Títol: RESOLUCIÓ D'UN MODEL D'EQUILIBRI AMB IDENTIFICACIÓ PRÈVIA DE PASSOS

Volum: 1 de 1

Alumne: MASSANET VILA, RAIMON JAUME

Director/Ponent: CODINA SANCHO, ESTEVE

Departament: EIO

Data: 27-06-2008

DADES DEL PROJECTE

Títol del Projecte: RESOLUCIÓ D'UN MODEL D'EQUILIBRI AMB IDENTIFICACIÓ PRÈVIA DE PASSOS

Nom de l'estudiant: MASSANET VILA, RAIMON JAUME
Titulació: ENGINYERIA EN INFORMÀTICA
Crèdits: 37,5
Director/Ponent: CODINA SANCHO, ESTEVE
Departament: EIO

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President: FERNÁNDEZ AREIZAGA, ELENA

Vocal: TIÑENA SALVAÑÀ, FRANCESC

Secretari: CODINA SANCHO, ESTEVE

QUALIFICACIÓ

Qualificació numèrica:
Qualificació descriptiva:

Data:

Als meus pares,

Índex

1 Introducció **7**

1.1 Programació matemàtica 7

1.2 Xarxes de transport urbanes 9

1.3 Planificació sistèmica 13

1.4 Resum 14

2 Consideracions de disseny **17**

3 Identificació prèvia de passos **21**

3.1 Resolució 22

4 Equilibri en xarxes de transport **27**

5 Resolució de l'equilibri d'usuaris **35**

5.1 Variables del problema 38

5.2 Funció objectiu 38

5.3 Base 40

5.4 Costs reduïts 40

5.5 Direcció de descens 41

5.6 Exploració lineal 42

5.7 Canvi de base 45

5.8 RGAP 45

6 Resultats **51**

7 Implementacions alternatives **63**

8 Cost del projecte **69**

A Glossari **73**

B Manual d'usuari de l'aplicació **79**

Prefaci

Aquest projecte va néixer per iniciativa del professor Esteve Codina Sancho. Aleshores ell era el meu professor de l'assignatura *Models d'Investigació Operativa per a la Presa de Decisions (M.I.O.P.D)* pertanyent a la carrera d'*Enginyeria Informàtica*. Un dia a classe va preguntar si hi havia algun alumne interessat en realitzar un Projecte Final de Carrera en la matèria que s'impartia a classe. A mi em va atraure l'idea i després de parlar-ne vam formalitzar els tràmits d'inscripció del projecte.

Si he ser ser sincer la *programació no lineal*, dins la qual s'emmarca aquest projecte, no era el meu punt fort. Així que la realització d'aquest projecte, més enllà de ser un requisit acadèmic per obtenir el títol d'Enginyer en Informàtica, m'ha donat la possibilitat de reforçar els meus coneixements en una àrea que necessitava ser reforçada.

Vull aprofitar aquest prefaci per agrair a les persones que m'han donat suport al llarg de la elaboració d'aquest projecte, en especial als meus pares, sense l'esforç dels quals segurament no hauria pogut accedir a la Universitat Politècnica de Catalunya (U.P.C) i no seria qui soc avui dia, al meu germà, perquè és algú a qui li puc demanar qualsevol tipus de dubte i sempre té una resposta, a la meva parella, que m'ha ajudat molts cops a no deixar de banda aquest projecte per culpa del treball, al professor Esteve Codina, per introduir-me en un món gairebé desconegut per mi i dirigir-me cap al bon

camí i molts altres que d'una forma o una altra heu contribuït amb el vostre granet de sorra en què aquest projecte veies, finalment, la llum.

Capítol 1

Introducció

1.1 Programació matemàtica

L'optimització és tan antiga com la Humanitat. Forma part de la natura humana, i de la Natura en general, intentar aconseguir el major guany amb el menor esforç.

Un exemple clar és l'economia, ciència optimitzadora per excel·lència, que intenta obtenir els màxims beneficis minimitzant els costs i treure així el màxim partit a uns recursos escassos.

Durant molts segles aquests tipus de problemes foren resolts a mà i de forma gairebé intuïtiva. Això és possible per a problemes senzills, però quan els problemes es compliquen no és fàcil dir intuïtivament quina és la millor solució. Calen eines i models matemàtics que ofereixin la solució òptima a un problema donat.

Tot i que Gauss ja proposà alguna tècnica d'optimització matemàtica a finals del segle XVIII i principis del XIX, la resposta a aquest problema sorgí durant la Segona Guerra Mundial, de mans de George Dantzig. En meitat de la guerra algú va pensar que seria útil disposar d'una eina que permetés

avaluar quines accions perjudicaven més a l'enemic i a l'hora beneficiaven més als aliats. Sorgiren així la programació lineal i l'algorisme que resol aquest tipus de problemes: l'*algorisme del símplex*.

L'únic que es necessita per trobar la millor solució a un problema és ser capaç de modelar-lo en termes de programació matemàtica. És a dir, cal definir quines són les variables i la funció que es vol optimitzar, així com les condicions a les que està sotmès el problema.

A la *Formulació 1* es mostra un problema de programació matemàtica típic:

$$\begin{aligned} \max(\min)_{x_1,x_2,\ldots,x_n} \quad & f_0(x_1, x_2, \ldots, x_n) \\ \text{Subjecte a:} & \\ & f_1(x_1, x_2, \ldots, x_n) \leq b_1 \\ & f_2(x_1, x_2, \ldots, x_n) = b_2 \\ & \vdots \\ & f_m(x_1, x_2, \ldots, x_n) \geq b_m \end{aligned}$$

Formulació 1: Exemple de problema d'optimització.

On $x_1, x_2, \ldots, x_n$ són les *variables del problema*, f_0 és la funció que és vol optimitzar i $f_1, f_2, \ldots, f_m$, són funcions que defineixen una sèrie de condicions que ha de complir la solució. La funció f_0 s'anomena *funció objectiu* i les condicions $f_1, f_2, \ldots, f_m$ es solen anomenar *constriccions*.

La programació matemàtica és la branca de la matemàtica que intenta resoldre problemes d'optimització d'aquest tipus.

En funció de la complexitat de la funció objectiu i de les constriccions dels problemes que intenta resoldre, la programació matemàtica es pot dividir en varies categories. Aquest és un llistat no exhaustiu i en ordre ascendent de complexitat d'algunes de les categories més conegudes:

Programació lineal. Resol problemes en els que tant la funció objectiu com totes les constriccions són de tipus lineal.

Programació entera. Adreça problemes en els que les variables només poden prendre valors enters. Un cas particular de programació entera és la programació binaria, que intenta resoldre problemes en els que les variables només poden prendre dos valors {0,1}.

Programació quadràtica. S'encarrega de la resolució de problemes en els que la funció objectiu és quadràtica i les constriccions són lineals.

Programació no lineal. Resol problemes en els que la funció objectiu i/o alguna de les constriccions són de tipus no lineal.

Tots aquests tipus de models matemàtics tenen infinitat d'aplicacions. Una de les més conegudes, i dins la qual s'emmarca aquest projecte, és l'aplicació de models de programació no lineal en optimització de *fluxos en xarxes* de transport urbanes.

1.2 Xarxes de transport urbanes

Una xarxa de transport urbana és un sistema molt complex. L'estat d'un punt de la xarxa en un moment donat depèn de les decisions preses per molts usuaris. Cada usuari decideix quina ruta agafar per anar des d'on es troba fins a la seva destinació. Aquesta decisió la pren tenint en compte el temps estimat que trigarà per cada ruta. El temps que trigarà per cada ruta està determinat pel volum d'usuaris que hi ha en la ruta, el qual, un altre cop, és funció de les decisions preses.

Des del punt de vista dels usuaris cada ruta té un temps de viatge associat i la preferència dels usuaris per la ruta és una funció decreixent respecte del

temps que es triga en recórrer la ruta. Des del punt de vista de les rutes, en canvi, el temps de viatge és una funció creixent respecte del nombre d'usuaris que la trien.

Aquest comportament és anàleg al comportament econòmic de l'oferta i la demanda d'un producte. La corba de quantitat de productes oferts creix a mesura que puja el preu, ja que als productors els resulta més rentable vendre a preus més elevats. La corba de productes demandats, en canvi, creix a mesura que baixa el preu ja que als consumidors els resulta més rentable comprar a preus baixos. El sistema s'estabilitza en el punt en el que el preu és tal que la corba de l'oferta i la demanda coincideixen. En aquest punt la totalitat del producte produït és consumit. En la *Figura 1.1* es pot apreciar de forma gràfica l'equilibri entre oferta i demanda.

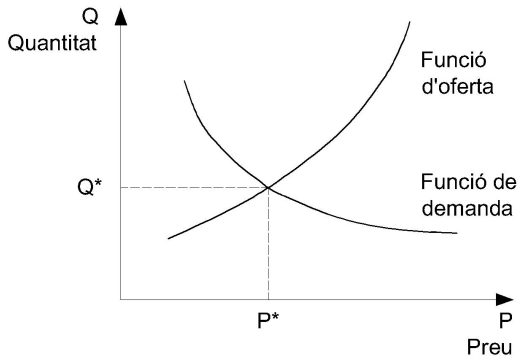

Figura 1.1: Descripció gràfica de les corbes d'oferta i demanda i del punt d'equilibri en un sistema econòmic.

Les xarxes de transport es comporten de forma similar, en espais diferents. Les corbes econòmiques estan definides en l'espai preu/quantitat, mentre que les corbes de transport es defineixen en l'espai flux/temps de viatge. En el

model econòmic es feien servir la corba d'oferta i la corba de demanda per trobar el punt d'equilibri. En el model de transport, en canvi, s'utilitzen la corba de rendiment i la corba de demanda de servei. La demanda de servei es refereix als usuaris que triaran aquella ruta per fer el seu viatge. El rendiment il·lustra el fet de què el temps de viatge en una ruta creix en funció del flux que suporta la ruta.

Igual que en el model econòmic s'arribarà a un punt d'equilibri quan les corbes de rendiment i de demanda coincideixin. Aquesta situació s'il·lustra a la *Figura 1.2*. Sempre que s'estigui en una situació que no sigui aquesta, les forces del sistema el reconduiran cap a l'equilibri. Si la ruta es troba en un punt en el que el flux x_1 és menor que el flux en l'equilibri x^* llavors el temps de viatge en aquella ruta serà molt baix i per tant la ruta resultarà atractiva a més viatgers, que en triar-la faran pujar el flux. En canvi, si el flux es troba en un flux x_2 major que x^* llavors el temps de viatge serà elevat i la ruta resultarà menys atractiva. Així doncs el flux baixarà cap a l'equilibri.

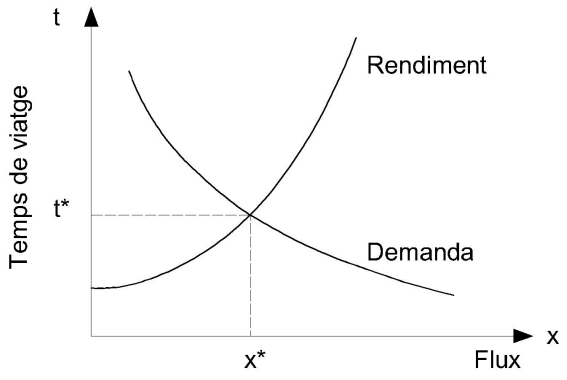

Figura 1.2: Descripció gràfica de les corbes de rendiment i demanda i del punt d'equilibri en una xarxa de transport urbana.

Per modelar el comportament en una xarxa de transport urbana s'utilitzen grafs dirigits, veure *Apèndix A*. Bàsicament, un graf dirigit és una estructura que consta de punts, anomenats nodes, connectats mitjançant arestes. Associat a cada aresta hi ha un valor que representa el cost de travessar aquella aresta. En el context de les xarxes de transport cada punt representa una intersecció entre vies de transport i cada aresta representa el fragment de via que discorre entre dues cruïlles. El cost associat a cada aresta representa el cost temporal de recórrer-la, és a dir, el temps de viatge entre els dos nodes connectats per l'aresta. Les xarxes es representen mitjançant arestes dirigides perquè el temps que es triga en anar des del node i al node j no té perquè ser el mateix que el que es triga per anar des del node j al node i.

Una xarxa de fluxos és un graf dirigit, les arestes del qual tenen assignat una certa quantitat de flux. Cada aresta té una capacitat màxima de flux que pot circular-hi. En la *Figura 1.3* es mostra una xarxa de fluxos d'exemple. Cada aresta té un flux assignat i una capacitat màxima — el primer i segon número entre parèntesi, respectivament.

