo es posible bajarse o subirse a un colectivo en las paradas pertenecientes a su recorrido, y por ende no interesaba conocer cada uno de los nodos que visitaba cada línea.

Para cargar las líneas se decidió generar para cada una un pequeño archivo de texto plano con el siguiente formato:

```
i_11 i_12
i_21 i_22
...
i_n1 i_n2
```

donde cada fila contenía dos números, correspondientes a los índices de dos arcos del grafo. Cada uno de esos pares de arcos fueron elegidos de manera que tuvieran un nodo extremo en común y ese nodo extremo fuera precisamente una parada de esa línea. Las paradas así descriptas estaban además ordenadas en el archivo en el orden en el que son visitadas por la línea de colectivo.

Para obtener los recorridos de cada línea de colectivos de la ciudad de Córdoba se utilizó el portal web Mi Autobus[1]. Dicho portal contiene los recorridos de todas las líneas de colectivos de la ciudad y ofrece la posibilidad de mostrar cada uno de ellos sobre un mapa, indicando además las paradas correspondientes.

En el proceso de cargar las líneas se dió varias veces la situación de que algunas líneas compartían dentro del grafo exactamente las mismas paradas. También se dieron casos de líneas cuyo conjunto de paradas estaba contenido dentro del conjunto de paradas de otra línea. En el primer caso se decidió declarar ambas líneas como una sola y en el segundo declarlas como una sola, eligiendo aquella que tuviera más paradas.

Todos los recorridos que se cargaron, y el detalle de que línea/s del sistema de transporte representa fueron siendo registradas en un archivo separado en un formato más comprensible (Ver el apéndice (B.1)). Esto permitió que cada vez que se iba a cargar una línea nueva, bastara una revisión rápida de los recorridos de

[1] http://www.miautobus.com

las líneas que ya se habían cargado para determinar si era efectivamente una línea nueva o no.

A continuación se muestra como ejemplo el contenido del archivo `linea_1`:

```
1741    1927
374     373
2279    351
2630    2634
235     1958
222     1768
```

En total se cargaron 91 archivos de recorridos que representan 127 líneas de colectivos de la ciudad de Córdoba.

3.3. La determinación de los pares origen-destino

A continuación se necesita determinar los pares origen destino y sus respectivas demandas.

Inicialmente se contó con la lista de domicilios de todos los estudiantes de la UNC, proporcionados por el sistema Guaraní. Si bien esta lista no representa el 100 % de los habitantes de la ciudad que necesitan ir a alguna dependencia de la UNC, es una proporción importante.

Como se dijo en la sección 3.1, el grafo que se construyó sólo describe las calles de la ciudad desde el punto de vista de cómo son recorridas por los colectivos. Sin embargo, los individuos realizan una parte de sus desplazamientos en colectivo y otra parte a pie, recorriendo arbitrariamente el grafo. En este punto es conveniente estudiar los tipos de caminos que habrá.

Se considerará entonces que los caminos se componen por tramos realizados a pie y tramos realizados en colectivo, en alguna proporción y orden. Se utilizará el criterio de que la caminata, tanto desde el origen hacia la parada más próxima como el de la última parada hasta el destino, sea de a lo sumo tres cuadras. Dicho criterio surge del análisis que existe en la literatura de que el área de cobertura del sistema de transporte es toda el área de la ciudad que se encuentre a 300 metros o menos de una parada de colectivo. Este criterio se menciona por ejemplo en [2] y en [6]. Se considerará que no hay caminata entre el viaje por línea nueva y dependencias de la UNC ni entre paradas existentes y nuevas pues será una restricción del modelo que las paradas de la línea nueva sólo puedan ser definidas en esos lugares. Los tipos de camino serán:

- Tipos de Camino de casa a UNC:
  - Camino compuesto por: Origen → caminata → parada existente → viaje en colectivo por línea existente → parada existente → caminata → destino

- Camino compuesto por: Origen → caminata → parada existente → viaje en colectivo por línea existente → parada existente → viaje en colectivo por línea nueva → destino
- Camino compuesto por: Origen → caminata → parada existente → viaje en colectivo por línea nueva → destino

- Tipos de Camino de UNC a casa:
  - Camino compuesto por: Origen → caminata → parada existente → viaje en colectivo por línea existente → parada existente → caminata → destino
  - Camino compuesto por: Origen → viaje en colectivo por línea nueva → parada existente → viaje en colectivo por línea existente → parada existente → caminata → destino
  - Camino compuesto por: Origen → viaje en colectivo por línea nueva → parada existente → caminata → destino

- Camino de UNC a UNC:
  - Camino compuesto por: Origen → viaje en colectivo por línea nueva → destino

En el caso de los caminos de Casa a UNC, se supondrá que el viaje se inicia en una parada existente y se buscarán las líneas que terminen en alguna parada que esté a lo sumo a 300 metros del destino. En el caso de los caminos de UNC a casa se buscarán las líneas que comiencen en paradas a lo sumo a 300 metros del origen y se supondrá que el viaje termina en una parada existente. Por como será la restricción de la ubicación de las paradas de la línea nueva, no habrá caminata en los viajes entre dependencias de la UNC. De esta forma se elimina el problema de tener que considerar los tramos de cada viaje que son realizadas a pie por los usuarios del sistema de transporte. En la sección 4.2 se explicará como se adaptaron los algoritmos de búsqueda existentes para ajustarlos a estas consideraciones.

Un invonveniente que se tuvo fue que los domicilios de los estudiantes no estaban clasificados por unidad académica, con lo cual no era posible saber a que dependencia de la UNC necesitaban dirigirse. Se pensó en asignarles dependencias aleatorias a cada uno, manteniendo la proporción de la matrícula que existe entre las distintas unidades académicas. También se pensó que como no se contaba con información de viajes entre dependencias de la UNC, se considerarían también algunos pares origen-destino generados aleatoriamente entre algunas dependencias de la misma facultad, por ejemplo:

- FaMAF y el OAC,
- FCEFyN sede ciudad universitaria y sede centro,

- FCM y Hospitales Escuela,
- FAUDI sede ciudad universitaria y sede centro.

En este punto se encontró la dificultad de que los datos que fueron provistos por el sistema Guaraní eran los domicilios de los estudiantes como habían sido cargados por ellos mismos al momento de completar su ficha de información personal por internet. Por ese motivo ocurrío que había domicilios que se encontraban mal cargados (por ejemplo, los campos "calle" y "número" no contenían la información correspondiente) y bastantes domicilios que habían sido cargados de diferentes maneras (por ejemplo el nombre de la calle a veces aparecía como "Av Vélez Sarsfield", y otras veces como "Vélez", etc).

Entonces, dado que habían aproximadamente 100.000 domicilios, no se encontró una forma eficiente de procesarlos. Además, teniendo en cuenta que la distribución de domicilios de estudiantes de la UNC no es fija a lo largo del tiempo, se decidió utilizar un enfoque diferente para construir los pares origen-destino.

Para el caso de los caminos de Casa a UNC, se consideraron por un lado las primeras paradas de cada línea dentro del grafo, y por otro lado todas las dependencias de la UNC, y se generaron todas las combinaciones posibles de pares que tuvieran origen en el primer conjunto y destino en el segundo. Para el caso de los caminos de UNC a Casa se consideraron por un lado todas las dependencias de la UNC, y por otro las últimas paradas de cada línea dentro del grafo, y análogamente, se generaron todas las combinaciones posibles de pares que tuvieran origen en el primer conjunto y destino en el segundo. Por último, para el caso de los caminos de UNC a UNC se generaron todas las combinaciones posibles de pares que tuvieran origen en alguna dependencia de la UNC y destino en otra dependencia.

Con estas consideraciones quedaron unívocamente definidos algunos de los orígenes y destinos de los pares (los correspondientes a las primeras o últimas paradas de las líneas de colectivo dentro del grafo) pero sigue habiendo cierta ambigüedad con respecto a los nodos que se corresponden con las dependencias de la UNC, sobre todo para las que se encuentran en el centro de la ciudad y las que se encuentran en Ciudad Universitaria. Esto es porque estos dos conjuntos de dependencias tienen una distribución geográfica amplia, que contiene muchos nodos del grafo. Cabe recordar que se desea que el conjunto de paradas de la nueva línea contenga paradas en estas ubicaciones. Sin embargo elegir como nodos de dependencias de la UNC a todos los nodos que sean efectivamente dependencias llevaría a que el modelo coloque paradas en todas esas ubicaciones. Esto último no sería conveniente, y se explicará en la sección 4.3 como se lidió con este problema.

Capítulo 4

Construcción del Modelo

A lo largo de este capítulo se explicará cómo se procesó la información que se obtuvo. Para ver los códigos y otros archivos que se mencionan en este capítulo dirigirse a los apéndices A y B.

4.1. La matriz de incidencia arco-nodo

En primer lugar fue necesario extraer la información de las coordenadas de las aristas definidas mediante el elemento ”línea“ en My Maps. Para ello se exportó el mapa en el archivo `grafo-terminado.kml`. La forma general de este archivo se muestra en el apéndice B.2.

Se utilizó la librería `xml.etree.ElemenTree` de Python para generar, a partir de este archivo, uno que consista en sólo las coordenadas de cada arco. Como se ve, las coordenadas de cada nodo extremo de cada arco tienen tres componentes. Por simplicidad se importó esa componente y fue luego ignorada a la hora de procesar esta información en Octave.

La librería mencionada no podía interpretar la segunda línea de este archivo, ya que su sintáxis no se correspondía con la sintáxis xml tradicional. Por eso se quitó la primer línea y la segunda se reescribió como `<kml>`.

Con este ajuste fue posible escribir un script en python `limpiar-kml.py` que devuelva un archivo conteniendo sólo las coordenadas de cada arista. Se ejecutó entonces:

```
$ python3 limpiar-kml.py > aristas_limpio
```

y se ubtuvo el archivo `aristas_limpio`, que contiene una matriz de dimensión 3071×6, donde 3071 es la cantidad de aristas del grafo. Con esta matriz se ejecutó en Octave la función `genera_nodos('aristas_limpio')` y se generaron tres archivos:

- `indice_aristas`: un archivo con una matriz de dimensión 3071×2, donde cada

fila contiene los índices de los nodos inicial y final de esa arista, respectivamente.

- `nodos`: un archivo con una matriz de dimensión 1808×2, donde 1808 es la cantidad de nodos del grafo. En cada fila están las coordenadas del nodo en el mapa.

- `aristas_redondeado`: la misma matriz del archivo `aristas_limpio` salvo que con menor cantidad de dígitos en cada coordenada (pero aún suficientes) para que sea manipulable después.[1]

Luego, se corrió en Octave la función `genera_matriz_A('indice_aristas')` y se generó el archivo `matriz_A`, que contiene una matriz de dimensión 6142×3. Este arreglo no es en realidad la matriz A de la definición 2.1.4, sino que cada fila de este archivo contiene en las primeras dos componentes una posición en la matriz A y en la tercera el valor en esa posición. Se eligió este formato pues A es una matriz rala, y de esta forma se ahorraba espacio de almacenamiento y en memoria. Para construir efectivamente la matriz A se debe ejecutar en Octave

```
> sparse(matriz_A(:,1),matriz_A(:,2),matriz_A(:,3));
```

Ahora bien, lo que se tiene hasta ahora es la matriz de incidencia arco-ruta del grafo del sector de la ciudad que se había seleccionado. Este grafo no contiene la información de las líneas de colectivos existentes. Para agregar ésta información se agregarán a este grafo nodos y aristas que representen los recorridos y paradas de las líneas de colectivos existentes. En este caso, el enfoque intuitivo sobre cómo agregar los recorridos sería agregando aristas que conecten los nodos de las paradas, en el orden en el que son visitadas. Sin embargo, con ese enfoque se podrían hacer transbordos arbitrarios en todas las paradas que sean compartidas por más de una ruta. En la figura 4.1 se ve un ejemplo de esta situación.

Si las aristas c y d pertenecen al grafo de las calles de la ciudad, el nodo p es una parada, las aristas a y b pertenecen a una línea que tiene parada en p y las aristas e y f pertenecen a otra línea que también tiene parada en p, esta configuración de nodos y aristas tiene el problema de que una ruta para algún par origen destino podría realizar un transbordo en el nodo p sin incrementar el costo de realizar ese transbordo. Esta situación no es deseable, pues en la realidad realizar un transbordo tiene un costo en tiempo (esperar el otro colectivo) y a veces monetario (se debe pagar otro pasaje). Se desea que este costo esté contemplado dentro del modelo y con esta disposición de nodos y aristas no es posible.

[1] Para obtener la lista de nodos se utiliza sobre las coordenadas de los extremos de las aristas una función de Octave que elimina repeticiones. El archivo que se exportó desde Google contiene algunas coordenadas que se corresponden con el mismo nodo pero difieren en menos de 10^{-14} grados. Por este motivo fue necesario redondearlos para que la función de Octave las reconozca como la misma coordenada.

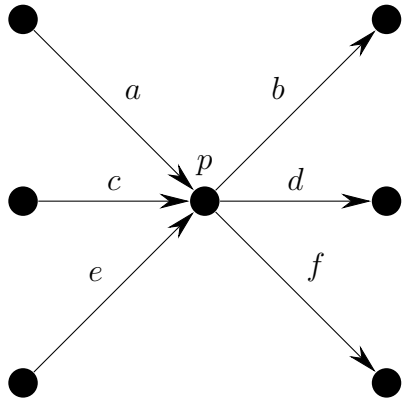


Figura 4.1: Problema de usar el enfoque intuitivo

Para resolver dicho problema se pensaron dos alternativas. En ambas se decidió agregar aristas con costo de transbordo. Estas aristas de transbordo tendrán un costo muy alto, lo que hará que en el equilibrio de Wardrop los usuarios tengan preferencia por realizar la menor cantidad de transbordos posibles. En las figuras 4.2(a) y 4.2(b) se ven ambas alternativas. Las aristas de transbordo se dibujaron más gruesas para clarificar que tienen un costo mayor.

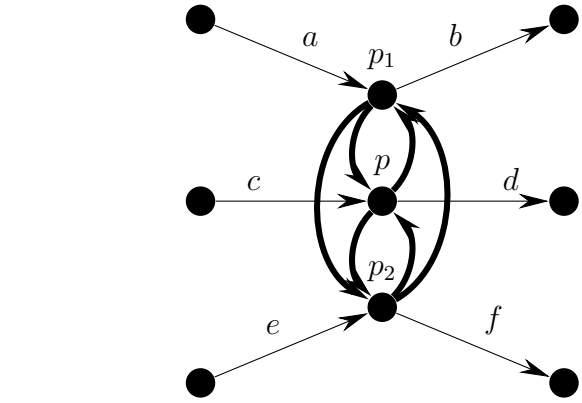
En las alternativa de la figura 4.2, nuevamente c y d son aristas de las calles y a y b, y e y f son aristas de dos líneas que tienen parada en el nodo p. En la alternativa de la figura 4.2(a) se agrega un nodo de parada por cada línea (p_1 y p_2), y se agregan aristas de transbordo en todas las combinaciones posibles.

En la alternativa de la figura 4.2(b) se reemplaza el nodo p por una arista de transbordo (t_1, t_2), y se agregan aristas de costo cero representando las decisiones que puede tomar un usuario en cada momento, de seguir en el colectivo, o realizar un transbordo. Las aristas de costo cero se dibujaron con línea de puntos.

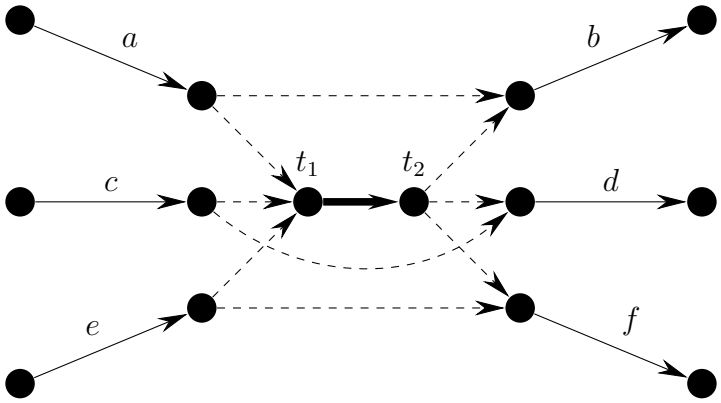
Para elegir una de estas dos alternativas se decidió contar cuántos nodos y aristas se agregaban por línea por parada en cada caso. En el primer caso, se agrega un nodo por línea por parada y se agregan $2 \cdot \binom{n+1}{2}$ aristas en cada parada compartida por n líneas. En el segundo caso, se agregan $2n+3$ nodos y $3n+4$ aristas respectivamente.

cant. líneas	Alternativa 1		Alternativa 2	
	nodos nuevos	aristas nuevas	nodos nuevos	aristas nuevas
1	1	2	5	7
2	2	6	7	10
3	3	12	9	13
4	4	20	11	16
5	5	30	13	19

Tabla 4.1: Comparación de nodos y aristas agregados en cada alternativa



(a) Alternativa con varias aristas de transbordo



(b) Alternativa con una arista de transbordo

Figura 4.2: Alternativas para modelar el transbordo

En la tabla 4.1 se ve una comparación entre nodos y paradas generados por cada alternativa. Allí se puede ver que como la primer alternativa depende de un número combinatorio crece mucho más rápido. Por este motivo se decidió utilizar la segunda alternativa para agregar la información de las líneas de colectivo existentes al grafo de las calles.

Primero, por una cuestión de comodidad, se decidió combinar en un sólo archivo los 91 archivos con información de líneas. Para ello se corrió `genera_rutas(91)` y se generó un archivo llamado `lineas`, que contiene una matriz de dimensión 23×182 que es simplemente una concatenación de los pares de columnas que hay en cada uno de los 91 archivos `linea_n`.

Por último, se corrió en Octave `genera_matriz_A_2('lineas','matriz_A')` y se generaron 6 archivos:

- `paradas`: un archivo que contiene una matriz de dimensión 1×327, donde 327 es la cantidad de paradas que hay en total en el grafo. Cada entrada es el índice del nodo sobre el que está la parada.

- `indice_paradas`: un archivo que contiene una matriz de dimensión 91×22 donde 91 es la cantidad de líneas y 22 es la cantidad máxima de paradas. Cada fila contiene los índices de las paradas de esa línea, según el archivo `paradas`, en el orden en el que son recorridas y completadas con ceros de ser necesario.

- `nodos_paradas`: un archivo que contiene una matriz de dimensión 327×4, donde 327 es la cantidad de paradas. Cada fila se inicia con el índice del nodo en el que se encuentra la parada en el grafo (p), y a continuación los índices de los tres nodos que genera para cada parada, i_1, i_2 e i_3. Donde la arista $i_1 \to i_2$ es la arista de transbordo. Además, las aristas que tenían origen en p ahora tienen origen en el nodo i_3.

- `nodos_lineas`: un archivo que contiene una matriz de dimensión 91×46, donde 91 es la cantidad de líneas y $46 = 2 \cdot 22 + 2$ siendo 22 la cantidad máxima de paradas. Cada fila contiene los índices de los nodos que genera para cada línea. El primer nodo es el nodo de inicio de la línea, y se conecta al segundo. A partir de ahí en cada parada hay dos nodos, conectados a los nodos de las paradas como se ve en la figura 4.2(b). El último nodo es el nodo de fin de línea y se conecta al de la última parada. La primera y la última arista generadas tienen costo cero, además de las que se explican en la figura 4.2(b).

- `matriz_A_2`: un archivo que contiene una matriz de dimensión 18012×3 organizada análogamente a la del archivo `matriz_A`, para construir la verdadera matriz con `sparse`.

- `reg_A`: un archivo que contiene una matriz de dimensión 1×4 donde cada entrada representa la cantidad de aristas que tenía el grafo a medida que se las iba agregando en los distitos pasos.

La función `genera_matriz_A_2` utiliza la función auxiliar `nodo.m` que dados los índices de dos aristas que compartan un nodo, devuelve el índice del nodo que comparten. Todos los archivos que se generaron serán utilizados en otras funciones.

El archivo `matriz_A_2` contiene la matriz de incidencia arco-nodo del grafo que se utilizará, la cual contiene las calles y las líneas de colectivos con sus paradas y aristas de transbordo modeladas como se ve en la figura 4.2(b).

4.2. Rutas y las matrices Δ y Γ

Ahora se necesitan encontrar las rutas posibles para cada par origen destino, para cada variante del grafo que se utilizará. A lo largo de ésta sección no se hará hincapié en ninguna de las variantes en particular, sino que se buscará encontrar

una solución general al problema para que pueda ser utilizada en cualquiera de los grafos en los que se necesite.

