UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

INGENIERÍA EN INFORMÁTICA

PROYECTO FIN DE CARRERA

Un algoritmo de descomposición simplicial desagregada con estrategias de identificación de pares (DSD-P) para el problema generalizado de asignación de tráfico

Autor: **Carlos Blanco Bueno**

Director: **Ricardo García Ródenas**

Coordinador: **Macario Polo Usaola**

Septiembre, 2006

Fecha de defensa:

Ciudad Real, día de de 2006

Tribunal:

Presidente D.

Vocal D.

Vocal D.

Calificación:

El Presidente Vocal Vocal

Agradecimientos

En especial a la Consejería de Educación y Ciencia de la Junta de Comunidades de Castilla La Mancha por la financiación del proyecto de investigación (Ref: 2005-COB-697): "*Desarrollo, implementación y validación de una metodología para la estimación de matrices de viajes origen-destino a partir de volúmenes de tráfico y matrices desactualizadas. Parte II: Estudio computacional*", en cuyo marco ha sido posible el desarrollo de este Proyecto Fin de Carrera.

También quería agradecer a mi director, Ricardo García Ródenas, el apoyo y el interés mostrado, ya que sin él no hubiera sido posible el desarrollo de este proyecto.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

Podemos ver como en los últimos años el tráfico urbano está incrementándose en gran medida, algo que repercute directamente en la calidad de vida del ciudadano. Hay una mayor preferencia por los vehículos privados frente al transporte público, aumentando la cantidad de vehículos por familia y el número de desplazamientos realizados.

Otro factor que influye en la cantidad de desplazamientos y la amplitud de estos es el modelo de ciudad difuso con el que nos encontramos en la actualidad y que tiende a separar funcionalmente población, comercio y servicios hacia la periferia y zona industrial en el exterior.

Esta gran cantidad de desplazamientos provocan una saturación de la red, una congestión, que influye negativamente en los ciudadanos. Si hablamos en términos medioambientales, según la Agencia Europea de Medioambiente, el tráfico urbano produce más del 70 % de las emisiones de monóxido de carbono, del 50 % de óxidos de nitrógeno y del 33 % de hidrocarburos. Según la Organización Mundial de la Salud, entre un 70 y 80 % de las ciudades europeas con más de 500.000 habitantes, no reúnen las condiciones mínimas de calidad atmosférica.

Toda esta contaminación influye en la salud de los ciudadanos, ya no solo la contaminación atmosférica, sino también la acústica o el estrés derivado de los atascos. Con la intención de evitar estos problemas aparecen algunas medidas, entre las que podemos mencionar un mayor uso del transporte público, políticas de aparcamiento o peatonalización de los centros urbanos.

1.1. Problema de asignación de tráfico

Los planificadores y gestores de los sistemas de transporte buscan resolver estos problemas, intentando responder de forma adecuada a las necesidades de movilidad de la población. La planificación, gestión y control del tráfico emplea modelos matemáticos como herramienta analítica que auxilian la toma de decisiones. Estos modelos deben resolverse por medio de métodos computacionales y de forma eficiente.

El proceso de planificación del transporte consta de las siguientes fases que pasamos a enumerar:

- Recopilación de datos.
- Análisis y ajuste de modelos.
  - Modelos de generación/atracción de tráfico.
  - Modelos de distribución zonal.
  - Modelos de distribución modal.
  - Modelos de asignación.
- Previsiones de la demanda.
- Evaluación de futuros escenarios.

Uno de los modelos más ampliamente usado es el modelo de asignación de tráfico. El Problema de Asignación de Tráfico (TAP) modeliza el comportamiento de los usuarios en la elección de su ruta para satisfacer su viaje en redes de tráfico con congestión partiendo de la información de la configuración de la red y de la demanda (matriz origen-destino).

Los modelos que integran la fase de asignación en redes congestionadas modelizan un equilibrio e Cournot-Nash, en la que los usuarios del sistema de transporte son los jugadores del juego de equilibrio, y en el que sus estrategias pueden consistir en la elección de la ruta, el modo de transporte, la realización o no del viaje, etc.

Este PFC desarrolla e implementa un algoritmo computacional para resolver el problema generalizado de asignación de tráfico.

1.2. Importancia del trabajo

Unos datos que pueden describir el interés suscitado entre los investigadores y profesionales es que en Google *(www.google.com)* la entrada 'traffic assignment problem' genera más de 8.000.000 de documentos. La monografía de Patriksson [Patriksson1994] recoge más de 1.000 artículos referidos al citado problema. Una búsqueda en Scopus *(www.scopus.com)* genera más de 1.000 artículos de investigación y más de 4.000 patentes relacionadas con el tema. Finalmente citar que en el proyecto de investigación más importante a nivel europeo (por lo menos bajo el punto de vista presupuestario), el proyecto GALILEO van a aplicarse en gestión y control de tráfico en un contexto dinámico.

El TAP comenzó a estudiarse en la década de los 50, del siglo pasado, siendo muy difícil hoy en día mejorar los algoritmos y modelos existentes o aportar algo novedoso y relevante.

El Estado-del-Arte en la resolución del Problema de Asignación de Tráfico Simétrico lo constituye el algoritmo de Bar-Gera y Boyce desarrollado en su tesis doctoral (ver [Bar-Gera and Boyce2002]).

Este algoritmo presenta una velocidad de convergencia superlineal, pero únicamente es aplicable al modelo separable que se formula como un problema de optimización. Hoy por hoy no se ha aplicado al modelo general de asignación de tráfico que es formulado como un problema de desigualdades variacionales.

Con este PFC se ha intentado cubrir este aspecto desarrollando un algoritmo de descomposición simplicial desagregada. Un aspecto destacable frente a la mayoría de los métodos de resolución es que opera en el espacio de flujo en los caminos en lugar de flujo en los arcos.

Muchos de problemas de planificación de tráfico, como regulación semafórica, determinación de peajes urbanos, se formulan mediante programación matemática con restricciones de equilibrio, también llamada binivel, y aunque existen multitud de resultados teóricos, no hay una aplicación directa al campo de la ingeniería del transporte. Al aplicarlos a problemas reales se nos presentan varias dificultades como el gran número de variables que hemos de manejar y las malas propiedades de los modelos, como la no diferenciabilidad y la no convexidad. Estos modelos emplean como submodelo al TAP y la solución de estos modelos binivel pasa por la solución eficiente del TAP.

Capítulo 2

Objetivos

2.1. Objetivos del proyecto

El objetivo principal de este proyecto es el desarrollo, implementación y estudio computacional de un algoritmo con velocidad de convergencia superlineal para el problema de asignación de tráfico (TAP) asimétrico, que sea aplicable a redes de grandes dimensiones.

Como objetivo secundario buscamos la creación de una toolbox de MATLAB, en la que se integren y documenten los algoritmos realizados.

2.2. Hipótesis de trabajo

Para establecer nuestra estrategia computacional nos basamos en la observación crucial de que en una red en equilibrio, el número de pares origen-destino que tienen más de un camino con flujo positivo es muy reducido, del orden de un 10 %. Esto quiere decir que casi todas las demandas se van a satisfacer asignando su flujo por un sólo camino. Este hecho está avalado por resultados numéricos desarrollados por Ricardo García y Bar-Gera.

Hemos comentado que uno de los grandes inconvenientes con los que nos encontramos a la hora de abordar un problema de asignación de tráfico asimétrico es la dimensionalidad de éste, ya que tenemos que manejar un gran número de variables para buscar la situación de equilibrio (flujo en los caminos y en los arcos de la red).

Si quisiéramos aplicar el método de Newton para resolver en desigualdad variacional que describe el problema de asignación de tráfico generalizado se requeriría resolver sistemas de ecuaciones lineales del orden del número de variables del problema.

El número de caminos en una red (grafo) crece exponencialmente con su tamaño. Lo que hace inviable la enumeración exhaustiva de loas variables incluso para redes de pequeño tamaño.

En este PFC se han aplicado dos estrategias computacionales para resolver los problemas anteriores:

Método de descomposición simplicial. Esto permite evitar la enumeración exhausti-

va de los caminos de la red y se van generando conforme éstos van siendo necesitados.

Estrategia de identificación de pares. De acuerdo a la hipótesis del trabajo solamente un pequeño número de pares requieren ser considerados en la situación de equilibrio.

Ambas estrategias reducirán drásticamente la dimensionalidad del problema y pueden aplicarse al método de Newton.

Para realizar el equilibrado según nuestra estrategia, deberemos calcular un gran número de sistemas de ecuaciones lineales, que a priori parecería que no podríamos ser capaces de resolver en unos tiempos aceptables computacionalmente. Es por esto por lo que hemos realizado pruebas previas en MATLAB con sistemas de distinta dimensionalidad (nxn), tal y como podemos ver en la tabla 2.1.

Tabla 2.1: Resultados computacionales LS en MATLAB

n	Tiempo (seg)
100	0.002
500	0.122
1000	0.866
2000	4.768
3000	16.562
4000	39.703
5000	87.067

Si consideramos todos los pares sería inabordable (por ejemplo la red de Barcelona tiene unos 8000). En nuestro, siguiendo la hipótesis planteada, nos quedamos en la práctica con aproximadamente el 7 % de los pares, considerando por cada uno unas tres variables (V_{p1}, V_{p2}, u). Por ejemplo una red de 10000 pares se quedaría con 2100 variables, en la que MATLAB emplearía unos 5 seg en resolver el LS. Estos resultados nos permiten seguir adelante con nuestra idea.

2.3. Restricciones

Factores dato

El entorno del algoritmo a desarrollar sería el de una consultoría en ingeniería de transporte. Éste debe poder ejecutarse en distintos sistemas operativos, por lo que el lenguaje utilizado ha de tener características multiplataforma.

Factores estratégicos

La elección del lenguaje de programación utilizado para la implementación del algoritmo contaba con dos alternativas principales: GAMS y MATLAB, habiéndonos decantado por MATLAB.

Con respecto a GAMS podemos decir que su principal ventaja es la rapidez con la que resuelve problemas de optimización. Disponemos de algoritmos de resolución del TAP ya implementados y los resultados nos dan buenos valores en la fase de equilibrado. Su inconveniente es son pocas las facilidades para la programación que nos ofrece, ya que no podemos manejar las estructuras de datos necesarias ni siquiera utilizar bucles y condiciones de forma básica, teniendo que recurrir a conjuntos.

MATLAB nos ofrece muchas más ventajas con las que poder codificar nuestro algoritmo de forma satisfactoria. Nos proporciona todas las características de un lenguaje de programación, manejo de estructuras de datos complejas, manejo de ficheros, manejo de volúmenes elevados de información, numerosos paquetes (toolbox) como el de optimización, etc.

2.4. Recursos

Este proyecto fin de carrera va a ser desarrollado contando con los siguientes recursos humanos, hardware y software.

Recursos humanos

- Alumno: Carlos Blanco Bueno.
- Director: Ricardo García Ródenas.
- Coordinador: Macario Polo Usaola.

Recursos hardware

- Ordenador PC Dell Optiplex GX520 (Pentium 4 a 3.00 GHz).
- Ordenador Portatil Acer Aspire 5610 (Core Duo T2300 a 1.66).
- Ordenador Power Mac G5 (2 x PowerPC G5 2.7 GHz, 8 GB RAM).
- Servicio de Supercomputación de la UCLM.

Recursos software

- MATLAB 7.1 R14 sp3, para la codificación del algoritmo DSD y otros programas complementarios de preprocesado de datos.
- GAMS para la generación de resultados numéricos usados en tests.
- Datos de redes de Bar-Gera y su recodificación, disponibles en Internet

  `(http://www.bgu.ac.il/~bargera/tntp)`

- LaTeX para la generación de toda la documentación del proyecto, haciendo uso de las herramientas WinEdt, MiKTeX, GSView y Acrobat Reader.
- Microsoft Visio 2003, Gimp,... para la generación de diagramas para la documentación técnica.
- OpenOffice 2.0 para elaborar la presentación.
- Documentación disponible en Internet correspondiente a las distintas herramientas y lenguajes empleados.

Capítulo 3

Antecedentes

Desde 1924 cuando Knight describió el comportamiento en redes congestionadas hasta la actualidad, la modelización del transporte ha sido y es un problema de gran interés en el que se han realizado grandes avances y aportaciones, de modo que el aportar algo nuevo o una mejora es bastante difícil.

En este capítulo analizaremos el Estado-del-Arte en problemas de asignación de tráfico, estudiando también los algoritmos utilizados para resolverlo. Nos centraremos en la modelización del problema estático de asignación de tráfico asimétrico, objeto de este proyecto.

3.1. Problema de asignación de tráfico (TAP)

El problema de asignación de tráfico parte de una red de transporte representada por un grafo y una matriz de viajes origen-destino con las demandas. En el grafo tenemos un conjunto de nodos y arcos que representan la configuración de la red. Existen unos nodos especiales, llamados centroides, los cuales generan o reciben demanda, así como unos arcos ficticios llamados conectores que se encargan de unir estos centroides con la red.

Para realizar la asignación de la matriz O-D a la red, utiliza un conjunto de reglas o principios que buscan unos objetivos: obtener buenas medidas agregadas de la red, estimar costes de viajes, obtener flujos razonables en los arcos, identificar arcos congestionados, estimar las rutas para cada par origen-destino, etc.

En todos estos métodos se supone que el viajero actúa de forma racional escogiendo siempre la ruta que percibe como de menor coste. En la elección de la ruta por parte del usuario influyen numerosos factores como pueden ser tiempo de viaje, distancia, obras, paisaje,... No podemos ni sería conveniente tenerlos todos en cuenta y establecer una expresión generalizada, así que se suelen utilizar aproximaciones.

Si todos los conductores siguen esta hipótesis de seleccionar la mejor ruta, la de coste mínimo, el sistema llegaría a una situación de equilibrio en la que ningún conductor podría mejorar el tiempo de viaje empleado cambiando de ruta.

En la mayoría de programas de asignación de tráfico se consideran dos factores: el tiempo y la distancia de viaje, ya que lo más común es que el usuario quiera conocer

el camino más rápido o el más económico (el de menor distancia). Se suele permitir que el usuario asigne pesos a estos dos factores y así obtener una ruta que minimice ese coste generalizado y satisfaga los intereses del usuario.

En problemas como el de tráfico urbano de vehículos privados sólo consideramos como factor a tener en cuenta el tiempo. Aun así, podemos ver como conductores con el mismo viaje escogen rutas distintas, y esto se debe a efectos de la congestión o a las percepciones que tiene cada conductor del coste del camino.

3.1.1. Modelos de asignación de tráfico

Los modelos de asignación de tráfico, como hemos comentado anteriormente, realizan la asignación de una demanda representada por una matriz origen-destino a una red de tráfico. Existen multitud de modelos, de forma que podemos agruparlos según la siguiente clasificación:

Modelos estáticos

Son los modelos más populares y conocidos y suponen que durante un determinado periodo de tiempo se mantienen las mismas condiciones de demanda, existiendo situaciones estacionarias de equilibrio. Suelen centrarse en un período de estudio, como pueden ser las horas punta, y trabajar con valores medios de demandas, tiempos, flujos, etc.

Según como percibe el usuario los costes, podemos clasificar estos modelos en deterministas y estocásticos. En los **deterministas** se supone que todos los conductores tienen información completa y perciben de igual forma los costes de la red y los **estocásticos** en los que cada usuario selecciona su ruta dependiendo de su percepción, de la forma en la que percibe los costes.

A su vez, dentro de cada uno de estos modelos podemos considerar los **costes simétricos** o **asimétricos**. Diremos que los costes son simétricos si la matriz Jacobiana de la función de coste en los arcos, $C(v)$, es una matriz simétrica y asimétrica en caso contrario. El caso más importante de costes simétricos ese el llamado **costes separables** que consiste en que el coste de un arco sólo depende del volumen de tráfico en ese arco. Matemáticamente los problemas con costes simétricos se resuelven mediante un modelo

de optimización y los de costes asimétricos mediante desigualdades variacionales.

También podemos distinguir entre modelos con o sin **congestión**, dependiendo de si tenemos en cuenta sus efectos, es decir, si consideramos que el coste de un arco depende o no del flujo en los arcos de la red.

En los modelos deterministas en los que no consideramos la congestión, también llamados de asignación todo o nada, primero se realiza un calculo de los caminos mínimos para cada par origen-destino y luego se asigna toda la demanda del par a dicho camino, obteniéndose los flujos de los caminos y posteriormente los de los arcos de la red. En modelos estocásticos se hace igual salvo en la forma en la que los usuarios eligen las rutas, ya que se hace mediante una matriz de probabilidad creada en función del coste de las rutas.

Si consideramos los efectos de la congestión, los modelos matemáticos más utilizados son los modelos de asignación de tráfico en equilibrio, en los que se busca que todos los caminos de un par tengan el mismo coste.

Aunque Knight en 1924 fue el primero en describir de forma intuitiva el comportamiento del tráfico en redes congestionadas, fue en 1952 cuando Wardrop lo formalizó estableciendo sus principios [Wardrop1952].

- **Primer principio de Wardrop**

  También llamado UE (User Equilibrium) establece que: "Los costes de viaje de todas las rutas usadas en el equilibrio son iguales y menores que en los que incurriría un único vehículo que utilizará una ruta distinta a la que utiliza en esos momentos". Quiere decir que en una situación de equilibrio el usuario no puede reducir su tiempo de viaje cambiando de ruta.

  Este principio es utilizado para modelizar el comportamiento de los usuarios y en él se asume que: todos los usuarios perciben el coste de la misma manera y conocen los costes de todas las rutas (tienen información perfecta).

  Los flujos asignados a la red que satisfacen este principio se les llama "flujos óptimos para el usuario", ya que cada usuario va por la ruta que percibe como mejor.

- **Segundo principio de Wardrop**

  También llamado SO (System Optimal) establece que: "Los usuarios eligen la ruta de modo que se minimice el tiempo total de transporte en la red".

Establece que el tiempo promedio de viaje es mínimo. Este principio es utilizado como criterio para diseñar la red de transporte. Los flujos asignados a la red que satisfacen este principio se les llama "flujos óptimos para el sistema".

- **Equilibrio estocástico**

  Cuando los usuarios de la red eligen su ruta, lo hacen según su percepción, seleccionando la que creen que tiene menor coste.

  La asignación estocástica, SUE (Stochastic User Equilibrium), sigue esta idea estableciéndose los costes mediante la suma de una parte fija y una aleatoria. Los usuarios eligen sus rutas dependiendo de la distribución de probabilidad de los costes aleatorios.

