Flux maxim

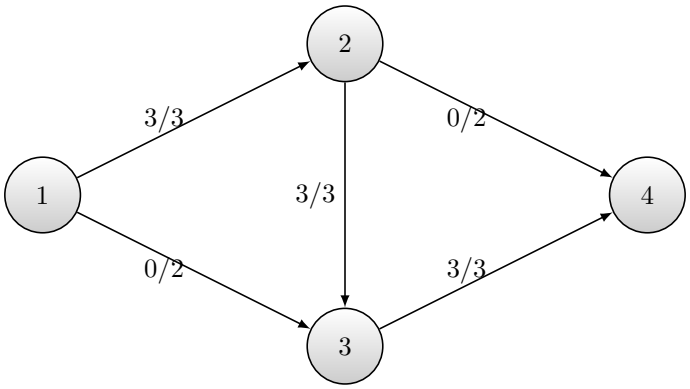
Alexandru Luchianov

Aprilie 2022

1 Prezentare a fluxului

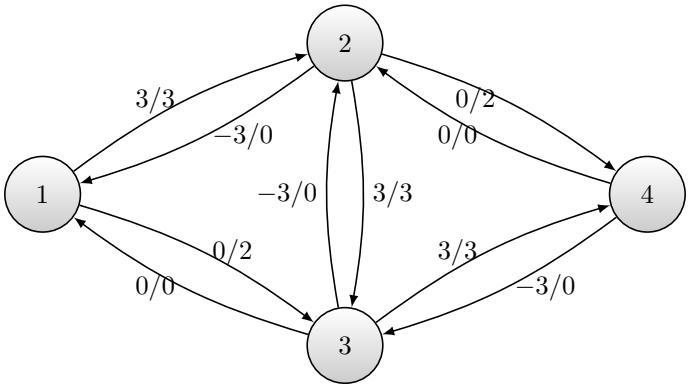
Problema clasică. Se dă un graf cu N noduri și M muchii. Pe fiecare muchie poate trece un număr de unități de flux mai mic sau egal cu capacitatea acesteia. Din fiecare nod trebuie să iasă la fel de mult flux pe cât intră, cu excepția nodurilor 1 și N. Nodul 1 se numește sursa și din el poate doar să iasă flux, iar nodul N se numește destinație și în el poate doar să intre. Se cere suma maximă a cantității de flux care intră în nodul N.

Prima încercare. O muchie *saturată* este o muchie pe care fluxul este egal cu capacitatea. Vom numi drum de *augmentare* un drum de la 1 la N pe care nu avem muchii saturate. Atâta timp cât există drumuri de *augmentare* , vom alege câte unul și vom trimite flux pe el până cel puțin o muchie îi devine saturată. Această soluție, din păcate, eșuează pentru rețeaua de mai jos dacă augmentăm mai întâi drumul $1 \to 2 \to 3 \to 4$.



Soluția corectă. Observăm faptul că soluția anterioară a eșuat deoarece odată ce a făcut o greșeală, nu a mai avut cum să o repare. Ne trebuie o metodă prin care să putem retrage fluxul trimis greșit. O metodă elegantă de a reprezenta operația de retragere a fluxului de pe o muchie este prin crearea unei muchii gemene. Pentru fiecare muchie orientată de la x la y de capacitate

z putem crea o muchie geamănă de la y la x de capacitate 0, iar de fiecare dată când trimitem flux pe una din ele să îl scădem de pe cealaltă. Cu această interpretare, rețeaua la care ne-am blocat anterior acum arată diferit:



După adăugarea muchiilor gemene a apărut un nou drum de augmentare $1 \to 3 \to 2 \to 4$ pe care mai putem trimite încă 2 unități. Se poate observa că, după augmentarea acestui drum, nu vom mai găsi alte drumuri de augmentare. Echivalența dintre absența drumurilor de augmentare și atingerea fluxului maxim o să fie demonstrată mai jos în secțiunea "Tăietura minimă".

Ford–Fulkerson. Această metodă de găsire a fluxului prin saturarea succesivă a drumurilor de augmentare se numește *Ford–Fulkerson*. In funcție de metoda prin care alegem drumurile de augmentare avem însă diferite nume și diferite complexități.