Per caracteritzar una xarxa de transport urbana també cal definir una matriu de demanda entre tot possible punt de partida i tot possible punt d'arribada. Òbviament, això no es pot fer per cada punt exacte d'una ciutat. El que es fa es definir una sèrie de zones urbanes que es tracten com una unitat. Al centre geogràfic de cada zona urbana s'hi col·loca un node especial anomenat centroide. El centroide d'una zona urbana agrega la demanda de tota la zona i està connectat a tots els punts que envolten la zona urbana. Aquest procediment simplifica molt els càlculs, però també empobreix la qualitat de les prediccions. La mida de les zones urbanes considerades determina la granularitat de les solucions obtingudes.

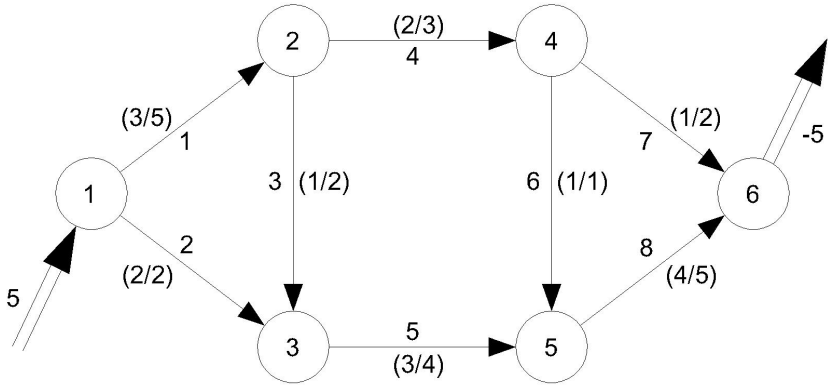

Figura 1.3: Exemple d'una xarxa de fluxos.

1.3 Planificació sistèmica

Els planificadors urbanístics necessiten predir el comportament d'una xarxa en unes circumstàncies donades. Per exemple, en avaluar l'impacte que una obra pública tindrà en l'estat del trànsit, o en planificar els temps en verd dels semàfors de la xarxa. També és molt útil poder comparar dos models urbanístics i determinar quin és el flux de trànsit en cada un per tal de poder optimitzar paràmetres derivats d'aquest flux, com ara el temps mitjà de viatge, o la contaminació acústica en certes zones sensibles.

Aquesta planificació no es pot fer en punts aïllats de la xarxa. El comportament descrit en la *Secció 1.2* és aplicable a cada aresta, però es veu influenciat pel comportament de totes les arestes de la xarxa. Tradicionalment s'intentava predir l'impacte d'una certa actuació urbanística en un punt determinat examinant només la zona concreta. S'ha demostrat, però, que les

actuacions en un punt de la xarxa poden tenir efectes col·laterals en punts allunyats de la xarxa [5].

Per il·lustrar aquesta afirmació consideri's la xarxa urbana descrita a la *Figura 1.4*. Les zones B i D representen zones amb molta activitat, mentre que les zones A i C representen zones relativament tranquil·les. Les arestes 1 i 2 són les úniques vies de comunicació entre els dos extrems de la ciutat. El volum que circula per l'aresta 2 és més gran que el volum que circula per l'aresta 1, ja que les zones B i D són més actives. Per avaluar els efectes que tindria en la xarxa urbana la construcció d'un centre comercial a D no es pot considerar només la xarxa urbana de D. Un nou centre comercial a D generaria més demanda de serveis des de totes les àrees cap a D, en especial des de B, ja que hi viu més gent. En conseqüència l'aresta 2 es saturaria i com a ruta alternativa molts usuaris de B triarien l'aresta 1 per creuar d'un extrem a l'altre. La conseqüència seria que el trànsit local a les àrees A i C es veuria seriosament afectat per usuaris que anirien des de B a D passant per A i C. L'estat a cada part de la xarxa ha de ser estudiat en relació a la totalitat del sistema.

1.4 Resum

En aquest projecte s'ha construït una aplicació que dóna una bona predicció de la distribució de fluxos en les arestes d'una xarxa de transport sota unes condicions determinades. Per tal de millorar l'eficiència de l'algorisme que calcula l'equilibri dels usuaris en la xarxa es fa una identificació prèvia de les possibles rutes entre cada parell origen-destí. Així el problema es resol en dues fases:

1. Identificació prèvia de passos per trobar un nombre determinat de rutes

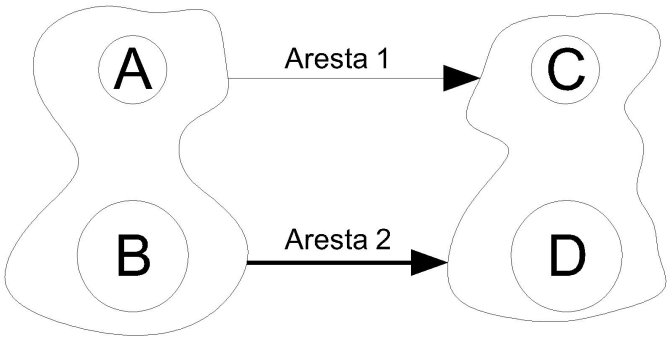

Figura 1.4: La construcció d'un centre comercial a la zona D tindria efectes col·laterals a les zones A i C.

plausibles entre tot parell de nodes origen-destí. Aquest procés es descriu en el *Capítol 3*.

2. Resolució del problema d'equilibri en la xarxa. Aquesta resolució es descriu en el *Capítol 4*.

Capítol 2

Consideracions de disseny

El requisit no funcional fonamental d'aquest projecte és l'eficiència. La mida de les xarxes amb les que aquesta aplicació treballa causa que una petita penalització en l'eficiència de l'algorisme es tradueixi en una penalització temporal gran. Per això, durant el desenvolupament de l'aplicació s'ha prioritzat que el codi fos eficient i s'han deixat una mica de banda altres aspectes com la modularitat, l'entenibilitat, etc.

Un altre objectiu de disseny de l'aplicació és que sigui portable. D'aquesta manera es pot executar el programa en qualsevol màquina sense necessitat d'haver d'escriure diferents versions.

Aquestes consideracions determinen que els llenguatges de programació utilitzats siguin Java i R. Java és un llenguatge eficient i totalment portable. A més, és una llenguatge orientat a objectes, la qual cosa facilita molt la tasca de programació de grafs, llistes i altres estructures de dades. R [4] és un *software* matemàtic de codi lliure. És molt eficient, tot i utilitzar un llenguatge interpretat, a l'hora de realitzar càlculs amb matrius i vectors. També és molt útil a l'hora de treballar amb funcions i gradients, ja que ofereix una eina per calcular analíticament el gradient d'una funció donada

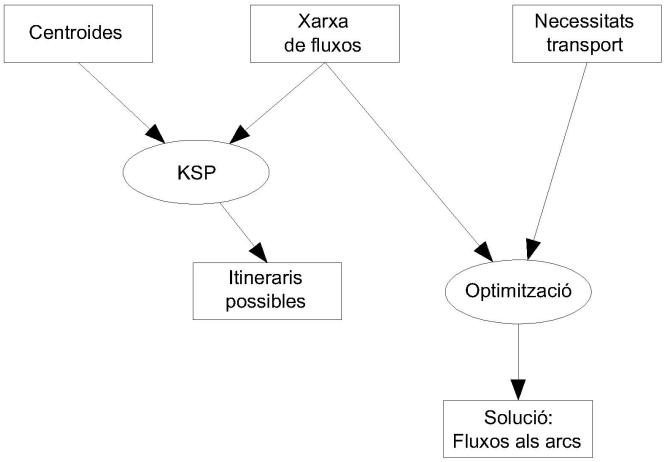

Figura 2.1: El diagrama de fluxos corresponent a l'aplicació desenvolupada.

en un punt determinat. Com que el llenguatge és interpretat, la portabilitat de l'aplicació només requereix tenir instal·lat la versió de R corresponent a la màquina a on es vol executar l'aplicació.

En la fase de disseny s'ha decidit dividir el sistema *software* en dos mòduls que resolen problemes independents:

Identificació de passos. El problema consisteix en trobar els k camins més curts entre tot parell de centroides.

Resolució de l'equilibri d'usuaris. Consisteix en resoldre el problema de programació no lineal que es mostra a la *Formulació 2*.

La raó d'aquesta divisió és que el problema de la identificació prèvia de passos es pot programar més fàcilment utilitzant un llenguatge orientat a objectes, com Java, mentre que la resolució de l'equilibri d'usuaris es pot resoldre de forma eficient utilitzant un llenguatge matemàtic com R. A la *Figura 2.1* es mostra un esquema del funcionament de l'aplicació.

El mòdul encarregat de la identificació prèvia de passos és, a més, el que s'encarrega de llegir les dades d'entrada en el format definit pel client. En obtenir els resultats els escriu en format *CSV* per tal de què el mòdul de resolució de l'equilibri d'usuaris els pugui llegir fàcilment.

Capítol 3

Identificació prèvia de passos

En aquest projecte s'ha fet una simplificació important del problema tot reduint el número de possibles rutes que es consideren entre tot parell origen-destí. Considerar totes les possibles rutes és molt costós en termes de temps de computació i poc eficient, ja que es consideren rutes que en cap cas entren en els plans dels usuaris. Al llarg de tot el projecte, veure *Capítol 4*, es fa la suposició plausible de què els usuaris sempre trien la ruta més curta disponible. Si es consideren les k rutes més curtes a priori com les rutes amb menor temps de viatge en solitari, llavors la més curta serà, en un moment donat, alguna d'aquestes k rutes.

A mesura que creix la demanda total de servei en la xarxa es van considerant cada cop més rutes alternatives, ja que les rutes principals es van saturant i el seu temps de viatge creix. Això fa que el paràmetre k de rutes que es consideren entre cada parell origen-destí s'hagi de triar en funció de la demanda agregada de servei. En aquest projecte no s'analitza com triar el paràmetre k sinó que aquest és introduït per l'usuari.

3.1 Resolució

L'impacte del temps que es triga en trobar les rutes més curtes entre tot parell de centroides origen-destí és molt significatiu respecte del temps que es triga en trobar el punt d'equilibri en la xarxa. Per tant, aquest punt s'ha de resoldre d'una forma eficient. El problema de trobar els k camins més curts entre un parell de nodes en un graf es coneix en el món de l'algorísmica amb el nom de *k shortest paths problem* o KSP. Per resoldre'l s'ha utilitzat l'algorisme de Lawler [3]. Aquest algorisme retorna, donat un parell de nodes origen-destí, una llista amb els k camins més curts, de forma eficient.

L'esquema general de l'algorisme és: considerar el conjunt de tots els camins entre un parell origen-destí donat i determinar el camí més curt d'aquest conjunt. Aquest és el camí més curt ($k = 1$). Dividir els camins restants en subconjunts mútuament exclusius i determinar el camí més curt de cada subconjunt. Com que els subconjunts cobreixen la totalitat dels camins restants, trobar el segon camí més curt consisteix en trobar el camí més curt d'entre els camins més curts de cada subconjunt. Aquest procés es repeteix fins que s'han trobat els k camins més curts. A *Algorisme 1* es pot consultar una formalització en pseudocodi de l'algorisme.

Un aspecte important d'aquest algorisme és la partició dels conjunts en subconjunts, que s'ha de fer seguint la següent definició:

Definició 3.1.1. Sigui S el conjunt de camins acíclics entre el node origen o i el node destí d tals que tots comencen amb la seqüència de G arestes anomenada $L^{inicials} = \{b_1, b_2, \ldots, b_G\}$, i tots exclouen les H arestes contingudes a $L^{excloses} = \{c_1, c_2, \ldots, c_H\}$, i sigui $P(S)$ el camí més curt a S, format per les arestes $\{\{L^{inicials}\}, a_1, a_2, \ldots, a_q\}$. L'algorisme de Lawler divideix el conjunt $S - P(S)$ en q subconjunts $S_1, S_2, \ldots, S_q$ on S_i és el con-

Algorisme 1 Formalització de l'algorisme de Lawler.

Require: o,d: nodes (origen i destí), $k > 0$, $k \in \mathbb{N}$

1: $S_{0,1} \leftarrow$ tots els camins de o a d
2: $P(S_{0,1}) \leftarrow$ camí més curt de $S_{0,1}$
3: $< S_{1,1}, S_{1,2}, \ldots, S_{1,q(1)} >$ $\leftarrow$ dividir $S_{0,1} - P(S_{0,1})$ en $q(1)$ subconjunts mútuament exclusius
4: $m \leftarrow 1$
5: **repeat**
6: $\quad < P(S_{m,1}), P(S_{m,2}), \ldots, P(S_{m,q(m)}) >$ $\leftarrow$ calcular els camins més curts de $S_{m,1}, S_{m,2}, \ldots, S_{m,q(m)}$
7: $\quad (m+1)$-èssim camí més curt $\leftarrow$ camí més curt d'entre

$$\{P(S_{1,1}), \ldots, P(S_{1,q(1)}), P(S_{2,1}), \ldots, P(S_{2,q(2)}), \ldots \\ \ldots, P(S_{m,1}), \ldots, P(S_{m,q(m)})\}$$

Require: Sigui $S_{a,j}$ el conjunt que conté l'$(m+1)$-èssim camí més curt.
8: $\quad < S_{m+1,1}, S_{m+1,2}, \ldots, S_{m+1,q(m+1)} >$ $\leftarrow$ dividir $S_{a,j}$ en $q(m+1)$ conjunts disjunts
9: $\quad m \leftarrow m+1$
10: **until** $m = k$

junt de camins acíclics entre o i d que comencen amb la seqüència d'arestes $L_i^{inicials} = \{\{L^{inicials}\}, a_1, a_2, \ldots, a_{i-1}\}$ i exclouen les arestes contingudes a $L_i^{excloses} = \{\{L^{excloses}\}, a_i\}$.

