STUDIA UNIVERSITATIS MOLDAVIAE, 2017, nr.7(107)
Seria "Ştiinţe exacte şi economice" ISSN 1857-2073 ISSN online 2345-1033 p.150-158

CZU: 519.17:519.688

PROBLEMA FLUXULUI MAXIM ÎN REŢELE – ANALIZA ŞI SINTEZA ALGORITMILOR DE SOLUŢIONARE

Tatiana PAŞA

Universitatea de Stat din Moldova

În lucrare se propune o prezentare generală a algoritmilor de soluționare a problemei fluxului maxim în rețele de transport. Sunt descrise tehnicile de bază utilizate de-a lungul timpului începând cu primul algoritm propus de L.R. Ford şi D.R. Fulkerson [1]. Se aduc referințe la acei autori care cercetează această problemă pentru cazuri speciale, cum sunt grafuri neorientate, bipartite sau cu câteva destinații și surse.

Cuvinte-cheie: *rețea de transport, flux maxim, capacitatea arcului, capacitatea vârfului, graf rezidual, tăietură minimă, drum de creștere, sursă, destinație.*

THE MAXIMUM FLOW PROBLEM ON THE NETWORKS – ANALYSIS AND SINTHESIS OF THE ALGORITHMS

In this paper, we describe generally the algorithms that solve the max-flow problem in the transport networks. We describe the basic techniques used over time, starting with the algorithm proposed by L.R. Ford and D.R. Fulkerson [1]. We also make a reference to the authors that investigate the special cases of this problem, i.e undirected graphs, bipartite or with several sources and sinks.

Keywords: *transportation network, maximum flow, arc capacity, residual graph, minimum cut, augmenting path, sours, destination.*

Introducere

Problema fluxului maxim în rețele de transport este una foarte răspândită și constă în determinarea mărimii fluxului maxim de produs ce poate fi transportat dintr-un punct, *sursă*, în alt punct, *destinație*, printr-o rețea, în care fiecare arc care unește două vârfuri are o capacitate limitată de transportare. În calitate de produs care trebuie transportat dintr-un punct în altul putem avea apă, gaz, petrol, curent electric, pietriș sau informație. În calitate de rețea putem considera rețeaua de drumuri care leagă câteva orașe, apeductul, liniile de tensiune electrică, liniile de comunicații și altele. Deci, această problemă permite modelarea unor procese economice, cum ar fi proiectarea liniilor de electricitate, a rețelei de comunicații sau a rețelei de drumuri, a conductei de apă sau gaze sau optimizarea problemelor de producere și păstrare a mărfii în depozite, precum și optimizarea fluxurilor de pasageri.

În [2] sunt descrise unele tehnici de soluționare a problemei fluxului maxim. Pentru prima dată această problemă a fost formulată în 1956 de către L.R. Ford şi D.R. Fulkerson [1, 3], fiind propus şi un algoritm de soluționare a acesteia care permite determinarea fluxului maxim în rețea bazat pe mărirea fluxului de-a lungul drumurilor de creștere. Pentru îmbunătățirea complexității acestui algoritm au fost utilizate căutarea celor mai scurte drumuri de creștere [4], fluxurile saturate [5], prefluxul [6], structurile de date [7], iar mai târziu structurile dinamice arborescente [8]. Soluționarea unui caz special al problemei fluxului maxim (rețea cu capacități unitare) este prezentată în [9] și [10], unde metoda Push-Relabel dă o complexitate liniară în raport cu dimensiunea grafului care descrie rețeaua. În [11] metoda Push-Relabel este prezentată ca o alternativă a metodei prefluxului.

În ultimul timp, un interes deosebit prezintă metodele de soluționare a acestei probleme, care, chiar dacă se execută într-un timp mai mare, pot fi implementate mai ușor. În acest context, în [12] este prezentat un algoritm de balansare potrivit pentru aplicarea paralelă și distribuită, care permite soluționarea problemei de flux maxim parametric. O implementare eficientă a algoritmului Push-Relabel utilizând calculul paralel este prezentată în [13]. În [14] este implementat în paralel un algoritm genetic, pentru a îmbunătăți viteza de soluționare a problemei parametrice. Un alt tip de probleme ce trezesc interesul este determinarea fluxului maxim în rețele cu câteva surse și câteva destinații [15]. În [16, 17 și 18] este studiată problema fluxului maxim aproximativ în grafuri neorientate.

În continuare, vom formula noțiunile de bază și vom face o prezentare generală a algoritmilor de bază, care permit soluționarea problemei fluxului maxim în ordinea cronologică a apariției lucrărilor în care aceștia

au fost descrişi şi analizaţi. Vom descrie cazurile speciale ale problemei formulate care sunt studiate în ultimii ani, dar şi soluţiile ce sunt propuse pentru soluţionarea lor mai eficientă.

Preliminarii

În continuare vom defini noţiunile de bază cu care vom opera în această lucrare, pentru formularea corectă a cărora au fost consultate sursele [1, 19-22], unde pot fi găsite şi demonstraţiile teoremelor.

Definiţie: *Graf neorientat* este o pereche arbitrară $G = (V, E)$, $E = \{(u, v) | u, v \in V, u \neq v\}$.

Definiţie: *Graf orientat* este o pereche arbitrară $G = (V, E)$, în care $E \subseteq V \times V$ [15].

Vom considera graful orientat $G = (V, E)$ cu mulţimea de vârfuri V, unde $|V| = n$, şi mulţimea de noduri E, unde $|E| = m$.

Definiţie: *Graf bipartit* este un graf neorientat $G = (V, E)$, dacă există V_1 şi V_2 astfel încât $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, astfel încât fiecare muchie din G are o extremitate în V_1 şi cealaltă în V_2.

Definiţie: Numim *reţea de transport* un graf orientat $G = (V, E)$, fără bucle, care satisface următoarele proprietăţi:

1) Există un vârf (sursă) $v_s \in V$ care nu are ascendenţi;
2) Există un vârf (destinaţie) $v_t \in V$ care nu are descendenţi;
3) Fiecărui arc îi este asociată o mărime c_e, pentru orice $e \in E$, numită *capacitatea arcului*.