Dacă alegem un drum de augmentare arbitrar, atunci, în cel mai rău caz, fiecare drum de augmentare va fi incrementat cu o singură unitate până când devine saturat și vom obține o complexitate totală de $\mathcal{O}(F_{max}M)$ unde F_{max} este fluxul maxim. Dacă capacitățile muchiilor nu sunt numere raționale, atunci nu avem nici măcar o garanție că algoritmul se va termina în timp finit sau măcar că va converge către fluxul maxim[1].

Edmonds–Karp. In cazul în care alegem mereu cel mai scurt drum de augmentare posibil, am obține complexitatea $\mathcal{O}(NM^2)$.

Se definește *graf rezidual* ca fiind graful în care există doar muchiile nesaturate. Notăm $dist_i$ distanța minimă de la nodul 1 la nodul i în graful rezidual, de asemenea, spunem că un nod se află pe nivelul i dacă acesta se află la distanța i față de 1.

Lema 1.1. *După saturarea celui mai scurt drum de augmentare nu va scădea niciodată $dist_i$ pentru orice i.*

[1] `en.wikipedia.org/wiki/Ford-Fulkerson_algorithm#Non-terminating_example`

Demonstrație. Să notăm nodurile de pe drumul de augmentare în următorul fel: $1 = a_0 a_1 a_2 \dots a_k = N$. Din minimalitatea drumului știm că $dist_{a_i} = i$. Atunci când trimitem flux pe muchia $a_i \to a_{i+1}$, se pot întâmpla 2 lucruri:

- Muchia geamănă $a_{i+1} \to a_i$ devine nesaturată și apare în graful rezidual. Astfel, obținem o muchie de la nivelul $i+1$ la nivelul i care nu ne afectează distanțele.
- Muchia $a_i \to a_{i+1}$ devine saturată și dispare din graful rezidual. Astfel, dispare o muchie de la nivelul i la nivelul $i+1$. Această schimbare poate crește distanțele, însă nu le poate reduce.

□

Lema 1.2. *Dacă saturăm mereu cel mai scurt drum, atunci o muchie poate deveni saturată de maxim* $\frac{N}{2}$ *ori.*

Demonstrație. Știm că la momentul în care $x \to y$ este saturată avem $dist_x + 1 = dist_y$, însă pentru ca o muchie să devină saturată o a doua oară trebuie ca aceasta să devină mai întâi nesaturată. Muchia $x \to y$ devine nesaturată atunci când $y \to x$ se află pe un drum de augmentare, însă ca să se întâmple acest lucru trebuie ca $dist_y + 1 = dist_x$. Deoarece distanțele nu pot să scadă, știm că de fiecare dată când o muchie devine saturată, iar apoi devine nesaturată, trebuie ca $dist_x$ să fi crescut cu cel puțin 2. Deoarece distanțele nu pot depăși N, putem satura o muchie doar de maxim $\frac{N}{2}$ ori. □

Găsirea unui drum de augmentare se poate face în $\mathcal{O}(M)$ și fiecare drum de augmentare conține cel puțin o muchie care va deveni saturată. Deoarece fiecare muchie poate să fie saturată de doar $\frac{N}{2}$ ori și avem M muchii normale și M muchii gemene, avem maxim MN drumuri de augmentare. Astfel, obținem complexitatea totala de $\mathcal{O}(NM^2)$. In cazul în care toate capacitățile muchiilor sunt întregi, știm și faptul că algoritmul nostru are complexitatea maxim $\mathcal{O}(F_{max}M)$. In cazurile în care știm că fluxul maxim din rețea va fi mic, aceasta limită superioară a complexității poate să fie avantajoasă.

Dinic. Algoritmul lui Dinic acționează în N faze, iar în fiecare fază se petrece următorul lucru:

- Construim graful rezidual și calculăm $dist_i$;
- Construim un *graf pe niveluri* în care avem doar muchiile $x \to y$ din graful rezidual pentru care $dist_x + 1 = dist_y$;
- Se găsește un *blocaj de flux* care este compus din multiple drumuri de augmentare folosind doar muchiile din graful pe niveluri, astfel încât nu mai putem adăuga și alte drumuri de augmentare la acest set. Se poate observa ușor că drumurile de augmentare din blocajul de flux au toate aceeași lungime minimă.