Modelos dinámicos

Los modelos dinámicos intentan resolver dos inconvenientes planteados a los modelos estáticos: que son incapaces de explicar evoluciones de los flujos de tráfico en períodos cortos de tiempo y que subestiman los tiempos totales de viaje.

Estos modelos estiman los flujos de los tramos de la red de manera variable con el tiempo, constituyendo una extensión del problema de asignación convencional o en equilibrio. Se considera que los usuarios van actualizando continuamente su información sobre el tráfico y cambiando su ruta por una que minimice su coste.

El enfoque dinámico es especialmente importante en sistemas avanzados de información sobre el viajero (ATIS) en los que un controlador central recomienda a los usuarios su ruta en tiempo real buscando satisfacer un equilibrio de usuario o la optimización del sistema, respondiendo a variaciones inesperadas en las condiciones de la red.

Para desarrollar estos modelos se usa principalmente simulación, programación matemática y control óptimo. Para modelizar el fenómeno de la propagación del tráfico se suelen utilizar las "funciones de salida de los arcos"que relacionan el flujo y el número de vehículos en los arcos.

Tabla 3.1: Notación empleada

$\mathcal{N}$: conjunto de nodos de la red de tráfico
$\mathcal{A}$: conjunto de arcos de la red de tráfico
$a \in \mathcal{A}$: un arco de la red
W: conjunto de pares origen-destino
$\omega \in W$: un par origen-destino
$\mathcal{P}_\omega$: conjunto de caminos que satisfacen el par origen-destino ω
$\mathcal{P}^*_\omega$: subconjunto de los caminos en equilibrio que satisfacen el par origen-destino ω
g_ω: demanda de tráfico para el par origen-destino ω
g: vector con las demandas de tráfico $(\ldots, g_\omega, \ldots)$
Δ: matriz de incidencia arco/camino donde $\delta_{ap} = 1$ si el arco a está en el camino p o 0 si no
Λ: matriz de incidencia par/camino donde $\lambda_{\omega p} = 1$ si el camino $p \in P_\omega$ o vale 0 en caso contrario.
h_p: flujo en el camino p
h: vector de flujos de caminos $(\ldots, h_p, \ldots)$
f_a: flujo en el arco a
f: vector de flujos de arcos $(\ldots, f_a, \ldots)$
$c_a(f)$: coste del arco a en función de su flujo
$c(f)$: vector de costes de arcos $(\ldots, c_a(f), \ldots)$
$C_p(h)$: coste del camino p en función de su flujo
$C(h)$: vector de costes de caminos

3.1.2. Formulación matemática

Nos vamos a centrar en los modelos estáticos de asignación de tráfico, concretamente en modelos deterministas con costes asimétricos, que se formulan mediante desigualdades variacionales.

Modelos de desigualdades variacionales

Con estos modelos podemos resolver problemas más generales que con la programación matemática convexa diferenciable. Los modelos de desigualdades variacionales también son conocidos como ecuación generalizada o problema de punto estacionario.

Formalmente se considera un conjunto cerrado y convexo $X \subset \Re^n$ y una función $F : X \mapsto \Re^n$ continua en X. El problema de desigualdades variacionales $VIP(F, X)$ consiste en encontrar un $x^* \in X$ cumpliendo que

$$F(x^*)^T(x - x^*) \geq 0, \quad \forall x \in X$$

Las condiciones de optimalidad de un problema en desigualdades variacionales $VIP(F, X)$ se pueden sintetizar en el siguiente teorema.

Teorema 1 *Sea X un conjunto no vacío, compacto y convexo de R^n y sea F una aplicación continua del conjunto X a R^n. Entonces existe solución del problema $VIP(F, X)$. Además, si F es estrictamente monótona en X, la solución es única.*

Nota: se dice que F es estrictamente monótona en X si

$$[F(x) - F(y)]^T(x - y) > 0, \quad \forall x \neq y; \forall x, y \in X$$

Formulación matemática del TAP

En este apartado vamos a abordar las formulaciones matemáticas del problema de asignación de tráfico, centrarnos en la resolución del caso asimétrico mediante desigual-

dades variacionales. Para sintetizar hemos recogido la notación que vamos a utilizar en la tabla 3.1.

Condición de equilibrio: Decimos que un par origen-destino $\omega \in W$ está en equilibrio cuando todos los caminos del par $p \in \omega$ cumplen que si el camino tiene flujo $h_p^* > 0$ su coste es el coste de equilibrio del par $C_p(h) = U_\omega$ y si el camino no tiene flujo $h_p^* = 0$ su coste es mayor o igual que el coste de equilibrio $C_p(h) \geq U_\omega$.

Para formular matemáticamente las condiciones de equilibrio necesitamos describir los requerimientos de factibilidad de los flujos. Considerando K_ω como el conjunto de caminos que satisfacen el par ω, estos serían los requerimientos:

1. La demanda de cada par origen-destino debe ser satisfecha por los caminos de ese par:

$$\sum_{k \in K_\omega} h_k = g_\omega, \quad \forall \omega \in W$$

2. Los flujos deben ser no negativos:

$$h_k \geq 0, \quad \forall k \in K$$

3. La relación entre los flujos en los arcos y los flujos en las rutas viene definida por:

$$\sum_{\omega \in W} \sum_{k \in K_\omega} \delta_{ak} h_k = v_a, \quad \forall a \in \mathcal{A}$$

donde $\delta_{ak} = 1$ si el camino k pasa por el arco a y 0 en caso contrario. Esta restricción indica que el flujo en un arco a es la suma del flujo de todos los caminos que emplean dicho arco.

El problema de asignación de tráfico fue formulado como un problema de desigualdades variacionales en el espacio de flujo en los arcos por Smith y Dafernos (ver [Dafermos1980]). De una forma general, el problema VIP$(c, \tilde{\Omega})$ sería el de encontrar $f^* \in \tilde{\Omega}$ tal que

$$c(f^*)^T(f - f^*) \geq 0, \quad \forall f \in \tilde{\Omega}$$

donde el espacio de flujos factibles viene dado por $\tilde{\Omega} = \{f \mid f = \Delta h,\ \Lambda h = g,\ h \geq 0\}$.

Este problema puede ser reformulado considerando la función de coste en términos de flujos de caminos variables h en vez de los flujos en arcos f. El VIP(C, Ω) consistiría en encontrar $h^* \in \Omega$ tal que

$$C(h^*)^T(h - h^*) \geq 0, \quad \forall h \in \Omega$$

donde $\Omega = \{h \mid \Lambda h = g,\ h \geq 0\}$.

Si suponemos una red formada por un conjunto de nodos $\mathcal{N}$, arcos $\mathcal{A}$, costes de arcos $c(f)$ y matriz origen-destino g, el problema de encontrar los flujos de equilibrio para esa demanda equivale a resolver el VIP$(c, \tilde{\Omega})$ o VIP(C, Ω) (ver [Dafermos1980]).

Hearn [Hearn1982] definió la función *gap* G asociada al VIP(C, Ω) como:

$$G(h) := \text{Maximizar}_{\tilde{h} \in \Omega} \ C(h)^T(h - \tilde{h})$$

La función del *gap* mide la mejora del VIP(C, Ω), de forma que la solución de equilibrio podría obtenerse como la minimización global del problema matemático no convexo y no diferenciable.

$$Minimizar_{h \in \Omega} \ G(h)$$

3.1.3. Un ejemplo numérico

En este apartado vamos a ilustrar el modelo de asignación de tráfico utilizando datos numéricos y diagramas para un ejemplo. Para ello utilizaremos la red de Nguyen Dupuis cuya topología aparece en la figura 3.1 y como matriz origen destino que se desea asignar a la red, la que aparece en la tabla 3.2.

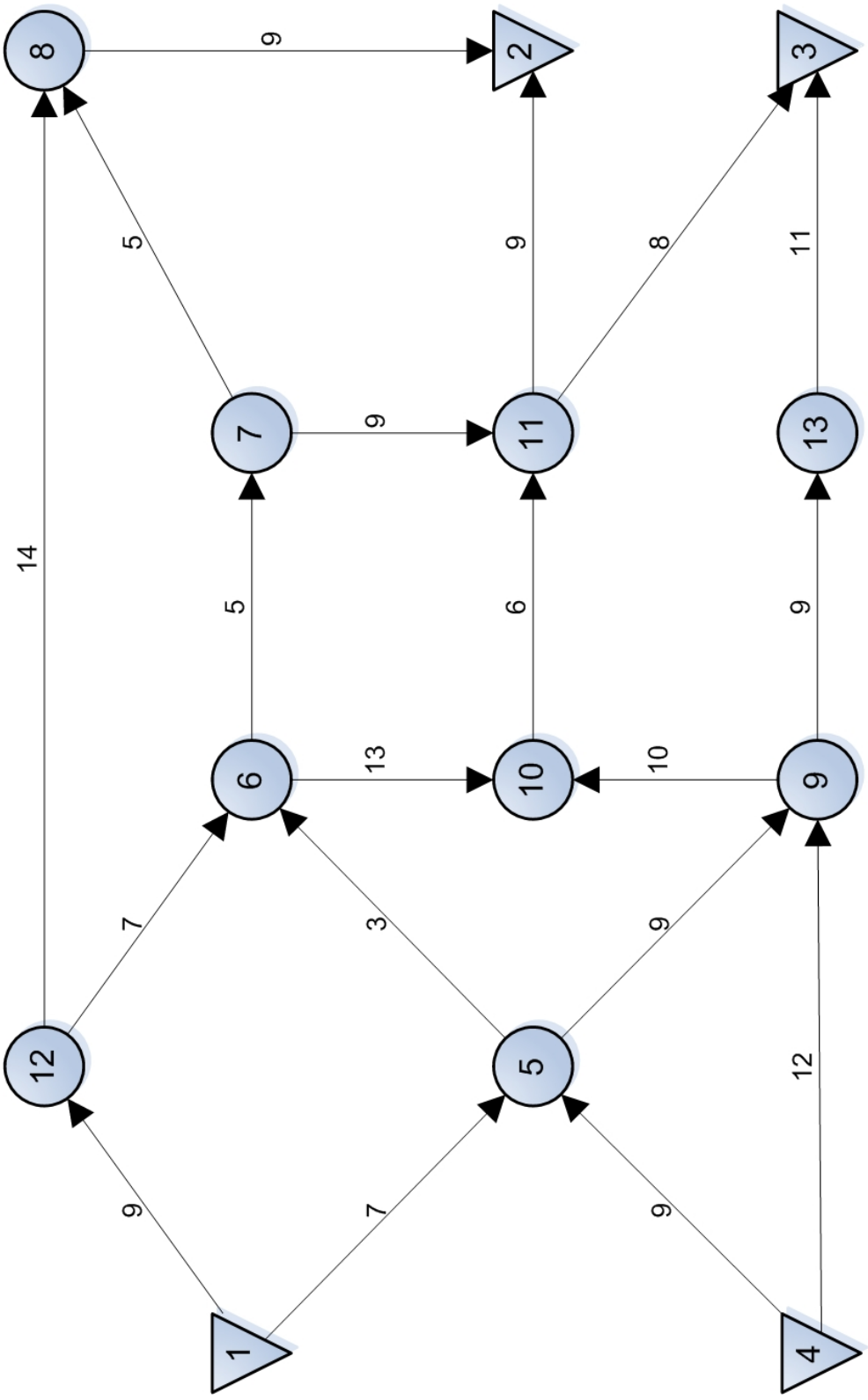

Figura 3.1: Red de Nguyen-Dupuis

Tabla 3.2: Matriz O-D para Nguyen-Dupuis

Par	Origen	Destino	Demanda
1	1	2	400
2	1	3	800
3	4	2	600
4	4	3	200

Las funciones de coste en los arcos tienen la forma $C_a(V_a) = t0_a + (A_a * V_a^{n_a})$, donde los parámetros se muestran en la tabla 3.3.Al asignar la demanda y resolver el problema sin considerar los efectos de la congestión obtendríamos los resultados que aparecen en la tabla 3.4, los cuales están representados gráficamente en la figura 3.2. Podemos ver como se ha generado un camino mínimo para cada demanda y se le ha asignado todo el flujo de ese par. El coste de los caminos mínimos es de 29, 32, 31 y 32 respectivamente, pero ese no es el coste real de los caminos, ya que después de asignarles un flujo debemos de recalcular los flujos de los arcos, los costes de los arcos y por último los costes de los caminos, de forma que el coste correcto es el que aparece en la tabla 3.4.

Tabla 3.3: Parámetros de funciones de coste de la red de Nguyen-Dupuis

Arco	$t0_a$	A_a	n_a	C_a
1	7	0.0125	1	7
2	9	0.01	1	9
3	9	0.01	1	9
4	12	0.005	1	12
5	3	0.0075	1	3
6	9	0.0075	1	9
7	5	0.0125	1	5
8	13	0.005	1	13
9	5	0.0125	1	5
10	9	0.0125	1	9
11	9	0.0125	1	9
12	10	0.005	1	10
13	9	0.005	1	9
14	6	0.0025	1	6
15	9	0.005	1	9
16	8	0.0025	1	8
17	7	0.005	1	7
18	14	0.01	1	14
19	11	0.01	1	11

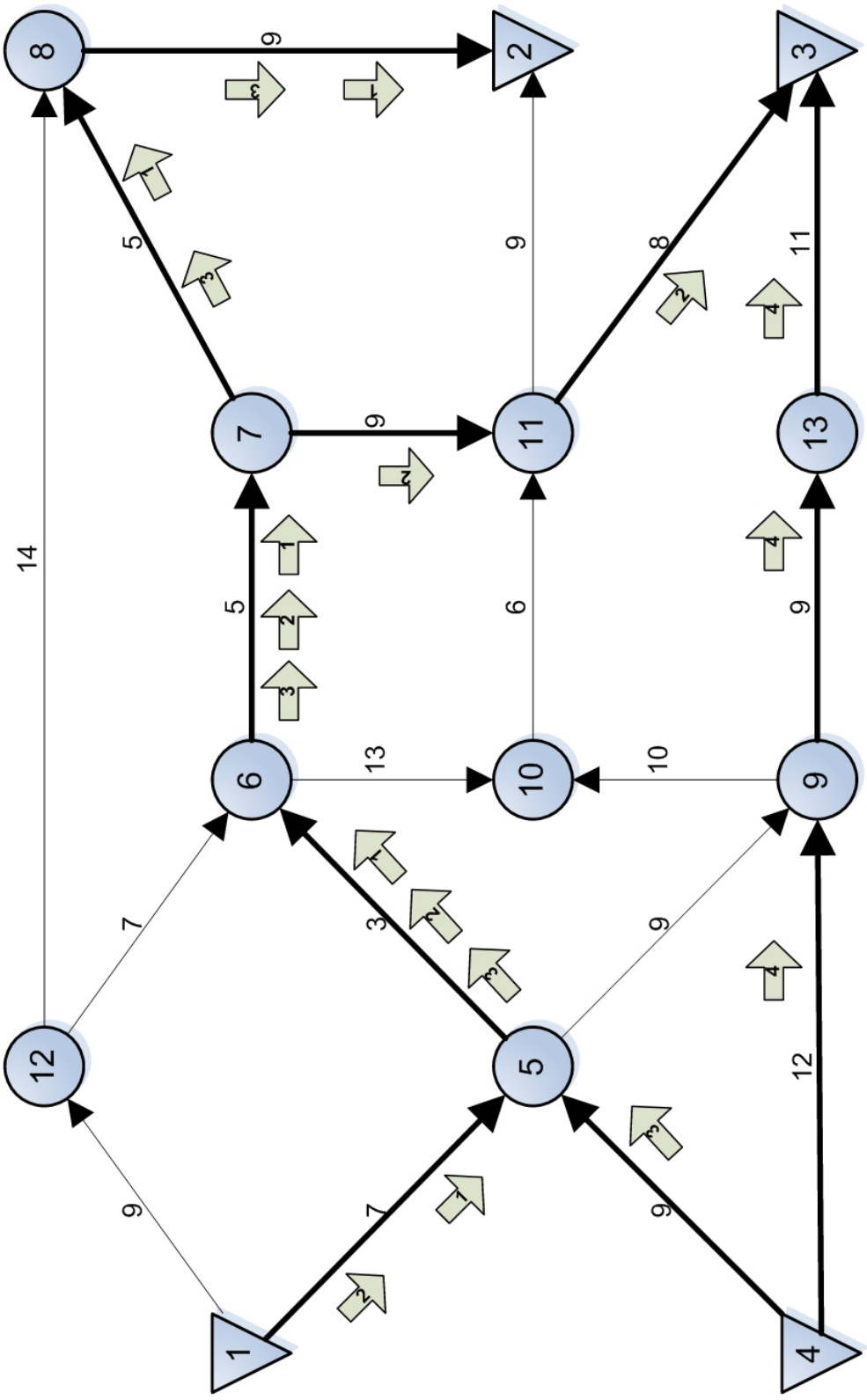

Figura 3.2: Solución del modelo de asignación sin congestión a la red de Nguyen-Dupuis

Tabla 3.4: Aplicación del modelo de asignación sin congestión a la red de Nguyen-Dupuis

Par	Origen	Destino	Demanda	Coste sin cong	Coste	Camino
1	1	2	400	29	105	$1 \mapsto 5 \mapsto 6 \mapsto 7 \mapsto 8 \mapsto 2$
2	1	3	800	32	101	$1 \mapsto 5 \mapsto 6 \mapsto 7 \mapsto 11 \mapsto 3$
3	4	2	600	31	98	$4 \mapsto 5 \mapsto 6 \mapsto 7 \mapsto 8 \mapsto 2$
4	4	3	200	32	36	$4 \mapsto 9 \mapsto 13 \mapsto 3$

Si consideramos los efectos de la congestión obtendríamos los resultados que aparecen en la tabla 3.5. Se observa que los usuarios del par 2 y 3 emplearán más de un camino, pero todos los caminos del mismo par poseen idéntico coste, y este es menor que cualquier coste de los caminos de la red.

Tabla 3.5: Aplicación de modelo de asignación en equilibrio (con congestión) a la red de Nguyen-Dupuis

Par	Origen	Destino	Flujo	Coste	Camino
1	1	2	400	47.53	$1 \mapsto 12 \mapsto 8 \mapsto 2$
2	1	3	253.8	55.57	$1 \mapsto 5 \mapsto 6 \mapsto 7 \mapsto 11 \mapsto 3$
			124.8	55.57	$1 \mapsto 12 \mapsto 6 \mapsto 10 \mapsto 11 \mapsto 3$
			361.5	55.57	$1 \mapsto 5 \mapsto 9 \mapsto 13 \mapsto 3$
			59.7	55.57	$1 \mapsto 5 \mapsto 6 \mapsto 10 \mapsto 11 \mapsto 3$
3	4	2	497.4	47.16	$4 \mapsto 9 \mapsto 10 \mapsto 11 \mapsto 2$
			102.5	47.16	$4 \mapsto 5 \mapsto 6 \mapsto 7 \mapsto 8 \mapsto 2$
4	4	3	200	43.91	$4 \mapsto 9 \mapsto 13 \mapsto 3$

3.2. Algoritmos para la resolución del TAP

En la monografía de Patriksson [Patriksson1994] podemos ver las estrategias que han sido aplicadas en la resolución del TAP, ya que se recogen en ella más de mil referencias bibliográficas relacionadas con este problema.

