VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
INFORMATIKOS KATEDRA

Baigiamasis bakalauro darbas

Rikiavimo tobulinimas genetiniais algoritmais

(Improving Sorting with Genetic Algorithms)

Atliko: 4 kurso 2 grupės studentas
Deividas Zaleskis (parašas)
Darbo vadovas:
Irmantas Radavičius (parašas)
Recenzentas:
doc. dr. Vardauskas Pavardauskas (parašas)

Vilnius
2022

Turinys

Sąvokų apibrėžimai 2
Įvadas 3
1. Pagrindinė tiriamoji dalis 5
Išvados 6
Conclusions 7
Literatūra 8

Sąvokų apibrėžimai

Sutartinių ženklų, simbolių, vienetų ir terminų sutrumpinimų sąrašas (jeigu ženklų, simbolių, vienetų ir terminų bendras skaičius didesnis nei 10 ir kiekvienas iš jų tekste kartojasi daugiau nei 3 kartus).

Įvadas

Duomenų rikiavimas yra vienas aktyviausiai tiriamų uždavinių informatikos moksle. Iš dalies tai lemia rikiavimo uždavinio prieinamumas ir analizės paprastumas. Formaliai rikiavimo uždavinys formuluojamas taip: duotai baigtinei palyginamų elementų sekai $S = (s_1, s_2, ..., s_n)$ pateikti tokį kėlinį, kad duotosios sekos elementai būtų išdėstyti monotonine (didėjančia arba mažėjančia) tvarka. Kadangi rikiavimo uždavinio sąlyga yra gana paprasta, tai suteikia didelę galimų implementacijų įvairovę. Todėl nauji rikiavimo algoritmai ir įvairūs patobulinimai egzistuojantiems algoritmams yra kuriami ir dabar.

Rikiavimo uždavinys yra fundamentalus, kadangi rikiavimas padeda pagrindą efektyviam kitų uždavinių sprendimui. Kaip to pavyzdį galima pateikti dvejetainės paieškos algoritmą, kurio prielaida, jog duomenys yra išrikiuoti, leidžia sumažinti paieškos laiko sudėtingumą iki $O(log\, n)$. Rikiavimas taip pat svarbus duomenų normalizavimui bei pateikimui žmonėms lengvai suprantamu formatu. Kadangi duomenų rikiavimas yra fundamentalus uždavinys, net ir nežymūs patobulinimai žvelgiant bendrai gali atnešti didelę naudą.

Rikiavimo uždaviniui spręsti egzistuoja labai įvairių algoritmų. Plačiausiai žinomi yra klasikiniai rikiavimo algoritmai: rikiavimas sąlaja (angl. merge sort), rikiavimas įterpimu (angl. insertion sort) ir greitojo rikiavimo algoritmas (angl. quicksort). Tiesa, šie algoritmai turi įvairių trukūmų: rikiavimas sąlaja dažnai veikia lėčiau nei nestabilūs algoritmai ir naudoja $O(n)$ papildomos atminties, rikiavimas įterpimu yra efektyvus tik kai rikiuojamų duomenų dydis yra mažas, o greitojo rikiavimo algoritmas blogiausiu atveju turi $O(n^2)$ laiko sudėtingumą. Todėl šiuo metu praktikoje plačiausiai naudojami hibridiniai rikiavimo algoritmai, kurie apjungia kelis klasikinius algoritmus į vieną panaudodami jų geriausias savybes. Pavyzdžiui, C++ programavimo kalbos standartinėje bibliotekoje naudojamas introspektyvaus rikiavimo (angl. introsort) algoritmas įprastai naudoja greitojo rikiavimo algoritmą, pasiekus tam tikrą rekursijos gylį yra naudojamas rikiavimas krūva (angl. heapsort) siekiant išvengti $O(n^2)$ laiko sudėtingumo, o kai rikiuojamų duomenų dydis yra pakankamai mažas, pasitelkiamas rikiavimas įterpimu, kadangi su mažais duomenų dydžiais jis yra efektyvesnis. Apibendrinant, pasitelkiant įvairius rikiavimo algoritmus ir jų unikalias savybes yra įmanoma rikiavimo uždavinį spręsti efektyviau.

Vienas iš teorine prasme įdomiausių klasikinių algoritmų yra Šelo rikiavimo algoritmas. Šelo algoritmą galima laikyti rikiavimo įterpimu optimizacija, kadangi atliekant rikiavimą yra lyginami ne tik gretimi elementai, kas leidžia kai kuriuos elementus perkelti į galutinę poziciją atliekant mažiau operacijų. Pagrindinė algoritmo idėja - išskaidyti rikiuojamą seką S į posekius $S_1, S_2, ..., S_n$, kur $S_i = (s_i, s_{i+h}, s_{i+2h}, ...)$ ir atskirai išrikiuoti kiekvieną posekį S_i. Įprastai tarpų rinkinys, kuriuo remiantis formuojami rikiuojami posekiai, vadinamas tarpų seka. Šelo algoritmo efektyvumas įprastai priklauso nuo pasirinktos tarpų sekos, todėl bendra teorinė šio algoritmo analizė yra labai sudėtinga. Taip pat reikia pastebėti, jog praktikoje efektyviausi yra eksperimentiškai gauti Šelo algoritmo variantai [Ciu01; Tok92].

Literatūroje galima rasti darbų [RBH+02; SY99], kuriuose genetiniai algoritmai yra taikomi

naujų Šelo algoritmo tarpų sekų radimui. Tiesa, šių darbų praktinio panaudojimo potencialas yra gana ribotas. Simpson-Yachavaram darbe daugiausia dėmesio skiriama atliekamiems palyginimams, teigiama, jog gautos tarpų sekos atlieka mažiausiai palyginimo operacijų, tačiau jos yra lyginamos tik su Sedgewick ir Incerpi-Sedgewick tarpų sekomis. Tačiau šiuo metu yra laikoma, jog vidutiniškai atliekamų palyginimų atžvilgiu optimaliausia yra Ciura [Ciu01] tarpų seka. Kursinio darbo metu buvo atliktas Ciura ir Simpson-Yachavaram tarpų sekų tarpusavio palyginimas vidutiniškai atliekamų priskyrimų atžvilgiu, gauti rezultatai tam neprieštaravo. Roos et al. darbe daugiauia dėmesio skiriama veikimo laikui, teigiama, jog gauta tarpų seka veikia greičiau nei kitos tirtos tarpų sekos, tačiau pateikiamuose rezultatuose ji lyginama tik su Simpson-Yachavaram tarpų seka, tiriami tik keli duomenų dydžiai. Kursinio darbo metu buvo atliktas platesnio masto Roos et al. ir kitų tarpų sekų tarpusavio palyginimas vidutinio veikimo laiko atžvilgiu, pagal gautus rezultatus Roos et al. sekos veikimo laikas buvo mažiausias, tačiau ji taip pat atliko daugiausia palyginimo ir priskyrimo operacijų.

Keliama tokia **hipotezė**:

Pasitelkiant genetinius algoritmus įmanoma sukonstruoti efektyvius Šelo algoritmo variantus, kurių vidutinis veikimo laikas būtų mažesnis nei šiuo metu žinomų variantų.

Siekiant patikrinti iškeltą hipotezę, reikia atlikti šiuos **uždavinius**:

- Išanalizuoti Šelo algoritmą ir jo variantus remiantis literatūra ir eksperimentiškai gautais duomenimis;
- Nustatyti kriterijus Šelo algoritmo variantų efektyvumui įvertinti;
- Realizuoti genetinį algoritmą Šelo algoritmo variantų generavimui;
- Pasitelkiant realizuotą genetinį algoritmą sugeneruoti Šelo algoritmo variantus;
- Eksperimentiškai palyginti sugeneruotų ir pateiktų literatūroje Šelo algoritmo variantų efektyvumą.

Šiame darbe atlikta:

- kol kas nieko

Darbas remiasi tokiomis prielaidomis:

- Atliekant Šelo algoritmo variantų generavimą ir tarpusavio palyginimą rikiuojamų duomenų palyginimo ir priskyrimo sudėtingumas laiko atžvilgiu yra $O(1)$;
- Atliekant Šelo algoritmo variantų tarpusavio palyginimą yra pakankama aprėpti geriausiai žinomus variantus (visų literatūroje pateiktų variantų tarpusavio palyginimas reikalautų atskiro tyrimo);
- Šelo algoritmo variantų tarpusavio palyginimui pasirinkti duomenų rinkiniai pakankamai tiksliai atspindi dažniausiai praktikoje sutinkamus duomenų rinkinius.

1. Pagrindinė tiriamoji dalis

Pagrindinėje tiriamojoje dalyje aptariama ir pagrindžiama tyrimo metodika; pagal atitinkamas darbo dalis, nuosekliai, panaudojant lyginamosios analizės, klasifikacijos, sisteminimo metodus bei apibendrinimus, dėstoma sukaupta ir išanalizuota medžiaga.

Išvados

Išvadose ir pasiūlymuose, nekartojant atskirų dalių apibendrinimų, suformuluojamos svarbiausios darbo išvados, rekomendacijos bei pasiūlymai.

Conclusions

Šiame skyriuje pateikiamos išvados (reziume) anglų kalba.

Literatūra

[Ciu01] Marcin Ciura. Best increments for the average case of shellsort. *International Symposium on Fundamentals of Computation Theory*, p.p. 106–117. Springer, 2001.

[RBH+02] Robert S Roos, Tiffany Bennett, Jennifer Hannon ir Elizabeth Zehner. A genetic algorithm for improved shellsort sequences. *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, p.p. 694–694, 2002.

[SY99] Richard Simpson ir Shashidhar Yachavaram. Faster shellsort sequences: A genetic algorithm application. *Computers and Their Applications*, p.p. 384–387, 1999.

[Tok92] Naoyuki Tokuda. An Improved Shellsort. *Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing '92, Volume 1 - Volume I*, p.p. 449–457, NLD. North-Holland Publishing Co., 1992. ISBN: 044489747X.