- Se augmentează simultan toate drumurile de augmentare din blocajul de flux.

Lema 1.3. *După augmentarea tuturor drumurilor din blocajul de flux, lungimea celui mai scurt drum de augmentare o să crească. Astfel, numărul de faze este maxim N.*

Demonstrație. Să presupunem că după augmentarea unui blocaj de flux s-a format un nou drum de augmentare de la 1 la N de lungime minimă. Deoarece acest drum nu exista înainte în graful pe niveluri, trebuie să aibă cel puțin o muchie care nu se află în graful pe niveluri în momentul în care am calculat blocajul. Insă, singurele astfel de muchii $x \to y$ sunt de 2 tipuri:

- Existau în graful rezidual, însă aveau $dist_y < dist_x + 1$ și nu au mai fost selectate pentru graful pe niveluri.
- Nu existau în graful rezidual, însă muchia lor geamănă se afla în blocajul de flux. Deoarece muchia $y \to x$ a fost selectată într-un drum din blocajul de flux, știm că atunci când am construit level graph-ul aveam $dist_y + 1 = dist_x$.

Niciuna din tipurile de muchii prezentate mai sus nu pot face parte dintr-un drum de augmentare de lungime minimă, ceea ce ne conduce la o contradicție. □

Găsirea blocajului Pentru a găsi blocajul vom aplica de mai multe ori următorul algoritm care începe din nodul sursă:

1. Notăm nodul curent cu x;
2. Dacă nodul curent este destinația înseamnă că am descoperit un drum de augmentare.
3. Altfel extragem o muchie nemarcată din level graph care merge din nodul nostru într-un alt nod y. Aplicăm procedura în mod recursiv pentru nodul y:
   - Dacă după ce am explorat recursiv nodul y am găsit un drum de augmentare, trimitem flux pe muchia de $x \to y$, scoatem flux de pe muchia geamănă $y \to x$ și returnăm înapoi faptul că s-a găsit un drum de augmentare
   - Altfel, marcăm muchia $x \to y$ și o evităm de acum pe parcursul acestei faze, deoarece știm că nu mai există drumuri de augmentare care să pornească din y.
4. Dacă nu s-a găsit niciun drum de augmentare, returnăm acest lucru.

Atunci când o muchie este procesată, aceasta ori este marcată, ori face parte dintr-un drum de augmentare. Deoarece putem avea maxim M drumuri de augmentare în blocajul de flux (fiecare drum de augmentare trebuie să satureze cel puțin o muchie), iar fiecare astfel de drum poate conține maxim N muchii, putem deduce faptul că numărul total de muchii procesate este mai mic de $\mathcal{O}(NM)$.

In concluzie, complexitatea finală a algoritmului lui Dinic este numărul de faze înmulțit cu complexitatea în care rezolvăm una din aceste faze, $\mathcal{O}(N^2M)$. Ca și la Edmonds-Karp putem găsi o limită superioară a complexității în cazul în care toate capacitățile sunt întregi. Fie F_{max} fluxul maxim în graf, atunci algoritmul lui Dinic va avea maxim complexitatea $\mathcal{O}(F_{max}N+NM)$, demonstrația acestui lucru poate să fie ușor obținută prin modificări minore aduse demonstrației de mai sus.

Algoritmii de flux, mai ales Dinic, se comportă mult mai bine în practică deoarece multe probleme au anumite proprietăți specifice.

Un exemplu des întâlnit de graf pe care acest algoritm funcționează semnificativ mai rapid este cel în care toate muchiile au capacitatea 1. In acest caz, blocajul de flux al fiecărei faze poate să fie găsit în $\mathcal{O}(M)$ deoarece toate drumurile de augmentare de la această fază sunt disjuncte. De asemenea, se poate deduce și o limită mai mică a numărului de faze ale algoritmului. După primele $\sqrt{M}$ faze toate drumurile de augmentare rămase au lungimea mai mare ca $\sqrt{M}$. Deoarece aceste drumuri de augmentare sunt disjuncte, iar lungimile lor sunt mai mari ca $\sqrt{M}$, putem spune că avem mai puțin de $\sqrt{M}$ drumuri de augmentare, astfel că algoritmul va termina în alte $\sqrt{M}$ faze. Combinând cele 2 argumente de mai sus putem demonstra o complexitate de $\mathcal{O}(M\sqrt{M})$ în acest caz.

