UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO
FAKULTETA ZA MATEMATIKO IN FIZIKO

Gaja Velkavrh

Problem maksimalnega pretoka

DIPLOMSKO DELO
NA INTERDISCIPLINARNEM UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Borut Robič

Ljubljana, 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00034/2012
Datum: 15.03.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko ter Fakulteta za matematiko in fiziko izdaja naslednjo nalogo:

Kandidat: **GAJA VELKAVRH**

Naslov: **PROBLEM MAKSIMALNEGA PRETOKA**
THE MAXIMUM-FLOW PROBLEM

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Predstavite razvoj učinkovitih algoritmov za problem maksimalnega pretoka. Opišite njihove glavne ideje in zveze s prejšnjimi algoritmi. Opišite eksaktne algortme in stanje na področju neeksaktnih algoritmov.

Mentor:

prof. dr. Borut Robič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan Fakultete za računalništvo in informatiko:

prof. dr. Nikolaj Zimic

Dekan Fakultete za matematiko in fiziko:

akad. prof. dr. Franc Forstnarič

UNIVERZA V LJUBLJANI • Fakulteta za matematiko in fiziko • 6

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisana Gaja Velkavrh,

z vpisno številko 63060248,

sem avtorica diplomskega dela z naslovom:

Problem maksimalnega pretoka.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom
  prof. dr. Boruta Robiča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 07.05.2012 Podpis avtorja/-ice:

Zahvala

Zahvaljujem se mentorju, prof. dr. Borutu Robiču, za predloge in strokovno pomoč pri izdelavi diplomske naloge ter za vse znanje, ki sem ga od njega prejela v času študija.

Prav tako se zahvaljujem vsem sošolcem, predvsem Anžetu, Blažu, Nejcu, Roku in Slavku za vso pomoč v času študija in dobro družbo na predavanjih.

Zahvala gre tudi mojim staršem, bratu Aljažu in fantu Mihi za vso spodbudo.

Dedu

Kazalo

Seznam uporabljenih kratic in simbolov

BFS	iskanje v širino (ang. Breadth - First Search)
DA	Dinitzov algoritem
DFS	iskanje v globino (ang. Depth - First Search)
EREW	ekskluzivno branje ekskluzivno pisanje (ang. Exclusive Read Exclusive Write)
FIFO	prvi noter, prvi ven (ang. First In, First Out)
FF	Ford - Fulkersonov algoritem
PRAM	vzporedni stroj z naključnim dostopom (ang. Parallel Random Access Machine)

Povzetek

Problem maksimalnega pretoka je eden izmed temeljnih problemov teorije pretoka v omrežju in je temeljito raziskan. V delu smo opisali razvoj učinkovitih algoritmov za problem maksimalnega pretoka in trenutno stanje na tem področju. Pri vsakem algoritmu smo opisali glavno idejo, ki jo je avtor uporabil, da je dosegel boljšo časovno zahtevnost in zvezo s prejšnjimi algoritmi. Algoritmov podrobneje nismo opisali, ker so nekateri, še posebno novejši, zelo zapleteni. Zato pa smo navedli literaturo, kjer so algoritmi podrobneje opisani.

Prvi, ki je definiral ta problem, je Dantzig (1951), rešila pa sta ga Ford in Fulkerson (1956) z uporabo metode povečevanja toka na poti. Od takrat je bilo algoritmov vedno več.

Edmonds in Karp (1968) sta pokazala, da ima algoritem Ford - Fulkerson časovno zahtevnost $O(nm^2)$, če tok povečujemo najprej najkrajšim potem. Neodvisno je Dinitz (1970) pokazal koncept omrežij z najkrajšimi potmi - večplastna omrežja - in dosegel časovno zahtevnost $O(n^2m)$. To mejo je izboljšal Karzanov (1974), ki je predstavil koncept predtokov v večplastnih omrežjih.

Kasneje sta to časovno zahtevnost izboljšala Goldberg in Tarjan (1986). Njun algoritem pošlje predtok iz vozlišč, ki so najbližje ponoru. Podatek o oddaljenosti je boljši kot večplastna omrežja, ker je enostavnejši za razumevanje in lažji za uporabo v vzporednih algoritmih. Z uporabo skaliranja presežka sta Ahuja in Orlin (1988) izboljšala Goldberg - Tarjanov algoritem.

Waissi (1992) je s postopno kombinacijo blokiranja predtokov in povratnih tokov prišel do takrat najhitrejšega algoritma za problem maksimalnega pretoka. S hranjenjem liste višin sosednjih vozlišč za vsako vozlišče so Pham, Lavallee, Bui in Do (2005) izboljšali Goldberg - Tarjanov algoritem. Trenutno najhitrejši aproksimacijski algoritem pa so si zamislili Christiano, Kelner, Mandry in Spielman (2010).

Ključne besede:

pretok v omrežju, maksimalni pretok, metoda pošiljanja predtoka, metoda označevanja, metoda blokiranja pretoka, metoda, ki pošlje ponovno označenim, metoda povečanja toka na poti.

Abstract

Maximum flow problem is one of the fundamental problems in network flow theory and has been extensively investigated. In this work we describe the development of efficient algorithms for the maximum flow problem and the current situation in this area. For each algorithm, we describe the main idea, which is used to achieve a better time complexity and the relationship with previous algorithms. We did not describe algorithms in more detail, because some, especially more recent, are very complicated. Therefore, we mentioned literature where algorithms are described in more detail.

The problem was first defined by Dantzig (1951) and solved by Ford and Fulkerson (1956), using the augmenting paths method. Since then has been an increasing number of algorithms.

Edmonds and Karp (1968) showed that the algorithm of Ford - Fulkerson has time complexity $O(nm^2)$, if the flow is increased firstly on shortest paths. Independently Dinitz (1970) demonstrated the concept of networks with the shortest paths - multi-layered networks - and reached the time complexity $O(n^2m)$. This limit was improved by Karzanov (1974), who introduced the concept of preflows in multi-layer networks.

This time complexity was later improved by Goldberg and Tarjan (1986). Their algorithm sends preflow from the nodes that are closest to the sink. Information about the distance is better than a multi-layered network, because it is simpler to understand and easier to use in parallel algorithms. By using excess scaling Orlin and Ahuja (1988) improved Goldberg - Tarjan's algorithm.

With progressive combination of blocking backflow and preflow Waisse (1992) showed the fastest algorithm for the maximum flow problem. With storage of the list of neighbor nodes heights for each node Pham, Lavallee, Bui and Do (2005) improved Goldberg - Tarjan's algorithm. Currently the fastest approximation algorithm is envisioned by Christiano, Kelner, Mandry and Spielman (2010).

Key words:

network flow, maximum flow, preflow-push method, labelling method, blocking flow method, push-relable method, augmenting path method.

Poglavje 1

Uvod

1.1 Motivacija in cilji

Dandanes so omrežja vsepovsod: energetska in električna omrežja pripeljejo svetlobo in energijo v naše domove. Telefonska omrežja nam omogočajo komunikacijo s celim svetom. Avtocestno, železniško omrežje in letalstvo nam omogočajo premagovati daljše razdalje za poslovne in privatne zadeve. Računalniško omrežje pa je spremenilo način širjenja informacij in vpliva na poslovno in zasebno življenje vseh nas. Na vseh teh področjih, in še v mnogo večih, želimo poslati neko entiteto (elektriko, osebo, vozilo, sporočilo, produkt, ...), iz ene točke v drugo, čim bolj učinkovito (udobno, hitro, poceni).

Problem maksimalnega pretoka, ki ga bomo obravnavali v tej diplomski nalogi, dopolnjuje problem minimalnega pretoka. Problem maksimalnega pretoka išče izvedljivo rešitev, ki pošlje maksimalno količino toka iz določenega izvora s v določen ponor t (iz začetnega v končno vozlišče).