En primer lugar se tendrá en cuenta que las rutas que se considerarán serán rutas que pasen a lo sumo una vez por cada arista. Es decir, una ruta será un subconjunto de las aristas. Esta es la forma en la que son consideradas en la definición de la matriz de incidencia arco-ruta (Definición 2.1.3). Esto es, más allá de que en la definición de ruta se considera el orden en el que son recorridas las aristas, en la forma de la matriz Δ sólo importa que aristas están incluidas y cuales no en cada ruta.

Pese a que en la matriz Δ no está presente la información de en qué orden son recorridas las aristas que forman parte de cada ruta, si será necesario tener presente eso a la hora de buscar las rutas.

En principio hay $2^{|\mathcal{A}|}$ formas posibles de elegir subconjuntos de aristas, lo que hace necesario encontrar formas eficientes de buscar entre todas esas posibilidades cuáles son efectivamente rutas, y cuáles son las que se corresponden con los pares origen-destino.

La estrategia que se eligió fue la de buscar las rutas de manera sucesiva para cada par origen-destino.

4.2.1. Rutas para un par origen-destino

Consideraremos el problema de buscar todas las rutas posibles entre dos nodos de un grafo, pues encontrar la matriz Δ se reduce a un problema de este tipo para cada par origen-destino.

El primer intento para resolver este problema fue usar un algoritmo de búsqueda en profundidad o DFS (por sus siglas en inglés, Deep First Search). Este algoritmo consiste en ir recorriendo todos los nodos en un determinado orden. El orden que se utiliza es partir del nodo origen, recorrer el grafo tan lejos como sea posible y una vez agotada un camino regresar e intentar otro camino diferente. En principio este algoritmo tiene como entradas un grafo $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$ y un nodo $n \in \mathcal{N}$, y devuelve todos los nodos de $\mathcal{G}$ a los que puede llegarse desde n.

Este algoritmo puede implementarse de manera recursiva o de manera iterativa. Inicialmente se optó por la manera recursiva, por ser más sencilla de implementar. Un pseudocódigo sería:

```
1 BusquedaEnProfundidad(G,n)
2     etiquetar n como descubierto
3     adyacentes=NodosAdyacentes(G,n)
4     para todo m en adyacentes
5         si m no está etiquetado como descubierto, entonces
6             ejecutar BusquedaEnProfundidad(G,m)
7     devolver descubiertos
```

donde `NodosAdyacentes(G,n)` es una función que busca los nodos a los que se puede llegar en un paso desde n dentro del grafo $\mathcal{G}$. Este pseudocódigo tuvo que ser modificado, primero requiriendo que devuelva un conjunto de rutas, y no un conjunto de nodos. También se agregó el nodo destino y el criterio de que debe llegar a ese nodo para devolver una ruta. El pseudocódigo modificado sería:

```
1  BusquedaEnProfundidad(G,n,m)
2      si n=m, entonces
3          devolver ruta vacía
4      si no
5          adyacentes=NodosAdyacentes(G,n)
6              para todo k en adyacentes
7                  G_aux=QuitarArista(G,n,k)
8                  rutas_aux=BusquedaEnProfundidad(G_aux,k,m)
9                  para toda ruta en rutas_aux
10                     ruta=AgregarArista(ruta,n,k)
11                 agregar rutas_aux a rutas
12     devolver rutas
```

donde `QuitarArista(G,n,k)` es una función que toma un grafo $\mathcal{G}$ y una arista (n, k) y devuelve el grafo que resulta de quitar del grafo $\mathcal{G}$ la arista (n, k). La función `AgregarArista(ruta,n,k)` toma una `ruta` que se inicia en el nodo k y devuelve la ruta que resulta de agregarle a `ruta` la arista (n, k).

La implementación realizada también cuenta la cantidad vértices que se recorren y si supera una cantidad máxima, devuelve la ruta vacía. Esto es para acelerar el proceso de buscar rutas y además ahorrarse devolver rutas que se alejen mucho de su destino. También se incluyó que la función `QuitarArista(G,n,k)` no sólo quitara la arista (n, k) sino también el nodo n, para evitar pasar dos veces por el mismo nodo.

Si bien la implementación recursiva encontraba todas las rutas hasta un largo determinado, era muy lenta. Además, Octave tiene un límite a la profundidad de recursión. Es decir, a la cantidad de veces que una función se puede llamar a sí misma. Y aunque este límite puede ser modificado, Octave no es rápido para ejecutar algoritmos recursivos. Por este último motivo se decidió implementar la versión iterativa del algoritmo de búsqueda en profundidad. El pseudocódigo de la versión iterativa es:

```
1  BusquedaEnProfundidad(G,n)
2      sea P una pila
3      P=Agregar(P,n)
4      mientras P no esté vacía
5          (P,n)=Quitar(P)
6          si n no fue etiquetado como descubierto
7              etiquetar n como descubierto
8              adyacentes=NodosAdyacentes(G,n)
```

```
9               para todo m en adyacentes
10                  P=Agregar(P,m)
11      devolver descubiertos
```

donde `Agregar(P,n)` es una función que toma una lista de nodos P y agrega el nodo n al final de esa lista, y `Quitar(P)` es una función que toma una lista de nodos P y devuelve la lista que resulta de quitar el último nodo de la lista P y el nodo quitado. Este pseudocódigo también tuvo que ser modificado para que devuelva rutas en lugar de nodos descubiertos. El pseudocódigo modificado sería:

```
1  BusquedaEnProfundidad(G,n,m)
2      sea P una pila
3      r=RutaInicial(n)
4      P=Agregar(P,r)
5      mientras P no esté vacía
6          (P,r)=Quitar(P)
7          si r llega a m
8              agregar r a rutas
9          si no
10             adyacentes=NodosAdyacentes(G,r)
11             r_aux=AgregarNodos(adyacentes,r)
12             P=Agregar(P,r_aux)
13     devolver rutas
```

donde la pila P es un conjunto de rutas, `RutaInicial(n)` es una función que toma un nodo n y devuelve la ruta formada por el único nodo n. `Quitar(P)` es exactamente igual al caso anterior y `Agregar(P,r)` funciona análogamente salvo que r es una lista de rutas (puede contener una, varias o ninguna ruta). La función `NodosAdyacentes(G,r)` toma un grafo $\mathcal{G}$ y una ruta r y devuelve la lista de nodos adyacentes al último nodo de la ruta r. Por último, `AgregarNodos` es una función que toma cada nodo en la lista `adyacentes` y comprueba si ya existe en r. Si no existe, construye la ruta que resulta de agregar ese nodo a r. Finalmente devuelve la lista de todas las rutas que construyó, pudiendo ser la lista vacía.

Además de estas modificaciones, se implementó también el conteo de nodos, y se estableció una cantidad máxima de nodos que podía tener una ruta, y si esa cantidad era superada se descartaba la ruta de la pila.

Esta implementación era mucho más rápida, y de hecho se ve en la tabla 4.2 como el tiempo de ejecución del algoritmo recursivo aumentaba de manera prácticamente exponencial al aumentar la distancia entre los nodos del par origen-destino.

Ahora bien, aunque el método iterativo reducía mucho los tiempos para nodos cercanos, para nodos muy alejados los tiempos crecían nuevamente más allá de tiempo razonables. Se estimó que en el grafo completo el promedio de tiempo para encontrar todas las rutas de un par origen destino (de hasta un largo dado) rondaba los $3 \cdot 10^{11}$ segundos (10.000 años aprox).

Distancia (en cant. de aristas)	Método recursivo	Método iterativo
2	0.26	0.12
4	7.15	0.17
6	1013.23	0.49

Tabla 4.2: Tiempo de ejecución (en segundos) de los métodos recursivo e iterativo

Se modificó la implementación y se agregó otro control para descartar aún más rutas. Una vez fijada la cantidad máxima de nodos n_{MAX} que eran aceptables por ruta, si una ruta en la pila con $n < n_{MAX}$ nodos cumplía que su último nodo estaba a una determinada "distancia" del nodo destino de manera tal que no era posible que con los $n_{MAX} - n$ nodos restantes pudiera llegar al destino, era descartada. Con esta modificación se redujo el tiempo promedio a $6{,}5 \cdot 10^8$ segundos (20 años aprox.), un tiempo menor, pero aún lejos de lo razonable para encontrar todas las rutas de un par origen-destino.

Se decidió incorporar un nuevo elemento a la implementación: la búsqueda de una ruta inicial. La idea era encontrar primero una ruta para el par origen-destino, y luego a partir de esa ruta construir una pila como la del algoritmo de búsqueda en profundidad, y a partir de esa pila continuar con la búsqueda usando la búsqueda en profundidad para encontrar más rutas "parecidas" a la ya encontrada. Se pensó acotar esta búsqueda a una cierto tiempo y/o cantidad de nuevas rutas encontradas para tener control sobre el tiempo de ejecución, sin perder la garantía de encontrar al menos una ruta para cada par origen-destino.

El algoritmo que se eligió para buscar esta primera ruta fue el algoritmo de búsqueda A*. Éste es un algoritmo de búsqueda de camino más corto que es una generalización del algoritmo de búsqueda de Dijkstra. Es un método herurístico de tipo mejor búsqueda primero o BFS (por sus siglas en inglés, Best First Search). Dado un par origen-destino, el algoritmo recorre el grafo construyendo un árbol de caminos parciales. Las hojas de este árbol se ordenan según una prioridad determinada por una función de costo más una función heurística. La primera es la suma de los costos de las aristas que se deben recorrer para llegar hasta la hoja y la función heurística estima el costo de llegar al destino. Los nodos ya visitados suelen denominarse el conjunto *cerrado* y las hojas del árbol suelen denominarse el conjunto *abierto*.

Para que el algoritmo funcione, la función heurística debe ser *admisible*, lo que significa que no debe sobrestimar el costo real de llegar al objetivo. Es decir, no debe suceder que exista una ruta que tenga un costo menor que el costo que estima la función heurística. Un pseudocódigo de este algoritmo sería:

```
1 BusquedaAEstrella(G,n,m)
2     h_aux=h(n,m)
3     abierto=Agregar([n h_aux],abierto)
```

```
4       mientras abierto no esté vacío
5           actual=Minimo(abierto)
6           si actual=m
7               terminar
8           cerrado=AgregarC(actual,cerrado)
9           adyacentes=NodosAdyacentes(G,actual)
10          para todo k en adyacentes
11              si k está en cerrado
12                  continuar
13              costo_aux=g(n,k)+h(k,m)
14              si k está en abierto
15                  si costo(k)>costo_aux
16                      costo(k)=costo_aux
17                      viene(k)=actual
18              si no
19                  abierto=Agregar([k costo_aux],abierto)
20                  viene(k)=actual
21      ruta=Reconstruir(viene,actual)
22      devolver ruta
```

donde `h(n,m)` es la función heurística que estima el costo de ir desde n hasta m; `Agregar` es una función que agrega al conjunto `abierto` el par $[n\ costo(n)]$, n es un nodo y $costo(n)$ es la función de costo para ese nodo. `Minimo` es una función que busca el nodo de menor costo dentro del conjunto `abierto`. `AgregarC` es una función que agrega el nodo `actual` al conjunto `cerrado` y a la vez lo quita del conjunto `abierto`. La función `g(n,k)` calcula el costo de llegar hasta la hoja k desde el nodo origen n y `Reconstruir` es una función que reconstruye el camino hacia atras desde el nodo `actual` hasta el nodo origen n usando la lista de nodos predecesores `viene`.

4.2.2. El algoritmo de búsqueda

Ahora bien, lo único que se necesita para implementar este algoritmo es asignarle un costo a las aristas y definir una función heurística. Como en nuestro ejemplo particular el grafo posee una disposición espacial determinada (los nodos tienen coordenadas geográficas que pueden pensarse como coordenadas de $\mathbb{R}^2$) se puede utilizar como costo de las aristas su longitud en alguna norma y como función heurística la distancia en esa misma norma. La desigualdad triangular provista por la norma tendrá como consecuencia que la función heurística así definida será *admisible* y el algoritmo funcionará.

A continuación se hicieron dos cosas: se generó un vector de costo de aristas y se asignaron coordenadas a todos los nodos. Lo primero se hizo de manera tal de poder usar más tarde ese vector de costos para el modelo y no sólo para buscar las

rutas. Lo segundo se hizo para poder calcular la función heurística. La norma que se eligió fue la norma 1, pues es la que más se parece a como son medidas las distancias recorridas en una ciudad. En Octave se corrió:

```
> genera_costo_aristas('reg_A','nodos_lineas', '
aristas_redondeado','indice_paradas','paradas','nodos')
```

y se generó el archivo `costo_aristas` que contiene una matriz de dimensión 9006×1. Se eligió para las aristas de transbordo un costo igual a 10 veces el costo promedio del resto de las aristas de costo no nulo. Luego se corrió:

```
> genera_nodos_ampliados('nodos','indice_paradas','paradas')
```

y se generó el archivo `nodos_amp` que contiene una matriz de dimensión 5239×2 con pares de coordenadas para todos los nodos del grafo. Es decir, tanto para los nodos que pertenecen al grafo inicial de las calles, como para los que fueron creados para modelar las paradas. A los nodos que fueron creados para modelar las paradas se les asignó como coordenadas las coordenadas de las paradas que modelan.

En la implementación final del algoritmo de búsqueeda de rutas se hicieron tres modificaciones más respecto a los pseudocódigos mostrados. Primero, en lugar de utilizar `si actual=m` como criterio de parada, se utilizó como criterio que la distancia entre `actual` y `m` sea menor a 300 metros. Ahora bien, este criterio así enunciado se puede utilizar para los viajes de casa a UNC, pero no funciona para los viajes de UNC a casa, pues los nodos que se consideran dependencias UNC no son necesariamente paradas, y por ende el algoritmo no podría ni siquiera comenzar en el grafo que tiene sólo las rutas. Para resolver esto, en los pares origen-destino correspondientes a los viajes de UNC a casa se realizó la búsqueda "contramano" desde la última parada hasta el nodo UNC, hasta obtener un viaje que se origine a menos de 300 m del nodo origen. Para que el algoritmo implementado pueda identificar cuando realizar la búsqueda "mano" y cuando "contramano", se decidió que los pares origen-destino correspondientes al primer caso serán guardados con sus índices positivos y para el segundo caso con sus índices negativos. Se decidió esto para no agregar una tercera columna al arreglo, y porque no afectaba la información guardada dado que todos los índices son positivos. Entonces la segunda modificación respecto al pseudocódigo mostrado fue permitir que el algoritmo distinga estos casos. La tercera modificación fue simplemente agregar un control de tiempo y de cantidad de rutas encontradas para controlar el tiempo de ejecución.

En definitiva se implementaron 6 funciones para buscar las rutas posibles para un par origen destino dado (Apéndice A). A continuación se las describe brevementes sin dar detalles de su implementación.

- `rutas.m`: Dado un grafo, un par origen-destino, y parámetros concernientes a los costos de las aristas y los tiempos máximos de ejecución y de cantidad de rutas, devuelve las rutas posibles que encuentre entre el origen y el destino.

Para encontrar las rutas primero encuentra una utilizando una búsqueda de tipo A*, y luego a partir de esa ruta encontrada encuentra más rutas con una búsqueda en profundidad. Usa `a_star.m`, `salientes.m`, `entrantes.m`, `dist_n.m` y `arista.m`.

- `a_star.m`: Dado un grafo y un par origen-destino, encuentra el camino más corto entre el origen y el destino utilizando un algoritmo de búsqueda A*. Usa `salientes.m`, `entrantes.m` y `arista.m`.

- `salientes.m`: Dado un grafo y un nodo, devuelve una lista de nodos a los que se puede llegar desde el nodo provisto atravesando exactamente una arista.

- `entrantes.m`: Dado un grafo y un nodo, devuelve una lista de nodos desde los que se puede llegar al nodo provisto atravesando exactamente una arista. Es una variante de `salientes.m` que se utiliza para recorrer el grafo "contramano".

- `dis_n.m`: Dado un grafo, un par de nodos, y un parámetro concerniente a los costos de las aristas, devuelve una estimación de la cantidad de aristas que se necesitan para construir una ruta entre los nodos provistos.

- `arista.m`: Dado un grafo y un par de nodos adyacentes, devuelve la ruta que está compuesta únicamente por la arista que tiene por extremos los nodos provistos.

4.2.3. Las matrices Δ y Γ

Teniendo listo el algoritmo que busca rutas dado un par origen destino, se pudo escribir la función `genera_D_G.m`. Esta función dado un grafo, un conjunto de pares origen-destino y parámetros concernientes a los costos de las aristas y los tiempos máximos de ejecución y de cantidad de rutas, construye las matrices Δ y Γ para un problema como el de (2.2) usando `rutas.m`. La implementación no asume nada sobre el tipo de grafo sobre el que se calcularán las rutas, con lo que sirve para todos los grafos sobre los que sea necesario calcular las matrices Δ y Γ.

Se debió realizar una consideración sobre la rutina `genera_D_G`. Dicha consideración fue permitir continuar la búsqueda aunque no se encontrara ninguna ruta entre un par origen-destino. Esto fue necesario pues en el caso de la búsqueda de rutas sobre el grafo que contiene sólo las líneas de colectivo había pares origen-destino para los que no existía ninguna ruta posible. Los índices de estos pares fueron guardados en un archivo `ignorados` para tener un registro de los mismos. Se tuvo que tener especial cuidado en esta parte pues los algoritmo de búsqueda devuelven listas vacías si no encuentran ninguna ruta. Eso era útil en el interior de las rutinas de búsqueda, pero en esta última etapa eso daba como resultado filas nulas en las matrices, lo

que se traduce como un sistema incompatible en las restricciones, y por ende en un problema infactible.

En la sección 5.4 se utilizará repetidas veces esta función para cada uno de los problemas de la forma (2.2) que se resuelvan. Como dato ilustrativo de los tiempos de ejecución que se lograron luego de todo el trabajo invertido en la confección de la función `rutas` se muestran los tiempos para los dos primeros problemas. El primero consiste en encontrar todas las rutas sobre el grafo que tiene sólo las líneas de colectivos, y el segundo es encontrar todas las rutas sobre el grafo que contiene las líneas de colectivo y las calles de la ciudad.

```
> genera_D_G('g_OD_s_l',A,p,4,1000)
...
Procesado par 2730 de 2730
Transcurrieron 22917 segundos. Tiempo restante estimado: 0
segundos
>

> genera_D_G('g_OD',A,p,4,100)
...
Procesado par 2990 de 2990
Transcurrieron 15006 segundos. Tiempo restante estimado: 0
segundos
>
```

Como se ve, se utilizó un tiempo máximo de busqueda en profundidad por par de 4 segundos, y se impuso una cota máxima de 1000 rutas encontradas por par origen-destino en el primer caso, y de 100 rutas por par origen-destino en el segundo caso. Estas ejecuciones fueron realizadas para probar la velocidad de los algoritmos de búsqueda implementados, pero las matrices Δ y Γ que se generaron no fueron utilizadas después. Se decidió no utilizarlas pues la rutina de minimización demoraba demasiado. Esto se explicará mejor en la sección 5.4.

Otra observación es que en el primer caso hubo 103 pares origen-destino para los que no encontró ninguna ruta. Para cada uno de estos pares, el algoritmo A* tuvo que recorrer prácticamente todo el grafo antes de concluir que no existía ninguna ruta. El segundo algoritmo encontró rutas para todos los pares origen-destino, lo cual era esperado dado que estaba buscando rutas sobre el grafo que contiene las líneas de colectivos y las calles.

4.3. El vector de pares origen-destino

En la sección 3.3 se clasificaron los caminos en tres tipos y se explicó cómo se generarían los pares origen-destino para cada uno. Sin embargo, no se definieron con

precisión los nodos correspondientes a las dependencias de la UNC. Éstos deben ser elegidos con cuidado, pues serán ubicaciones de paradas potenciales de la nueva línea. No deben elegirse tantos de manera que la cantidad de paradas de la nueva línea sea demasiado grande, ni tan pocos que no se logre la cobertura de las dependencias de la UNC.

Para resolver este problema se comenzó identificando todos los nodos que se correspondieran con dependencias de la UNC. Para ésto, se utilizó el grafo que se tenía en My Maps y se describieron los nodos con pares de aristas que lo compartan, como se hizo con las paradas. Se marcaron 41 nodos correspondientes a dependencias de la UNC. Al OAC, el Hospital Nacional de Clínicas y el Hospital Universitario de Maternidad y Neonatología se les asignó un nodo a cada uno. Para las dependencias en el centro de la ciudad se designaron 5 nodos y para las dependencias en ciudad universitaria se designaron los restantes 34 nodos.