En la resolución de este problema se han desarrollado algoritmos altamente especializados, como los métodos de **descomposición simplicial**, que aprovechan todas las características del problema y permiten resolver redes de transporte de gran tamaño.

3.2.1. Descomposición simplicial

El algoritmo de *descomposición simplicial* (SD), tuvo su origen en los trabajos de [Holloway1974] y [Hohenbalken1977]. La forma clásica de la descomposición simplicial fue primeramente descrita para problemas de optimización no lineales con restricciones lineales y ha sido generalizada para abordar problemas [P(f, X)] de la forma *minimizar* $f(x)$, $x \in X$, donde X es un conjunto compacto, convexo y no vacío; y $f : X \mapsto \Re$ es continuamente diferenciable y pseudoconvexa en X.

En los algoritmos de descomposición simplicial se itera entre dos subproblemas: el problema maestro restringido (RMP) y el problema de generación de columnas (CG). Cada problema se puede interpretar como dos aproximaciones al problema original P(f, X) obtenidas respectivamente utilizando una aproximación (*interior*) de la región factible y otra de la función objetivo.

- **Problema maestro restringido (RMP)**.

  Estos métodos construyen y resuelven una aproximación al problema original, obtenida reemplazando la región factible por un conjunto convexo (e.g. poliedral) que es una *aproximación interior* de dicha región.

  Consiste en resolver el [RMP$(f, \hat{X})$] que denotaremos *minimizar* $f(x)$, $x \in \hat{X} \subset X$. La región factible del RMP $\hat{X}$ debe contener el segmento definido por la última columna generada $(\hat{y})$ y por la solución al último RMP $(\hat{x})$, esto es $[\hat{y}, \hat{x}] \subset \hat{X}$. Esta exigencia garantiza que los RMP mejoran monótonamente la aproximación a la solución del problema.

  La definición de la región factible del RMP $\hat{X}$ se ha definido tradicionalmente de dos maneras: i) mediante la envoltura convexa de un subconjunto de las columnas generadas o ii) mediante la envoltura afín de un conjunto de las columnas generadas, intersección con la región factible. Ambas definiciones garantizan la satisfacción de la propiedad anterior y de la exigencia $\hat{X} \subset X$ debido a que los esquemas de descomposición simplicial han sido aplicados a regiones factibles X convexas.

- **Problema de generación de columnas (CGP)**.

  Esta aproximación interior es mejorada (aumentada) mediante la generación de un vector (o columna) en el conjunto factible, a través de la resolución de otra aproximación al problema de optimización original pero en este caso es la función objetivo

la que es aproximada. Se resuelve el $[\mathrm{CG}(\hat{f}, X)]$ que se denotará *minimizar* $\hat{f}(x)$, $x \in X$.

Los subproblemas CG en los que no se obtienen necesariamente puntos extremos, no se almacena la columna generada sino su prolongación a la frontera (relativa) de X respecto a la actual aproximación a la solución.
Esto es, $y := \hat{x} + \hat{\ell}(\hat{y} - \hat{x})$, donde $\hat{\ell} := \max\{\, \ell \mid \hat{x} + \ell(\hat{y} - \hat{x}) \in X\}$,
siendo respectivamente $\hat{x}$ y $\hat{y}$ la última solución al RMP y al subproblema CG. La motivación de esta operación de prolongación radica en que el número de variables del RMP se mantiene constante pero produce una región factible más amplia.

3.2.2. Algoritmos de descomposición simplicial

Los métodos de descomposición simplicial pueden clasificarse según varios criterios:

- El espacio de flujos usado: si usamos los flujos de arcos tenemos una descomposición simplicial agregada (ASD) y si usamos flujos de caminos tenemos una descomposición simplicial desagregada (DSD).
- La estrategia de resolución usada en el RMP: como Newton, Jacobi o Proyección.
- La regla usada para eliminar / introducir columnas en el RMP.

A continuación vamos a describir como ha sido el desarrollo de los algoritmos simpliciales, destacando principalmente tres etapas, y terminando con una tabla resumen en el que aparecen las características de los principales algoritmos desarrollados.

Subproblemas generadores de columnas en el conjunto de puntos extremos

En esta primera etapa, los algoritmos son desarrollados para problemas con restricciones lineales con estructura especial, como las de red, lo que conduce a problemas CG lineales con una estructura especial, que permite su resolución con especializaciones eficientes del método simplex. Los RMP en las primeras iteraciones tienen muy pocas variables y una estructura muy simple en las restricciones (combinación convexa) lo que permite la aplicación eficiente de algoritmos de convergencia superlineal.

La experiencia con el método de descomposición simplicial (SD) ha mostrado que hace rápidos progresos inicialmente, y rápidamente alcanza una solución casi óptima, especialmente cuando se aplica un algoritmo de convergencia cuadrática para resolver los RMP, pero esta eficiencia se reduce cerca de la solución óptima. El algoritmo SD es menos eficiente para problemas en los que la dimensión de la *cara óptima del problema* es elevada. Esto es debido a que, en estas circunstancias, el número de puntos extremos, necesarios para expresar una solución óptima como combinación convexa de ellos, es elevado y, por tanto, también lo es el número de iteraciones necesarias para identificar gran parte de ellos. Esto conduce a que se deba resolver gran cantidad de RMP y que los últimos posean gran cantidad de variables.

Una primera solución a este problema fue propuesta por Lawphongpanich y Hearn [Lawphongpanich and Hearn1984], [Hearn et al.1985] y [Hearn et al.1987]. Proponen un ASD, los métodos de Jacobi y Proyección para la etapa de RMP y eliminar las columnas con peso 0. Estos autores introducen un parámetro r para limitar el número de columnas almacenadas en el RMP. Cuando el número de columnas retenidas en el RMP alcanza esta cantidad, la última columna generada se intercambia por la columna de menor peso en la combinación convexa de la última solución del RMP. Esta solución no es enteramente satisfactoria ya que requiere elevados valores del parámetro r para mantener una convergencia superlineal presente en el esquema original de SD.

Pang y Yu [Pang and Yu1984] proponen un ASD en el que el RMP es aproximado usando sólo un paso del método de aproximación lineal. Marcotte y Guélat [Marcotte and Guélat1988] usaron un ASD y aproximaron el RMP aplicando un método de Newton truncado.

Los problemas de flujos en redes multiproducto suelen tener dos formulaciones alternativas, una en el espacio de flujos en los arcos y otra en el espacio de flujos en los caminos. El espacio de flujo en los caminos tiene una estructura de producto cartesiano, donde cada conjunto es el espacio de flujo para cada producto.

Montero [Montero1992] y Montero y Barceló [Montero and Barceló1996] estudiaron el rendimiento de los algoritmos ASD en redes reales en relación a la técnica de cálculo de caminos mínimos usada, el criterio de parada para el RMP y la regla usada para eliminar columnas.

Larsson y Patriksson [Larsson and Patriksson1992] proponen un algoritmo DSD para resolver el modelo de asignación de tráfico diagonal formulado como un problema de

optimización. Consideran una SD en el espacio de flujos en los caminos, lo que denominan *descomposición simplicial desagregada* (DSD) donde, en lugar de tomar combinación convexas en el espacio total, consideran el producto cartesiano de envoltura convexas para definir los RMP. Este algoritmo resuelve un mucho menor número de RMPs pero con un mayor número de variables, pero el resultado global presenta mejores resultados computacionales que los anteriores.

[Marín1995] especializa el RSD para problemas con restricciones laterales. En este algoritmo el problema maestro se obtiene como la intersección de un símplice con el conjunto definido por las restricciones laterales. El subproblema lineal es modificado para que la función objetivo recoja una estimación de los multiplicadores de Lagrange de las restricciones laterales.

Subproblemas generadores de columnas en la frontera relativa

La explicación de la conducta de los algoritmos SD y RSD se encuentra en que emplean una aproximación de primer orden de f para definir los problemas CG. El problema CG en estos dos algoritmos coincide con el del algoritmo de Frank–Wolfe.

Es sabido que la calidad de estas direcciones de búsqueda se deteriora rápidamente. La razón es que la sucesión de derivadas direccionales $\{\nabla f(\hat{x}^t)^T d^t\}$ en las direcciones de búsqueda $d^t := \hat{y}^t - \hat{x}^t$ tienden a cero pero la sucesión $\{d^t\}$ no converge a 0; lo que implica que las direcciones de búsqueda tienden rápidamente a ser ortogonales al gradiente de f y por tanto la calidad de las columnas generadas (entendida como la mejora inducida en el RMP) será rápidamente reducida.

Una conclusión natural es emplear en el problema CG una mejor aproximación de f, de ello se podría esperar una mejor calidad en las columnas y por tanto una mejor aproximación interior en el RMP. [Larsson et al.1997], basándose en esta observación, extendieron el algoritmo RSD a métodos de generación de columnas no lineales. Esta clase se denomina *descomposición simplicial no lineal* (*NSD*). Estos métodos poseen menor sensibilidad a la dimensión de la cara óptima debido a que emplean un menor número de columnas para describir la solución óptima. Esto hace que los métodos requieran un menor número de iteraciones y permite elegir un valor menor del parámetro r. Como los problemas CG son no lineales, las columnas ya no son necesariamente puntos extremos. Estos autores prolongan la columna obtenida a la frontera relativa.

Subproblemas generadores de puntos en la frontera relativa

La convergencia del algoritmo NSD puede ser establecida mediante resolución truncada tanto del problema CG como del RMP. Este resultado tiene una gran importancia práctica pero mucho más desde un punto de vista teórico. Esto permitió en [García2001], [García et al.2003] considerar el propio problema original como la mejor aproximación al mismo.

La diferencia sustancial con los anteriores métodos de descomposición simplicial radica en la definición del problema CG. En los esquemas anteriores el problema CG se interpreta como una aproximación al problema original. El nuevo esquema, denominado CGA, construye las columnas resolviendo aproximadamente el problema original a través de la realización de un número de iteraciones de algún algoritmo eficiente. En este contexto, el énfasis se sitúa en la elección de los algoritmos empleados en el CGP y no en la forma de aproximar el problema original.

Los algoritmos CGA pueden ser vistos como algoritmos modulares de programación matemática no lineal, donde se dispone de un algoritmo para resolver eficientemente el RMP y de otro para resolver el problema original. Este último algoritmo es el procedimiento para obtener las columnas en el CGP. Esta clase puede ser interpretada como un principio para acelerar la convergencia de un algoritmo convergente de puntos factible (algoritmo empleado en el CGP) mediante un esquema de descomposición simplicial.

En el trabajo [García2001] se hace un estudio computacional de la clase de algoritmos CGA empleando dos problemas de flujos en redes no lineales. El primero es el problema uniproducto de flujos en redes no lineales con restricciones de capacidad y el segundo es un problema de flujos multiproducto para la asignación en redes multimodales de transporte. La contribución fundamental de este estudio numérico, es que se ha mostrado que la clase CGA mejora sustancialmente, y en algunos casos espectacularmente, los algoritmos de descomposición simplicial RSD, SD y NSD.

La tabla 3.6 describe las características de los principales algoritmos simpliciales desarrollados en la literatura. En la primera columna es el acrónimo del método, en la segunda se indica que tipo de región factible posee el problema de interés, en la tercera columna se indica que operación (envoltura convexa o afín) se efectúa sobre las columnas actualmente retenidas para definir la región factible del RMP, en la cuarta columna se indica si el método aplica o no la operación de prolongación a la frontera (relativa) de X

descrita anteriormente y en la última columna la función objetivo del subproblema CG.

Tabla 3.6: Características de los algoritmos simpliciales

	X	$\hat{X}$	Prolongación	$\hat{f}(x)$
SD	poliedro	envoltura convexa	no	$\hat{f}(x) = \nabla f(\hat{x})^T x$
RSD	poliedro	envoltura convexa y restricción nº columnas por r	no	$\hat{f}(x) = \nabla f(\hat{x})^T x$
DSD	poliedro con estructura de producto cartesiano	producto cartesiano de envoltura convexa	no	$\hat{f}(x) = \nabla f(\hat{x})^T x$
NSD	convexo	envoltura convexa o afín	si	$\hat{f}(x) = \nabla f(\hat{x})^T x + \varphi(x, \hat{x})$ donde φ es continua, respecto a la variable x es convexa y continuamente diferenciable y cumple que $\varphi(x, x) = 0$ y $\nabla_x \varphi(\hat{x}, \hat{x}) = 0$.
CGA	convexo	envoltura convexa o afín	si	$\hat{f}(x) = f(x)$

Capítulo 4

Un algoritmo de Descomposición Simplicial Desagregada con Identificación de Pares

Los algoritmos de descomposición simplicial desagregada, que hemos visto en el capítulo anterior, constituyen el Estado-del-Arte para el problema de asignación de tráfico, pero no están desarrollados de la forma más eficiente ya que el coste computacional aumenta innecesariamente según avanzamos en iteraciones del algoritmo. Es por esto que en el desarrollo de nuestro algoritmo de descomposición simplicial, y basándonos en la hipótesis ya comentada de que prácticamente la totalidad de las demandas son satisfechas por un sólo camino, proponemos añadir a la estrategia de la descomposición simplicial una estrategia de identificación de pares que nos permita reducir la dimensionalidad del problema maestro restringido. En este capítulo pasaremos a describir y analizar el algoritmo propuesto.

4.1. Descripción del algoritmo

La formulación del VIP(C, Ω) requiere la enumeración de todos los caminos, y esto es una tarea inabordable para redes de tamaño medio. El esquema de descomposición simplicial desagregada evita este inconveniente generando las rutas según las va necesitando. Por esta razón el DSD puede ser visto como una estrategia que permite aplicar los métodos de resolución de desigualdades variacionales a la formulación del VIP(C, Ω).

Los algoritmos DSD se basan en la realización de varias iteraciones en las que se resuelven dos subproblemas de desigualdades variacionales: el problema de generación de columnas (CGP) y el problema maestro restringido (RMP). En el CGP se generan nuevos caminos a partir de la situación del flujo actual, y en el RMP se obtiene la situación de equilibrio sobre el conjunto de caminos previamente generados, obteniéndose un nuevo patrón de flujo donde reiniciar el proceso.

4.1.1. Problema de generación de columnas

En la fase **CGP** generamos nuevos puntos extremos del poliedro que delimita la región factible, cuya envoltura convexa determinará una aproximación interna en la región factible original. Denotando por CGP(ℓ) el problema de generación de caminos en la iteración ℓ. Sea $\bar{h}^{\ell-1}$ el vector de flujo actual el CGP(ℓ) se puede definir como encontrar $\bar{h}^{\ell} \in \Omega$ tal que resuelva exactamente el VIP

$$C(h^{\ell-1})^T(h - \bar{h}^{\ell}) \geq 0, \quad \forall h \in \Omega$$

Este problema es resuelto usando un algoritmo de caminos mínimos, generándose un camino óptimo para cada par origen-destino ω, denotado por p_{ω}^{ℓ}.

En función del conjunto actual de caminos para el par origen-destino ω, el cuál es denotado por $p_{\omega}^{\ell-1}$ para cada par $\omega \in W$, el DSD almacena para la iteración ℓ el conjunto de caminos

$$p_{\omega}^{\ell} = p_{\omega}^{\ell-1} \cup \{p_{\omega}^{\ell}\}, \quad \forall \omega \in W$$

Tras realizar un estudio de algoritmos de caminos mínimos existentes, nos centramos por su rapidez en el algoritmo L2queue de G.Gallo y S.Pallottino [?], el cuál constituye el Estado-del-Arte y se puede aplicar satisfactoriamente a problemas de asignación de tráfico. Se basa en calcular los caminos mínimos de un origen a un subconjunto de destinos. En el siguiente capítulo veremos en más detalle cómo hemos adaptado y modificado el L2queue.

4.1.2. Problema maestro restringido

La fase **RMP** resuelve el problema original sobre los caminos generados previamente, obteniendo una solución aproximada. El RMP es un problema de desigualdades variacionales con restricciones simples, cuya solución define el siguiente punto de la aproximación en el CGP. Con el RMP(ℓ) obtenemos unos flujos en equilibrio para el subconjunto de caminos de la red.

Para definir el RMP, introducimos la siguiente notación, mediante la que definimos el espacio de flujos de caminos para el par $\omega \in W$ como:

$$\Omega_{\omega} = \left\{ h \in \Re^{|\mathcal{P}_{\omega}|} \mid \sum_{p \in \mathcal{P}_{\omega}}, h_p = g_{\omega}, \ h_p \geq 0 \ \ \forall p \in \mathcal{P}_{\omega} \right\}$$

donde $\mathcal{P}_\omega$ es el conjunto de todos los caminos de la red que satisfacen la demanda del par $\omega \in W$. Denotamos los elementos de Ω_ω como h_ω.

El espacio de flujos Ω puede ser expresado como un producto cartesiano de conjuntos Ω_ω, tal y como sigue:

$$\Omega = \prod_{\omega \in W} \Omega_\omega$$

Dado un vector de flujos $h \in \Omega$, este es expresado como $h = (h_{\omega_1}, \ldots, h_{\omega_q})$ donde h_ω is un elemento de Ω_ω y q es el cardinal del conjunto W, por ejemplo $|W|$.

La región factible restringida en la iteración ℓ es denotada por Ω^ℓ y está definida como el siguiente producto cartesiano:

$$\Omega^\ell = \prod_{\omega \in W} \Omega^\ell_\omega$$

donde

$$\Omega^\ell_\omega = \left\{ h_\omega \in \Omega_\omega |\ \ h_p = 0 \ \ \forall p \in \mathcal{P}_\omega - \mathcal{P}^\ell_\omega \right\}$$

De esta forma el RMP en la iteración ℓ puede definirse como encontrar $h^\ell \in \Omega^\ell$ tal que

$$C(h^\ell)^T(h - h^\ell) \geq -\epsilon^\ell, \quad \forall h \in \Omega^\ell$$

donde ϵ^ℓ es un número positivo proporcionado. El punto h^ℓ es conocido como una solución $\epsilon^\ell-$óptima para el RMP(ℓ) y define el siguiente CGP.