Problem se je začel razvijati v 50-ih letih, ko so raziskovalci začeli kazati vse večje zanimanje za problem minimalnega pretoka in njegovih podproblemov - problem najkrajše poti, problem maksimalnega pretoka, problem dodelitve - predvsem zaradi pomembnosti teh modelov v realnem svetu. Začetniki so bili Dantzig, Ford in Fulkerson.

1.2 Struktura diplomskega dela

Poleg uvoda in zaključka diplomsko delo sestavljajo še štiri osrednja poglavja. Drugo poglavje opisuje problem pretoka v omrežju na splošno in nato še malo bolj podrobno o maksimalnem pretoku. V tretjem poglavju so kronološko opisani algoritmi za reševanje problema maksimalnega pretoka. Četrto poglavje opisuje aproksimacijske algoritme za reševanje tega problema. V petem poglavju so zbrani še ostali algoritmi, ki rešujejo problem maksimalnega pretoka, a jih zaradi preobsežnosti nismo podrobneje predstavili. Zadnje poglavje pa vsebuje zaključne ugotovitve in oporne točke za nadaljnje delo.

Opisi algoritmov imajo enako strukturo: v prvem delu so predstavljeni avtorji, letnica objave članka in časovna zahtevnost algoritma. Sledijo ideja in metode algoritma. Na koncu pa je literatura našteta po abecednem vrstnem redu.

Poglavje 2

Splošno o pretokih

2.1 Problem najcenejšega pretoka

Problem najcenejšega pretoka[1] je najbolj temeljni od vseh problemov pretoka v omrežju. Pri tem problemu želimo najti najcenejšo pot po omrežju, da bi zadostili zahtevam nekaterih vozlišč po razpoložljivih zalogah v drugih vozliščih.

Naslednje posebne različice problema najcenejšega pretoka, povzete po Ahuja et. al. [2], imajo ključno vlogo v teoriji in uporabi pretokov v omrežju.

Problem najkrajše poti

Problem najkrajše poti[2] je najpreprostejši od vseh problemov pretoka v omrežju. Pri tem problemu želimo najti najkrajšo ali najcenejšo pot od izvora s do ponora t, pri čemer ima vsaka povezava svojo ceno. Rešitev problema najcenejšega pretoka bi poslala tok iz s skozi vsa ostala vozlišča i na najkrajši poti.

Problem maksimalnega pretoka

Problem maksimalnega pretoka[3] išče rešitev, ki pošlje maksimalno možno količino toka od izvora s do ponora t.

[1] ang. minimum cost flow problem
[2] ang. shortest path problem
[3] ang. maximum flow problem

Problem dodeljevanja

Podatki pri problemu dodeljevanja[4] sestojijo iz dveh enako velikih množic N_1 in N_2 ($|N_1| = |N_2|$). Vsak element $e_1 \in N_1$ želimo povezati z natanko enim elementom $e_2 \in N_2$, tako da bo celotna dodelitev elementov N_1 elementom N_2 najcenejša. Primer problema dodeljevanja je dodeljevanje ljudi na projekte, najemnikov v stanovanja, ...

Problem transporta

Problem transporta[5] je poseben problem najcenejšega pretoka. Vozlišča so razdeljena v dve, ne nujno enaki množici, povezave pa so le med vozlišči različnih množic. Množica N_1 predstavlja ponudnike in množica N_2 odjemalce, na primer: v N_1 so skladišča, v N_2 stranke, povezava $(i, j) \in N_1 \times N_2$ pa predstavlja distribucijski kanal med skladiščem i in stranko j.

Problem kroženja

Pri problemu kroženja[6] tok kroži po omrežju, ki nima izvora ali ponora. Primer takega problema je kroženje letala med mesti, kjer želimo zmanjšati ceno cikla.

Problem konveksnih stroškov pretoka

Pri problemu najcenejšega pretoka predpostavljamo, da cena toka narašča linearno s količino toka. Pri problemu konveksnih stroškov pretoka[7] pa je cena toka konveksna funkcija količine toka. Konveksnost funkcije lahko preverimo z drugim odvodom (če obstaja): če je drugi odvod funkcije na danem intervalu pozitiven, je funkcija na tem intervalu konveksna. Na primer, cena toka variira glede na izgubo energije v električnem omrežju zaradi upornosti vodnikov.

Problem splošnega pretoka

Pri problemu najcenejše poti povezave tok ohranjajo, nasprotno pa pri problemu splošnega pretoka[8] povezave lahko tok porabijo ali ustvarijo. Primeri tega problema so (1) izguba električne energije pri prenosu, (2) tok vode po ceveh se izgubi zaradi slabega tesnenja in (3) prevoz hitro pokvarljivega blaga.

[4] ang. assignment problem
[5] ang. transportation problem
[6] ang. circulation problem
[7] ang. convex cost flow problem
[8] ang. generalized flow problem

Problem pretoka večih stvari po isti poti

Pri problemu najcenejše poti po povezavah potuje le ena entiteta (voda ali elektrika ali plin ...), pri problemu pretoka večih stvari[9] pa potuje po istem osnovnem omrežju več stvari/ljudi, vendar z različnimi izvori in tudi različnimi ponori. Primeri tega problema so (1) transport potnikov z različnih izvorov na različne destinacije v mestu, (2) prenos podatkov v komunikacijskem omrežju med različnimi pari pošiljatelj - prejemnik in (3) transport žita iz držav, ki ga proizvajajo, v države, ki ga porabljajo.

2.2 Problem maksimalnega pretoka

Definicija omrežja:

Omrežje je označen, povezan in usmerjen graf s pozitivnimi cenami (kapacitetami) povezav, ki ima dve posebni vozlišči: izvor in ponor. Vsako povezavo v grafu si lahko predstavljamo kot kanal/cev, po katerem se pretaka tekočina, kapaciteta povezave pa je zmogljivost kanala. Tekočina teče po omrežju tako, da v nobenem kanalu ne preseže njegove kapacitete. Problem maksimalnega pretoka maksimizira količino tekočine, ki lahko teče od izvora do ponora.

Problem maksimalnega pretoka je klasičen optimizacijski problem, ki se ga lahko uporablja na več področjih (transport, urniki, vozni red . . .). V zadnjem času pa se uporablja tudi na nekaterih novih področjih: kodiranje omrežja in brezžična ad hoc omrežja.

Na splošno lahko pri reševanju maksimalnega pretoka uporabimo dve metodi: označevalno Ford - Fulkersonovo metodo[10] in metodo pošiljanja predtoka[11], ki sta jo predstavila Goldberg in Tarjan, potem, ko sta prvotno idejo o predtoku dobila od Karzanova.

[9] ang. multicommodity flow problem
[10] ang. labelling method
[11] ang. preflow-push method

Klasične metode za reševanje problema maksimalnega pretoka so: Ford - Fulkersonova metoda iskanja poti, ki jim lahko povečamo tok[12], zelo podobna metoda blokiranja pretoka[13] Dinitza in metoda, ki pošlje tok ponovno označenim[14], ki jo je predstavil Goldberg in kasneje izpopolnil s Tarjanom.

Veliko algoritmov za boljšo časovno zahtevnost uporablja skalirne metode (skaliranje kapacitete povezav ali presežka) in posebne podatkovne strukture. Skaliranje presežka, ki sta ga uporabila Ahuja in Orlin ter dinamična drevesna struktura Sleatorja in Tarjana omogočata pohitritev veliko algoritmov maksimalnega pretoka. Skaliranje je linearna preslikava, ki poveča ali zmanjša kapaciteto povezav ali presežek za faktor, ki ga imenujemo faktor skaliranja.

[12]ang. augmenting path
[13]ang. blocking flow method
[14]ang. push-relable method

Poglavje 3

Eksaktni algoritmi

3.1 Uvod

Notacija

Za definicijo časovne zahtevnosti se uporabljajo različne notacije: O, o, Ω, ω in θ. V tem poglavju smo uporabili le O notacijo, ki definira zgornjo mejo asimptotične rasti funkcije. Enotne so tudi ostale označbe: n pomeni število vozlišč, m število povezav, kapaciteta povezav pa je v območju $[1, \ldots, U]$.

3.2 George Dantzig (1951)

Dantzig (1914 - 2005) je bil ameriški matematik in profesor na univerzi Stanford. Njegova simpleksna metoda za reševanje linearnega programiranja reši problem maksimalnega pretoka v času $O(n^2mU)$.

Ideja algoritma

Dantzig je bil prvi, ki je formuliral problem maksimalnega pretoka kot linearni program. Sistem linearnih enačb pa je nato rešil s simpleksno metodo.

Bibliografske opombe

Simpleksne metode v tej diplomski nalogi ne bomo podrobneje opisali. Njen opis lahko preberete v Vašek [6]. Več o Dantzigovem algoritmu pa lahko preberete v Dantzig [7].

3.3 Lester Ford - Delbert Fulkerson (1956)

Ameriška matematika Lester Ford (rojen 1927) in Delbert Fulkerson (1924 - 1976) sta pokazala prvi algoritem za reševanje problema pretoka v omrežju. Algoritem sta Ford in Fulkerson[1] uporabila kot dokaz *max-flow-min-cut* trditve (Trditev *max-flow-min-cut* pravi, da je vrednost maksimalnega pretoka enaka kapaciteti minimalnega prereza.). Hitrost njunega algoritma pa je odvisna od kapacitete povezav v omrežju, števila vozlišč in povezav. Časovna zahtevnost je $O(nmU)$.

Ideja algoritma

Ford in Fulkerson sta predstavila metodo označevanja, ki povečuje pretok na poteh od izvora do ponora, ki jim lahko povečamo tok.

Algoritem FF najde maksimalen pretok tako, da na vsaki ponovitvi postopno povečuje trenutni tok s pošiljanjem dodatne količine toka na vsaki poti od izvora do ponora, kolikor dolgo je mogoče.

Trditev, ki sta jo dokazala Ford in Fulkerson pravi, da je trenutni pretok maksimalen, če ne obstaja nobena pot iz izvora do ponora, ki ji lahko povečamo tok. To velja pod pogojem, da so vhodni podatki - kapacitete in začetni tok - naravna števila, saj je potem tudi trenutni tok naravno število, in na koncu maksimalen. To pomeni, da je algoritem FF končen in psevdopolinomski za naravna števila.

Algoritem povečevanja toka na poti je prvi psevdopolinomski algoritem za problem maksimalnega pretoka. Za splošni problem, ko kapacitete in tokovi niso naravna števila, sta Ford in Fulkerson pokazala, da se njun algoritem včasih ne ustavi in vrednost pretoka takrat konvergira k $\frac{1}{4}$ maksimalnega toka. Pustila sta odprto vprašanje o obstoju polinomskega, ustavljivega in konvergentnega algoritma za splošni problem. To pa so kasneje neodvisno pokazali Edmonds in Karp leta 1969 (podpoglavje 3.4) ter Dinitz leta 1970 (podpoglavje 3.5).

[1] FF: algoritem Ford - Fulkerson

Bibliografske opombe

Algoritem FF je opisan v Dinitz [9], Ford in Fulkerson [12], Goldberg in Rao [15], Goldberg et al. [17], Pham et al. [19] ter Wilf [25]. Od vseh teh je v Dinitz [9] algoritem FF opisan najbolj razumljivo, poleg tega pa so tu opisane tudi njegove izboljšave.

3.4 Jack Edmonds - Richard Karp (1968)

Leta 1968 sta američana Jack Edmonds (rojen 1934) in Richard Karp (rojen 1935) predstavila prvi algoritem za problem pretoka v omrežju, ki je odvisen le od števila vozlišč in števila povezav. Dosegla sta časovno mejo $O(nm^2)$.

Dokazala sta ustavljivost in polinomski čas algoritma FF, če so uporabljene le najkrajše poti, ki jim lahko povečamo tok.

Ideja algoritma

V svojem algoritmu sta uporabila FF metodo označevanja in pokazala, da lahko napačna izbira poti, ki jim lahko povečamo tok, vodi do več računskih problemov. Da se izognemo tem težavam, sta v svojem članku Edmonds in Karp [10] podala pravila izbire.

Pokazala sta dve strategiji za izbiro naslednje poti, ki ji lahko povečamo tok v polinomskem času. Prva strategija je, da v vsaki ponovitvi izberemo najkrajšo pot, ki ji lahko povečamo tok. Pri tem je dolžina poti število povezav na poti. Druga strategija je, da v vsaki ponovitvi izberemo najdebelejšo pot, kjer je debelina poti minimum prostih kapacitet v povezavah na poti.

Bibliografske opombe

Algoritem je kot izboljšava algoritma FF opisan v Dinitz [9] ter v članku Edmonds in Karp [10].

3.5 Yefim Dinitz (1970)

E. A. Dinic je do leta 1990 deloval v Rusiji, nato se je preselil v Izrael in se preimenoval v Yefim Dinitz. V literaturi se uporabljata dva priimka, Dinic (do leta 1990) in Dinitz (od leta 1990). Ker se od leta 1990 uporablja Dinitz, bomo ta priimek uporabljali tudi mi.

Dinitzov algoritem[2], znan tudi kot Dinicev algoritem, je eden izmed prvih, (močno) polinomskih algoritmov za problem maksimalnega pretoka, ki je lahek za implementacijo in eden izmed hitrejših v praksi. V danem omrežju lahko izračunamo maksimalen pretok v času $O(n^2m)$.

Najbolj učinkoviti algoritmi za problem maksimalnega pretoka temeljijo na metodi blokiranja pretoka in metodi, ki pošlje tok ponovno označenim. Prvi algoritem, ki uporablja metodo blokiranja pretoka, je razvil prav Dinitz, in sicer v okviru metode iskanja poti, ki jim lahko povečamo tok.

Ideja algoritma

Metoda blokiranja pretoka:

metoda najde pot iz s v t in pošlje kolikor toka je mogoče, da zasiči vsaj eno povezavo na tej poti in nato odstrani zasičene povezave. To ponavlja dokler je t dosegljiv iz s.

Metoda iskanja poti, ki jim lahko povečamo tok:

neodvisno od Edmonds - Karpovega algoritma, je Dinitz predlagal iskanje poti, ki jim lahko povečamo tok, v fazah. Vsaka faza vsebuje več ponovitev spreminjanja toka z uporabo najkrajših poti, fiksne dolžine l. Tok najprej povečamo najkrajšim potem; s tem se dolžina najkrajših poti, ki jim lahko povečamo tok, povečuje. V vsaki fazi razširjen algoritem iskanja v širino[3] zgradi večplastno strukturirano omrežje L, kjer imajo vse poti dolžino l. Kot unija vseh najkrajših poti dolžine l, je večplastno omrežje prisotno čez celo fazo, dokler ne izgine. Po vsaki ponovitvi algoritma FF iz večplastnega omrežja odstranimo slepa vozlišča. Ko večplastno omrežje izgine, ne obstaja nobena pot, ki ji lahko povečamo tok dolžine $\leq l$ glede na trenutni tok. Zato bo dolžina poti v naslednjem večplastnem omrežju $> l$.

Bibliografske opombe

Algoritem je opisan v Dinic [8], Dinitz [9], Edmonds in Karp [10], Goldberg in Rao [15] ter Goldberg et al. [17]. Izmed vseh je ideja algoritma najbolj razumljivo opisana v Dinitz [9], kjer so opisane tudi njegove izboljšave; celoten algoritem pa je opisan v Dinic [8].

[2]DA: Dinitzov algoritem

[3]ang. BFS - Breadth First Search

3.5.1 Boris Cherkassky (1977)

Rus Boris Cherkassky je posplošil algoritem DA in dosegel časovno zahtevnost $O(n^2\sqrt{m})$.

Ideja algoritma

Cherkassky je odsvetoval gradnjo večplastnih omrežij, namesto tega je menil, da je zadosti izračunati le število plasti (rang) vsakega vozlišča $dist(v,t)$ do oddaljenosti $l = dist(s,t)$. To lahko dosežemo z uporabo algoritma BFS iz t na nenasičenih povezavah v nasprotni smeri (iz ponora t v izvor s).

Celotna faza je izvedena z enim iskanjem v globino[4] iz s. Namesto, da za vsako plast omrežja uporabimo seznam povezav, vsako zasičeno povezavo ali povezavo, ki ne gre iz vozlišča $ranga\, i$ v vozlišče $ranga\, i + 1$, preprosto preskočimo. Ker se DFS vrača po nekaterih povezavah, ki v nadaljevanju ne bodo več uporabljene, povezav ni potrebno odstranjevati.

Ko so izhodne povezave vozlišča $v \neq s, t$ zasičene (v postane slepo vozlišče), se DFS vrača nazaj. Ko s postane slepo vozlišče, se DFS in ta faza zaključi. Če DFS prispe v t, smo dobili pot, ki ji lahko povečamo tok. Tej poti tok povečamo in nato DFS nadaljuje na začetku povezave, ki je bila med zadnjo spremembo toka zasičena in je najbližje s.

Edini podatek, ki ga potrebujemo v vseh fazah, je preostala zmogljivost c_f vseh povezav. Po zadnji fazi pretok izračunamo kot razliko zmogljivosti povezav c in c_f.

Bibliografske opombe

Ideja algoritma je lepo opisana v Dinitz [9]. Članek je bil prvotno napisan v ruščini, v angleščino pa je bil preveden šele leta 1994 - Cherkassky [4]. Slednji je zelo tehničen in zato tudi težje berljiv.

3.5.2 Shimon Even - Alon Itai (1979)

Izraelca Shimon Even (1935 - 2004) in Alon Itai sta predstavila ključno izboljšavo algoritma DA - drugačen način iskanja poti, ki jim lahko povečamo tok: iskanje po principu DFS, v kombinaciji z odstranjevanjem slepih vozlišč.

Časovna zahtevnost njunega algoritma je enaka zahtevnosti algoritma DA, $O(n^2m)$.

[4] ang. DFS - Depth First Search

Ideja algoritma

V vsaki ponovitvi algoritma DFS zgradi pot, vozlišče za vozliščem, iz s. Takšna gradnja se ustavi, bodisi v t (uspeh), ali v slepem vozlišču. V drugem primeru se DFS vrne do vozlišča, ki je pripeljalo do tega slepega vozlišča, in iz L odstrani nekoristno vozlišče (nobena pot od s do t v L ne vsebuje tega vozlišča). Nato DFS nadaljuje kot je opisano zgoraj, z daljšanjem poti ter vračanjem in brisanjem, če je le-to potrebno. Proces se konča, bodisi v s, ko nima nobene povezave po kateri bi šel ali ko prispe v t. V slednjem primeru je zgrajena pot, ki ji lahko povečamo tok. Nato se po tej poti pošlje dodaten tok, kot pri algoritmu FF, in iz L odstrani vse nasičene povezave (vedno obstaja vsaj ena taka povezava).

Bibliografske opombe

Algoritem je podrobneje opisan v Dinitz [9] ter Even [11].

3.5.3 Zvi Galil - Amnon Naaman (1979)

Izraelca Zvi Galil (rojen 1947) in Amnon Naaman sta s posplošitvijo algoritma DA dosegla časovno zahtevnost $O(nm \log^2 n)$ za redke grafe.

Ideja algoritma

Predlagala sta posplošitev iskanja poti, ki jim lahko povečamo tok, s shranjevanjem dela prejšnjih poizvedb povečevanja toka na poti.

Bibliografske opombe

Algoritem je podrobneje opisan v Galil ter Naaman [13].

3.5.4 Daniel Sleator - Robert Tarjan (1983)

Američana Daniel Sleator in Robert Tarjan (rojen 1948) sta izboljšala Galil - Naamanov algoritem tako, da porabi $O(nm \log n)$ časa. To sta dosegla z uporabo dreves in ne le poti.

Ideja algoritma

Z uporabo dinamičnih drevesnih struktur algoritem najde poti, ki jim lahko povečamo tok in spremlja kapacitete povezav. Algoritem ohranja zbirko dreves, v katerih ima vsako vozlišče največ eno nenasičeno izhodno povezavo. Te povezave so kandidati za pot, ki ji lahko povečamo tok. Sprva je vsaka točka svoje drevo.

Bibliografske opombe

Algoritem je opisan v Sleator [21] ter Sleator in Tarjan [22].

3.6 Alexander Karzanov (1974)

Rus Alexander Karzanov je bil prvi, ki se pri iskanju maksimalnega pretoka ni osredotočil na idejo iskanja poti, ki jim lahko povečamo tok, ampak je predlagal metodo pošiljanja predtoka. Predtok je kot tok, le da tok, ki priteče v vozlišče, lahko preseže tok, ki odteče iz vozlišča. Tok, ki ostane v vozlišču, imenujemo *presežek*. Njegovo idejo sta kasneje uporabila tudi Goldberg in Tarjan, uporablja pa se tudi v drugih algoritmih za problem maksimalnega pretoka. Karzanov je bil tudi prvi, ki je predlagal, da je iskanje blokiranega pretoka ločen problem in predlagal uporabo predtokov za njegovo rešitev.

Algoritem ima časovno zahtevnost $O(n^3)$. To je bila v tistem času najboljša zahtevnost za goste grafe.

Ideja algoritma

Metoda pošiljanja predtoka:

predpostavimo, da nobena povezava nima negativnih vrednosti. Vozlišče je aktivno, če je presežek pozitiven. Naj bo vozlišče v aktivno, $e(v, u)$ pa naj bo moč pretoka, ki ga je še možno poslati po povezavi $v - u$. Priliv δ po e pošlje δ enot pretoka iz v do u. S tem se zmanjša $presežek(u)$ za δ in poveča $presežek(v)$ za δ.

Metoda blokiranja pretoka ima dve fazi, *Pošiljanje* in *Uravnovešanje*:

Pošiljanje uredi vozlišča po BFS zaporedju od s do t in pošlje čim več toka iz vozlišča v v vse njegove izhodne povezave. Če uspe vozlišče v, $v \neq s$ poslati ves tok iz vhodnih povezav, po izhodnih povezavah, se ustvari uravnovešen tok, drugače pa se ustvari *presežek*, ki se bo popravljal v drugi fazi. Ko se konča prva faza, je v nekaterih vozliščih *presežek*, ta tok imenujemo predtok.

V drugi fazi *Uravnovešanje* poišče vozlišča, ki imajo *presežek* in nimajo nezasičenih izhodnih povezav. *Uravnovešanje* pošlje tok iz teh vozlišč nazaj po vhodnih povezavah, s tem pa ustvari presežek v predhodnih vozliščih, kjer pa lahko obstajajo še druge nenasičene povezave. Po tem sledi novo *Pošiljanje* in novo *Uravnovešanje*, dokler ni uravnovešen predtok v vseh vozliščih razen s. Rezultat je zaželjeni blokiran pretok.

Bibliografske opombe

Algoritem je podrobneje opisan v Dinitz [9] ter Karzanov [18].

3.6.1 Gary Waissi (1992)

Profesor na Univerzi v Arizoni, GaryWaissi, je v svojem članku izboljšal Karzanov algoritem in dosegel časovno zahtevnost $O(n^3)$. Algoritem spada v tako imenovane fazne algoritme, in se uporablja za tip Dinitzovih večplastnih omrežjih. S časovno zahtevnostjo največ $O(n^3)$ je bil algoritem v tistem času eden izmed hitrejših algoritmov za problem maksimalnega pretoka v gostih omrežjih, kjer $m \approx n^2$. Časovna zahtevnost algoritma v acikličnih omrežjih pa je največ $O(n^2)$.

Ideja algoritma

Algoritem postopoma pretvori kombinacijo blokiranja predtokov in povratnih tokov v maksimalni pretok v omrežju. Z razliko od drugih algoritmov maksimalnega pretoka, ta algoritem obravnava omrežje bolj simetrično, ko poskuša povečati pretok na obeh korakih: *korak naprej* in *korak nazaj*.

Med *korakom naprej* se tok postopoma povečuje na vseh izhodnih povezavah izvora, od prve do zadnje plasti, plast za plastjo. Med *korakom nazaj* pa se tok postopoma povečuje na vseh vhodnih povezavah v ponor, v obratnem vrstnem redu, od ponora do izvora, in pošilja umetni pretok skozi omrežje.

Algoritem najde maksimalni pretok v dveh fazah. V prvi fazi se večplastno omrežje generira s pomočjo algoritma DA ali katerega koli drugega algoritma tega tipa, ki ohranja acikličnost večplastnih omrežij. V drugi fazi pa predlagani algoritem najde maksimalni pretok v tem omrežju. Ponavljajoče reševanje maksimalnega pretoka v tem omrežju pripelje do maksimalnega pretoka v originalnem omrežju.

Bibliografske opombe

Algoritem je opisan v članku Waissi [24].

3.7 Andrew Goldberg - Robert Tarjan (1986)

Američana Andrew Goldberg in Robert Tarjan (rojen 1948) sta izboljšala Goldbergov FIFO algoritem iz leta 1985, ki reši problem maksimalnega pretoka v času $O(n^3)$. V svojem algoritmu sta v celoti razvila metodo, ki pošlje tok ponovno označenim, ki jo je predhodno naznanil že Goldberg sam. Uporabila sta tudi metodo pošiljanja predtoka, ki ga je predhodno uporabil že Karzanov. Časovno zahtevnost sta na $O(nm \log(n^2/m))$ izboljšala tudi z uporabo dinamičnih drevesnih struktur.

Ideja algoritma

Ideja metode pošiljanja predtoka je, kot pri označevalni metodi, poiskati najkrajšo pot, ampak toka ne pošljemo po poti od izvora do ponora. Namesto tega procesiramo presežek v vozliščih: vozlišča pošljejo tok po svojih povezavah na osnovi znanja, ki ga ima vozlišče o sebi in sosednjih vozliščih. Ta algoritem sprejema odločitve na lokalni ravni in je s tem primeren tudi za porazdeljeno različico v asihronih omrežjih.

Vozliščem dodelimo različne višine. Izvor je najvišje, ponor je na dnu, ostala vozlišča pa so razporejena glede na oddaljenost od ponora. Pošiljanje je možno samo iz vozlišča, ki je višje, v vozlišče, ki je nižje.

Potek metode pošiljanja predtoka: izberemo aktivno vozlišče v iz G in poljubno dopustno povezavo $e \in \mathbf{G}$ ter pošljemo δ enot toka čez povezavo (izvorno vozlišče je sprva aktivno). Če ni več dopustnih povezav, vozlišče pa je še vedno aktivno, potem vozlišču dvignemo višino in s tem ustvarimo nove dopustne povezave in nadaljujemo s pošiljanjem. Ta zanka se ponavlja, dokler niso vsa vozlišča neaktivna.

Bibliografske opombe

Algoritem je lepo opisan v Goldberg in Tarjan [16] ter Pham et al. [19]. Leta 2005 so ga izpopolnili Pham, Lavallee, Bui in Do. Izpopolnitve algoritma so opisane v podpoglavju 3.9.

3.7.1 Ravindra Ahuja - James Orlin (1989)

Američana Ravindra Ahuja in James Orlin (rojen 1953) sta z uporabo skaliranja presežka izboljšala Goldberg - Tarjanovo metodo pošiljanja predtoka in metodo, ki pošlje tok ponovno označenim. Njun algoritem porabi $O(nm + n^2 \log U)$ časa.

Algoritem izvede skaliranje ponovitev. Tok pošljemo iz vozlišč z dovolj velikim presežkom v vozlišča z dovolj majhnim presežkom.

Ideja algoritma

Eden od načinov za uporabo predtoka je, da pošljemo tok iz vozlišča z *velikim presežkom* v vozlišče z *majhnim presežkom* ali v ponor. Posledica tega je hitro zmanjšanje največjega presežka.

Za natančnost metode je potrebno določiti, kdaj je presežek velik in kdaj majhen. V ta namen algoritem skaliranja uporablja *mejo presežka* $\triangle$ in celoštevilski skalirni faktor $k \geq 2$. Za vozlišče v rečemo, da ima *velik presežek*, če je njegov presežek večji od $\triangle/k$ in *majhen presežek* v nasprotnem primeru. Med izvajanjem algoritma je k fiksen, medtem ko se $\triangle$ periodično zmanjšuje. Sprva je $\triangle$ najmanjša potenca k, tako da $\triangle \geq U$.

Algoritem vsebuje več skalirnih faz, med katerimi je $\triangle$ konstanten. V vsaki fazi se izmenjujeta koraka *pošiljanje* in *ponovno označevanje* ter upoštevata zgornje pravilo. Ko nobeno aktivno vozlišče nima več *velikega presežka*, zamenjamo $\triangle$ z $\triangle/k$. Algoritem se zaključi, ko ni več aktivnih vozlišč.

Bibliografske opombe

Algoritem je opisan v Ahuja in Orlin [1], kasneje sta ga Ahuja in Orlin izpopolnila s Tarjanom (podpoglavje 3.8).

3.8 Ravindra Ahuja - James Orlin - Robert Tarjan (1989)

Američani Ravindra Ahuja, James Orlin (rojen 1953) in Robert Tarjan (rojen 1948) so s spremembami Ahuja - Orlinovega algoritma dobili časovno zahtevnost $O(nm+n^2\sqrt{\log U})$ brez uporabe dinamičnih dreves in $O(nm\log(\frac{n}{m}\sqrt{\log U}+2))$ z njihovo uporabo.

Ideja algoritma

Ahuja, Orlin in Tarjan so pokazali, da kombinacija Ahuja - Orlinovega algoritma, Tarjanovega algoritma valov[5] (opisan v Tarjan [23]) in uporaba dinamičnih dreves zmanjša prvotno časovno zahtevnost Ahuja - Orlinovega algoritma.

Algoritem valov:

metoda valov najde blokiran predtok in ga postopno pretvori v blokiran pretok z uravnoteženjem vozlišč, z zaporednimi *koraki naprej* in *nazaj* po omrežju. Vsako vozlišče je v enem od dveh stanj: neblokirano ali blokirano. Neblokirano vozlišče lahko postane blokirano, obratno pa ne. Vozlišče i uravnotežimo tako, da povečamo odhodni tok, če je i neblokirano in zmanjšamo dohodni tok, če je i blokirano.

Bibliografske opombe

Algoritem je opisan v članku Ahuja et al. [3].

[5] ang. wave algorithm

3.9 Thuy Lien Pham - Ivan Lavallee - Marc Bui - Si Hoang Do (2005)

Francozi Thuy Lien Pham, Ivan Lavallee, Marc Bui in Si Hoang Do so v svojem delu nadgradili Goldberg - Tarjanov algoritem (1986), ki je opisan v podpoglavju 3.7.

Predpostavili so, da vsako vozlišče pozna kapaciteto svojih vhodnih in izhodnih povezav, poleg tega pa izvorno vozlišče pozna število vseh vozlišč v grafu. Na vseh vozliščih se izvrši isti (lokalni) algoritem in vozlišča si izmenjujejo informacije s sosednjimi vozlišči, dokler ne dobimo maksimalnega pretoka. Algoritem se lahko uporablja tudi, kadar ima podan graf več izvorov in/ali ponorov.

Zahtevnost komunikacije je $O(n^2m)$, kar predstavlja vsa poslana sporočila. Časovna zahtevnost pa je izmerjena z maksimalnim številom izvedbe algoritma, ki je $O(n^2)$.

Ideja algoritma:

Vsako vozlišče v omrežju ima lahko svoj proces. Ta proces lahko z ostalimi procesi v sosednjih vozliščih komunicira direktno skozi dvosmerne komunikacijske povezave. V tem asihronem modelu vsak proces zaganja isti lokalni algoritem na lokalnih podatkih. Vozlišče lahko algoritem zažene v poljubnem trenutku ali ob prejetju sporočila, ki sproži zagon algoritma. Komunikacija poteka preko sporočil.

Algoritem se izvede v dveh fazah:

- inicializacija višine vozlišča

  Predpostavimo, da izvor pozna število vozlišč v grafu in označi višino ponora z 0, višino izvora pa z n. Višina ostalih vozlišč se izračuna po principu BFS, kjer se tudi ostalim vozliščem nastavi vrednost višine. Vozlišča si med seboj pošiljajo sporočila in, ko zadnje sporočilo prispe v izvor, le-ta nastavi svoj *presežek* tako, da zasiči svoje izhodne povezave. Nato gre algoritem v drugo fazo.

- presežek je poslan iz izvora navzdol proti ponoru

  Pošiljanje je možno le iz višjega vozlišča v nižje (v vozlišče z nižjo višino). Povratnica zadnjega sporočila poslanega v izvorno vozlišče, sproži proceduro, ki dovoli poslati *presežek* v sosednja vozlišča. Za izvor to pomeni, da obremeni svoje izhodne povezave. Ko tok prispe v vozlišče, postane *presežek* v vozlišču pozitiven, nato pa vozlišče ponovi postopek in pošlje svoj *presežek* v svoja sosednja vozlišča, ki imajo manjšo višino. Procedura pregleda izhodne povezave vozlišča in pošlje kolikor toka je mogoče. Ko vozlišče nima več nižje ležečih vozlišč (oz. ni več izhodnih povezav), tok postane lokalno ujet v vozlišču. Takemu vozlišču moramo dvigniti višino za toliko, da bo lahko poslal ves preostali *presežek*.

Razlika med tem in Goldberg - Tarjanovim algoritmom je v tem za koliko dvignemo vozlišče z ujetim tokom. Če vozlišče dvignemo le za toliko, da dobimo novo dopustno vozlišče in pošljemo njegov presežek po povezavi, potem se bo moralo to vozlišče dvigovati dokler bo njegov presežek pozitiven. Zato vozlišču dvignemo višino do praga, ki je ravno pravšnji, da lahko po njem pošlje ves svoj preostali presežek. Za hitro iskanje tega pragu, za vsako vozlišče hranimo listo sosednjih višin v naraščajočem vrstnem redu.

Ker smo v asihrono porazdeljenem omrežju, se lahko zgodi, da kdaj želi vozlišče poslati tok drugemu vozlišču, ki pa si je dvignil višino in je ta zdaj višja od pošiljateljevega. To se lahko zgodi le takrat, ko vozlišče prejme tok od vozlišča z nižjo višino - pošiljatelj še ni osvežil svoje nove višine. Prejemnik mora odgovoriti z neuspešnim sporočilom (pošiljanje toka ni dovoljeno) pošiljatelju z namenom, da opusti poslan tok. Sčasoma noben tok ne doseže več ponora. Ko nadaljujemo s premikanjem vozlišč navzgor, se preostali tok vrne nazaj v izvor. Algoritem se prekine, ko so vsi tokovi poslani v ponor ali vrnjeni v izvor.

Bibliografske opombe

Članek Pham et al. [19] je napisan zelo berljivo in ga priporočam tudi za bolj podrobno branje. V uvodu je lepo povzeto predhodnje delo na področju maksimalnega pretoka, nato pa sledi opis njihovega algoritma.

3.10 Ostalo

V literaturi smo zaznali tudi druge algoritme. Menimo, da niso imeli bistvenega pomena pri razvoju algoritma za problem maksimalnega pretoka in jih zato navajamo brez opisov.

V Indiji so leta 1978 Vishv Malhotra, Pramodh Kumar in Maheshwari vsebinsko izboljšali Karzanov algoritem, a tudi njihov algoritem porabi $O(n^3)$ časa. Izraelec Zvi Galil je istega leta posplošil algoritem Cherkasskya in dosegel časovno zahtevnost $O(n^{5/3}m^{2/3})$. Leto kasneje, 1979, je prav tako izraelec Yossi Shiloach na podlagi algoritma DA, odkril podoben algoritem kot Galil - Naamanov z isto časovno zahtevnostjo $O(nm \log^2 n)$. Američana Goldfarb in Rao sta leta 1988 zasnovala simpleksni algoritem, ki z največ nm pivoti reši problem maksimalnega pretoka v času $O(n^2m)$. Nemci Cheriyan, Hagerup in Mehlhornom so leta 1991 predstavili algoritem s časovno zahtevnostjo $O(n^3)$.

To so le nekateri algoritmi, ki smo jih zasledili v literaturi, zagotovo pa obstaja še več algoritmov, a v literaturi, ki smo jo pregledali, niso bili omenjeni.

3.11 Povzetek

	Leto	Raziskovalci	Časovna zahtevnost	Referenca	država
1	1951	Dantzig	$O(n^2mU)$	[7]	ZDA
2	1955	Ford - Fulkerson	$O(nmU)$	[9]	ZDA
3	1968	Edmonds - Karp	$O(nm^2)$	[9, 10]	ZDA
4	1970	Dinitz (Dinic)	$O(n^2m)$	[8]	Izrael (Rusija)
5	1974	Karzanov	$O(n^3)$	[9, 18]	Rusija
6	1977	Cherkassky	$O(n^2\sqrt{m})$	[4]	Rusija
7	1979	Even - Itai	$O(n^2m)$	[11]	Izrael
8	1979	Galil - Naaman	$O(nm\log^2 n)$	[13]	Izrael
9	1983	Sleator - Tarjan	$O(nm\log n)$	[21, 22]	ZDA
10	1985	Goldberg	$O(n^3)$	[14]	ZDA
11	1986	Goldberg - Tarjan	$O(nm\log(n^2/m))$	[16]	ZDA
12	1987	Ahuja - Orlin	$O(nm + n^2\log U)$	[1]	ZDA
13	1989	Ahuja - Orlin - Tarjan	$O(nm\log(\frac{n}{m}\sqrt{\log U} + 2))$	[3]	ZDA
14	1992	Waissi	$O(n^3)$	[24]	ZDA
15	2005	Pham - Lavallee - Bui - Do	$O(n^2)$	[19]	Francija

Tabela 3.1: Zgodovina problema maksimalnega pretoka

Poglavje 4

Aproksimacijski algoritmi

Ker se je reševanje problema maksimalnega pretoka le počasi izboljševalo, so popustili zahteve in se posvetili aproksimacijskim algoritmom.