Claramente, sobre los tres primeros nodos mencionados no es necesario realizar ningún tipo de selección. La selección se realizará sobre los últimos dos conjuntos. Para ello primero se corrió en Octave:

```
> load nodos
> load paradas
> dlmwrite('paradas_maps',nodos(paradas,:),'     ')
```

y luego se escribió un script en python para generar un archivo `kml` con esas coordenadas para poder importarlas en My Maps. Se ejecutó entonces:

```
$ python3 generar-kml-paradas.py > paradas.kml
```

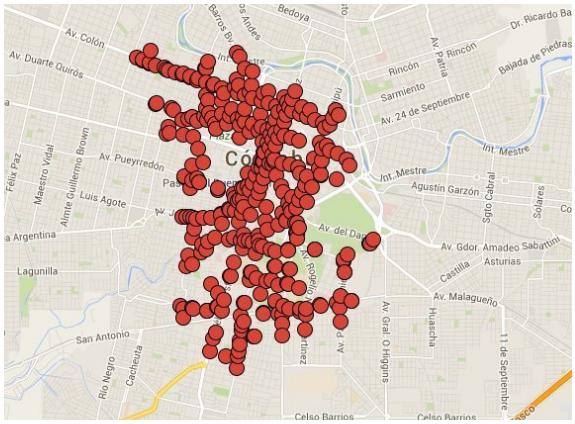
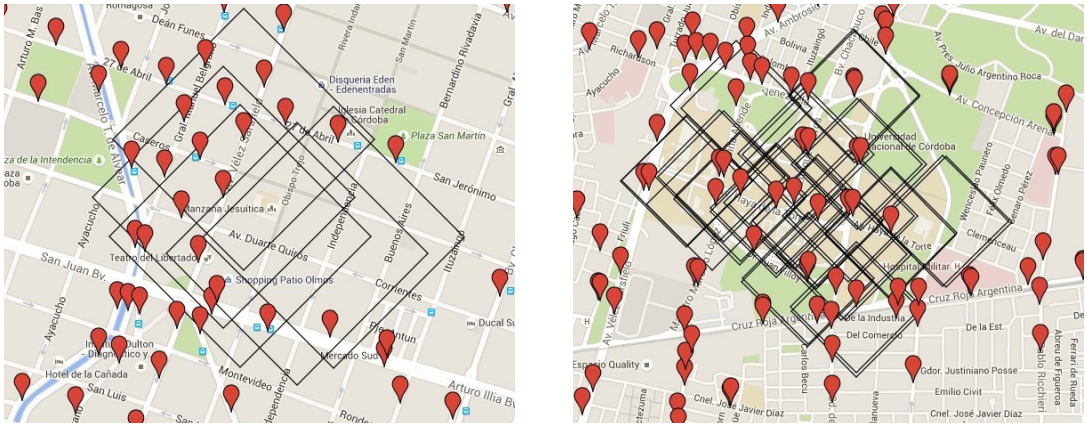


Figura 4.3: Paradas dentro del grafo

El resultado se muestra en la figura 4.3. Luego se decidió que el criterio para seleccionar nodos representativos de éstos dos conjuntos iba a ser elegir la menor cantidad de nodos de manera tal que la unión de las vencidades de los nodos seleccionados cubra las mismas paradas que las de todos los nodos en cada conjunto.

La idea de elegir la menor cantidad de nodos con el criterio mencionado es similar a la de elegir el menor subcubrimiento que siga siendo cubrimiento del conjunto de paradas en consideración. Las vecindades se eligieron como bolas de radio igual a 300 metros en la norma 1, pues esa es la norma que se utilizará luego para encontrar las rutas entre los pares origen-destino. En las figuras 4.4(a) y 4.4(b) se ven las vecindades.



(a) Vecindades en el centro de la ciudad (b) Vecindades en ciudad universitaria

Figura 4.4: Vecindades

Luego de aplicar el criterio mencionado se redujo la cantidad de nodos de dependencias de la UNC de 41 a 15, quedando 3 para el centro de la ciudad y 9 para ciudad universitaria. Estos 15 lugares fueron guardados en el archivo `lugares_UNC` en el mismo formato utilizado para las paradas de las distintas líneas de colectivo. Esto es, en dos columnas donde cada fila tiene los índices de dos aristas que comparten el nodo que se desea describir. El contenido de este archivo es:

```
547     475
85      1
202     127
362     356
363     247
247     248
612     733
722     724
725     723
636     640
660     649
2160    658
668     673
688     683
713     711
```

En este punto, para generar los pares origen-destino fue necesario realizar una consideración más. Los nodos de las dependencias de la UNC no se corresponden necesariamente con paradas, de hecho los del OAC y lo Hospitales Escuelas no se corresponden con paradas. Ahora bien, como se explicó en la sección 2.3, primero se averiguará el valor de la función objetivo sobre un sub-grafo compuesto sólo por los recorridos actuales. En este sub-grafo entonces no existirán nodos para algunas de las dependencias. Esto no generará un problema para los tipos de camino que sean de UNC a casa o viceversa, como se explicó en la sección 4.2, pues uno de los nodos de origen o destino si pertenecerá al sub-grafo. Pero si generará problemas en los tipos de camino de UNC a UNC, pues en este caso ninguno de los nodos de origen o destino formarán parte del grafo. Por este motivo se decidió no considerar los pares origen-destino de los tipos de camino de UNC a UNC para la parte de determinar el valor de la función objetivo en el sub-grafo.

Ahora si, con todas estas consideraciones se generaron dos conjuntos de pares origen destino: uno para el problema con el sub-grafo que contiene sólo los recorridos de los colectivos y otro para el grafo completo. Primero se corrió en Octave:

```
> genera_g_OD('lugares_UNC','nodos_paradas','nodos_lineas',1)
```

y se generó un archivo `g_OD`, que contenía una matriz de dimensión 2730×2. Cada fila de esta matriz contiene los índices de los nodos de origen y destino respectivamente de cada par origen-destino para los tipos de camino de casa a UNC y de UNC a casa. Como después es necesario procesar de manera distinta los pares según tuvieran destino u origen en UNC, los índices de los nodos de los pares del segundo tipo se guardaron con signo negativo para poder diferenciarlos. En la sección 4.2 se explicó porqué es necesaria esta distinción.

Por último, fue necesario agregar los pares origen destino del tercer tipo de camino: de UNC a UNC. Para esto, se usó que se sabe a que dependencias de la UNC pertenecen cada conjunto de nodos en el archivo `lugares_UNC`. Se corrió en Octave lo siguiente:

```
1  > l=load('lugares_UNC');
2  > g=load('g_OD');
3  > l=nodo(l(:,1),l(:,2));
4  > l=l(:);
5  > # pares OD con origen en OAC
6  > origenes=l(1);
7  > destinos=l(2:end);
8  > g1(:,1)=reshape(repmat(origenes,1,rows(destinos))',[],1);
9  > g1(:,2)=repmat(destinos,rows(origenes),1);
10 > # pares OD con origen en hospitales escuela
11 > origenes=l(2:3);
12 > destinos=l(setdiff([1:length(l)],[2 3]));
13 > g2(:,1)=reshape(repmat(origenes,1,rows(destinos))',[],1);
```

```
> g2(:,2)=repmat(destinos,rows(origenes),1);
> # pares OD con origen en el centro
> origenes=l(4:6);
> destinos=l(setdiff([1:length(l)],4:6));
> g3(:,1)=reshape(repmat(origenes,1,rows(destinos))',[],1);
> g3(:,2)=repmat(destinos,rows(origenes),1);
> # pares OD con origen en ciudad universitaria
> origenes=l(7:15);
> destinos=l(setdiff([1:length(l)],7:15));
> g4(:,1)=reshape(repmat(origenes,1,rows(destinos))',[],1);
> g4(:,2)=repmat(destinos,rows(origenes),1);
> g_s_l=g;
> g=[g;g1;g2;g3;g4;-g1;-g2;-g3;-g4];
> dlmwrite("g_OD",g,' ')
> dlmwrite("g_OD_s_l",g_s_l,' ')
```

Y se obtuvieron los archivos `g_OD_s_l` y `g_OD`, donde el primero será utilizado sobre el sub-grafo que contiene sólo las líneas de colectivo y el segundo será utilizado sobre todo el grafo. Nótese que además de encontrar los pares origen-destino para todas las combinaciones posibles de viajes entre dependencias de la UNC se los guardó en dos copias, una con signo positivo y otra con signo negativo. Esto es para que el algoritmo de búsqueda de rutas encuentre rutas entre los distintos nodos de la UNC en ambos sentidos.

4.4. La función $T(v)$

Como se dijo en la sección 2.3, se definirá la función $T(v)$ de manera distinta para resolver el modelo en cada una de las dos versiones del grafo. Una es la que contiene sólo las líneas de colectivo y la otra la que contiene además las calles de la ciudad.

En la primera versión sólo interesa conocer el valor de la función objetivo. Es decir, sólo interesa saber cuál es la situación actual del sistema. En ese sentido sólo se necesita conocer el equilibrio de Wardrop para la red tal cual está. Por este motivo las funciones de costo de las aristas serán constantes y su valor será el costo de recorrerlas.

Para asignarles el costo de recorrerlas se utilizará la norma 1 sobre los extremos de cada arista. Este no será el valor exacto del costo de recorrer esa arista. Por un lado se dijo que lo que se intentará optimizar es el tiempo y no la distancia. Las velocidades a las que se recorre cada arista pueden variar según sea una zona congestionada o según el horario. Sin embargo, como la zona de la ciudad que se está considerando tiene un porcentaje bastante alto de calles céntricas, se puede suponer sin perder demasiada precisión que la velocidad de recorrer las aristas es constante.

Otro problema de utilizar la norma 1 es que tampoco representa exactamente la distancia que debe recorrer un colectivo para ir entre un extremo y otro. Sólo refleja el valor real del largo de la arista cuando ésta tiene direccion Norte-Sur o Este-Oeste, pero se consideró poco consistente cambiar en esta etapa la norma utilizada.

En definitiva las funciones de costo para el primer problema serán:

$$t_a(v_a) = c_a, \ \forall a \in \mathcal{A}$$

Por otra parte, para el segundo problema se va a utilizar el equilibrio de Wardrop para encontrar flujos que describan la ubicación de la nueva línea. Para ésto, se dijo también en la sección 2.3 que se iba a definir una penalización sobre los usuarios que recorran "solos" las calles de la ciudad, con el espíritu de lograr que en la afectación óptima haya flujos nulos en casi todas las calles. Aquellas calles en las que haya flujos no nulos serán las candidatas a ser recorridas por la nueva línea.

Para definir esta penalización se pensaron algunos criterios:

- La función de penalización debe ser suave
- Para flujos pequeños la penalización debe ser grande, y por ende el costo de recorrer esa arista será alto
- Para flujos grandes la penalización debe ser muy pequeña, y por ende el costo de recorrer esa arista será aproximadamente igual al largo de la arista en la norma 1.

Para describir la función que se decidió elegir para cumplir con estos criterios nombraremos las aristas de los dos sub-grafos como $\mathcal{A}_c$ para el caso de solo calles y $\mathcal{A}_l$ para el caso de solo líneas de colectivo existentes. Claramente se cumple $\mathcal{A}_c \cup \mathcal{A}_l = \mathcal{A}$ y $\mathcal{A}_c \cap \mathcal{A}_l = \emptyset$. Con esta notación, la función que se definió fue:

$$t_a(v_a) = \begin{cases} c_a & \text{si } a \in \mathcal{A}_l \\ c_a(1 + e^{-v_a}) & \text{si } a \in \mathcal{A}_c \end{cases}$$

Hasta aquí se han definido las funciones $t_a(v_a)$ para ambos casos, pero la función objetivo es $T(v)$. De la ecuación (2.3a) se ve que no es difícil calcular $T(v)$ si se conocen las funciones $t_a(v_a), \ \forall a \in \mathcal{A}$. Si se denota $C = (c_a)_{a\in\mathcal{A}}$, $C_l = (c_a)_{a\in\mathcal{A}_l}$, $v_l = (v_a)_{a\in\mathcal{A}_l}$ y a las funciones para cada problema $T_1(v)$ y $T_2(v)$ respectivamente se tiene:

$$T_1(v) = C^T v$$

$$T_2(v) = \sum_{a\in\mathcal{A}_c} c_a(v_a - e^{-v_a}) + C_l^T v_l$$

Se supone que cada función está definida en su dominio. Por ejemplo, $T_1(v)$ está definida sobre el grafo que contiene sólo las líneas de colectivo existentes, y por ende el vector v en este caso contiene sólo los flujos de esas aristas y es distinto del vector v que aparece como argumento de T_2. De esa forma se evita tener que definir notación (como v_c, por ejemplo) que no agrega nada para entender que es lo que hace cada función.

Capítulo 5

Problema de Optimización

5.1. Condiciones de optimalidad

En primer lugar se introducirá la terminología básica que se usará a lo largo de este capítulo.

Definición 5.1.1 *Sea una función* $f : \mathbb{R}^n \to \mathbb{R}$. *Se define un* Problema de optimización sin restricciones *al problema de encontrar un vector* $x^* \in \mathbb{R}^n$ *tal que* $f(x^*) \leq f(x)$, $\forall x \in \mathbb{R}^n$ *y se lo denota como:*

$$\begin{array}{rl} \textit{minimizar} & f(x) \\ \textit{sujeto a} & x \in \mathbb{R}^n \end{array}$$

Se dirá que f *es la* función objetivo.

De manera análoga, se define un Problema de optimización con restricciones *al problema de encontrar un vector* $x^* \in \Omega \subset \mathbb{R}^n$ *tal que* $f(x^*) \leq f(x)$, $\forall x \in \Omega$ *y se lo denota como:*

$$\begin{array}{rl} \textit{minimizar} & f(x) \\ \textit{sujeto a} & x \in \Omega \end{array}$$

Dado un $x \in \mathbb{R}^n$ *cualquiera, si cumple* $x \in \Omega$ *se dirá que* x *es* factible. *También se llamará a* Ω *el* conjunto factible.

Para ambos tipos de problema, el mencionado vector x^* *se llama* mínimo global.

Si en cambio sucede que existe un número real $\epsilon > 0$ *tal que el vector* x^* *cumple que* $f(x^*) \leq f(x)$, $\forall x \in \mathbb{R}^n$ *con* $\|x - x^*\| < \epsilon$ *para el caso sin restricciones, o* $f(x^*) \leq f(x)$, $\forall x \in \Omega$ *con* $\|x - x^*\| < \epsilon$ *para el caso con restricciones, se dice que* x^* *es un* mínimo local. *Aquí* $\|\cdot\|$ *denota la norma usal de* $\mathbb{R}^n$.

A continuación se darán varios resultados que serán de utilidad en las secciones 5.2 y 5.3. Algunos de estos resultados tienen versiones más generales, sin embargo

se eligieron estas formulaciones para necesitar una menor cantidad de pruebas y definiciones para poder mostrar los resultados importantes en las secciones siguientes.

Proposición 5.1.2 (Condición necesaria de optimalidad) *Sea $f : \mathbb{R}^n \to \mathbb{R}$ una función continuamente diferenciable y $\Omega \subset \mathbb{R}^n$ un conjunto convexo. Dado el problema:*

$$(*) \begin{cases} \text{minimizar} & f(x) \\ \text{sujeto a} & x \in \Omega \end{cases}$$

Si el punto factible $x^* \in \mathbb{R}^n$ es un mínimo local de $(*)$, entonces:

$$\langle \nabla f(x^*), x - x^* \rangle \geq 0, \ \forall x \in \Omega.$$

Aquí $\langle \cdot, \cdot \rangle$ denota el producto interno usual de $\mathbb{R}^n$ y ∇f denota el gradiente de f. Se dirá que un punto x^* que cumple $\langle \nabla f(x^*), x - x^* \rangle \geq 0, \ \forall x \in \Omega$ es un punto estacionario*.*

El siguiente lema se usará para probar condiciones de optimalidad necesarias más específicas para problemas en los que Ω no sólo es convexo, sino que además está dado por un sistema de igualdades y desigualdades lineales. Si $g_i : \mathbb{R}^n \to \mathbb{R}$ es una función, y $g_i(x) \geq 0$ es una de las restricciones de desigualdad, diremos que dicha restricción está *activa en x* si $g_i(x) = 0$ y diremos que está *inactiva en x* si $g_i(x) > 0$. Por convención, las restricciones de igualdad se consideran activas en todos los puntos que las satisfagan.

Proposición 5.1.3 (Lema de Farkas) *Sean $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ y $B \in \mathbb{R}^{r \times n}$. Las siguientes proposiciones son equivalentes:*

(i) $\langle x, y \rangle \leq 0, \ \forall y \in \{y \in \mathbb{R}^n \mid Ay = 0, \ By \leq 0\}$

(ii) $\exists \lambda \in \mathbb{R}^m, \ \mu \in \mathbb{R}^r_{\geq} \mid x = A^T\lambda + B^T\mu$

Aquí $By \leq 0$ significa que la desigualdad se cumple en cada componente de By, y $\mathbb{R}^r_{\geq} = \{x \in \mathbb{R}^r \mid x \geq 0\}$.

Las demostraciones de las proposiciones 5.1.2 y 5.1.3 se encuentran en [1], proposiciones 2.1.2 y B.16d respectivamente.

Ahora con este lema se probará una versión de las condiciones de optimalidad necesarias para un caso particular como el del problema (2.4).

Proposición 5.1.4 (Condiciones necesarias de optimalidad) *Sea $f : \mathbb{R}^n \to \mathbb{R}$ una función continuamente diferenciable, $\Gamma \in \mathbb{R}^{m \times n}$, $g \in \mathbb{R}^m$ y $\Omega = \{x \in \mathbb{R}^n \mid \Gamma x = g, \ x \geq 0\} \subset \mathbb{R}^n$. Dado el problema de minimización con restricciones:*

$$\begin{aligned} &\textit{minimizar} \quad f(x) \\ &\textit{sujeto a} \quad x \in \Omega. \end{aligned}$$

Si el vector factible $x^* \in \Omega \subset \mathbb{R}^n$ es un mínimo local del problema, entonces existen $\lambda \in \mathbb{R}^m$ y $\mu \in \mathbb{R}^n$ tales que:

(i) $0 = \nabla f(x^*) + \Gamma^T \lambda - \mu$

(ii) $\mu \geq 0$

(iii) $\mu_i x_i^* = 0,\ \forall i = 1, \ldots, n$

A los vectores λ y μ se los llama *multiplicadores de Lagrange*, a la condición (iii) se la llama *condición de holgura complementaria*, y a las condiciones $(i) - (iii)$ junto con las condiciones de factibilidad se las llama *condiciones de optimalidad necesarias KKT*, por Karush-Kuhn-Tucker, o *condiciones necesarias de optimalidad de primer orden*

Demostración: En primer lugar, Ω es convexo por ser intersección de dos convexos: el núcleo de una transformación afín, y el conjunto $\mathbb{R}^n_{\geq}$. Luego, por la proposición 5.1.2 se tiene $\langle \nabla f(x^*), x - x^* \rangle \geq 0,\ \forall x \in \Omega$.

A continuación, dado $x \in \Omega$ se define el *cono tangente a Ω en x*, denotado $T_\Omega(x)$, como:

$$T_\Omega(x) = \{d \in \mathbb{R}^n \mid \exists d_k \to d,\ t_k \to 0^+ \text{ tales que } t_k > 0,\ x + t_k d_k \in \Omega,\ \forall k \in \mathbb{N}\}$$

Con esta definición se puede realizar el siguiente razonamiento:

$$\begin{aligned} x^* \text{ es mínimo local} &\implies \langle \nabla f(x^*), x - x^* \rangle \geq 0,\ \forall x \in \Omega \iff \\ &\iff \langle \nabla f(x^*), t(x - x^*) \rangle \geq 0,\ \forall x \in \Omega,\ \forall t > 0 \iff \\ &\iff \langle \nabla f(x^*), d \rangle \geq 0,\ \forall d \in T_\Omega(x^*). \end{aligned}$$

Entonces, hasta acá se tiene

$$x^* \text{ es mínimo local} \implies \langle \nabla f(x^*), d \rangle \geq 0,\ \forall d \in T_\Omega(x^*). \tag{5.1}$$

Ahora bien, se puede reescribir el conjunto $T_\Omega(x)$ para el caso donde Ω no es un conjunto arbitrario, sino que es conocido y tiene la forma dada. En este caso, dado $x \in \Omega$, se tiene:

$$T_\Omega(x) = \{d \in \mathbb{R}^n \mid \Gamma d = 0 \text{ y } d_i \geq 0,\ \forall i \in I_x\}$$

Donde $I_x = \{i \mid x_i = 0\}$, es decir, es el conjunto de las restricciones de desigualdad activas.

