Moderní algoritmy pro hledání maximálního toku v síti

PALUBJÁK, Petr

Ing., ✉ Ústav automatizace a informatiky, FSI VUT Brno, Technická 2, Brno, 616 69

palubjak@kn.vutbr.cz

Abstrakt: *Článek je věnován problematice hledání maximálního toku v sítích obecného tvaru. Pro řešení tohoto problému, který vyplynul z praxe při hledání optimální dopravy produktů ke spotřebiteli, existuje celá skupina algoritmů. V článku jsou zmíněny pouze algoritmy, které přinesly určitou revoluci v efektivnosti hledání maximálního toku v síti. Mimo obecný úvod do problematiky a srovnání jednotlivých algoritmů je největší důraz kladen na tzv. Algoritmus tří Indů (V.M. Malhotra, M. Pramodth Kumar, S. N. Maheshwari), který dodnes dosahuje nejlepších výsledků při řešení v hustých sítích. V článku je podrobně popsána jeho architektura a je porovnána jeho rychlost s ostatními algoritmy na různých typech sítí.*

Klíčová slova: *Toky v síti, Maximální tok, Algoritmus tří Indů*

1 Úvod

Pojem toky v sítích je abstrakcí zažitých fyzikálních toků (tok vody, elektřiny, automobilů, dat, materiálu apod.) v příslušných sítích (vodovodní, elektrické, dopravní, počítačové apod.). Důležitým podnětem pro rozvoj teorie byly problémy vznikající z dopravy produktů ke spotřebitelům. Takovéto úlohy lze formulovat jako úlohy lineárního programování. Jedná se ovšem o dost speciální úlohy, a proto se oplatí hledat speciální algoritmy pro jejich řešení. Všeobecně lze pomocí toků v sítích lehce formulovat i řadu zajímavých a důležitých kombinatorických úloh, které často vůbec nemají souvislost s reálnými toky.

Pro algoritmy zmíněné v tomto článku byl vytvořen program *Toky v sitich* v jazyku Delphi, který umožňuje porovnat jednotlivé algoritmy na různých typech sítí a demonstrovat postup výpočtu algoritmů .

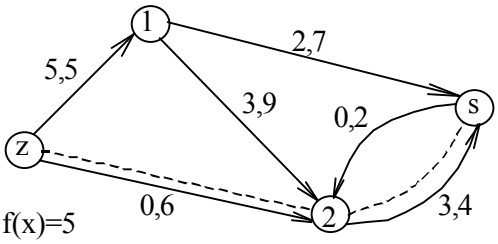
2 Oblasti použití

Prvotní oblastí, která v podstatě inicializovala vznik teorie kolem toků v sítích, bylo řešení problémů spojených s dopravou produktů ke spotřebitelům. Kromě původního požadavku na přepravu maximálního množství hmoty při omezené propustnosti přepravních linek se začala požadovat i ekonomičnost přepravy. V současnosti se této teorie využívá při optimalizaci s úspěchem i v jiných oblastech, např. při návrhu a provozu přepojovaných počítačových sítí.

3 Úvod do problematiky hledání maximálního toku v síti

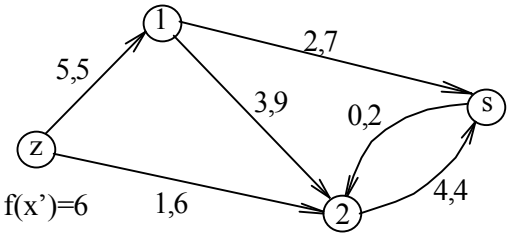
Protože kapacitní omezení(c) a podmínky kontinuity jsou lineární (v x_{ij}) a účelová funkce $f(x)$ je též lineární, je úloha o maximálním toku (mezi zdrojem z a spotřebičem s) úlohou lineárního programování. Jde ale o úlohu velmi speciální, a proto má smysl hledat speciální algoritmy pro její řešení. V této úloze lineárního programování je množina přípustných řešení neprázdná (např. existuje nulový tok, tj. $x_{ij} = 0$ pro všechny $ij \in H(G)$), uzavřená a ohraničená

(kapacitně). Proto spojitá (dokonce lineární) funkce $f(x)$ má maximum. Jinak řečeno, maximální tok vždy existuje.