4.1 Uvod

Po definiciji aproksimacijskega algoritma v Robič [20], je algoritem A *aproksimacijski algoritem* za problem $P = (I, S, m, cilj)$, če za vsako nalogo $x \in I$ vrne neko dopustno rešitev $A(x) \in S(x)$. Med dopustne rešitve štejemo tiste rešitve, katerih vrednost $m(x, A(x))$ se le malo razlikuje od optimalne vrednosti $m^*(x)$. Zato moramo ocenjevati razliko med vrednostjo $m(x, A(x))$ približne in vrednostjo $m^*(x)$ optimalne rešitve. Pri tem si lahko pomagamo z absolutno napako, relativno napako ali performančnim količnikom.

Za dani optimizacijski problem $P = (I, S, m, cilj)$, nalogo $x \in I$ in dopustno rešitev $y \in S(x)$ je:

- absolutna napaka rešitve y naloge x je $D(x, y) = |m^*(x) - m(x, y)|$. Pravimo, da je algoritem A *absolutni aproksimacijski algoritem za* P, če obstaja konstanta $k \geq 0$, da je $D(x, A(x)) \leq k$ za vsako nalogo $x \in I$.

- relativna napaka rešitve y naloge x je $E(x, y) = \frac{|m^*(x) - m(x,y)|}{max\{|m^*(x), m(x,y)|\}}$. Velja $0 \leq E(x, y) \leq 1$. Ko je $E(x, y) = 0$, je y optimalna rešitev, ko pa je $E(x, y) = 1$, je približna rešitev y slabša. Pravimo, da je algoritem A ϵ *aproksimacijski algoritem za* P, če obstaja konstanta $\epsilon \in [0, 1]$, da je $E(x, A(x)) \leq \epsilon$ za vsako nalogo $x \in I$.

- performančni količnik rešitve y naloge x je $R(x,y) = max\left\{\frac{m(x,y)}{m^*(x)}, \frac{m^*(x)}{m(x,y)}\right\}$. Velja $1 \leq R(x,y)$. Ko je $R(x,y) = 1$, je y optimalna rešitev, ko pa je rešitev y slabša, se $R(x,y)$ poljubno veča. Pravimo, da je algoritem A r - *aproksimacijski algoritem za* P, če obstaja konstanta $r \geq 1$, da je $R(x, A(x)) \leq r$ za vsako nalogo $x \in I$.

Notacija

Poleg standardne O notacije smo v tem razdelku uporabili tudi notacijo $\widetilde{O}(f(m)) = O(f(m)\log^c f(m))$. Ostale oznake ($n$, m in U) so enake kot v prejšnjem poglavju.

4.2 Andrew Goldberg - Satish Rao (1998)

Do leta 2010, ko so Christiano, Kelner, Madry, Spielman in Teng predstavili svoj algoritem, je bil to najhitrejši $(1-\epsilon)$ aproksimacijski algoritem, s časovno zahtevnostjo $\tilde{O}(m\sqrt{n}\epsilon^{-1})$.

Američana Andrew Goldberg in Satish Rao sta aproksimacijski algoritem razvila iz eksaktnega algoritma za izračun maksimalnega $s-t$ pretoka v usmerjenih in neusmerjenih grafih, ki ima časovno zahtevnost $O(min(n^{2/3}, m^{1/3})$ $m\log(n^2/m)\log U)$. Kar je veliko izboljšanje glede na prejšnje čase.

Ideja algoritma

Predstavila sta nov pristop za reševanje problema maksimalnega pretoka. Pristop temelji na dodeljevanju dolžin povezavam, ki so odvisne od preostalih vrednosti pretoka in preostalih zmogljivosti povezav.

Bibliografske opombe

Algoritem je podrobneje opisan v Goldberg in Rao [15].

4.3 Paul Christiano - Jonathan Kelner - Aleksander Madry - Daniel Spielman (2010)

Američani Paul Christiano, Jonathan Kelner, Aleksander Madry in Daniel Spielman (rojen 1970) so predstavili nov pristop za izračun aproksimacijskegaa maksimalnega pretoka v neusmerjenih grafih. Pretok se izračunava z reševanjem problema zaporednih električnih tokov. Vsak električni tok je podan kot rešitev sistema linearnih enačb v Laplacovi matriki in to se lahko približno izračuna v skoraj linearnem času.

Z uporabo tega pristopa so razvili do sedaj najhitrejši znan aproksimacijski algoritem za izračun maksimalnega $s-t$ pretoka. Za graf, ki ima n vozlišč in m povezav, algoritem izračuna $(1-\epsilon)$ aproksimacijski maksimalni pretok v času $\widetilde{O}(mn^{1/3}\epsilon^{-11/3})$. Pred tem algoritmom je bil najhitrejši algoritem, ki je odvisen le od n in m, algoritem Goldberg - Rao (1998) (opisan v 4.2), ki izračuna aproksimacijski maksimalen pretok v času $\widetilde{O}(m\sqrt{n}\epsilon^{-1})$.

Ideja algoritma

Pokazali so, da lahko aproksimacijsko rešitev za vsak električni pretok najdemo v času $\widetilde{O}(m)$ z uporabo algoritma za reševanje sistema linearnih enačb v Laplacovih matrikah.

Vsako povezavo vhodnega grafa si predstavljamo kot upornik z uporom ena in nato izračunamo električni pretok, ki nastane, ko pošljemo tok od izvora s do ponora t. Ta tok upošteva omejitve pretoka, ampak lahko ne upošteva kapacitete povezav. Da bi to odpravili, so povečali upor vsake povezave v sorazmerju s količino toka, ki teče skozi njo - s tem se kaznuje povezave, ki so presegle svoje sposobnosti - in izračuna električni pretok z novimi upori.

To so kombinirali še s skalirnimi tehnikami in dobili $(1-\epsilon)$ - aproksimacijski maksimalen $s-t$ pretok v času $\widetilde{O}(mn^{1/3}\epsilon^{-11/3})$.

Bibliografske opombe

Algoritem je predstavljen v Christiano et al. [5], kjer so avtorji pustili odprto vprašanje o razširitvi algoritma na usmerjene grafe in upajo, da bo ravno njihov algoritem pripeljal do aproksimacijskega algoritma, ki reši problem maksimalnega pretoka v skoraj linearnem času.

Poglavje 5

Ostali

5.1 Uvod

V literaturi smo zasledili tudi druge pristope reševanja problema maksimalnega pretoka, vendar jih bomo zaradi preobsežnosti diplomske naloge tu le omenili.

5.2 Verjetnostni algoritmi

Izraelec Noga Alon je leta 1989 predstavil algoritem, ki temelji na permutacijah in ima časovno zahtevnost $O(nm + n^{8/3} \log n)$.

Nemca Cheriyan in Hagerup sta leta 1990 predstavila naključni nedeterministični algoritem, katerega časovna zahtevnost je $O(nm + n^2 \log^2 n)$.

Valerie King, Satish Rao in Robert Tarjan so leta 1991 na podlagi algoritma Cheriyan - Hagerup predstavili izboljšan deterministični algoritem s časovno zahtevnostjo $O(mn(\log_{m/n \log n} n)$, ki so ga uporabili za igro *ubij vozlišče*[1]. Pravila igre so opisana v članku Phillips - Westbrook iz leta 1993 (Online load balancing and network flow). Prav v tem članku sta Steven Phillips in Jeffery Westbrook predstavila prav tako deterministično algoritem za to igro, s časovno zahtevnostjo $O(nm(\log_{m/n} n + n^2 \log^{2+\epsilon} n))$.