En efecto, si $x \in \Omega$ y $d \in T_\Omega(x)$, existen sucesiones $d_k \to d$ y $t_k \to 0^+$ que cumplen $t_k > 0$ y $x + t_k d_k \in \Omega$, $\forall k \in \mathbb{N}$. Esto quiere decir por un lado que $\Gamma(x + t_k d_k) = g$, $\forall k \in \mathbb{N}$. Como $x \in \Omega$, se tiene $\Gamma x = g$ y como $t_k > 0$, $\forall k \in \mathbb{N}$ vale $\Gamma d_k = 0$, $\forall k \in \mathbb{N}$. Como las transformaciones lineales son continuas y $d_k \to d$, entonces $\Gamma d = 0$. Por otro lado se tiene $(x + t_k d_k)_i \geq 0$, $\forall i \in I_x$, $\forall k \in \mathbb{N}$. Usando que $I_x = \{i \mid x_i = 0\}$ y que $t_k > 0$, $\forall k \in \mathbb{N}$ se obtiene $(d_k)_i \geq 0$, $\forall i \in I_x$, $\forall k \in \mathbb{N}$. Tomando límite, esto implica finalmente $d_i \geq 0$, $\forall i \in I_x$.

Para la vuelta, sea $x \in \Omega$ y $d \in \mathbb{R}^n$ tal que $\Gamma d = 0$ y $d_i \geq 0$, $\forall i \in I_x$. Para ver $d \in T_\Omega(x)$ sea t_k cualquier sucesión de números reales que cumpla $t_k \to 0^+$ y $t_k > 0$, $\forall k \in \mathbb{N}$ y sea $d_k = d$, $\forall k \in \mathbb{N}$. Si ahora se fija $k \in \mathbb{N}$ se tiene por un lado que $\Gamma(x + t_k d_k) = \Gamma x + t_k \Gamma d = g$, pues $x \in \Omega$ implica $\Gamma x = g$ y $d_k = d$, $\forall k \in \mathbb{N}$ y $\Gamma d = 0$ por hipótesis. Por otro lado, si además se fija $i \in I_x$ vale $(x + t_k d_k)_i = x_i + t_k d_i \geq 0$, pues $x_i = 0$ por la defínción de I_x y $t_k > 0$ por hipótesis. Luego se cumple la igualdad de conjuntos.

Si a continuación se denota con Id_x a la matriz que resulta de elegir las filas de la matriz identidad de rango n que se correspondan con los índices en I_x se puede escribir:

$$T_\Omega(x) = \{d \in \mathbb{R}^n \mid \Gamma d = 0,\ Id_x d \geq 0\}.$$

Combinando esta última forma de escribir $T_\Omega(x)$ con (5.1) se obtiene:

$$x^* \text{ es mínimo local } \Rightarrow \langle -\nabla f(x^*), d\rangle \leq 0,\ \forall d \in \{d \in \mathbb{R}^n \mid \Gamma d = 0,\ -Id_x d \leq 0\},$$

donde se cambió el signo de $\nabla f(x^*)$ y de Id_x y se invirtieron las desigualdades correspondientes para poder utilizar el Lema de Farkas.

Ahora el Lema de Farkas provee la existencia de multiplicadores de Lagrange $\lambda \in \mathbb{R}^m$ y $\tilde{\mu} \in \mathbb{R}^{|I_x|}_{\geq}$ tales que $-\nabla f(x^*) = \Gamma^T \lambda - Id_x^T \tilde{\mu}$. Se tiene que los multiplicadores $\tilde{\mu}$ se corresponden con las restricciones de desigualdad activas. Si se define un vector de multiplicadores $\mu \in \mathbb{R}^n$ donde los multiplicadores correspondientes a las restricciones de desigualdad activas tengan el correspondiente valor dado por $\tilde{\mu}$ y el resto sean cero, se podría escribir:

$$0 = \nabla f(x^*) + \Gamma^T \lambda - \mu,$$

que es precisamente la condición (i) que se buscaba. Como $\tilde{\mu} \geq 0$ y μ a lo sumo agrega componentes nulas, vale (ii). Por último, por como fue construido μ, los multiplicadores correspondientes a restricciones activas ($x_i = 0$) tienen permitido tomar valores positivos mientras que los correspondientes a restricciones inactivas sólo pueden valer 0. Es decir, a lo sumo uno de los dos, x_i o μ_i, pueden ser distintos de cero para cada i, y por lo tanto se cumple también (iii).

□

Se concluye esta sección con el enunciado de un teorema que servirá de herramienta para explicar porqué funciona el método del gradiente proyectado. Su demostración también se encuentra en [1], proposición 2.1.3.

Proposición 5.1.5 (Teorema de Proyección) *Sea $\Omega \subset \mathbb{R}^n$ un conjunto no vacío, cerrado y convexo. Entonces vale lo siguiente:*

(i) Para todo $z \in \mathbb{R}^n$ existe un único $x^* \in \Omega$ que minimiza $\|z - x\|$ sobre todos los $x \in \Omega$. Este vector se llama proyección de z en Ω *y se denota $P_\Omega(z)$.*

(ii) Dado $z \in \mathbb{R}^n$ y $x^* \in \Omega$ se tiene:

$$x^* = P_\Omega(z) \iff \langle z - x^*, x - x^* \rangle \leq 0, \ \forall x \in \Omega.$$

(iii) La función $f : \mathbb{R}^n \to \Omega$ definida por $f(x) = P_\Omega(x)$ es continua y no expansiva, es decir:

$$\|P_\Omega(x) - P_\Omega(z)\| \leq \|x - z\|, \ \forall x, z \in \mathbb{R}^n.$$

(iv) Si además Ω es un subespacio vectorial de $\mathbb{R}^n$ y son dados $z \in \mathbb{R}^n$ y $x^* \in \Omega$ se tiene que $x^* = P_\Omega(z)$ si y solo si $z - x^*$ es ortogonal a Ω, es decir:

$$x^* = P_\Omega(z) \iff \langle z - x^*, x \rangle = 0, \ \forall x \in \Omega.$$

5.2. El equilibrio de Wardrop como problema de optimización

En primer lugar se dará nuevamente el enunciado del teorema 2.2.2 y se dará la demostración del mismo.

Teorema 2.2.2 *Sea $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$ un grafo dirigido representando una red de transporte, g una demanda de esa red y $t_a(v_a)$, $a \in \mathcal{A}$ las funciones de costo por arco. Si se cumple que la red es:*

- *fuertemente conectada,*
- *con demandas positivas,*
- *con funciones de costo por arco positivas y continuas,*

entonces las condiciones de optimalidad de primer orden del siguiente problema:

$$
\begin{aligned}
&\textit{minimizar} \quad T(v) = \sum_{a\in\mathcal{A}} \int_0^{v_a} t_a(s)ds \\
&\textit{sujeto a} \qquad \sum_{r\in\mathcal{R}_{pq}} h_{pqr} = g_{pq} && \forall (p,q) \in \mathcal{C}, \\
&\qquad\qquad h_{pqr} \geq 0 && \forall r \in \mathcal{R}_{pq},\ \forall (p,q) \in \mathcal{C}, \\
&\sum_{(p,q)\in\mathcal{C}} \sum_{r\in\mathcal{R}_{pq}} \delta_{pqra} h_{pqr} = v_a && \forall a \in \mathcal{A},
\end{aligned}
$$

son equivalentes a las condiciones de equilibrio del usuario (2.2).

Demostración: En primer lugar, como se explicó en la sección 2.3, el problema (2.3) es equivalente al problema (2.4):

$$
\begin{aligned}
\text{minimizar} \quad & T(\Delta^T h) \\
\text{sujeto a} \quad & \Gamma h = g \\
& h \geq 0
\end{aligned}
$$

Para clarificar la notación se escribirá $f(h) = T(\Delta^T h)$. Para fijar las dimensiones de cada uno de los elementos de este problema se recuerda que $v \in \mathbb{R}^{|\mathcal{A}|}$ y que $T : \mathbb{R}^{|\mathcal{A}|} \to \mathbb{R}$. Se establece que $h \in \mathbb{R}^n$ y $g \in \mathbb{R}^m$. Luego $\Delta \in \mathbb{R}^{n\times|\mathcal{A}|}$ y $\Gamma \in \mathbb{R}^{m\times n}$. Como la función $T(v)$ está definida como $T(v) = \sum_{a\in\mathcal{A}} \int_0^{v_a} t_a(s)ds$ se tiene que es diferenciable y su gradiente es $\nabla T(v) = (t_a(v_a))_{a\in\mathcal{A}}$. Como por hipótesis las funciones de costo por arco $t_a(v_a)$ son continuas, vale que $T(v)$ es continuamente diferenciable. Además, como las transformaciones lineales son continuamente diferenciables y composición de continuamente diferenciables es continuamente diferenciable, se tiene que $f(h)$ también es continuamente diferenciable.

Entonces que el problema (2.4) cumple las hipótesis de la proposición 5.1.4. Para conservar la notación se dirá ahora que $\Omega = \{h \in \mathbb{R}^n \mid \Gamma h = g,\ h \geq 0\} \subset \mathbb{R}^n$.

Sea $h^* \in \Omega$ un mínimo local del problema (2.4), por la proposición 5.1.4 existen multiplicadores de Lagrange $\lambda \in \mathbb{R}^m$ y $\mu \in \mathbb{R}^n$ que satisfacen junto con h^* las condiciones de optimalidad de primer orden:

(i) $0 = \nabla f(h^*) + \Gamma^T \lambda - \mu$

(ii) $\mu \geq 0$

(iii) $\mu_i h_i^* = 0,\ \forall i = 1, \ldots, n$

(iv) $\Gamma h^* = g$

(v) $h^* \geq 0$

Ahora se debe ver que estas condiciones $(i)-(v)$ son equivalentes a las condiciones del equilibrio del usuario de Wardrop dadas por las ecuaciones (2.2a) a (2.2e).

$(\implies)$:

En primer lugar se ve a simple vista que las condiciones (iv) y (v) implican directamente las condiciones (2.2c) y (2.2d).

Si despeja μ de (i) y se reemplaza en (ii) y en (iii) se tiene:

$$\nabla f(h^*) + \Gamma^T \lambda \geq 0 \tag{5.2a}$$

$$(\nabla f(h^*) + \Gamma^T \lambda)_i h_i^* = 0, \ \forall i = 1, \ldots, n \tag{5.2b}$$

A continuación, se necesita saber cuál es la forma de $\nabla f(h)$. Primero se definirá $T_a(v_a) = \int_0^{v_a} t_a(s)ds$, de manera que $T(v) = \sum_{a \in \mathcal{A}} T_a(v_a)$. También se definirá Δ_a como la a-ésima columna de la matriz Δ, y se tratará a ambas como función o como arreglo indistintamente. Claramente se cumple $\langle \Delta_a, h \rangle = v_a$, es decir, el flujo en la arista v_a es igual a la suma de los flujos de las rutas que utilizan esa arista y Δ_a es el vector que contiene justamente esa información. Se tiene entonces:

$$\frac{\partial f(h)}{\partial h_i} = \frac{\partial T(\Delta^T h)}{\partial h_i} = \sum_{a \in \mathcal{A}} \frac{\partial T_a(\Delta_a(h))}{\partial h_i} = \sum_{a \in \mathcal{A}} \frac{dT_a(\langle \Delta_a, h \rangle)}{dv_a} \frac{d\Delta_a(h)}{dh_i} = \sum_{a \in \mathcal{A}} t_a(v_a)\delta_{ia}.$$

En 2.1.7 se definió el costo de recorrer la ruta r correspondiente al par (p, q) como $c_{pqr} = \sum_{a \in r} t_a(v_a)$. Si se utiliza la reindexación de la sección 2.3 y se considera sencillamente a las rutas con un único índice se tendría $c_i = \sum_{a \in i} t_a(v_a)$. Por último, por como quedó definida Δ en la sección 2.3:

$$\delta_{ij} = \begin{cases} 1 & \text{si la ruta } i \text{ contiene la arista } j \\ 0 & \text{en todo otro caso,} \end{cases}$$

resulta claro que:

$$\frac{\partial f(h)}{\partial h_i} = \sum_{a \in \mathcal{A}} t_a(v_a)\delta_{ia} = \sum_{a \in i} t_a(v_a) = c_i.$$

En la definición 2.3.1 se ve que la matriz Γ tiene exactamente un 1 en cada columna. Resulta entonces que la ecuación (5.2a) es $c_i + \lambda_{j_i} \geq 0$ y la ecuación (5.2b) es $(c_i + \lambda_{j_i})h_i^* = 0$, donde j_i es el índice del vector λ que corresponda para que $\lambda_{j_i} = (\Gamma^T \lambda)_i$. Más aún, tenemos que cada componente de λ se corresponde con una de las restricciones de igualdad del problema y que cada restricción de igualdad se corresponde exactamente con un par origen-destino. Por la forma de la matriz Γ se puede concluir que en realidad j_i es el índice del par origen destino al cual corresponde la ruta i.

Si llamamos $\pi = -\lambda$ y reindexamos todo de vuelta como estaba al principio, las ecuaciones (5.2a) y (5.2b) implican las condiciones (2.2b) y (2.2a) respectivamente.

Para completar la ida, basta ver que las condiciones $(i) - (v)$ también implican (2.2e). Para ello se utilizarán las hipótesis de demandas positivas y de funciones de costo por arco positivas. Para más claridad del argumento, en esta última parte se usará la indexación dada por las definiciones de la sección 2.1. Ahora bien, como las demandas eran todas positivas, es decir $g_{pq} > 0$, $\forall (p,q) \in \mathcal{C}$, se tiene que $\forall (p,q) \in \mathcal{C}$, existe al menos una ruta $r \in \mathcal{R}_{pq}$ tal que su flujo es $h_{pqr} > 0$. Luego, por la condición de holgura complementaria (2.2a) debe suceder $c_{pqr} - \pi_{pq} = 0$. Los costos por ruta se obtienen sumando los costos por arista, que son positivos por hipótesis, y por lo tanto vale $\pi_{pq} = c_{pqr} \geq 0$.

($\Longleftarrow$):

Para la vuelta se supone que existen un vector de flujos por ruta h^*, un vector de costos por ruta c y un vector de costos de equilibrio π que satisfacen las condiciones (2.2). Análogamente a la ida, se ve que las condiciones (2.2c) y (2.2d) implican directamente las condiciones (iv) y (v). Si se llama $\lambda = -\pi$, (2.2a) y (2.2b) se escriben como:

$$h^*_{pqr}(c_{pqr} + \lambda_{pq}) = 0,\ \forall r \in \mathcal{R}_{pq},\ \forall (p,q) \in \mathcal{C} \tag{5.3a}$$

$$c_{pqr} + \lambda_{pq} \geq 0,\ \forall r \in \mathcal{R}_{pq},\ \forall (p,q) \in \mathcal{C} \tag{5.3b}$$

En la ida se vió que $\nabla f(h) = c$. Usando eso y las propiedades de Γ se pueden escribir estas últimas ecuaciones en forma matricial. Luego, (5.3a) y (5.3b) implican (5.2b) y (5.2a), respectivamente.

Por último, basta con llamar $\mu = \nabla f(h^*) + \Gamma^T \lambda$ y se obtienen entonces (i), (ii) y (iii) automáticamente.

□

Se ha probado que para encontrar un vector de flujos por ruta que satisfaga las condiciones del equilibrio del usuario de Wardrop, basta resolver un problema de optimización asociado. Para concluir esta sección se mostrará que existe al menos un vector de flujos por ruta que satisface dicho equilibrio.

Teorema 5.2.1 *Bajo las mismas hipótesis del teorema 2.2.2, el problema de optimización* (2.4) *admite al menos una solución.*

Demostración: Se utilizará el Teorema de Weierstrass, que resumidamente dice que las funciones continuas sobre compactos alcanzan su mínimo (y su máximo) dentro de ese conjunto.

Para ello, se verá que la función objetivo del problema (2.4) es continua, y que el conjunto factible es compacto. Se llamará Ω al conjunto factible.

En la demostración del teorema 2.2.2 se vió que la función objetivo del problema es continuamente diferenciable, luego es continua. Como el conjunto factible es un

subconjunto de $\mathbb{R}^n$, para ver que es compacto basta ver que es cerrado y acotado (Teorema de Heine-Borel).

En la demostración de la proposición 5.1.4 se mostró que Ω es convexo, pues es intersección de dos convexos: el núcleo de una transformación afín y el conjunto $\mathbb{R}^n_{\geq}$. Ahora bien, estos dos conjuntos son además cerrados. El primero por ser homeomorfo al núcleo de Γ, que es cerrado por ser subespacio vectorial de dimensión finita. El segundo es cerrado por como fue definido. Luego Ω es cerrado por ser intersección finita de cerrados.

Para ver Ω acotado veamos que existe un número real $C_h > 0$ tal que $h \in \Omega$ implica $\|h\| \leq C_h$. Para ver esto, primero se recuerda que cada componente del vector h representa el flujo por una ruta correspondiente a un par origen-destino. La suma de las componentes correspondientes a todas las rutas de un mismo par deben sumar la demanda para ese par (dicho de otra forma, $\Gamma h = g$). Como además se tiene $h \geq 0$, cada una de las componentes de h puede a lo sumo ser igual a la demanda para su par origen-destino. Se tiene entonces que $h_{pqr} \leq g_{pq},\ \forall r \in \mathcal{R}_{pq},\ \forall (p,q) \in \mathcal{C}$ y por lo tanto $\|h\|^2 \leq n(\sum_{i \in \mathcal{C}} g_i)^2 = C_h^2$, donde n es la cantidad total de rutas para todos los pares origen-destino y C_h es sólo un nombre para la cota.

□

5.3. El método del gradiente proyectado

En esta sección se describirá uno de los métodos que existen para resolver los problemas de la forma (2.4), y se mostrarán los resultados que garantizan que el mismo encuentra un mínimo local.

El método que se eligió forma parte de un conjunto bastante grande de métodos llamados *métodos de descenso*. Estos métodos son iterativos y consisten en partir de un punto factible x^0 (una suposición inicial) y generar sucesivamente puntos factibles x^1, x^2,..., tales que el valor de la función objetivo en esos puntos es cada vez menor, es decir, $f(x^{k+1}) < f(x^k)$.

Para generar el vector x^{k+1} a partir del vector x^k, estos métodos generalmente buscan encontrar un vector d^k (llamado *dirección de descenso*) y un escalar α^k (llamado *longitud de paso*) de manera tal que $x^{k+1} = x^k + \alpha^k d^k$.

Entonces estos métodos reducen el problema de encontrar el mínimo de una función al problema de encontrar en cada paso una dirección de descenso y una longitud de paso de manera tal que la siguiente iteración "mejore" el valor de la función y que la sucesión de vectores generada converja a un mínimo de la función. En lo sucesivo, vamos a suponer que la función objetivo es continuamente diferenciable y la llamaremos f. Esto nos permite contar con el polinomio de Taylor de la función objetivo. Para el caso en el que $x^{k+1} = x^k + \alpha^k d^k$, se tiene que el polinomio de Taylor

de orden 1 alrededor de x^k, valuado en x^{k+1} es:

$$f(x^{k+1}) = f(x^k) + \alpha^k \langle \nabla f(x^k), d^k \rangle + o(\alpha^k) \tag{5.4}$$

Donde $o(\alpha^k)$ es la notación o minúscula y representa que si el resto del polinomio de Taylor es $R(\alpha^k)$ entonces $\lim_{\alpha^k \to 0} \frac{R(\alpha^k)}{\alpha^k} = 0$. De esta primera expresión es posible razonar que una dirección será dirección de descenso si $\langle \nabla f(x^k), d^k \rangle < 0$, y que para un $\alpha^k > 0$ lo suficientemente pequeño, el segundo término dominará sobre el tercero y se obtendrá un descenso de la función objetivo para el punto x^{k+1}.

Lo expuesto hasta ahora funciona para problemas sin restricciones, pero el problema (2.4) es un problema con restricciones y eso obliga a tener en cuenta otras consideraciones. Además de la condición de que la dirección elegida sea efectivamente de descenso ($\langle \nabla f(x^k), d^k \rangle < 0$) se debe pedir que el punto x^{k+1} siga siendo factible. Para ello se debe elegir una dirección d^k que sea *factible*, en el sentido que exista algún escalar $\alpha > 0$ tal que $x^k + \alpha d^k$ sea factible. Es claro que además de agregar esta consideración sobre la elección de d^k, debe agregarse la consideración al elegir α^k de que x^{k+1} siga siendo factible además de mejorar el valor de la función objetivo.

Los métodos de descenso para problemas con restricciones que buscan direcciones de descenso y longitudes de paso teniendo en cuenta todas las consideraciones que se expusieron hasta ahora reciben el nombre de *métodos de dirección de descenso factible*.

Dentro de los métodos de dirección de descenso factible, se eligió el *método del gradiente proyectado*. Este método usa una dirección de descenso factible que se obtiene resolviendo un subproblema cuadrático.