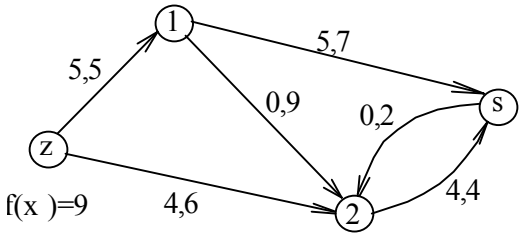


Obr. 3.1a: Tok x (druhé číslo udává kapacitu hrany).

Příklad na obrázku 3.1a ukazuje tok x o velikosti 5. Na každé hraně ij je dvojice čísel (x_{ij}, c_{ij}). Na obrázku 3.1b je tok x', který jsme získali z toku x tak, že na cestě z-2-s jsme zvětšili tok x o jednotku. Ovšem tímto způsobem již nelze získat větší tok z x', protože na každé z–s cestě existuje aspoň jedna hrana ij taková, že $x_{ij}' = c_{ij}$. Takováto hrana se nazývá nasycená a cesta obsahující nasycenou hranu blokovaná.



Obr. 3.1b: Tok x' získaný z x zvětšením po cestě z-2-s.



Obr. 3.1c: Maximální tok.

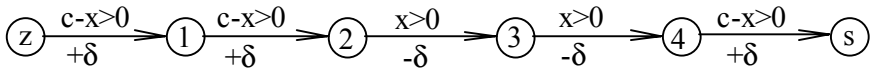
Bohužel, blokující tok nemusí být tokem maximálním. Z obrázku 3.1c je vidět, že tok lze ještě zvětšit přes hranu vzad. Blokujícím tokem nelze aproximovat maximální tok dokonce ani v necyklickém grafu, kde jsou všechny kapacity jednotkové.

Věta o maximálním toku a minimálním řezu

Jedním z prvních, kdo navrhli silnější prostředek na zvětšování daného toku x byli pánové Ford a Fulkerson (1957). Nechť P je nějaká cesta, nechť P^+ je množina hran orientovaných ve směru cesty P a P^- je množina hran orientovaných proti směru cesty P. Čísla $c_{ij} - x_{ij}$ pro $ij \in P^+$ a x_{ij} pro P^- se nazývají rezervy na cestě P. Minimální z rezerv na cestě P se nazývá rezerva cesty P. Když je rezerva nějaké cesty P kladná, tak se P označuje jako rezervní cesta. Takovéto označení je oprávněné, protože je-li $\delta > 0$ rezerva z–s cesty P, tak funkce

$$x': H(G) \to R, \quad kde \quad x'_{ij} = \begin{cases} x_{ij} + \delta, & pro \ \ ij \in P^+ \\ x_{ij} - \delta, & pro \ \ ij \in P^- \\ x_{ij} \end{cases}$$

jinak je opět tokem v G a $f(x') = f(x) + \delta$. Totiž kapacitní omezení i podmínky kontinuity se dodrží, jak je vidět z příkladu na obrázku 3.2 (např. ve vrcholu 1 bude o δ víc přitékat, ale i o δ víc odtékat).

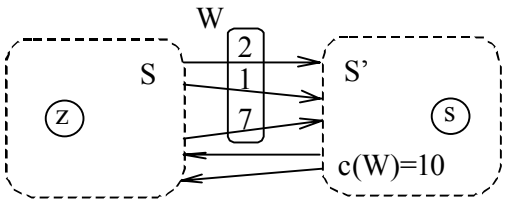


Obr. 3.2: Zlepšující cesta.

Jestliže G je graf a $A,B \subseteq V(G)$, tak $(A,B)_G = (AxB) \cap H(G)$. Nechť $S \subset V(G)$, $S'=V(G) - S$ a nechť $z \in S$ a $s \in S'$. Potom množinu hran $W = (S, S')_G$ nazýváme (hranovým) z–s řezem. Je zřejmé, že každá z–s cesta obsahuje aspoň jednu hranu z libovolného z–s řezu. Obrázek 3.3 ilustruje tento pojem. Kapacita $c(W)$ řezu W je součet kapacit jednotlivých hran řezu. Řez s minimální kapacitou se označuje jako řez minimální.

Pro libovolnou funkci $h : H(G) \to R$ a množiny $A,B \subseteq V(G)$ budeme stručně psát: $h(A,B) = \sum_{ij \in (A,B)_G} h_{ij}$

Jestliže x je nějaký z–s tok a $(S, S')_G$ je nějaký z–s řez, tak zřejmě platí vztah $f(x) = x(S, S') - x(S', S) \leq c(S, S')$, protože platí podmínky kontinuity a kapacitní omezení. Tato nerovnost nám připomíná tzv. slabou větu o dualitě v lineárním programování. Proto očekáváme, že v extrémních případech dosáhneme rovnosti. Opravdu tomu tak je a následující věta může být odvozena i z věty o dualitě v lineárním programování, ale tady upřednostníme původní grafový důkaz (jak ho podali Ford a Fulkerson), který je jednodušší a konstruktivnější.



Obr. 3.3: z-s řez kapacity 10.

Věta 1. Velikost maximálního z–s toku se rovná kapacitě minimálního z–s řezu.

Důkaz: Nejdříve poznamenejme, že věta má vždy smysl, protože maximální tok vždy existuje a minimální řez též.

Nechť x^* je maximální z–s tok a c^* nechť je kapacita minimálního z–s řezu. Podle dříve uvedeného vztahu je $f(x^*) \leq c^*$. Nyní najdeme takový z–s řez W, že $f(x^*) = c(W)$, a tím bude důkaz hotový. Nechť $S \subseteq V(G)$ je množina vrcholů obsahující zdroj z a každý takový vrchol v, pro který existuje rezervní z–v cesta. Je zřejmé, že $s \notin S$, protože jinak by existovala zlepšující z–s cesta pro x^*, což protiřečí maximalitě toku x^*. Proto je tedy $s \in S'$. Každá hrana ij jdoucí z S do S' je nasycená , protože jinak by z existence rezervní z–i cesty vyplývala existence rezervní z–j cesty, což je nemožné, protože $j \in S'$. Analogicky na každé hraně vedoucí z S' do S je tok x nulový. Takto $f(x^*) = x^*(S, S') = c(S, S')$.

Z algoritmického hlediska je důležité následující tvrzení (Ford-Fulkerson 1957).

Věta o optimalitě toku. Nějaký z–s tok x je maximální $\leftrightarrow$ neexistuje rezervní z–s cesta pro x.

Důkaz: ($\rightarrow$) Pomocí zvětšující cesty by se dal tok x zvětšit.
($\leftarrow$) Nechť $S = \{ v \mid$ existuje rezervní z–v cesta pro $x \}$. Potom tak jako v předešlém důkazu vidíme, že $f(x) = c(S, S')$, a proto x je maximální tok a $(S, S')_G$ minimální řez.

4 Historický vývoj významných algoritmů

Prvním algoritmem pro hledání maximálního toku byl Ford-Fulkersonův (1957) algoritmus. Jako jediný nezaručuje nalezení maximálního toku (v případě neceločíselných kapacit). Maximální tok určuje opakovaným hledáním rezervních *z-s* cest, jejichž pořadí není specifikováno. Tento nedostatek vyřešili ve svých modifikovaných algoritmech pánové Edmonds a Karp (1972), kteří hledají vždy nejkratší rezervní cestu nebo rezervní cestu s maximální kapacitou. Významný pokrok přinesl Dinicův (1972) algoritmus, který převádí problém hledání maximálního toku na opakované hledání nasyceného toku v síti speciálního tvaru. Z Dinicova algoritmu potom vychází Algoritmus tří Indů pánů Malhotry, Pramodh Kumara a Maheshwariho (1978), kteří zefektivnili část algoritmu pro hledání nasyceného toku. Jejich algoritmus patří dodnes mezi nejrychlejší při hledání maximálního toku v hustých sítích. S revolučním algoritmem přišel v roce 1988 Goldberg. Tento algoritmus dosahuje dobrých výsledků jak v řídkých tak i v hustých sítích.

5 Algoritmus tří Indů

Algoritmus tří Indů patří dodnes mezi nejrychlejší při řešení hustých grafů, kde počet uzlů~(počet hran)^2. Je rychlejší než všechny dosud uvedené algoritmy (Ford-Fulkerson, Edmonds-Karp, Dinic). Tento algoritmus se jmenuje podle svých indických tvůrců, kterými byli pánové V. M. Malhotra, M. Pramodh Kumar a S. N. Maheshwari. Algoritmus vychází z podobného algoritmu navrženého A. V. Karzanovem. Jeho algoritmus pracuje s tzv. předtokem, který se potom vyrovná na tok. Karzanovův algoritmus umožňuje nalezení maximálního toku v čase $O(n^3)$. Je to ovšem algoritmus velmi složitý a právě algoritmus tří Indů je stejně rychlý jako Karzanovův algoritmus a přitom je o trochu jednodušší.

Popis algoritmu:

Vstupní data: Vrstvená síť $S = (G, z, s, c)$.

Úkol: Nalézt nasycený tok t v síti S.

Pomocné proměnné: Každý vrchol a každá hrana grafu G může být otevřená nebo uzavřená, dále se použijí M^+, M^-, fronty obsahující vrcholy grafu G, pro každý vrchol v čísla $r_{in}(v)$, $r_{out}(v)$, $r(v)$, $f^+(v)$, $f^-(v)$ (M^+ obsahuje vrcholy v, pro něž je třeba zvýšit tok hranami z nich vycházejícími o celkově $f^+(v)$, obdobně M^- obsahuje vrcholy v, pro něž je třeba zvýšit tok hranami do nich vstupujícími o celkově $f^-(v)$), R^+, R^-, fronty obsahující vrcholy grafu G(R^+ obsahuje vrcholy určené k uzavření, protože z nich vycházejí jen uzavřené hrany, R^- obsahuje vrcholy určené k uzavření, protože do nich vstupují jen uzavřené hrany).

1. [*Inicializace.*]
   Nastavit přípustný tok t na nulový tok; každý vrchol i hrana jsou otevřené; pro každý vrchol položíme:
   $r_{in}(v) = \sum (c(h) - t(h))$,
   kde součet se bere přes všechny otevřené hrany končící ve vrcholu v,

obdobně se určí

$r_{out}(v) = \sum (c(h) - t(h))$,

kde součet se bere přes všechny otevřené hrany začínající ve vrcholu v.

2. [*Určení rezerv vrcholů.*]

Položíme:

$r(z) = r_{out}(z)$; $r(s) = r_{in}(s)$;

$r(v) = \min(r_{in}(v), r_{out}(v))$, pokud $z \neq v \neq s$.

Číslo $r(v)$ udává, o kolik je možno zvýšit tok procházející vrcholem v.

3. [*Určení referenčního vrcholu.*]

Z otevřených vrcholů vybereme ten, který má nejmenší hodnotu $r(v)$, a označíme jej v_0. Přitom zrovna nastavíme u všech vrcholů hodnotu $f^+(v)$ a $f(v)$ na nulu.

Položíme $f^+(v_0) = r(v_0)$ a $f(v_0) = r(v_0)$.

Je-li $v_0 \neq s$, položíme $M^+ = \langle v_0 \rangle$, jinak je M^+ prázdná fronta.

Je-li $v_0 \neq z$, položíme $M^- = \langle v_0 \rangle$, jinak je M^- prázdná fronta.

4. [*Zvětšení toku.*]

Tok ze zdroje do referenčního vrcholu v_0 i tok z v_0 do spotřebiče zvýšíme o r_0. Tuto operaci lze provést proto, že ostatní vnitřní vrcholy sítě mají rezervu pro zvýšení toku jimi procházejícího alespoň o r_0.

Modifikaci toku provedeme takto:

4a. [*Zvětšení toku směrem ke spotřebiči.*]

Nechť v je první vrchol z fronty M^+. Zvolíme otevřenou hranu (v, w) vycházející z vrcholu v a položíme:

$d = \min (f^+(v), c(v, w) - t(v, w))$.