Trobar el camí més curt dins de cada subconjunt S_i és fàcil. Només cal trobar el camí més curt entre el node destí de l'aresta a_{i-1} i el node destí d, tot excloent les arestes contingudes a $L_i^{excloses}$ i les arestes contingudes a $\{\{L^{inicials}\}, a_1, a_2, \ldots, a_{i-2}\}$. Per fer-ho s'esborren temporalment aquestes arestes del graf i mitjançant l'algorisme A^* [1] trobem el camí més curt. A continuació es tornen a afegir les arestes al graf.

Aquest algorisme és més eficient que la majoria dels seus competidors ja que el número de camins mínims calculats sempre és el doble del número de camins mínims desitjats. La seva complexitat és $O(k \cdot n \cdot c(spt))$, on n és el número de nodes de la xarxa i $c(spt)$ és la complexitat de l'algorisme utilitzat

per calcular el camí més curt entre dos nodes.

Com s'ha dit abans, per calcular el camí més curt entre dos nodes s'utilitza l'algorisme A*, que realitza una cerca *best-first* des del node origen o fins que troba el node destí d. A *Algorisme 2* es pot veure una formalització en pseudocodi de l'algorisme. q és una cua ordenada de camins. L'ordre és creixent respecte del cost total de cada camí.

Algorisme 2 Formalització de l'algorisme A*

Require: o, d són els nodes origen i destí, respectivament
1: $tancats \leftarrow \emptyset$
2: $q \leftarrow$ nova-cua(nou-cami(o))
3: **repeat**
4: $p \leftarrow$ extreure-primer(q)
5: $x \leftarrow$ darrer-node(p)
6: **if** $x \in tancats$ **then**
7: continuar
8: **end if**
9: **if** $x = d$ **then**
10: **return** p
11: **end if**
12: afegir(x,$tancats$)
13: **for all** $y \in$ successors(x) **do**
14: encuar(afegir(y,p),q)
15: **end for**
16: **until** empty(q)
17: **return** error

La complexitat de l'algorisme A* és polinòmica respecte de la longitud del camí més curt quan l'heurístic utilitzat per ordenar els possibles camins compleix la següent condició:

$$|h(x) - h^*(x)| \leq O(\log h^*(x))$$

on h^* és l'heurístic *perfecte*, és a dir, el que retorna la distància exacta al node destí.

En el cas en què no es té cap tipus d'informació sobre la distància que hi ha fins al node destí, la complexitat és exponencial en cas pitjor.

Per tant, la complexitat de trobar els k camins més curts entre un parell de nodes és $O(k \cdot n^n)$. Si N és el número de centroides definits a la xarxa, llavors la complexitat global del mòdul és $O(N^2 \cdot k \cdot n^n)$.

Aquest mòdul està programat en Java. S'ha creat una estructura de dades pròpia per treballar amb grafs. La forma més fàcil d'implementar els grafs és mitjançant una matriu d'adjacències. Però aquest enfoc pot malgastar molta memòria si els grafs són grans i poc densos, com és el cas d'una xarxa de transport, veure *Capítol 1.* Per tant, s'utilitza una estructura de llista dinàmica de nodes. Cada node conté un apuntador a l'inici d'una altra llista dinàmica que conté les arestes que surten del node. A la *Figura 3.1* es mostra un esquema de l'estructura de dades descrita.

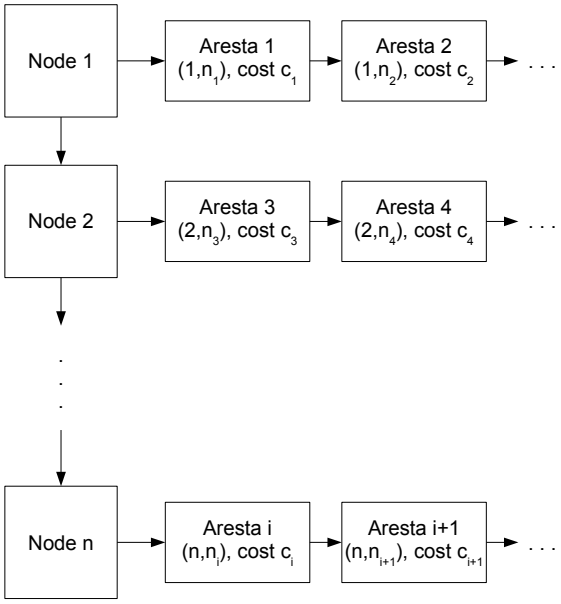

Figura 3.1: Esquema de l'estructura de dades *Graph*.

Capítol 4

Equilibri en xarxes de transport

En aquest capítol s'explica com es troba el punt d'equilibri en una xarxa de transport donada.

Com ja s'ha introduït al *Capítol 1* una xarxa de transport tendeix cap al punt d'equilibri com a resultat de la interacció entre les decisions preses pels usuaris i la densitat de trànsit en les vies.

El problema de trobar el punt d'equilibri es redueix a:

Donades una xarxa de transport, les funcions de cost de cada aresta, i una matriu de demandes per cada parell origen-destí, trobar l'assignació de flux a cada aresta.

En el punt d'equilibri aquesta assignació ha de satisfer la condició d'*equilibri d'usuaris*:

Cap usuari pot millorar el seu temps de viatge canviant de ruta.

Com que els temps de viatge de cada aresta depenen del flux que suporta l'aresta, es d'esperar que els usuaris sempre aniran per rutes que tinguin costs iguals. Suposem un parell origen-destí qualsevol i dues rutes alternatives amb les funcions de temps de viatge mostrades a la *Figura 4.1.* A la gràfica (b) es pot veure que qualsevol càrrega de la ruta 1 menor que el valor q' comportarà

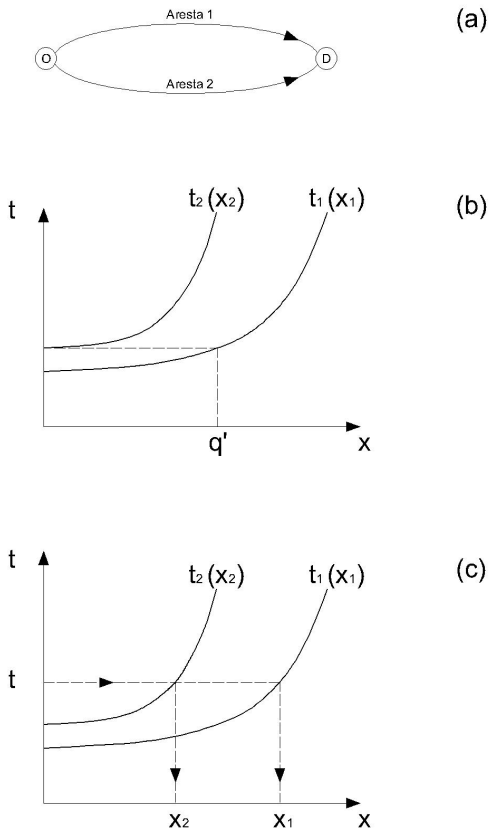

Figura 4.1: Les funcions de temps de viatge de dues rutes entre un origen i un destí d'una xarxa de transport.

que cada nou usuari trii la ruta 1 ja que resulta més atractiva. Per valors $x > q'$ la ruta 1 deixarà de ser la més atractiva i els següents usuaris es desviaran cap a la ruta 2. Com a conseqüència el flux a la ruta 2 augmentarà i el de la ruta 1 disminuirà. Si aquestes són les dues úniques rutes entre Oi D llavors per cada càrrega de flux x_1 en la ruta 1 la càrrega de flux a la ruta 2 serà l'x_2 que es mostra a la figura (c). Així doncs, el cost temporal a les dues rutes, per càrregues de la ruta 1 superiors a q', serà el mateix.

El principi d'equilibri en una xarxa de transport es pot reformular doncs com:

El temps de viatge en totes les rutes utilitzades és el mateix, i és menor

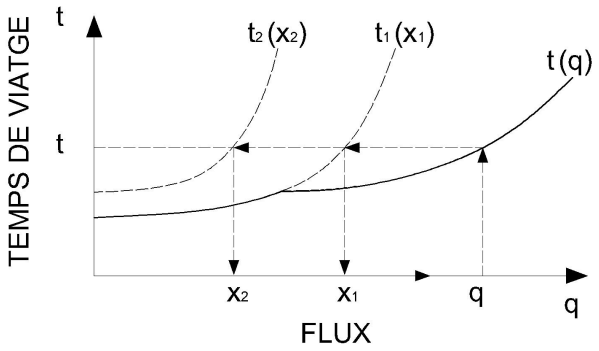

Figura 4.2: Funció de temps de viatge agregada de dues rutes.

o igual que el temps de viatge que experimentaria un sol usuari en qualsevol de les rutes alternatives que no s'utilitzen.

D'aquesta definició d'equilibri es dedueix que donat un temps de viatge t es pot assignar el flux corresponent a cada ruta projectant t horitzontalment en cada ruta, tal com es mostra a la *Figura 4.1* (c). Aquest valor de t es pot trobar sumant les inverses de les funcions de temps de cada ruta, com es mostra a la *Figura 4.2*.

Utilitzant aquesta funció de temps agregada es pot, donada una quantitat determinada de flux entre l'origen i el destí, trobar el temps de viatge corresponent. Una vegada obtingut el temps de viatge, utilitzant les funcions de temps individuals es pot obtenir el flux a cada ruta.

Aquest mètode gràfic és útil per xarxes petites, però quan es tracta de trobar el punt d'equilibri en xarxes grans es necessiten mètodes numèrics. El que es fa es formular un problema de programació matemàtica equivalent a trobar el punt d'equilibri i resoldre'l.

Una possible formulació d'aquest problema es mostra a la *Formulació 2*,

on:

- y_a és el flux assignat a l'aresta a.
- t_a és la funció de cost per l'aresta a.
- x_k^{pq} és el flux assignat al k-èssim camí que va de p a q.
- δ es una matriu tal que:

$$\delta_{a,k}^{pq} = \begin{cases} 1 & \text{si l'aresta } a \text{ forma part del } k\text{-èssim camí de } p \text{ a } q \\ 0 & \text{altrament} \end{cases}$$

- h_{pq} és la quantitat d'usuaris que van de p a q.

$$min_{\mathbf{x}} \quad z(\mathbf{x}) = \sum_a \int_0^{y_a} t_a(w)dw \tag{4.1}$$

Subjecte a:

$$y_a = \sum_p \sum_q \sum_k x_k^{pq} \cdot \delta_{a,k}^{pq} \quad \forall a \tag{4.2}$$

$$\sum_k x_k^{pq} = h_{pq} \quad \forall p, q \tag{4.3}$$

$$x_k^{pq} \geq 0 \quad \forall k, p, q \tag{4.4}$$

Formulació 2: Problema d'equilibri d'usuaris en una xarxa de transport.

El cost de les arestes que s'utilitza és del tipus:

$$t_a(y_a) = t0_a \cdot \left(1 + \alpha_a \cdot \left(\frac{y_a}{c_a}\right)^{\beta_a}\right) \tag{4.5}$$

On:

- y_a és el flux assignat a l'aresta a.

- Els paràmetres α_a i β_a depenen del tipus de via que representa l'aresta i modulen com afecta la saturació de la via al temps que es triga en recórrer-la.

- El valor $t0_a$ és el temps que es triga en recórrer l'aresta en condicions ideals, i es pot estimar com $t0_a = d_a/v_a$ on d_a és la distància separa les cruïlles entre les que discorre l'aresta i v_a és la velocitat màxima permesa en l'aresta. d_a s'obté calculant la distància euclidiana entre les coordenades dels nodes origen i destí de l'aresta, $d_a = \sqrt{(x_d - x_o)^2 + (y_d - y_o)^2}$.