En lo sucesivo el problema en consideración será el de encontrar un mínimo para una función continuamente diferenciable $f(x)$, sujeto a que $x \in \Omega$, donde Ω es un subconjunto no vacío, convexo y cerrado de $\mathbb{R}^n$. Cada paso de este método tiene la forma:

$$x^{k+1} = x^k + \alpha^k(\bar{x}^k - x^k)$$

Donde $\bar{x}^k = P_\Omega(x^k - s^k \nabla f(x^k))$. Es decir, primero se considera una dirección de descenso en la mejor dirección, que es $-\nabla f(x^k)$, y se realiza un paso $-s^k \nabla f(x^k)$ en esa dirección. Como ese paso podría caer en un punto no factible, se proyecta el resultado $x^k - s^k \nabla f(x^k)$ dentro del conjunto factible y se obtiene el vector factible $\bar{x}^k$, que además determina una dirección factible $\bar{x}^k - x^k$. Por último, se realiza un paso a lo largo de esa dirección usando una longitud de paso α^k. En este caso, por como fue definida la dirección de descenso factible se tiene que $\alpha^k \in (0, 1]$.

Con respecto al escalar s^k, también podría ser visto como una longitud de paso. Para ello se puede pensar en la variante de considerar $\alpha^k = 1$ en todos los pasos, y se tendría entonces $x^{k+1} = P_\Omega(x^k - s^k \nabla f(x^k))$. En este caso se puede interpretar a s^k como una longitud de paso sobre el arco de la proyección del gradiente.

En este trabajo no se considerará la variante de longitud de paso sobre el arco de la proyección del gradiente. Es decir, se considerará $s^k = s$, $\forall k \in \mathbb{N}$. Esto se realiza por dos razones: en el momento de buscar una longitud de paso adecuado, se necesita calcular muchas proyecciones y eso es computacionalmente caro, como se verá más adelante. La segunda razón es que en experimentos computacionales no se vió una velocidad de convergencia más rápida.

Hasta ahora se explicó como se obtendrá la dirección de descenso factible, pero falta contar con un método para elegir una longitud de paso α^k en esa dirección. Para esta parte se usará un regla de reducción sucesiva de la longitud de paso conocida como *Regla de Armijo sobre la dirección factible*. Esta regla consiste en seleccionar una longitud de paso inicial α_0 y si el vector $x^k + \alpha_0 d^k$ no mejora el valor de la función ($f(x^k + \alpha_0 d^k) \geq f(x^k)$) se reduce sucesivamente la longitud de paso en un determinado factor β hasta que el valor de f es mejorado. Más aún, se pide que la mejora en la función sea mayor a una determinada cota.

Formalmente, la Regla de Armijo consiste en fijar escalares α_0, β y σ, con $0 < \beta, \sigma < 1$, y se elige $\alpha^k = \beta^{m_k}\alpha_0$, donde m_k es el primer entero no negativo m tal que:

$$f(x^k) - f(x^k + \beta^m \alpha_0 d^k) \geq -\sigma \beta^m \alpha_0 \langle \nabla f(x^k), d^k \rangle, \tag{5.5}$$

donde σ determina cuanto debe mejorar la función objetivo en cada paso y se elige cercano a 0, habitualmente $\sigma \in [10^{-5}, 10^{-1}]$ [1]. Se tiene entonces que en cada iteración de la regla se realiza un paso en la dirección factible con una longitud de paso $\beta^m \alpha_0$ hasta que se satisface la desigualdad anterior. Luego, la mejora en la función objetivo no debe ser sólo distinta de cero, sino lo suficientemente grande para satisfacer dicha desigualdad. Existe una variante de esta regla, llamada *Regla de Armijo sobre el arco*, que en lugar de variar α varía s de la definición del método del gradiente proyectado. Ya se explicó que no se eligió esta variante por la cantidad de proyecciones que necesita para ser implementada.

Hasta aquí se describió en qué consiste el método del gradiente proyectado y la regla de Armijo, pero nada se dijo con respecto a su convergencia. A continuación se expondrán resultados que garantizan la convergencia de ambos.

Con respecto a la regla de Armijo, se la describió diciendo que cada α^k se elegía en función de un menor entero no negativo m_k, pero no se dijo nada de si ese entero existe siempre o si puede suceder que la regla itere sin encontrar dicho entero. Pues bien, se mostrará que la regla de Armijo como fue definida encuentra una longitud de paso que satisface (5.5) en una cantidad finita de iteraciones.

Proposición 5.3.1 *Sea $f : \mathbb{R}^n \to \mathbb{R}$ una función continuamente diferenciable, $x \in \mathbb{R}^n$ un vector fijo, y $d \in \mathbb{R}^n$ una dirección de descenso de f en x, es decir $\langle \nabla f(x), d \rangle < 0$. Y sean los escalares $\alpha_0 > 0$, β y σ, con $0 < \beta, \sigma < 1$, y la sucesión $\{\beta^m \alpha_0\}_{m \in \mathbb{N} \cup \{0\}}$. Entonces existe un entero no negativo $\bar{m}$ tal que $m > \bar{m}$ implica:*

$$f(x) - f(x + \beta^m \alpha_0 d) \geq -\sigma \beta^m \alpha_0 \langle \nabla f(x), d \rangle$$

Idea de la Demostración: En primer lugar se debe observar que si se tiene un escalar $p < 0$ entonces existe un $\bar{\alpha} > 0$ tal que si $\alpha < \bar{\alpha}$ se cumple $-p > \frac{o(\alpha)}{\alpha}$, pues $\lim_{\alpha^k \to 0} \frac{o(\alpha^k)}{\alpha^k} = 0$. Como $\langle \nabla f(x), d \rangle < 0$ y $0 < \sigma < 1$ se puede tomar $p = (1 - \sigma)\langle \nabla f(x), d \rangle$ y se tendrá que existe un escalar $\bar{\alpha} > 0$ tal que:

$$\begin{aligned}
0 &> \alpha p + o(\alpha) && \forall 0 < \alpha < \bar{\alpha} \\
0 &> \alpha(1 - \sigma)\langle \nabla f(x), d \rangle + o(\alpha) && \forall 0 < \alpha < \bar{\alpha} \\
-\alpha\langle \nabla f(x), d \rangle - o(\alpha) &> -\sigma\alpha\langle \nabla f(x), d \rangle && \forall 0 < \alpha < \bar{\alpha} \\
f(x) - f(x + \alpha d) &> -\sigma\alpha\langle \nabla f(x), d \rangle && \forall 0 < \alpha < \bar{\alpha},
\end{aligned}$$

donde en el último paso se utilizó el polinomio de Taylor de la ecuación (5.4). Si ahora se considera la sucesión $\{\beta^m \alpha_0\}_{m \in \mathbb{N} \cup \{0\}}$, se tiene que tiende a cero pues $0 < \beta < 1$. Luego, existe un entero no negativo $\bar{m}$ tal que $\beta^m \alpha_0 < \bar{\alpha}$, $\forall m > \bar{m}$. Combinando este hecho sobre esta sucesión con la última de las desigualdades anteriores se ve que este $\bar{m}$ es precisamente el que se estaba buscando.

□

Se debe notar que este resultado garantiza que la regla de Armijo encontrará una longitud de paso adecuada en una cantidad finita de iteraciones, pero no dice que ese paso sea precisamente el asociado a $\bar{m}$. Podría suceder que para un $m < \bar{m}$ se cumpla (5.5) y la regla termine antes.

Ahora falta ver que el método del gradiente proyectado también converge a una solución. En primer lugar se tiene que el método se detiene si y solo si llega a un punto estacionario x^*. Es decir, en su siguiente iteración obtendrá el mismo punto. Esto es así pues x^* es un punto estacionario si y solo si $\langle \nabla f(x^*), x - x^* \rangle \geq 0$, $\forall x \in \Omega$. Pero esto es equivalente a:

$$\langle (x^* - s\nabla f(x^*)) - x^*, x - x^* \rangle \leq 0, \ \forall x \in \Omega, \ \forall s > 0,$$

y esto último a su vez es equivalente a $x^* = P_\Omega(x^* - s\nabla f(x^*))$ por la proposición 5.1.5. Se tiene entonces que la dirección de búsqueda es el vector nulo, y por ende en su siguiente paso el método no avanzará en ninguna dirección.

No obstante, esto no garantiza que el método llegue efectivamente a un punto estacionario. Bien podría suceder que itere sin acercarse a ningún punto estacionario. Por ejemplo, podría suceder que si bien en cada paso la dirección factible cumpla $\langle \nabla f(x), d \rangle < 0$, el valor de $\langle \nabla f(x), d \rangle$ sea demasiado pequeño para acercarse de manera significativa a un punto estacionario. El siguiente resultado garantiza que eso no sucede. Sobre este resultado también se dará sólo una idea de su demostración. La demostración completa está en [1], proposición 2.3.1.

Proposición 5.3.2 *Se considera el problema de minimizar la función continuamente diferenciable $f : \mathbb{R}^n \to \mathbb{R}$ sobre Ω, subconjunto no vacío, convexo y cerrado de $\mathbb{R}^n$. Si $\{x^k\}$ es una sucesión generada con el método del gradiente proyectado, con cada α^k elegido con la regla de Armijo sobre la dirección factible, todo punto límite de x^k es estacionario.*

Idea de la demostración: Primero se prueba que la sucesión de direcciones $\{\bar{x}^k - x^k\}$ cumple la siguiente propiedad: si la subsucesión $\{x^k\}_{k\in K}$ converge a un punto no estacionario $\bar{x}$, entonces:

$$\limsup_{k\to\infty,\ k\in K} \|\bar{x}^k - x^k\| < \infty \tag{5.6a}$$

$$\limsup_{k\to\infty,\ k\in K} \langle \nabla f(x^k), \bar{x}^k - x^k\rangle < 0 \tag{5.6b}$$

Esta propiedad garantiza que no suceda que las direcciones se hagan demasiado pequeñas y el método no se acerque a ningún punto estacionario.

Para ver (5.6a) se usa primero que la proyección es continua por la proposición 5.1.5, parte (iii). Luego:

$$\limsup_{k\to\infty,\ k\in K} \bar{x}^k = P_\Omega(\bar{x} - s\nabla f(\bar{x})).$$

De donde sigue (5.6a) pues $\|\bar{x}^k - x^k\|$ converge a $\|P_\Omega(\bar{x} - s\nabla f(\bar{x})) - \bar{x}\|$.

Para ver (5.6b) se usa que por la proposición 5.1.5, parte (ii) se tiene:

$$\langle x^k - s\nabla f(x^k) - \bar{x}^k, x - \bar{x}^k\rangle \le 0,\ \forall x \in \Omega.$$

En particular, si $x = x^k$ se tiene $\langle \nabla f(x^k), \bar{x}^k - x^k\rangle \le -\frac{1}{s}\|x^k - \bar{x}^k\|^2$, y tomando límite se obtiene:

$$\limsup_{k\to\infty,\ k\in K} \langle \nabla f(x^k), \bar{x}^k - x^k\rangle \le -\frac{1}{s}\|\bar{x} - P_\Omega(\bar{x} - s\nabla f(\bar{x}))\|^2.$$

Luego, como $\bar{x}$ es un punto no estacionario, el lado derecho de la desigualdad es negativo, de donde se sigue (5.6b).

Ahora para ver la proposición se supone por el contrario que no vale. Es decir, se supone que existe un punto límite $\bar{x}$ que no es estacionario. Como $\{f(x^k)\}$ es monótonamente no creciente, o bien converge a un valor finito, o bien diverge a $-\infty$. Como f es continua, $f(\bar{x})$ es un punto límite de $\{f(x^k)\}$, y por lo tanto la sucesión $\{f(x^k)\}$ converge a $f(\bar{x})$. Se tiene entonces que la sucesión $\{f(x^k)\}$ es de Cauchy, luego el lado izquierdo de la condición de Armijo, desigualdad (5.5), tiende a cero y eso implica que $\alpha^k\langle \nabla f(x^k), \bar{x}^k - x^k\rangle \to 0$.

Sea ahora una subsucesión $\{x^k\}_{k\in K}$ que converge a $\bar{x}$. Por (5.6b) se tiene que la sucesión $\{\alpha_k\}_K \to 0$. Luego, por la definición de la regla de Armijo, esto quiere

decir que existe un índice $\bar{k} \geq 0$ tal que para todo $k \in K,\ k \geq \bar{k}$ el paso inicial α_0 debe ser reducido por lo menos una vez. Esto es:

$$f(x^k) - f(x^k + (\frac{\alpha^k}{\beta}(\bar{x}^k - x^k) < -\sigma\frac{\alpha^k}{\beta}\langle\nabla f(x^k), \bar{x}^k - x^k\rangle,\ \forall x \in K,\ k \geq \bar{k}$$

Ahora, llamando $p^k = (\bar{x}^k - x^k)/\|\bar{x}^k - x^k\|$ y $\bar{\alpha}^k = \alpha^k\|\bar{x}^k - x^k\|/\beta$, por (5.6a) se tiene $\{\bar{\alpha}^k\}_K \to 0$, y como $\|p^k\| = 1$ para todo $k \in K$, existe una subsucesión $\{p^k\}_{\bar{K}}$ tal que $\{p^k\}_{\bar{K}} \to \bar{p}$. Luego:

$$\begin{aligned}\frac{f(x^k) - f(x^k + \bar{\alpha}^k p^k)}{\bar{\alpha}^k} &< -\sigma\langle\nabla f(x^k), p^k\rangle,\ \forall k \in \bar{K},\ k \geq \bar{k}\\ -\langle\nabla f(x^k + \tilde{\alpha}^k p^k, p^k\rangle &< -\sigma\langle\nabla f(x^k), p^k\rangle,\ \forall k \in \bar{K},\ k \geq \bar{k},\end{aligned}$$

donde en el segundo paso se utilizó el teorema del valor medio y $\tilde{\alpha}^k \in [0, \bar{\alpha}^k]$. Tomando límite se obtiene:

$$\begin{aligned}-\langle\nabla f(\bar{x}), \bar{p}\rangle &\leq -\sigma\langle\nabla f(\bar{x}), \bar{p}\rangle\\ 0 &\leq (1-\sigma)\langle\nabla f(\bar{x}), \bar{p}\rangle\\ 0 &\leq \langle\nabla f(\bar{x}), \bar{p}\rangle,\end{aligned}$$

donde en el último paso se usó $\sigma < 1$. Combinando (5.6a) y (5.6b) con la definición de p^k y $\bar{p}$ se obtiene que $\langle\nabla f(\bar{x}), \bar{p}\rangle < 0$,lo cual es absurdo. Este absurdo provino de suponer que no valía la proposición.

□

Hasta aquí se tiene garantizado que el método del gradiente proyectado junto con la regla de armijo convergen a un punto estacionario de la función. Para el caso particular de la función que se tiene en este problema se puede ver que además este punto es un mínimo local.

En primer lugar se debe recordar que se definieron dos funciones objetivos. En la sección 4.4 se explicó que para el primer problema (grafo sólo con las líneas de colectivo) los costos por arista son constantes y si se recuerda la definición de la función objetivo $T(v)$ como la suma de las integrales de las funciones de costo por arista, resulta que es una función lineal. En este caso tenemos que la función es convexa, y por ende todo punto estacionario es mínimo global.

La función objetivo para el segundo caso (grafo completo) es cóncava. Esto es pues los costos por arista son constantes para las líneas de colectivo existentes y de la forma $c_a(1 + e^{-v_a})$ para las aristas de las calles. Luego la función objetivo tiene algunas componentes lineales, y otras de la forma $c_a(v_a - e^{-v_a}$, y ambas son funciones cóncavas.

Teniendo que la función objetivo es cóncava, y sabiendo que un punto estacionario es mínimo local, máximo local o un punto de inflexión, se puede razonar lo siguiente. Dado un punto estacionario encontrado con el método del gradiente proyectado con la regla de Armijo:

Si se supone que el punto es de inflexión en el conjunto factible, entonces es punto de inflexion en $\mathbb{R}^n$. Luego el gradiente de la función en ese punto se anula. Pero la función es cóncava, y eso implicaría que se trata de un maximizador. De aquí que un punto estacionario encontrado con este método no puede ser punto de inflexión.

Si se supone que el punto es un máximo local, como la función es cónvaca, se tiene que es un máximo global. Ahora bien, el método del gradiente proyectado con la regla de Armijo genera una sucesión de puntos monótona decreciente. Se sabe que el método para si y solo si encuentra un punto estacionario. Luego si la sucesión de puntos generada tiene más de un punto, cualquiera de los puntos anteriores al último es un punto en el que la función objetivo tiene un valor mayor que el valor encontrado en el punto estacionario. Pero esto contradice el hecho de que el punto estacionario encontrado sea un máximo. De aquí que un punto estacionario así encontrado tampoco puede ser máximo local.

En conclusión, un punto estacionario encontrado con el método del gradiente proyectado con la regla de Armijo sólo puede ser un mínimo local si la función en cóncava, que es el caso de la función objetivo del problema de este trabajo.

5.4. Solución final

En esta sección se explicará primero cómo se implementó lo expuesto hasta ahora para diseñar la nueva línea de colectivo propuesta, y luego se describirá cómo se ejecutó la implementación y que resultados se obtuvieron.

5.4.1. Implementación

Tanto para el método del gradiente proyectado como para la regla de Armijo se utilizaron las implementaciones del curso de Optimización no lineal dictado en 2014 en la FaMAF [4].

Para llamar a la función de la regla de Armijo fue necesario implementar una función que realice la proyección en el conjunto factible $P_\Omega(h)$. Es decir, que resuelva el subproblema al que se hizo referencia en la sección 5.3. Dicho problema puede ser formulado como:

$$\begin{array}{rc} \text{minimizar} & \frac{1}{2}\|\xi - h\|^2 \\ \text{sujeto a} & \Gamma\xi = g \\ & \xi \geq 0 \end{array}$$

O equivalentemente, en la forma de un problema cuadrático:

$$\begin{array}{rc} \text{minimizar} & \frac{1}{2}\xi^T Id_n \xi + \langle \xi, -h \rangle \\ \text{sujeto a} & \Gamma\xi = g \\ & \xi \geq 0, \end{array} \tag{5.7}$$

donde Id_n es la matriz identidad de dimensión $n \times n$. Se intentó resolver el problema en esta última formulación usando la función `qp` de Octave, pero para una matriz Γ de dimensión aproximadamente 2700×30000 cada proyección demoraba en el orden de las 12 horas. Dado que en el método de gradiente proyectado se necesitan hacer muchas proyecciones, este tiempo resultaba prohibitivo.

Se reparó entonces en que el problema podía ser reducido a varios problemas significativamente más sencillos.

Para explicar esto se analizará la estructura de la matriz Γ. Se define 1_n como la matriz en $\mathbb{R}^{1\times n}$ que tiene todas sus entradas iguales a 1, y se recuerda que en la sección 2.3 se definió $r_i = |\mathcal{R}_i|$, es decir, r_i es la cantidad de rutas para el par origen destino $i \in \mathcal{C}$. Con esta notación, se tiene que la matriz Γ tiene la forma:

$$\Gamma = \begin{pmatrix} 1_{r_1} & & & \\ & 1_{r_2} & & \\ & & \ddots & \\ & & & 1_{r_{|\mathcal{C}|}} \end{pmatrix}$$

Ahora, dado un vector $x \in \mathbb{R}^{\bar{r}}$, donde $\bar{r} = \sum_{i\in\mathcal{C}} r_i$, denotamos con x_{r_1} a las primeras r_1 componentes, x_{r_2} a las siguientes r_2 componentes, etc. Así tenemos que el problema de proyección puede ser escrito como:

$$\begin{array}{rc} \text{minimizar} & \displaystyle\sum_{i\in\mathcal{C}} \left(\frac{1}{2} \xi_{r_i}^T Id_{r_i} \xi_{r_i} + \langle \xi_{r_i}, -h_{r_i} \rangle \right) \\ \text{sujeto a} & 1_{r_i} \xi_{ri} = g_i, \ \forall i \in \mathcal{C} \\ & \xi_{r_i} \geq 0, \ \forall i \in \mathcal{C}. \end{array}$$

Nótese que $g \in \mathbb{R}^{|\mathcal{C}|}$ no fue reindexado, sino que se utilizó el sentido usual de g_i. A continuación se usará que como el problema es cuadrático, un punto es solución del problema si y solo si es estacionario. Luego, h^* es solución del problema si y solo si se cumplen las condiciones KKT para ese punto. Esto es, si y solo si existen $\lambda \in \mathbb{R}^{|\mathcal{C}|}$ y $\mu \in \mathbb{R}^{\bar{r}}$ tales que:

(i) $0 = h^* - h + \Gamma^T \lambda - \mu$

(ii) $\mu \geq 0$

(iii) $\mu_j h_j^* = 0, \ \forall j = 1, \ldots, \bar{r}$

(iv) $\Gamma h^* = g$

(v) $h^* \geq 0$

Si ahora se reescriben estas condiciones con la notación recién definida se tiene:

$$\left.\begin{array}{ll} (a) & 0 = h^*_{r_i} - h_{r_i} + 1^T_{r_i}\lambda_i - \mu_{r_i} \\ (b) & \mu_{r_i} \geq 0 \\ (c) & \mu_j h^*_j = 0,\ \forall j = 1, \ldots, \bar{r_i} \\ (d) & 1_{r_i} h^*_{r_i} = g_i \\ (e) & h^*_{r_i} \geq 0 \end{array}\right\} \forall i \in \mathcal{C}$$

Nuevamente nótese que, al igual que g, λ no fue reindexado. Ahora bien, al ser sólo una reindexación, se tiene que las condiciones $(i) - (v)$ valen si y solo si las condiciones $(a) - (e)$ valen $\forall i \in \mathcal{C}$. Pero si se fija $i \in \mathcal{C}$ se tiene que las condiciones $(a) - (e)$ valen si y solo si $h^*_{r_i}$ es solución del problema:

$$\begin{array}{rc} \text{minimizar} & \frac{1}{2}\xi^T_{r_i} Id_{r_i}\xi_{r_i} + \langle \xi_{r_i}, -h_{r_i} \rangle \\ \text{sujeto a} & 1_{r_i}\xi_{ri} = g_i \\ & \xi_{r_i} \geq 0. \end{array} \tag{5.8}$$

Luego se tiene que h^* es solución de (5.7) si y solo si $h^*_{r_i}$ es solución de (5.8), $\forall i \in \mathcal{C}$. Gracias a esta última conclusión, se puede encontrar una solución al problema (5.7) encontrando la solución a cada uno de los $|\mathcal{C}|$ problemas más sencillos de la forma (5.8).