Potom změníme hodnoty popisující příslušné vrcholy hrany. Čísla $t(v, w)$ a $f^+(w)$ zvýšíme o d a čísla $f^+(v)$, $r_{out}(v)$ a $r_{in}(w)$ snížíme o d. Pokud $w \neq s$ a w není ve frontě M^+, přidáme w na konec fronty M^+. Je-li nyní $f^+(v) = 0$, vyjmeme v z M^+. Je-li $c(v, w) = t(v, w)$, provedeme následující tři příkazy:

- uzavřeme hranu (v, w),
- jestliže z vrcholu v nevychází žádná otevřená hrana, přidáme v na konec fronty R^+,
- jestliže do vrcholu w nevstupuje žádná otevřená hrana, přidáme w na konec fronty R^-.

4b. [*Test fronty M^+.*]

Pokud fronta M^+ není prázdná, pokračuj bodem 4a.

4c. [*Zvětšení toku směrem ke zdroji.*]

Nechť v je první vrchol z fronty M^-. Zvolíme otevřenou hranu (u, v) vycházející z vrcholu v a položíme:

$d = \min (f(v), c(u, v) - t(u, v))$.

Potom změníme hodnoty popisující příslušné vrcholy hrany. Čísla $t(u, v)$ a $f(u)$ zvýšíme o d a čísla $f(v)$, $r_{out}(u)$ a $r_{in}(v)$ snížíme o d. Pokud $u \neq z$ a u není ve frontě přidáme u na konec fronty M^-. Je-li nyní $f(v) = 0$, vyjmeme vrchol v z M^-. Je-li $c(u,v) = t(u, v)$, provedeme následující tři příkazy:

- uzavřeme hranu (u, v).
- jestliže z vrcholu u nevychází žádná otevřená hrana, přidáme u na konec fronty R^+.
- jestliže do vrcholu v nevstupuje žádná otevřená hrana, přidáme v na konec fronty R^-.

4d. [*Test fronty* M^-.]
Pokud fronta M^- není prázdná, pokračuj bodem 4c.

5. [*Uzavírání hran a vrcholů.*]

5a. [*Zavírání vrcholů (a hran) bez výstupních hran.*]
Nechť v je první vrchol ve frontě R^+. Uzavřeme vrchol v a vyjmeme jej z fronty R^+. Uzavřeme všechny hrany vstupující do vrcholu v a pokud přitom vzniknou vrcholy, z nichž nevycházejí otevřené hrany, přidáme je na konec fronty R^+.

5b. [*Test fronty* R^+.]
Pokud fronta R^+ není prázdná, pokračuj bodem 5a.

5c. [*Zavírání vrcholů (a hran) bez vstupních hran.*]
Nechť v je první vrchol ve frontě R^-. Uzavřeme vrchol v a vyjmeme jej z fronty R^-. Uzavřeme všechny hrany vystupující z vrcholu v a pokud přitom vzniknou vrcholy, do nichž nevstupují otevřené hrany, přidáme je na konec fronty R^-.

5d. [*Test fronty* R^-.]
Pokud fronta R^- není prázdná, pokračuj bodem 5c.

6. [*Test ukončení.*]
Je-li zdroj uzavřen, výpočet je hotov a tok t je nasycený tok. V opačném případě pokračujeme bodem 2.

Jak je vidět ze vstupních dat, algoritmus pracuje s vrstvenou sítí. Proto pro práci se sítí obecného tvaru musí být doplněn o část, která „připraví“ algoritmu síť a nechá proběhnout výpočet nad vrstvenou sítí.

Popis algoritmu pro práci se síti obecného tvaru:

Vstupní data: Síť $S = (G, s, z, c)$.

Úkol: Nalézt maximální tok t v obecné síti S.

Pomocné proměnné: S' a S'', sítě se stejnou množinou vrcholů a týmž zdrojem a spotřebičem jako S; q, tok v síti S''.

1. [*Inicializace.*]
Pro všechny hrany h grafu G položíme $t(h) = 0$.

2. [*Vytvoření sítě S'.*]
Vytvoří se síť taková, že uspořádaná dvojice (v, w) je její hranou jestliže:

buď $(v,w) \in H(G)$ a $c(v, w) - t(v, w) > 0$,
nebo $(w,v) \in H(G)$ a $t(w, v) > 0$.