Pot fi modelate cu uşurinţă situaţiile în care avem mai multe surse şi/sau destinaţii. Pentru a transforma aceste structuri în reţelele de transport standard, se va adăuga o sursă fictivă din care pornesc arce în fiecare dintre sursele reţelei iniţiale şi/sau destinaţie fictivă, în care intră arce din fiecare dintre destinaţiile reţelei iniţiale. Sursa fictivă va conţine totalul produsului din sursele iniţiale şi destinaţia fictivă va avea capacitatea de a primi totalul de produs din toate destinaţiile iniţiale.

În caz general, fiecărui arc i se poate asocia o mărime care descrie volumul de produs transportat, timpul parcurs dintr-un vârf în altul, distanţa parcursă între două puncte sau o altă mărime care ar descrie fluxul din reţea.

Definiţie: Numim *flux* într-o reţea funcţia $f: E \to \mathbb{R}$ care respectă următoarele proprietăţi:

1) Restricţii de capacitate: pentru orice $e \in E$ are loc $f(e) \leq c_e$;
2) Simetria: pentru orice $(v_1, v_2) \in V$ are loc $f(v_1, v_2) = -f(v_2, v_1)$;
3) Conservarea fluxului: $\sum_{x \in E^+(v)} x(e) - \sum_{x \in E^-(v)} x(e) = 0$, unde $E^+(t)$ este mulţimea de arce care intră în $v_t \in V$ (destinaţie) şi $E^-(s)$ – mulţimea de arce care ies din $v_s \in V$(sursă).

Restricţiile de capacitate presupun că de-a lungul unui arc nu poate fi transportat mai mult produs decât o cantitate anterior stabilită.

Simetria presupune că pentru fiecare vârf cantitatea de produs care intră în el este egală cu cantitatea de produs care iese din vârf, adică coincid după modul.

Conservarea fluxului presupune că cantitatea de produs care iese din sursă ajunge integral la destinaţie, adică nu sunt pierderi în vârfuri. Nu poate avea loc cazul când avem surplus în destinaţie, ceea ce ar presupune că în unele vârfuri are loc producere.

Definiţie: Numim *capacitate reziduală a unui arc* $c_f(u, v) = c(u, v) - f(u, v)$ fluxul care poate fi transportat suplimentar din u în v fără depăşirea capacităţii $c(u, v)$.

Definiţie: Fie graful $G = (V, E)$ cu fluxul f. Numim *graf rezidual* $G_f(V, E_f)$, astfel încât $E_f = \{(u, v) \in V | c_f(u, v) = c(u, v) - f(u, v) > 0\}$.

Definiţie: Numim *drum de creştere* o cale $p = (v_1, v_2, \dots, v_k)$ în graful rezidual cu $c_f(v_i, v_{i+1}) > 0$, $i = \overline{1, k-1}$, unde $v_1 = v_s$, iar $v_k = v_t$.

Definiţie: Numim *capacitate reziduală a unui drum* p cantitatea maximă de flux $c_f(p) = min\{c_f(u, v) | (u, v) \in p\}$ ce poate fi transportată de-a lungul său.

Cel puţin un arc pe drumul de creştere p are capacitatea reziduală $c_f(u, v) > 0$ până la îmbunătăţire şi $c_f(u, v) = 0$ după îmbunătăţire, un astfel de arc va fi exclus din graful G_f şi îl vom numi *arc saturat*.

Definiţie: Numim *tăietură a unui graf* $G = (V, E)$ o partiţie (S, T), pe care o notăm S/X, a nodurilor, astfel încât $v_s \in S$ şi $v_t \in T$.

Definiţie: *Capacitatea tăieturii* este suma capacităţilor tuturor arcelor cu vârful iniţial în S şi vârful final în T, $(S, T) = \sum_{u \in S, v \in T} c(u, v)$.

Definiţie: Mărimea maximă a fluxului net care intră în destinaţie $|f| = \sum_{(v, v_t) \in E} f(v, v_t) - \sum_{(v_t, v) \in E} f(v_t, v)$ se numeşte *flux maxim* în reţea.

Teoremă: Pentru un graf $G = (V, E)$ cu fluxul f următoarele afirmații sunt echivalente:

1) f este flux maxim în G;
2) graful rezidual G_f nu conține drumuri de creștere;
3) există o tăietură (S, T) în G, astfel încât fluxul net care trece prin tăietură este egal cu capacitatea tăieturii.

Teorema Fluxul Maxim - Tăietură Minimă. *Mărimea fluxului maxim, din sursă în destinație, într-un graf orientat $G = (V, E)$ este egală cu capacitatea tăieturii minime care separă sursa de destinație.*

Demonstrație: Demonstrația este prezentată în [1].

Definiție: Numim *capacitatea vârfului* v, unde $v \in G_0(V_0, E_0)$ este vârf al grafului stratificat, mărimea $cap(v) = min\{\sum_{(u,v)\in E_0} cap(u, v), \sum_{(v,w)\in E_0} cap(v, w)\}$.

Pentru vârfurile v_s și v_t se vor considera sumele capacităților arcelor care intră în vârf, respectiv care ies din vârf egale cu zero.

Definiție: Numim *flux saturat* [2] într-un graf orientat fluxul f, unde fiecare drum $v_s \to v_t$ conține un arc cu capacitatea reziduală zero.

Definiție: Numim *cel mai scurt flux de saturare* un flux f, într-un graf $G = (V, E)$, dacă $f(u, v) > 0$ este o muchie de pe un drum minim din v_s în v_t și fluxul f saturează orice drum minim din v_s în v_t.