Aquesta transformació del problema de trobar l'equilibri en una xarxa en un problema de programació matemàtica es coneix com el problema de Beckmann. Tot i que aquesta transformació es coneix des dels anys 50s no va ser fins als anys 70s, quan, gràcies a l'aparició de programes informàtics capaços de resoldre aquests problemes, aquesta transformació va esdevenir clau en la planificació de xarxes de transport.

Aquest és un problema no lineal, ja que la funció objectiu no és lineal. Per poder resoldre un problema no lineal cal que tant la funció objectiu com totes les restriccions siguin continues i diferenciables. A més, és convenient que la funció objectiu sigui convexa. El problema formulat compleix tots aquests requisits.

El mètode de Frank i Wolfe [2] fou proposat l'any 1956 per resoldre problemes quadràtics i, posteriorment, va ser generalitzat i aplicat a qualsevol funció no lineal, continua, diferenciable i convexa. A cada iteració l'algorisme considera un punt concret de la funció objectiu i, tenint en compte la forma de la corba al voltant del punt, calcula la direcció de major descens. Aquesta direcció es calcula basant-se en el gradient negatiu de la funció en el punt, però també es tria de tal manera que la factibilitat es mantingui

al llarg de la direcció. Això es pot aconseguir suposant una solució factible qualsevol $\mathbf{y}$. La direcció des del punt actual, $\mathbf{x}^+$, al punt $\mathbf{y}$ és el vector unitari

$$\frac{(\mathbf{y}-\mathbf{x}^+)^t}{||\mathbf{y}-\mathbf{x}^+||} \tag{4.6}$$

i el pendent d'aquest vector s'obté projectant el gradient negatiu de la funció objectiu en el punt $\mathbf{x}^+$ sobre la direcció:

$$-\bigtriangledown z(\mathbf{x}^+)\cdot\frac{(\mathbf{y}-\mathbf{x}^+)^t}{||\mathbf{y}-\mathbf{x}^+||} \tag{4.7}$$

El guany aconseguit en la funció objectiu en passar de $\mathbf{x}^+$ a $\mathbf{y}$ és

$$-\bigtriangledown z(\mathbf{x}^+)\cdot(\mathbf{y}-\mathbf{x}^+)^t \tag{4.8}$$

Per tant, trobar la direcció de descens es redueix a maximitzar aquest guany, tot mantenint la factibilitat del punt $\mathbf{y}$, com es mostra a la *Formulació 3*. Una vegada calculada la direcció de descens òptima $\mathbf{d}$, l'algorisme fa una

$$min \quad \bigtriangledown z(\mathbf{x}^+)\cdot(\mathbf{y}-\mathbf{x}^+)^t=\sum_i\frac{\partial z(\mathbf{x}^+)}{\partial x_i}\cdot(y_i-x_i^+)$$

Subjecte a:

$$y_a=\sum_p\sum_q\sum_k x_k^{pq}\cdot\delta_{a,k}^{pq}\quad\forall a$$

$$\sum_k x_k^{pq}=h_{pq}\quad\forall p,q$$

$$x_k^{pq}\geq 0\quad\forall k,p,q$$

Formulació 3: Formulació del problema que troba la millor direcció de descens

passa en aquesta direcció:

$$\mathbf{x}^{++} = \mathbf{x}^{+} + \alpha \cdot \mathbf{d} \tag{4.9}$$

Per això s'ha de calcular prèviament el valor d'α que comporta un major descens de la funció objectiu. Un altre cop, per calcular aquest valor s'ha de tenir cura de què es mantingui la factibilitat del problema. Per un valor d'α el valor de la funció objectiu és:

$$h(\alpha) = \mathbf{x}^{+} + \alpha \cdot \mathbf{d} \tag{4.10}$$

per tant, el millor valor s'obté minimitzant aquesta funció sobre un interval acotat per mantenir la factibilitat.

Capítol 5

Resolució de l'equilibri d'usuaris

La resolució de l'equilibri d'usuaris es fa aplicant l'algorisme del gradient reduït [6] al problema no lineal formulat a la *Secció 4*.

L'algorisme del gradient reduït, és un mètode de resolució de problemes no lineals d'eficàcia i eficiència provades. Com l'algorisme del símplex aquest mètode també divideix les variables en bàsiques i no bàsiques, amb la diferència de què en els problemes no lineals les variables no bàsiques no tenen per que ser 0.

$$\mathbf{x} = (\mathbf{y}, \mathbf{z}) \tag{5.1}$$

$\mathbf{y}$ són les variables bàsiques i $\mathbf{z}$ les no bàsiques. Sigui el vector $\mathbf{r}$ el vector de costs reduïts, llavors:

$$\begin{cases} z_j > 0 & \Rightarrow f \text{ decreix en la direcció } -r_j \\ z_j = 0 \wedge r_j < 0 & \Rightarrow z_j \text{ creix } \rightarrow f \text{ decreix} \end{cases} \tag{5.2}$$

Per tant, la direcció de descens és:

$$\mathbf{d} = \begin{cases} -r_j & \text{si } r_j < 0 \vee z_j > 0 \\ 0 & \text{altrament} \end{cases} \tag{5.3}$$

A *Algorisme 3* es mostra la formalització en pseudocodi d'una iteració de l'algorisme del gradient reduït.

Aquest és un mètode molt eficient per resoldre problemes de programació lineal. Tot i així té algunes debilitats:

- Si la curvatura en un punt varia molt en una direcció o en una altra l'algorisme pot tardar molt en convergir cap a un òptim local.
- Trobar un punt òptim en la direcció de descens requereix de costosos càlculs addicionals. Veure la *Secció 5.6*.
- Si l'algorisme es troba amb una "vall" que conté un òptim local, però que té una curvatura molt lleugera, l'algorisme pot perdre moltes iteracions canviant lleugerament de direcció i fent passes molt petites. Aquest cas s'il·lustra a la *Figura 5.1*.

L'algorisme del gradient reduït calcula, a mesura que evoluciona, el camins més curts entre un parell de centroides de la xarxa de transport. La peculiaritat de la implementació proposada en aquest projecte és que els camins s'han precalculat amb anterioritat per tot parell de centroides i, per tant, no s'ha realitzar el càlcul a cada passa. A més, el mètode de Frank i Wolfe falla si en algun moment hi ha en la xarxa alguna aresta de cost negatiu. Els camins disponibles es calculen suposant que la xarxa és buida, $y_a = 0 \quad \forall a$, de manera que els costs són sempre positius. Aquest procediment impedeix que el programa falli i a més es simplifica molt cada iteració del mètode del gradient reduït.

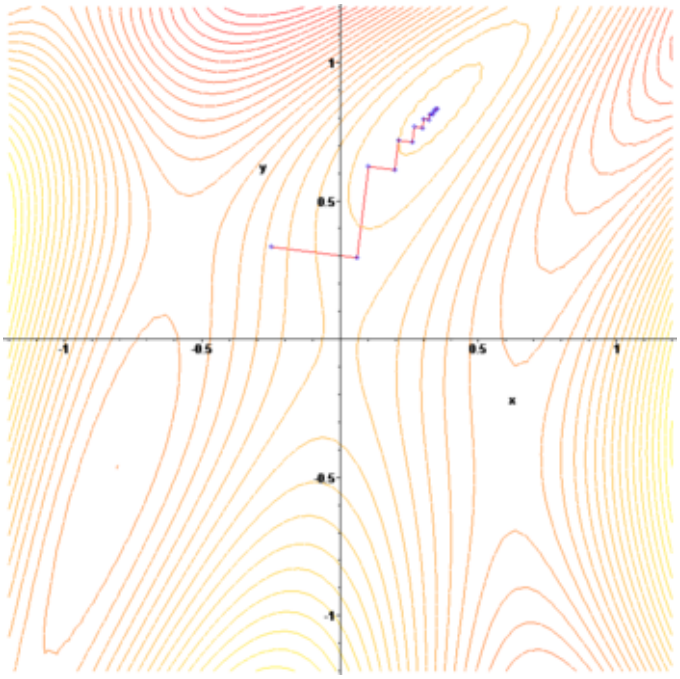

Figura 5.1: Exemple de funció en la que l'algorisme del gradient reduït té problemes per trobar l'òptim local.

Així doncs, el problema que es planteja és el que es mostra a la *Formulació 5*, en el que les variables estan agrupades en k grups $\mathbf{x}^l = (x_1^l, x_2^l, \ldots, x_{n_l}^l)$ de n_l variables cadascun. Cada grup de variables x^l està subjecte a les constriccions $\sum_{i=1}^{n_l} x_i^l = t^l$ amb $t^l > 0$. L'estructura del problema fa que en tot moment només hi ha una variable de cada grup a la base, que es representa mitjançant índexs dobles: $I_B = \{(1, i_1), (2, i_2), \ldots, (k, i_k)\}$

A *Algorisme 5* es mostra la implementació de l'algorisme del gradient reduït que s'ha utilitzat en aquest projecte.

L'aplicació presentada en aquest projecte és, per tant, una aproximació eficient al mètode de Frank i Wolfe que pot ser utilitzada quan aquest mètode sigui massa costós o quan en algun moment de l'algorisme el cost d'alguna aresta es faci negatiu i provoqui un error.

A continuació s'explica amb més detall cada una de les parts claus de

l'algorisme aplicat al problema.

5.1 Variables del problema

La primera particularitat d'aquest problema és que la funció objectiu està expressada en funció dels fluxos assignats a les arestes mentre que les constriccions estan formulades en funció dels fluxos als camins. Aquests dos grups de variables estan relacionats mitjançant la constricció 4.2.

Les variables del problema són els fluxos assignats als camins. Per tant, cada cop que es vol avaluar la funció objectiu s'ha de realitzar la conversió entre els fluxos als camins i els fluxos a les arestes.

Per fer-ho de forma eficient s'ha definit la matriu $\mathbf{D}$ amb tantes files com arestes i tantes columnes com camins de manera que:

$$\mathbf{D}_{i,j} = \begin{cases} 1 & \text{si l'aresta } i \text{ pertany al camí } j \\ 0 & \text{altrament} \end{cases} \tag{5.7}$$

D'aquesta forma el càlcul dels fluxos a les arestes es pot realitzar de forma eficient mitjançant un producte de matrius:

$$\mathbf{y} = \mathbf{D} \cdot \mathbf{x} \tag{5.8}$$

5.2 Funció objectiu

En el càlcul de la funció objectiu intervenen una sèrie de paràmetres que no es modifiquen durant l'execució de l'algorisme i per tant poden calcular-se només un cop.

Si es desenvolupa la integral a dins el sumatori de la funció objectiu 4.1

tenint en compte 4.5 s'obté una formulació alternativa de la funció objectiu:

$$z(\mathbf{y}) = \sum_a \int_0^{y_a} t_a(w)\, dw = t0_a \cdot y_a + \frac{t0_a \cdot \alpha_a}{(\beta_a + 1) \cdot c_a^{\beta_a+1}} \cdot y_a^{\beta_a+1} \tag{5.9}$$

El terme

$$\frac{t0_a \cdot \alpha_a}{(\beta_a + 1) \cdot c_a^{\beta_a+1}}$$

no depèn del valor de y_a i per tant es pot calcular al principi de l'algorisme i utilitzar-lo cada cop que es vulgui avaluar el valor de la funció objectiu. La matriu $\mathbf{R}$ és la que emmagatzema aquests càlculs:

$$R_a = \frac{t0_a \cdot \alpha_a}{(\beta_a + 1) \cdot c_a^{\beta_a+1}} \tag{5.10}$$

Així es pot reescriure el càlcul del valor de la funció objectiu com:

$$z = \sum_a S_a \tag{5.11}$$

on:

$$S_a = t0_a \cdot y_a + R_a \cdot y_a^{(\beta_a+1)} \tag{5.12}$$

La matriu $\mathbf{S}$ ha de ser calculada a cada iteració ja que depèn del valor de y_a.

Per valors de demanda de servei grans el valor de la funció objectiu pot arribar a valors molt grans. Això pot fer que en el càlculs dels costs reduïts, l'exploració lineal o el valor de gap, dels que es donaran més detalls a continuació, es pateixi una pèrdua de precisió. Per solucionar aquest problema es realitza un escalat de la funció objectiu. En la primera iteració es calcula el seu valor normalment. A la resta d'iteracions es divideix el seu valor pel valor obtingut en la primera iteració. D'aquesta forma, a cada iteració el

valor de la funció objectiu s'expressa en tant per u del valor inicial.

5.3 Base

Les variables x_k^{pq} estan agrupades per parells p-q origen-destí. L'estructura del problema fa que només pugui haver-hi una variable de cada grup a la base. A la variable que està a la base pel grup pq la denotem per $x_{i_{pq}}^{pq}$, mentre que a la resta de variables del grup pq que no estan a la base les denotem x_j^{pq}. Per tant, la base $\mathbf{b}$ és una matriu amb una columna i tantes files com parells origen-destí, en la que la posició b_i conté l'índex de la variable del grup i que està a la base.