El esquema anterior describe la adaptación del algoritmo DSD al VIP(C, Ω). En número de variables del RMP(ℓ) coincide con el número de caminos considerados en la iteración ℓ y el número de restricciones coincide con el número de pares origen destino. En el peor de los casos, el número de variables del RMP(ℓ) podría ser $\ell\,|W|$, lo que indica que cuando el algoritmo DSD converge puede resolver RMPs de gran escala.

4.1.3. Identificación de pares

Buscando limitar el tamaño de estos problemas añadimos al DSD una estrategia de **identificación de pares** para restringir el conjunto de pares origen destino utilizados en el RMP(ℓ).

El esquema propuesto considera un subconjunto de pares origen destino W^ℓ en cada iteración, el cuál está formado por los pares que tienen dos o más caminos en equilibrio. Es decir,

$$W^\ell := \left\{\omega \in W \,/\, \left|\mathcal{P}_\omega^\ell\right| > 1\right\}$$

Para los pares $\omega \in W - W^\ell$ toda la demanda g_ω es asignada al único camino contenido en $\mathcal{P}_\omega^\ell$. En nuestro DSD, el RMP(ℓ) tiene como objetivo calcular el flujo de los caminos para los pares identificados W^ℓ mientras que mantiene fijado el flujo de los caminos que satisfacen los pares $W - W^\ell$. El problema de desigualdades variacionales se resuelve de forma aproximada encontrando un $h^\ell \in \Omega^\ell$ que satisfaga:

$$RMP(\ell): \qquad \sum_{\omega \in W^\ell} C_\omega(h^\ell)^T(h_\omega - h_\omega^\ell) \geq -\varepsilon^\ell, \quad \forall h \in \Omega^\ell$$

donde el único camino para cada par $\omega \in W - W^\ell$ tiene un flujo fijado de g_ω.

$C_\omega(h^\ell)$ denota el vector formado por todas las componentes (caminos) de $C(h^\ell)$ asociados al par ω.

Nuestro DSD necesita de un **criterio de actualización** para el conjunto W^ℓ basado en una regla para añadir nuevos pares al conjunto y otra para la eliminar pares existentes. Estas reglas están basadas en la función gap asociada al par $\omega \in W$, definida por:

$$G_\omega(h) := Maximizar_{\bar{h}_\omega \in \Omega_\omega} C_\omega(h)^T(h_\omega - \bar{h}_\omega)$$

Observamos que $G(h) := \sum_{\omega \in W} G_\omega(h)$, $G_\omega(h) \geq 0$ y que $G(h^*) = 0$ si y solo si

$G_\omega(h^*) = 0, \forall \omega \in W$.

- Regla para la introducción de pares en la iteración ℓ.

$$W^\ell := W^{\ell-1} \cup \left\{ \omega \in W - W^{\ell-1} \,/\, G_\omega(h^{\ell-1}) > \delta^\ell > 0 \right\}$$

  G_ω puede calcularse en cada iteración a partir de la solución del CGP(ℓ).

- Regla para la eliminación de pares en la iteración ℓ.

  Una vez hemos resuelto el RMP(ℓ), eliminamos de W^ℓ los pares que tienen asociados sólo un camino con flujo positivo. Esta estrategia por si sola no garantiza la convergencia del algoritmo porque, tal y como ocurre en el ASD, pueden aparecer un determinado tipo de ciclos que hacen que el algoritmo pierda su convergencia. Para solventar este problema los pares de W^ℓ sólo son eliminados en aquella iteración que haya producido un descenso suficiente de la función gap, o en otras palabras, si satisface que $G_\omega(h^\ell) \leq \eta^\ell$.

4.1.4. Esquema del algoritmo

En la tabla 5.2 podemos ver como quedaría de forma esquemática el algoritmo DSD con identificación de pares propuesto.

4.2. Propiedades matemáticas

El siguiente resultado prueba la convergencia de el algoritmo bajo la hipótesis de la convergencia del algoritmo usado para resolver los RMPs.

Teorema 2 (Teorema de convergencia) *Según el* ***teorema de convergencia****, si suponemos que el algoritmo genera una sucesión infinita* $\{h^\ell\}$*, entonces cualquier sucesión convergente de* $\{h^\ell\}$ *converge en una solución del VIP*(C, Ω).

Tabla 4.1: Algoritmo DSD con identificación de pares.

0. (*Inicialización*): Escoger un parámetro de tolerancia $\bar{\epsilon}$, tres sucesiones de números positivos $\{\epsilon^\ell\}$, $\{\gamma^\ell\}$ y $\{\delta^\ell\}$ que converjan a 0, $\epsilon^\ell < \gamma^\ell$, $\delta^{\ell+1} < \gamma^\ell$ e inicializar $\ell := 1$. Asignar $h^0 \in \Omega$, tomando $\mathcal{P}_\omega^0 := \left\{p \in \mathcal{P}_\omega \,/\, h_p^0 > 0\right\}$ y $W^0 = \{\omega \in W / \, |\mathcal{P}_\omega^0| > 1\}$.
1. (*Problema de generación de columnas*): Resolver el CGP(ℓ). Almacenar la solución en $\bar{h}^\ell$ y calcular $G_\omega(h^{\ell-1})$ para todos los $\omega \in W$.
2. (*Criterio de parada*): Si $h^{\ell-1}$ resuelve el CGP(ℓ) entonces parar ($h^{\ell-1}$ resuelve el VIP(C, Ω)). Si no, continuar.
3. (*Fase de identificación de pares*): Para todos los pares $\omega \notin W^\ell$ que satisfacen la condición $G_\omega(h^{\ell-1}) > \delta^\ell$, actualizar el conjunto de caminos

$$\mathcal{P}_\omega^\ell := \mathcal{P}_\omega^{\ell-1} \cup \{p \in \mathcal{P}_\omega \,/\, \bar{h}_p > 0\}$$

   y activamos el par

$$W^\ell := W^{\ell-1} \cup \{\omega\}$$

4. (*Problema maestro restringido*): Encontrar una ϵ^ℓ-solución h^ℓ del RMP(W^ℓ).
5. (*Fase de eliminación de pares*): Para los pares $\omega \in W^\ell$ que satisfacen la condición $G_\omega(h^\ell) < \gamma^\ell$ actualizar el conjunto de caminos

$$\mathcal{P}_\omega^{\ell+1} := \mathcal{P}_\omega^\ell - \{p \in \mathcal{P}_\omega^\ell \,/\, h_p^\ell = 0\}$$

   y si además el par ω satisface que $|\mathcal{P}_\omega^\ell| = 1$, tomar

$$W^{\ell+1} := W^\ell - \{\omega\}$$

6. (*Actualizar*): Asignar $\ell := \ell + 1$ y volver al paso 1.

A continuación podemos ver la demostración. Partiendo de $h^\ell \in \Omega$ para todo ℓ y siendo Ω un conjunto compacto, tiene que existir una subsucesión convergente $\{h^\ell\}_{\ell \in L}$. Denotamos por h^* el punto límite de la sucesión $\{h^\ell\}_{\ell \in L}$.

La demostración de la convergencia esta basada en la función del gap, la cual satisface $G(h^*) = 0$ si y solo si h^* es una solución del VIP(C, Ω).

Consideramos $\omega' \in W$ arbitrario. Deberíamos probar que $G_{\omega'}(h^*) = 0$, y para ello debemos considerar tres casos:

- Caso A.

  Suponemos que el par ω' pertenece a un número finito de conjuntos $W^\ell, \ell \in L$. En este caso existe una iteración $\tilde{\ell}$ en la que el par ω' no está en el conjunto $W^{\tilde{\ell}}$ ni en sus siguientes y si tomamos esta iteración como punto de inicio, nunca llegaremos a satisfacer el criterio del paso 3 (inserción de pares) y entonces:

  $$G_{\omega'}(h^\ell) < \delta^{\ell+1}, \quad \forall \ell \geq \tilde{\ell} \text{ y } \ell \in L$$

  Tomando límites por ambos lados obtenemos:

  $$\lim_{\ell \in L, \ell \geq \tilde{\ell}} G_{\omega'}(h^\ell) = G_{\omega'}(h^*) = 0$$

- Caso B.

  Suponemos que el par $\omega' \in W$ aparece un número infinito de veces en los conjuntos W^ℓ, con $\ell \in L$ y que la cantidad de veces que el par ω' satisface la condición de eliminación de pares es finita. En este caso existirá un número entero positivo tal que a partir de él $\Omega^\ell_{\omega'} \subset \Omega^{\ell+1}_{\omega'}$. Esto es debido a que a partir de dicha iteración solamente se pueden introducir caminos. Puesto que $\Omega_{\omega'}$ tiene un número finito de caminos, la sucesión $\Omega^\ell_{\omega'}$ tiene que converger a algún conjunto $\tilde{\Omega}$. Por tanto tiene que existir un entero τ tal que:

  $$\Omega^\ell_{\omega'} = \tilde{\Omega}_{\omega'}, \quad \forall \ell > \tau$$

  De la definición del RMP(ℓ),

  $$\sum_{\omega \in W^\ell} C_\omega(h^\ell)^T (h_\omega - h^\ell_\omega) \geq -\varepsilon^\ell, \quad \forall h \in \Omega^\ell,$$

Si evaluamos la expresión anterior en vectores de la forma $(h^{\ell}_{\omega_1}, \cdots, h_{\omega'}, \cdots h^{\ell}_{\omega_q}) \in \Omega^{\ell}$, obtenemos:

$$C_{\omega'}(h^{\ell})^T(h_{\omega'} - h^{\ell}_{\omega'}) \geq -\varepsilon^{\ell}, \quad \forall h_{\omega'} \in \Omega^{\ell}_{\omega'}$$

o equivalentemente

$$max_{h'_{\omega} \in \Omega^{\ell}_{\omega'}} C_{\omega'}(h^{\ell}_{\omega'})^T(h^{\ell}_{\omega'} - h'_{\omega}) \leq \varepsilon^{\ell}$$

Para todo $\ell > \tau$ no se añaden nuevas columnas a $\tilde{\Omega}_{\omega'}$, y esto es debido a que el CGP(ℓ) para el par ω' genera un camino ya considerado en $\tilde{\Omega}_{\omega'}$.

$$G_{\omega'}(h^{\ell}) = C_{\omega'}(h^{\ell})^T(h^{\ell}_{\omega'} - \bar{h}^{\ell+1}_{\omega'}) = max_{h \in \Omega_{\omega'}} C_{\omega'}(h^{\ell})^T(h^{\ell}_{\omega'} - h_{\omega'}) =$$

$$= max_{h_{\omega'} \in \tilde{\Omega}_{\omega'}} C_{\omega'}(h^{\ell})^T(h^{\ell}_{\omega'} - h_{\omega'}) \leq \varepsilon^{\ell}$$

Como ε^{ℓ} es una sucesión que converge a cero y $G_{\omega'}(\cdot)$ es una función no negativa continua en $\Omega_{\omega'}$,

$$\lim_{\ell > \tau, \ell \in L} G_{\omega'}(h^{\ell}) = G_{\omega'}(h^*) = 0.$$

- Caso C.

  Suponemos que ω' aparece un número infinito de veces en los conjuntos W^{ℓ}, $\ell \in L$ y que satisface el criterio de eliminación de pares un número infinito de veces. Por lo tanto existe un subconjunto infinito de índices $\tilde{L} \subset L$, que satisface el criterio de eliminación de pares

$$G_{\omega'}(h^{\ell}) < \gamma^{\ell+1}, \quad \forall \ell \in \tilde{L}$$

Debido a que γ^{ℓ} es una sucesión que converge a cero y $G_{\omega'}(\cdot)$ es una función no negativa continua en $\Omega_{\omega'}$,

$$\lim_{\ell \in \tilde{L}} G_{\omega'}(h^{\ell}) = G_{\omega'}(h^*) = 0$$

Usando la relación entre la función del gap y la función del gap asociada a los pares, llegamos a

$$G(h^*) = \sum_{\omega \in W} G_{\omega}(h^*) = 0$$

Por lo tanto, h^* es solución del VIP(C, Ω). Como la sucesión L es arbitraria, el teorema queda demostrado.

Condición de complementariedad estricta: Con el objetivo de completar el resultado anterior pasamos a introducir el concepto de complementariedad para el VIP(C, Ω). La condición de complementariedad estricta se satisface para una solución en equilibrio h^* del VIP(C, Ω) si y sólo si

$$C_p(h^*) - U_\omega > 0 \text{ cuando } h_p^* = 0, \quad \forall p \in \mathcal{P}_\omega, \forall \omega \in W$$

La velocidad de convergencia local del algoritmo DSD depende de la velocidad de convergencia del método usado para aproximar el VIP(C, Ω).

Teorema 3 (Velocidad de convergencia) *El teorema de la* ***velocidad de convergencia****, considerando las hipótesis del teorema de convergencia, que el conjunto de soluciones del VIP(C, Ω) es único y que la solución en equilibrio satisface la condición de complementariedad, establece que la velocidad de convergencia del algoritmo DSD es la misma que la del método usado para resolver los problemas maestros restringidos RMPs.*

4.3. Algoritmos para el RMP (fase de equilibrado)

Los teoremas anteriores nos indican la importancia que tiene en el rendimiento del algoritmo DSD el algoritmo usado para resolver los RMPs. El teorema de convergencia nos muestra como la convergencia del DSD, así como la velocidad de convergencia, dependen de las propiedades de convergencia del algoritmo usado en el RMP(ℓ).

Para resolver el RMP(C, Ω^ℓ) proponemos **métodos de linealización** (ver [Pang and Chan1982], [Montero and Barceló1996]). Estos métodos generan una serie de problemas de desigualdades variacionales de la siguiente forma: sea $\hat{h}$ una solución factible para el RMP(ℓ), $\hat{h} \in \Omega^\ell$, aproximamos la función de coste $C(h)$ no lineal mediante la siguiente expresión lineal, reemplazando el RMP(ℓ) por VIP$(\tilde{C}(\cdot, \hat{h}), \Omega^\ell)$:

$$\tilde{C}(h,\hat{h}) = C(\hat{h}) + A(\hat{h})(h-\hat{h})$$

Para simplificar esta notación se suele expresar como:

$$\hat{C}(h) = \hat{b} + A(\hat{h})h$$

Asumiremos que esta aproximación satisface las hipótesis de que $\tilde{C} : \Omega \times \Omega \rightarrow \Re^n$ es continua en $\Omega \times \Omega$ y la matriz $A(\hat{h})$ es definida positiva para todo $\hat{h}$.

La elección de la matriz $A(\hat{h})$ puede realizarse de varias formas, siendo las más típicas:

- Método de Newton: $A(\hat{h}) = \nabla C(\hat{h})$
- Método quasi-Newton: $A(\hat{h}) \approx \nabla C(\hat{h})$
- Método de Jacobi linealizado: $A(\hat{h}) =$ la diagonal de $\nabla C(\hat{h})$
- Método de Proyección: $A(\hat{h})$ es una matriz simétrica positiva
- Método Gauss-Seidel linealizado($\gamma = 1$): $A(\hat{h}) = L(\hat{h}) + D(\hat{h})/\gamma$ o $U(\hat{h}) + D(\hat{h})/\gamma$ donde $D(\hat{h})$ es la diagonal de $\nabla C(\hat{h})$ y $L(\hat{h})$ y $U(\hat{h})$ son las matrices triangulares superior e inferior de $\nabla C(\hat{h})$.

La convergencia global de estos métodos se demuestra bajo la condición de que la función de coste es fuertemente monótona, $C(h)$ es estrictamente monótona si $[C(h_1) - C(h_2)]^T(h_1 - h_2) > 0, \quad \forall h_1 \neq h_2$. Se sabe que en los problemas de asignación de tráfico la función de coste no es estrictamente monótona, de modo que no tendríamos garantías sobre la convergencia del DSD al aplicar al RMP(C, Ω^ℓ) métodos de linealización que requieren de una fuerte monotonía para converger.

La introducción de búsquedas lineales para una determinada función de mérito nos han permitido relajar estas hipótesis de modo que podamos garantizar la convergencia de los métodos de linealización (ver el teorema 6.12 [Patriksson1999]).

En la tabla 4.2 resumimos cómo sería el método de linealización modificado. Asumimos que el VIP(C, Ω) satisface que C es continuamente diferenciable y monótona en Ω. Usaremos como función objetivo la función del gap restringida, que se define como:

$$G^\ell(h) := \text{Maximizar}_{h \in \Omega^\ell} C(h^\ell)^T (h - h^\ell)$$

Tabla 4.2: Método de linealización modificado para resolver el RMP(ℓ)

---

0. (*Inicialización*): Partir de un punto inicial $\hat{h}$ que es la solución del VIP$(\tilde{C}(\cdot, h^{\ell-1}), \Omega^\ell)$.
1. (*Calcular la dirección de búsqueda*): Resolver el VIP$(\tilde{C}(\cdot, \hat{h}), \Omega^\ell)$ y obtener una solución $\bar{h}$. La dirección de búsqueda sería $d := \bar{h} - \hat{h}$.
2. (*Criterio de parada*): Si $\hat{h}$ resuelve el VIP$(\hat{C}, \Omega^\ell)$ entonces parar, ya que $\hat{h}$ resuelve el RMP(ℓ). Si no, continuar.
3. (*Dirección de búsqueda en la función del gap restringido*): Moverse una cantidad α en la dirección de búsqueda, de forma que asignamos como nueva solución actual $\hat{h} = \hat{h} + \alpha^* d$ donde

$$\alpha^* \in \arg\ \text{minimizar}\{G^\ell(\hat{h} + \alpha d) \mid \hat{h} + \alpha d \in \Omega^\ell\}$$

4. (*Criterio de parada*): Si $G^\ell(\hat{h}) \leq \epsilon^\ell$ entonces $h^\ell := \hat{h}$ y parar. Si no, volver al paso 1.

Podemos utilizar varios métodos para resolver el RMP(ℓ). El mejor sería el método de Newton ya que presenta una velocidad de convergencia cuadrática, mientras que otros métodos como los de proyección presentan una convergencia lineal, necesitando muchas iteraciones para llegar a una solución con cierta precisión, aunque por otro lado el cálculo de la dirección de búsqueda en estos métodos es más sencillo.

Método de Frank-Wolfe para obtener la dirección de búsqueda.