Definiție: Numim *preflux* [11] într-o rețea de transport funcția $f: E \to \mathbb{R}$ care respectă următoarele proprietăți:

1) pentru orice $e \in E$ are loc $f(e) \leq c_e$;
2) pentru orice $(v_1, v_2) \in V$ are loc $f(v_1, v_2) = -f(v_2, v_1)$
3) pentru orice $v \in V$ are loc $\sum_{x\in E^+(v)} x(e) > \sum_{x\in E^-(v)} x(e)$, unde $E^+(t)$ este mulțimea de arce care intră în $v_t \in V$ (destinație) și $E^-(s)$ – mulțimea de arce care ies din $v_s \in V$.

Deci, preflux este fluxul pentru care se permite ca cantitatea totală de flux care intră în vârf să fie mai mare decât cantitatea de flux care iese din vârf.

Definiție: Numim *flux leftmost* [23] fluxul pentru care graful rezidual nu conține cicluri reziduale în sensul direcției arcelor de ceasornic.

Definiție: Numim *graf stratificat* toate drumurile minime între v_s și v_t în graful rezidual G_f, care formează subgraful G_0 construit utilizând căutarea în lățime modificată [5].

Algoritmi de soluționare a problemei fluxului maxim

După cum se cunoaște, fluxul într-o rețea de transport care este descrisă de un graf orientat este limitat de capacitățile arcelor, deci trebuie respectate restricțiile asupra mărimii fluxului care poate fi transportat de-a lungul unui arc. Un alt aspect care trebuie prevăzut în calculul mărimii fluxului maxim este conservarea fluxului.

Pentru prima dată soluția problemei fluxului maxim a fost formulată de către L.R. Ford și D.R. Fulkerson în 1956 [1]. *Algoritmul Ford – Fulkerson* [2] permite determinarea fluxului maxim în rețea iterativ, iar corectitudinea metodei reiese din *teorema fluxului maxim – tăietură minimă* și are la bază ideea că atât timp cât există un drum de creștere care unește sursa cu destinația vom putea mări fluxul cu mărimea reziduală a acelui drum. În algoritm nu se specifică care este modalitatea de alegere a drumului de creștere în graful rezidual.

Algoritmul începe cu un flux admisibil și constă din doi pași importanți:

1) marcarea unui drum rezidual pentru fluxul din v_s în v_t, astfel încât $f(e) < c_e$, $e \in E$ pentru arcele care ies din vârf și $f(e) > 0$, $e \in E$ pentru arcele care intră în vârf;

2) dacă este găsit un drum rezidual, atunci are loc modificarea acestui flux, iar în caz contrar fluxul obținut este de mărime maximală. Orice nod poate fi în una dintre stările: nemarcat, marcat și cercetat sau marcat și necercetat. Deci, vom spune că avem flux maxim în cazul când în procesul de marcare nu se ajunge la vârful v_t.

În cazul în care capacitățile arcelor sunt mărimi întregi, timpul de execuție a algoritmului este $O(|E|f)$, deoarece timpul de determinare a fiecărui drum de creștere este de $O(|E|)$ (pentru care se poate utiliza căutarea în lățime sau căutarea în adâncime), iar creșterea fluxului este întotdeauna unitară, unde $|E|$ este numărul de arce în graf, iar f este fluxul de valoare maxim.

J.Edmonds și R.Karp [4] au descris în 1972 metoda celui mai scurt drum de creștere care este o adaptare a algoritmului Ford - Fulkerson. Cel mai scurt drum se va obține utilizând căutarea în lățime și asociind fiecărui arc lungimea unu. După cum se poate observa, fiecare următorulul drum obținut nu poate fi mai scurt, ci trebuie doar să crească. Fiecare creștere a unuia dintre aceste drumuri va satura cel puțin un arc, deci numărul total de creșteri este de ordinul $|E||V|$. Timpul pentru determinarea unui drum de creștere este $O(|E|)$; în așa fel, obținem că timpul total de execuție a algoritmului propus de Edmond și Karp este $O(|V||E|^2)$ și este strict polinomial.

Metoda fluxurilor de saturare, propusă de E.Dinic [5] în 1970, constă în determinarea tuturor drumurilor minime între v_s și v_t în G_f și saturarea acestor drumuri. Subgraful G_0, numit graf stratificat, este construit utilizând căutarea în lățime modificată. Aceasta constă în următoarele: după determinarea distanțelor de la sursa v_s la fiecare vârf v, $dist(v)$, se vor elimina toate vârfurile u diferite de v_t care au $dist(u) \geq dist(v_t)$ și vor fi păstrate doar acele arce (u, v) pentru care $dist(u) = dist(v) + 1$. În final, noul graf este parcurs în sens invers, începând cu v_t, eliminând vârfurile care nu au fost atinse împreună cu arcele incidente. În cazul în care un drum de creștere conține $O(|V|)$ arce și fiecare creștere saturează un arc, atunci timpul de execuție a algoritmului fluxului de saturare este $O(|V||E|)$, deci algoritmul lui Dinic de determinare a fluxului maxim este de $O(|V|^2|E|)$.

A.Karzanov în lucrarea sa din 1974 [6] propune un algoritm care are ca scop reducerea numărului de iterații necesar pentru construcția celui mai scurt flux de saturare de la $O(|V||E|)$ la $O(|V|)$.

Ideea de bază a algoritmului lui Karzanov este ca la fiecare iterație să fie saturat un vârf și nu un arc, ca în cazul algoritmului lui Dinic. Deoarece există cel mult $|V|$ vârfuri în graful stratificat, numărul de iterații va fi cel mult $|V|$. Algoritmul va începe cu un vârf v care are cea mai mică capacitate $cap(v)$ și vom împinge un flux de mărime $cap(v)$ până la destinație v_t. Graful G_0 nu conține vârfuri din care nu se ajunge la destinație. Această proprietate va fi respectată atunci când eliminăm vârfuri și arce din graf, deci după eliminarea vârfului saturat și a arcelor asociate lui va exista neapărat un drum prin care se va ajunge la destinație.

Determinarea vârfului de capacitate minimă se poate face în complexitate $O(|V|)$. La o iterație pot fi vizitate $O(|V|)$ vârfuri și deci doar $O(|V|)$ arce nu vor fi eliminate. Deoarece există $O(|V|)$ iterații, complexitatea totală este $O(|V|^3)$.