El càlcul de la base inicial factible és fa de forma trivial introduint la primera variable de cada grup pq. Inicialment, s'assigna tot el flux h_{pq} a la variable bàsica del grup pq i 0 a la resta de variables del grup.

5.4 Costs reduïts

El càlcul dels costs reduïts es fa a partir del gradient de la funció objectiu.

Concretament:

$$r_l^{pq} = \begin{cases} g_l^{pq} - g_{i_{pq}}^{pq} & \text{si } l \neq i_{pq} \\ 0 & \text{altrament} \end{cases} \tag{5.13}$$

on g_j^{pq} és el gradient de la funció objectiu respecte de la variable x_j^{pq}:

$$g_j^{pq} = \frac{\partial z}{\partial x_j^{pq}} \tag{5.14}$$

El gradient de la funció objectiu és fàcilment calculable respecte dels

fluxos a les arestes a partir de 4.1:

$$g_a = \frac{\partial z}{\partial x_a} = t_a(x_a) \tag{5.15}$$

Però les variables del problema són els fluxos als camins, per tant el gradient s'ha de calcular respecte d'aquestes variables. Si es considera un camí de p a q, el seu flux és la suma dels fluxos de les arestes que en formen part. Per tant, el gradient de la funció objectiu respecte del flux en un camí es la suma dels gradients de la funció objectiu respecte dels fluxos en les arestes que formen part del camí. Per fer aquest càlcul de forma eficient utilitzem la matriu $\mathbf{D}$:

$$\mathbf{g} = \mathbf{D}^t\mathbf{s} \tag{5.16}$$

on:

$$s_a = t0_a \cdot \left(1 + \alpha_a \cdot \left(\frac{y_a}{c_a}\right)^{\beta_a}\right) \tag{5.17}$$

La matriu $\mathbf{s}$ ha de ser calculada a cada iteració de l'algorisme ja que el seu valor depèn de y_a.

5.5 Direcció de descens

Per calcular la direcció de descens es distingeix entre variables bàsiques i variables no bàsiques.

Per les variables no bàsiques només té sentit continuar descendint si són estrictament majors que 0 o si els costs reduïts són negatius. En cas contrari s'obtendrien valors negatius de la variable — prohibit per 4.4 — o s'augmentaria el valor de la funció objectiu — que és el contrari del que es pretén.

Per tant:

$$d_j^{pq} = \begin{cases} -r_j^{pq} & \text{si } r_j^{pq} < 0 \text{ ó } x_j^{pq} > 0 \\ 0 & \text{altrament} \end{cases} \tag{5.18}$$

La direcció de descens en les posicions de les variables bàsiques pren el valor del negatiu del sumatori de les variables no bàsiques del grup:

$$d_{i_{pq}}^{pq} = - \sum_{j=1_{j \neq i_{pq}}}^{n_{pq}} d_j^{pq} \tag{5.19}$$

5.6 Exploració lineal

Un cop calculada la direcció de descens cal trobar un nou punt en aquesta direcció:

$$\mathbf{x}^+ = \mathbf{x} + \alpha \cdot \mathbf{d} \tag{5.20}$$

tal que $z(\mathbf{x}^+) < z(\mathbf{x})$

Si es defineix $h(\alpha)$ tal que:

$$h(\alpha) = f(\mathbf{x} + \alpha \cdot \mathbf{d}) \tag{5.21}$$

el que es pretén, idealment, és trobar un α^* tal que:

$$\alpha^* = min_{\alpha \geq 0} \quad h(\alpha) \tag{5.22}$$

Calcular el valor d'α^* no és fàcil i és un aspecte fonamental d'aquest algorisme. Hi ha diferents enfocs per determinar aquest valor:

- Fixar un valor constant. Aquesta estratègia és, òbviament, la més eficient, però dóna resultats poc satisfactoris, ja que no es pot determinar a priori quin serà un bon valor d'α^* per un problema concret.

- Fer una optimització no lineal sobre la direcció **d** per trobar el punt òptim. Aquest enfoc ofereix els millors resultats, però és computacionalment molt costós i, per tant, poc pràctic a l'hora de la veritat.

- Aproximar el valor d'α^*, suposant que $h(\alpha)$ és quadràtica. Aquesta estratègia dóna uns resultats prou bons sense penalitzar excessivament l'eficiència global de l'algorisme.

En aquest projecte es fa servir l'última opció, ja que combina eficiència i eficàcia en la mesura justa.

En avançar en la direcció de descens algunes variables es fan més petites. La restricció 4.4 obliga a què totes les variables siguin positives. Per tant, el primer que cal fer és trobar els valors d'α que fan que alguna variable es torni 0. Aquests valors imposen una fita superior més enllà de la qual no es pot anar en la direcció de descens. Fem aquest càlcul per les variables bàsiques i les no bàsiques per separat, per després poder decidir si hi ha canvi de base o no.

El límit que imposen les variables bàsiques és:

$$\alpha_1 = \begin{cases} +\infty & \text{si } d_{i_{pq}}^{pq} \geq 0,\ \forall p,q \\ min_{1 \leq pq \leq k} \left\{ \frac{-x_{i_{pq}}^{pq}}{d_{i_{pq}}^{pq}} \middle| d_{i_{pq}}^{pq} < 0 \right\} & \text{altrament} \end{cases} \tag{5.23}$$

El límit imposat per les variables no bàsiques és:

$$\alpha_2 = \begin{cases} +\infty & \text{si } d_j^{pq} \geq 0\ \forall (pq, j \neq i_{pq}) \\ min_{pq, j \neq i_{pq}} \left\{ -\frac{x_j^{pq}}{d_j^{pq}} \middle| d_j^{pq} < 0 \right\} & \text{altrament} \end{cases} \tag{5.24}$$

Una vegada fixats α_1 i α_2 es fa una optimització quadràtica per trobar el valor òptim entre 0 i $\theta = min(\alpha_1, \alpha_2)$ que minimitzi 5.21

Com que es suposa $h(\alpha)$ quadràtica, es fa un ajust quadràtic per expressa-la com:

$$h(\alpha) = \frac{1}{2} \cdot a \cdot \alpha^2 + b \cdot \alpha + c \tag{5.25}$$

Agafant com punts de pas de la funció:

$$h_0 = h(0) \tag{5.26}$$

$$h_1 = h(\theta) \tag{5.27}$$

$$h_2 = h\left(\frac{\theta}{2}\right) \tag{5.28}$$

s'obtenen els valors:

$$a = \frac{-8}{\theta^2} \cdot \left(h_2 - h_0 - \frac{h_1 - h_0}{2}\right) \tag{5.29}$$

$$b = \frac{-1}{\theta} \cdot (h_1 - h_0 - 4 \cdot (h_2 - h_0)) \tag{5.30}$$

$$c = h_0 \tag{5.31}$$

A continuació es calcula el resultat de la optimització:

$$\alpha_3 = min_{\alpha \geq 0}\, h(\alpha)$$

que, al ser h és fàcilment calculable com:

$$\alpha_3 = \begin{cases} -\frac{b}{a} = \frac{\theta}{4} \cdot \frac{h_1 - h_0 - 4 \cdot (h_2 - h_0)}{2 \cdot (h_2 - h_0) - (h_1 - h_0)} & \text{si } b < 0 \\ 0 & \text{si } b \geq 0 \end{cases} \tag{5.32}$$

El valor α^* que finalment s'aplica és:

$$\alpha^* = min\{\alpha_1, \alpha_2, \alpha_3\} \tag{5.33}$$

5.7 Canvi de base

Si el valor α^* calculat a l'apartat anterior és major que el límit que imposaven les variables bàsiques, és a dir, si $\alpha_3 \geq \alpha_1$, això indica que cal fer un canvi de base.

Sigui s tal que:

$$\alpha_1 = \frac{x^s_{i_s}}{d^s_{i_s}}$$

llavors $x^s_{i_s}$ surt de la base. La nova variable que entra a la base és qualsevol ${x^+}^s_j > 0$.

5.8 RGAP

La funció RGAP — *relative gap* — es defineix per tal de quantificar la magnitud de la millora obtinguda en la última passa donada per l'algorisme. Utilitzant aquest valor es pot aturar l'algorisme quan es considera que s'ha entrat en una dinàmica en la que les petites millores obtingudes a cada iteració fan que no mereixi la pena l'esforç de continuar.

Per calcular aquest valor primer s'obté el resultat de la següent optimització:

$$min_{\mathbf{w}} \quad \mathbf{g}^t \cdot \mathbf{w} \tag{5.34}$$

Subjecte a les mateixes constriccions que el problema original.

Aquest resultat es pot calcular de forma trivial assignant el valor h_{pq} a la posició del grup pq en la que el gradient és mínim i 0 a la resta de variables del grup.

Una vegada trobat aquest punt $\mathbf{w}$ el valor de gap relatiu és calcula com:

$$gap = \frac{\mathbf{g}^t \cdot (\mathbf{w} - \mathbf{x}^+)}{\mathbf{g}^t \cdot \mathbf{x}^+} \tag{5.35}$$

El criteri d'aturada de l'algorisme és que aquest valor sigui menor que un cert llindar ϵ, parametritzable, de l'ordre de 10^{-2}.

Algorisme 3 Formalització de l'algorisme del gradient reduït en la iteració i.

1: $\mathbf{x}^i = (\mathbf{y}, \mathbf{z})$
   $\mathbf{z}$ variables no bàsiques
   $\mathbf{y}$ variables bàsiques
2: Base: $\mathbf{I}_B^i = \{1, 2, \ldots, m\}$
3: Costs:

$$\mathbf{c}_B = \bigtriangledown_y f(\mathbf{x}^i)$$
$$\mathbf{c}_N = \bigtriangledown_z f(\mathbf{x}^i)$$

4: Costs reduïts: $\mathbf{r} = \mathbf{c}_N - \mathbf{Y}^t \cdot \mathbf{c}_B$
5: Direcció de descens de les no bàsiques:

$$\triangle z_j = \begin{cases} -r_j & \text{si } r_j < 0 \vee z_j > 0 \\ 0 & \text{altrament} \end{cases}$$

6: **if** $\triangle z_j = 0$ **then**
7: S'ha trobat l'òptim
   retornar $\mathbf{x}^i$
8: **else**
9: Direcció de descens de les bàsiques:
   $\triangle y = -\mathbf{Y}^t \cdot \triangle \mathbf{z}$
10: Càlcul de la passa:

$$\alpha_1 = max\{\alpha \geq 0 | \mathbf{y}^i + \alpha \cdot \triangle \mathbf{y} \geq 0\}$$
$$\alpha_2 = max\{\alpha \geq 0 | \mathbf{z}^i + \alpha \cdot \triangle \mathbf{z} \geq 0\}$$
$$\alpha_3 = min_{\alpha \geq 0} f(\mathbf{x}^i + \alpha \cdot \triangle x)$$
$$\alpha^* = min\{\alpha_1, \alpha_2, \alpha_3\}$$

11: $\mathbf{x}^{i+1} = \mathbf{x}^i + \alpha^* \cdot \triangle x$
12: **if** $\alpha_3 \geq \alpha_1$ **then**
13: Canvi de base:
   S'anul·la la variable bàsica p. Escollir variable no bàsica q
   $I_B^{i+1} = I_B^i \cup \{q\} \setminus \{p\}$
14: **end if**
15: Tornar a 3
16: **end if**

$$min_{\mathbf{x}} \quad f(\mathbf{x}^1\ \mathbf{x}^2\ \cdots\ \mathbf{x}^k) \tag{5.4}$$

Subjecte a:

$$\sum_{i=1}^{n_l} x_i^l = t^l \quad l = 1, 2, \ldots, k \tag{5.5}$$

$$x_i^l \geq 0 \tag{5.6}$$

Formulació 4: Formalització del problema que es resol en aquest projecte.