Un caz particular pentru care se poate demonstra o limită chiar mai strânsă este problema cuplajului pe graf bipartit[2]. Această problemă este de cele mai multe ori rezolvată folosind algoritmul Hopcroft-Karp.[3] Acest algoritm are complexitatea de $\mathcal{O}(N\sqrt{M})$ și este de fapt un caz particular al algoritmului lui Dinic.

2 Tăietura minimă

O *s-t-tăietura* este o partiționare a grafului în 2 mulțimi S și T astfel încât $s \in S$ și $t \in T$. Costul unei tăieturi este egal cu suma capacităților muchiilor orientate de la S la T. Din afirmația precedentă este ușor de dedus faptul că fluxul maxim de la s la t nu poate să fie mai mare decât costul oricărei s-t-tăieturi. Notăm s-t-tăietura cu cel mai mic cost drept *s-t-tăietura minimă*.

Theorem 2.1. *Fluxul maxim de la s la t este egal cu s-t-tăietura minimă.*

Demonstrație. Dacă încă există drumuri de augmentare în graf, atunci este clar că nu am atins fluxul maxim, altfel putem presupune că nu mai există drumuri de augmentare. Putem partiționa graful în 2 mulțimi, S si T, astfel încât S

[2] `infoarena.ro/problema/cuplaj`

[3] `en.wikipedia.org/wiki/Hopcroft-Karp_algorithm`

să conțină toate nodurile accesibile de la s mergând doar pe muchii nesaturate. Deoarece am presupus că nu mai există drumuri de augmentare, T trebuie să conțina pe t. Toate muchiile de la S la T sunt saturate, iar pe toate muchiile de la T la S nu circulă deloc flux deoarece altfel muchiile lor gemene ar fi desaturate. Astfel, putem spune că fluxul curent este egal cu suma capacităților muchiilor orientate de la S la T, însă acesta este și costul partiționării în S și T. Deoarece știm că toate fluxurile trebuie să fie mai mici sau egale cu toate tăieturile și noi am identificat un flux și o tăietura de cost egal, putem spune că acestea sunt fluxul maxim și tăietura minimă. □

Un avantaj neașteptat al teoremei de mai sus este demonstrarea faptului că odată ce nu mai există drumuri de augmentare înseamnă ca am atins fluxul maxim.

Alte aplicații ale tăieturii minime într-un graf sunt problema selecției proiectelor și problema setului închis maxim[4].

3 Imbunătățiri prin scalare

O metodă de a îmbunătăți complexitatea algoritmilor prezentați mai sus în grafurile pe care toate capacitățile sunt întregi este prin scalare. Fie F_{max} fluxul maxim. Metoda scalării acționează în $\log F_{max} + 1$ pași, acești pași sunt numerotați descrescător de la $\log F_{max}$ la 0. In cadrul pasului i considerăm doar drumurile de augmentare care acceptă cel puțin 2^{i+1} unități de flux.

Tehnica de mai sus poate îmbunătăți complexitatea algoritmului Edmonds-Karp la $\mathcal{O}(M^2 \log F_{max})$ și a algoritmului lui Dinic la $\mathcal{O}(MN \log F_{max})$. In ciuda complexității teoretice superioare, această îmbunătățire nu ameliorează în practică timpul de execuție chiar atât de mult. Demonstrația acestor complexități se bazează în principal pe faptul că la pasul i nu vor mai exista drumuri de augmentare pe care să putem împinge cel puțin 2^{i+1} unități de flux. Astfel, la pasul i vom considera maxim M drumuri de augmentare. Datorită acestei limite a numărului de drumuri de augmentare știm că fiecare pas procesat cu Edmonds-Karp va avea complexitatea $\mathcal{O}(M^2)$ și fiecare pas procesat cu Dinic va avea complexitatea $\mathcal{O}(NM)$. Aceste lucruri revin din observațiile noastre anterioare asupra complexităților acestor algoritmi în cazul în care toate muchiile au capacități întregi.

Metoda scalării poate îmbunătăți complexitatea teoretică, însă este necesar ca muchiile să aibă capacități întregi. In practică, aceasta nu produce diferențe majore de performanță.