En el paso 1 del método de linealización planteado (tabla 4.2) tenemos que calcular la dirección de búsqueda. Hemos planteado un método que resuelve el VIP$(\hat{C}, \Omega^\ell)$ mediante la resolución de un número finito de sistemas de ecuaciones lineales $LS(\mathcal{P}_A)$ de la forma:

$$\begin{array}{lcll} C_p(\hat{h}) + \sum_{q\in\mathcal{P}_A} A_{pq}(\hat{h})(h_q - \hat{h}_q) & = & U_\omega, & p \in \mathcal{P}_A \\ \sum_{p\in\mathcal{P}_A} h_p & = & g_\omega, & \omega \in W^\ell \\ h_p & = & 0, & p \in \mathcal{P}^\ell - \mathcal{P}_A \end{array}$$

donde $\mathcal{P}_A$ es el subconjunto de caminos del conjunto actual $\mathcal{P}^\ell$ y se va actualizando de una iteración a otra, de forma que en la última iteración sólo contiene el subconjunto de los caminos con flujo positivo de la solución del VIP$(\hat{C}, \Omega^\ell)$.

Este sistema tiene $|\mathcal{P}_A|+|W^\ell|$ variables, (h_p, U_ω) con $p \in \mathcal{P}_A$ y $\omega \in W^\ell$, lo que corresponde a $|\mathcal{P}_A|+|W^\ell|$ ecuaciones. El algoritmo DSD con identificación de pares es una estrategia que limita la dimensión de estos sistemas de ecuaciones.

El método propuesto podría ser visto como una especialización del algoritmo de Frank-Wolfe, aplicado a una reformulación del VIP$(\hat{C}, \Omega^\ell)$ mediante un problema de optimización equivalente.

Para la formulación del TAP como un problema de optimización, asumiremos que existe un vector $\bar{h}$ de flujos en equilibrio, entonces podemos decir que un vector h^* está en equilibrio si y solo si existen dos vectores π^* y U^* tales que (h^*, U^*, π^*) es una solución del problema de optimización $OP(C, \Omega)$:

$$\begin{array}{lll} \text{Minimizar}_{\pi,h,U} & Z = \pi^T h & \\ \text{Sujeto a:} & C(h) = \Lambda^T U + \pi & \\ & \Delta h = g & \\ & \pi, h \geq 0 & \end{array}$$

La necesidad de asegurar la convergencia del algoritmo DSD nos lleva a usar un método de linealización en el que la matriz $A(\hat{h})$ sea definida positiva (para asegurar la condición de monotonía), ya que en este caso, el problema de optimización OP $(\hat{C}, \Omega^\ell)$ se convierte en cuadrático convexo. Para resolver el OP$(\hat{C}, \Omega^\ell)$ proponemos utilizar una especialización del método de Frank-Wolfe (FW) tal y como podemos ver en la tabla 4.3.

Usando el método de FW el algoritmo converge en un número finito de iteraciones. Bajo las mismas condiciones del teorema de la velocidad de convergencia y considerando

Tabla 4.3: Método FW para resolver el VIP$(\hat{C}, \Omega^\ell)$

1.0. (*Inicialización*): Tenemos una solución actual $\hat{h}$ a la que se le han añadido caminos en la etapa CGP. En ella $\hat{h} \geq 0$, pero los nuevos caminos generados en el CGP no tienen flujo y son de mínimo coste.

1.1. (*Criterio de parada*): Calcular $\hat{\pi}$ y comprobar si $\hat{\pi}^T \hat{h} = 0$. Si se cumple, salir, ya que $\hat{h}$ es solución del VIP$(\hat{C}, \Omega^\ell)$. En caso contrario, continuar con el paso 1.2.

1.2. (*Actualización de caminos considerados*): Modificar el conjunto de caminos $\mathcal{P}_A$ para considerar los caminos mínimos ($\hat{\pi}_p = 0$) y los que tienen flujo y no son de mínimo coste $\hat{\pi}_p > 0$ y $\hat{h}_p > 0$.

$$\mathcal{P}_A = \{p \in \mathcal{P}^\ell \mid \hat{h}_p \hat{\pi}_p > 0\} \cup \{p \in \mathcal{P}^\ell \mid \hat{\pi}_p = 0\}.$$

1.3. (*Obtención de dirección de búsqueda*): Obtener una nueva solución provisional $\bar{h}$ tras resolver el LS$(\mathcal{P}_A)$. Lo cual define la dirección de búsqueda $d = \bar{h} - \hat{h}$.

1.4. Calcular el paso (cuanto nos podemos mover en la dirección de búsqueda) por cada par y almacenar en λ.
En el cálculo de λ distinguir dos casos, para cada $\omega \in W_\ell$:

   a) Si todos los $p \in P_\omega$ tienen flujo positivo en la solución provisional, $\bar{h}_p > 0$, tomar $\lambda_\omega = 1$ (moverse el máximo en esa dirección).

   b) Si no son todos positivos, calcular el máximo movimiento sin salirnos de la región factible, siendo $\lambda_\omega = \text{mín}_{d_p<0} \frac{-\hat{h}_p}{d_p}, \quad \forall p \in P_\omega$

1.5. (*Actualización de la solución*): Por todo $\omega \in W^p$, moverse la cantidad λ_ω en la dirección de búsqueda, obteniendo como solución actual $\hat{h}_\omega \rightarrow \hat{h}_\omega + \lambda_\omega d_\omega$. Volver al paso 1.1.

que los métodos de linealización generan una secuencia $\{\hat{h}^s\}$ la cual converge a la solución h^*, podemos decir que existe un entero τ de forma que el método FW converge en una sola iteración (tras resolver un único sistema $LS(\mathcal{P}_A)$) para todo $s \geq \tau$.

Resumen de ideas principales

A continuación vamos a intentar fijar ideas de lo visto anteriormente. Para resolver VIP(C,Ω^ℓ) empleamos métodos de linealización que requieren resolver varias desigualdades lineales afines.

A su vez para resolver cada VIP($\hat{C}, \Omega^\ell$) tenemos que resolver varios sistemas de ecuaciones lineales denominados LS($\mathcal{P}_A$). La estrategia de identificación de pares reduce la dimensionalidad de este sistema. Notar que resolver VIP($\hat{C}, \Omega^\ell$) es resolver el siguiente sistema de igualdades y desigualdades

$$\begin{aligned} &C(\hat{h}) + A(\hat{h})(h - \hat{h}) = \Lambda^T U + \pi \\ &\Delta h = g \\ &\pi, h \geq 0 \end{aligned}$$

El algoritmo de FW va eligiendo caminos (identificando cuales van a llevar flujo en la situación de equilibrio $h_p > 0$). Para dichos caminos (almacenados en el conjunto $\mathcal{P}_A$) se sabe que el multiplicador $\pi_p = 0$. Si relajamos la condición $h \geq 0$ el anterior problema se convierte en un sistema de ecuaciones lineales. Si su solución es no negativa se trata del flujo en equilibrio pero si existe alguna componente con flujo negativo tenemos que plantear un nuevo sistema. Este proceso converge en un número finito.

Si la aproximación se toma $A(\hat{h}) = DC(\hat{h})$ (método de Newton) la velocidad de convergencia es cuadrática pero los problemas VIP($\hat{C}, \Omega^\ell$) son difíciles de resolver.

Métodos de Proyección y Jacobi linearizado

Ahora vamos a discutir las aproximaciones (método de proyección y de Jacobi linearizado) en las que el VIP($\hat{C}, \Omega^\ell$) se puede obtener de forma cómoda. El inconveniente es que la velocidad de estos métodos es lineal.

Tabla 4.4: Método de Jacobi para resolver el VIP$(\hat{C}, \Omega^{\ell})$

1. Ordenar los caminos de menor a mayor basándonos en el vector b.
   ($b_1 < b_2 < ... < b_s$, siendo s en número de caminos del par).
2. Tomar $q = 2$.
3. Calcular u:
$$u = \frac{g_i + \sum_{p=1}^{q} \frac{b_p}{D_p}}{\sum_{p=1}^{q} \frac{1}{D_p}}$$
4. Si $u - b_q > 0$ entonces tomar $q = q + 1$ y volver al paso 3.
   Si $u - b_p \leq 0$ entonces ir al paso 5.
5. Calcular la situación de equilibrio para los caminos ordenados $\{1, 2, ..., q-1\}$ empleando la fórmula:
$$u_i = \frac{b_i + \sum_{p=1}^{q-1} \frac{b_p}{D_p}}{\sum_{p=1}^{q-1} \frac{1}{D_p}}$$
$$\bar{h}_p = \frac{u_p - b_p}{D_p} \qquad p = 1, ...q-1$$

En este método calculamos el gap restringido, el cuál se realiza sobre los caminos y costes mínimos (u) de los caminos considerados en el RMP, tal y como podemos ver:

$$GAP_{Restringido} = \sum_{p \in K^l} h_p C_p - \sum_{i \in I} u_i^* g_i$$

donde p son los caminos del RMP e i los pares retenidos en el RMP.

Tabla 4.5: Método de Proyección para resolver el VIP$(\hat{C}, \Omega^{\ell})$

Consiste en tomar

$$D = \alpha I, \quad \text{siendo } I \text{ la matriz identidad.}$$

Quedando las fórmulas para obtener la solución de los problemas lineales:

$$u = \frac{g_i + \sum_{p=1}^{q} \frac{b_p}{\alpha}}{\sum_{p=1}^{q} \frac{1}{\alpha}} = \frac{g_i + \frac{1}{\alpha}\sum_{p=1}^{q} b_p}{q\frac{1}{\alpha}} = \alpha g_i + \sum_{p=1}^{q} \frac{b_p}{q}$$

$$h_p = \frac{\alpha_g + \bar{b} - b_p}{\alpha} = g + \frac{1}{\alpha}(\bar{b} - b_p)$$

4.4. Ejemplo numérico

En esta sección analizaremos de forma práctica como iríamos alcanzando la solución en equilibrio en un ejemplo utilizando el algoritmo DSD propuesto, en el cual utilizamos el método de Newton para el equilibrado en el RMP. Para ello utilizaremos la red de Nguyen Dupuis vista en el capítulo anterior, cuya configuración aparece en la figura **??** y cuya matriz origen destino aparece en la tabla 3.2.

La idea es centrarse en la etapa de equilibrado realizada con el método de Newton, por lo que no entraremos en detalles de flujos y costes de arcos, estructura de nodos y arcos de los caminos y demás detalles.

En cada iteración del algoritmo DSD agregamos nuevos caminos en al etapa CGP y mejoramos la solución actual $\hat{h}$ en la etapa RMP, prestando especial interés en esta y analizando las situaciones que se van dando y cómo responde el algoritmo. Al comienzo de cada iteración mostramos la matriz par-camino en la que podemos ver en la situación actual qué caminos se están utilizando para satisfacer cada par.Para reflejar la situación de cada paso del algoritmo de equilibrado utilizamos un conjunto de tablas con los datos internos del sistema.

DSD Iteración 1

En la primera iteración tenemos un conjunto de caminos, creado en la etapa CGP, en el que cada par tiene asociado el camino mínimo del par para la situación que no existiera flujo en la red.

Pares	Caminos
ω_1	p_1
ω_2	p_2
ω_3	p_3
ω_4	p_4

La solución en esta primera iteración corresponde a la solución sin considerar los efectos de la congestión. En ella toda la demanda de cada par es asignada a su único camino asociado (camino mínimo).

$\hat{h}_p$	C_p
400	105
800	101
600	98
200	36

DSD Iteración 2

En la segunda iteración del DSD ya se han actualizado los costes iniciales de la red debido a que circula flujo en la misma y congestiona las rutas iniciales, de forma que tras la fase CGP se generan nuevos caminos mínimos por par que son añadidos al conjunto de caminos. En el par ω_4 no se añade un nuevo camino debido a que el camino mínimo obtenido en la fase CGP es el mismo que el teníamos.

Pares	Caminos	
ω_1	p_1	p_5
ω_2	p_2	p_6
ω_3	p_3	p_7
ω_4	p_4	

RMP Paso 1

La configuración del sistema en este punto es la que mostramos a continuación. En este punto es en el que comprobamos si la solución está en equilibrio y es la solución que estamos buscando, en cuyo caso saldremos del equilibrado. Tenemos una solución actual $\hat{h}$, en la que por los nuevos caminos no hay asignado flujo. Estos nuevos caminos son los de menor coste $\pi_p = 0$, es por esto por lo que tenemos que pasar a equilibrar los pares ω_1, ω_2 y ω_3. El par ω_4 sólo tiene un camino y es eliminado del equilibrado (fundamento de la estrategia de identificación de pares).

$\hat{h}_p$		C_p		U_ω	π_p	
400	0	105	44	44	61	0
800	0	101	51	51	50	0
600	0	98	38	38	60	0
200		36		36	0	

Al resolver el LS y obtenemos la siguiente solución provisional $\bar{h}$. Podemos ver como algunos flujos obtenidos son negativos, la solución provisional obtenida se ha salido de la región factible. Esto se arreglará en el siguiente paso, en el que moveremos la solución actual $\hat{h}$ lo máximo en la dirección de búsqueda hacia $\bar{h}$ pero sin entrar en la negatividad de los flujos.

$\bar{h}_p$		C_p		U_ω	π_p	
490	-90	59	44	44	15	0
200	600	62,9	62,9	62,9	0	0
-100	700	98	51,2	51,2	46,8	0
200		39,5		39,5	0	

A continuación debemos realizar el movimiento máximo dentro de la dirección de búsqueda $\hat{h}_\omega = \hat{h}_\omega + \lambda_\omega d_\omega$. Para ello debemos saber cuanto nos podemos mover por cada par, así que calculamos λ obteniendo:

λ_1	λ_2	λ_3	λ_4
0	1	0,84	0

Podemos interpretar este movimiento obtenido. En el par ω_1 no debemos movernos ($\lambda_1 = 0$) ya que la solución provisional entraba en valores negativos, nos quedamos con la solución actual. En el par ω_2 nos movemos completamente hacia la solución provisional

($\lambda_2 = 1$) ya que hemos obtenido valores positivos. En el par ω_3 debemos de movernos casi todo hacia la solución provisional ($\lambda_3 = 0{,}84$), pero no todo porque obtendríamos un flujo negativo.

RMP Paso 2

La solución actual $\hat{h}$ ha cambiado después del movimiento, obteniendo la siguiente configuración. Podemos ver como ω_3 ya está en equilibrio, asignándose toda la demanda por el camino de mínimo coste. Sin embargo ω_1 y ω_2 no lo están y su flujo esta asignado a caminos no mínimos, así que debemos equilibrarlos.

$\hat{h}_p$		C_p		U_ω	π_p	
400	0	58,5	44	44	14,5	0
200	600	62	64,5	62	0	2,4
0	600	98	50	50	48	0
200		39		39	0	

Tras resolver el LS con los pares ω_1 y ω_2 obtenemos la siguiente solución provisional $\bar{h}$.

$\bar{h}_p$		C_p		U_ω	π_p	
399,5	0,5	59,7	59,7	59,7	0	0
238,8	561,2	63,7	63,7	63,7	0	0
0	600	98	49,9	49,9	48,1	0
200		39		39	0	

La solución obtenida muestra flujos positivos, por lo que podemos movernos el máximo en la dirección de búsqueda.

λ_1	λ_2	λ_3	λ_4
1	1	0	0

Tras realizar esto, obtenemos como solución actual $\hat{h}$, que es exactamente la misma configuración que teníamos en la solución provisional $\bar{h}$. Podemos ver como ahora sí se cumple el criterio de parada ya que todos los caminos que tienen flujo asignado tienen asociados un coste mínimo igual al coste de equilibrio ($\hat{h}_p \cdot \pi_p = 0$), así que salimos y pasamos a la siguiente iteración.

DSD Iteración 3

En esta iteración, el camino p_3 es desechado en el proceso previo de eliminación de caminos, ya que $\hat{h}_3 = 0$. A continuación se realiza otra fase de CGP en la que generamos un nuevo camino mínimo por par según la configuración actual de la red, siempre y cuando el camino no esté repetido, quedando el conjunto de caminos de esta forma:

Pares	Caminos		
ω_1	p_1	p_5	p_8
ω_2	p_2	p_6	p_9
ω_3	p_7		
ω_4	p_4		

RMP Paso 1

La solución actual del sistema $\hat{h}$ es la siguiente. Tenemos que equilibrar los pares ω_1 y ω_2, ya que la demanda no va por los caminos de coste mínimo.

$\hat{h}_p$		
399,5	0,5	0
238	561	0
600		
200		

C_p		
59,7	59,7	42,6
63,7	63,7	47
50		
39		

U_ω
42,6
47
50
39

π_p		
17,1	17,1	0
16,7	16,7	0
0		
0		

Tras la resolución del LS obtenemos la solución provisional $\bar{h}$.

$\bar{h}_p$		
48	-268	620
264	210	325
600		
200		

C_p		
52,2	59,7	52,2
54,3	54,3	54,3
47		
43,8		

U_ω
52,2
54,3
47
43,8

π_p		
0	7,5	0
0	0	0
0		
0		

Nos movemos en la dirección de búsqueda. Para ello calculamos λ.

λ_1	λ_2	λ_3	λ_4
0,0018	1	0	0

En el par ω_1, la solución provisional nos dice que tenemos que quitar flujo del camino p_5 y darle más al p_8, pero podemos ver como el λ ha salido muy pequeño, así que nos moveremos muy poco, lo justo para quitar de la solución actual el valor $\hat{h}_5 = 0,48$ que apenas es significativo. En el par ω_2 nos movemos todo hacia la solución provisional, ya que han salido todos los flujos positivos.

RMP Paso 2

La solución actual $\hat{h}$ ha cambiado tras movernos en la dirección de búsqueda calculada. Podemos ver como todavía hay caminos en los pares ω_1 y ω_2 que tienen flujo y no son de coste mínimo, así que realizamos otro equilibrado.

$\hat{h}_p$

398	0	2
264	210	325
600		
200		

C_p

64,6	59,7	39,1
65,7	53,4	58,7
49		
43,8		

U_ω

39,1
53,4
49
43,8

π_p

25,2	20,6	0
12,2	0	5,2
0		
0		

Obtenemos la siguiente solución provisional $\bar{h}$ al resolver el LS.

$\bar{h}_p$

-72	0	472
369	75	355
600		
200		

C_p

64,6	59,7	47,1
55,1	55,1	55,1
48,6		
44,3		

U_ω

47,1
55,1
48,6
44,3

π_p

17,4	12,5	0
0	0	0
0		
0		

Calculamos λ, nos movemos en la dirección de búsqueda y obtenemos otra solución actual $\hat{h}$.