V prvním případě je kapacita hrany (v, w) v síti *S'* rovna číslu $c(v, w) - t(v, w)$ a ve druhém případě číslu $t(w, v)$.

3. [*Vytvoření sítě S''.*]
Jestliže v síti *S''* neexistuje orientovaná cesta ze zdroje do spotřebiče, výpočet končí a *t* je maximální tok. V opačném případě vytvoříme síť *S''* tak, že ze sítě *S'* vynecháme hrany, které neleží na nejkratší orientované cestě ze zdroje do spotřebiče v síti *S''*. (Délka orientované cesty je rovna počtu hran, které na ní leží.) Hrany, které je třeba vynechat z *S'*, určíme takto:

- pro každý vrchol *v* určíme $\rho(z,v)$, délku nejkratší orientované cesty ze zdroje do *v*,
- pro každý vrchol *v* určíme $\sigma(v,s)$, délku nejkratší orientované cesty z vrcholu *v* do spotřebiče,
- hranu (v, w) vynecháme z *S'* právě tehdy, jestliže je

$$\rho(z,v)+1+\sigma(v,s) > \rho(z,s) .$$

4. [*Nalezení nasyceného toku v síti S''.*]
Pomocí *Algoritmu tří Indů* (popsán výše) nalezneme nasycený tok *q* v síti *S''*.

5. [*Zvětšení toku t.*]
Pro každou hranu (v, w) sítě *S''* položíme:
je-li $(v,w) \in H(G)$, pak $t(v, w) = t(v, w) + q(v, w)$,
je-li $(w,v) \in H(G)$, pak $t(w, v) = t(w, v) - q(w, v)$.

6. [*Skok na bod 2.*]

Algoritmus byl prakticky realizován a testován v programu *Toky v sítích*.

U algoritmu tří Indů již není tak zřejmé, že v okamžiku zastavení algoritmu je *t* nasycený tok. Lze to ale jednoduše dokázat.

Není těžké ověřit, že je algoritmus navržen tak, aby se nikdy neporušilo omezení toku kapacitami. Kromě toho platí, že položíme-li $f^+(v) = 0$, pokud $v \in M^+$, a $f(v) = 0$, pokud $v \in M^-$, pak součet toku hranami vstupujícími do vrcholu *v* a čísla $f^+(v)$ je roven součtu toku hranami vystupujícími z vrcholu *v* a čísla $f(v)$ pro kterýkoliv vrchol a kterýkoliv okamžik výpočtu. Jelikož na konci výpočtu jsou funkce f^+ a f nulové, je *t* tok.

Je ještě nutné dokázat, že v okamžiku, kdy začínáme provádět bod 4a, existuje otevřená hrana vycházející z *v*. To plyne z toho, že celkové zvýšení toku hranami vycházejícími z *v* nepřekročí r_0, zatímco rezerva pro toto zvýšení je $r_{out}(v)$ a platí $r_0 \le r(v) \le r_{out}(v)$. Obdobnou úvahu lze provést i pro bod 4c.

Nyní dokážeme, že pro každou orientovanou cestu $z = v_0, \ldots, v_k = s$ platí v každém okamžiku výpočtu následující tvrzení: obsahuje-li cesta uzavřenou hranu, pak obsahuje také nasycenou hranu.

Tvrzení zřejmě platí na začátku výpočtu. Předpokládejme nyní, že platí až do jistého okamžiku τ, ale pak se jeho platnost poruší. Jelikož nasycené hrany zůstávají nasycenými, je τ jistě okamžik, kdy se uzavře některá hrana (v_{i-1}, v_i). Kdyby k uzavření došlo v bodě 4, pak víme, že hrana (v_{i-1}, v_i) je nasycená, takže by tvrzení platilo dále. Kdyby k uzavření došlo v bodě 5, pak již předtím musely být uzavřeny buď všechny hrany vstupující do v_{i-1}, nebo všechny hrany vystupující z v_i, takže již před τ existovala na zkoumané cestě uzavřená hrana, a tedy také nasycená hrana. Došli jsme tedy ke sporu s předpokladem, že platnost našeho tvrzení se během výpočtu může porušit.