A.Karzanov a descris și metoda prefluxului, care constă în modificarea fluxului pe un singur arc în schimbul modificării fluxului de-a lungul unui drum de creștere și permite determinarea fluxului de saturare mult mai rapid.

Un caz special al problemei fluxului maxim este când toate arcele au capacitate unitară. S.Karzanov [9], S.Even și R.E. Tarjan [10] au arătat că în acest caz algoritmul prefluxului atinge un timp liniar proporțional în raport cu dimensiunea grafului, deoarece se reduce numărul calculelor prefluxului și calculele devin mai rapide. Algoritmul Even-Tarjan începe cu un flux nul și fiecare fază începe cu o căutare în lățime de la v_s, care este marcat cu 0 (zero). Fiecare următorul vârf accesibil din v este marcat cu $\lambda(v) + 1$. Se construiește graful auxiliar care conține toate drumurile din v_s în v_t ce conțin $\lambda(v_t)$ arce. Pe un arc va fi transmis un flux înainte (în direcția parcurgerii arcului) dacă $f(e) < c(e)$ și va fi transmis un flux înapoi (în direcție opusă parcurgerii arcului) dacă $f(e) > 0$. Deoarece fiecare fază va lua nu mai mult de $O(|V||E|)$ pași, iar numărul total de faze fiind cel mult $|V| - 1$, putem concluziona că algoritmul are complexitatea $O(|V|^2|E|)$.

A fost formulată și demonstrată teorema, conform căreia algoritmul lui Dinic se va executa în cel mult $O(|V|^{2/3}|E|)$ pași pentru cazul când într-o rețea de transport capacitățile sunt egale cu unu și conține doar arce. În cazul rețelelor de transport care au toate capacitățile unitare, iar fiecare vârf, în afară de v_s și v_t, conține doar un arc care intră și un arc care iese din el, atunci algoritmul lui Dinic se va executa în $O(|V|^{1/2}|E|)$ timp.

Manipulările cu prefluxul au motivat dezvoltarea structurilor de date care permit modificarea mărimii fluxului mult mai eficient de-a lungul unui drum. Pentru prima dată ele au fost descrise în 1980 de către Z.Galil și A.Naamad [7]; acestea prezintă un algoritm bazat pe algoritmul lui Dinic care păstrează unele secvențe de drumuri, astfel încât să nu trebuiască mai târziu din nou redescoperite după ștergerea unui arc saturat. Asupra secvențelor de drum sunt descrise operațiile de concatenare, tăiere și ștergere. După manipulările efectuate asupra secvențelor de drum neapărat este executat pasul care transferă fluxul în nodurile secvenței de drum pentru a corecta mărimea fluxului în arcele care nu au fost șterse. Se demonstrează că algoritmul descris de Galil și Naamad va fi executat în $O(|E||V|log^2|V|)$ timp.

D.D Sleator și R.E. Tarjan [24, 8] au descris arborii binari de căutare care sunt structuri de date ce permit accesarea, adăugarea și ștergerea mult mai ușoară a nodurilor. Algoritmul Sleator-Tarjan are la bază lucrarea [5] și utilizează structurile de date dinamice pentru determinarea drumurilor de creștere și evidența capacităților arcelor. Conform algoritmului propus, avem o mulțime de arbori ce conțin vârfuri din care iese cel puțin un arc nesaturat, acel arc fiind candidat de a forma un drum de creștere. Utilizând structurile de date dinamice arborescente, modificarea mărimii fluxului pe un drum din k arce se va efectua în $O(\log k)$ timp. Acest fapt conduce la atingerea teoretică a timpului de determinare a prefluxului făcând-l aproape liniar. Pentru executarea algoritmului va fi nevoie de $O(|E||V| \log|V|)$ timp.

A.V. Goldberg și R.E. Tarjan [25] au dezvoltat metoda Push-Relabel care are la bază două operații: Push – împinerea fluxului și Relabel – marcarea din nou a vârfurilor, fiind o alternativă a metodei prefluxurilor. Algoritmul păstrează prefluxul în rețeaua inițială și împinge excesul de flux local către destinație de-a lungul celor mai scurte drumuri din rețeaua reziduală. Această împingere a fluxului modifică rețeaua reziduală și drumurile către destinație pot deveni saturate. Excesul ce nu poate fi transportat către destinație este întors în sursă la fel de-a lungul celor mai scurte drumuri. La finalul algoritmului prefluxul devine flux care reprezintă fluxul maxim. Operația fină de creștere asigură flexibilitate, care este utilizată în descrierea algoritmilor mai rapizi.

Implementarea secvențială a algoritmului conduce la rularea lui în $O(|V|^3)$ timp. În cazul în care sunt încorporați arborii structurilor de date dinamice descrise de Sleator și Tarjan, se va obține o versiune a algoritmului ce va rula în $O(|E||V|\log(\frac{|V|^2}{|E|}))$ timp. În cazul implementării paralele, care poate fi aplicată asupra algoritmului, rularea are loc în $O(|V|^2\log|V|)$ timp utilizând $|V|$ procesoare și $O(|E|)$ spațiu.

R.K. Ahuja și J.B. Orlin [26] propun, în 1989, o modificare a algoritmului Goldberg – Tarjan, care efectuează $\log U$ iterații de creștere, iar fiecare iterație necesită $O(|V|)$ împingeri nesaturate, prin care fluxul din nodurile cu un exces suficient de mare este împins în nodurile cu exces suficient de mic atât timp cât excesele nu sunt prea mari. În așa fel, timpul de calcul al algoritmului este $O(|V||E| + |V|^2\log|V|)$. Utilizând structuri de date elementare, algoritmul împinge fluxul printr-un nod într-o unitate de timp. Cu o mașină paralelă cu acces aleatoriu PRAM (Paralel Random Access Machine), utilizând modelul EREW (Eclusive Read Exclusiv Write), algoritmul se execută în $O(|V|^2\log U\log p)$, unde $p = \lceil|E|/|V|\rceil$ este numărul de procesoare. În [27] este descris un algoritm care utilizează arborii binari pentru atingerea timpului de execuție a algoritmului $O(|V||E|\log\left(\left(\frac{|V|}{|E|}\right)(\log U)^{\frac{1}{2}}+2\right)+|E|\log_{|V|}U)$.

Problema determinării fluxului maxim în rețele cu câteva surse și câteva destinații a fost studiată de Miller și Naor în [28], unde se propune un algoritm care se execută secvențial în $O(\sqrt{|V|})$ timp, iar implementarea sa paralelă utilizând $O(\sqrt{|V|})$ procesoare necesită $O(I(|V|)log^2|V|)$ timp, unde $I(|V|)$ este timpul de calcul al sumei a $|V|$ valori.

Mult timp a existat un decalaj mare între cazul grafurilor cu capacități unitare și cazul general, dar acesta a fost micșorat de A.V. Goldgerg și S.Rao [11]. Rezolvând problema cu capacități întregi, ei au folosit o funcție neunitară și, combinând-o cu noi tehnici de analiză, au ajuns la algoritmul binar al fluxurilor saturate. Este extinsă metoda fluxurilor saturate și sunt utilizate funcțiile lungimilor binare care asociază lungimea 0 (zero) arcurilor de capacitate mare și lungimea 1 (unu) arcurilor de capacitate mică. Un fapt important este adaptivitatea funcțiilor: pragul lungimii este definit de mărimea unui flux rezidual. Această adaptare este un element important pentru îmbunătățirea timpului de execuție a algoritmului. Fiecare iterație a algoritmului este dominată de calculul fluxului saturat într-un graf aciclic. Algoritmul propus de Goldgerg și Rao necesită pentru execuție $O(\min\left(|V|^{2/3},\sqrt{|E|}\right)|E|\log(\frac{|V|^2}{|E|})\log U)$ timp cu capacitățile arcelor ce au mărimile întregi $[1,2,\dots,U]$.

Conform algoritmului Y.Boycov și V.Kolmogorov [29], creșterile nu se efectuează de-a lungul celor mai scurte drumuri și acesta nu tinde să fie polinomial, fiind o îmbunătățire a algoritmului de creștere a drumurilor. Utilizând căutarea bidirecțională pentru găsirea drumurilor de creștere și combinat cu o metodă mai bună de păstrare a datelor obținute la căutările precedente, duce la mărirea vitezei următoarelor căutări. Algoritmul poate fi modificat astfel încât să devină un algoritm polinomial fără a sacrifica din ușurința implementării practice.

În 2006, R.Tarjan, J.Ward, B.Zhang, Y.Zhou și J.Mao [12] prezintă un algoritm de balansare, care permite calculul fluxului maxim în rețele, cu capacități întregi de mărimea maximă U, în $O(|V|^2|E|\log(|V|U))$ timp. Cu toate că este mai lent comparativ cu alți algoritmi, acesta este foarte ușor de implementat și potrivit pentru aplicarea paralelă și distribuită. Algoritmul începe cu un flux inițial, cum ar fi fluxul nul, cu un exces fictiv de $+\infty$ în sursă și un exces $-\infty$ în destinație. Repetând pașii de balansare a arcelor până când toți pașii transferă o cantitate destul de mică de flux astfel obținându-se chiar fluxul maxim. Acest algoritm este aplicabil și în cazul soluționării problemei fluxului maxim parametric.

Un algoritm de determinare a fluxului maxim în grafuri orientate planare, care este, de fapt, o generalizare a algoritmului Ford-Fulkerson, a fost prezentat de G.Borradaile și P.Klein [23] în 2009, care se execută în $O(|V|\log|V|)$. Algoritmul începe cu o mărime nulă a fluxului *leftmost* și este urmat de saturarea repetată a drumului rezidual leftmost de la sursă la destinație.

În [15] este descris algoritmul ce permite determinarea fluxului maxim într-o rețea cu câteva surse și câteva destinații pe un graf planar orientat. Algoritmul utilizează pseudofluxul și schema divide et impera. Graful inițial este divizat în două subgrafe și recursiv se soluționează problema fluxului maxim în fiecare din ele. Ca urmare, nu vor exista în niciun graf drumuri reziduale, deci nu vor exista nici în graful inițial. Pseudofluxul obținut poate fi transformat în flux maxim în timp linear utilizând tehnici prezentate în lucrare. Aceasta duce la executarea algoritmului în $O(|V|log^3|V|)$ timp.

Algoritmul pentru determinarea fluxului maxim și a tăieturii minime într-un graf orientat cu capacități unitare, care este prezentat în [30], constă în reducerea problemei fluxului maxim la o problemă perfectă bipartită de tip b-matching și rulează în $O\left(|V|^{\frac{10}{7}}\right) = O(|V|^{1.43})$.

J.B. Orlin [31] a arătat că problema fluxului maxim poate fi soluționată în $O(|V||E| + |E|^{\frac{31}{16}} log^{\,2}|V|)$ timp, iar când $|E| = (|V|^{\left(\frac{16}{15}\right)-\varepsilon})$ timpul de execuție este $O(|V||E|)$. Dimensiunea descompunerii fluxului poate avea mărimea $\Omega(|V||E|)$, deci timpul de implementare dorit trebuie să fie $O(|V||E|)$. Asta nu este limita de jos pentru calculul fluxurilor, ci poate fi reprezentat în spațiul de mărimea $O(|V|)$, iar arborii dinamici pot fi utilizați pentru creșterea fluxului în timp logaritmic. În aceeași lucrare, Orlin descrie două proceduri de căutare: *forward-search* și *backward-search*, care sunt niște modificări ale algoritmilor standard de căutare în grafuri și permit soluționarea problemei fluxului maxim în $O(|V|^2/\log|V|)$ timp în cazul când rețeaua satisface condiția $|E| = O(|V|)$.

