UNIVERSITAT DE BARCELONA

Facultat de Matemàtiques
i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

Algorismes de flux màxim en xarxes de flux

Autor: Albert Bigas Comas

Director: Antoni Benseny Ardiaca
Departament de Matemàtiques i Informàtica
Barcelona, 24 de gener de 2021

Abstract

The aim of this project is to study the maximum flow problem in a flow network. It consists of finding the maximum flow capacity that any given flow network can hold, from a source node to a sink node. We will use the generic preflow-push algorithm, the FIFO preflow-push algorithm and the highest-label preflow-push algorithm. Furthermore, we will use an adaptation of the generic preflow-push specific for bipartite networks that will be used in a real life situation such as the Baseball Elimination. Finally, we would be able to analyse the direct implementation of the algorithms as well as examining the results in a simulator.

Resum

El tema principal d'aquest treball és estudiar el problema de flux màxim en una xarxa de flux, que consisteix en trobar la quantitat màxima de flux que pot passar per una xarxa de flux, des d'un node font fins un node pou. Farem servir l'algorisme genèric preflow-push, l'algorisme FIFO preflow-push i l'algorisme highest-label preflow push. A més d'una adaptació del genèric preflow-push per xarxes de flux bipartides, que ens servirà per veure una aplicació directa a la vida real com és el Baseball Elimination. Finalment, podrem veure la implementació directe dels algorismes i visualitzar els resultats amb un simulador.

2010 Mathematics Subject Classification. 68R10, 90C35, 94C15

Agraïments

Primer de tot m'agradaria donar les gràcies a l'Antoni Benseny Ardiaca per la seva dedicació i la seva ajuda constant al llarg de tot el treball. També m'agradaria agrair a la meva família per permetre que hagi pogut arribar fins aquí. I als meus amics per tot el seu suport.

Índex

Introducció

La teoria de grafs és una branca de les matemàtiques i la informàtica que es dedica a l'estudi dels grafs, definits com un conjunt de vèrtexs units per unes arestes, els quals són un dels principals objectes d'estudi de la matemàtica discreta.

Es creu que el seu origen prové del famós problema *Els set ponts de Königsberg* sorgit l'any 1736 a la ciutat russa Königsberg (actualment anomenada Kaliningrad), però que fou alemanya fins a la fi de la II Guerra Mundial. A la ciutat hi passava el riu Pregel que envoltava dues grans illes a les quals es podia accedir mitjançant un total de set ponts. El problema plantejava si era possible creuar-los tots sense repetir-ne cap i tornar al mateix punt de sortida.

A la Figura [1] podem apreciar l'abstracció que va fer servir Leonhard Euler per demostrar que el problema no tenia solució. Concretament va substituir les illes i les riberes per vèrtexs i els ponts per arestes que els enllaçaven. El resultat final conforma l'estructura que anomenem graf i, per aquest motiu, es creu que l'origen prové d'aquí.

Figura 1

A partir d'aquest primer problema, la teoria de grafs ha evolucionat molt i, gràcies a això, altres branques de les matemàtiques, com la topologia i el càlcul diferencial, també. A més a més, juga un paper vital per a diversos problemes de modelatge en el món real com ara en viatges, transport, control del trànsit, comunicacions, aplicacions informàtiques...

Quan les arestes d'un graf són dirigides i tenen una capacitat associada, podem dir que sorgeix el concepte de xarxa. Hi ha moltes xarxes per les quals circula flux, i en elles es plantegen diversos problemes com podria ser el problema de flux màxim, un dels més estudiats en aquest àmbit.

Però no va ser fins a l'any 1947, durant la Guerra Freda, que en va aparèixer el primer. El problema consistia en saber quin era el mínim nombre d'estacions del sistema ferroviari que s'havien de bombardejar per tal que cap tren es pogués desplaçar entre la Unió Soviètiva i qualsevol país de l'est d'Europa.

No obstant, no es va trobar un mètode per resoldre el problema fins al 1956, que Lester Randolph Ford i Delbert Ray Fulkerson van desenvolupar l'algorisme de Ford-Fulkerson per trobar el flux màxim. Un algorisme que es basa en buscar camins on es pugui augmentar el flux fins que aconsegueixi el flux màxim.

Des de llavors, els mètodes han evolucionat molt fent servir noves tècniques de disseny que han conduït a una disminució del temps d'execució. Una d'elles seria la que fa servir el concepte preflow, referit a l'empenta de flux a través de la xarxa, i que Alexander Viktorovich Karzanov l'any 1974 va ser el primer que en va parlar. Gràcies a això, Andrew Vladislav Goldberg i Robert Tarjan van desenvolupar l'algorisme preflow-push, que és tal i com es coneix avui en dia.

Actualment es coneixen diverses aplicacions al món real on poder aplicar aquests algorismes, com podrien ser problemes per trobar els fluxos màxims de corrent en un circuit elèctric; d'aigua, gas o petroli en una xarxa de canonades; de trànsit en una xarxa viària; de productes en un sistema de línia de producció; de persones en un sistema de transport públic...

Motivació

Al llarg del grau només hem cursat una assignatura de grafs, fet que m'ha motivat a realitzar un treball que tingués una base de teoria de grafs. A més a més, volia combinar-ho amb una part de programació ja que, a part de considerar-lo un àmbit molt interessant, també és una eina molt útil per diferents blocs a la vida real.

Estructura de la memòria

Aquesta memòria consta de 4 capítols diferents. El primer d'ells, conté unes definicions necessàries per tal de comprendre el cos del treball.

El segon capítol, conté la descripció dels algorismes utilitzats per resoldre el problema de flux màxim, juntament amb uns exemples per veure'n la resolució pas a pas.

Al tercer hi trobem una aplicació directa d'un dels algorismes, concretament el problema d'eliminació del beisbol.

I finalment, l'últim conté l'explicació detallada de la implementació dels algorismes i el funcionament del simulador creat per visualitzar els resultats.

Capítol 1

Preliminars

En aquest treball resolem un problema de flux màxim en xarxes de flux mitjançant uns algorismes concrets. Per tal d'entendre'ls bé, prèviament hem d'introduir unes nocions bàsiques sobre teoria de grafs i presentar-ne la notació.

Per començar, és necessari saber què és un dígraf.

Definició 1.0.1. Un graf dirigit o dígraf $G = (N, A)$ és un parell ordenat on

1) N és un conjunt no buit de nodes.

2) $A \subset N\text{x}N$ és un conjunt d'arcs.

Un element $a = (u, v) \in A$ és un arc que va des del node u fins al node v on $u, v \in N$. Anomenem D_u al conjunt de nodes adjacents a $u \in N$, que són tots aquells nodes enllaçats a u per un arc. I, en funció de si l'arc surt del node u (arc de sortida) o l'arc entra al node u (arc d'entrada), podem definir els dos subconjunts de nodes adjacents des de u i cap a u de la manera següent, respectivament:

$$D_u^+ = \{v \in N | (u, v) \in A\},$$
$$D_u^- = \{v \in N | (v, u) \in A\}.$$

Un cop definit què és un dígraf, podem passar a definir el concepte de xarxa de flux.

Definició 1.0.2. Una xarxa de flux $NF = (G, c, f, s, t)$ és un graf dirigit $G = (N, A)$ on cada arc $a \in A$ té associats uns valors de capacitat i de flux i del conjunt de nodes N, en distingim dos: l's anomenat font, que és el node per on comença a sortir el flux i pel qual no n'hi entra; i el t anomenat pou, que és el node que l'hi arriba el flux però que no n'hi surt. Definim la funció capacitat dels arcs d'una xarxa de flux NF com una aplicació $c : A \to R^+$. I la funció flux com l'aplicació $f : A \to R^+$ que compleix les dues propietats següents:

i) Condició de viabilitat: Requereix que cada arc porti una quantitat de flux no negativa que mai pot superar la seva capacitat, és a dir,

$$0 \leq f(a) \leq c(a) \quad \forall a \in A.$$

ii) Conservació de flux: Requereix que, per a qualsevol node $u \in N$ diferent dels nodes font i pou, el flux que hi entra ha de ser el mateix que el que hi surt, és a dir,

$$\sum_{v \in D_u^+} f(u,v) = \sum_{v \in D_u^-} f(v,u).$$

És intuïtiu que el flux total que surt des del node font ha de ser el mateix que el flux total que arriba al node pou. A continuació proporcionem una prova formal:

Lema 1.0.3. *Sigui una xarxa de flux $NF = (G, c, f, s, t)$. Aleshores* $\sum_{v \in D_s^+} f(s,v) - \sum_{v \in D_s^-} f(v,s) = \sum_{v \in D_t^-} f(v,t) - \sum_{v \in D_t^+} f(t,v)$.

Demostració: $\sum_{v \in D_s^+} f(s,v) + \sum_{v \in D_t^+} f(t,v) + \sum_{u \neq s,t} \sum_{v \in D_u^+} f(u,v) = \sum_{u,v \in N, u \neq v} f(u,v) = \sum_{v \in D_s^-} f(v,s) + \sum_{v \in D_t^-} f(v,t) + \sum_{u \neq s,t} \sum_{v \in D_u^-} f(v,u)$. □

Vist això, com hem dit prèviament a la introducció, les xarxes de fluxos tenen moltes aplicacions a la vida real i una d'elles seria el problema d'eliminació al beisbol, al qual dedicarem un espai en aquest treball. En aquest cas, necessitem conèixer el concepte de xarxa de flux bipartida.

Definició 1.0.4. Una xarxa de flux $NF = (G, c, f, s, t)$ és bipartida si el conjunt de nodes N del graf dirigit G es pot partir en 2 subconjunts N_1 i N_2 tal que per a cada arc $a = (u, v) \in A$ es compleix que $u \in N_1$ i $v \in N_2$, o bé que $u \in N_2$ i $v \in N_1$. Solem representar la xarxa bipartida usant la notació $G = (N_1 \cup N_2, A)$ on $n_1 = |N_1|$ i $n_2 = |N_2|$.

Capítol 2

Algorismes de flux màxim

Al llarg de la història s'han estudiat diversos algorismes per resoldre el problema de flux màxim en una xarxa de flux. En aquest problema ens trobem una xarxa on cada arc té associat una capacitat i es desitja enviar el màxim de flux possible entre dos nodes especials, el node font i el node pou, sense superar en cap cas la capacitat dels arcs.

Els algorismes que analitzem al llarg del treball són algorismes de tipus preflow-push, on la seva idea principal és empènyer el flux des de vèrtexs amb excés de flux cap al node pou t, utilitzant camins que no són necessàriament els camins més curts d's a t, sinó només estimacions actuals per a aquests camins. Per descomptat, es pot produir que l'excés de flux no es pugui impulsar cap endavant des d'un vèrtex v; en aquest cas, s'ha d'enviar de nou a la font per un camí adequat. L'elecció de tots aquests camins està controlada per una determinada funció d'etiquetatge al conjunt de vèrtexs. Aquesta estratègia permet a aquests algorismes obtenir una velocitat que els fa ser un dels algorismes més eficients del problema de flux màxim.

Durant el procés d'aquests algorismes, s'utilitza el concepte "preflow" que traduirem com a "preflux", on la condició de conservació de flux a la xarxa de flux definida prèviament no es complirà, és a dir, els nodes no necessàriament han de conservar el flux durant el procés, per tant la quantitat de flux que entra en un node pot ser més gran que la quantitat de flux que en surt. Aquesta propietat es manté al llarg de tot l'algorisme; només és al final que el preflux es converteix en flux, i aleshores ja és màxim.

Per realitzar els algorismes, partim d'una xarxa de flux NF amb un flux determinat f. Aleshores, podem considerar els subconjunts següents:

- $I_f :=$ Conjunt d'arcs on es pot augmentar el flux.
- $R_f :=$ Conjunt d'arcs on es pot disminuir el flux.

Si $f(a) > 0$ llavors $a \in R_f$, és a dir, podem disminuir el flux de l'arc a. Mentre que si $f(a) < c(a)$ aleshores $a \in I_f$, per tant podem augmentar el flux de l'arc a.

Per tant podem definir el conjunt d'arcs en què es pot augmentar o disminuir el flux com $A_f = I_f \cup R_f$.

En aquest punt, podem definir què és una xarxa residual associada a la xarxa de flux.

Definició 2.0.1. Una xarxa residual $NF(f) = ((N, A_f), r, f, s, t)$ és una xarxa de flux associada a la xarxa de flux $NF = ((N, A), c, f, s, t)$ amb una capacitat residual definida com una funció $r : A_f \to R^+$ que està formada de la manera següent:

- Si l'arc $a = (u, v) \in I_f$, aleshores construïm un arc $a' = (u, v)$ (amb el mateix sentit que l'arc a) amb el valor de la capacitat residual $r(u, v) = c(a) - f(a)$.

- Si l'arc $a = (u, v) \in R_f$, aleshores construïm un arc $a' = (v, u) = -a$ (amb sentit contrari a l'arc a) amb el valor de la capacitat residual $r(v, u) = f(a)$.

A continuació, necessitem definir el concepte de preflux, introduït prèviament quan no es complia la condició de conservació de flux. I a més a més, també definirem el concepte d'excés d'un node que va lligat al mateix context.