4 Problema circulației

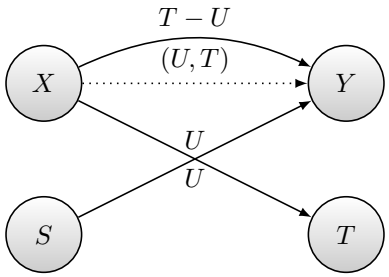
O problemă strâns legată de cea a fluxului maxim este cea a circulației într-un graf. Aceasta este de cele mai multe ori enunțată în felul următor:[5]

[4] `codeforces.com/blog/entry/101354`

[5] `en.wikipedia.org/wiki/Circulation_problem`

Se dă un graf cu N noduri și M muchii. Pe fiecare muchie poate trece un număr de unități de flux mai mic sau egal cu capacitatea superioară a acesteia și un număr de unități de flux mai mare sau egal cu capacitatea inferioară a acesteia. Din fiecare nod trebuie să iasă la fel de mult flux pe cât intră. Să se găsească pentru fiecare muchie un număr de unități de flux care să treacă prin ea astfel încât să respectăm toate condițiile simultan.

Problema circulației poate să fie reformulată într-o instanță a problemei fluxului maxim prin adăugarea unor noduri și muchii fictive. O muchie $X \to Y$ cu capacitate inferioară U și capacitate superioară T poate să fie reprezentată în rețeaua noastră de flux fictivă printr-o muchie $X \to Y$ de capacitate $T - U$, o muchie $X \to T$ de capacitate U și o muchie de $S \to Y$ de capacitate U ca și în figura de mai jos:



Este usor de observat faptul că, dacă noua noastră rețea de flux admite un flux maxim în care toate muchiile adiacente cu sursa sunt saturate, atunci am găsit și o circulație validă. In același timp, dacă nu am reușit să identificăm un astfel de flux maxim, putem spune că nu există nicio circulație validă în graf.

5 Flux maxim de cost minim

O modificare des întâlnită a problemei fluxului maxim este cea de a impune fiecărei muchii un anumit cost pentru fiecare unitate de flux care o traversează.[6]

In acest articol vom presupune că nu există cicli de cost negativ în graful nostru deoarece algoritmii care se ocupa de astfel de situații sunt semnificativ mai complicați, iar cele mai multe rețele nu conțin cicli de cost negativ.

Pentru a rezolva aceasta problemă putem adapta strategia noastră anterioară la costuri. Singura diferență este că, pentru fiecare muchie de cost z, alegem costul muchiei gemene ca $-z$. Apoi, căutăm în mod repetat drumul de augmentare de cost minim.

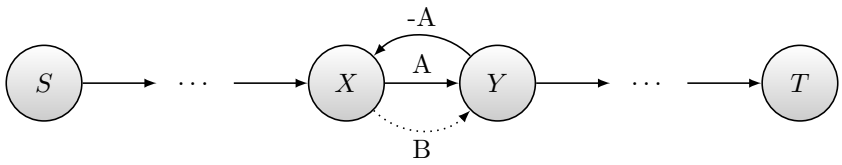
Demonstrația corectitudinii acestei strategii se găsește mai jos.

Lema 5.1. *Dacă la început nu există cicli de cost negativ în rețeaua de flux, atunci nici după augmentarea succesivă a drumurilor de cost minim nu vor exista cicli de cost negativ.*

[6] `infoarena.ro/problema/fmcm`

Demonstrație. Să presupunem că după augmentarea unui drum de cost minim s-a format un ciclu de cost negativ. Deoarece înainte de această operație nu exista niciun ciclu de cost negativ, înseamnă că ciclul nostru negativ conține muchia geamănă a unei muchii de pe drumul de augmentare.

Fie nodurile din lanțul de augmentare $S \dots XY \dots T$, unde $Y \to X$ este o muchie din ciclul de cost negativ. Notăm cu B costul lanțului de la X la Y din acest ciclu, iar cu A costul muchiei de la X la Y. Din ipoteză cunoaștem faptul că costul ciclului, $B - A$, trebuie să fie negativ, dar asta ar însemna că am putea înlocui muchia $X \to Y$ din drumul de augmentare cu lanțul $X \to Y$ din ciclu și am obține un drum de cost mai mic. Însă, noi deja am presupus anterior că am ales drumul minim, ceea ce ne conduce la o contradicție.