λ_1	λ_2	λ_3	λ_4
0,84	1	0	0

En el par ω_1, la solución provisional nos indica que tenemos que quitar flujo del camino p_1 en favor del camino p_8, dejando a p_5 sin flujo. El λ_1 obtenido hace que todo el flujo que quedaba en p_1 vaya por p_8, dejando a p_1 y p_5 con flujo 0.

RMP Paso 3

Después del equilibrado anterior ya tenemos el par ω_1 equilibrado, pero aun tenemos que equilibrar el par ω_2.

$\hat{h}_p$			C_p			U_ω	π_p		
0	0	400	64,4	59,7	45,7	45,7	18,8	13,9	0
369	75	355	57,5	54,4	56	54,4	3	0	1,6
600			48,6			48,6	0		
200			44,3			44,3	0		

Tras realizar un LS sobre los caminos del par ω_2 obtenemos la siguiente solución provisional $\bar{h}$.

$\bar{h}_p$			C_p			U_ω	π_p		
0	0	400	64,6	59,7	46,2	46,2	18,3	13,5	0
320	120	350	55,4	55,4	55,4	55,4	0	0	0
600			48,8			48,8	0		
200			44,3			44,3	0		

En la solución obtenida tenemos todos los flujos positivos, así que nos movemos completamente hacia la dirección de búsqueda calculada ($\lambda_1 = 1$). Podemos ver como ya tenemos todos los pares equilibrados, se cumple la condición de parada ($\hat{h}_p * \pi_p = 0$)

DSD Iteración 4

En la siguiente iteración eliminamos los caminos p_1 y p_5, ya que tienen flujo 0. A continuación realizamos la etapa CGP agregando al conjunto de caminos nuevos caminos mínimos no repetidos, quedando la configuración de la siguiente forma:

Pares	Caminos			
ω_1	p_8			
ω_2	p_2	p_6	p_9	p_{10}
ω_3	p_7	p_{11}		
ω_4	p_4			

RMP Paso 1

La solución actual $\hat{h}$ es la siguiente. Según podemos ver, hemos de equilibrar los pares ω_2 y ω_3, ya que en ellos se ha introducido un nuevo camino mínimo para el que no se ha asignado demanda.

$\hat{h}_p$			
400			
320	120	350	0
600	0		
200			

C_p			
46,2			
55,4	55,4	55,4	54,8
48,8	42,4		
44,3			

U_ω
46,2
54,8
42,4
44,3

π_p			
0			
0,6	0,6	0,6	0
6,3	0		
0			

Tras resolver el LS hemos obtenido la siguiente solución provisional $\bar{h}$.

$\bar{h}_p$			
400			
250	120	360	60
500	100		
200			

C_p			
47,5			
55,5	55,5	55,5	55,5
47,1	47,1		
43,9			

U_ω
47,5
55,5
47,1
43,9

π_p			
0			
0	0	0	0
0	0		
0			

Podemos ver como la solución provisional es factible, ya que todos los flujos son positivos, así que nos movemos todo lo que podemos en esa dirección de búsqueda ($\lambda_2 = 1$ y $\lambda_3 = 1$), obteniendo como solución actual $\hat{h}$ la misma que la provisional. Esta solución ya está en equilibrio, y cumplimos el criterio de parada, así que pasamos a la siguiente iteración.

DSD Iteración 5

Realizamos la fase CGP y se genera un nuevo camino mínimo para el par ω_2, quedando el siguiente conjunto de caminos:

Pares	Caminos				
ω_1	p_8				
ω_2	p_2	p_6	p_9	p_{10}	p_{12}
ω_3	p_7	p_{11}			
ω_4	p_4				

RMP Paso 1

La solución actual es la siguiente:

$\hat{h}_p$

400				
250	120	360	60	0
500	100			
200				

C_p

47,5				
55,5	55,5	55,5	55,5	55,5
47,1	47,1			
43,9				

π_p

0				
0	0	0	0	0
0	0			
0				

Podemos ver como la situación actual, con el nuevo camino incluido, ya por sí es solución válida y no hay que equilibrar ningún par. Así que salimos del equilibrado.

A continuación se eliminará el camino p_{12} ya que tiene flujo 0, y la etapa CGP ya no añadirá ningún camino mínimo nuevo. El sistema está en equilibrio y salimos del algoritmo DSD con esta solución.

Notar que en este ejemplo los costes en los arcos son lineales por lo que se ha encontrado la solución exacta al problema.

Notar que el número de sistemas lineales que resuelve es elevado pero existe un resultado teórico que nos dice que cuando el algoritmo progresa y esté en una solución casi-óptima con un solo sistema LS obtenemos la solución del VIP$(\hat{C}, \Omega^{\ell})$.

Capítulo 5

Codificación Eficiente

En este capítulo analizaremos como hemos codificado nuestro algoritmo DSD con identificación de pares propuesto, analizando las estructuras de datos usadas, el algoritmo DSD y las subetapas CGP y RMP. La idea general ha sido la de buscar la mejor optimización posible, para ello hemos tomado decisiones de diseño tras analizar varias opciones y hemos desarrollado varias versiones de los algoritmos, de forma que obtengamos un mejor tiempo de cómputo.

5.1. Estructuras de datos

Los datos de entrada de los algoritmos de asignación de tráfico son una configuración de la red de transporte y una matriz origen-destino con la demanda a asignar. Normalmente, cuando se estudian redes reales y se almacena toda esta información, se hace intentando seguir un standard, el formato empleado por Bar-Gera en su tesis doctoral.

Según el formato Bar-Gera, la configuración de la red viene expresada en un fichero de texto plano, en el que aparece una cabecera y a continuación una línea por cada arco, en la que aparecen todos sus datos. Presentando la siguiente estructura:

```
~ InitNode TermNode Capacity Length FreeFlowTime B PowerSpeedLimit
Toll Type;

 1     5  1    7.0  7.0  0.0125  1 0 0 1 ;
 1    12  1    9.0  9.0  0.01    1 0 0 1 ;
 4    5   1    9.0  9.0  0.01    1 0 0 1 ;
 4    9   1   12.0 12.0  0.005   1 0 0 1 ;
 5    6   1    3.0  3.0  0.0075  1 0 0 1 ;
 5    9   1    9.0  9.0  0.0075  1 0 0 1 ;
...
```

El fichero Bar-Gera con la matriz origen-destino está estructurado por orígenes, y dentro de cada uno va indicando los destinos y la demanda a asignar. Presenta la siguiente estructura:

```
Origin 1
 2 : 400 ; 3 : 800 ;
Origin 2
Origin 3
Origin 4
 2 : 600 ;  3 : 200 ;
```

Debido a la extensión del formato Bar-Gera y a los datos de redes reales que podemos encontrar, hemos optado por incorporarlo. Sin embargo en nuestro algoritmo buscando una mayor optimización usamos unas estructuras internas propias basadas en punteros que nos permiten un acceso más rápido a los elementos y que posteriormente pasamos a comentar. Por este motivo hemos desarrollado un programa complementario (cuyo código puede ser consultado en el Manual de Código) que se encarga de leer los datos comentados en formato Bar-Gera y pasarlos a nuestro formato propio, precalculando los índices de punteros necesarios, algo que sólo habrá que hacer esta vez y luego permitirá que el algoritmo gane en velocidad de cómputo.

La idea básica de nuestra estructura de datos interna está basada en la propuesta de Gallo-Pallotino en el código L2queue. Originalmente utiliza tres vectores para representar la red: 'fout', 'nd' y 'nxtout'. El vector 'fout' tiene de tamaño el número de nodos y en sus casillas presenta un puntero al vector 'nd' en el que aparecen los nodos destino, estableciéndose así los arcos de la red. El vector 'nxtout' nos proporciona un puntero hacia el siguiente nodo final del nodo origen que estamos procesando, de forma que sepamos cuando parar de leer en el vector 'nd'. En nuestro caso hemos optado por eliminar este vector 'nxtout' ya que mediante programación podemos saber exactamente el número de posiciones que hemos de leer en 'nd'.

La matriz origen-destino se puede considerar como una red en la que los arcos nos dicen el origen y destino del par y el coste del arco la demanda que debemos asignar. Realizando este paralelismo utilizamos una estructura similar a la anterior, haciendo uso de los vectores 'foutTrips' y 'ndTrips'.

En figura 5.1 podemos ver un ejemplo de como quedarían estas estructuras con la red de Nguyen-Dupuis.

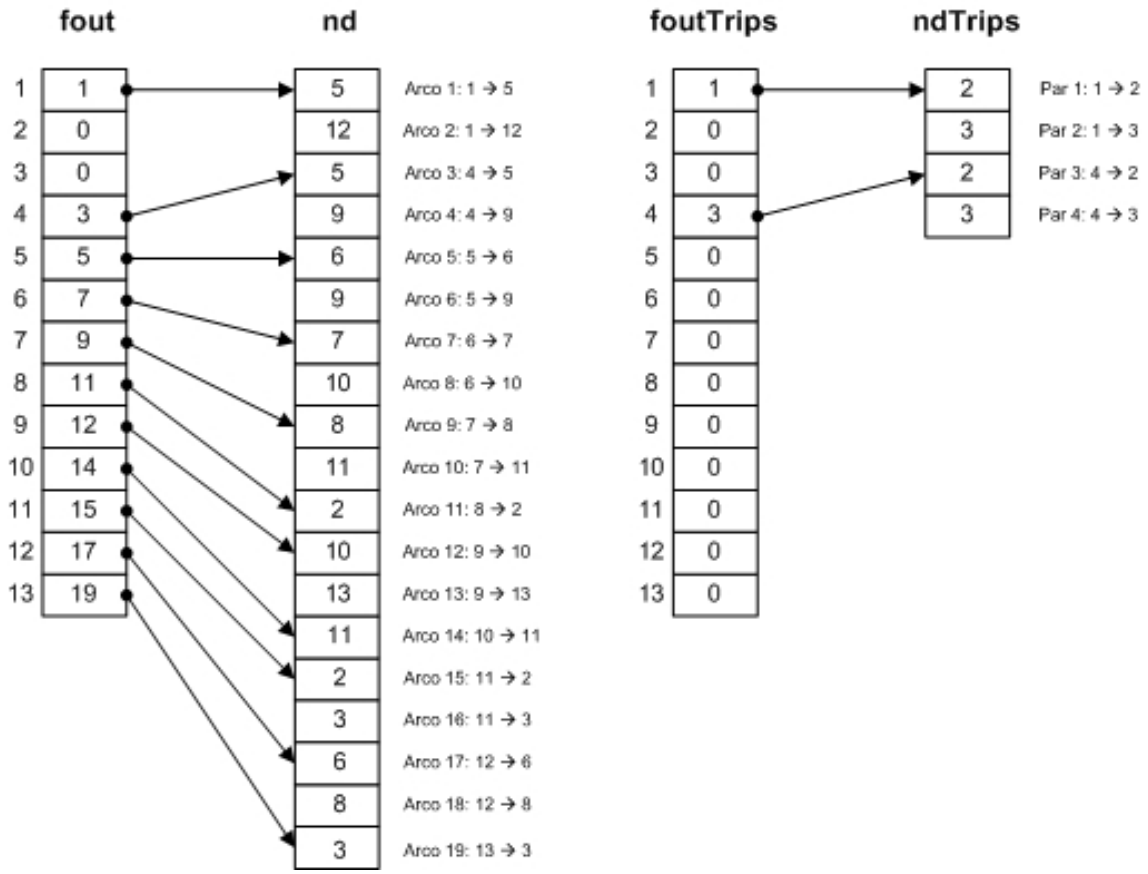

Figura 5.1: Estructuras de datos

Para el resto de datos, planteamos inicialmente un enfoque centrado en la creación de estructuras de tipo arco y camino que encapsularan los valores asociados de flujo, coste,... Tras comprobar los tiempos de cálculo empleados con el manejo de estas estructuras y compararlos con el uso de vectores, vimos que era mejor la segunda opción. De esta forma, podemos ver a continuación el resto de estructuras que usamos en nuestro programa.

fout	arcosVa	caminosVp	demandasTrips
nd	arcosCa	caminosCp	ParesIdentificados
foutTrips	arcosKa	caminosWp	vParesIdentificados
ndTrips	arcost0		paresConsiderados
	arcosNa		caminosIdentificados
	arcosAa		vCaminosIdentificados
	arcosCa1		caminosConsiderados

Además de estos vectores, mantenemos cuatro matrices en las que podemos consultar rápidamente las relaciones entre pares, caminos, arcos y nodos, que necesitamos en el desarrollo del algoritmo. Estas matrices por motivos de optimización están sobredimensionadas, estableciendo unos límites de número de arcos por camino y número de caminos por par, definidos en el algoritmo como parámetros de entrada.

deltaka: Secuencia de arcos de cada camino.

deltakn: Secuencia de nodos de cada camino.

daltawk: Caminos asociados a cada par.

deltaak: Caminos que utilizan cada arco.

5.2. Algoritmo DSD

El proceso de creación del algoritmo DSD lo comenzamos en un primer lugar mediante la codificación de un algoritmo DSD clásico, para el cuál se realizaron varias versiones. La resolución de la fase de equilibrado se realizó utilizando el paquete de optimización de MATLAB, concretamente la minimización con restricciones 'fmincon'. A este algoritmo se le añadió la estrategia de identificación de pares, así como otras mejoras como la incorporación del hessiano a la función a minimización para reducir el tiempo de cómputo, pero tal y como comentamos en el siguiente capítulo no obtuvimos los resultados esperados.

Como hemos visto, en nuestro algoritmo DSD con identificación de pares, utilizamos para la etapa de equilibrado la resolución de varios sistemas lineales más rápidos de resolver, y además actuamos sobre un subconjunto de caminos considerados lo cual reduce la dimensionalidad del sistema a resolver.

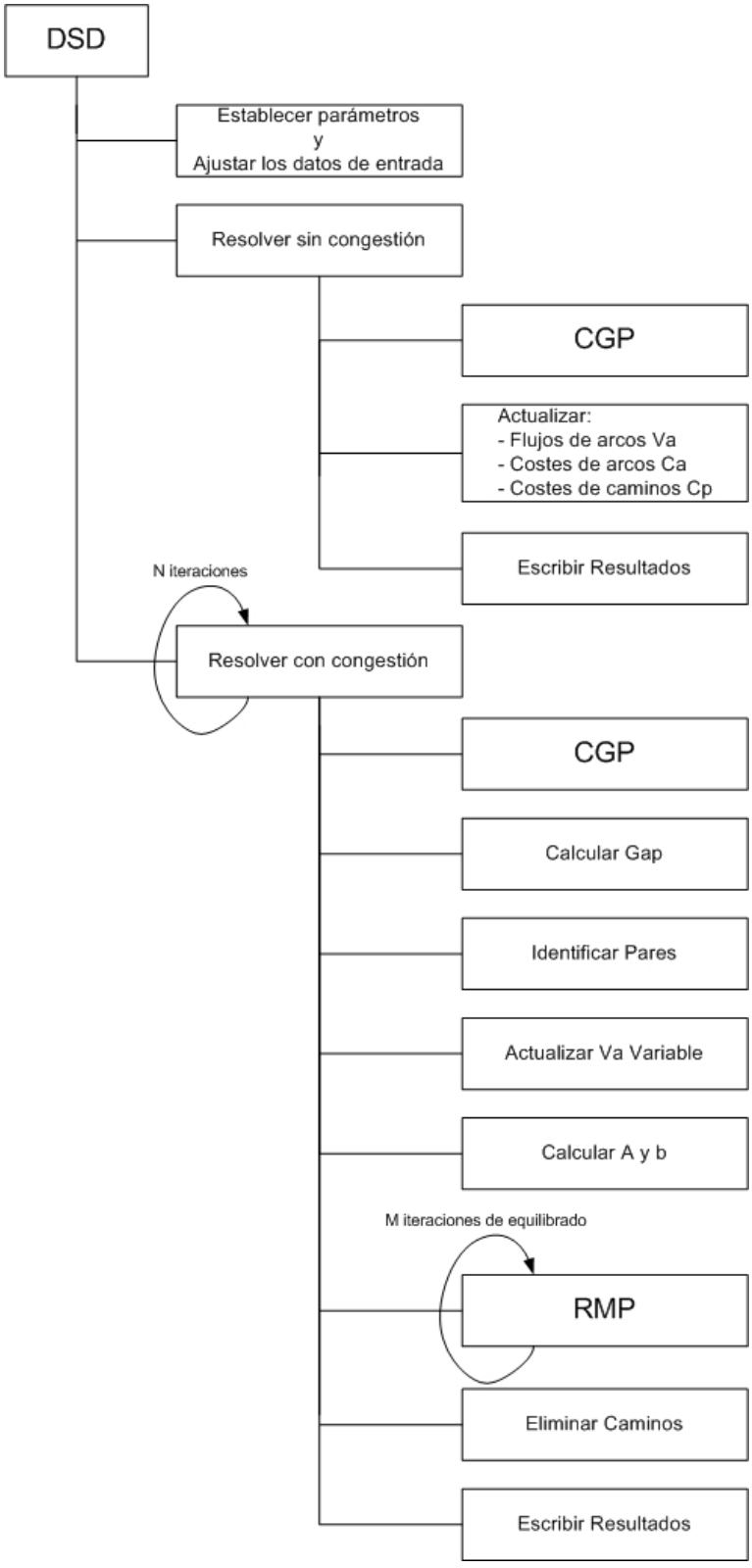

Figura 5.2: Esquema del algoritmo DSD

Después del desarrollo de dicho algoritmo realizamos un proceso de optimización del mismo, en el que analizamos el tiempo de cómputo de cada parte, detectando cuellos de botella y mejorando diversas partes del algoritmo. Podemos comentar algunas mejoras realizadas, como el cálculo de A y b de forma simétrica o la substitución de las comparaciones de pertenencia de un elemento a un vector que inicialmente realizábamos mediante la función 'ismember' de MATLAB, la cual presenta un cuello de botella importante y es una operación muy utilizada en todo el algoritmo, y modificamos para realizarlas mediante la consulta rápida y directa en unos vectores de pertenencia.

En la figura 5.2 podemos ver la estructura de nuestro algoritmo DSD. En esta sección nos centramos en la estructura del DSD, mientras que en las siguientes secciones estudiaremos sus dos subproblemas principales: la fase de generación de columnas CGP y la fase de equilibrado RMP.

Podemos ver como, tras una etapa de inicialización, pasamos a resolver el problema sin considerar los efectos de la congestión, es decir, generamos un camino mínimo para cada par y asignamos toda la demanda del par a dicho camino. Tras esto, actualizamos la configuración de la red y pasamos a resolver el problema considerando los efectos de la congestión.