Definició 2.0.2. Definim el concepte preflux com la funció $f : A \to \mathbb{R}^+$ que compleix les dues propietats següents:

i) Condició de viabilitat: Requereix que cada arc porti una quantitat de flux no negativa que mai pot superar la seva capacitat, és a dir,

$$0 \leq f(a) \leq c(a) \quad \forall a \in A.$$

ii) No hi ha necessàriament conservació de flux: Per a qualsevol node $u \in N$ amb $u \neq \{s, t\}$,

$$\sum_{v \in D_u^+} f(u, v) - \sum_{v \in D_u^-} f(v, u) \geq 0.$$

Definició 2.0.3. I per a aquests fluxos no conservatius definim l'excés dels nodes com la funció $e : N \to \mathbb{R}$ tal que $e(u) = \sum_{v \in D_u^+} f(u, v) - \sum_{v \in D_u^-} f(v, u)$ on $u \in N$. I en el cas que $e(u) > 0$, direm que el node u està actiu.

Finalment, abans de passar a explicar els algorismes, necessitem definir el concepte d'etiquetatge d'un node, també introduït prèviament i el concepte d'arc admissible.

Definició 2.0.4. Definim la funció d'etiquetatge com l'aplicació $d : N \to N$ que compleix les 3 propietats següents:

i) $d(t) = 0$.

ii) $d(s) = |N|$.

iii) $d(u) \leq d(v) + 1$ per qualsevol arc $a = (u, v) \in A$.

Aquestes etiquetes de distància del node $u \in N$ on $u \neq \{s, t\}$, realment representen la fita inferior de la longitud del camí més curt des del node u fins al node t de la xarxa residual.

Definició 2.0.5. Un arc $a = (u, v) \in A_f$ de la xarxa residual és admissible si $d(u) = d(v) + 1$.

2.1 Algorisme genèric preflow-push

Per aplicar l'algorisme genèric partirem d'una xarxa NF amb flux nul. Així doncs, com que no hi ha cap arc $a = (u, v) \in R_f$, la xarxa residual associada coincideix amb la inicial. Primerament, s'empeny tot el preflux possible des del node font fins a tots els seus nodes adjacents i s'estableixen les etiquetes de distància inicials de tots els nodes, de tal manera que compleixin les propietats esmentades prèviament. Aquest etiquetatge es duu a terme de la manera següent:

1) Etiquetem el node font i el node pou tal com s'havia definit, és a dir, $d(s) = |N|$ i $d(t) = 0$.

2) Etiquetem un node qualsevol $u \in N$ amb $u \neq \{s, t\}$, a partir de les etiquetes de distància dels nodes de D_u^+, és a dir, si tenim l'arc $a = (u, v) \in A_f$ aleshores $d(u) = d(v) + 1$. En cas que aquests no tinguin etiquetes de distància, realitzarem el pas 2 per a aquests nodes, i així successivament fins a tenir-los tots etiquetats.

Un cop establertes totes les etiquetes de distància, se selecciona un node actiu $u \in N$ (cal tenir en compte que, en el primer pas, es converteixen tots els nodes adjacents al node font en nodes actius). Llavors, si la xarxa residual conté un arc admissible $a = (u, v) \in A_f$ on $v \in N$ és qualsevol altre node i es compleix que $r(u, v) > 0$, podem empènyer δ unitats de preflux des de u fins a v on δ ve descrit de la manera següent:

$$\delta = \min\{e(u), r(u, v)\}.$$

Aleshores, el valor d'excés del node u disminueix en δ unitats, mentre que el valor d'excés del node v augmenta en δ unitats.

En canvi, si la xarxa no conté cap arc admissible, reeetiquetem el node u per:

$$d(u) = \min(d(v) + 1)$$

on v ha de ser un node adjacent a u i complir que $r(u, v) > 0$. Finalment, si no trobem cap node actiu, l'algorisme pararà. En aquest punt, com que no hi haurà cap node en excés, el preflux complirà la propietat de conservació del flux i per tant, ja podrem parlar de flux. Aleshores, haurem trobat el flux màxim que es defineix com:

$$F_M = \sum_{v \in D_t^-} f(v, t).$$

Aquest últim resultat ens l'assegura el teorema següent:

Teorema 2.1.1. *L'algorisme de preflow-push ens dóna un flux i aquest és màxim.*

Demostració: La podem trobar a la referència [Hochstättler & Schliep 2010]. □

2.2 Algorisme FIFO preflow-push

Abans de poder descriure aquest algorisme, necessitem introduir el concepte d'"examinació dels nodes". A cada iteració de l'algorisme genèric preflow-push, com hem vist anteriorment, se selecciona un node u que estigui amb excés i a continuació es realitza un impuls de preflux (que pot ser que sigui saturant o que no ho sigui) o es reetiqueta el node.

Definició 2.2.1. Direm que un impuls de δ unitats de preflux d'un node u a un node v és saturant si $\delta = r((u,v))$ i no saturant si $\delta \neq r((u,v))$.

Observació 2.2.2. Un impuls no saturant del node u, aconsegueix que el node u passi d'estar actiu a no estar-ho, és a dir, $e(u) = 0$.

Si l'algorisme genèric realitza doncs un impuls de preflux, llavors el node u podria ser que encara estigués actiu, però no és una condició obligatòria que l'algorisme torni a seleccionar aquest node a la iteració següent.

No obstant, és fàcil incorporar la regla que sempre que l'algorisme seleccioni un node actiu, continuï impulsant preflux des d'aquest node fins que l'excés d'aquest node sigui 0 o fins que l'algorisme reetiqueti el node, en el cas que no hi haguessin arcs admissibles.

Conseqüentment, l'algorisme realitzarà diversos impulsos saturants seguits d'impulsos no saturants o reetiquetatges. I ens referim a aquesta seqüència d'operacions com a "examinació dels nodes".

Per tant, el que introduïm a l'algorisme FIFO preflow-push respecte al genèric és aquesta regla de selecció dels nodes. A continuació, procedim a explicar l'algorisme:

Partim d'una xarxa de flux NF amb preflux nul, empenyem tot el preflux possible des del node font fins a tots els seus nodes adjacents i establim les etiquetes de distància a tots els nodes seguint la mateixa estructura que a l'algorisme genèric.

Llavors, es crea un vector que l'anomenem $LLISTA$ on es posen tots els nodes actius de la xarxa ordenats. Se selecciona el primer node $u \in N$ del vector i es realitza l'impuls de δ unitats de preflux o la reetiquetació del node en funció de si hi ha arcs admissibles o no.

Posteriorment, s'observa si hi ha més nodes actius a la xarxa i, en cas afirmatiu, s'afegeixen a la cua del vector $LLISTA$.

L'algorisme va examinant el primer node del vector fins que aquest es torna inactiu o és reetiquetat; en el primer cas, el node s'elimina del vector, mentre que en l'últim cas, s'afegeix el node examinat a la cua del vector. En tots dos casos, es passa a examinar el node següent.

L'algorisme s'acabarà en el moment que $LLISTA = \emptyset$ ja que no tindrem nodes actius a la xarxa i, per tant, el preflux es convertirà en flux. Haurem trobat el flux màxim F_M que està definit de la mateixa manera que en l'algorisme genèric.

Exemple 2.2.3. Un cop explicat el procediment de l'algorisme, procedim a veure un exemple.

Volem trobar el flux màxim de la xarxa de flux que tenim a la figura [2.1]. En cadascun dels arcs hi ha la informació del preflux i de la capacitat ($preflux[capacitat]$). En primer lloc, s'envia el màxim de preflux des de la font als nodes adjacents, com podem observar a la figura [2.2] i es posa l'etiqueta de distància a tots els nodes.

En aquest punt, les etiquetes de distància són: $d(s) = 5$, $d(1) = 2$, $d(2) = 1$, $d(3) = 1$ i $d(t) = 0$. I el vector dels nodes amb excés és: $LLISTA = \{1, 2\}$ ja que $e(1) = 2 > 0$, $e(2) = 3 > 0$ i $e(3) = 0$.

Aleshores, seleccionem el node 1 que és el primer element del vector i mirem si té cap arc admissible per passar el preflux en excés. N'hi ha 3 de possibles: $\{(1,s),(1,2),(1,3)\}$. I triem l'arc $a = (1,2)$ que és l'únic admissible ja que $d(1) = d(2) + 1$ i $r((1,2)) = 3 \neq 0$,

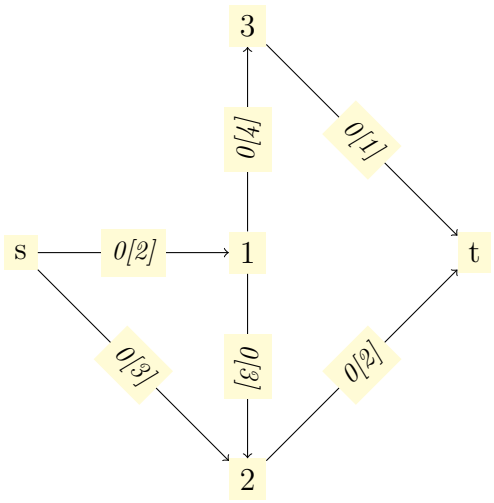

Figura 2.1

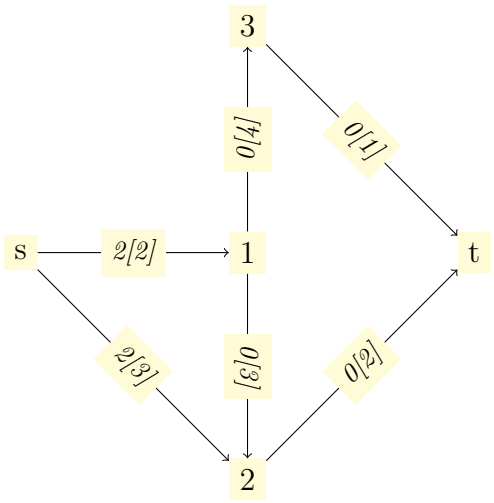

Figura 2.2

mentre que els altres dos no compleixen una de les dues condicions. Per tant, enviem $\delta = \min\{e(1), r((1,2))\} = \min\{2,3\} = 2$ unitats de preflux, com podem observar a la figura [2.3].

Ara tenim que $e(1) = 2 - 2 = 0$ i $e(2) = 3 + 2 = 5$, aleshores podem eliminar el node 1 del vector: $LLISTA = \{2\}$.

En aquest punt, seleccionem el node 2 que té 3 possibles arcs admissibles, però l'únic que compleix les condicions d'arc admissible és l'$a = (2,t)$ ja que $d(2) \neq d(1) + 1$ i $d(2) \neq d(s) + 1$. Així que enviem $\delta = \min\{e(2), r((2,t))\} = \min\{5,2\} = 2$ unitats de preflux, com podem observar a la figura [2.4].

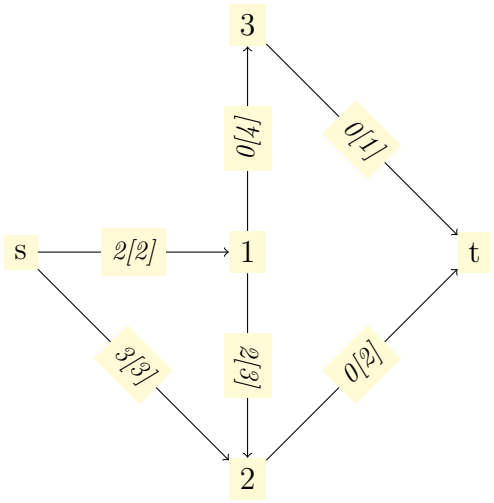

Figura 2.3

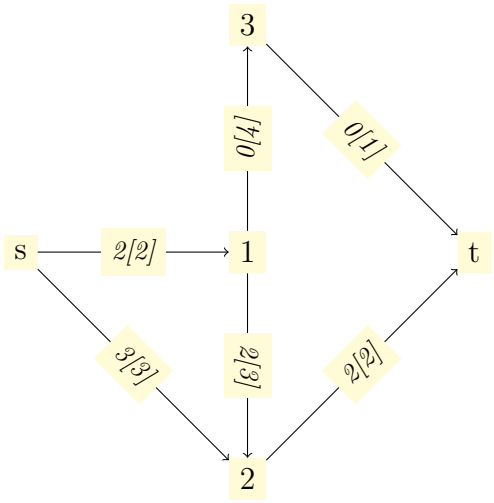

Figura 2.4

En aquest cas $e(2) = 5 - 2 = 3$, mentre que els excessos dels nodes 1 i 3 no s'han modificat, aleshores el vector $LLISTA = \{2\}$ és el mateix. Per tant, tornem a seleccionar el node 2.

Ara no té cap arc admissible, així que el reetiquetem: $d(2) = \min\{d(s)+1, d(1)+1\} = \min\{5+1, 2+1\} = 3$. No tenim en compte el valor de $d(t) + 1$, ja que $r((2,t)) = 0$ i la condició de reetiquetar imposa que $r((u,v)) > 0$ per a qualsevol parella de nodes.

Com que al vector $LLISTA$ només hi ha un element, tornem a seleccionar el node 2 i ara tenim que l'arc $a = (2, 1)$ és admissible ja que $d(2) = d(1) + 1$ i $r((2, 1)) \neq 0$. Aleshores enviem $\delta = \min\{e(2), r((2, 1))\} = \min\{3, 2\} = 2$ unitats de preflux, com podem observar a la figura [2.5].

Si actualitzem els excessos, tenim que $e(1) = 0 + 2 = 2$, $e(2) = 3 - 2 = 1$ i $e(3) = 0$. Per tant, $LLISTA = \{2, 1\}$ i tornem a seleccionar el node 2, però en aquest cas no torna a tenir arcs admissibles, així que el reetiquetem: $d(2) = \min\{d(s) + 1\} = 6$ i per tant posem el node 2 a la cua del vector: $LLISTA = \{1, 2\}$.

A continuació, seleccionem el node 1 i dels tres possibles arcs admissibles, l'únic que compleix les condicions de ser-ho és l'$a = (1, 3)$. Per tant, enviem $\delta = \min\{e(1), r((1, 3))\} = \min\{2, 4\} = 2$ unitats de preflux, com podem observar a la figura [2.6].

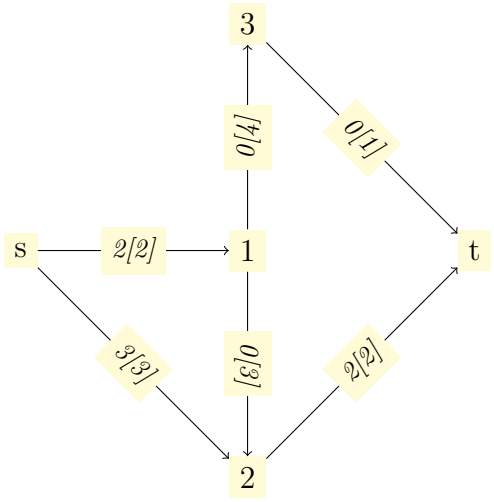

Figura 2.5

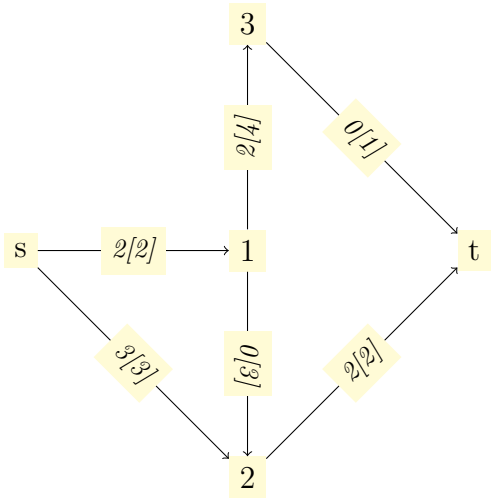

Figura 2.6

Els excessos dels nodes seran $e(1) = 2 - 2 = 0$, $e(2) = 1$ i $e(3) = 0 + 2 = 2$ i en conseqüència, eliminem el node 1 del vector i hi afegim el node 3: $LLISTA = \{2, 3\}$.

Les etiquetes de distància en aquest moment són: $d(s) = 5$, $d(1) = 6$, $d(2) = 2$, $d(3) = 1$ i $d(t) = 0$.

Ara seleccionem el node 2 i l'arc admissible $a = (2, s)$. Aleshores podem empènyer $\delta = \min\{e(2), r((2, s))\} = \min\{1, 3\} = 1$ unitats de preflux, com podem observar a la figura [2.7].

Podem observar que $e(1) = 0$, $e(2) = 1 - 1 = 0$ i $e(3) = 2$, és a dir, el node 2 s'ha tornat inactiu, així que $LLISTA = \{3\}$. Aleshores seleccionem l'únic node del vector, el 3; prenem l'únic arc admissible que té, l'$a = (3, t)$ i enviem $\delta = \min\{e(3), r((3, t))\} = \min\{2, 1\} = 1$ unitats de preflux, com podem observar a la figura [2.8].

Ara $e(3) = 2 - 1 = 1$, així que el tornem a seleccionar però en aquest cas no té arcs admissibles. Per tant reetiquetem el node 3: $d(3) = \min d(1) + 1 = 3$ i el vector $LLISTA = \{3\}$ no ha variat.

Novament prenem el node 3 que ara té l'arc admissible $a = (3, 1)$ ja que $d(3) = d(1)+1$ i $r((3, 1)) = 2 \neq 0$ i empenyem $\delta = \min\{e(3), r((3, 1))\} = \min\{1, 2\} = 1$ unitats de preflux, com podem observar a la figura [2.9].

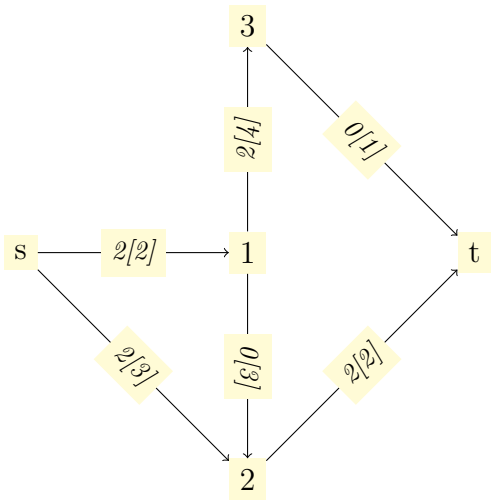

Figura 2.7

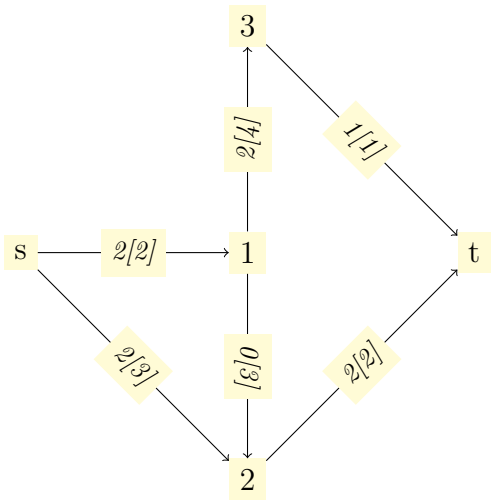

Figura 2.8

Els excessos dels nodes seran $e(1) = 0 + 1 = 1$, $e(2) = 0$ i $e(3) = 1 - 1 = 0$ i en conseqüència, $LLISTA = \{1\}$. Per tant, seleccionem el node 1 que no té cap arc admissible, així que el reetiquetem: $d(1) = \min\{d(s) + 1, d(2) + 1, d(3) + 1\} = \min\{5 + 1, 6 + 1, 3 + 1\} = 4$ i el vector $LLISTA$ no es modifica.

Les etiquetes de distància en aquest moment són: $d(s) = 5$, $d(1) = 6$, $d(2) = 4$, $d(3) = 3$ i $d(t) = 0$.

Prenem el node 1 que ara té com a arc admissible l'arc $a = (1, 3)$, per tant enviem $\delta = \min\{e(1), r((1, 3))\} = \min\{1, 3\} = 1$ unitats de preflux, com podem observar a la figura [2.10].

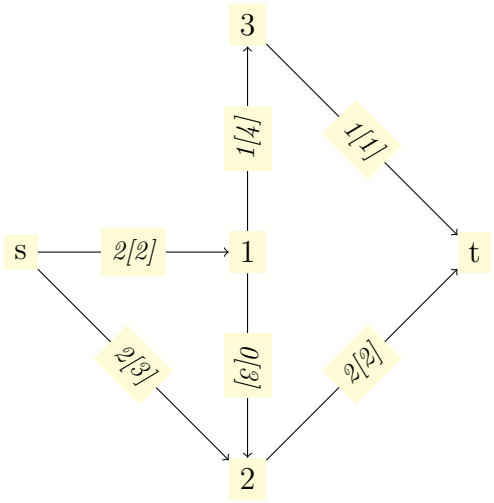

Figura 2.9

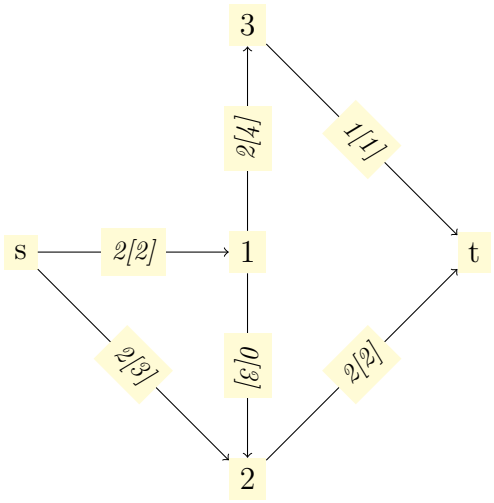

Figura 2.10

Novament modifiquem els excessos: $e(1) = 1 - 1 = 0$, $e(2) = 0$ i $e(3) = 0 + 1 = 1$. Aleshores $LLISTA = \{3\}$ i seleccionem el node 3 que no té arcs admissibles, així que el reetiquetem: $d(3) = \min\{d(1) + 1\} = 5$.

Com que el vector $LLISTA$ no s'ha modificat, tornem a seleccionar el node 3 que ara té l'arc $a = (3, 1)$ que és admissible i enviem $\delta = \min\{e(3), r((3, 1))\} = \min\{1, 2\} = 1$ unitats de preflux, com podem observar a la figura [2.11].

El node 3 ha passat a estar inactiu mentre que el node 1 ha passat a estar actiu amb $e(1) = 1$, així que $LLISTA = \{1\}$ i prenem aquest node que no té arcs admissibles. De manera que el reetiquetem: $d(1) = \min\{d(s)+1, d(2)+1, d(3)+1\} = \min\{5+1, 6+1, 5+1\} = 6$ i el vector $LLISTA$ no es modifica.

Per tant seleccionem novament el node 1, que té l'arc $a = (1, s)$ que és admissible i enviem $\delta = \min\{e(1), r((1, s))\} = \min\{1, 2\} = 1$ unitats de preflux, com podem observar a la figura [2.12].

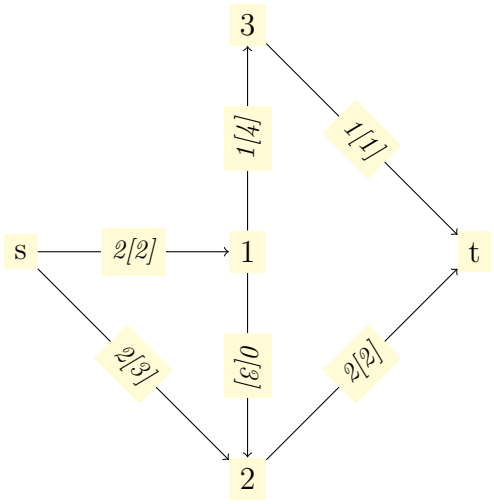

Figura 2.11

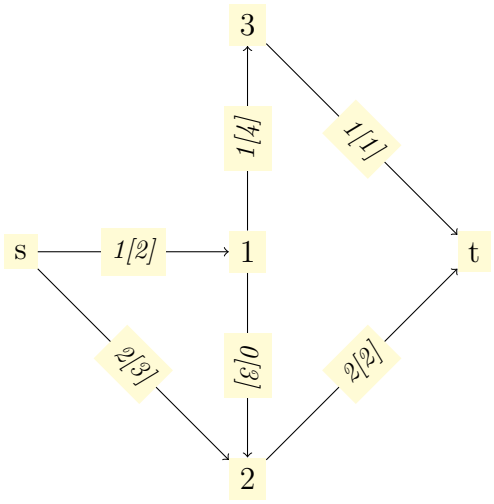

Figura 2.12

En aquest punt, $e(1) = 1 - 1 = 0$, $e(2) = 0$ i $e(3) = 0$, així que $LLISTA = \emptyset$, és a dir, ja no tenim nodes actius i per tant l'algorisme pararà. En aquest punt, el preflux compleix les condicions per ser flux en tots els nodes i podem calcular el flux màxim que en aquest cas és:

$$F_M = \sum_{v \in D_t^-} f(v, t) = 3.$$

2.3 Algorisme highest-label preflow-push

Aquest algorisme manté el concepte introduït d'"examinació dels nodes" de l'algorisme FIFO i, a més a més, sempre empeny preflux des del node actiu amb l'etiqueta de distància més gran.

Per poder explicar-lo, primer definim:

$$h^* = \max\{d(u) | u \text{ és un node actiu}\}.$$

L'algorisme comença de la mateixa manera que ho feien els dos anteriors, és a dir, parteix d'una xarxa NF amb preflux nul, a continuació s'empeny tot el preflux possible des del node font fins a tots els seus nodes adjacents i s'estableixen les etiquetes de distància inicials de tots els nodes.

Llavors a l'hora de seleccionar un node, l'algorisme examina els nodes amb l'etiqueta de distància igual a h^* i empeny el preflux als nodes amb distància igual a $h^* - 1$, i aquests al seu torn, empenyen preflux als nodes amb distància $h^* - 2$ i així successivament fins que l'algorisme reetiqueti un node o tots els nodes estiguin inactius.

En el cas que es reetiqueti un node, l'algorisme repeteix el mateix procés; mentre que en el cas que tots els nodes estiguin inactius, l'algorisme haurà acabat. I per tant haurem trobat el flux màxim F_M.

Aquest algorisme és actualment el mètode més eficient per resoldre el problema de flux màxim a la pràctica perquè realitza el menor nombre d'impulsos no saturants.

Exemple 2.3.1. Per il·lustrar intuïtivament aquest fet, considerem l'exemple següent on farem una comparació entre l'algorisme FIFO preflow-push i l'algorisme highest-label.

Volem trobar el flux màxim F_M de la xarxa de flux mostrada a la figura [2.13].

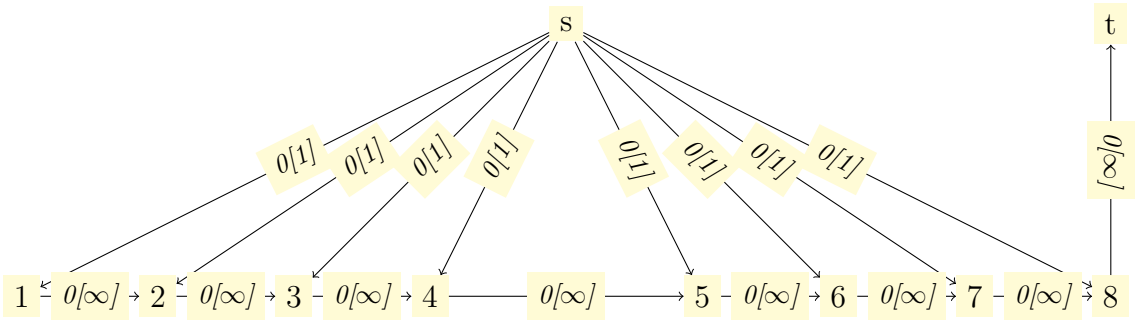

Figura 2.13

Al primer pas, qualssevol dels dos algorismes envia el preflux des del node s fins als seus nodes adjacents i per tant, tots els nodes exceptuant el node font i el node pou estaran actius, és a dir, $e(1) = ... = e(8) = 1 > 0$, com ho podem veure a la figura [2.14].

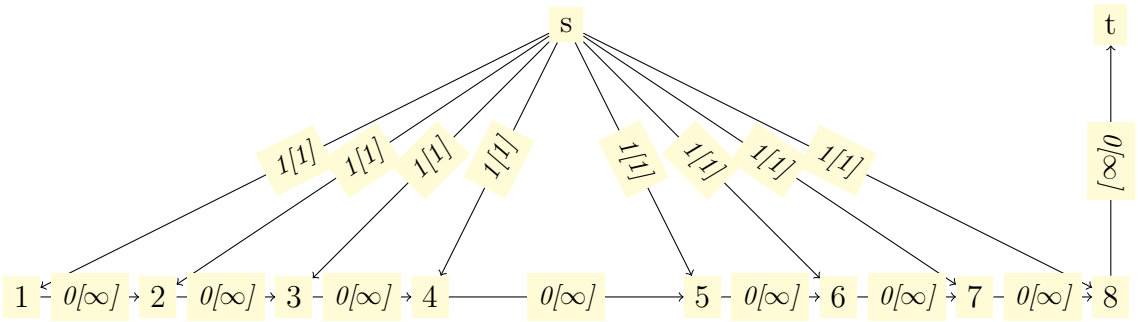

Figura 2.14

En aquest punt tenim les etiquetes de distància següents: $d(s) = 10$, $d(1) = 8$, $d(2) = 7$, $d(3) = 6$, $d(4) = 5$, $d(5) = 4$, $d(6) = 3$, $d(7) = 2$, $d(8) = 1$ i $d(t) = 0$.

A continuació l'algorisme de highest-label examina els nodes $\{1, 2, ..., 7, 8\}$ amb aquest ordre ja que $d(1) = d(2) + 1$, $d(2) = d(3) + 1$, ..., $d(7) = d(8) + 1$ i $d(8) = d(t) + 1$. Per tant, el node 1 empeny tot l'excés al node 2, aquest al node 3 i així successivament fins que el node 8 empeny tot el preflux al node pou. Ho mostra la figura [2.15].

En canvi, l'algorisme FIFO realitza moltes més empentes. Com que al final del primer pas, hem vist que tots els nodes exceptuant dels nodes font i pou estaven en excés, tenim que el vector $LLISTA = \{1, 2, ..., 7, 8\}$. Aleshores examinarem el node 1, seleccionarem l'arc admissible $a = (1, 2)$ i realitzarem l'empenta d'1 unitat de preflux com podem veure a la figura [2.16].

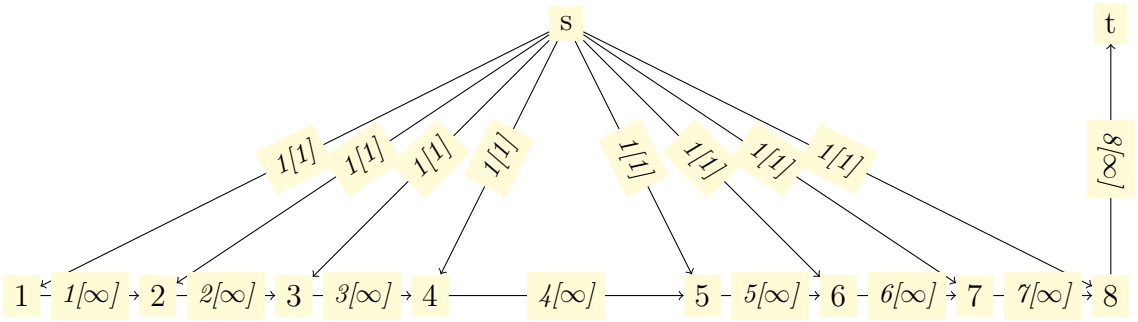

Figura 2.15

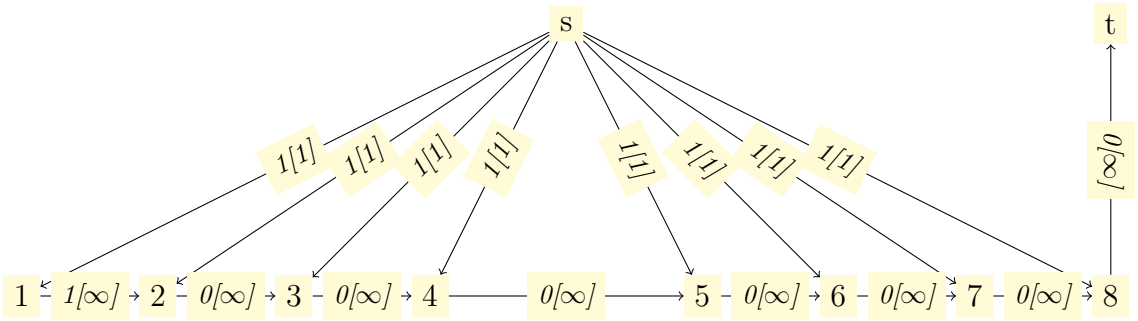

Figura 2.16

En aquest punt, observarem que el node 1 es tornarà inactiu i en conseqüència s'eliminarà del vector $LLISTA$.

Al pas següent, seleccionarem el node 2 que tindrà l'arc admissible $a = (2, 3)$ i per tant es farà una empenta de 2 unitats de preflux com podem veure a la figura [2.17].

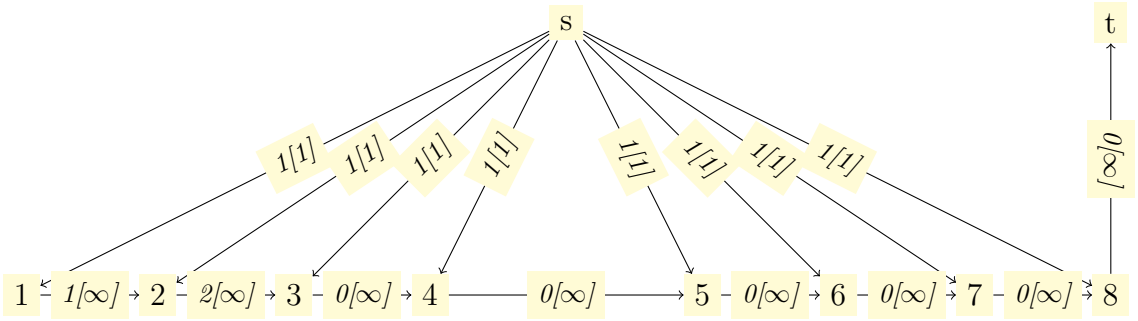

Figura 2.17

I així successivament ho faríem amb tots els nodes del vector $LLISTA$ fins que aconseguíssim que $LLISTA = \emptyset$. En aquest moment obtindríem la mateixa xarxa que a la figura [2.15].

És fàcil comprovar que l'algorisme FIFO haurà realitzat 8 passos, tants com nodes hi havia al vector $LLISTA$, mentre que l'algorisme highest-label només ha necessitat 1 pas.

2.4 Algorisme preflow-push en xarxes bipartides

En aquest apartat descriurem una especialització dels algorismes de tipus preflow-push però adaptats a resoldre problemes de flux màxim en xarxes bipartides.

En el pitjor dels casos, és a dir, quan els subconjunts de nodes N_1 i N_2 són d'una mida comparable, el comportament d'aquests algorismes és similar als algorismes originals; en canvi, si un dels subconjunts és substancialment més gran que l'altre, els algorismes en xarxes bipartides són més ràpids que els originals.

Assumirem que $n_1 \leq n_2$ i que el node $s \in N_2$. En el cas hipotètic que $s \in N_1$, podríem crear un nou node $s\prime \in N_2$ i afegir un nou arc $a = (s\prime, s) \in A$ amb una capacitat suficientment gran.

En aquest treball examinarem només l'algorisme genèric preflow-push per a xarxes bipartides, tot i que les idees que comentarem es poden aplicar de la mateixa manera pels algorismes FIFO i highest-label.

L'algorisme comença de la mateixa manera que els altres, és a dir, parteix d'una xarxa amb flux nul, s'empeny tot el preflux possible des del node font cap a tots els seus nodes adjacents (en el cas que $s \in N_1$, prèviament enviaríem tot el preflux possible des del node $s\prime$ al node s) i s'estableixen les etiquetes de distància inicials a tots els nodes, amb l'única diferència als algorismes anteriors que l'etiqueta de distància del node font serà $d(s) = 2n_1$.

Aquesta modificació construeix la idea següent: per enllaçar els impulsos no saturants de qualsevol algorisme preflow-push, normalment fem servir una funció definida en termes de tots els nodes actius de la xarxa. Si tots els nodes de la xarxa poden estar actius, l'algorisme realitzarà un nombre d'impulsos no saturants. No obstant, si només permetem que estiguin actius els nodes de N_1, com que $n_1 \leq n$ podem obtenir un límit més ajustat al nombre d'impulsos no saturants.

Afortunadament, l'estructura especial de les xarxes bipartides ens permet idear un algorisme que sempre aconsegueix mantenir els nodes d'N_2 inactius. Aconseguim aquest objectiu començant per una solució on els únics nodes actius es trobin a N_1 i realitzant impulsos de longitud 2; és a dir, empenyem el preflux sobre 2 arcs admissibles consecutius perquè qualsevol excés sempre torni a un node de N_1 i cap node de N_2 es faci actiu.

L'empenta serà de $\delta = \min\{e(u), r(u,v), r(v,w)\}$ unitats, on (u,v) i (v,w) són els dos arcs admissibles consecutius.

Com passa a l'algorisme genèric, en el cas que no hi hagi arcs admissibles reetiquetarem el node corresponent, és a dir, si és el primer node el que no té arcs admissibles, doncs serà aquest el que reetiquetem; mentre que si és el segon, reetiqueterem el segon. L'algorisme acabarà quan no hi hagin nodes actius.

En aquest punt, com que no hi haurà cap node en excés, el preflux complirà la propietat de conservació del flux i per tant, ja podrem parlar de flux. Aleshores, haurem trobat el flux màxim $F_M = \sum_{v \in D_t^-} f(v,t)$.

Exemple 2.4.1. Un cop explicat el procediment de l'algorisme, procedim a veure'n un exemple.

Volem trobar el flux màxim de la xarxa de flux que tenim a la figura [2.18]. En primer lloc, s'envia el màxim de preflux des de la font als nodes adjacents, com podem observar a la figura [2.19] i es posa l'etiqueta de distància a tots els nodes.

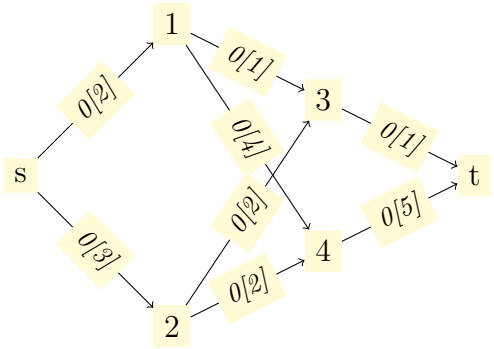

Figura 2.18

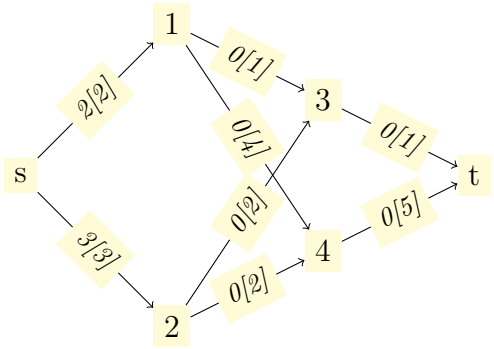

Figura 2.19

En aquest punt tenim que tots els nodes de N_1 estan actius, mentre que els d'N_2 inactius, és a dir, $e(1) = 2 > 0$, $e(2) = 3 > 0$, $e(3) = 0$ i $e(4) = 0$. I les etiquetes de distància són: $d(s) = 6$, $d(1) = 2$, $d(2) = 2$, $d(3) = 1$, $d(4) = 1$ i $d(t) = 0$.

Seleccionarem el node 1 que té aquests 2 arcs admissibles: $\{(1,3),(1,4)\}$ ja que $d(1) = d(3)+1 = d(4)+1$, $r((1,3)) > 0$ i $r((1,4)) > 0$. I ens quedarem amb el primer. Llavors el node 3 només té l'arc admissible $(3,t)$. Aleshores enviem $\delta = \min\{e(1), r((1,3)), r((1,4))\} = \min\{2,1,1\} = 1$ unitats de preflux, com podem observar a la figura [2.20].

D'aquesta manera, tenim que $e(1) = 2-1 = 1$ i $e(3) = 3$. Així que tornem a seleccionar el node 1. En aquest cas només té com a arc admissible l'$a = (1,4)$ i el node 4 només té l'arc $(4,t)$ que compleix les condiccions de ser admissible. Aleshores enviem $\delta = \min\{e(1), r((1,4)), r((4,t))\} = \min\{1,4,5\} = 1$ unitats de preflux. Ho podem veure a la figura [2.21].

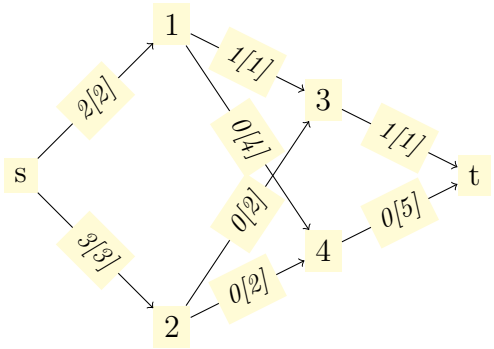

Figura 2.20

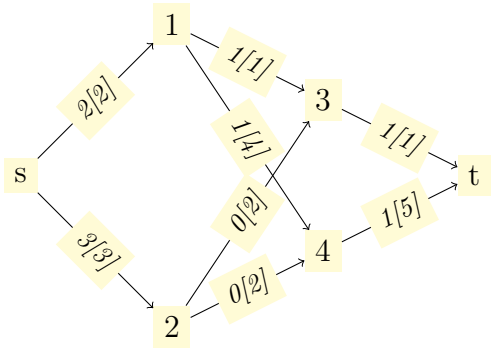

Figura 2.21

Ara tenim que el node 1 ha passat a estar inactiu, mentre que el node 2 segueix actiu amb $e(2) = 3$. Llavors, observem que aquest node té 2 possibles arcs admissibles: $\{(2,3),(2,4)\}$. Seleccionem el primer, però comprovem que el node 3 no té cap arc admissible, així que el reetiquetem: $d(3) = \min\{d(1)+1\} = 3$; no tenim present l'etiqueta de distància dels nodes 2 i t ja que $r((3,2)) = r((3,t)) = 0$.

Continuem tenint només el node 2 actiu, per tant el seleccionem i en aquest cas només té l'arc $(2,4)$ com a admissible. Llavors, el node 4 té l'arc $(4,t)$ que compleix les propietats

d'admissible. Per tant enviem $\delta = \min\{e(2), r((2,4)), r((4,t))\} = \min\{3,2,4\} = 2$ unitats de preflux, com podem observar a la figura [2.22].

Com a conseqüència, $e(2) = 3 - 2 = 1$ i tornem a seleccionar el node 2 que és l'únic que està actiu tot i que ara no té arcs admissibles, així que el reetiquetem: $d(2) = \min\{d(s)+1, d(3)+1\} = \min\{6+1, 3+1\} = 4$.

En aquest punt tenim les següents etiquetes de distància: $d(s) = 6$, $d(1) = 2$, $d(2) = 4$, $d(3) = 3$, $d(4) = 1$ i $d(t) = 0$.

Novament ens quedem amb l'únic node actiu, el 2; que després de reetiquetar-lo ha aconseguit l'arc admissible $(2,3)$ amb $r((2,3)) = 2$ i el node 3 té l'arc $(3,1)$ que també és admissible. Així que enviem $\delta = \min\{e(2), r((2,3)), r((3,1))\} = \min\{1,2,1\} = 1$ unitats de preflux, com podem observar a la figura [2.23].

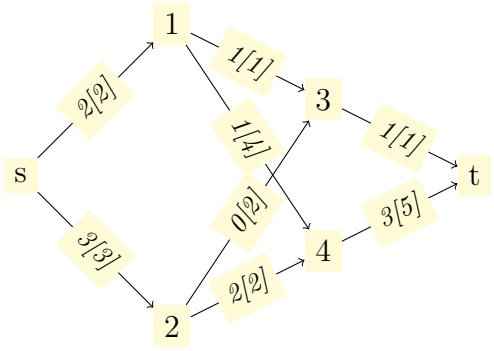

Figura 2.22

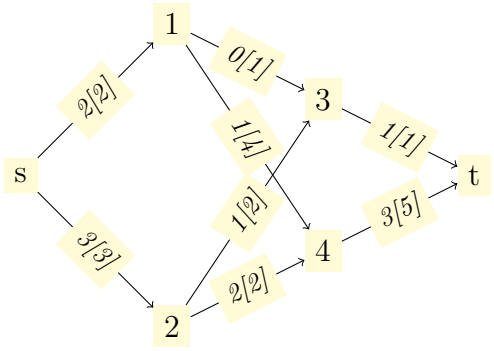

Figura 2.23

Després d'aquesta empenta, tenim que $e(1) = 0 + 1 = 1$ i $e(2) = 1 - 1 = 0$, és a dir, el node 1 ha passat d'estar inactiu a actiu i el node 2 el cas contrari. Aleshores seleccionem el node 1 que té com a únic arc admissible l'arc $(1,4)$ amb $r((1,4)) = 3$ i el node 4 també té només un arc admissible, l'arc $(4,t)$ amb $r((4,t)) = 4$. D'aquesta manera enviem $\delta = \min\{e(1), r((1,4)), r((4,t))\} = \min\{1,3,2\} = 1$ unitats de preflux, com ho mostra la figura [2.24].

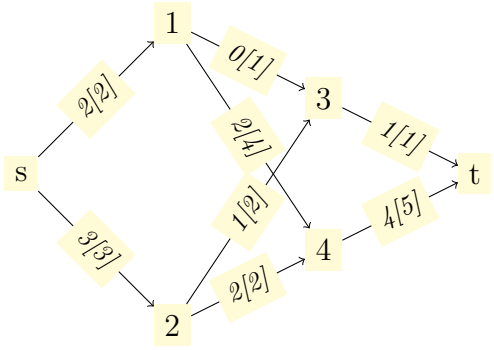

Figura 2.24

Finalment podem comprovar que $e(1) = e(2) = 0$, és a dir, tots els nodes d'N_1 han deixat d'estar actius i per tant l'algorisme pararà. En aquest punt, el preflux compleix les condicions per ser flux en tots els nodes i podem calcular el flux màxim:

$$F_M = \sum_{v \in D_t^-} f(v,t) = 5.$$

Capítol 3

Problema d'eliminació del beisbol

Una de les aplicacions a la vida real dels algorismes que acabem de veure seria el problema d'eliminació del beisbol, que tracta del següent:

En un moment concret de la temporada, cadascun dels $(n + 1)$ equips de la lliga americana, que els enumerem des de l'equip 0 fins l'equip n, han jugat una sèrie de partits. Suposem que l'equip "i" ha guanyat w_i dels partits que ja ha jugat i g_{ij} és el nombre de partits que els equips "i" i "j" encara han de jugar l'un contra l'altre. I a més a més, suposem que els partits no poden acabar en empat.

Llavors, un aficionat ansiós i optimista d'un dels equips vol saber si el seu equip encara té l'oportunitat de guanyar la lliga.

Direm que podem eliminar un equip en concret de la lliga, suposem que és l'equip 0, si per a cada resultat dels partits sense jugar, un equip almenys obtindrà més victòries que l'equip 0. En aquest cas, no serà el campió.

Deixem que w_{max} denoti w_0 més el nombre total de partits que encara ha de jugar l'equip 0, que en el millor de tots els casos serà el màxim nombre de victòries que pot aconseguir l'equip 0.

Aleshores, podem transformar aquest problema en un problema de flux màxim en una xarxa bipartida de la manera següent:

Definim el conjunt de tots els equips de la lliga excepte l'equip 0 com $S = \{1, ..., n\}$ i el dígraf $G = (N_1 \cup N_2, A)$ amb $N_1 = \{m_{xy} \,|\, x, y \in S,\, x \neq y\} \cup \{t\}$ i $N_2 = \{s\} \cup S$ on m_{xy} representa un partit entre l'equip "x" i l'equip "y", s el node font i t el node pou.

Llavors, cada node "partit" m_{xy} té un arc d'entrada $a = (s, m_{xy}) \in A$ amb una capacitat de g_{xy} ja que els partits que juguen entre els equips "x" i "y" tenen un total de g_{xy} victòries per assignar.

A més a més, cada node m_{xy} té dos arcs de sortida $(m_{xy}, x) \in A$ i $(m_{xy}, y) \in A$ on els fluxos d'aquests arcs representen al nombre de victòries de l'equip "x" i de l'equip "y" respectivament, entre els partits g_{xy} addicionals que aquests dos equips encara han de jugar l'un contra l'altre, i els assignem una capacitat d'∞.

I per últim, cada node "equip" $i \in S$, té un arc de sortida $a = (i, t) \in A$ amb un flux x_{it} que representa al nombre total de partits addicionals que guanya l'equip "i" i una capacitat d'$(w_{max} - w_i)$, que garanteix que l'equip "i" no pugui rebre prou flux dels nodes de "partits" per superar l'equip 0 en el total de victòries ja que si l'equip 0 vol guanyar

la lliga, necessitem una manera d'assignar totes les victòries dels altres partits que no impliquin l'equip 0, de manera que el total de victòries de qualsevol equip no superi w_{max}. És a dir,

$$w_{max} \geq w_i + x_{it} \ \forall i \in S.$$

Que ho podem reescriure com:

$$x_{it} \leq w_{max} - w_i \ \forall i \in S.$$

El teorema següent ens assegura quan l'equip escollit pot o no guanyar la lliga després d'haver trobat el flux màxim a la xarxa de flux:

Teorema 3.0.1. *L'equip* 0 *pot acabar la temporada en* 1*a posició* $\Leftrightarrow$ *hi ha un flux factible al dígraf, és a dir, que satisfà les condicions de viabilitat i de conservació de flux, que satura totes les arestes* $(s, m_{xy}) \in A$.

Demostració:
$\Rightarrow$) Suposem que l'equip 0 pot acabar la temporada en $1a$ posició, aleshores cada equip $i \in S$ guanya com a molt $(w_0 + g_{0j} - w_i)$ dels partits que queden, $\forall j \in S$.

Llavors per cada partit entre els equips "i" i "j" que guanya l'equip "i", afegim una unitat de flux al llarg del camí $s \to m_{ij} \to i \to t$.

Com que hi ha exactament g_{ij} partits entre els equips "i" i "j", tots els arcs $(s, m_{ij}) \in A$ estan saturats. I com que cada equip $i \in S$ guanya com a molt $(w_0 + g_{0j} - w_i)$ dels partits que queden $(\forall j \in S)$, el flux resultant és factible ja que $0 \leq f(a) \leq c(a)$, $\forall a \in A$.

$\Leftarrow$) Suposem ara que tenim un flux factible f que satura tots els arcs $a = (s, m_{xy}) \in A$, és a dir, $f(a) = c(a)$. I també que l'equip "i" guanya exactament $f(m_{ij}, i)$ partits contra l'equip "j", $\forall i, j \in S$.

Aleshores tenim que els equips "i" i "j" juguen $f(m_{ij}, i) + f(m_{ij}, j) = f(s, m_{ij}) = g_{ij}$ partits, de manera que podem assegurar que es juguen tots els partits restants.

A més a més, cada equip $i \in S$ guanya un total de $\sum_{j \in S} f(m_{ij}, i) = f(i, t) \leq (w_0 + g_{0j} - w_i)$ dels propers partits $(\forall j \in S)$, on a la desigualtat hem fet servir la hipòtesi que tenim un flux factible. I per tant, si arreglem la desigualtat, tenim que:

$$w_i + \sum_{j \in S} f(m_{ij}, i) \leq w_0 + g_{0j} \ , \forall j \in S.$$

És a dir, si a l'equip "i" li sumem els partits guanyats fins llavors més tots els que li queden per jugar, com a molt serà igual a $w_0 + g_{0j}$. De manera que en podem concloure que si l'equip 0 guanya tots els partits que li queden, acabarà la temporada en $1a$ posició.

□

Exemple 3.0.2. Podem considerar l'exemple de la lliga americana de l'est de Beisbol al 30 d'agost del 1996 formada per 5 equips amb les dades mostrades a la taula [3.1], on hi podem observar els partits guanyats i perduts per cada equip, el total de partits que els hi falten jugar per acabar la lliga i el total de partits que han de jugar uns contra els altres, respectivament.

EQUIP	WIN-LOST	PENDENTS	NYY	BAL	BOS	TOR	DET
New York Yankees	75-59	28	-	3	8	7	3
Baltimore Orioles	71-63	28	3	-	2	7	4
Boston Red Sox	69-66	27	8	2	-	0	0
Toronto Blue Jays	63-72	27	7	7	0	-	0
Detroit Tigers	49-86	27	3	4	0	0	-

Taula 3.1

Suposem que en aquest precís moment de la temporada, els aficionats de qualsevol equip volguessin saber si el seu equip encara té opcions de guanyar la lliga.

Comencem pel cas d'un aficionat de Detroit Tigers. Com hem explicat prèviament, hem de crear una xarxa de flux amb 4 nodes "equips", 6 nodes "partits", el node font, el node pou i els arcs corresponents amb les seves capacitats, com podem visualitzar a la Figura [3.1].

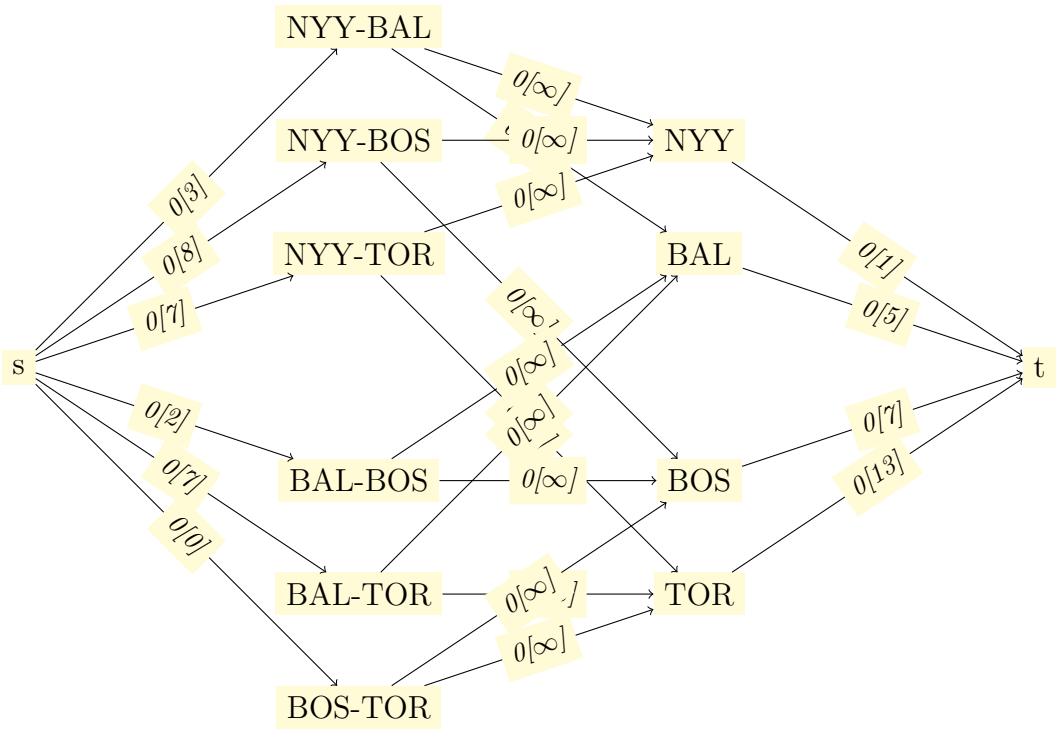

Figura 3.1

És fàcil veure que la capacitat total de les arestes $(s, m_{xy}) \in A$ és (3+8+7+2+7+0) = 27. D'altra banda, la capacitat total de les arestes $(i, t) \in A$ és 26, la qual cosa implica que el flux máxim seria com a màxim 26, per tant al realitzar l'algorisme preflow-push per xarxes bipartides no trobarem cap flux factible que saturi totes les arestes $(s, m_{xy}) \in A$.

Si ara ens fixem en el cas d'un aficionat de Toronto Blue Jays, podem crear la xarxa de flux amb els nodes adients i els arcs amb les seves capacitats corresponents. La veiem a la Figura [3.2].

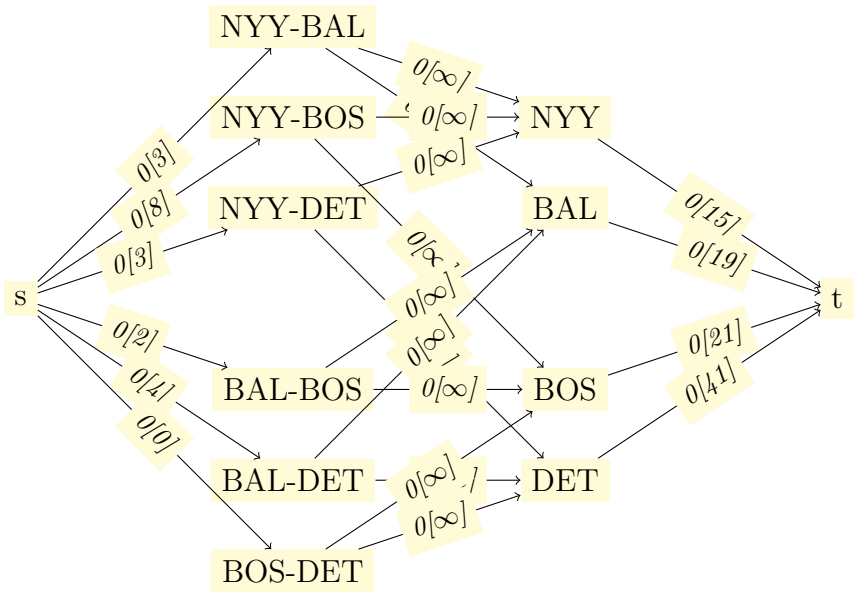

Figura 3.2

Llavors, a l'aplicar l'algorisme genèric preflow-push per a xarxes bipartides obtenim la xarxa de la Figura [3.3].

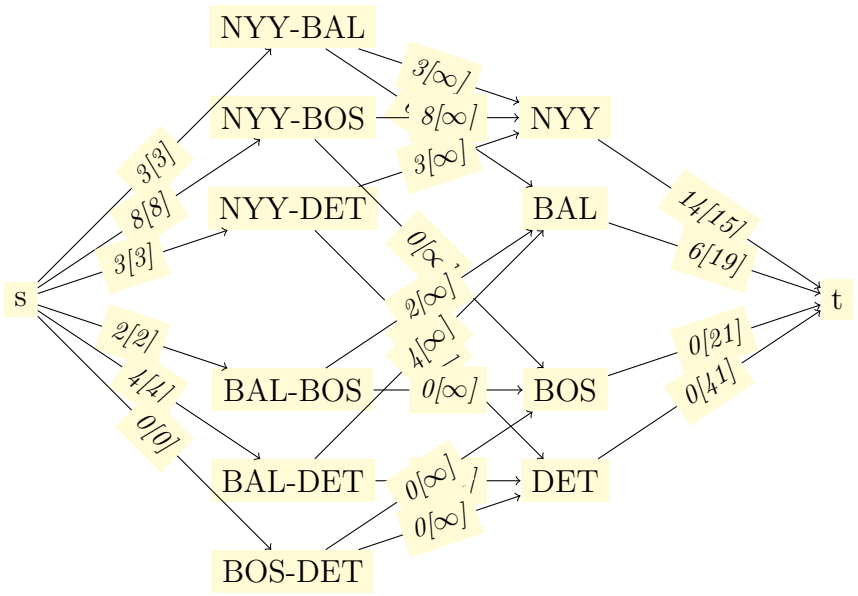

Figura 3.3

Podem comprovar que hem trobat un flux factible que satura les arestes $(s, m_{xy}) \in A$, d'aquesta manera pel Teorema [3.0.1] podem assegurar que els Toronto Blue Jays encara tenen opcions de guanyar la lliga.

Si féssim el mateix pels equips de Boston Red Sox i Baltimore Orioles, és fàcil assegurar que també trobaríem un flux factible que saturés les arestes que surten del node font. I com a conseqüència encara tenen opcions de quedar primers.

Capítol 4

Implementació

Els algorismes treballats al llarg del treball s'han dut a terme amb una aplicació informàtica anomenada simulador mitjançant el programa **Visual Studio**.

Es tracta d'un programa central que té diversos mòduls ***.cpp** on a cada un d'ells es realitzen diferents tasques. L'objectiu d'aquest apartat és explicar el funcionament de cada un i com interactuen entre ells per tal d'arribar al resultat desitjat.

4.1 Entrada i sortida d'informació

El primer mòdul que tractem és l'**NF_inout.cpp**. És l'encarregat de llegir la informació, guardar-la en memòria i fer-ne la sortida.

Per als algorismes utilitzats necessitem les dades d'una xarxa de flux o bé les d'una xarxa de flux bipartida. El primer cas l'obtenim llegint les dades d'uns fitxers anomenats **NF*.dat**. Quan el simulador rep aquestes dades té les opcions d'aplicar els algorismes genèric i FIFO preflow-push. Mentre que si es llegeixen les dades d'uns fitxers anomenats **BPNF*.dat** obtindrem una xarxa de flux bipartida. En el cas que el simulador rebi aquestes, podrà aplicar l'algorisme preflow-push en xarxes bipartides. El símbol * representa un nombre que va canviant en funció de l'arxiu que vulguem usar.

Tant els arxius **NF*.dat** com els arxius **BPNF*.dat** tenen l'estructura següent:

1) A la primera línia trobem 2 valors, el de **vn** i el d'**an** que representen al nombre total de nodes i al nombre total d'arcs de les xarxes de flux, respectivament.

2) A les **vn** línies següents tenim la informació de cada un dels nodes. En cada línia trobem 3 valors: el primer correspon a l'identificador numèric del node i els altres dos als valors posicionals, començant per la columna en què es troba i acabant per la fila.

3) A les **an** línies següents tenim la informació de cada un dels arcs. En cada línia trobem 4 valors: el primer correspon a l'identificador de l'arc, els dos següents al node sortida i al node arribada de l'arc respectivament i, per últim, la capacitat de l'arc.

4) Finalment, a l'última línia trobem 2 valors: els identificadors del node font i del node pou.

Lectura dels fitxers NF*.dat

Un cop hem vist el format dels fitxers, podem passar a veure com guardem la informació llegida en memòria. En aquest cas utilitzem la funció anomenada **NF_read** que usa les variables **vn** i **an** i les funcions següents:

- El dígraf D construït com un **vector<vector<vertex>>** on guardem tots els nodes i per cada un d'aquests nodes v, es crea la seva llista d'adjacència D_v.
- Un vector de punts Np construït com un **vector<point>** on cada **point** és un parell de coordenades.
- Un vector de capacitats NC construït com un **vector<capacity>** on cada **capacity** representa a la capacitat d'un arc.
- Un vector de fluxos NF construït com un **vector<flow>** on cada **flow** representa al flux d'un arc. Inicialment és un vector de zeros.

Els arcs estan guardats a les llistes d'adjacències del dígraf D i s'identifiquen amb un **vip pair<vertex, index>** anomenat NA, on **vip** és una parella d'un vèrtex i un índex associada a un element de les llistes d'adjacències. S'hi identifiquen el doble d'arcs que hi ha a la xarxa de flux inicial per tal de poder crear la xarxa residual corresponent. A continuació observem el procés d'identificació dels arcs:

Si volem guardar l'arc $a = (v, u)$, aleshores busquem la llista d'adjacència del node v i posem el node u com a adjacent del node v incorportant-lo a l'última posició de la llista. I en aquest arc li assignem l'identificador d'arc **a**. Seguidament, busquem la llista d'adjacència del node u i hi afegim el node v a l'última posició. I en aquest arc $a' = (u, v)$ li assignem l'identidicador d'arc **a+an**. Podem veure-ho gràficament a la Figura [4.1].

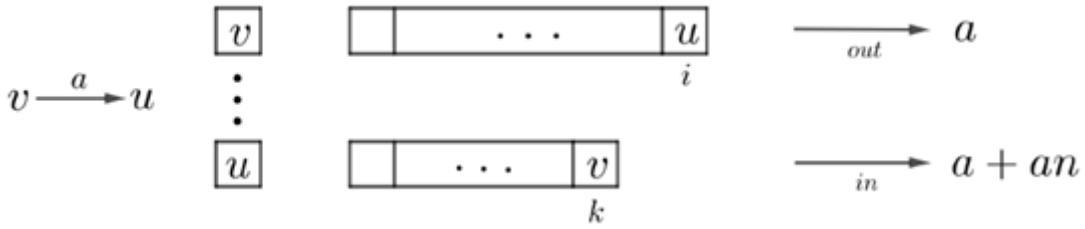

Figura 4.1

Un cop tenim tota aquesta informació, podem afegir la capacitat dels arcs als que tinguin associat un índex **a<an** ja que són els arcs de la xarxa de flux inicial. I a partir d'aquí ja es podria construïr la xarxa residual necessària per poder realitzar els algorismes.