Esta idea se implementó en la función `qp_s.m` y esa es la función de proyección que se utiliza en la implementación del problema (Apéndice A). Cada uno de los subproblemas de la forma (5.8) se resuelven con la función `qp` de Octave. Se probó con la misma matriz Γ de aproximadamente 2700×30000 que se mencionó antes, y el tiempo de proyección bajó al orden de los 30 segundos.

5.4.2. Ejecución

Como se dijo en la sección 2.3, primero se averiguará el valor de la función objetivo del problema (2.4) para el caso del grafo que tiene sólo las líneas de colectivos existentes y luego se averiguará sobre todo el grafo cual es la distribución de flujos que satisface el equilibrio del usuario de Wardrop, que se espera que represente la nueva línea.

En la sección 4.2 se explicó cómo se iban a construir las matrices Δ y Γ. Allí se explicó que los algoritmos que buscan las rutas posibles para cada par origen-destino usan para determinar si una ruta en evaluación es una de las rutas buscadas el criterio de los 300 m. Si bien se intentó buscar un algoritmo que sirva para todos los casos en los que fuera necesario construir estas matrices, se vió que ese criterio no era conveniente para el caso del grafo completo. Se llegó a esa conclusión al observar que en la afectación de equilibrio había bastantes flujos que "se cortaban" a 300m

de algunos nodos de la UNC. En esa situación, no iba a suceder que se obtuviera un recorrido "continuo" para la nueva línea. Se decidió entonces modificar los códigos de `rutas.m` y `a_star.m`, cambiando el criterio de los 300 m por un criterio de 1 m. Si bien se sabe que no hay ninguna cuadra de la ciudad que sea tan corta, se eligió ese criterio para obligar a que todas las rutas salgan de (resp. lleguen a) su correspondiente origen (resp. destino), sin poner una condición del tipo "igual a cero", ya que por errores de redondeo podría no darse nunca. Los códigos con el criterio de los 300 m se guardaron como `rutas_s_l.m` y `a_star_s_l.m`, pues sólo serán utilizados para el caso del grafo que contiene sólo las líneas de colectivos existentes. Análogamente se guardó `genera_D_G_s_l.m`, que es idéntico a `genera_D_G.m` salvo que llama a `rutas_s_l.m` en lugar de a `rutas.m`.

En la sección 4.2.3 se mostraron dos ejecuciones de la función que genera las matrices Δ y Γ, y se dijo que esas matrices no fueron utilizadas luego. La matriz Γ tenía una dimensión de 2627×288057 en el primer caso (un promedio de 110 rutas por par origen-destino), y de 2990×113561 en el segundo (un promedio de 38 rutas por par origen-destino). Sobre estos datos se notó primero que daban un indicio de cual era el orden de magnitud de la cantidad máxima de rutas que la función `genera_D_G.m` era capaz de encontrar. Cabe recordar que dicha función no encuentra todas las rutas posibles aunque tenga infinitos recursos, sólo encuentra aquellas que "no se desvían demasiado" del camino óptimo. Lo segundo que se notó fue que los tiempos para realizar las proyecciones aumentaban significativamente al aumentar el número de rutas por par. Para el primer caso las proyecciones tardaban en el orden de los 15 minutos, y para el segundo en el orden de los 5 minutos. Ambos tiempos resultan prohibitivos dado que, como se dijo antes, se necesitan realizar muchas de estas proyecciones con el método del gradiente proyectado.

Se probó correr la función `genera_D_G.m` con varios valores para las cotas de tiempo y cantidad máxima de rutas por par, tratando de conseguir tiempos razonables de ejecución para esta función y para la proyección. Se llegó a la conclusión de que unos valores convenientes eran 3 segundos para el tiempo de búsqueda y 30 rutas por par.

Una vez finalizado este análisis se comenzó con la ejecución de todo lo implementado hasta ahora. Se generaron las matrices Δ y Γ para el caso del grafo que contiene sólo las líneas de colectivos corriendo en Octave:

```
> reg_A=load('reg_A');
> A=load('matriz_A_2');
> A=sparse(A(:,1),A(:,2),A(:,3));
> i_aux=[1:(reg_A(2)-reg_A(1))/4]'*4-2+reg_A(1);
> i_aux=[i_aux;[reg_A(2)+1:reg_A(4)]'];
> i_A=setdiff([1:rows(A)],i_aux);
> A(i_A,:)=zeros(length(i_A),columns(A));
> [i j x]=find(A);
> A=[i j x];
```

```
> c_a=load('costo_aristas');
> i_c_a=setdiff([1:rows(c_a)],i_aux);
> c_a(i_c_a)=zeros(length(i_c_a),1);
> c_a=c_a(find(c_a));
> p=mean(c_a);
> genera_D_G_s_l('g_OD_s_l',A,p,3,30)
```

Como se dijo en la sección 4.4, para este caso los costos por arista son constantes, y si se recuerda la definición de la función objetivo $T(v)$ como la suma de las integrales de las funciones de costo por arista, resulta que es una función lineal. Debido a esto, se tiene que el problema (2.4) es un problema lineal y puede ser resuelto con la función `glpk` de Octave. Se corrió entonces:

```
> c_a=load('costo_aristas');
> i_c_a=setdiff([1:rows(c_a)],i_aux);
> c_a(i_c_a)=zeros(length(i_c_a),1);
> Delta=load('Delta_s_l');
> Delta=sparse(Delta(:,1),Delta(:,2),Delta(:,3));
> if (columns(Delta)<rows(c_a))
> Delta(rows(Delta),rows(c_a))=0;
> endif
> c=c_a'*Delta';
> Gamma=load('Gamma_s_l');
> Gamma=sparse(Gamma(:,1),Gamma(:,2),Gamma(:,3));
> b=ones(rows(Gamma),1); % el vector b para glpk
> [xopt, fmin, errnum, extra] = glpk (c, Gamma, b);
```

Se obtuvo un valor de `fmin = 144.67` y de `errnum = 0`. Para interpretar el valor de `fmin` sería necesario realizar primero varias conversiones de unidades, pues es la suma de los costos de recorrer cada ruta, donde el costo de cada ruta es el costo de recorrer cada arista que compone esa ruta y los costos de las aristas fueron computados como la norma 1 de la diferencia de sus extremos en las coordenadas geográficas. Sería necesario convertir eso a alguna unidad de distancia y estimar alguna velocidad para convertir eso a tiempo. Además, se debe recordar que algunas aristas tienen costos de transbordos que fueron fijados en 10 veces el costo promedio del resto de las aristas y no se sabe a priori cuántos transbordos hay en la solución óptima. Se decidió utilizar este valor para compararlo con el valor de una función objetivo análoga en el óptimo de un problema como este que incluya a la nueva línea.

A continuación se generaron las matrices Δ y Γ para el problema con el grafo completo. Se ejecutó en Octave:

```
> reg_A=load('reg_A');
> A=load('matriz_A_2');
> c_a=load('costo_aristas');
```

```
> c_a=c_a(find(c_a));
> p=mean(c_a);
> genera_D_G('g_OD',A,p,3,30);
```

En este caso la función objetivo no es lineal, por lo que tuvo que ser definida como una función en Octave. Se la implementó de manera que calcule el valor de la función propiamente dicho, y el gradiente de la misma en el punto (Apéndice A).

Para resolver este problema se utilizaron los códigos mencionadas al principio de esta sección, que implementan el método del gradiente proyectado y la regla de Armijo. Se corrió en Octave:

```
> global reg_A c_a Delta Gamma
> reg_A=load('reg_A');
> c_a=load('costo_aristas');
> Delta=load('Delta');
> Delta=logical(sparse(Delta(:,1),Delta(:,2),Delta(:,3)));
> Gamma=load('Gamma');
> Gamma=logical(sparse(Gamma(:,1),Gamma(:,2),Gamma(:,3)));
> for i=1:rows(Gamma)
> h0(min(find(Gamma(i,:))))=1;
> endfor
> if(length(h0)<columns(Gamma))
> h0(columns(Gamma))=0;
> endif
> h0=h0(:);
> [h,f,err,k,salida]=min_gradPr(@T,h0,@(x)qp_s(Gamma,x),[100
1e-12],@reg_armijoPr,[100 1000 1 1 0.5 1e-4]);
```

Las variables que son definidas como globales son utilizadas en el interior de algunas de las funciones que utiliza esta implementación. Las matrices `Delta` y `Gamma` se eligieron de tipo booleano pues están compuestas sólo por unos y ceros, y de esa manera se ahorraba mucho espacio en memoria. El vector `h0` es la estimación inicial para el método. Es un vector factible que representa el flujo factible donde toda la demanda de cada par utiliza la primera ruta disponible.

El cuarto argumento de `min_gradPr` son los parámetros del método. La primera entrada es la cantidad máxima de iteraciones y el segundo es la tolerancia que debe superar el método para detenerse. El valor de dicha tolerancia fue elegido menor que `eps*29118`(estaba en el orden de $6e-12$), pues 29118 es el tamaño del vector `h`. El último argumento de `min_gradPr` son los parámetros de la regla de Armijo. La primera entrada es la cantidad máxima de iteraciones que realizará la regla, la segunda es el valor inicial del parámetro s. Se eligió tan grande porque para valores menores el método avanzaba muy lentamente hacia la solución. La tercera entrada es el factor de reducción del factor s, que es 1 pues no se está utilizando la Regla de Armijo sobre el arco, sino sobre la dirección factible. Las entradas cuarta, quinta y sexta son los valores de los parámetros α_0, β y σ de la regla de Armijo.

El método del gradiente proyectado realizó 22 iteraciones y en todas se utilizó la regla de Armijo con α_0 como α^k, es decir que en todas las iteraciones la proyección del gradiente mejoró el valor de la función objetivo sin necesidad de buscar un punto intermedio a lo largo de la dirección factible definida por esa proyección. El valor final de la función objetivo fue `f = 137.40`, la tolerancia que alcanzó el método fue `err = 3.1689e-14` y demoró 1210 segundos.

Para visualizar el resultado se necesitan los flujos por arista, por lo que se corrió en Octave:

```
> v=Delta'*h;
> dlmwrite('v_1',v,' ')
```

A continuación se decidió preparar estos datos para poder importarlos como un archivo `kml` a My Maps. Para ello se decidió mostrar las aristas con diferente color de línea y grosor según el volumen de flujo que hubiera por ellas. La función `prepara.m` se encarga de ello. El vector `p` que se utiliza como argumento de la función es un vector que determina los extremos de los intervalos dentro de los cuales se les asigna a cada arista un estilo (color y grosor) acorde a su volumen. Dicho vector se definió como `p=[5 10 15 20 40]` en Octave. Como se dijo en la sección 3.1, My Maps admite hasta 2000 dentro de un archivo `kml` importado. La función `prepara` se encarga de este inconveniente generando dos archivos de coordenadas de aristas, uno con las primeras 2000 y otro con las restantes.

Se utilizó la función en python `genera-kml.py` para generar el archivo `kml` para cada uno de los conjuntos de aristas mencionados. Se corrió:

```
$ mv mapa1 input
$ python3 genera-kml.py > mapa_1.kml
$ mv input mapa1
$ mv mapa2 input
$ python3 genera-kml.py > mapa_2.kml
$ mv input mapa2
```

Se obtuvieron dos archivos: `mapa_1.kml` y `mapa_2.kml`, con una sintaxis similar a la del archivo `grafo-terminado.kml` mencionado en la sección 4.1. La única diferencia era la sección de estilos que se preparó por separado y fue incluida a mano posteriormente en cada uno de los archivos obtenidos. La sección de estilos que se preparó está en el apéndice B.3. Los grosores y colores varían en función del volumen de flujo en cada arista, yendo de más delgado y rojo a más grueso y verde a medida que el flujo aumenta. Las aristas con flujo nulo tienen color negro.

Por último, se importaron esos archivos en My Maps y se obtuvo el resultado de la figura 5.1.

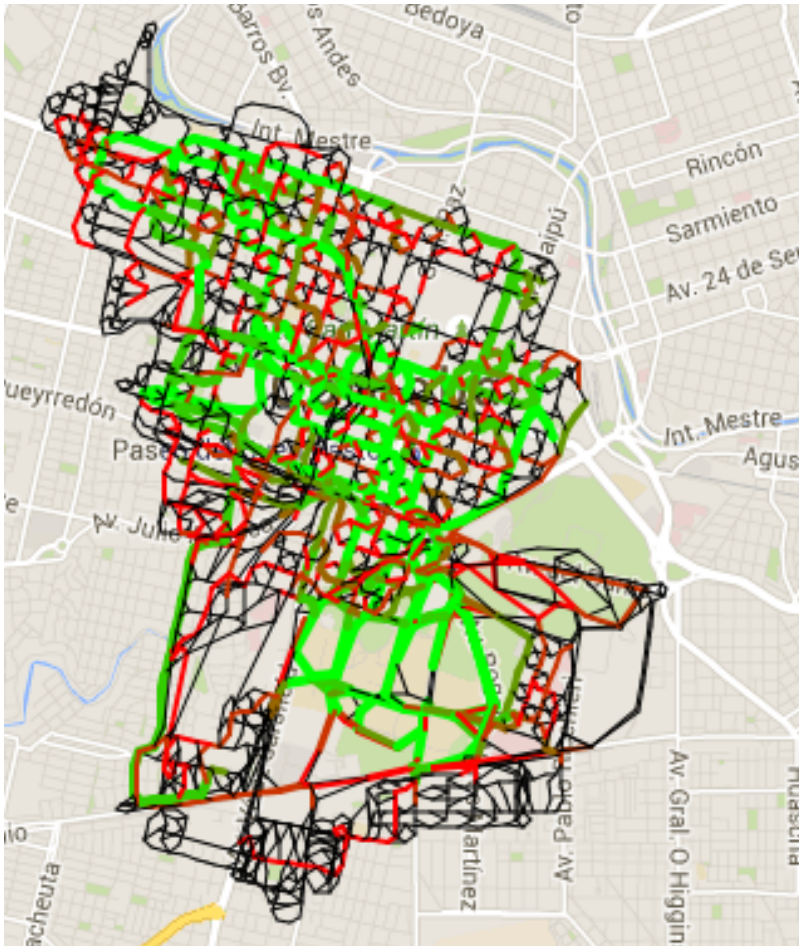


Figura 5.1: Flujos en las aristas del grafo de las calles

5.5. Análisis del resultado

Se debe recordar que el objetivo era diseñar una línea de colectivo mediante el siguiente procedimiento: resolver un problema de optimización para obtener flujos de equilbrio sobre un grafo que contiene tanto las calles de la ciudad como las líneas de colectivo existentes, donde el equilibrio es en el sentido del equilbrio del usuario de Wardrop. Los costos de recorrer las aristas que no sean líneas de colectivos existentes fueron elegidos de manera de penalizar usuarios que recorran de manera solitaria las aristas correspondientes a las calles. Con esas consideraciones, se esperaba que al observar los flujos sobre las calles, las únicas aristas con flujos positivos sean aquellas por las que debe ir la nueva línea de colectivo.

En la figura 5.1 se ven aristas con flujos no nulos en buena parte del grafo (mas de la mitad de las aristes de hecho). Por este motivo se decidió colorearlos en función del volumen de flujo.

Un primer enfoque para analizar este resultado es suponer que se cuenta con una distribución de flujos como la deseada. Es decir, que representan el recorrido de la nueva línea de colectivo con las aristas de las calles que tienen flujo no nulo. Si ese fuera el caso, es posible que para alguno de los pares origen-destino en consideración la ruta que le corresponde recorrer en esa distribución de flujos simplemente no haya sido encontrada por el algoritmo de búsqueda de rutas.

Para explicar mejor este punto se debe recordar que si bien el modelo necesita que se calculen todas las rutas posibles para cada par origen-destino, esto no pudo hacerse de manera computacionalmente eficiente. En lugar de eso, se decidió elegir sólo algunas de esas rutas posibles. El criterio fue encontrar la mejor ruta posible en el sentido de las distancias, y a partir de esa buscar rutas "similares". Esa segunda búsqueda se realizó con el algoritmo de búsqueda en profundidad con varias restricciones:una cantidad máxima de aristas que podía tener una ruta, una cantidad máxima de rutas que podía encontrar, y una cantidad máxima de tiempo que podía emplear en buscar esas nuevas rutas.

Estas restricciones introducen al menos tres causas por las que la ruta que se necesita para algún par origen-destino podría no haber sido encontrada. La primera es que la ruta tenga más aristas que la cota que se impuso, la segunda es que la búsqueda en profundidad no haya encontrado esa ruta entre las primeras que encontró y detuvo su búsqueda por alcanzar la cantidad máxima de rutas y la tercera es que no se haya contado con tiempo suficiente.

Se intentó mejorar el resultado tratando de identificar aristas que claramente no puedan ser parte del recorrido final (ya sea porque estén lejos de todos los nodos de la UNC y tengan poco flujo, o porque estén cerca de algun nodo, pero en una dirección que se estimó que no convenía que fuera recorrida por la nueva línea). Se supuso que esta acción tendría como consecuencia que aumente la probabilidad de que el algoritmo que busca rutas encuentre aquellas que se necesitan. Se identificaron 72 aristas que se juzgó que no eran candidatas a formar parte del recorrido de la nueva línea, y se quitaron manualmente del grafo. Se corrieron todos los algoritmos nuevamente y se obtuvo el resultado de la figura 5.2

Se ve que hubo algunas mejoras en la distribución de los flujos (por ejemplo desaparecieron algunos flujos que estaban muy alejados de cualquier nodo UNC y eran muy pequeños) pero eso sigue estando lejos de definir con claridad el recorrido de una nueva línea.