Algorisme 4 Algorisme implementat en aquest projecte

1: $I_B \leftarrow$ selecció d'una base inicial factible
Simplement s'ha de verificar que $x^l_{i_l} = t^l \quad \forall l$ i que $x^l_j = 0 \quad \forall (j,l) \notin I_B$
2: **if** $RGAP(\mathbf{x}) \leq \epsilon$ **then**
3: $\mathbf{x}$ és un punt molt proper a l'òptim
retornar x
4: **else**
5: Càlcul dels costs reduïts
$r^l_j = \frac{\partial f}{\partial x^l_j} - \frac{\partial f}{\partial x^l_{i_l}} \quad \forall (l,j) \notin I_B$
$r^l_{i_l} = 0 \quad l = 1, 2, \ldots, k$
6: Càlcul de la direcció de descens

$$d^l_j = \begin{cases} -r^l_j & \text{si } r^l_j < 0 \vee x^l_j > 0 \\ 0 & \text{altrament} \end{cases} \qquad \forall (l,j) \notin I_B$$

$d^l_{i_l} = -\sum_{j=1_{j \neq i_l}}^{(} n_l) d^l_j \quad \forall (l, i_l) \in I_B$
7: Exploració lineal

$$\alpha_1 = \begin{cases} \infty & \text{si } d^l_{i_l} \geq 0 \; \forall l \\ min_{1 \leq l \leq k} \left\{ \frac{-x^l_{i_l}}{d^l_{i_l}} \middle| \, d^l_{i_l} < 0 \right\} & \text{altrament} \end{cases}$$

$$\alpha_2 = \begin{cases} \infty & \text{si } d^l_j \geq 0 \; \forall (l,j) \notin I_B \\ min_{(l,j) \notin I_B} \left\{ \frac{-x^l_j}{d^l_j} \middle| \, d^l_j < 0 \right\} & \text{altrament} \end{cases}$$

$\alpha_3 = min_{\alpha \geq 0} \, f(\mathbf{x} + \alpha \cdot \mathbf{d})$
$\alpha^* = min \, \{\alpha_1, \alpha_2, \alpha_3\}$

8: $\mathbf{x}^+ = \mathbf{x} + \alpha^* \cdot \mathbf{d}$
9: **if** $\alpha_3 \geq \alpha_1$ **then**
10: Canvi de base
Si p és tal que $\alpha_1 = -\frac{x^p_{i_p}}{d^p_{i_p}}$
llavors triar $(\mathbf{x}^+)^p_j > 0$
$\mathbf{I}^+_B = (\mathbf{I}_B - \{(p, i_p)\}) \cup \{p, j\}$
11: **end if**
12: Nova iteració sobre $\mathbf{x}^+$ i $\mathbf{I}^+_B$ Tornar a 2
13: **end if**

Capítol 6

Resultats

En aquest capítol es presenten alguns resultats obtinguts per l'aplicació en algunes xarxes de prova. El número màxim de camins considerat per cada possible parell origen-destí és 5.

Per cada problema es mostra la següent informació:

- Una representació gràfica de la xarxa d'entrada si està disponible.
- Una taula amb els fluxos entre tot parell de centroides origen-destí.
- Una taula amb els paràmetres de les arestes.
- Una taula amb els passos identificats per l'aplicació.
- Una taula amb els resultats obtinguts a cada iteració de l'algorisme amb les següents columnes:

    1. **Iteració:** Número de iteració a la que corresponen els valors.
    2. **Funció objectiu:** Valor de la funció objectiu en la iteració.
    3. **min(r):** Valor del mínim cost reduït en la iteració.
    4. **Gap:** Valor de *gap* obtingut en la iteració.

5. **Factibilitat:** Booleà que indica si el punt considerat en la iteració és factible.

- Una taula amb els valors dels fluxos als camins en el punt òptim.

El problema 3 és una xarxa de 349 nodes — dels quals 23 són centroides — i 846 arestes. El número de camins trobat per l'aplicació és de 2530. Les dimensions del problema fan que no sigui convenient mostrar aquí més que la informació justa, veure 6.

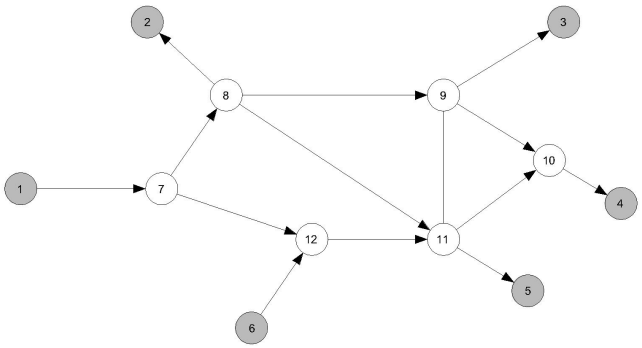

Figura 6.1: PROBLEMA 1 – Xarxa de transport d'entrada. Els nodes en gris corresponen a centroides.

En la *Figura 6.4 i 6.5* es mostra la relació entre el temps de càlcul requerit per l'algorisme del gradient reduït en funció de la demanda agregada de servei en una xarxa. Per obtenir aquests resultats s'han multiplicat les matrius de demandes origen-destí dels problemes 1 i 3 per un paràmetre k entre 0 i 100. Els resultats obtinguts demostren que el temps de càlcul no depèn de la demanda de servei agregada.

Taula 6.1: PROBLEMA 1 – Fluxos entre els parells de centroides origen-destí.

		Destins					
		1	**2**	**3**	**4**	**5**	**6**
	1	0	10	15	5	10	0
	2	0	0	0	0	0	0
Orígens	**3**	0	0	0	0	0	0
	4	0	0	0	0	0	0
	5	0	0	0	0	0	0
	6	0	15	10	10	20	0

Taula 6.2: PROBLEMA 1 – Paràmetres de les arestes.

Identificador	**Node origen**	**Node destí**	**Capacitat**	α	β	**t0**
1	1	7	2	0.15	4	10
2	7	8	4	0.15	4	20
3	8	2	4	0.15	4	20
4	8	9	3	0.15	4	25
5	8	11	2.5	0.25	4	10
6	9	3	5	0.10	3	20
7	9	10	3	0.15	4	25
8	10	4	2	0.15	4	10
9	7	12	4	0.15	4	20
10	6	12	2.5	0.20	4	15
11	12	11	4	0.15	4	20
12	11	9	2.5	0.20	4	15
13	11	10	2	0.15	4	10
14	11	5	2.5	0.20	4	15

Taula 6.3: PROBLEMA 1 – Passos identificats per l'aplicació.

ID camí	**ID parell origen-destí**	**Camí**	**Cost**
1	2	1,7,8,2	50.0
2	3	1,7,8,11,9,3	75.0
3	3	1,7,8,9,3	75.0
4	3	1,7,12,11,9,3	85.0
5	4	1,7,8,11,10,4	60.0
6	4	1,7,12,11,10,4	70.0
7	4	1,7,8,9,10,4	90.0
8	4	1,7,8,11,9,10,4	90.0
9	4	1,7,12,11,9,10,4	100.0
10	5	1,7,8,11,5	55.0
11	5	1,7,12,11,5	65.0
12	33	6,12,11,9,3	70.0
13	34	6,12,11,10,4	55.0
14	34	6,12,11,9,10,4	85.0
15	35	6,12,11,5	50.0

Taula 6.4: PROBLEMA 1 – Resultats obtinguts a cada iteració de l'algorisme.

Iteració	**Funció objectiu**	**min(r)**	**gap**	**Factibilitat**
1	2062195.175	-0.0397	0.0262	TRUE
2	0.9486	-0.0155	0.0191	TRUE
3	0.9387	-0.0147	0.0083	TRUE

Taula 6.5: PROBLEMA 1 – Fluxos als camins en el punt òptim.

Variable	**Valor**
1	10
2	7.5803
3	7.4197
4	7.9936e-15
5	0.8063
6	1.1102e-15
7	4.1937
8	0
9	-1.1102e-16
10	10
11	0
12	10
13	10
14	0
15	20

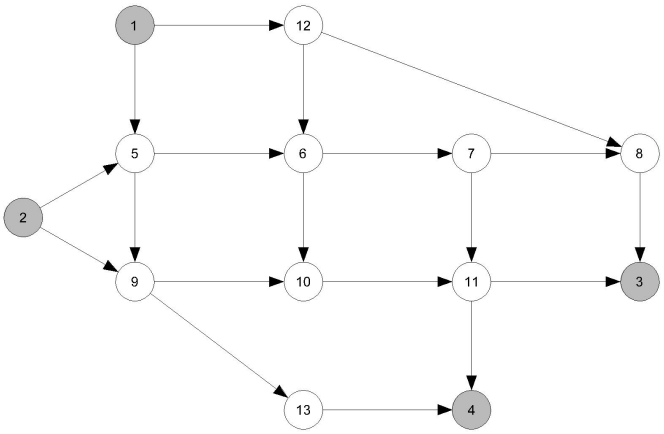

Figura 6.2: PROBLEMA 2 – Xarxa de transport d'entrada. Els nodes en gris corresponen a centroides.

Taula 6.6: PROBLEMA 2 – Fluxos entre els parells de centroides origen-destí.

		Destins			
		1	**2**	**3**	**4**
	1	0	0	400	800
Orígens	**2**	0	0	600	200
	3	0	0	0	0
	4	0	0	0	0

Taula 6.7: PROBLEMA 2 – Paràmetres de les arestes.

Identificador	**Node origen**	**Node destí**	**Capacitat**	α	β	**t0**
1	1	12	2	0.15	4	10
2	1	5	4	0.15	4	20
3	2	5	4	0.15	4	20
4	2	9	3	0.15	4	25
5	5	6	2.5	0.25	4	10
6	5	9	5	0.10	3	20
7	12	6	3	0.15	4	25
8	12	8	2	0.15	4	10
9	6	7	4	0.15	4	20
10	6	10	2.5	0.20	4	15
11	7	8	4	0.15	4	20
12	7	11	2.5	0.20	4	15
13	8	3	2	0.15	4	10
14	9	10	2.5	0.20	4	15
15	9	13	2.5	0.20	4	15
16	10	11	2.5	0.20	4	15
17	11	3	2.5	0.20	4	15
18	11	4	3	0.15	4	25
19	13	4	2.5	0.20	4	15

Taula 6.8: PROBLEMA 2 – Passos identificats per l'aplicació.

ID camí	**ID parell origen-destí**	**Camí**	**Cost**
1	3	1,12,8,3	30.0
2	3	1,5,6,10,11,3	75.0
3	3	1,12,6,10,11,3	80.0
4	3	1,5,6,7,11,3	80.0
5	3	1,5,6,7,8,3	80.0
6	4	1,5,9,13,4	70.0
7	4	1,5,6,10,11,4	85.0
8	4	1,12,6,10,11,4	90.0
9	4	1,5,6,7,11,4	90.0
10	4	1,5,9,10,11,4	95.0
11	7	2,9,10,11,3	70.0
12	7	2,5,6,10,11,3	75.0
13	7	2,5,6,7,11,3	80.0
14	7	2,5,6,7,8,3	80.0
15	7	2,5,9,10,11,3	85.0
16	8	2,9,13,4	55.0
17	8	2,5,9,13,4	70.0
18	8	2,9,10,11,4	80.0
19	8	2,5,6,10,11,4	85.0
20	8	2,5,6,7,11,4	90.0

Taula 6.9: PROBLEMA 2 – Resultats obtinguts a cada iteració de l'algorisme.

Iteració	**Funció objectiu**	**min(r)**	**gap**	**Factibilitat**
1	15212958145691.4	-0.0113	0.877	TRUE
2	0.3665	-0.0041	0.4632	TRUE
3	0.2015	-0.0016	0.3651	TRUE
4	0.185	-0.0011	0.2529	TRUE
5	0.1738	-6e-04	0.2424	TRUE
6	0.1729	-6e-04	0.0896	TRUE
7	0.1642	-3e-04	0.1168	TRUE
8	0.1595	-3e-04	0.0752	TRUE
9	0.1568	-2e-04	0.1078	TRUE
10	0.1547	-3e-04	0.0564	TRUE
11	0.1535	-1e-04	0.0474	TRUE
12	0.1511	-1e-04	0.0263	TRUE
13	0.1508	0	0.0267	TRUE
14	0.1508	0	0.026	TRUE

Taula 6.10: PROBLEMA 2 – Fluxos als camins en el punt òptim.

Variable	**Valor**
1	400
2	0
3	0
4	0
5	0
6	519.5287
7	0
8	90.8372
9	145.7804
10	43.8538
11	323.1750
12	0
13	128.4568
14	121.8379
15	26.5303
16	0
17	3.3439
18	46.4453
19	0
20	150.2108

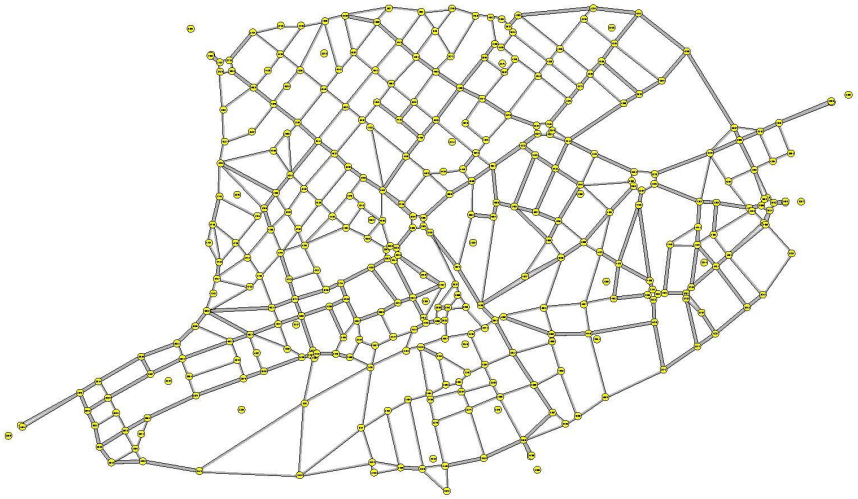

Figura 6.3: PROBLEMA 3 – Xarxa de transport d'entrada.