Sortida d'informació

Els arxius **NF*.out** o **BPNF*.out** i els arxius **NF*.tex** o **BPNF*.tex** se centren en la sortida d'informació. En ells hi trobem els diferents passos realitzats pels algorismes, a més del flux màxim.

NF_write és la funció encarregada d'escriure la informació dels arxius **NF*.out** i **BPNF*.out**. En primer lloc, escriu el nombre total de nodes **vn** i el nombre total d'arcs **an**.

A continuació, per a cada un dels nodes, escriu la seva llista d'adjacència i, per a cada arc que tingui identificador **a<an**, ens dona la seva capacitat i el seu flux.

I per últim, escriu el flux màxim F_M de la xarxa de flux.

A la Figura [4.2] en podem veure un exemple.

```
1  Network with 5 vertices and 6 arcs
2  Adjacencies lists
3  0   :       1 (0)[2/0]  2 (1)[3/0]     {0}
4  1   :       0 (6)   2 (2)[3/0]  3 (3)[4/0]     {0}
5  2   :       0 (7)   1 (8)   4 (4)[2/0]     {0}
6  3   :       1 (9)   4 (5)[1/0]     {0}
7  4   :       2 (10)  3 (11)     {0}
8  Arcs
9  0 1 (0) [2/0]
10 0 2 (1) [3/0]
11 1 2 (2) [3/0]
12 1 3 (3) [4/0]
13 2 4 (4) [2/0]
14 3 4 (5) [1/0]
15 FLOW from 0 to 4: 0
```

Figura 4.2

Mentre que **NF_tex** és la funció encarregada d'escriure la informació dels arxius **NF*.tex** i **BPNF*.tex**. Ens proporcionen un diagrama de la xarxa de flux al pla. Sempre es mostraran els arcs amb índex **a<an** amb la seva capacitat i flux corresponent.

Totes les figures del Capítol [2] estan fetes amb aquests dos arxius.

4.2 Algorismes

En aquest apartat explicarem com hem programat els algorismes genèric i FIFO preflow-push per a xarxes de flux i el genèric per a xarxes de flux bipartides que els trobem a l'arxiu **NF_compute.cpp**.

La tria entre els tres algorismes la fa l'usuari a través del simulador com explicarem més endavant.

4.2.1 Algorisme genèric preflow-push

Aquest algorisme parteix de la xarxa residual obtinguda utilitzant les funcions descrites prèviament i al primer pas es fa servir la funció **NF_PreflowInit** que fa el següent:

Envia tot el preflux possible des del node font imposant que $NF[a] = NC[a]$ per als arcs que surten d'aquest node. A continuació, crea dos vectors nous:

- Un vector de distàncies ND construït com un **vector<length>** on cada **length** representa l'etiqueta de distància d'un node.
- Un vector d'excessos NE construït com un **vector<excess>** on cada **excess** representa el flux de cada node que ens servirà per veure si aquest té excés o no.

Llavors utilitza la funció **NF_distances** per etiquetar cada node, tenint en compte que $ND[s] = |N|$, $ND[t] = 0$ i per la resta de nodes es troba el camí més curt de la xarxa residual des del propi node fins al node pou. I finalment s'escriuen els excessos dels nodes.

Un cop fet això, per trobar el flux màxim de la xarxa de flux s'utilitza la funció **NF_PreflowPush**:

A cada pas, es busca un node amb excés **xv**. I llavors, per trobar-li un arc admissible de la xarxa residual on poder-li enviar preflux es fa el següent:

Es busca un node **axv** que pertanyi a la llista d'adjacència del node **xv**, es crea l'arc **a** corresponent i es mira que compleixi una de les dues condicions:

- Si $a < an$, $(NC[a] - NF[a]) > 0$ i $ND[xv] = ND[axv] + 1$, llavors haurem trobat un arc admissible i s'enviaran δ unitats de preflux des de **xv** fins a **axn** on δ està definit com:
$$\delta = \min\{NE[xv], NC[a] - NF[a]\}.$$
- Si $a \geq an$, $NF[a - an] > 0$ i $ND[xv] = ND[axv] + 1$, llavors haurem trobat un arc admissible i s'enviaran δ unitats de preflux des de **xv** fins a **axn** on δ està definit com:
$$\delta = \min\{NE[xv], NF[a - an]\}.$$

En el cas de no complir-se cap de les dues condicions, significarà que no hi ha arcs admissibles i, per tant, es reetiquetarà el node **xv** fent el següent:

Es consideren tots els nodes adjacents **axn** del node **xv**, es crea l'arc **a** corresponent i per a cada un d'ells es fa:

- Si $a < an$, $NF[a] < NC[a]$ i $ND[xv] > ND[axv]+1$, llavors $ND[xv] > ND[axv]+1$.
- Si $a \geq an$, $NF[a-an] > 0$ i $ND[xv] > ND[axv]+1$, llavors $ND[xv] > ND[axv]+1$.

Fet que ens assegura que $ND[xv] = \min\{ND[axv] + 1\}$, $\forall\, axv \in D_{xv}$.

En el moment que no es pugui trobar cap node amb excés, l'algorisme para i el flux màxim és:
$$F_M = NE[t].$$

4.2.2 Algorisme FIFO preflow-push

Aquest algorisme comença d'igual forma que l'algorisme genèric, fent servir la mateixa funció **NF_PreflowInit**. Però, a l'hora de trobar el flux màxim, fem servir la funció **NF_FIFOPreflowPush**:

Primer de tot es creen dos vectors nous:

- Un vector de nodes NV construït com un **vector<vertex>** on cada **vertex** representarà un node actiu de la xarxa de flux.
- Un vector de booleans Nl construït com un **vector<bool>** que tindrà la mida dels nodes totals de la xarxa, i marcarem cada posició com a "true" o "false" en funció de si el node corresponent estarà actiu o no, respectivament.

Llavors, es busquen els nodes actius **xv**, s'afegeixen al vector de nodes i s'escriu "true" al vector de booleans a l'índex corresponent.

A continuació es busca un node **axv** que pertanyi a la llista d'adjacència del primer node **xv** del vector, es crea l'arc **a** i es mira si compleix les mateixes condicions que l'algorisme anterior per trobar un arc admissible. En cas de no complir-se'n cap, es reetiqueta el node també de la mateixa manera.

En aquest punt es mira si han variat els excessos dels nodes. En el cas que algun s'hagi tornat actiu, s'afegeix a la cua del vector de nodes i es canvia "false" per "true" al seu índex del vector de booleans. Mentre que si el node seleccionat s'ha tornat inactiu, s'elimina de la primera posició del vector de nodes, tota la resta avança una posició i es canvia "true" per "false" al seu índex del vector de booleans.

Finalment, en el cas que s'hagi reetiquetat el node **xv**, aquest es trasllada a la cua del vector de nodes i tota la resta avança una posició.

En el moment que el vector de nodes no tingui mida, voldrà dir que no hi ha cap node amb excés. Per tant l'algorisme es para i el flux màxim és:

$$F_M = NE[t].$$

4.2.3 Algorisme genèric preflow-push per xarxes bipartides

Finalment, ens centrem en la programació de l'algorisme per xarxes bipartides que també comença igual que els dos algorismes anteriors, fent servir la funció **NF_PreflowInit**. Però en aquest cas, fem servir la funció **BPNF_PreflowPush** per trobar el flux màxim:

A cada pas, es busca un node amb excés **xv**. Llavors, per trobar-li un arc admissible de la xarxa residual es busca un node **axv** que pertanyi a la llista d'adjacència del node **xv**, es crea l'arc **a** corresponent i es mira que compleixi una de les dues condicions:

- Si $a < an$, $(NC[a] - NF[a]) > 0$ i $ND[xv] = ND[axv] + 1$, llavors haurem trobat un arc admissible i per trobar-li un arc admissible al node **axv**, es busca un node **bxv** que pertanyi a la llista d'adjacència del node **axv**, es crea l'arc **b** corresponent i es mira que compleixi:
  - Si $b < an$, $(NC[b] - NF[b]) > 0$ i $ND[axv] = ND[bxv] + 1$, aleshores haurem trobat el segon arc admissible i s'enviaran δ unitats de preflux des de **xv** fins a **bxn** on δ està definit com:

    $$\delta = \min\{NE[xv], NC[a] - NF[a], NC[b] - NF[b]\}.$$

  - Si $b \geq an$, $NF[b - an] > 0$ i $ND[axv] = ND[bxv] + 1$, aleshores haurem trobat el segon arc admissible i s'enviaran δ unitats de preflux des de **xv** fins a **bxn** on δ està definit com:

    $$\delta = \min\{NE[xv], NC[a] - NF[a], NF[b - an]\}.$$

- Si $a \geq an$, $NF[a - an] > 0$ i $ND[xv] = ND[axv] + 1$, llavors haurem trobat un arc admissible i per trobar-li un arc admissible al node **axv**, es busca un node **bxv** que pertanyi a la llista d'adjacència del node **axv**, es crea l'arc **b** corresponent i es mira que compleixi:

– Si $b < an$, $(NC[b] - NF[b]) > 0$ i $ND[axv] = ND[bxv] + 1$, aleshores haurem trobat el segon arc admissible i s'enviaran δ unitats de preflux des de **xv** fins a **bxn** on δ està definit com:

$$\delta = \min\{NE[xv], NF[a - an], NC[b] - NF[b]\}.$$

– Si $b \geq an$, $NF[b - an] > 0$ i $ND[axv] = ND[bxv] + 1$, aleshores haurem trobat el segon arc admissible i s'enviaran δ unitats de preflux des de **xv** fins a **bxn** on δ està definit com:

$$\delta = \min\{NE[xv], NF[a - an], NF[b - an]\}.$$

I ara tenim les dues situacions següents:

1) En el cas de no trobar el primer arc admissible **a**, es reetiquetarà el node **xv** fent el següent:

   Es consideren tots els nodes adjacents **axn** del node **xv**, es crea l'arc **a** corresponent i per cada un d'ells es fa:

   - Si $a < an$, $NF[a] < NC[a]$ i $ND[xv] > ND[axv] + 1$, llavors $ND[xv] > ND[axv] + 1$.
   - Si $a \geq an$, $NF[a - an] > 0$ i $ND[xv] > ND[axv] + 1$, llavors $ND[xv] > ND[axv] + 1$.

   Fet que ens assegura que $ND[xv] = \min\{ND[axv] + 1\}$, $\forall\, axv \in D_{xv}$.

2) En el cas d'haver trobat el primer arc admissible **a** però no el segon, el **b**, es reetiquetarà el node **axv** fent:

   Es consideren tots els nodes adjacents **bxn** del node **axv**, es crea l'arc **b** corresponent i per cada un d'ells es fa:

   - Si $b < an$, $NF[b] < NC[b]$ i $ND[axv] > ND[bxv] + 1$, llavors $ND[axv] > ND[bxv] + 1$.
   - Si $b \geq an$, $NF[b - an] > 0$ i $ND[axv] > ND[bxv] + 1$, llavors $ND[axv] > ND[bxv] + 1$.

   Fet que ens assegura que $ND[axv] = \min\{ND[bxv] + 1\}$, $\forall\, bxv \in D_{axv}$.

Tant si passa una situació com l'altra, es torna a l'inici de la funció **BPNF_PreflowPush**.

En el moment que no es pugui trobar cap node amb excés, l'algorisme para i el flux màxim és:

$$F_M = NE[t].$$

4.3 Visualització

A l'arxiu **NF_View.cpp** hi trobem les funcions encarregades de la visualització de les xarxes de flux.

Es fa servir la biblioteca gràfica **Opengl** i a l'apartat [4.5] podrem veure que representem cada node amb un quadrat on a l'interior d'ell hi ha l'identificador corresponent, i cada arc amb una fletxa que indica el seu sentit i sobre la qual es pot visualitzar el seu flux i la seva capacitat.

4.4 Controls

En aquest apartat parlem de l'arxiu **MF_Control.cpp** que utilitza els arxius **GL_glut.cpp** i **GL_util.cpp**. Aquí es fa servir la biblioteca **Fast Light Toolkit**.

Són els encarregats de la programació dels controls que observem al simulador, fet que ens permet escollir amb el cursor quin és l'algorisme que volem dur a terme i veure'n resultats gràfics i numèrics.

4.5 Simulador

A la Figura [4.3] podem veure el simulador encarregat per dur a terme els algorismes de flux màxim.

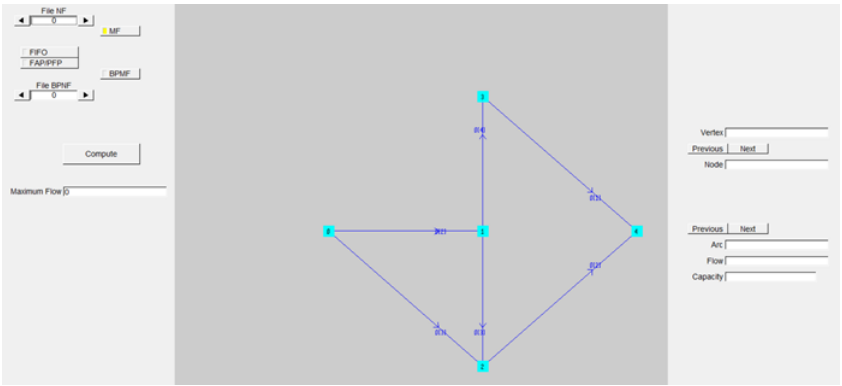

Figura 4.3

A la part central, hi ha una finestra gràfica on es poden visualitzar les xarxes de flux que desitgem.

