VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
INFORMATIKOS KATEDRA

Kursinis darbas

Rikiavimo tobulinimas genetiniais algoritmais

(Improving sorting with genetic algorithms)

Atliko: 3 kurso 2 grupės studentas
Deividas Zaleskis (parašas)

Darbo vadovas:
lekt. Irmantas Radavičius (parašas)

Vilnius
2021

Turinys

Įvadas

Darbo tikslas: pritaikyti genetinius algoritmus Šelo algoritmo tarpų sekoms generuoti.
Darbo uždaviniai:

- Nustatyti kriterijus tarpų sekų efektyvumui įvertinti.
- Paruošti aplinką eksperimentų vykdymui.
- Naudojant genetinius algoritmus sugeneruoti pasirinktas tarpų sekas.
- Atliekant eksperimentus įvertinti sugeneruotų ir pateiktų literatūroje tarpų sekų efektyvumą.

Viena pagrindinių informatikos sąvokų yra algoritmas. Formaliai algoritmą galima apibūdinti kaip baigtinę seką instrukcijų, nurodančių kaip rasti nagrinėjamo uždavinio sprendinį. Algoritmo koncepcija egzistuoja nuo senovės laikų [Knu72], tačiau atsiradus kompiuteriams, tapo įmanoma algoritmų vykdymą automatizuoti, paverčiant juos mašininiu kodu suprantamu kompiuteriams [WWG51]. Taip informatikos mokslas nuo teorinių šaknų [Tur37] įgavo ir taikomąją pusę. Beveik visus algoritmus galima suskirstyti į dvi klases: kombinatorinius algoritmus ir skaitinius algoritmus. Skaitiniai algoritmai sprendžia tolydžius uždavinius: optimizuoti realaus argumento funkciją, išspręsti tiesinių lygčių sistemą su realiais koeficientais, etc. Kombinatoriniai algoritmai sprendžia diskrečius uždavinius ir operuoja su diskrečiais objektais: skaičiais, sąrašais, grafais, etc. Vienas žinomiausių diskretaus uždavinio pavyzdžių yra duomenų rikiavimas.

Duomenų rikiavimas yra vienas pamatinių informatikos uždavinių. Matematiškai jis formuluojamas taip: duotai baigtinei palyginamų elementų sekai $S = (s_1, s_2, ..., s_n)$ pateikti tokį kėlinį, kad pradinės sekos elementai būtų išdėstyti didėjančia (mažėjančia) tvarka [RB13]. Rikiavimo uždavinys yra aktualus nuo pat kompiuterių atsiradimo ir buvo laikomas vienu pagrindinių diskrečių uždavinių, kuriuos turėtų gebėti spręsti kompiuteris [Knu70]. Rikiavimo uždavinio sprendimas dažnai padeda pagrindą efektyviam kito uždavinio sprendimui, pavyzdžiui, atliekant paiešką sąraše, galima taikyti dvejetainės paieškos algoritmą tik tada, kai sąrašas yra išrikiuotas. Kadangi rikiavimo uždavinys yra fundamentalus, jam spręsti egzistuoja labai skirtingų algoritmų.

Rikiavimo algoritmų yra įvairių: paremtų palyginimu (elementų tvarką nustato naudojant palyginimo operatorius), stabilių (nekeičia lygių elementų tvarkos), rikiuojančių vietoje (nenaudoja pagalbinių duomenų struktūrų), etc. Asimptotinis rikiavimo algoritmų sudėtingumas laiko atžvilgiu taip pat skiriasi: bogo rikiavimo (angl. bogosort) [GHR07] blogiausiu atveju atliekamas palyginimų skaičius yra neribotas, rikiavimas įterpimu (angl. insertion sort) blogiausiu atveju atlieka $O(n^2)$ palyginimų [BFM06], o asimptotiškai optimalūs palyginimu paremti algoritmai, pavyzdžiui, krūvos rikiavimas (angl. heapsort) [For64] blogiausiu atveju atlieka $O(n \log n)$ palyginimų [SS93]. Nepaisant rikiavimo algoritmų įvairovės, nėra algoritmo, kuris būtų geriausias visais atvejais, nes praktinis efektyvumas priklauso nuo daugelio veiksnių.

Šelo rikiavimo algoritmas (angl. shellsort) [She59] yra paremtas palyginimu, rikiuojantis vietoje ir nestabilus. Šelo algoritmą galima laikyti rikiavimo įterpimu modifikacija, kuri lygina ne

gretimus, o toliau vienas nuo kito esančius elementus, taip paspartindama jų perkėlimą į galutinę poziciją. Pagrindinė algoritmo idėja - išskaidyti rikiuojamą seką S į posekius $S_1, S_2, ..., S_n$, kur kiekvienas posekis $S_i = (s_i, s_{i+h}, s_{i+2h}, ...)$ yra sekos S elementai, kurių pozicija skiriasi h. Išrikiavus visus sekos S posekius S_i su tarpu h, seka tampa h-išrikiuota. Remiantis tuo, jog sekai S esant h-išrikiuota ir ją k-išrikiavus, ji lieka h-išrikiuota [GK72], galima kiekvieną algoritmo iteraciją mažinti tarpą, taip vis didinant sekos S išrikiuotumą. Pritaikant šias idėjas ir rikiavimui naudojant mažėjančią tarpų seką su paskutiniu nariu 1, kuris garantuoja rikiavimą įterpimu paskutinėje iteracijoje, galima užtikrinti, jog algoritmo darbo pabaigoje seka S bus pilnai išrikiuota. Įvertinant aukščiau aprašytas mintis, nesunku pastebėti tarpų sekų įtaką Šelo rikiavimo algoritmo veikimui.

Šelo rikiavimo algoritmo efektyvumas tiesiogiai priklauso nuo pasirinktos tarpų sekos. Weiss atlikto tyrimo [Wei91] rezultatai rodo, jog su Sedgewick pasiūlyta seka algoritmas veikia beveik dvigubai greičiau nei Šelo pradinis variantas, kai $n = 1000000$. Yra įrodyta, kad šio algoritmo blogiausiu atveju atliekamų palyginimų skaičiaus apatinė riba yra $\Omega(\frac{n\,log^2\,n}{log\,log\,n^2})$ [PPS92], taigi jis nėra asimptotiškai optimalus. Tiesa, kol kas nėra rasta seka, su kuria algoritmas pasiektų šią apatinę ribą. Kiek žinoma autoriui, asimptotiškai geriausia tarpų seka yra rasta Pratt, kuri yra formos 2^p3^p ir turi $\Theta(n\,log^2\,n)$ asimptotinį sudėtingumą [Pra72], tačiau praktikoje ji veikia lėčiau už Ciura [Ciu01] ar Tokuda [Tok92] pasiūlytas sekas. Daugelio praktikoje naudojamų sekų asimptotinis sudėtingumas laiko atžvilgiu lieka atvira problema, nes jos yra rastos eksperimentiškai, tad sunku rasti matematinį modelį tinkamą jų analizei. Vienas iš metodų, kuriuos galima taikyti efektyvių tarpų sekų radimui, yra genetinis algoritmas.

Genetinis algoritmas yra metaeuristika (metodas rasti euristikas), paremta biologijos žiniomis apie natūralios atrankos procesą. Genetiniai algoritmai taikomi sprendžiant paieškos ir optimizavimo uždavinius ir naudoja biologijos įkvėptus paveldėjimo, mutacijos, atrankos bei rekombinacijos operatorius.

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.

1.1. Poskyris

Citavimo pavyzdžiai nebereikalingi.

1.1.1. