Toky v sieťach

Ročníkový projekt – report
Adam Struharňanský, študent
doc. RNDr. Robert Lukoťka, PhD., školiteľ

Úvod

Problém maximálneho toku v sieti po prvý raz vyslovil T. E. Harris už v 1954 ako zjednodušený model železničných tratí. Už o dva roky objavili L. R. Ford a D. R. Fulkerson metódu, pomocou ktorej bol tento problém riešiteľný. Prvotné dôvody z reálneho sveta na riešenie tohto problému boli hlavne zo strategických vojenských dôvodov, kde bolo potrebné nájsť, ako najlacnejšie znefunkčniť nepriateľovu prepravnú sústavu, respektíve vybudovať svoju čo najodolnejšie [1], ale neskôr prišli dôvody aj z civilného sveta, ako napríklad rozvrhovanie posádok v letectve, alebo rozlíšenie obrázku na popredie a pozadie, a ďalšie.
Desaťročia boli nachádzané na tento problém nové riešenia pomocou rôznych metód, pričom sa neustále zefektívňovali použité prostriedky. Dnes je jedným z najlepších riešenie J. B. Orlina spojené s riešením V. Kinga, S. Raoa, a R. Tarjana, kde časová zložitosť takéhoto algoritmu je O(|V|*|E|) [2].

Definícia problému

Nech graf G = (V,E) s dvoma špeciálnymi vrcholmi a,z, kde a je zdroj a z je výtok, pričom tento graf má ohodnotené orientované hrany, kde hodnotou hrany označujeme jej kapacitu, a teda koľko môže cez túto pretiecť. Tok je funkcia, ktorá každej hrane priradí kladné číslo, pričom toto nesmie byť väčšie ako jej hodnota, a musí platiť, že súčet tokov na hranách vedúcich do vrchola je rovný súčtu vrcholov vedúcich z vrchola, čo platí pre každý vrchol až na zdroj a výtok. Hodnota toku je je množstvo prechádzajúce zo zdroja do výtoku.
Problém maximálneho toku rieši nájdenie takého toku s najväčšou hodnotou.

Moja úloha

Porovnal som viaceré algoritmy a prístupy, ako vyriešiť daný problém. Naprogramoval som vlastný kód na základe Ford-Fulkersenovej metódy a Edmond-Karpovej implementácie, našiel som implementácie jednoduchého Dinicovho algoritmu a Malhotra-Kumar-Maheshwariho algoritmu pod vhodnými licenciami, naprogramoval som kód na prevod na riešenie pomocou lineárneho programovania (LP) s použitím voľne dostupného solvera, a interface-u naprogramovaným mojím školiteľom, a nakoniec som použil OR-tools – open source softvér pre optimalizáciu nielen problému maximálneho toku. Všetky kódy boli napísané v jazyku C/C++, až na program na volanie OR-tools, kvôli oveľa jednoduchšej inštalácií a implementácií, pričom však tento softvér je napísaný tiež v C/C++ (prípadne CC), a iba zanedbateľný čas je použitý na réžiu.
Tiež som si vytváral vlastné grafy na porovnávanie. Pri mojej implementácií prevodu na LP som však potreboval zabezpečiť, aby bol daný graf acyklický, a preto všetky vytvorené grafy sú acyklické. Tiež som nevytváral násobné hrany. Tieto, v závislosti od implementácie grafu, vieme

celkom ľahko odstrániť (spojenie do jednej), pričom ich existencia by tiež vedela komplikovať niektoré implementácie.
Ako porovnávaciu metódu som bol použitý reálny čas behu funkcie na výpočet maximálneho toku, pričom do tohto nebol zahrnutý čas, ktorý bol potrebný na vytvorenie daného grafu v tej ktorej implementácií (predpokladám, že tento by bol aj tak veľmi podobný, keďže je to pri všetkých iba takmer to isté vloženie vždy tých istých vrcholov a hrán). Toto porovnávanie som použil aj preto, pretože je zložité porovnávať takto nesúvisiace algoritmy nejak inak, keďže napríklad o samotnej implementácií OR-tools, alebo tohto konkrétneho LP solvera viem iba veľmi málo.
Vopred som predpokladal, že algoritmy ktoré majú vopred zadané časové zložitosti, budú zoradené podľa týchto, môj prepis na LP bude najpomalší, keďže pri mojej implementácií môže trvať exponenciálne dlhý čas, a OR-tools bude najrýchlejší.

Implementácia Ford-Fulkersonovho metódy, pomocou Edmond-Karpovej implementácie

Táto metóda vznikla ako prvá na riešenie daného problému. Rieši ho pomocou metódy zväčšujúcich/zlepšujúcich sa ciest. Edmond-Karpová implementácia je plne definovaná Ford-Fulkersonová metóda. Myšlienka zlepšujúcich sa ciest je nasledovná: ak existuje nejaká cesta zo zdroja do ústia, touto pošleme nejaký tok. Poslanie toku znamená vhodne upraviť graf – každá hrana cez ktorú sme poslali tok, bude zmenšená o daný tok, a pre každú hranu (a,b) sa vytvorí hrana (b,a) (alebo ak už existovala tak sa zväčší o hodnotu toku) s hodnotou toku. Tento postup opakujeme, kým existujú nenulové cesty zo zdroja k ústiu. Rozdiely pri implementácií tejto metódy sú hlavne pri nachádzaní nejakej cesty zo zdroja do ústia. Vyhľadávanie do šírky (BFS), používa práve Edmond-Karpova implementácia, ktorá je použitá aj v tomto riešení. Časová zložitosť tejto implementácie je $O(|V|*|E|^2)$. [3]

Dinicov algoritmus a Malhotra-Kumar-Maheshwariho algoritmus

Tieto algoritmy používajú tiež zlepšujúce sa cesty, sú si všetky teda podobné, ale od Edmond-Karpovej implementácie prinášajú nové koncepty, a sú schopné mať lepšie časové zložitosti, Dinicov algoritmus $O(|V|^2*|E|)$[4], Malhotra-Kumar-Maheshwariho algoritmus $O(|V|^3)$[5].

Riešenie ako lineárny program

Pri tomto riešení sú vytvorené obmedzenia vyplývajúce priamo z definície problému. V každom vrchole okrem zdroja a ústia vytvoríme obmedzenie aby rozdiel vchádzajúceho a vychádzajúceho toku bude nulový, aby každá hrana mala kapacitu najviac svojej hodnoty, a maximalizácia vstupných hodnôt do ústia/ výstupných zo zdroja.
Žiaľ, toto som si pri implementovaní môjho riešenia neuvedomil, a moje obmedzenia boli, že pre každú cestu zo zdroja do ústia musí platiť, že tok cez ňu bude nezáporný, a pre všetky cesty ktoré obsahujú nejakú hranu musí platiť, že súčet ich tokov je menší ako hodnota tejto hrany. Počet možných ciest ktoré vedú zo zdroja do ústia však môže byť až exponenciálny.

OR-tools

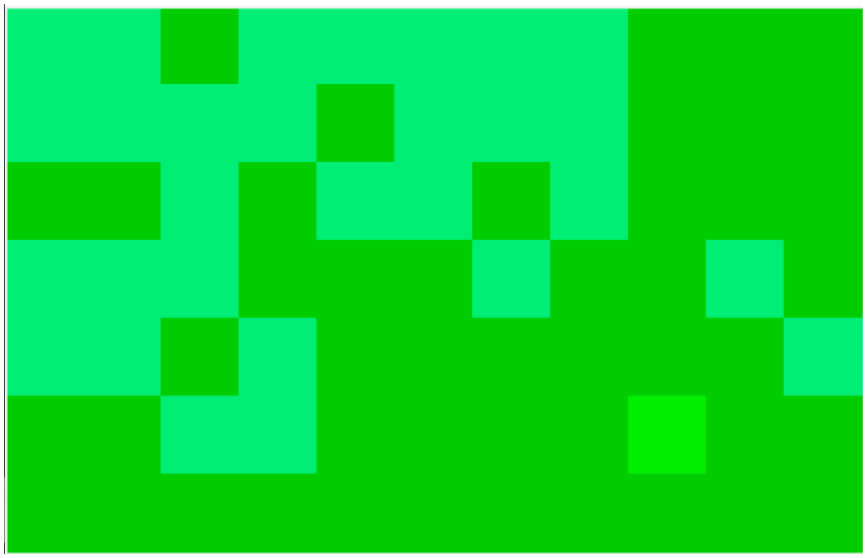
Open source softvér na optimalizáciu, vytvorený na riešenie nielen tokov v sieťach, vyhral viacero zlatých medaily v súťaži optimalizačných softvérov. Neviem presne aké optimalizačné techniky v ňom boli použité, ani ako presne sa dostal k výsledku. Bol braný ako čierna skrinka, do ktorej po vložení grafu dostaneme riešenie.

Výsledky

V prílohe 1 sú aj číselné výsledky testov, v tomto texte sú iba heatmapy výsledkov týchto testov. Nižšie je legenda, a potom pre každý test je obrázok, ktorý v ktorom stĺpce označujú testy s rovnakými počtami vrcholov, zľava doprava idú od 50 vrátane až po 100 vrátane, skoky sú o 5. Riadky označujú grafy, ktoré majú (takmer, presné opísanie pri funkcii tvorenia grafu) všetky vrcholy s rovnakým počtom hrán, pričom hrana vrcholu je tu braná aj ktorá doň vstupuje, aj ktorá z neho vystupuje, zhora nadol od 3 vrátane po 9 vrátane. V každom teste označenom jedným štvorčekom bol daný algoritmus spustený na 50 rôznych grafoch.

Obr. 1: Legenda

Farby na Obrázku 1 označujú jednotlivé časy behu programu, zľava doprava sú to intervaly v mikrosekundách <0,10000), <10000,30000), <30000,60000), <60000,100000), <100000,300000), <300000,600000), <600000,1000000), <1000000,3000000), <3000000,6000000), <6000000,10000000), <10000000,30000000), <30000000,60000000), a dlhšie ako 600000000.



Obr. 2: OR-tools

Obr. 3: Dinicov algoritmus

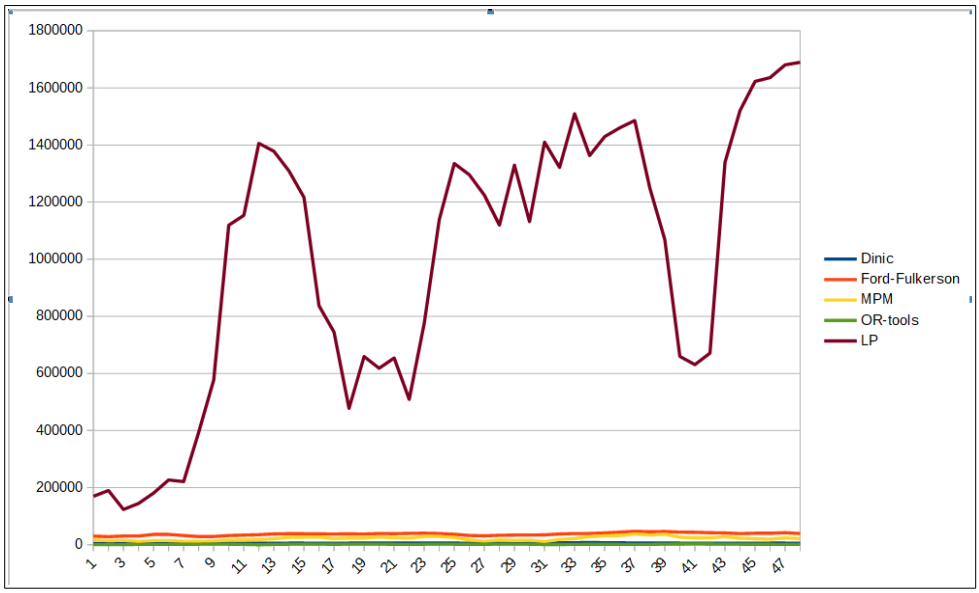
Obr. 4: Malhotra-Kumar-Maheshwariho algoritmus

Obr. 5: LP

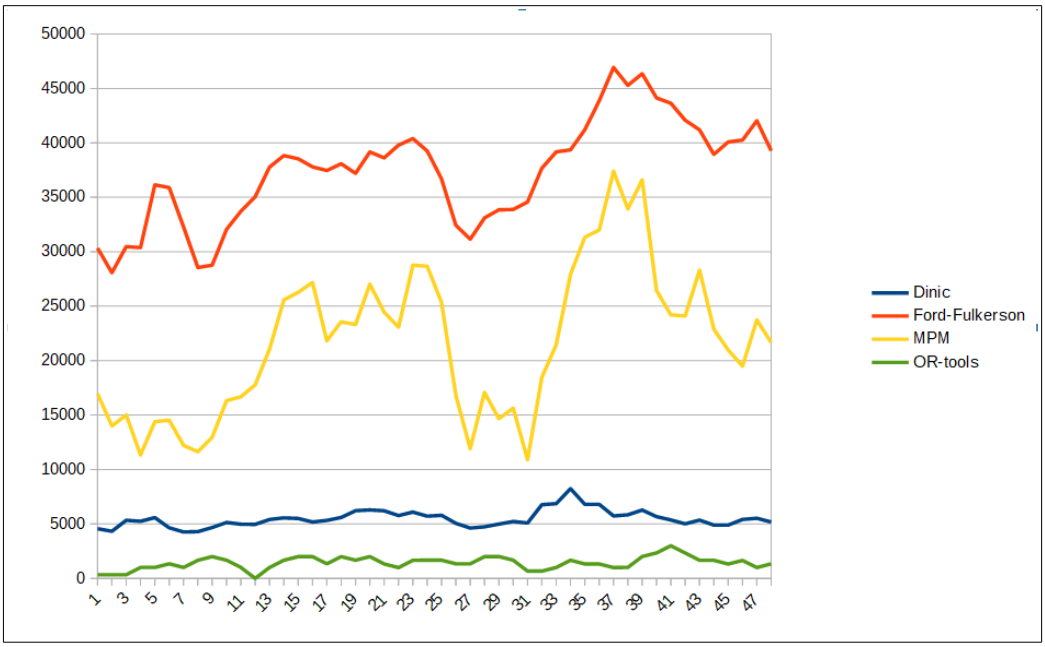
Obr. 6: Ford-Fulkerson, Edmond-Karp implementácia

Pri testoch na LP pri mojej implementácií bol tento algoritmus tak neefektívny, že som bol nútený ho pre nedostatok prostriedkov vypnúť po prejdený na grafoch s vrcholmi s iba 3 hranami. Zároveň však treba poznamenať, že pri tomto bolo do celkového času zarátané aj nájdenie všetkých ciest. Tiež treba poukázať na to, že OR-tools boli spúšťané z iného jazyka, a je možnosť, že pri zisťovaní času v Pythone mohol byť tento zaokrúhlený na väčšie jednotky ako v C++. Potom pri krátkych časoch behu tohto programu nedokázal zistiť skutočný beh programu, pretože tento bol zaokrúhlený na 0. (Je možné že v Pythone bol čas počítaní na milisekundy, v C++ na mikrosekundy).

Ďalej boli urobené testy pre grafy, kde v každom grafe mal každý vrchol menej ako 5 hrán (súčet vstupných aj výstupných), a mal 1,5*|V| hrán. Pri tomto boli urobené testy pre grafy s počtom vrcholov od 50 vrátane po 100 vrátane. Vo výsledkoch boli drobné anomálie, preto som sa rozhodol vytvoriť nové hodnoty pre každý algoritmus, kde i-ta hodnota je priemer účinnosti tohto algoritmu na grafoch o veľkostiach 49 + i, 50 + i, 51 + i. Anomálie vznikli nielen kvôli konštrukcii grafov, ale aj z dôvodu zaokrúhlenia niektorých časov behov na 0.



Obr. 7: Porovnanie všetkých algoritmov na podmnožine grafov



Obr. 8: Porovnanie algoritmov bez LP na podmnožine grafov

Zhrnutie

Vo všetkých porovnaniach sme dospeli k záveru, že tieto implementácie algoritmov môžeme zoradiť od najlepšieho takto: OR-tools, Dinicov, Malhotra-Kumar-Maheshwariho, Ford-Fulkersonov, a moja implementácia pomocou LP.

Môžme si všimnúť, že Malhotra-Kumar-Maheshwariho algoritmus bol horší ako Dinicov, aj keď má lepšiu časovú zložitosť, čo nesedí s našimi očakávaniami. Je to spôsobené tým, že v tejto implementácií je zrejme spomalený veľkou konštantou. Ak si pozrieme jeho výsledky, tak presne ako vyplýva zo zložitosti, pri pridávaní hrán ale rovnakom počte vrcholov je stále rovnako rýchly. Ak by sme testovali na hustejších grafoch, bol by pravdepodobne rýchlejší.

Ďalej rýchlosť OR-tools nemusí byť skutočne takáto nízka, môže to byť tiež spôsobené zlým počítaním času v inom programovacom jazyku.
Nakoniec si môžeme všimnúť, že moja implementácia na LP je neporovnateľne horšia, ako hociktorý iný skúmaný algoritmus.

Referencie

[1] Schrijver, A. (2002). "On the history of the transportation and maximum flow problems", str 441-442
[2] Orlin, James B. (2013). "Max flows in O(nm) time, or better", str 1
[3] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Cliffors (2001). "Introduction to algorithms", str 714-730
[4] Dinitz, Yefim (1970). "Algorithm for solution of a problem of maximum flow in a network with power estimation"
[5] Malhotra, V.M.; Kumar, M. Pramodh; Maheshwari, S.N. (1978). "An O($|V|^3$) algortithm for finding maximum flows in networks"