Jelikož na konci výpočtu jsou uzavřeny všechny hrany vycházející ze spotřebiče, musí být na každé cestě ze spotřebiče do zdroje alespoň jedna nasycená hrana, a tedy tok t je nasycený.

Časová náročnost algoritmu

Algoritmus tří Indů zpracuje síť o n vrcholech v čase $O(n^2)$ a s pomocí algoritmu na vytvoření vrstvené sítě umožňuje nalézt maximální tok v obecné síti v čase $O(n^3)$.

Nyní provedeme důkaz, že časová složitost Algoritmu tří Indů pro vrstvenou síť je $O(n^2)$. Jelikož se v každé iteraci bodů 2 až 6 uzavře referenční vrchol v_0 a uzavřené vrcholy se již neotvírají, provede se těchto iterací nejvýše n. Je-li m počet hran sítě, pak k provedení bodu 1 stačí čas $O(m)$, k jednomu provedení bodu 2 čas $O(n)$, bodu 3 čas $O(n)$ a bodu 6 konstantní čas, takže celkový čas spotřebovaný body 1 až 3, 6 je $O(m) + n[O(n) + O(n) + O(1)] = O(n^2)$. V bodě 5 se každá hrana a každý vrchol zpracovává nejvýše jednou za celý výpočet a věnuje se jim vždy konstantní množství času, takže bod 5 celkově požaduje čas $O(m) = O(n^2)$. Čas spotřebovaný v bodě 4 „připíšeme na účet“ jednotlivých hran a vrcholů.

Dojde-li v bodě 4a nebo 4c k nasycení, a tedy také k uzavření hrany, pak čas spotřebovaný tímto provedením zmíněné partie programu se připíše uvedené hraně. Každé hraně se připisuje nejvýše jednou za celý výpočet. Pokud nedošlo k uzavření hrany, došlo k vynechání vrcholu v z fronty M^+ nebo M^-. Čas potom připíšeme vrcholu v. Každému vrcholu je v každé z iterací bodů 2 až 6 připisováno nejvýše jednou za bod 4a a jednou za bod 4c.

Jelikož jedno provedení bodu 4a nebo 4c trvá konstantní čas, byl hranám připsán čas $O(m)$ a vrcholům $O(n^2)$.

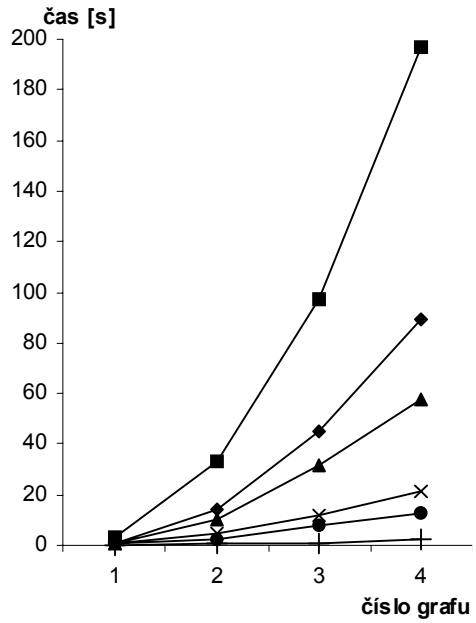
6 Porovnání algoritmů hledajících maximální tok v síti

Protože časová náročnost daného algoritmu je příliš ovlivněna schopností programátora při implementaci daného algoritmu, operačním systémem, strukturou programu ve vyšším programovacím jazyku, dobře či méně dobře optimalizovanými kompilátory atd. , a proto použijeme pro porovnání efektivnosti časové složitosti jednotlivých algoritmů.