Otra razón por la que el resultado puede no haber sido el esperado esta relacionado con la naturaleza del problema de optimización. Es decir, aunque se contara con recursos ilimitados para encontrar todas las rutas para cada par origen-destino, podría obtenerse un resultado no deseado (con gran cantidad de flujos por todo el grafo). Esto es así pues se tiene que la función objetivo es cóncava, y no se puede garantizar unicidad del mínimo. Teniendo en cuenta esto podría haber sucedido incluso que esten presentes entre las rutas encontradas aquellas que representan el

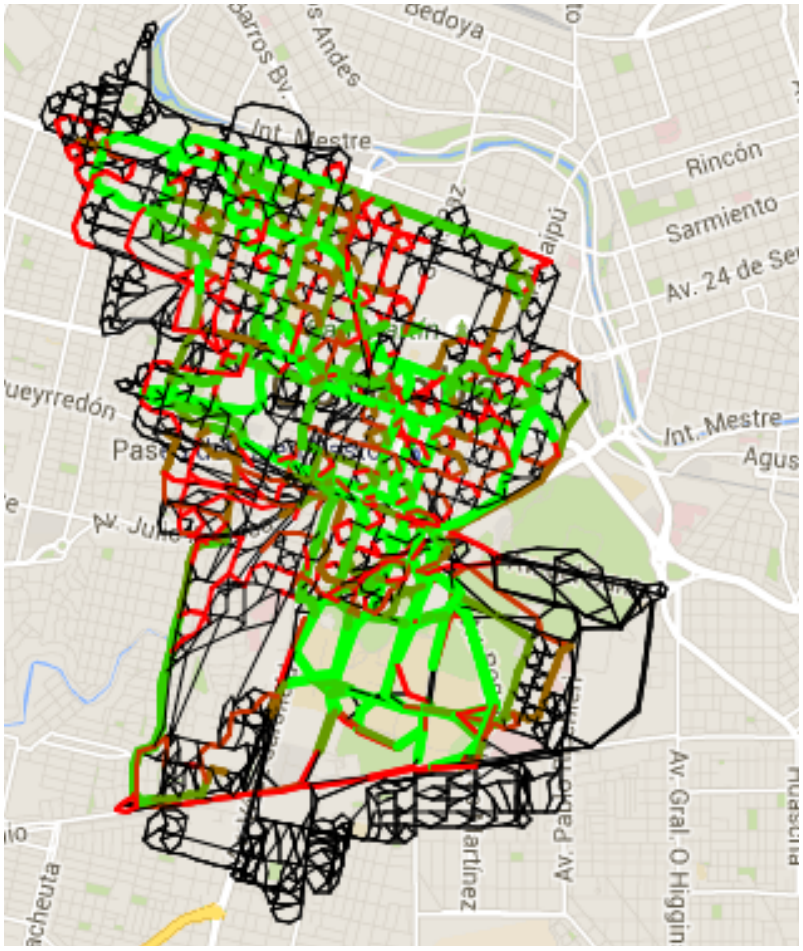


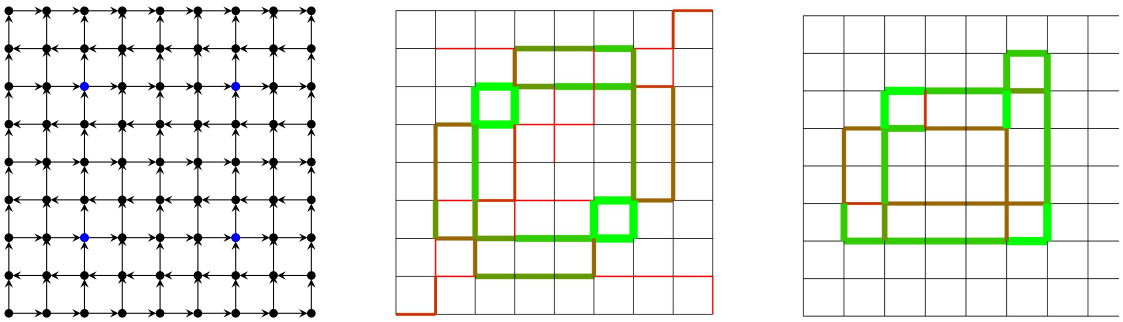
Figura 5.2: Flujos en las aristas del grafo de las calles

equilibrio que se buscaba, pero que ese punto no haya sido encontrado al resolver el problema de optimización, sino otro mínimo local.

Cabe destacar que además de estas dos razones enunciadas (que se cree que son las principales) se pensaron otras razones que también podrían haber introducido errores en el planteo y la solución del problema. Esas razones se enumerarán en la sección 6.1.

Un último comentario es que la mayoría de los códigos que fueron utilizados se fueron probando en un grafo simplificado pues era más fácil depurarlos de esa manera. Se podía incluso revisar rutas encontradas buscando arista por arista, entre otras cosas. En la figura 5.3(a) se ve cómo es este grafo. Los puntos en azul son los nodos UNC que se eligieron sobre ese grafo, y las líneas existentes que se supusieron eran dos, que unían las esquinas opuestas del grafo (que según en el análisis que se realiza en este trabajo contarían como 4, pues hay una ida y una vuelta para cada una). En la figura 5.3(b) se ve el resultado de correr todos los códigos sobre ese

grafo, y en la figura 5.3(c) se ven un segundo resultado obtenido luego de identificar y quitar manualmente 9 aristas. En esta última figura (con un poco de imaginación) se puede ver cuales serían los recorridos horario y antihorario. Este comentario se realiza sólo para mostrar que pese a todas las vertientes de errores que puede tener este modelo, había sido probado con (relativo) éxtio en una versión simplificada del problema.



(a) Grafo simplificado (b) Primer resultado (c) Recorridos obtenidos

Figura 5.3: Desempeño del modelo en un problema simplificado

Capítulo 6

Conclusión

Si bien no fue posible encontrar el recorrido de las líneas que se deseaban proponer sobre el grafo construido con datos reales, el modelo utilizado si pudo hacerlo (con alguna ayuda) sobre un grafo simplificado. Para llegar a la versión final del modelo utilizado y su implementación fue necesario tomar muchas decisiones que potencialmente podrían ser la causa del resultado obtenido. Tampoco debe perderse de vista la escala del problema, tanto en cantidad como en tipos de variables involucradas.

Cabe destacar la diversidad de herramientas matemáticas que son necesarias para intentar resolver problemas reales, aún de la manera simplificada que se encaró en este trabajo. Fue necesario utilizar herramientas provenientes de los algoritmos de búsqueda de rutas en redes, el concepto del equilibrio de Wardrop, que fue utilizado fuertemente a lo largo de todo el trabajo, y buena parte de los resultados relacionados con la programación no-lineal y los problemas de optimización, los que a su vez descansan sobre resultados generales de Álgebra y Análisis.

Problemas de este tipo suelen ser atacados por equipos interdisciplinarios durante bastante tiempo, llegando a veces a resultados parciales como este. Es claro que la escala del problema lo hace bastante desafiante, y suponer que puede ser resuelto con un modelo tan simplificado como el utilizado en este trabajo es cuando menos ambicioso.

No obstante, un modelo que devuelve resultados parciales como este puede servir como herramienta en la toma de decisiones de quienes diseñan los recorridos de colectivos en cualquier ciudad.

Una posibilidad podría ser interpretar que a mayor flujo en una arista, mayor sería el impacto positivo de que una nueva línea de colectivo la recorra. El razonamiento sería que son aristas que muchos usuarios encontraron conveniente recorrer, y que no pudieron hacerlo por una línea de colectivo existente y por eso lo hicieron en la parte del grafo que se corresponde con las calles.

Otra potencialidad del modelo puede ser medir el impacto de sacar una línea de transporte en un sistema de transporte dado. Es decir, supongamos que se desea

evaluar el impacto que tendría el cese de actividad de una línea. Se podrían calcular los flujos óptimos en el caso de que esa línea no exista y averiguar si son significativamente superiores en la parte de las calles con respecto al caso en el que esa línea esté operativa. Si los flujos no son significativamente diferentes, se podría concluir que esa línea es redundante con respecto al resto del sistema, es decir, que los usuarios pueden llegar de igual manera a su destino aún si ese recorrido no existiera. Y si ese fuera el caso, podría decidirse el cese de actividad en esa línea, sin que haya un impacto negativo en los usuarios.

Por último, resulta muy interesante como un modelo inicialmente pensado para estimar la matriz origen-destino de una red congestionada pudo ser adaptado al diseño de nuevos recorridos de colectivos, y como esa adaptación podría a su vez permitir aplicaciones que no se tuvieron en cuenta en un primer momento. Resulta evidente la importancia de la investigación en este tipo de problema por la utilidad que tiene para mejorar la calidad de vida en una sociedad, y por la riqueza matemática que esconden, en término de variedad de conocimientos que son necesarios para resolverlos.

6.1. Trabajo a futuro

Se hizo sobrada mención de que existían numerosos factores que potencialmente podrían mejorar el desempeño del modelo. Posibles trabajos a futuro podrían indagar sobre que impacto real tienen dichos factores sobre el resultado del modelo y mejorarlos de ser posible. A continuación se mencionan aquellos que fueron identificados y que podrían sugerir trabajos a futuro:

- **Sobre el modelo**
  - Los archivos exportados desde My Maps contenían tres coordenadas para cada punto, pero la tercera fue ignorada. Se podría construir un modelo que utilice esa tercera coordenada para situar diferentes partes del grafo a distintas alturas. Eso podría simplificar la construcción de grafos tan complejos como los que se necesitan para estos problemas, e incluso se podrían considerar diferentes alturas para diferentes modalidades de transporte (colectivos urbanos, colectivos interurbanos, a pie, etc). De esta forma se podría descartar las suposiciones que se hacen sobre las caminatas, e incluirlas en el modelo, con la restricción de los 300 m. Esto también permitiría omitir otras suposiciones que se hicieron a la hora de diseñar los algoritmos que buscan rutas, y aumentar la fidelidad del modelo a la realidad.
  - Se podrían ensayar otras elecciones para las funciones de costo por arista que cumplan mejor el objetivo que se desea, que es forzar a que los usuarios del sistema se desplacen 'juntos'por las aristas de las calles.

- Si se encuentra una forma rápida de calcular las matrices Δ y Γ (Ver *Sobre la implementación*) se podría modificar el modelo y convertirlo en problema binivel. El problema a considerar sería el de minimizar una función objetivo del tiempo, sobre el conjunto de recorridos posibles para la nueva línea, sujeto al equilibrio del usuario de Wardrop.
- Se podría cambiar la forma en la que se eligen los pares origen-destino, y/o los nodos de la UNC y examinar la sensibilidad del resultado a cambios en los criterios utilizados en este trabajo.
- Si se observa el resultado obtenido para el grafo pequeño que se mostró al final de la sección 5.4 puede notarse como el modelo intentó usar las mismas aristas para la ruta horaria y la antihoraria. Esto tiene su explicación en la forma de las funciones de costo. Es posible que la mejor ruta que se está buscando no sea aquella en la que los sentidos horario y anti-horario compartan la mayor cantidad posible de aristas. Una alternativa que se podría ensayar es dividir el problema en dos, y diseñar cada ruta por separado (posiblemente modificando varias de las suposiciones realizadas sobre el problema con las dos rutas, como la conformación de los pares origen-destino)

- **Sobre el problema de optimización**

  - El método elegido para resolver el problema de optimización itera sobre puntos factibles, es decir en el interior del conjunto factible. Existen métodos que iteran sobre puntos no factibles, es decir en el exterior del conjunto factible. Se tiene que como las funciones elegidas son exponenciales, aun para distancias relativamente pequeñas por fuera del conjunto factible, crecen rápidamente al punto de producir desbordamientos de memoria. Se pueden buscar variantes de la función objetivo que sin dejar de cumplir su objetivo, sean acotadas fuera del conjunto factible y permitan ser tratadas computacionalmente.
  - Se dijo que no se puede garantizar unicidad del mínimo para el problema de este trabajo, con lo cual es posible que cambiando la estimación inicial se obtengan distintos resultados. El estudio sobre cuál es el efecto de cambiar este punto inicial, y encontrar algún criterio para seleccionarlo podría representar una mejora significativa en el resultado. También se podría intentar utilizar una función objetivo convexa que conserve las propiedades de penalización deseadas, y de esa forma obtener unicidad del mínimo.
  - En [1] se menciona que en los métodos de descenso a veces es conveniente realizar un escalado del gradiente antes de realizar el paso. Dicho escalado es mejor mientras más "natural" sea en el contexto del problema

de optimización. En este caso fue necesario usar factores de escala inicial (parámetro s del método del gradiente proyectado) bastante grandes, pues de lo contrario el método demoraba demasiado en converger. Posiblemente sea posible encontrar un factor de escala ideal en algún sentido en el contexto del problema, en lugar de definir ese valor por prueba y error.

- **Sobre la implementación**

  - Las dos tareas que más trabajo computacional requieren (construir las matrices Δ y Γ y realizar las proyecciones del método del gradiente proyectado) son paralelizables. La primera consiste en calcular rutas para cada par origen-destino, y las búsqueda de rutas para un par es independiente de la búsqueda en otro par, por lo tanto son tareas que pueden realizarse en simultáneo sin afectar el resultado final. La segunda también es paralelizable por lo que se explicó en la sección 5.4. Cada proyección es descompuesta en muchas proyecciones más sencillas, que son independientes unas de las otras, y por ende también paralelizables. Esto podría permitir aprovechar un clúster y aumentar significativamente la cantidad de rutas calculadas, sin comprometer el tiempo que demoren las proyecciones.
  - En la implementación de la búsqueda de rutas se decidió primero encontrar la ruta más corta, y luego encontrar rutas "parecidas" a esa. Para la primera búsqueda se decidió utilizar el algoritmo A*, sin embargo podría no ser el mejor algoritmo de búsqueda para este tipo de grafo. Para la segunda búsqueda se decidió utilizar una búsqueda en profundidad. Este algoritmo encuentra nuevas rutas en un orden particular en el que las nuevas rutas suelen compartir el principio de su recorrido con las rutas anteriores. Como esa búsqueda se interrumpe con cotas de tiempo y cantidad de rutas, es muy posible que varias de las rutas encontradas sean iguales en la mayor parte de su recorrido y sólo cambien algunas cuadras antes de llegar a destino. Se puede estudiar cuanto impacta este hecho sobre el resultado obtenido, y en caso de un impacto significativo, se pueden buscar algoritmos que encuentren rutas en otro orden más conveniente.
  - En el cálculo de las proyecciones, se aprovecha la estructura de bloques de Γ, y luego se utiliza la función `qp` de Octave para resolver cada uno de los subproblemas. Ahora bien, cada uno de esos subproblemas a su vez tienen una estructura particular, pues la matriz de restricciones tiene todas sus entradas iguales. Probablemente sea posible explotar aún más la estructura particular del problema para encontrar implementaciones más veloces.

- **Sobre otros problemas**

  - Se dijo que este modelo podría no sólo servir para diseñar por completo un recorrido nuevo, sino para ayudar en el diseño del mismo, o para medir el impacto de alguna decisión. Es posible que puedan modificarse distintos elementos del modelo (las funciones de costo por arista, la forma de elegir los pares origen-destino, la forma de calcular las rutas, etc) para adaptarlos a cada uno de esos otros problemas específicos, ya que este modelo fue pensado particularmente desde el enfoque de diseñar una nueva ruta.

Agradecimientos

A mi director, por las varias jornadas que invirtió en ayudarme a dilucidar varios de los aspectos del problema. A mi co-director, por apuntarnos en la dirección adecuada con respecto al uso del equilibrio de Wardrop y las ventajas que eso suponía, que fueron en definitiva una parte fundamental de este trabajo.

A mi familia, por su apoyo incondicional para que pueda terminar la carrera sin tener que preocuparme por otra cosa que estudiar.

A mis amigos, en especial a Mariano por ayudarme con los códigos en Python y a Mauricio por ayudarme con la tediosa tarea de cargar los archivos de los recorridos de las líneas de colectivos existentes.

A La Bisagra en general, y a El GURI en particular, por enseñarme a pensar mi paso por la universidad no sólo desde el lado académico, sino también desde el lado humano y social, y a valorar el impacto que tiene el conocimiento en la construccion de una sociedad más justa, libre y soberana.

Al Estado, por poder garantizar una educación superior gratuita y de calidad, y por avanzar en la construcción de mejor un sistema científico, a donde quienes somos entusiastas de la matemática no nos sintamos inclinados a estudiar otras carreras similares por miedo a no tener una salida laboral. Y por extensión a la sociedad en su conjunto, que hace posible la educación gratuita con sus impuestos, y no siempre ve los beneficios que eso debería suponer. Espero que este trabajo sirva como humilde aporte para devolverle a la sociedad una parte de todo lo que inviritió en mi educación.

Bibliografía

[1] Dimitri P Bertsekas. *Nonlinear programming*. Athena scientific, 1999.

[2] Sebastián Luis Bonino, María Gabriela Capdevila, Mariela González, y María Celeste Comes Brunetto. Mecanismo de gestión territorial: propuesta para mejorar el acceso a la educación pública superior a través del nuevo sistema de transporte urbano masivo de pasajeros. Presentado en: X Bienal del coloquio de transformaciones territoriales: Desequilibrios regionales y políticas públicas, una agenda pendiente, 2014.

[3] Stella C Dafermos y Frederick T Sparrow. The traffic assignment problem for a general network. *Journal of Research of the National Bureau of Standards, Series B*, 73(2):91–118, 1969.

[4] Damián Fernández Ferreyra. Curso de optimización no lineal - famaf, 2014. http://www.famaf.unc.edu.ar/~dfernandez/optNoLineal/.

[5] William Prager. *Problems of traffic and transportation*. 1954.

[6] Tom Reinhold. More passengers and reduced costs—the optimization of the berlin public transport network. *Journal of Public Transportation*, 11(3):4, 2008.

[7] Jorgelina Walpen. *Sobre la resolución del problema de ajustar la matriz Origen Destino en una red de tráfico vehicular congestionada*. Tesis Doctoral, Universidad Nacional de Rosario, 2015.

Apéndice A

Códigos Fuente

Todos los códigos que se mencionan se encuentran disponibles en la URL:

https://github.com/njares/TF_Diseno

Apéndice B

Otros archivos generados

B.1. Líneas cargadas: archivo `lineas_hr`

En la primer columna están los nombre de cada uno de los archivos generados. En la segunda están los nombres de los recorridos de las líneas de colectivos que representa. En la tercera hay una descripción de las calles por las que va ese recorrido.

```
1  linea_1   10 IDA          Illia, Chacabuco, 27 de Abril,
2            11 VUELTA       Fragueiro, Sarmiento
3            12 IDA
4  linea_2   10 VUELTA       Castro Barros, Avellaneda, Colón, Gral
5            11 IDA          Paz, Illia
6            12 VUELTA
7            14 IDA
8            17 IDA
9            17 IDA ESP
10 linea_3   13 VUELTA       Catro Barros, Avellaneda, Colón, Gral
11           19 VUELTA       Paz, Hirigoyen, Valparaiso, Filloy,
12           19 VUELTA ESP   Maestro Lopez, Cruz Roja
13 linea_4   14 VUELTA       Illia, Balcarce, 27 de Abril,
14                           Fragueiro, Sarmiento
15 linea_5   15 IDA          Avellaneda, Colón
16 linea_6   15 VUELTA       San Jerónimo, Corro, Sarmiento
17 linea_7   16 IDA          Saravia, Nores, Cruz Roja, Maestro
18           16 ESP IDA      Lopez, Richardson, Belgrano, Colón
19           66 IDA
20           66 ESP IDA
21 linea_8   16 VUELTA       San Jerónimo, Velez Sarsfield,
22           16 ESP VUELTA   Hirigoyen, los nogales, Medina
23                           Allende, Cruz Roja
24 linea_9   17 VUELTA       Illia, Chacabuco, San Jeronimo,
```

```
          17 VUELTA ESP  27 de Abril, Fragueiro, Humberto
                         Primo
linea_10  18 IDA         Gral Paz, Velez Sarfield, Yrigoyen,
                         Valaparaiso, Juan Filloy, Maestro
                         Marcelo Lopez, Cruz Roja Argentina
linea_11  18 VUELTA      De la Industria, Nores Martinez, Cruz
                         Roja, Marcelo López, Juan Filloy,
                         Valparaíso, Chacabuco, San Jeronimo, 27
                         de Abril, Jujuy
linea_12  19 IDA         De la Industria, Nores Martinez, Cruz
          19 IDA ESP     Roja, Marcelo López, Juan Filloy,
          13 IDA         Valparaíso, Chacabuco, San Jeronimo, 27
                         de Abril, Corro, Fragueiro, Humberto
                         Primo
linea_13  20 IDA         Rogelio Martinez, Cruz Roja, Marcelo
                         López, Medina Allende, Haya de la
                         Torre, Enrique Barros, Los Nogales,
                         Valparaíso, Chacabuco, San Jeronimo, 27
                         de Abril, Paraguay, Colón, Sta Fe
linea_14  20 VUELTA      Sta Fe, Colón, Salta, Obispo
                         Salgueiro, Lugones, Valparaíso, Los
                         Nogales, Enrique Barros, Haya de la
                         Torre, Medina Allende, Marcelo López,
                         Cruz Roja, Valparaiso
linea_15  21 IDA         Richieri, Concepción Arenales,
          23 VUELTA      Valparaíso, Chacabuco, San Jeronimo, 27
                         de Abril, Paraguay, Colón, Sta Fe
linea_16  21 VUELTA      Sta Fe, Colón, Velez Sarsfield,
          23 IDA         Hirigoyen, Valparaíso, Concepción
                         Arenales, Richieri
linea_17  22 IDA         Javier Diaz, Bellardinelli, Cruz
                         Roja, Marcelo López, Medina Allende,
                         Haya de la torre, Enrique Barros, Los
                         Nogales, Valparaíso, Chacabuco, San
                         Jeronimo, 27 de Abril, Paraguay,
                         Colón, Sta Fe
linea_18  22 VUELTA      Sta Fe, Colon, Velez Sarsfield,
                         Hirigoyen, Valparaiso, Los Nogales,
                         Enrique Barros, Haya de la Torre,
                         Medina Allende, Marcelo Lopez, Cruz
                         Roja, Valparaiso
linea_19  24 IDA         Sta Fe, Colon, Salta, Obispo
                         Salguero, Lugones, Valparaiso
linea_20  24 VUELTA      Nores Martinez, Haya de la Torre,
```

```
                                Enrique Barros, Los Nogales,
                                Valparaiso, Chacabuco, San Jeronimo,
                                27 de Abril, Paraguay, Colon, Sta Fe
linea_21  25  IDA               Peron, San Jeronimo, 27 de Abril,
          27  IDA               Paraguay, Colon, Sta Fe
linea_22  25  VUELTA            Sta Fe, Colon, Olmos
          27  VUELTA
linea_23  26  IDA               Bellardinelli, Cruz roja, Marcelo
                                López, Medina Allende, Haya de la
                                Torre, Enrique Barros, Los Nogales,
                                Valparaíso, Chacabuco, San Jeronimo, 27
                                de Abril, Paraguay, Colón, Sta Fe
linea_24  26  VUELTA            Sta Fe, Colon, Velez Sarfield, Illia,
                                Ituzaingo, Yrigoyen, Valparaiso, Los
                                Nogales, Enrique Barros, Haya de la
                                Torre, Medina Allende, Marcelo lopez,
                                Cruz roja, Bellardinelli
linea_25  28  IDA               Sta Fe, Colon, Velez Sarfield,
                                Hirigoyen, Valparaiso, Concepcion
                                Arenales, Richieri, Julio Argentino
                                Roca, Dante
linea_26  28  VUELTA            Julio Roca, Richieri, Concepción
                                Arenales, Valparaíso, Chacabuco, San
                                Jeronimo, 27 de Abril, Paraguay,
                                Colón, Sta Fe
linea_27  29  IDA               Peron, San Jeronimo, Velez Sarfield,
                                Hirigoyen, Valparaiso
linea_28  29  VUELTA            Rogelio Martinez, Haya De la torre,
                                Enrique Barros, Los Nogales,
                                Valparaiso, Chacabuco, Illia, Alvear,
                                Simon Bolivar, Colon, Olmos
linea_29  30  IDA               Velez Sarsfield, Belgrano, Tucuman
          33  IDA
          51  IDA
linea_30  30  VUELTA            Gral Paz, Velez Sarfield
          33  VUELTA
linea_31  31  IDA               Peron, San Jeronimo, Velez Sarfield,
                                Richardson, Sta Maria, Corro
linea_32  31  VUELTA            Corro, Naciones Unidas, Friuli,
                                Ernesto Romagosa, Santiago Caceres,
                                Alejandro centeno, Ayacucho, Santiago
                                Caceres, Belgrano, Colon
linea_33  32  IDA               Naciones Unidas, Cruz Roja, Marcelo
                                Lopez, Medina Allende,Richardson,
```

```
                                Ambrosio Olmos, Chacabuco, 27 de
                                Abril, Belgrano, Tucuman
linea_34 32 VUELTA              Gral Paz, Velez Sarfield, Pueyrredon,
                                Independencia, Medina Allende,
                                Marcelo Lopez,Cruz Roja, Friuli,
                                Naciones Unidas
linea_35 34 IDA                 Velez Sarfield, Ambrosio Olmos,
                                Chacabuco, San Jeronimo, 27 de Abril,
                                Belgrano, Colon
linea_36 34 VUELTA              Gral Paz, Velez Sarfield, Hirigoyen,
                                Ambrosio Olmos, Velez Sarfield
linea_37 35 IDA                 Elpidio Gonzalez, Marcelo T de
                                Alvear, Belgrano, 27 de Abril,
                                Paraguay, Colon, Sta Fe
linea_38 36 IDA                 Velez Sarfield, Belgrano, 27 de
                                Abril, Paraguay, Colon, Sta Fe
linea_39 36 VUELTA              Sta Fe, Colon, Gral Paz, Velez
                                Sarsfield
linea_40 40 VUELTA              Gral Paz, Velez Sarfield, Duarte
         42 IDA                 Quiros
linea_41 41 IDA                 Gral Paz, Velez Sarfield, Hirigoyen,
                                Valparaiso
linea_42 41 VUELTA              Marcelo Lopez, Haya de la torre,
                                Enrique Barros, Valparaiso,
                                Chacabuco, Maipu
linea_43 42 VUELTA              Duarte Quiros, Misiones, Caseros,
                                Mariano Moreno, Rodriguez Peña, Colon,
                                Emilio Olmos, Salta, 25 de Mayo,
                                Maipu
linea_44 43 IDA                 Batalla de Cepeda, Caseros, Velez
                                Sarsfield, Bv San Juan, Illia
linea_45 43 VUELTA              Peron, San Jeronimo, 27 de Abril,
                                Paso de los Andes, Bv San Juan
linea_46 44 IDA                 Gral Paz, 27 de Abril, Paso de los
                                Andes, Bv San Juan, Montevideo
linea_47 44 VUELTA              Batalla de cepeda, Caseros, Corro,
                                Colon, Emilio Olmos, Salta, 25 de
                                Mayo, Maipu
linea_48 43 UNIV IDA            Batalla de Cepeda, Caseros, Velez
                                Sarsfield, Yrigoyen, Valparaiso, Haya
                                de la Torre
linea_49 43 UNIV VUELTA         Haya de la Torre, Enrique Barros,
                                Valparaiso, Chacabuco, 27 de Abril,
                                Paso de los Andes, Bv San Juan
```

```
linea_50 45 IDA          Colon, Velez Sarfield, Hirigoyen,
                         Valparaiso
linea_51 45 VUELTA       Cruz Roja, Marcelo Lopez, Haya de la
                         Torre, Enrique Barros,Los Nogales,
                         Chacabuco, 27 de Abril, Rodriguez
                         Peña, Colon
linea_52 50 IDA          Velez Sarsfield
         51 IDA
         53 VUELTA
         TA IDA
linea_53 50 VUELTA       Velez Sarsfield, Belgrano, Colón,
                         Salta, 25 de Mayo, Chacabuco
linea_54 52 IDA          Velez Sarsfield, Richardson, Medina
                         Allende, Cruz Roja, Friuli, Velez
                         Sarsfield
linea_55 52 VUELTA       Velez Sarsfield, Friuli, Cruz Roja,
                         Maestro Lopez, Richardson, Ambrosio
                         Olmos, Chacabuco
linea_56 54 IDA          Velez Sarsfield, Illia
linea_57 54 VUELTA       Peron, 27 de Abril, Jujuy
linea_58 60 IDA          San Jerónimo, Paso de los Andes,
         62 IDA          Pueyrredon
linea_59 60 VUELTA       Pueyrredón, Paso de los andes, Funes,
         62 VUELTA       Mariano Moreno, Colón
linea_60 61 IDA          Gral Paz, Achaval Rodriguez, Alvear,
         65 IDA          Roca, Elpidio Gonzalez
linea_61 61 VUELTA       Elpidio Gonzalez, Roca, Belgrano,
                         Colón
linea_62 63 VUELTA       Pueyrredon, Arturo Bas, Peredo,
                         Belgrano, Colón
linea_63 64 IDA          San Jerónimo, Cañada, Pueyrredón
linea_64 64 VUELTA       Pueyrredon, Arturo Bas, Peredo,
                         Belgrano, Colón
linea_65 65 VUELTA       Roca, Pueyrredon, Chacabuco
linea_66 66 VUELTA       Gral Paz, Hirigoyen, Los Nogales,
         66 ESP VUELTA   Medina Allende, Cruz Roja
linea_67 67 IDA          De la Industria, Nores, Cruz Roja,
                         Maestro Lopez, Filloy, Valparaiso,
                         Chacabuco, 27 de Abril, Paso de los
                         Andes, Pueyrredon
linea_68 67 VUELTA       Pueyrredon, Paso de los Andes, Funes,
                         Mariano Moreno, Caseros, Velez
                         Sarsfield, Hirigoyen, Valparaiso,
                         Filloy, Marcelo Lopez, Cruz Roja
```

```
linea_69 68 VUELTA      Sucre, Colon
linea_70 70 IDA         Perón, San Jeronimo, Rodriguez Peña,
         74 IDA         Colón, Remonda
         75 VUELTA
linea_71 70 VUELTA      Colon
         72 IDA
         74 VUELTA
         75 IDA
linea_72 71 IDA         Belardinelli, Cruz Roja, Marcelo
                        Lopez, Haya de la torre, Enrique
                        Barros, Los Nogales, Chacabuco, 27 de
                        Abril, Mendoza, Duarte Quiros
linea_73 71 VUELTA      Duarte Quiros, Misiones, Caseros,
                        Velez, Hirigoyen, Los Nogales,
                        Enrique Barros, Haya de la Torre,
                        Medina Allende, Cruz Roja,
                        Belardinelli
linea_74 72 VUELTA      Rosario de Santa Fe, Salguero, San
                        Jerónimo, Rodriguez Peña, Colón
linea_75 73 IDA         Belardinelli, Cruz Roja, Marcelo
                        Lopez, Haya de la Torre, Enrique
                        Barros, Los Nogales, Chacabuco,
                        Illia, Belgrano, Colón
linea_76 73 VUELTA      Peron, San Jerónimo, Velez, Hirigoyen,
                        Los Nogales, Enrique Barros, Haya de
                        la Torre, Medina Allende, Cruz Roja,
                        Belardinelli, Inchauspe
linea_77 80 IDA         Illia, Chacabuco, 27 de abril,
                        Rodriguez Peña, Colón
linea_78 80 VUELTA      Colon, Salguero
linea_79 81 IDA         Javier Diaz, Belardinelli, Cruz Roja,
                        Marcelo Lopez, Haya de la Torre,
                        Enrique Barros, Los Nogales,
                        Chacabuco, 27 de Abril, Rodriguez
                        Peña, Colón
linea_80 81 VUELTA      Colon, Velez Sarsfield, Hirigoyen,
                        Valparaiso, Haya de la Torre, Medina
                        Allende, Cruz Roja, Belardinelli,
                        Javier Diaz
linea_81 82 IDA         Illia, Chacabuco, 27 de Abril,
                        Paraguay, Colón
linea_82 82 VUELTA      Colon, Gral Paz, Illia
linea_83 83 IDA         Cruz Roja, Nores, Haya de la Torre,
                        Valparaiso, Chacabuco,
```

```
linea_84 83 VUELTA      Castro Barros, Avellaneda, Colon,
                        Velez Sarsfield, Hirigoyen,
                        Valparaiso, Cruz Roja
linea_85 84 VUELTA      Colon
linea_86 B80 IDA        Perón
linea_87 TA VUELTA      Belgrano
linea_88 TB IDA         Colón
linea_89 TB VUELTA      Santa Rosa, Avellaneda, Colón
linea_90 TC IDA         Roca, Belgrano, Colón
linea_91 TC VUELTA      Lima, Gral Paz, Achaval Rodriguez,
                        Cañada, Roca
```

B.2. Archivo `grafo-terminado.kml`

Las líneas donde están las coordenadas, como por ejemplo la línea 15, son demasiado largas para la página y están cortadas por el ajuste de linea. Por este motivo no se aprecia que entre las tres primeras coordenadas y las tres segundas coordenadas hay un espacio que las separa.

```
<?xml version='1.0' encoding='UTF-8'?>
<kml xmlns='http://www.opengis.net/kml/2.2'>
    <Document>
        <name>grafo-terminado</name>
        <description><![CDATA[]]></description>
        <Folder>
            <name>Capa 1</name>
            <Placemark>
                <name>Línea 1</name>
                <styleUrl>#line-000000-1-nodesc</styleUrl>
                <ExtendedData>
                </ExtendedData>
                <LineString>
                    <tessellate>1</tessellate>
                    <coordinates>-64.204761,-31.406126,0.0
-64.205668,-31.405856,0.0</coordinates>
                </LineString>
            </Placemark>

...

            <Placemark>
                <name>Línea 3071</name>
                <styleUrl>#line-000000-1-nodesc</styleUrl>
                <ExtendedData>
```

```
30712                   </ExtendedData>
30713                   <LineString>
30714                       <tessellate>1</tessellate>
30715                       <coordinates>-64.187209,-31.421847,0.0
    -64.186828,-31.422592999999996,0.0</coordinates>
30716                   </LineString>
30717               </Placemark>
30718           </Folder>
30719           <Style id='line-000000-1-nodesc-normal'>
30720               <LineStyle>
30721                   <color>ff000000</color>
30722                   <width>1</width>
30723               </LineStyle>
30724               <BalloonStyle>
30725                   <text><![CDATA[<h3>$[name]</h3>]]></text>
30726               </BalloonStyle>
30727           </Style>
30728           <Style id='line-000000-1-nodesc-highlight'>
30729               <LineStyle>
30730                   <color>ff000000</color>
30731                   <width>2.0</width>
30732               </LineStyle>
30733               <BalloonStyle>
30734                   <text><![CDATA[<h3>$[name]</h3>]]></text>
30735               </BalloonStyle>
30736           </Style>
30737           <StyleMap id='line-000000-1-nodesc'>
30738               <Pair>
30739                   <key>normal</key>
30740                   <styleUrl>#line-000000-1-nodesc-normal</
    styleUrl>
30741               </Pair>
30742               <Pair>
30743                   <key>highlight</key>
30744                   <styleUrl>#line-000000-1-nodesc-highlight</
    styleUrl>
30745               </Pair>
30746           </StyleMap>
30747       </Document>
30748   </kml>
```

B.3. Archivo de estilos para los mapas `kml`

```
<Style id='line-000000-1-nodesc-normal'>
    <LineStyle>
        <color>ff000000</color>
        <width>1</width>
    </LineStyle>
    <BalloonStyle>
        <text><![CDATA[<h3>$[name]</h3>]]></text>
    </BalloonStyle>
</Style>
<Style id='line-000000-1-nodesc-highlight'>
    <LineStyle>
        <color>ff000000</color>
        <width>2.0</width>
    </LineStyle>
    <BalloonStyle>
        <text><![CDATA[<h3>$[name]</h3>]]></text>
    </BalloonStyle>
</Style>
<StyleMap id='line-000000-1-nodesc'>
    <Pair>
        <key>normal</key>
        <styleUrl>#line-000000-1-nodesc-normal</styleUrl>
    </Pair>
    <Pair>
        <key>highlight</key>
        <styleUrl>#line-000000-1-nodesc-highlight</styleUrl>
    </Pair>
</StyleMap>
<Style id='line-000000-2-nodesc-normal'>
    <LineStyle>
        <color>ff0000ff</color>
        <width>1.5</width>
    </LineStyle>
    <BalloonStyle>
        <text><![CDATA[<h3>$[name]</h3>]]></text>
    </BalloonStyle>
</Style>
<Style id='line-000000-2-nodesc-highlight'>
    <LineStyle>
        <color>ff0000ff</color>
        <width>2.5</width>
    </LineStyle>
    <BalloonStyle>
        <text><![CDATA[<h3>$[name]</h3>]]></text>
```

```
45      </BalloonStyle>
46  </Style>
47  <StyleMap id='line-000000-2-nodesc'>
48      <Pair>
49          <key>normal</key>
50          <styleUrl>#line-000000-2-nodesc-normal</styleUrl>
51      </Pair>
52      <Pair>
53          <key>highlight</key>
54          <styleUrl>#line-000000-2-nodesc-highlight</styleUrl>
55      </Pair>
56  </StyleMap>
57  <Style id='line-000000-3-nodesc-normal'>
58      <LineStyle>
59          <color>ff0033cc</color>
60          <width>2</width>
61      </LineStyle>
62      <BalloonStyle>
63          <text><![CDATA[<h3>$[name]</h3>]]></text>
64      </BalloonStyle>
65  </Style>
66  <Style id='line-000000-3-nodesc-highlight'>
67      <LineStyle>
68          <color>ff0033cc</color>
69          <width>3</width>
70      </LineStyle>
71      <BalloonStyle>
72          <text><![CDATA[<h3>$[name]</h3>]]></text>
73      </BalloonStyle>
74  </Style>
75  <StyleMap id='line-000000-3-nodesc'>
76      <Pair>
77          <key>normal</key>
78          <styleUrl>#line-000000-3-nodesc-normal</styleUrl>
79      </Pair>
80      <Pair>
81          <key>highlight</key>
82          <styleUrl>#line-000000-3-nodesc-highlight</styleUrl>
83      </Pair>
84  </StyleMap>
85  <Style id='line-000000-4-nodesc-normal'>
86      <LineStyle>
87          <color>ff006699</color>
88          <width>2.5</width>
```

```
89      </LineStyle>
90      <BalloonStyle>
91          <text><![CDATA[<h3>$[name]</h3>]]></text>
92      </BalloonStyle>
93  </Style>
94  <Style id='line-000000-4-nodesc-highlight'>
95      <LineStyle>
96          <color>ff006699</color>
97          <width>3.5</width>
98      </LineStyle>
99      <BalloonStyle>
100         <text><![CDATA[<h3>$[name]</h3>]]></text>
101     </BalloonStyle>
102 </Style>
103 <StyleMap id='line-000000-4-nodesc'>
104     <Pair>
105         <key>normal</key>
106         <styleUrl>#line-000000-4-nodesc-normal</styleUrl>
107     </Pair>
108     <Pair>
109         <key>highlight</key>
110         <styleUrl>#line-000000-4-nodesc-highlight</styleUrl>
111     </Pair>
112 </StyleMap>
113 <Style id='line-000000-5-nodesc-normal'>
114     <LineStyle>
115         <color>ff009966</color>
116         <width>3</width>
117     </LineStyle>
118     <BalloonStyle>
119         <text><![CDATA[<h3>$[name]</h3>]]></text>
120     </BalloonStyle>
121 </Style>
122 <Style id='line-000000-5-nodesc-highlight'>
123     <LineStyle>
124         <color>ff009966</color>
125         <width>4</width>
126     </LineStyle>
127     <BalloonStyle>
128         <text><![CDATA[<h3>$[name]</h3>]]></text>
129     </BalloonStyle>
130 </Style>
131 <StyleMap id='line-000000-5-nodesc'>
132     <Pair>
```

```
133             <key>normal</key>
134             <styleUrl>#line-000000-5-nodesc-normal</styleUrl>
135         </Pair>
136         <Pair>
137             <key>highlight</key>
138             <styleUrl>#line-000000-5-nodesc-highlight</styleUrl>
139         </Pair>
140     </StyleMap>
141     <Style id='line-000000-6-nodesc-normal'>
142         <LineStyle>
143             <color>ff00cc33</color>
144             <width>3.5</width>
145         </LineStyle>
146         <BalloonStyle>
147             <text><![CDATA[<h3>$[name]</h3>]]></text>
148         </BalloonStyle>
149     </Style>
150     <Style id='line-000000-6-nodesc-highlight'>
151         <LineStyle>
152             <color>ff00cc33</color>
153             <width>4.5</width>
154         </LineStyle>
155         <BalloonStyle>
156             <text><![CDATA[<h3>$[name]</h3>]]></text>
157         </BalloonStyle>
158     </Style>
159     <StyleMap id='line-000000-6-nodesc'>
160         <Pair>
161             <key>normal</key>
162             <styleUrl>#line-000000-6-nodesc-normal</styleUrl>
163         </Pair>
164         <Pair>
165             <key>highlight</key>
166             <styleUrl>#line-000000-6-nodesc-highlight</styleUrl>
167         </Pair>
168     </StyleMap>
169     <Style id='line-000000-7-nodesc-normal'>
170         <LineStyle>
171             <color>ff00ff00</color>
172             <width>4</width>
173         </LineStyle>
174         <BalloonStyle>
175             <text><![CDATA[<h3>$[name]</h3>]]></text>
176         </BalloonStyle>
```

```
177 </Style>
178 <Style id='line-000000-7-nodesc-highlight'>
179     <LineStyle>
180         <color>ff00ff00</color>
181         <width>5</width>
182     </LineStyle>
183     <BalloonStyle>
184         <text><![CDATA[<h3>$[name]</h3>]]></text>
185     </BalloonStyle>
186 </Style>
187 <StyleMap id='line-000000-7-nodesc'>
188     <Pair>
189         <key>normal</key>
190         <styleUrl>#line-000000-7-nodesc-normal</styleUrl>
191     </Pair>
192     <Pair>
193         <key>highlight</key>
194         <styleUrl>#line-000000-7-nodesc-highlight</styleUrl>
195     </Pair>
196 </StyleMap>
```