Tendința ultimilor ani

În ultimii ani, chiar dacă mai sunt unele intenții de a îmbunătăți algoritmii de soluționare a problemei fluxului maxim în rețea, cercetătorii care studiază această problemă își schimbă prioritățile:

- o îmbunătățire a algoritmilor este considerată implementarea lor mai ușor la calculator atât în varianta secvențială, cât și în cea paralelă;
- soluționarea aproximativă a problemei în timp aproape-liniar pentru cazul rețelelor complicate, care altfel ar necesita mult mai multe resurse și timp;
- determinarea fluxului maxim în rețele cu câteva surse și/sau destinații;
- aplicarea algoritmilor clasici pentru soluționarea problemei fluxului maxim în rețele dinamice standard și în cele bipartite.

În [32] este prezentată o variantă modificată a algoritmului Push-Relabel de soluționare a problemei fluxului maxim în rețele cu $|E| = O(|V|)$, care se execută în $O(|E||V|)$ timp. Algoritmul propus este descris în [25]. În lucrare sunt tratate dificultățile asociate îmbunătățirii metodei Push-Relabel în rețelele compacte, după cum urmează: 1) rețeaua reziduală urmează să fie înnoită după fiecare transfer al capacităților, în care scop pot fi utilizate structurile de date dinamice arborescente Sleator-Tarjan; 2) este necesară garantarea că împingerile de-a lungul pseudoarcelor vor respecta restricțiile capacităților arcelor din rețeaua reziduală originară.

În [33] este descris un algoritm îmbunătățit, mult mai ușor de implementat decât cel descris în [29], pentru soluționarea problemei în grafuri orientate care calculează fluxul maxim în $O(|E|^{10/7}U^{1/7})$ timp, unde U este cea mai mare capacitate întreagă a arcului. La fiecare etapă a algoritmului se va determina un drum de creștere în graful curent rezidual, care în graful original va codifica soluția obținută mai târziu. Algoritmul se va finisa atunci când graful rezidual nu mai admite creșterea fluxurilor, ceea ce înseamnă că nu există drumuri din v_s în v_t, iar soluția găsită anterior este soluția optimă.

Într-un șir de lucrări, precum [16, 17 și 18], este studiată problema fluxului maxim aproximativ în grafuri neorientate și sunt descriși algoritmii care se execută în timp aproape liniar utilizând tehnici liniar algebrice și fluxuri electrice. O abordare nouă de calcul aproximativ al fluxului maxim în grafuri neorientate este introdusă de P.Cristiano, J.Kelner, A.Madry, D.A. Spielman și S-H. Teng în lucrarea comună [16]. Fluxul se obține în urma soluționării unei serii de probleme ale fluxului electric, fiecare flux electric este dat de soluția unui sistem de ecuații liniare în matricea Laplaciană, ceea ce poate fi aproximativ calculat în timp aproape–liniar. În așa fel, algoritmul permite obținerea unui flux $(1-\varepsilon)$ aproximativ în $O(|E||V|^{1/3}\varepsilon^{-11/3})$ timp. În [17] este introdusă o aproximare a problemei fluxului maxim în grafuri neorientate utilizând mai multe aproximări. Pe parcursul algoritmului se va păstra un flux care poate avea câteva excese și insuficiențe reziduale. Un instrument de soluționare aproximativă a problemei fluxului în grafuri neorientate este prezentat și în [18], care asigură algoritmi asimptotic mai rapizi. Acest fapt permite să determinăm o $\varepsilon-$ aproximare a fluxului maxim în $O(||E||^{1+O(1)}\varepsilon^{-2})$ timp.

În [34] autorul descrie un algoritm pentru soluționarea problemei fluxului maxim în rețele bipartite, obținut ca o îmbunătățire a parcurgerii nodurilor active dat de algoritmul prezentat în [35]. Nodurile sunt cercetate în ordine descrescătoare a distanței de marcare, iar excesul lor total sau parțial va fi împins cât mai aproape de destinație.

Nicoleta Aversalor prezintă în lucrarea [36] problema fluxului maxim în rețele parametrice dinamice, în care propune un algoritm de soluționare a acestei problemei pentru o secvență de valori ale parametrului. Fluxul crește treptat de-a lungul celor mai scurte drumuri de la sursă la destinație. În [37] este propus un algoritm care soluționează problema fluxului maxim în rețele parametrice dinamice fără limite de jos, care constă în aplicarea algoritmului de determinare a fluxului maxim într-o rețea statică obișnuită prin extinderea rețelei dinamice originale.

Problema fluxului maxim în rețele bipartite dinamice este studiată de către C.Șchiopu în [38], care propune soluționarea acesteia reformulând-o ca pe o problemă într-o rețea bipartită statică.

În [39] autorii arată că fluxul în rețele aciclice poate fi calculat mult mai rapid decât în alte rețele. Ei prezintă două modalități de soluționare a problemei fluxului maxim în timp puternic polinomial și arată că calculul și aproximarea fluxului întreg este NP- complexă.

Printre alți autori care au cercetat, într-un fel sau altul, problema fluxului maxim în rețele mai putem menționa pe următorii: Richard Peng, Pierre Bonami, Dorian Mazauric, Yann Vaxes, Tibor Illes, Richard Molnar-Szipai, Hed Sagi, Kaplan Haim, Kohli Pushmeet, Yin Tat Lee, Aaron Sodford, Han-Ying Wei, Zhi-Xiong Su și alții. În Republica Moldova cu studierea acestei teme s-au ocupat Sergiu Cataranciuc, Angela Niculița, Valeriu Ungureanu, Dumitru Lozovanu, Dumitru Zambițchi.

Concluzii

Pentru soluționarea problemei fluxului maxim au fost descrise un șir de tehnici și metode. Dacă inițial cercetătorii încercau să reducă complexitatea algoritmilor de soluționare, în ultimii ani se caută varii modalități de soluționare a unor cazuri speciale ale problemei de bază, cum sunt grafurile neorientate, bipartite sau rețele cu câteva destinații și surse, dar și posibilități de implementare a algoritmilor pentru soluționarea altor probleme.