A la part superior esquerra, hi ha l'opció de triar entre fitxers NF i fitxers BPNF, que representen a xarxes de flux i xarxes de flux bipartides, respectivament. I, a través de les fletxes que hi ha al costat del número 0, es poden escollir entre diversos exemples de xarxes.

En aquesta part, també podem escollir quin dels 3 algorismes es vol dur a terme. Per realitzar l'algorisme genèric preflow-push, se selecciona MF i FAP/PFP alhora; per realitzar el FIFO preflow-push, se selecciona MF, FAP/PFP i FIFO alhora també; i perquè

es dugui a terme l'algorisme genèric per xarxes de flux bipartides, se selecciona BPMF i FAP/PFP alhora.

A la secció *Maximum Flow* hi trobarem el resultat del flux màxim en cada pas de l'algorisme que haguem escollit.

En el moment que tinguem activades totes aquelles opcions desitjades, fem un clic al botó *Compute*, encarregat d'iniciar el càlcul del flux màxim.

A la part dreta, hi trobem la informació de qualsevol node o arc seleccionat, donant-nos l'identificador que li corresponen. I en el cas dels arcs, a més a més ens mostra el flux i la capacitat que té. Els botons *Previous* i *Next* ens permeten moure'ns per tota la xarxa de flux.

A la Figura [4.4] es pot veure un exemple d'una xarxa de flux un cop s'ha realitzat l'algorisme FIFO. Com podem observar, les caselles MF, FAP/PFP i FIFO estan seleccionades. El flux màxim és 4 i l'arc seleccionat és el número 2, que va des del node 1 fins al node 2, té 0 de flux i una capacitat d'1 unitat.

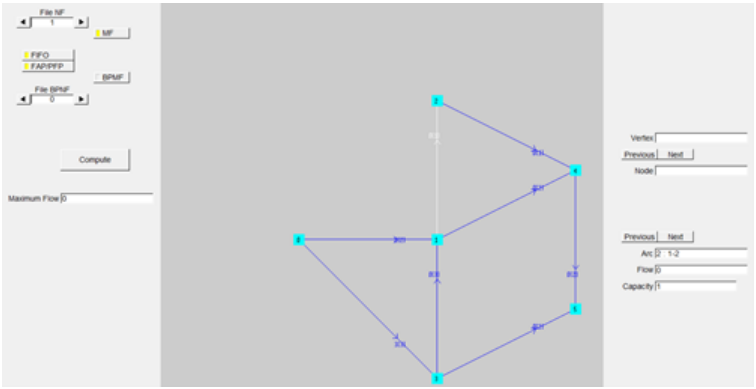

Figura 4.4

I finalment a la Figura [4.5] es pot visualitzar un exemple d'una xarxa de flux bipartida un cop s'ha realitzat l'algorisme genèric per xarxes de flux bipartides. Com podem observar, les caselles BPMF i FAP/PFP estan seleccionades. El flux màxim és 5 i en aquest cas hem seleccionat el node número 1.

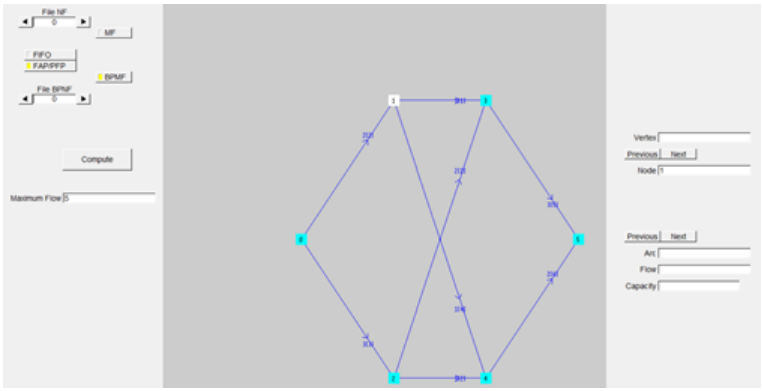

Figura 4.5

Capítol 5

Resultats i conclusions

5.1 Resultats

Com he explicat a la introducció, els algorismes de flux màxim tenen moltes aplicacions directes al món real. Al capítol 3, n'hem vist una al problema d'eliminació del beisbol. I a continuació, n'observarem una altra on l'objectiu és calcular el flux màxim d'una xarxa de canonades de distribució d'aigua d'un municipi.

En aquest cas, representem la xarxa de canonades mitjançant un dígraf connectat ponderat en el qual les àrees de districte es representen, mitjançant vèrtexs; les línies per les quals circula l'aigua, mitjançat arcs; i la màxima quantitat d'aigua que pot circular per cada línia, representarà la capacitat de l'arc corresponent.

A la Figura [5.1] podem observar l'abstracció a una xarxa de flux de la xarxa de canonades del municipi de Pyigyitagon, Mandalay, Myanmar.

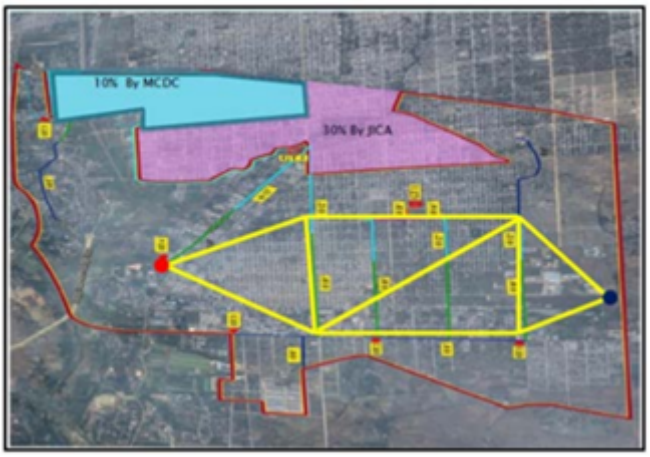

Figura 5.1

La xarxa de flux de canonades és un graf dirigit amb dos nodes distingits: font i pou. Com que les mides de les canonades utilitzades en aquesta xarxa no són les mateixes, la capacitat de cada arc també és diferent. Per tant, només pot mantenir un flux d'una certa quantitat d'aigua. En qualsevol lloc on es trobin aquestes canonades, la quantitat total d'aigua que entra a la cruïlla ha de ser igual a la quantitat que en surt. Cada arc entre

dos nodes té una capacitat no negativa c i rep un flux f, on la quantitat de flux no pot excedir la seva capacitat.

A la Figura [5.2] tenim la xarxa de flux on cada arc té 2 números: el primer representa a les capacitats i el segon al flux que hi circula en un moment determinat(litres/segon).

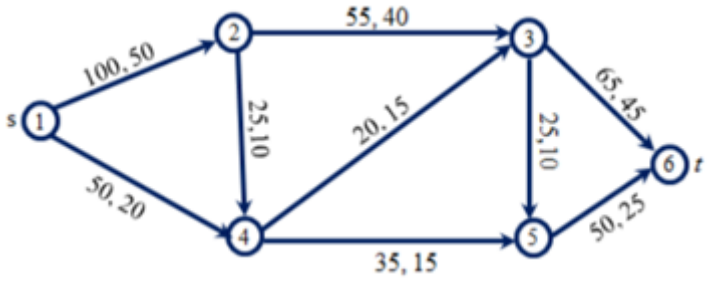

Figura 5.2

Finalment, si apliquem l'algorisme preflow-push, tenim que el flux màxim que va des del node font fins al node pou és de 110. Per tant, podem concloure que la màxima quantitat d'aigua que pot circular des de l'àrea de districte 1 fins l'àrea de districte 6 del municipi de Pyigyitagon és de 110 litres/segon. A la figura [5.3] en podem veure el resultat del simulador.

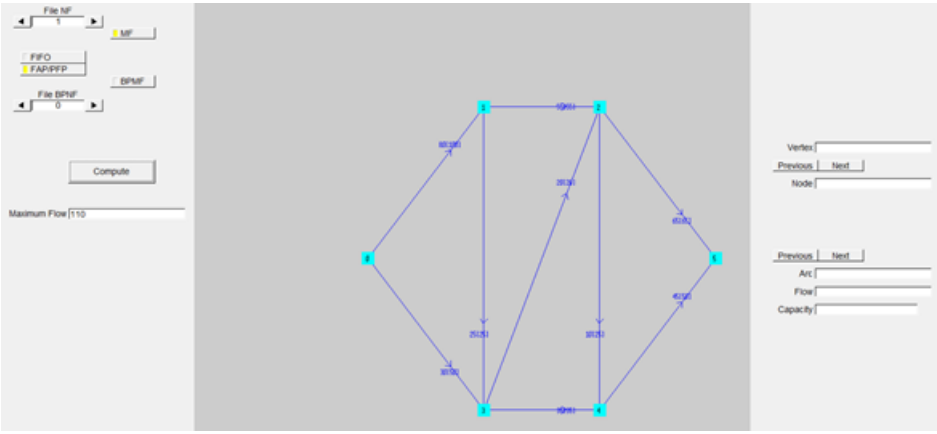

Figura 5.3

5.2 Conclusions

Conclou aquí l'estudi d'un problema que ha abarcat tant teoria de grafs com programació, dos dels camps que he trobat més interessants al llarg del grau. Concretament el problema de flux màxim tot veient diferents algorismes de tipus preflow-push. I he escollit estudiar aquests tipus d'algorismes, i no d'altres, per la seva eficiència.

A més a més, m'interessava trobar una aplicació directa dels algorismes, i així ho hem vist al problema d'eliminació del beisbol on, en un moment concret d'una temporada, es pot comprovar si un equip encara pot ser campió o no.

Finalment, gràcies al simulador, he pogut incrementar el meu coneixement d'un llenguatge informàtic que no tenia gaire familiaritzat i m'ha permès entendre millor els algorismes ja que els he pogut visualitzar pas a pas.

Per acabar, voldria especificar alguna possible continuació d'aquest treball. Als algorismes de tipus preflow-push que hem utilitzat, cada arc $a \in A$ tenia una una capacitat c(a), i buscàvem un flux f que satisfés les desigualtats $0 \leq f(a) \leq c(a)$; així doncs, una modificació natural del problema seria permetre que cada arc especifiqués un límit inferior en el seu valor de flux, a més de la seva capacitat o límit superior. És a dir, que existís una altra funció $l : A \to R^+$ tal que, a l'hora de buscar el flux màxim f, s'hagués de complir que $l(a) \leq f(a) \leq c(a)$, per a cada arc $a \in A$.

I una altra modificació seria que la xarxa de flux tingués més d'un node font i/o més d'un node pou.

Referències

[Ahuja, Magnanti & Orlin 1993] K. Ahuja, Ravindra; L. Magnanti, Thomas; B. Orlin, James: *Network flows: theory, algorithms and applications*. Prentice Hall, Upper Saddle River, New Jersey, 1993. ISBN: 0-13-617549-X.

[Jungnickel 2007] Jungnickel, Dieter: *Graphs, Networks and Algorithms. Third Edition*. Springer, Universität Augsburg, 2007. ISBN: 978-3-642-09186-5.

[Foulds 1991] Foulds, L.R.: *Graph Theory Applications*. Springer-Verlag, Hamilton, New Zeland, 1991. ISBN: 0-387-97599-3.

[Evans & Minieka 1992] Evans, J.R.; Minieka, E.: *Optimization algorithms for networks and graphs. Second Edition*. Marcel Dekker, New York, 1992. ISBN: 0-8247-8602-5.

[Gibbons 1985] Gibbons, Alan: *Algorithmic Graph Theory*. Cambridge University Press, 1985. ISBN: 978-0-521-24659-0.

[Hochstättler & Schliep 2010] Hochstättler, Winfried; Schliep, Alexander: *CATBox: An Interactive Course in Combinatorial Optimization*. Springer-Verlag, Berlin, 2010. ISBN: 978-3-540-14887-6.

[Ahuja, Orlin, Stein & Tarjan 1994] Ahuja, Ravindra K.; Orlin, James B.; Stein, Clifford; Tarjan, Robert E.: *Improved algorithms for bipartite network flow*. Society for Industrial and Applied Mathematics, 1994.
`https://pdfs.semanticscholar.org/ae81/0ac01e698f7735d93fe55c5f68f4986e3e08.pdf?_ga=2.249393732.21725191.1607773558-489810198.1607773558`

[Waine 1999] Waine, Kevin D.: *A new property and a faster algorithm for baseball elimination*. Princeton University, 1999.
`https://www.cs.princeton.edu/ wayne/papers/baseball.pdf`

[Schrijver 2002] Schrijver, Alexander: *On the history of the transportation and maximum flow problem*. University of Amsterdam, 2002.
`https://www.homepages.cwi.nl/ lex/files/histtrpclean.pdf`

[Kyi & Naing 2018] Kyi, Myint Than; Naing, Lin Lin: *Application of Ford-Fulkerson Algorithm to Maximum Flow in Water Distribution Pipeline Network*. International Journal of Scientific and Research Publications, 2018.
`http://www.ijsrp.org/research-paper-1218/ijsrp-p8441.pdf`