Para resolver el problema considerando la congestión, deberemos realizar varias iteraciones consistentes en añadir nuevos caminos, identificar, realizar varios equilibrados y eliminar caminos. Podemos centrarnos en este punto y explicar brevemente lo que se hace en cada etapa.

CGP: Añadimos nuevos caminos resolviendo el subproblema CGP (comentado en detalle más adelante).

Calcular Gap: Calculamos el gap como la diferencia entre el coste de todos los caminos menos el coste de los caminos nuevos de la última iteración, y el gap relativo como el gap partido del valor absoluto de la función objetivo en la solución anterior.

Identificar Pares: Actualiza el conjunto de pares a considerar en el equilibrado. Para ello selecciona aquellos que tengan más de un camino asociado.

Actualizar Va Variable: Actualiza los flujos de los arcos de acuerdo con la información del flujo actual de los caminos, pero sólo considerando los caminos que han sido identificados.

Calcular A y b: Considerando los pares identificados construimos la matriz A y el vector b que nos definen las ecuaciones del LS.

La matriz A está formada por cuatro partes componentes y el vector b por dos, tal y como vemos en el siguiente esquema:

A		b
A_1	A_2	b_1
A_3	A_4	b_2

La parte superior A_1, A_2 y b_1 refleja las restricciones que nos proporcionaran un coste de equilibrio en la solución, mientras que la parte inferior A_3, A_4 y b_2 nos refleja las restricciones que hacen que el flujo de los caminos de cada par satisfaga la demanda.

Pasamos a describir lo que representa concretamente cada parte de la matriz y vector.

A_1**:** grado de relación de unos caminos con otros, calculado mediante la suma de las derivadas de sus arcos.

A_2**:** relación de los pares y caminos, indicando con '-1' los caminos que satisfacen cada par y '0' los que no.

A_3**:** relación de los pares y caminos, indicando con '1' los caminos que satisfacen cada par y '0' los que no.

A_4**:** ceros.

b_1**:** los costes de los caminos, calculados mediante la suma del valor del cte de sus arcos.

b_2**:** las demandas totales de los pares identificados.

RMP: Realizamos varias iteraciones de equilibrado en las que resolvemos el subproblema RMP (comentado en detalle más adelante).

Eliminar Caminos: Eliminamos aquellos caminos con flujo cero o próximo a cero (menor que un ϵ definido).

5.3. Algoritmo para CGP

La etapa de generación de columnas se basa en el cálculo de caminos mínimos. Tras el estudio realizado, seleccionamos como algoritmo base el L2queue de G.Gallo y S.Pallottino, que es el que actualmente proporciona mejores resultados, pero modificándolo para poder usar nuestras estructuras de datos. Podemos ver un esquema completo de la etapa CGP en la figura 5.3.

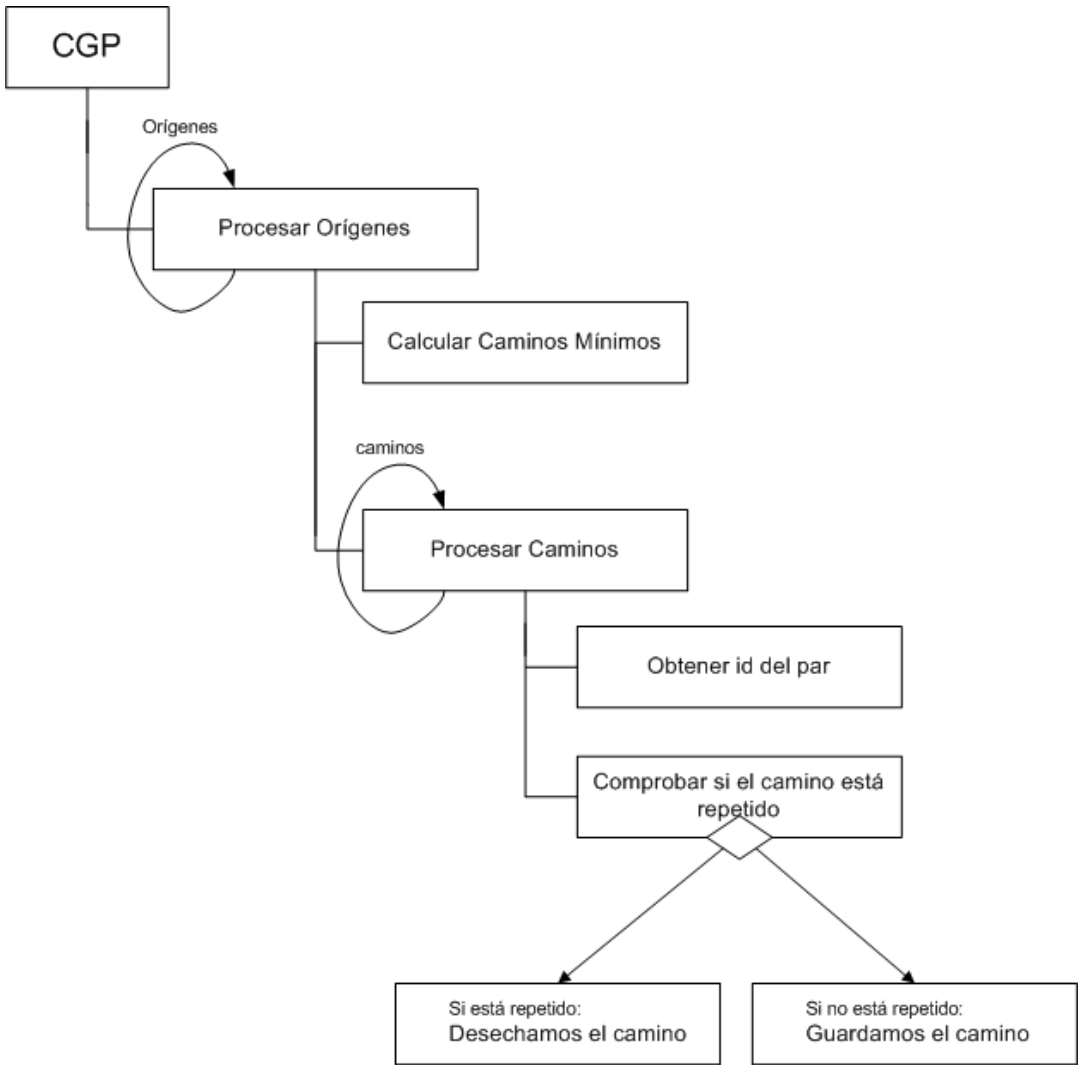

Figura 5.3: Esquema del algoritmo CGP

Procesar orígenes: El algoritmo CGP se basa en recorrer todos los orígenes de demandas, calculando para cada origen los caminos mínimos de ese origen a todos los destinos indicados.

Calcular caminos mínimos: El algoritmo de calculo de caminos mínimos parte de un origen dado, el número de destinos a calcular y un vector 'shdist' en el que indicamos con '1' nodos destino de los que queremos calcular su camino mínimo o '-1' para los demás. La salida es un vector de distancias mínimas calculadas y un vector de predecesores mediante el cuál podemos conocer la secuencia de nodos de los caminos mínimos calculados.

Toda esta información referente a cuales nodos destino debemos considerar para cada origen, viene dada en la matriz origen-destino. Internamente utilizamos una estructura de punteros en los que a cada origen considerado le corresponde una fila de 'shdist' con sus nodos destino a considerar. Como esta información es leída del fichero Bar-Gera de demandas, nuestro algoritmo de transformación de datos genera estas estructuras, lo cuál acelerará el proceso de CGP al no tener que ir leyendo las demandas en cada iteración.

Procesar caminos: Una vez calculados los caminos mínimos para el origen que estamos procesando, pasamos a analizar cada uno de estos caminos generados.

Obtener id del par: Cada demanda está asociada a un identificador ω, que utilizamos para obtener datos sobre la misma. En este punto y conociendo el nodo origen y el destino, calculamos el identificador del par asociado a dicho camino calculado.

Comprobar si el camino está repetido: Puede ser que el camino mínimo obtenido sea igual al calculado en iteraciones anteriores. Sólo guardamos en el sistema caminos nuevos, por lo que debemos realizar una comprobación de repetición de ese nuevo camino con los demás caminos asociados al par.

5.4. Algoritmo para RMP

En la etapa de equilibrado intentamos mejorar la solución actual del sistema. Para ello nos basamos en la resolución de sistemas lineales que nos proporcionan una solución provisional que nos define una dirección de búsqueda en que mover la solución actual para mejorarla y acercarnos aún más a la solución óptima.

La estrategia utilizada consiste en reducir la dimensionalidad del problema, para ello, diferenciamos entre flujo fijo y flujo variable. Los pares no identificados (1 solo camino) son los que conforman el flujo fijo, el cuál se asigna a la red, y es con los pares no equilibrados, flujo variable, con los que debemos de realizar los sistemas de ecuaciones lineales de modo que vayamos obteniendo la solución esperada. En cada equilibrado se consideran unos caminos en el LS dependiendo de las circunstancias, pasando de un equilibrado a otro a considerarse flujos variables como flujos fijos y viceversa.

Algoritmo para el método de Newton

En la figura 5.4 podemos ver un esquema del algoritmo de equilibrado basado en el método de Newton que hemos desarrollado, pasando a continuación a describir de forma breve las distintas etapas de dicho algoritmo.

Inicializar estructuras: Inicialización de vectores auxiliares considerando sólo los caminos y pares identificados

Mejoramos la solución: Mejoramos la solución actual hasta que se cumpla el criterio de parada.

Calcular U y π: Calculamos el vector U(ω) en el que cada elemento es el coste del camino con menor coste del par, y el vector π(p) en el que cada elemento representa la diferencia del coste del camino con respecto al coste mínimo en el par, reflejado por U.

Criterio de parada: Comprobamos si todos los caminos con flujo van por caminos de coste mínimo ($V_p(p) \cdot \pi_p(p) = 0$). Si se cumple el criterio de parada, salimos porque tenemos solución, actualizando antes la configuración de la red (V_a, C_a y C_p).

Construir el sistema de ecuaciones: Partimos de la matriz A y el vector b calculados en la iteración correspondiente del DSD. Este A y b queda reducido según los caminos considerados ene este equilibrado. Si los caminos considerados en esta iteración son distintos a los de la última, además debemos de actualizar b.

Resolver el LS: Resolvemos el sistema de ecuaciones lineales, obteniendo una solución provisional. Recalculamos C_p, $U(\omega)$ y $\pi(p)$. Tenemos una dirección de búsqueda.

Calcular desplazamiento: Hay que elegir un λ adecuado para cada par, que nos asegure que no nos vamos a salir. Miramos si todos los flujos de los caminos de ese par son >0, si son positivos, $\lambda_\omega = 1$, si no, hay que obtenerlo calculando un mínimo $\lambda_\omega = \text{mín}_{d_p<0} \frac{-\hat{h}_p}{d_p}, \quad \forall p \in \omega$.

Actualizar solución: Nos movemos un poco en la dirección de búsqueda de forma que 'caminosVp = solucionActual + λ * dirección'. Actualizamos la configuración de la red (V_a, C_a y C_p).

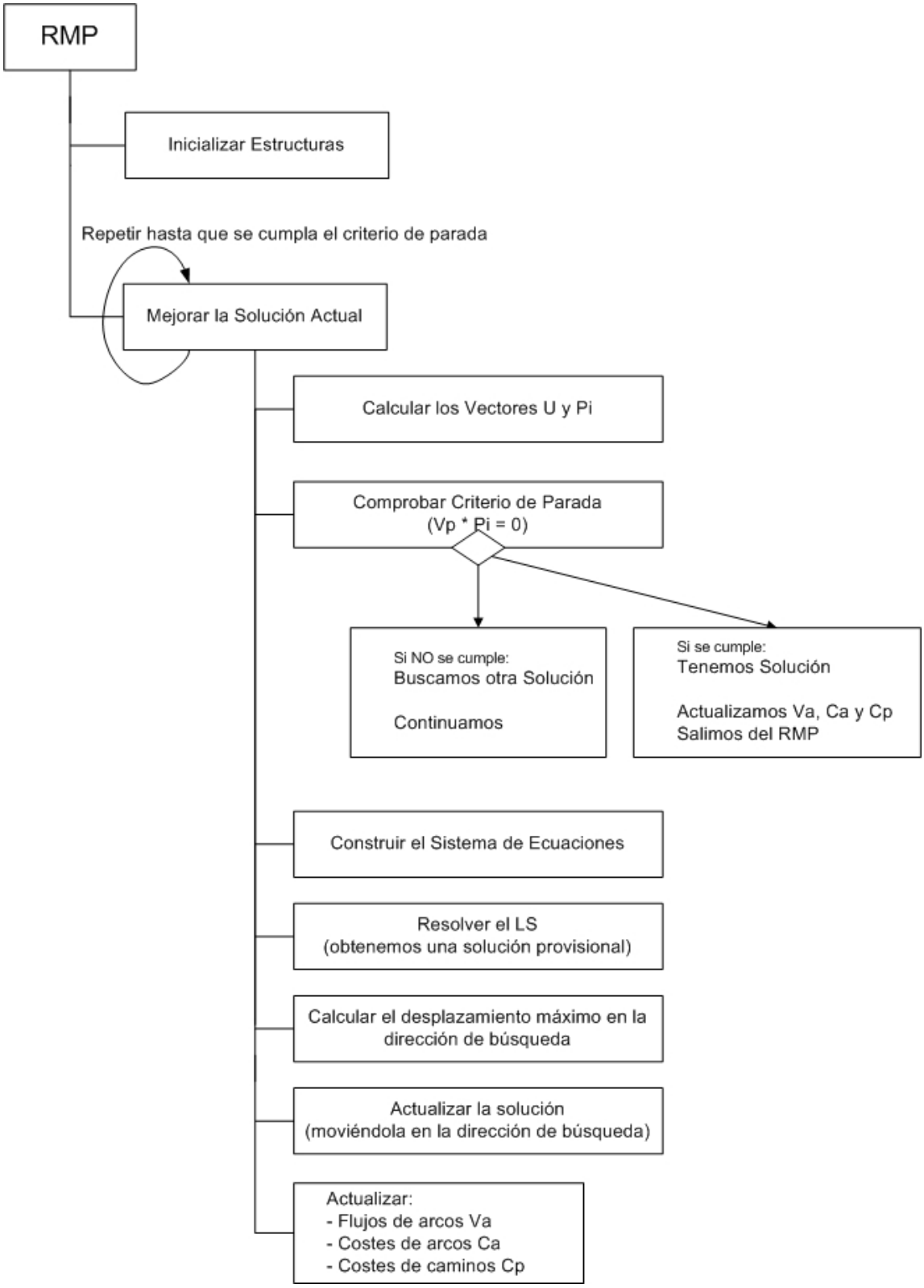

Figura 5.4: Esquema del algoritmo RMP Newton

Capítulo 6

Resultados

Una vez analizado y creado nuestro propio algoritmo de asignación de tráfico DSD con estrategia de identificación de pares, pasamos en este capítulo a su aplicación a problemas reales y a la interpretación de los resultados obtenidos.

6.1. Algoritmo CGP

Como hemos comentado, en la fase de CGP utilizamos un algoritmo de caminos mínimos basado en el L2queue de Gallo y Pallottino, pero optimizado mediante la utilización de unas estructuras de datos propias basadas en las propuestas por Bertsekas.

La prueba de este algoritmo nos ha dado unos resultados excelentes. A continuación podemos ver la tabla 6.1 con las pruebas realizadas y los resultados computacionales obtenidos. En ella también aparecen la dimensionalidad de la red a modo de número de nodos y arcos, y la dimensionalidad de la matriz origen-destino, en la que podemos ver cuantos orígenes y pares hemos tenido que considerar, y por lo tanto, el número de caminos mínimos que ha tenido que calcular el algoritmo en cada caso.

Los tiempos de nuestro CGP han sido obtenidos en MATLAB con un PC portátil (Intel Core Duo T2300 con 1GB de RAM), mientras que los resultados mostrados en la última columna corresponden al DSD sin congestión (1 iteración) obtenidos en GAMS mediante el servicio de Supercomputación de la UCLM.

Tabla 6.1: Resultados computacionales CGP

Red	Nodos	Arcos	Orígenes	Pares	Tiempo CGP sin congestión MATLAB (seg)	Tiempo DSD sin congestión GAMS (seg)
Nguyen Dupuis	13	19	2	4	0.0028	0.171
SiouxFalls	24	76	24	528	0.0130	0.565
Hull	501	798	16	142	0.1245	1.156
Winnipeg	1052	2836	136	4345	1.7000	1544
Barcelona	1020	2522	97	7922	1.4583	18638
Chicago Sketch	933	2949	385	93169	5.4795	

Estos tiempos son los de la fase CGP completa, lo que quiere decir que incluyen la realización de otros procesos como la comprobación de caminos repetidos, el almacenamiento de nuevos caminos en todas las estructuras,... Por lo tanto, la ejecución aislada del

algoritmo de cálculo de caminos mínimos nos proporciona unos resultados aún mejores.

Si comparamos los resultados que hemos obtenido con nuestro CGP utilizando un PC portátil y los obtenidos por el DSD sin congestión en GAMS utilizando el servicio de Supercomputación, podemos ver como la disminución de tiempo de cómputo es bastante considerable. Señalar que en GAMS hacemos un DSD completo, pero sin congestión, lo que supone una única iteración consistente en una etapa de CGP y otra de RMP.

6.2. Algoritmos DSD previos

Como ya comentamos en el capítulo de codificación, en primer lugar, y antes de desarrollar nuestro algoritmo DSD, implementamos y estudiamos un algoritmo DSD básico que posteriormente fuimos optimizando y añadiendo la estrategia de identificación de pares.

El primer algoritmo desarrollado, **DSD clásico** aplicado a problemas simétricos formulados como modelos de optimización, utiliza en la etapa de equilibrado el paquete de optimización de MATLAB, concretamente la minimización con restricciones de una función objetivo 'fmincon'. Para utilizar esta minimización, además de definir la función objetivo, debemos de proporcionar una serie de restricciones: por un lado proporcionamos las condiciones de la satisfacción de la demanda mediante igualdades representadas por una matriz 'Aeq' y un vector 'beq', y por otro lado establecemos la condición de que todos los flujos obtenidos tienen que ser positivos, indicando unos límites superior 'ub = infinito' e inferior 'lb = 0'.

La función 'fmincon' presenta varios modos de utilización, según lo queramos aplicar a problemas de pequeña, media o gran escala. Además permite controlar y optimizar su funcionamiento mediante la configuración de parámetros de precisión, tolerancias o número de iteraciones y mediante la proporción de gradientes y hessianos.