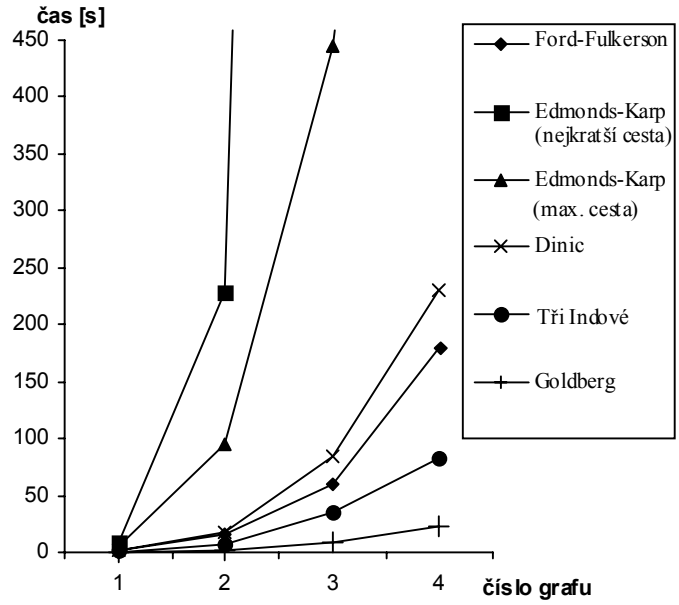
Číslo algoritmu	Datum vzniku	Autor algoritmu	Časová složitost
1	1957	Ford-Fulkerson	$qO(m), q \in (0,\infty)$
2	1972	Edmonds a Karp (nejkratší cesta)	$O(nm^2)$
3	1973	Edmonds a Karp (cesta s max. kapacitou)	$O(n^2 m \ln x)$
4	1972	Dinic (rezervní síť)	$O(n^2 m)$
5	1978	Malhotra, Pramodh Kumar, Maheshwari	$O(n^3)$
6	1988	Goldberg	$O(mn \log n^2/m)$

Tabulka 6.1: Porovnání časové složitosti jednotlivých algoritmů (*n*-vrcholy, *m*-hrany).

Jednotlivé algoritmy byly naprogramovány v jazyce Delphi a testovány na různých topologiích sítí. Z porovnání teoretických předpokladů (tabulka 6.1) a praktických výsledků (obr.6.1 a obr.6.2) je zřejmé, že algoritmy dosahují předpokládané rychlosti výpočtů. Nejlepších výsledků dosáhl samozřejmě algoritmus Goldbergův.



Obr.6.1: Řídká síť



Obr.6.2: Hustá síť

7 Závěr

Účelem článku bylo seznámit čtenáře se základy problematiky hledání maximálního toku v síti. Byly popsány základní problémy řešené v praxi, které si vynutily vznik této teorie a základní algoritmy řešící danou problematiku. Podrobně zpracován byl Algoritmus tří Indů, který patří i dnes spolu s Goldbergovým algoritmem k nejlepším při hledání maximálního toku v hustých sítích. K praktickému řešení konkrétní sítě nebo k demonstraci jednotlivých algoritmů lze s úspěchem použít program *Toky v sitich* .

8 Literatura

BEASLEY, J. E.: *OR-Library*. http://mscmga.ms.ic.ac.uk/jeb/orlib/netflowinfo.html, London, 1996.

DINITZ, Y., GARG, N. & GOEMANS, M.: On the Single – Source Unsplittable Flow Problem. In *Combinatorica*, Vol. 19, No. 1, 1999, pp. 1-25.

GOEMANS, M.: *Networks Flow*. Massachusetts Institute of Technology, Laboratory for Computer Science, Lecture Notes, 1994.

GOLDBERG, A. V. And Tarjan, R. E. A New Approach to the Maximum-Flow Problem. In *Journal of the Association for Computing Machinery*. Vol. 35, No. 4, 1988, pp. 921-940.

JANEČEK, J. *Distribuované systémy*. Praha: ČVUT, 1996.

KUČERA, L. *Kombinatorické algoritmy*. Praha: SNTL, 1983.

PALUBJÁK, P. *Toky v sítích.* [Diplomová práce.] Brno: VUT FSI Brno. Ústav automatizace a informatiky, 2000.

PLESNÍK, J. *Grafové algoritmy.* Bratislava: Alfa, 1983.

SKOROBOHATYJ, G. *ZIB Homepage*. ftp://ftp.zib.de/pub/Packages/mathprog/maxflow, 1999.