Taula 6.11: PROBLEMA 3 – Resultats obtinguts a cada iteració de l'algorisme.

Iteració	**Funció objectiu**	**min(r)**	**gap**	**Factibilitat**
1	104824539.4521	-2.8122e-05	0.2161	TRUE
2	0.9965	-2.8095e-05	0.2095	TRUE
3	0.9881	-2.8031e-05	0.2071	TRUE
4	0.9851	2.8009e-05	0.2064	TRUE
5	0.9842	-2.8003e-05	0.2049	TRUE
6	0.9823	-2.7988e-05	0.2048	TRUE
7	0.9823	-2.7987e-05	0.2034	TRUE
8	0.9804	-2.7973e-05	0.2027	TRUE
9	0.9796	-2.7967e-05	0.1973	TRUE
10	0.9728	-2.7915e-05	0.1958	TRUE
11	0.9711	-2.7902e-05	0.1853	TRUE
12	0.9584	-2.7809e-05	0.1849	TRUE
13	0.9578	-2.7805e-05	0.1842	TRUE
14	0.957	-2.7799e-05	0.1842	TRUE
15	0.957	-2.7799e-05	0.1832	TRUE
16	0.9558	-2.7791e-05	0.1794	TRUE
17	0.9513	-2.7759e-05	0.175	TRUE
18	0.9462	-2.7723e-05	0.1749	TRUE
19	0.9461	-2.7722e-05	0.1746	TRUE
20	0.9458	-2.7720e-05	0.1717	TRUE
21	0.9423	-2.7684e-05	0.1717	TRUE
22	0.9423	-2.7019e-05	0.1709	TRUE
23	0.9414	-2.7014e-05	0.1705	TRUE
24	0.9409	-2.7012e-05	0.1664	TRUE
25	0.9362	-2.6989e-05	0.1637	TRUE
26	0.9331	-2.6974e-05	0.1617	TRUE
27	0.9309	-2.5782e-05	0.16	TRUE
28	0.9289	-2.5769e-05	0.16	TRUE
29	0.9289	-2.5769e-05	0.16	TRUE

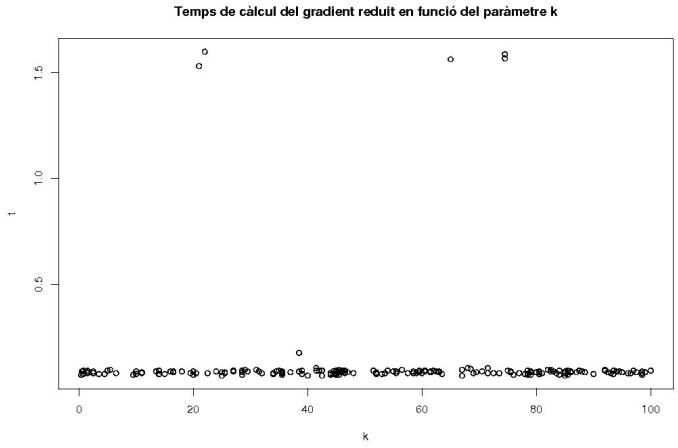

Figura 6.4: Temps de càlcul requerit per l'algorisme del gradient reduït en funció del paràmetre k aplicat a la matriu de demanda del problema 1.

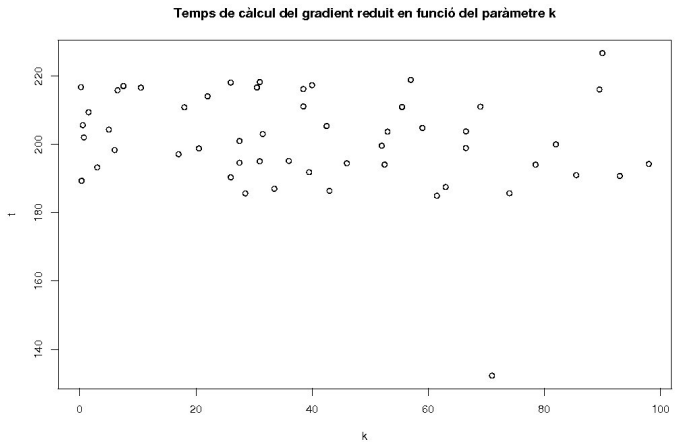

Figura 6.5: Temps de càlcul requerit per l'algorisme del gradient reduït en funció del paràmetre k aplicat a la matriu de demanda del problema 3.

Capítol 7

Implementacions alternatives

A més de la implementació de l'algorisme del gradient reduït, en aquest projecte s'ha construït un programa genètic que també resol el problema.

Els programes genètics són programes que resolen problemes inspirant-se en el procés evolutiu de les especies dirigit per la selecció natural. Aquest tipus de programes són útils quan no es coneix la millor forma de resoldre un problema, o quan la implementació d'un mètode exacte resulta massa complicada. Els programes genètics són capaços de resoldre problemes molt complicats amb una estructura molt senzilla i fàcil de programar. La contrapartida, però, és el temps de computació que requereixen. Com que no estan dirigits per cap tipus de coneixement sobre el problema, sinó que simplement es basen en creuaments gairebé aleatoris, aquests tipus de programes poden trigar molt en trobar una solució, fins i tot podrien no trobar-la mai. Tanmateix a la pràctica aquests algorismes no es comporten tan malament com diu la teoria i poden retornar resultats interessants.

Els algorismes genètics intenten emular el procés de selecció natural de les espècies, en el que només els individus més adaptats sobreviuen i es reprodueixen. Per poder-los aplicar a un problema cal definir, en el context

del problema, que és un individu, quina és la funció d'adaptació, i com es realitzen els aparellaments:

- Els individus: En el context d'un algorisme genètic un individu és una representació d'una solució del problema. En el cas del problema que es vol resoldre, un individu és una assignació de fluxos als camins codificada com un vector de valors reals.

- Una funció d'adaptació: Aquesta és la funció que descriu com s'adapta un individu al seu entorn i determina la probabilitat d'aparellar-se que té. És aquesta funció la que dirigeix l'evolució dels individus i, per tant, és aquesta funció la que es maximitza o minimitza. En el cas del problema que es vol resoldre aquest paper el juga la funció objectiu.

- Mètode d'aparellament: Cal definir, finalment, com dos individus s'aparellen per generar-ne un de nou. Per al problema en qüestió s'ha definit que el resultat d'emparellar dues solucions és una solució tal que cada assignació de flux a un camí és la mitjana aritmètica de les assignacions dels seus pares al camí. En els mecanismes reproductius de la Natura es produeixen errors de còpia del material genètic, la qual cosa dóna lloc a noves mutacions que poden ser beneficioses per l'espècie. Aquí també es simula aquest comportament afegint un error aleatori en realitzar la mitjana de les solucions dels pares. La magnitud de l'error depèn de la proximitat de les solucions dels pares. D'aquesta manera s'aconsegueix també un efecte de consanguinitat, explicat amb més detall més endavant.

Per adaptar un algorisme genètic al problema, però, cal fer algunes definicions més que són requerides per l'estructura del problema.

- Població: La població és l'estructura a on es mantenen tots els individus generats. En aquest cas s'ha definit la població com una cua d'individus ordenats en ordre ascendent segons el seu valor de la funció d'adaptació. És a dir, el primer individu de la cua és el que té un valor de la funció objectiu més baix, i l'últim és el que té el valor de la funció objectiu més alt. Els individus al capdavant de la cua són els "millors" i els de la part del darrera són els "dolents". Per raons d'estalvi de memòria, i també per descartar individus molt dolents es manté una població d'individus constant. Si es genera un individu nou que es millor que l'individu més dolent, es descarta l'últim element de la cua i s'afegeix el nou individu.

- Funció de viabilitat: En el problema que es vol resoldre no tots els individus són vàlids, sinó que s'han de complir unes constriccions. Per tal de què l'algorisme es comporti degudament cal definir una funció que indiqui si un nou individu és viable o no. Si l'individu no és viable es considera un avort i no s'afegeix a la població d'individus. Per aquest problema s'han modificat una mica les constriccions. Les constriccions de demanda del problema són del tipus $\sum_{pq} x_k^{pq} = h_{pq}$. Com que l'algorisme treballa amb valors reals i en cada creuament fa la mitjana aritmètica de dues solucions, és gairebé impossible complir aquestes restriccions, ja que la igualtat estricte mai es compleix. Per tant, el que s'ha fet es redefinir les constriccions de demanda a la forma $\sum_{pq} x_k^{pq} \geq h_{pq}$. Aquesta modificació és més permissiva i no afecta a la solució perquè la funció objectiu és minimitzar els fluxos x_k^{pq} i, per tant, la solució tendirà cap a punts que compleixin $\sum_{pq} x_k^{pq} = h_{pq}$.

- Funció de probabilitat: La funció d'adaptació indica com de bona és

una solució, però no diu com escollir dos individus perquè es reprodueixin. Això és el que defineix la funció de probabilitat. L'objectiu és seleccionar dos individus "bons" i creuar-los per generar una nova solució. Però també es vol permetre una certa variabilitat. Si sempre s'escullen els individus més adaptats es corre el risc d'estancar-se en una solució que, tot i ser la millor de les que s'han generat, no és prou bona. Per això es crea una funció que genera dos números aleatoris seguint una distribució donada i es creuen els individus que estan en les posicions generades. La funció ha de retornar valors entre 1 i n, on n és el número d'individus a la població. La distribució de probabilitat que s'ha utilitzat és la que es mostra a la *Figura 7.1*, ja que presenta una probabilitat decreixent respecte de l'adaptació dels individus, però no en descarta cap. Òbviament aquesta funció ha de complir $\int_1^n f = 1$.

- Funció de millora: Aquest algorisme no té cap manera de saber si una solució és òptima o no. Per tant, cal alguna condició que determini quan s'ha de deixar de generar nous individus. El que es fa es crear una funció de millora relativa, del tipus:

$$RGAP = \frac{z_i - z_{i+1}}{z_i} \tag{7.1}$$

  on z_i és el valor del millor individu en la passa i i z_{i+1} és el valor del millor individu en la passa $i + 1$. Aquest valor dóna una idea de la magnitud de la millora aconseguida en la passa $i + 1$. Si aquest valor és menor que un paràmetre ϵ es para l'algorisme i es dóna per bona la solució z_{i+1}.

- Funció de consanguinitat: En la Natura, els fills d'individus amb genomes molt semblants donen lloc a individus amb grans mutacions. D'aquest

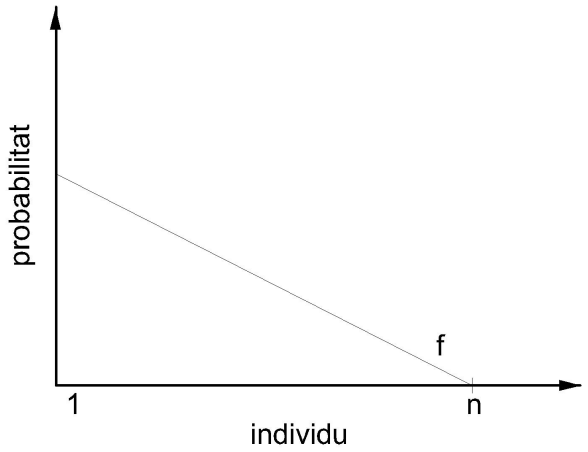

Figura 7.1: Forma de la funció de distribució utilitzada per seleccionar els individus a emparellar en l'algorisme genètic.

fenomen se'n diu consanguinitat. L'explicació d'aquest fenomen no és gaire clara, però sembla raonable pensar que poblacions amb alts índexs de consanguinitat són poblacions reduïdes i molt a prop de l'extinció. Així doncs, sembla que els mecanismes de reproducció, que també són producte de la selecció natural generen grans mutacions quan es detecta que la població es troba en un punt crític per tal d'accelerar el procés evolutiu i donar més possibilitats de supervivència a l'espècie. Aquesta idea ha estat adaptada a l'algorisme genètic construït. Així, en creuar dues solucions molt similars s'introdueixen errors aleatoris més grans del normal, donant lloc a noves solucions molt llunyanes que poden obrir nous camins per explorar. Aquest ajust pot fer que l'algorisme es desestanqui quan està en un òptim local i portar-lo cap a un altre òptim millor. Òbviament, aquest és un procés aleatori, el fet que les noves mutacions portin a bon port depèn gairebé exclusivament de la sort, igual que a la Natura.