Tabel

Algoritmi de soluționare a problemei fluxului maxim

Nr.	***Autori / referințe***	**Anul**	***Complexitate***	***Comentarii***
1	Ford & Fulkerson / [1, 2]	1956	$O(\|E\|f)$	f - flux maxim
3	Dinic / [5]	1970	$O(\|V\|^2\|E\|)$	Fluxuri saturate
2	Edmonds & Karp / [4]	1972	$O(\|V\|\|E\|^2)$	Cel mai scurt drum
4	Karzanov / [6]	1974	$O(\|V\|^3)$	
5	Even & Tarjan / [10]	1975	$O(\|V\|^2\|E\|)$	Prefluxul
6	Galil &Naamad / [7]	1980	$O(\|E\|\|V\|log^2\|V\|)$	Structuri de date
7	Sleator & Tarjan / [8, 24]	1983	$O(\|E\|\|V\|\log\|V\|)$	Arbori binari de căutare
8	Goldberg & Tarjan / [25]	1988	$O(\|E\|\|V\|\log(\frac{\|V\|^2}{\|E\|}))$	Push-Relabel
			$O(\|V\|^2\log\|V\|)$	Implementare paralelă
9	Ahuja & Orlin / [26]	1989	$O(\|V\|\|E\| + \|V\|^2\log\|V\|)$	
			$O(\|V\|^2\log U\log p)$	$p = \lceil\|E\|/\|V\|\rceil$, implementare paralelă
10	Ahuja, Orlin & Tarjan/[27]	1989	$O(\|V\|\|E\|\log\left(\left(\frac{\|V\|}{\|E\|}\right)(\log U)^{\frac{1}{2}}+2\right) + \|E\|\log_{\|V\|}U)$	U - capacitate maximă
11	Miller & Naor / [28]	1995	$O(\sqrt{\|V\|})$	
			$O(I(\|V\|)log^2\|V\|)$	Implementare paralelă
12	Goldberg & Rao / [11]	1998	$O(\min\left(\|V\|^{2/3},\sqrt{\|E\|}\right)\|E\|\log(\frac{\|V\|^2}{\|E\|})\log U)$	
13	Tarjan, Ward, Zhong, Zhou &Mao / [12]	2006	$O(\|V\|^2\|E\|\log(\|V\|U))$	U - capacitate maximă
14	Borradaile & Klein / [23]	2009	$O(\|V\|\log\|V\|)$	Leftmost

15	Borradaile, Klein, Mazes, Nussbaum, Wulff-Nulsen / [15]	2011	$O(\|V\|log^3\|V\|)$	Pseudoflux
16	Cristiano, Kelner, Madry, Spielman & Teng / [16]	2011	$O(\|E\|\|V\|^{1/3}\varepsilon^{-11/3})$	Flux $(1-\varepsilon)-$ aproximativ
17	Kelner, Lee, Orecchia & Sidford / [18]	2014	$O(\|\|E\|\|^{1+O(1)}\varepsilon^{-2})$	Flux $\varepsilon-$ aproximativ
18	Madry / [30]	2013	$O\left(\|V\|^{\frac{10}{7}}\right) = O(\|V\|^{1.43})$	
19	Orlin / [31]	2013	$(\|V\|\|E\| + \|E\|^{\frac{31}{16}} log\ ^2\|V\|)$	
			$O(\|V\|\|E\|)$	$\|E\| = (\|V\|^{(\frac{16}{15})-\varepsilon})$
			$O(\|V\|^2/\log\|V\|)$	$\|E\| = O(\|V\|)$
20	Mehta / [32]	2014	$O(\|E\|\|V\|)$	$\|E\| = O(\|V\|)$
21	Madry / [33]	2016	$O(\|E\|^{10/7}U^{1/7})$	

În Tabel sunt prezentați algoritmii pentru care autorii au demonstrat complexitatea teoretică de executare; eficiența altora este demonstrată experimental.

Referinnțe:

1. FORD, L.R., FULKERSON, D.R. Maximal flow through a network. In: *Canadian Journal of Mathematics*, 1956, vol.8, p.399-404.
2. GOLDBERG, V., TARJAN, R.E. Eficient maximum flow algorithms. In: *Communications of the ACM* , August, 2014, vol.*57*, no.8, p.82-89.
3. FORD, L.R., FULKERSON, D.R. *Flows in networks.* New Jersy, USA: The RAND Corporation, Pricenston University Press, 1962.
4. ENDMONDS, J., KARP, R. Theoretical improvements in algorithmic efficiency for network flow problems. In: *Journal of the ACM*, 1972, vol.19, p.248-264.
5. DINIC, E.A. Algorithm for solution of a problem of maximum flow in networks with power estimation. In: *Soviet Mathematics Doklady*, 1970, vol.11, p.1277-1280.
6. KARZANOV, A.V. Determining the maximal flow in a network by the method of preflows. In: *Soviet Mathematics Doklady*, 1974, vol.15, p.434-437.
7. GALIL, Z., NAAMAD, A. An O(EVlog2V) Algorithm for the Maximal Flow Problem. In: *Journal of Computer and System Sciences*, 1980, vol.21, p.203-217.
8. SLEATOR, D.D., TARJAN, R.E. Self-Adjusting Binary Search Trees. In: *Journal of the Association for Computing Machinery*, 1985, vol.32, no.3, p.652-686.
9. КАРЗАНОВ, А.В. Точная Оценка алгоритма махимального потока применённого к задаче о представителях. In: *Problems in Cibernetics*, 1973, vol.5, p.66-70.
10. EVEN, S., TARJAN, R. E. Network flow and testing graph connectivity. In: *SIAM Journal of computing*, 1975, vol.4, no.4, p.507-518.
11. GOLDBERG, A.V., RAO, S. Beyond the Flow Decomposition barrier. In: *Journal of the ACM, 1998*, vol.45, no.5, p.783-797.
12. TARJAN, R., WARD, J., ZHANG, B., ZHOU, Y., MAO, J. Balancing Applied to Maximum Network Flow. In: *Prodeeding of the 14th European Symposium on Algorithms* Berlin: Spriger-Verlag, 2006, p.612-623.
13. BAUMSTARK, N., BLELLOCH, G., SHUN, I. Efficient Implementation of Syncronouns Parallel Push-Relabel Algorithm, 2015, *https://arxiv.org/pdf/1507.01926.pdf* [Accesat: decembrie 2017].
14. SURAKHI, O.M., QATAWNEH, M., & OFEISHAT, H.A. A Parallel Genetic Algorithm for Maximum Flow Problem. In: *International Journal of Advanced Computer Science and Aplications*, 2017, vol.8 no.6, p.159-164.
15. BORRADAILE, G., KLEIN, P., MOZES, S., NUSSBAUM, Y., & WULFF-NILSEN, C. Multiple-Source Multiple-Sink Maximum Flow in Directed Planar Graphs in Near-Linear Time. In: *Preceeding of the 52th Annual iEEE Symposium on Fundations of Computer Science*, New York: IEEE Press, 2011, p.170-179.
16. CRISTIANO, P., KELNER, J.A., MADRY, A., SPIELMAN, D.A., & TENG, S.-H. Electrical Flows, Lapcacian Systems, and Faster Aproximation of Maximum Flow in Undirected Graphs. In: *Preceeding of the Annual ACM Symposium on Theory of Computing*. New York: ACM Press, 2011, p.273-282.
17. SHERMAN, J. Nearly Maximum Flows in Nearly Linear Time. *Proceeding at the annual IEEE Symposium on Fondations of Computer Science - FOCS 2013*, Los Alamitos, CA: IEEE Computer Soc., 2013, p.263-269.