Aplicando el DSD clásico a la red de Nguyen Dupuis, configurando 'fmincon' como problema de media escala, proporcionando el gradiente y hessiano, y realizando un ajuste de parámetros, obtenemos la solución con precisión de 10^-2 en 2.7808 segundos, lo cual es una solución bastante mala.

Comprobamos que este resultado es debido a un cuello de botella presente en la

función 'fmincon'. La única manera de mejorarlo sería configurándolo para la resolución de problemas de gran escala, pero este modo de funcionamiento representa un problema adicional, que puede ser consultado en la documentación del paquete de optimización de MATLAB. Dicho problema consiste en que no se nos permite especificar a la vez los dos tipos de restricciones necesarias: límites inferior y superior y restricciones en forma de igualdades. Si queremos que satisfaga la demanda obtendremos flujos negativos y si indicamos que sean positivos la demanda no será cubierta.

El algoritmo DSD clásico fue mejorado introduciendo la estrategia de **identificación de pares**. En este caso se mejoraron los tiempos de cómputo obtenidos con el algoritmo clásico, pero siendo aún mayores que los obtenidos por otros algoritmos desarrollados en GAMS, que podemos decir que constituye el Estado-del-Arte en optimización. Además seguíamos encontrándonos con el problema del cuello de botella y la utilización de optimizaciones de gran escala.

Tras realizar un estudio sobre algoritmos DSD con identificación de pares desarrollados en **GAMS** podemos decir que GAMS es muy bueno en la resolución de problemas de optimización, y al no presentar este cuello de botella, presenta mejores resultados que los obtenidos hasta el momento. Pero también es cierto que en la fase de CGP y debido a que GAMS presenta bastantes limitaciones como lenguaje de programación y manejo de bucles, condiciones, estructuras,... obtenemos peores resultados que con nuestro algoritmo de cálculo de caminos mínimos (ver tabla 6.1).

6.3. Algoritmo DSD

Este cuello de botella comentado, es solventado por nuestro algoritmo DSD desarrollado, ya que no nos basamos en la resolución de una función de minimización sino en la resolución de varios sistemas de ecuaciones lineales, rápidos de resolver y reducidos de dimensionalidad debido a la aplicación de la estrategia de identificación de pares.

Al aplicarlo a la red de Nguyen-Dupuis, tal y como podemos ver en la tabla 6.2, obtenemos la solución de mayor precisión en un tiempo total de 0.29867 segundos con el método de Newton, siendo los valores de gap obtenidos: Gap = 5.8208e-011 y Gap Relativo = 6.8457e-016. Si en vez de utilizar el método de Newton en el equilibrado, optamos por otros métodos como el de Jacobi o Proyección, los resultados obtenidos son

peores, ya que estos algoritmos además de proporcionarnos una peor precisión, requieren de un mayor número de iteraciones para alcanzar la solución. Recordemos que con el DSD clásico obtuvimos un tiempo de 2.7808 segundos, siendo la solución de menor precisión.

Tabla 6.2: Resultados computacionales para Nguyen-Dupuis

	Tiempo	Gap	Gap Relativo
MATLAB	0.3	5.8208e-11	6.8457e-16
GAMS	0.5	3.1699e-03	3.7281612e-08
DSD clásico	2.8	2.4452e-02	2.8758e-007

El método de Newton nos proporciona la solución exacta, pero podemos ver como en los resultados obtenidos nos da un pequeño error de precisión. Esto es debido a la precisión con la que trabaja MATLAB por defecto.

Al intentar aplicar el método de Newton a otras redes nos encontramos con una problemática, las matrices de restricciones generadas para realizar los sistemas lineales son singulares, su determinante vale 0, lo que hace que no tengan inversa y que al intentar resolver el sistema lineal MATLAB nos avise de que los resultados obtenidos pueden ser imprecisos debidos al redondeo, lo cuál hace que la solución se degenere y nos de resultados erróneos. ¿Por qué no es aplicable?

Los métodos de linealización tienen asegurada su convergencia bajo hipótesis de que la función de coste es fuertemente monótona. En dicho caso el problema de desigualdades variacionales tiene solución única.

Un hecho conocido es que incluso cuando la función de coste en los caminos $C(v)$ es fuertemente monótona entonces no se puede asegurar que la función de coste en los caminos $C(h)$ también lo sea. De hecho, lo usual es que la situación de equilibrio de flujo en los arcos es única pero sin embargo existe infinitos equilibrios de flujo en los caminos. Se sabe que cuando la función $C(h)$ es monótona el conjunto de soluciones de equilibrio es compacto y convexo.

Todo este preámbulo es necesario para explicar el por qué de la no aplicabilidad del método de Newton a redes reales de tráfico. Supóngase que el flujo en equilibrio en los arcos es único, lo denotamos por v^*, entonces consideramos el siguiente sistema de ecuaciones lineales en los flujos en los caminos h:

$$U^* = \Delta h$$

Este sistema tiene $|A|$ ecuaciones y $|K|$ caminos, cumpliéndose que

$$|K| >> |W| > |A|$$

donde $|W|$ es el número de pares origen-destino en la red. Incluso si considerásemos únicamente un conjunto de caminos $\bar{K}$ que satisfaga las restricciones de demanda (esto último lo cumple la estrategia de generación de caminos empleada por el DSD), la anterior desigualdad también se cumple:

$$|\bar{K}| > |W| > |A|$$

Entonces el sistema de ecuaciones lineales tiene más variables que ecuaciones y por tanto debe ser incompatible (no tiene solución) o compatible indeterminado (infinitas soluciones). Como existe al menos un vector de flujo h en equilibrio, tendremos que el sistema presenta infinitas soluciones.

Tendremos que todos los vectores de flujos en los caminos h que satisfagan $U^* = \Delta h$ no varían el flujo en los arcos, por tanto todos los caminos de la red poseen el mismo coste que en la situación de equilibrio. Esto es:

$$C_p(h) = \sum_{a \in A} v_{pa} C_a(\Delta h) = \sum_{a \in A} v_{pa} C_a(v^*) = C_p(h^*)$$

donde h^* es un equilibrio.

Volviendo al método de Newton, éste resuelve la desigualdad variacional asociada a una función de coste afín $C(h) = \Delta h + b$ sobre un subconjunto de caminos $\bar{K}$ que satisfacen las condiciones de demanda. Para ello plantea los problemas $LS(P_A)$:

$$\begin{aligned} &\Delta\bar{h} + b = \Lambda^T u \qquad\qquad \bar{h} \in \bar{\Omega} \\ &\Lambda\bar{h} = g \end{aligned}$$

Estos sistemas lineales tienen $|\bar{K}| + |\bar{W}|$ ecuaciones y $|\bar{K}| + |\bar{W}|$ variables pero no en todos los casos es un sistema compatible determinado.

Supongamos que u^* es el coste de equilibrio en los pares del sistema

$$\Delta\bar{h} + b = \Lambda^T u^*$$

tiene $|W|$ ecuaciones y $|W|$ incógnitas, pero cualquier solución $\bar{h}$ de $u^* = \Delta h$, por lo razonado anteriormente, tiene el mismo coste y por tanto satisface el sistema anterior. Es decir, en esta situación tendremos un sistema compatible indeterminado del que MATLAB no sabe encontrar una solución, mostrando errores de mal escalamiento y teniendo errores de redondeo que hacen inviable el método.

Existen inversas generalizadas que podrían subsanar estos problemas pero no están implementadas en MATLAB. En las iteraciones iniciales se obtiene la situación descrita anteriormente. El número de caminos identificados es muy superior al de arcos. Esta situación cambia cuando el algoritmo progresa.

Capítulo 7

Conclusiones

7.1. Conclusiones

Tras realizar un estudio sobre la mejor forma de representar internamente la configuración de la red y las demandas, optamos de forma acertada por unas **estructuras de datos** basadas en un sistema de vectores y punteros que nos ha permitido un acceso directo a la información y una mayor optimización del algoritmo. No por esto hemos dejado a un lado el **standard Bar-Gera**, creando para ello un programa de conversión de datos de formato Bar-Gera a formato MATLAB con nuestras estructuras de datos propias, en el que aprovechamos para precalcular todo la información posible para librar al algoritmo DSD de alguna carga computacional innecesaria.

Se ha desarrollado y optimizado un **algoritmo de cálculo de caminos mínimos**, base de la etapa CGP, que nos proporciona unos resultados computacionales excelentes. Ha sido un acierto intentar mejorar el algoritmo L2queue de Gallo y Pallotino utilizando una modificación de las estructuras de datos.

La creación de un algoritmo de descomposición simplicial desagregada con identificación de pares ha sido cubierta mediante el desarrollo de un algoritmo **DSD-P** completo en el que utilizamos en la etapa CGP nuestro algoritmo de caminos mínimos y en la etapa RMP el método de Newton el cual nos proporciona una gran precisión y rápida convergencia hacia la solución. Los resultados obtenidos por este algoritmo han sido mejores que los obtenidos mediante implementaciones de DSD clásicos y DSD-P basados en funciones de minimización, e incluso mejores que los obtenidos por algoritmos equivalentes desarrollados en GAMS.

7.2. Futuras mejoras

Algoritmo DSD híbrido

La principal mejora que podemos plantear, y en la que estamos trabajando en la actualidad, es la creación de un algoritmo DSD híbrido que utilice varias fases de equilibrado con métodos distintos.

Hemos visto como en la fase de equilibrado por un lado tenemos el método de Newton el cual converge más rápido hacia la solución y nos ofrece una mayor precisión, pero presenta problemas de aplicabilidad bajo ciertas condiciones. Sin embargo, por otro lado tenemos los métodos de Jacobi y Proyección, menos costosos computacionalmente, y que aunque no son tan precisos y requieren de muchas iteraciones para llegar a la solución, no presentan estos problemas.

Sabemos que en las primeras iteraciones del algoritmo, en el equilibrado consideramos muchos caminos y es posteriormente cuando según avanzamos en iteraciones cuando tenemos identificados los caminos buenos y necesitamos equilibrar menos para ir obteniendo una mayor precisión.

La idea consiste en utilizar ambos métodos de equilibrado. En un primer lugar utilizamos Jacobi o Proyección para acercarnos a la solución aunque sea de una forma imprecisa, pero que nos permite ir identificando y quedarnos con pocos caminos, que son los que tendrán flujo positivo en la situación de equilibrio pero no están muy bien equilibrados. Es en este punto cuando utilizamos el método de Newton sobre ese conjunto de caminos, que está formado sólo por los caminos buenos y ya no presenta el problema de aplicabilidad. Con Newton nos acercamos rápidamente a la solución y ganamos la precisión que no hemos podido alcanzar con Jacobi o Proyección.

El esquema del algoritmo sería el mismo, pero ahora la etapa de RMP que utilizamos puede ser una u otra dependiendo de lo avanzado del estado del DSD. En las primeras iteraciones del DSD utilizamos el equilibrado con Proyección o Jacobi. Cuando detectamos que ya no se añaden nuevos caminos en la fase CGP o se añaden muy pocos, podemos decir que casi tenemos los caminos de la solución aunque con poca precisión. En desde este punto cuando en las siguientes iteraciones del DSD cambiamos el método RMP por el de Newton, alcanzando una mayor precisión en la solución.

Otras mejoras

Podemos realizar también otras mejoras en el rendimiento del algoritmo mediante la optimización del código desarrollado. Entre ellas podemos destacar la siguiente:

Intentar implementar las cuatro matrices delta de forma que las filas sean dinámicas, ya que los limites establecidos, de número de arcos por camino y número de caminos

por par, influyen en el tiempo de cómputo del algoritmo. MATLAB no permite declarar matrices en las que cada fila tenga distinto número de elementos así que hemos intentado realizar la mejora utilizando estructuras, pero las pruebas que hemos realizado nos han arrojado resultados computacionalmente peores ya que el manejo de estructuras de realiza MATLAB es bastante más lento que el realiza con matrices y vectores.

Bibliografía

Problema de asignación de tráfico

[Bar-Gera and Boyce2002] Bar-Gera, H. and D. Boyce (2002). Origin-based network assignment. in *Transportation Planning. State of the Art.* M Patriksson and M. Labbé (Eds.) Kluwer Academic Publishers

[Bertsekas and Gafni1982] Bertsekas, D. P. and E. Gafni: 1982, 'Projections methods for variational inequalities with applications to the traffic assignment problem'. *Mathematical Programming Study* **17**, 139–159.

[Bertsekas1991] Bertsekas, D. P.: 1991, 'Linear network optimization. Alghoritms and codes'. The MIT Press.

[Castillo et al.2002] Castillo, E., A. Conejo, P. Pedregal, R. García, and N. Alguacil: 2002, *Building and solving mathematical programming models in engineering and science.* New York: Pure and Applied Series (PAMS), John Wiley & Sons, Inc.

[Gallo and Pallotino1989] G.Gallo, S.Pallotino, et al: 14 Abril 1989, 'FORTRAN codes for network optimization'. *European Journal of Operational Research* Volume 39, Issue 3, Pages 348-350.

[Dafermos1980] Dafermos, S. C.: 1980, 'The traffic equilibrium and variational inequalities'. *Transportation Science* **14**, 42–54.

[García and Marín 2005] García, R. and A. Marín: 2005,'Network Equilibrium Models with Combined Modes: models a and solution algorithms'. *Transportation Research-B* **39(3)**, 223-254.

[García et al.2003] García, R., A. Marín, and M. Patriksson: 2003, 'Column generation algorithms for nonlinear optimization, I: Convergence analysis'. *Optimization* **52**, 171–200.

[García2001] García, R. (2001). *Metodología para el diseño de redes de transporte y para la elaboración de algoritmos en programación matemática convexa diferenciable.* Ph. D. thesis, Escuela Técnica Superior de Ingenieros Aeronaúticos, Universidad Politécnica de Madrid.

[Giannessi et al.2002] Giannessi, F., A. Maugeri, and P. M. Pardalos (2002). *Equilibrium problems: nonsmooth optimization and variational inequality models.* Dordrecht: Kluwer Academic Publishers.

[Hearn1982] Hearn, D. W.: 1982, 'The gap function of a convex program'. *Opt. Res. Lett.* **1**, 67–71.

[Hearn et al.1985] Hearn, D. W., S. Lawphongpanich, and J. A. Ventura (1985). Finiteness in restricted simplicial decomposition. *Operations Research Letters 4*, 125–130.

[Hearn et al.1987] Hearn, D. W., S. Lawphongpanich, and J. A. Ventura: 1987, 'Restricted simplicial decomposition: Computation and extensions'. *Mathematical Programming Study* **31**, 99–118.

[Hohenbalken1977] Hohenbalken, B. v. (1977). Simplicial decomposition in nonlinear programming algorithms. *Mathematical Programming13*, 49–68.

[Holloway1974] Holloway, C. A. (1974). An extension of the Frank and Wolfe method of feasible directions. *Mathematical Programming6*, 14–27.

[Larsson and Patriksson1992] Larsson, T. and M. Patriksson: 1992, 'Simplicial decomposition with disaggregated representation for the traffic assignment problem'. *Transportation Science* **26**, 4–17.

[Larsson et al.1997] Larsson, T., M. Patriksson, and C. Rydergren: 1997, 'Applications of simplicial decomposition with nonlinear column generation to nonlinear network flows'. In: P. M. Pardalos, W. W. Hager, and D. W. Hearn (eds.): *Network Optimization*, No. 450 in Lecture Notes in Economics and Mathematical Systems. Berlin: Springer-Verlag, pp. 346–373.

[Lawphongpanich and Hearn1984] Lawphongpanich, S. and D. W. Hearn: 1984, 'Simplicial decomposition of the asymmetric traffic problem'. *Transportation Research* **18B**, 123–133.

[Marcotte and Guélat1988] Marcotte, P. and J. Guélat: 1988, 'Adaptation of a modified Newton method for solving the asymmetric traffic equilibrium problem'. *Transportation Science* **22**, 112–124.

[Marín1995] Marín, A. (1995). Restricted simplicial decomposition with side constraints. *Networks26*, 199–215.

[Montero1992] Montero, L.: 1992, 'A simplicial decomposition approach for solving the variational inequality formulation of the general traffic assignment problem for large scale networks'. Ph.D. thesis, Universidad Politécnica de Cataluña, Barcelona, Spain.

[Montero and Barceló1996] Montero, L. and J. Barceló: 1996, 'A simplicial decomposition algorithm for solving the variational inequality formulation of the general traffic assignment problem'. *TOP, Journey of the Spanish Statistical and Operation Research Society* **4**, 225–256.

[Nguyen and Dupuis1984] Nguyen, S. and C. Dupuis: 1984, 'An efficient method for computing traffic equilibria in networks with asymmetric transportation costs'. *Transportation Science* **18**, 185–202.

[Pang and Chan1982] Pang, J. S. and D. Chan: 1982, 'Iterative methods for variational and complementary problems'. *Mathematical Programming* **24**, 284–313.

[Pang and Yu1984] Pang, J. S. and C. S. Yu: 1984, 'Linearized simplicial decomposition methods for computing traffic equilibria on networks'. *Networks* **14**, 427–432.

[Patriksson1994] Patriksson, M. (1994). *Traffic Assignment Problem. Models and Methods.* Utrecht, The Netherlands: VSP.

[Patriksson1999] Patriksson, M.: 1999, *Nonlinear programming and variational inequality problems. A unified approach.* Dordrecht: Kluwer Academic Publishers.

[Smith1979] Smith, M. J.: 1979, 'The existence, uniqueness and stability of traffic equilibria'. *Transportation Research B* **13**, 295–304.

[Wardrop1952] Wardrop, J. G.: 1952, 'Some Theoretical Aspects of Road Traffic Research'. In: *Proceedings of the Institute of Civil Engineers Part II.* pp. 325–378.

Matlab

[MathWorks Inc 2004] Matlab Optimization Toolbox User´s guide. Version 3. MathWorks Inc, 2004.

[MathWorks Inc 2004] Matlab Genetic Algorithm and Direct Search Toolbox. User´s guide Version 1. MathWorks Inc, 2004.

[MathWorks Inc 2004] Using Matlab Graphics. Version 7. MathWorks Inc, 2004.

[MathWorks Inc 2004] Matlab Programming. Version 7. MathWorks Inc, 2004.

[MathWorks Inc 2004] Matlab Mathematics. Version 7. MathWorks Inc, 2004.

[MathWorks Inc 2004] Matlab Desktop Tools and Development Environment. Version 7. MathWorks Inc, 2004.

[MathWorks Inc 2004] Getting Started with Matlab. Version 7. MathWorks Inc, 2004.

Apéndice 1 - Manual de código