[1] ang. the node kill game

5.3 Vzporedni algoritmi

Algoritem Shiloach - Vishkin iz leta 1982 je bil prvi vzporedni algoritem. Algoritem na n procesorjih porabi $O(n^2 \log n)$ časa in $O(nm)$ prostora. Kasneje, leta 1986, je Vishkin sam izboljšal porabo prostora na $O(n^2)$.

Ta algoritem je vplival tudi na delo Goldberga in Tarjana. Vzporedna implementacija algoritma, ki smo ga že opisali v podpoglavju 3.7, na n procesorjih, porabi $O(n^2 \log n)$ časa in $O(m)$ prostora.

Indijec Ravinda Ahuja in angležJames Orlin sta prav tako predstavila vzporedno implementacijo algoritma za problem maksimalnega pretoka, ki porabi $O(n^2 \log U \log p)$ časa, v modelu PRAM, s pravicami EREW na p procesorjih, kjer je $\lceil p = m/n \rceil$.

5.4 Porazdeljeni algoritmi

Leta 1995 je čehinja Lenka Motyčkova predstavila porazdeljen algoritem za problem maksimalnega pretoka s časovno zahtevnostjo $O(n^2 \log^3 n)$ in komunikacijsko zahtevnostjo $O(n^2(\log^3 n + \sqrt{m}))$.

Algoritem Pham - Lavallee - Bui - Do, opisan v 3.9, je porazdeljen algoritem za problem maksimalnega pretoka, ki temelji na Goldberg - Tarjanovi metodi pošiljanja predtoka. Algoritem ima časovno zahtevnost $O(n^2)$ in zahtevnost komunikacije $O(n^2 m)$.

Poglavje 6

Zaključek

V diplomskem delu smo želeli predstaviti razvoj algoritmov za problem maksimalnega pretoka.

6.1 Ugotovitve

Za reševanje problema maksimalnega pretoka obstaja veliko algoritmov. Razvijati so jih začeli že v sredini 20. stoletja, ko so se pojavile potrebe po čim hitrejšemu pretoku surovin. V nekaterih virih smo zasledili, da je bil prvi, ki je začel z iskanjem maksimalnega pretoka v omrežju, Tolstoj. Leta 1930 je v svojem članku opisal problem transporta v Sovjetski uniji. Idejo za to je dobil v Sovjetskem železniškem sistemu in prevozu različnih surovin med mesti. Leta 1951 pa je bil Dantzig prvi, ki je definiral problem maksimalnega pretoka.

Sprva so bili algoritmi samo eksaktni, kasneje so zaradi zahtevnosti problema začeli razvijati tudi aproksimacijske algoritme in se tako hoteli čim bolj približati linearni časovni zahtevnosti.

6.2 Nadaljnje delo

Za nadaljnje delo bi predlagali pregled vzporednih, verjetnostnih ali porazdeljenih algoritmov, ki smo jih v tem diplomskem delu le omenili. Zanimivo bi bilo preučiti, če lahko predstavljene algoritme paraliziramo in nato dobljene časovne zahtevnosti primerjamo z neparaliziranimi, ki so opisane v tej diplomski nalogi.

Literatura

[1] R. K. Ahuja and J. B. Orlin, "A fast and simple algorithm for maximum flow problem", *Operations Ressearch*, št. 37, str. 748-759, 1989.

[2] R. K. Ahuja, T. L. Magnanti and J. B. Orlin. *Network flows: Theory, Algorithms and Applications*. Prentice Hall, Upper Saddle River, New Jersey, 1993.

[3] R. K. Ahuja, J. B. Orlin and R. E. Tarjan, "Improved time bounds for the maximum flow problem", *SIAM Journal Computing*, št. 18, zv. 5, str. 939-954, oktober 1989.

[4] B.V. Cherkassky, "A fast algorithm for constructing a maximal flow through a network", *American Mathematical Society Translation*, št. 158, zv. 2, str. 23-30, 1994.

[5] P. Christiano, J. A. Kelner, A. Madry, D. Spielman and S. H. Teng, "Electrical Flows, Laplacian Systems, and Faster Approximation of Maximum Flow in Undirected Graphs", *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, oktober 2010.

[6] Vašek Chvatal, *Linear programming*. W. H. Freeman, 1983.

[7] G.B. Dantzig, "Application of the Simplex Method to a transportation problem", v T. C. Koopmans, *Cowles commission for research in economics*, št. 13, pogl. XXIII, str. 359-373, John Wiley & Sons New York, London 1951

[8] E. A. Dinic, "Algorithm for solution of a problem in networks with power estimation", *Soviet Mathematics Doklady*, št. 11, str. 1277-1280, 1970.

[9] Y. Dinitz, "Dinitz' Algorithm: The Original Version and Even's Version", *Theoretical Computer Science: Essays in Memory of Shimon Even, Oded*

Goldreich, Arnold L. Rosenberg, and Alan L. Selman, Eds. LNCS Festschrift, št. 3895, str. 218-240, Springer-Verlag, 2006.

[10] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems", *Journal of Association for Computing Machinery*, št. 19, zv. 2, str. 248-264, april 1972.

[11] S. Even, *Graph Algorithms*, Computer Science Press, Rockville, MD, 1979.

[12] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network", *Canadian Journal of Mathematics*, št. 8, str. 399-404, 1956.

[13] Z. Galil and A. Naaman, "An $O(EV \log^2 V)$ algorithm for the maximal flow problem", *Journal of Computer and System Sciences*, št. 21, str. 203-217, 1980.

[14] A. V. Goldberg, "A new max-flow algorithm", *Tehnical report MIT/LCS/TM-291*, Laboratory for computer science, MIT, Cambridge, Massachusetts, 1985.

[15] A. V. Goldberg and S. Rao, "Beyond the flow decomposition barrier", *Journal of the Association for Computing Machinery*, št. 45, zv. 5, str. 783-797, september 1998.

[16] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem", *Journal of Association for Computing Machinery*, št. 35, zv. 4, str. 921-940, oktober 1988.

[17] A. V. Goldberg, E. Tardos and R. E. Tarjan, "Network flow algorithms", *Algorithms and combinatorics*, št. 9, 1990.

[18] A.V. Karzanov, "Determining the maximum flow in a network by the method of preflows", *Soviet Mathematics Doklady*, št. 15, str. 434-437, 1974.

[19] T. L. Pham, I. Lavallee, M. Bui, S. H. Do, "A distributed algorithm for the maximum flow problem", *Proceedings of the 4th International Symposium on Parallel and Distributed Computing*, 2005.

[20] B. Robič. *Aproksimacijski algoritmi.* Založba FE in FRI, 2009.

[21] D. D. Sleator, "An $O(mn \log n)$ algorithm for maximum network flow", *Tehnical report*, STAN-CS-80-831, Computer Science department, Stanford University, Stanford, California, 1980.

[22] D. D. Sleator and R. E. Tarjan, "A data structure for dynamic trees", *Journal of Computer and System Sciences*, št. 26, str. 362-391, 1983.

[23] R. E. Tarjan, *Data Structures and Network Algorithms*, SIAM, Philadelphia, Pennsylvania, 1983.

[24] G. R. Waissi, "A new Karzanov-type $O(n^3)$ max-flow algorithm", *Mathematical and Computer Modelling*, št. 16, zv. 2, str. 65-72, 1992.

[25] H. S. Wilf, *Algorithms and complexity*. University of Pennsylvania, Philadelphia, Pennsylvania, 1994.

Stvarno kazalo