18. KELNER, J.A., LEE, Y.T., ORECCHIA, L., & SIDFORD, A. An Almost-Linear-Time Algoritm for Aproximate Max Flow in Undirected Graphs, and its Multicomodity. In: *Proceedings of the Twenty-Fifth Annual ACM - SIAM Symposium on Discrete Algorithms*. Philadelphia: SIAM, 2014, p.217-226.
19. БЕРЖ, К. *Теория графов и её применения.* Москва: Издательство Иностранной литературы, 1962.
20. УПСОН, Р. *Введение в теорию графов.* Москва: Мир, 1978.
21. CORMEN, T.H., LEISERSON, C.E., & RIVEST, R.L. *Introduction to Algorithms*. 2nd Edition. London: England MIT Press, 2002.
22. CORLAT, S., CORLAT, A. *Grafuri. Noțiuni. Algoritmi. Implementări.* Chișinău, 2012.
23. BORRADAILE, G., KLEIN, P. An O(nlogn) algorithm for maximum st-flow in a directed planar graph. In: *Journal of the ACM*, 2009, vol.56, no.2, p.1-34.
24. SLEATOR, D.D., TARJAN, R.E. A data structure for dynamic trees. In: *Journal of Computer and System Sciences*, 1983, vol.26, no.3, p.362-391.
25. GOLDBERG, A.V., TARJAN, R.E. A New Approach to the Maximum-Flow Problem. *Journal of the Association for Computing Machinery*, 1988, vol.35, no.4, p.921-940.
26. AHUJA, R.K., ORLIN, J.B. A fast and simple algorithm for the maximum flow problem. In: *Operations Research*, 1989, vol.37, no.5, p.748-759.
27. AHUJA, R.K., ORLIN, J.B., TARJAN, R.E. Improved time bounds for the maximum flow problem. In: *SIAM Journal on Computing*, 1989, vol.18, no.5, p.939-954.
28. MILLER, G.L., NAOR, J. Flow in planar graphs with multiple sources and sinks. In: *SIAM Journal on Computing*, 1995, vol.24, no.5, p.1002-1017.
29. BOYCOV, Y., KOLMOGOROV, V. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. In: *IEEE Transactions on PAMI*, 2004, vol.26, no.9, p.1124-1137.
30. MARDY, A. Navigating central path with electrical flows From flows to matchings and back. In: *Preceeding of the Annual IEEE Symposium on Foundations of Computer Science*. New York: IEEE Press, 2013, p.253-262.
31. ORLIN, J.B. Max Flows in O(nm) Time, or Better. In: *Journal of the ACM*, June 1-4, 2013, arxivorg/pdf/1310.7840.pdf [Accesat: decembrie 2017].
32. MEHTA, R. *A New Push-Relabel Algorithm for Sparse Networks*, 2014, arxiv.org/pdf/1310.7840.pdf [Accesat: decembrie 2017].
33. MADRY, A. Computing Maximum flow with Augmenting Electrical Flows. In: *57th Annual IEEE Sympposium om Fondations of Computer Science - FOCS 2016*. Los Alamiotos, CA: IEEE Comp. Soc., 2016, p.593-602.
34. CIUPALĂ, L. Wave algorithm for maximum flow in semi-bipartite networks. In: *Bulletin of the Transilvania University of Brașov,* 2016, vol.9, no.58, p.111-118.
35. CIUPALĂ, L.A generic preflow algorithm for maximum flow in semibipartite networks. In: *Bulletin of the Transilvania University of Brașov,* 2014, vol.7, no.56, p.103-108.
36. AVERSALOV, N. Maximum flow over time problem in parametric networks. In: *Buletin of the Transilvania University of Brașov, Series III: Mathematics, Informatics, Physics,* 2014, vol.7, no.56-1, p.95-102.
37. AVERSALOR, N. Maximum flows in parametric dynamic networks with lower bounds. In: *Annals of the University of Craiova, Mathematics and Computer Science Series*, 2016, vol.43, no.2, p.188-199.
38. AVERSALOV, N. Maximum flow over time problem in parametric networks. In: *Buletin of the Transilvania University of Brașov*. Series III: *Mathematics, Informatics, Physics*, 2014, vol.7, no.56-1, p.95-102.
39. HOLZHAUSER, M., KRUMKE, S. O., & THIELEN, C. Maximum flows in generelized processing networks. In: *Journal Comb. Optim.*, 2017, vol.33, no.4, p.1226-1256.

Prezentat la 10.12.2017