ViDA: Vizualizácia distribuovaných algoritmov

Michal Anderle[1]* Ján Hozza[1]†

Školiteľ: Jakub Kováč[2]‡

[1] Katedra informatiky, FMFI UK, Mlynská Dolina 842 48 Bratislava
[2] Katedra informatiky, FMFI UK, Mlynská Dolina 842 48 Bratislava

Abstrakt: Koník je úžasný

Kľúčové slová: vizualizácia, distribuované algoritmy

1 Úvod

V zimnom semestri sme absolvovali predmet Úvod do distribuovaných algritmov a veľmi nás zaujala prednášaná téma. Napadlo nás, že táto téma, aj keď je zložitá, by sa dala vysvetliť bežnému človeku, ktorý má o ňu záujem bez toho, aby si musel študovať zložito písané knihy. Tak vznikol náš ročníkový projekt.

Ako cieľ sme si stanovili nielen zvizualizovanie niektorých dobre známych algoritmov, ale taktiež vytvorenie istej interaktivity s používatľom, ktorá by mu pomohla prehĺbiť a lepšie si zapamätať novonadobudnuté znalosti. Myslíme si totiž, že v tejto oblasti existuje mnoho pekných úvah a trikov, ktoré je dobré si osvojiť. A formou interaktívnej vizualizácie, by sme chceli pomôcť vo výučbe a spoznávaní týchto algoritmov, ale takisto umožniť užívateľovi, aby si sám vyskúšal naprogramovať niektoré z algoritmov. Toto by malo preklenúť medzeru medzi porozumením a schopnosťou aplikácie.

Medzi algoritmy, ktoré sme zatiaľ zvizualizovali patrí distribuované prehľadávanie do šírky a taktiež voľba šéfa na úplnom grafe. Zamerali sme sa však na to, aby užívateľ bol schopný sám vytvoriť algoritmus, ktorý mu naša aplikácia zvizualizuje.

Zvyšok článku je organizovaný nasledovne: v sekcii 2 sa zameriame na samotnú implementáciu, ako funguje a prečo sme zvolili zrovna túto možnosť. V sekcii 3 popíšeme konkrétne algoritmy, ktoré vizualizujeme a v sekcii 4 spomenieme naše plány do budúcnosti.

a

*zaba@ksp.sk
†janoh@ksp.sk
‡kuko@ksp.sk

2 Vybraté distribuované algoritmy

V našom ponímaní sú distribuované algoritmy výpočtový model, pracujúci nasledovne. Máme niekoľko počítačov, z ktorých sú niektoré dvojice spojené obojsmernou komunikačnou linkou. Inak povedané, počítače tvoria neorientovaný graf.

Do každého z týchto počítačov sa nahrá ten istý program a všetky sa naraz spustia. Programy si môžu interne čokoľvek počítať a zároveň dokážu poslať správu po ľubovoľnej linke. O správach na linkách vieme len to, že v konečnom čase dorazia na druhý koniec a keď jedne počítač pošle po jednej linke viac správ, tak dorazia v tom poradí, v akom boli poslané. O poradí doručenia správ na rôznych liniek nevieme nič a každá správa sa môže doručovať ľubovoľne dlho. Nemáme teda žiadne garancie, či správa dorazí do 10 minút a kľudne sa môže stať, že niektoré správy prejdú tísícky hrán, zatiaľ čo iné len jednu.

Úlohou distribuovaných algoritmov je následne riešiť rôzne problémy, napríklad ako medzi sebou zvoliť jedného šéfa, poslať nejakú správu ostatným počítačom (nie len susedom) alebo prehľadať graf.

Snaha je používať také algoritmy, ktoré dokopy pošlú čo najmenší počet správ.

Momentálne sú v aplikácii naimplementované vizualizácie troch algoritmov.

2.1 Voľba šéfa na úplnom grafe s počtom správ $On\log n$

Ako názov naznačuje, v tejto sekcii máme n počítačov a všetky dvojice počítačov sú spojené linkou. Navyše, má každý počítač svoj jednoznačný identifikátor (ľubovoľné celé číslo, nazvime ho *ID*), bez identifikátorov by sa totiž problém nedal riešiť, kvôli symetrii.

Počítače na začiatku poznajú len svoje *ID* a vidia $n-1$ portov očíslovaných číslami 1 až $n-1$ v ľubovoľnom poradí. Programy sú spustené v rovnakom čase a môžu začať posielať správy. Chceme nájsť algoritmus, ktorý pre ľubovoľné identifikátory, časovanie správ a poradie portov skončí tak, že práve

jeden počítač bude vedieť o sebe, že je šéf a ostatné budú vedieť, že nie sú šefovia.

Dá sa dokázať, že každý takýto algoritmus musí poslať aspoň $O(n \log n)$ správ a my si povieme o jednom z takých, ktoré to dokážu.

Algoritmus funguje tak, že na to, aby sa stal počítač šéfom, musí mať $n-1$ `vazalov` a sám nesmie byť nikomu `vazalom`. Každý počítač je `vazalom` najviac jedného počítača. Z tohto vyplýva, že šéf bude najviac jeden. Navyše si označme $level = poetvazalov$.

Na začiatku sa chce každý počítač stať šéfom a teda snaží sa získať nejakých `vazalov`. To robí tak, že pošle nejakému susedovi správu `"Mám level x, id y a chcem ťa zajať."`

Ak sused už nie je `vazalom` niekoho iného, lexikograficky porovná svoju dvojicu $(level, id)$ s dvojicou so správy. Ak je jeho dvojica väčšia, správu ignoruje a teda pôvodný počítač nikdy `nezajme` všetkých, čím zahynie jeho šanca sa šéfom. Inak mu odpovie `"Som tvojim vazalom"`. Pôvodný počítač si zvýši level a snaží sa zajať niekoho ďalšieho. Takto pokračuje, až kým sa nestane, že mu niekto neodpovie, alebo sa nestane šéfom.

Druhá možnosť je, že sused už je vazalom nejakého počítača, vtedy požiada o pomoc svojho panovníka. Podľa *levelu* a *id* panovníka sa rovnakým spôsobom rozhodne, či sa správa ignoruje, alebo sa sused stane vazalom vyzívateľa a panovník sa stane porazeným (nebude nikoho vazalom, kým sa ho niekto priamo nepokúsi zajať, ale už nebude zajímať ďalších vazalov).

Ak ste to náhodou z tohoto popisu nepochopili, tak je to dôvod na to, aby ste si vyskúšali našu čarovnú aplikáciu.

3 blabla

Pri vývoji aplikácie sme sa zamerali na to, aby sa dali jednoduchým spôsobom pridávať nové algoritmy. Dokonca pridávať nové algoritmy si môže aj užívateľ bez toho, aby musel upravovať zdrojový kód aplikácie.

Pre pridanie nového vlastného algoritmu stačí napísať program, ktorý následne bude príjmat

Momentálne sú podporované programy v programovacom jazyku C++, pričom programátor

Poďakovanie

Neďakujeme FMFI UK za podporu.