□

Lema 5.2. *Strategia de augmentare succesivă a celor mai scurte drumuri găsește mereu soluția de cost minim.*

Demonstrație. Să presupunem că strategia de mai sus ne conduce la o soluție care nu este optimă. Atunci noi vom putea transforma soluția noastră în soluția optimă prin augmentarea mai multor cicli. Se poate observa faptul că augmentarea unui ciclu este echivalentă cu înlocuirea unui lanț dintr-un drum de augmentare cu un alt lanț. Din acești cicli, cel puțin unul trebuie să aibă cost negativ deoarece altfel nu am obține o soluție mai bună. Existența unui ciclu negativ ne conduce la o contradicție. □

Putem căuta în mod repetat cel mai scurt drum de augmentare folosind algoritmul Bellman-Ford[7]. Astfel, putem rezolva problema fluxului maxim de cost minim în $\mathcal{O}(F_{max}NM)$, însă o complexitate mai bună este posibilă.

Un algoritm care poate să găsească drumul cel mai scurt într-un graf într-un mod mai eficient este algoritmul lui Dijkstra[8], însă acesta se bazează pe faptul că toate costurile sunt non-negative, ceea ce nu este aplicabil în cazul nostru. Putem evita aceasta limitare prin construcția unui graf auxiliar cu costuri non-negative.

In acest scop vom introduce variabilele ajutătoare acc_i. Fie aceste variabile acc_i alese in așa fel astfel încât pentru fiecare muchie nesaturată $x \to y$ de cost z să se respecte inegalitatea:

$$acc_y \leq acc_x + z$$

[7] `infoarena.ro/problema/bellmanford`
[8] `infoarena.ro/problema/dijkstra`

Folosind această informație vom construi un graf secundar în care pentru fiecare muchie $x \to y$ de cost z din graful original vom avea o muchie $x \to y$ de cost $z + acc_x - acc_y$. Acest graf secundar are avantajul că toate muchiile au costuri non-negative datorită inegalității de mai sus și fiecare drum din graful original corespunde unic unui drum din graful secundar. De exemplu, costul drumului celui mai scurt de la x la y în graful original diferă printr-o constantă, $acc_x - acc_y$, față de costul drumului celui mai scurt de la x la y în graful secundar. Astfel, putem identifica drumul de cost minim folosind Dijkstra.

Singura problemă rămasă este identificarea unor valori acc_i care să respecte inegalitatea. Pentru început, putem alege acc_i ca fiind costul minim al unui drum de la sursă la nodul i. O metodă ușoară de a identifica un set de valori valid este aceea de a alege acc_i ca fiind distanța de la sursă la i. Această distanță poate să fie calculată folosind algoritmul Bellman-Ford. Nu putem menține aceleași valori acc_i pe toată durata algoritmului deoarece după augmentarea fiecărui drum anumite muchii pot deveni nesaturate și astfel pot introduce noi restricții. În mod simetric unele muchii pot devine saturate și elimina din restricții.

Cea mai simplă metodă de a recalcula acc_i după augmentarea drumului de cost minim este chiar inițializarea acestuia cu $dist_i$, distanța de la sursa la nodul i. Acest $dist_i$ a fost deja calculat atunci când am identificat drumul de augmentare de cost minim.

Pentru fiecare muchie $x \to y$ de cost z care era nesaturată înainte de augmentarea drumului știm că $dist_y \leq dist_x + z$ datorită inegalității distanțelor în graf, astfel mai rămâne de tratat doar cazul muchiilor care au devenit nesaturate dupa augmentare.

Pentru ca o muchie $x \to y$ de cost z să devină nesaturată, muchia sa geamănă, $y \to x$ de cost $-z$, trebuie să facă parte din drumul minim de augmentare. Acest lucru implică faptul că $dist_y - z = dist_x$. Trecând termenul z în partea cealaltă obținem $dist_y = dist_x + z$, exact cazul de egalitate pentru inegalitatea pe care doream să o obținem.

Astfel, am obținut un algoritm care găsește fluxul maxim de cost minim în $\mathcal{O}(F_{max} M log N)$ sau $\mathcal{O}(F_{max}(N log N + M))$ în funcție de implementarea algoritmului Dijkstra.