A *Algorisme 5* es pot veure un esquema del funcionament d'un algorisme genètic un cop fetes aquestes definicions.

Algorisme 5 Esquema de funcionament d'un algorisme genètic.

1: $p \leftarrow$ generar-poblacio()
2: **repeat**
3: $< i, j > \leftarrow$ nombres-aleatoris()
4: nou-individu $\leftarrow$ creuar($p[i]$,$p[j]$)
5: afegir(p,nou-individu)
6: **until** $RGAP \leq \epsilon$

Capítol 8

Cost del projecte

Aquest projecte es divideix en quatre fases:

1. Anàlisi del problema: En aquesta fase un analista estudia el problema i n'analitza els punts claus. El perfil d'analista, en aquest cas, pot correspondre a un enginyer informàtic o a un llicenciat en matemàtica. En aquesta fase també cal fer una revisió dels algorismes més adients per a implementar cada part del problema.

2. Disseny: En aquesta fase un arquitecte *software*, típicament un enginyer informàtic, estudia l'anàlisi realitzat en la fase anterior i defineix la divisió modular de l'aplicació, decideix els llenguatges de programació utilitzats i la plataforma en la que es desplegarà el sistema *software*.

3. Implementació: En aquesta fase un programador construeix l'aplicació basant-se en el disseny realitzat en la fase anterior.

4. Proves: En la fase final del projecte un analista realitza una sèrie de proves al sistema i en verifica el bon funcionament.

El pressupost estimat d'aquest projecte és el següent:

Tasca	**Hores**	**Perfil**	**Cost per hora**	**Cost**
Anàlisi del problema	260	Analista	20 €/hora	5200 €
Disseny	120	Arquitecte software	30 €/hora	3600 €
Implementació	400	Programador	15 €/hora	6000 €
Proves	80	Analista	20 €/hora	1600 €

El **cost total** aproximat és de 16400 €.

Bibliografia

[1] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of a*. *J. ACM*, 32(3):505–536, 1985.

[2] M. Frank and P. Wolfe. *An algorithm for quadratic programming*, 1956.

[3] E. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart, Winston, 1976.

[4] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[5] Y. Sheffi. *Urban transportation networks*. Prentice-Hall, 1985.

[6] P. Wolfe. The reduced gradient method. Unpublised manuscript.

Apèndix A

Glossari

Graf: Un graf $G = (V, A)$ és un model matemàtic format per un conjunt V de n nodes $\{v_1, v_2, \ldots, v_n\}$ i un conjunt A de m arestes $\{a_1, a_2, \ldots, a_m\}$, on cada aresta $a_i = (v^i_{inicial}, v^i_{final})$ estableix una relació entre un parell de nodes.

Graf no dirigit: Un graf no dirigit $G = (V, A)$ és un graf que compleix:

$$a_i = (v^i_{inicial}, v^i_{final}) \in A \Rightarrow (v^i_{final}, v^i_{inicial}) \in A \quad \forall i$$

Graf dirigit: Un graf dirigit $G = (V, A)$ és un graf que compleix:

$$\exists i \quad a_i(v^i_{inicial}, v^i_{final}) \in A \ \wedge \ (v^i_{final}, v^i_{inicial}) \notin A$$

Graf amb cost: Un graf $G = (V, A)$ pot tenir una funció de cost associada:

$$c : VxV \longrightarrow \mathbb{R}$$

que relaciona cada parell de nodes (o cada aresta) amb un valor real.

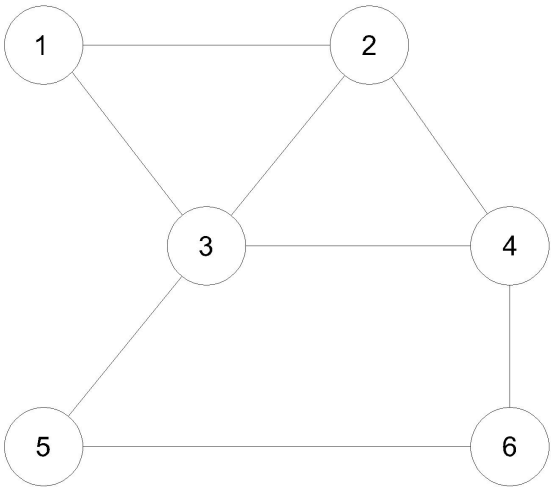

Figura A.1: Exemple de graf no dirigit.

En aquest cas la definició de graf dirigit i no dirigit és una mica diferent:

Graf no dirigit: Un graf $G = (V, A)$ amb funció de cost c associada és no dirigit si:

$$c(v_i, v_j) = c(v_j, v_i) \quad \forall i, j$$

Graf dirigit: Un graf $G = (V, A)$ amb funció de cost c associada és dirigit si:

$$\exists i, j \quad c(v_i, v_j) \neq c(v_j, v_i)$$

Xarxa de fluxos: Una xarxa de fluxos és un graf $G = (V, A)$ dirigit i amb funció de cost associada. A més de la funció de cost, una xarxa de fluxos també té associades:

- Una assignació de fluxos a les arestes: $x : VxV \longrightarrow \mathbb{R}$ que representa la quantitat de flux que circula per l'aresta.
- Una funció de capacitat $u : VxV \longrightarrow \mathbb{R}$ que determina la quanti-

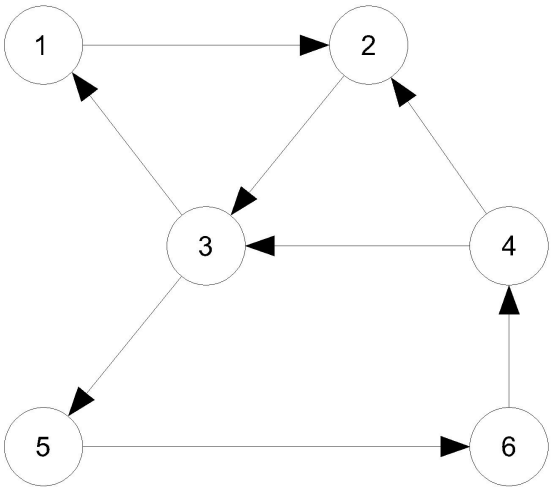

Figura A.2: Exemple de graf dirigit.

tat màxima de flux que pot circular per l'aresta.

- Una funció de demanda en els nodes: $b : V \longrightarrow \mathbb{R}$ que representa la quantitat de flux que entra o surt de la xarxa en cada node.

Camí: Un camí p de longitud r en un graf $G = (V, A)$ és una successió de nodes $p = \{v_1^p, v_2^p, \ldots, v_r^p\}$ tal que:

$$(v_i^p) \in V \quad \forall\, 1 \leq i \leq r$$
$$(v_i^p, v_{i+1}^p) \in A \quad \forall\, 1 \leq i \leq r-1$$

Cost d'un camí: El cost d'un camí p en un graf $G = (V, A)$ amb funció de cost c és:

$$c(p) = \sum_{i=1}^{r-1} c(v_i, v_{i+1})$$

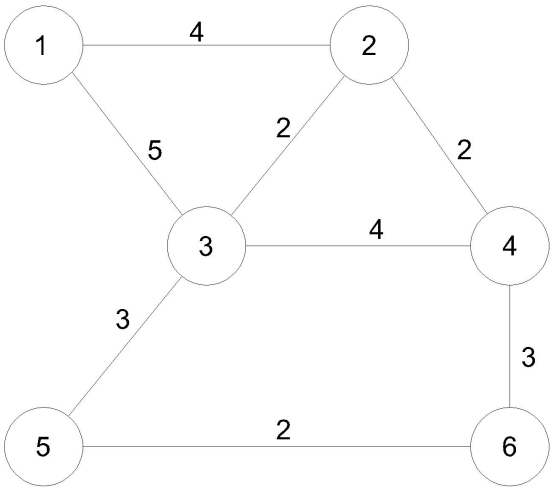

Figura A.3: Exemple de graf amb cost associat no dirigit.

Camí més curt: Es diu que p és el camí més curt de s a t si i només si:

$$p = \{s, v_2^p, \ldots, v_{r-2}^p, t\}$$

i

$$c(p) \leq c(q) \quad \forall\, q = \{s, v_2^q, \ldots, v_{r-2}^q, t\} \ \wedge \ q \neq p$$

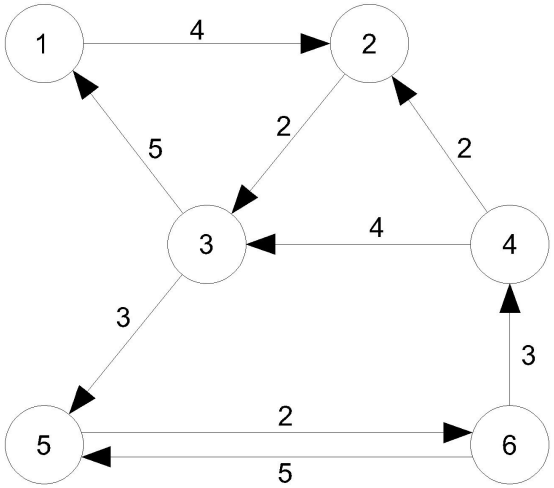

Figura A.4: Exemple de graf amb cost associat dirigit.

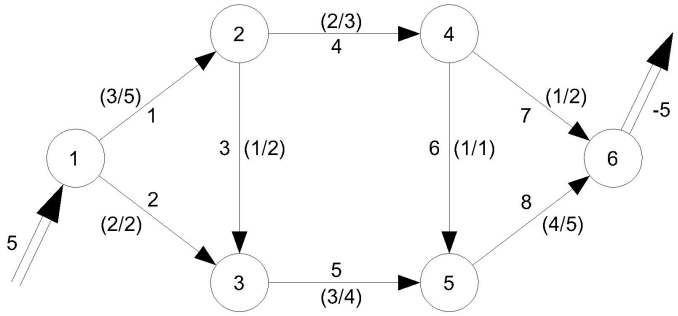

Figura A.5: Exemple de xarxa de fluxos. Els nombres entre parèntesis al costat de cada aresta a_i representen $(x(a_i)/u(a_i))$. Es pot observar també que $b(1) = 5$ i $b(6) = -5$, per la resta d'arestes a_i, $b(a_i) = 0$

Apèndix B

Manual d'usuari de l'aplicació

Per a poder executar l'aplicació desenvolupada en aquest projecte cal tenir correctament instal·lats un entorn d'execució Java — *Java Runtime Environment* — i el programa R, ambdós de lliure distribució. Cal que tant la instrucció *java* com la instrucció *R* estiguin al *path* del sistema operatiu.

Una vegada correctament instal·lat l'entorn només cal copiar els fitxers *PFC.jar* i *gradient_reduit.R* en un directori local. Per engegar el programa cal executar la instrucció *java -jar PFC.jar ruta k tipus_entrada*. Els tres paràmetres són obligatoris:

- ruta: Aquest paràmetre indica la ruta completa — acabada en / o \ — cap al directori a on l'aplicació trobarà les dades d'entrada. El programa espera trobar o bé dos fitxers en format DAT o bé tres fitxers en format CSV.

- **k**: Aquest és el paràmetre k que determina quants camins es tindran en compte en fer la identificació prèvia de passos.

- **tipus_entrada**: Aquest paràmetre determina si el programa llegirà l'entrada en format DAT o en format CSV. Si $tipus_entrada = 1$ llavors

l'aplicació espera trobar dos fitxers DAT en el directori. Aquests fitxers tenen un format definit pel client i han de tenir un nom determinat.

- **SxF.dat**: Aquest fitxer conté l'estructura de la xarxa.
- **SxFmat1.dat**: Aquest fitxer conté la informació de demanda entre parells origen-destí.

Si $tipus_entrada = 0$ llavors l'aplicació espera trobar una sèrie de fitxers CSV, generats per la mateixa aplicació en execucions anteriors o per l'usuari. Aquests fitxers han de tenir un nom determinat.

- **nodes.csv**: Aquest fitxer conté informació sobre els nodes de la xarxa.
- **links.csv**: Aquest fitxer conté informació sobre les arestes de la xarxa.
- **travel.csv**: Aquest fitxer conté la demanda de cada parell origen-destí.

L'aplicació retorna el resultat en format CSV, creant una sèrie de fitxers al mateix directori d'entrada. A més dels fitxers *nodes.csv*, *links.csv* i *travel.csv* que es generen si no existien prèviament, també es generen:

- **D.csv**: Aquesta és la matriu **D** que s'utilitza en la resolució d'equilibri, veure *Capítol 5*.
- **paths.csv**: Conté una taula amb tots els camins identificats l'aplicació.
- **solution.csv**: Conté els valors de cada variable en el punt d'equilibri calculat per l'aplicació.
- **report.csv**: Conté una taula amb alguns valors importants de l'algorisme en cada iteració — entre aquests valors es troba el de la funció objectiu.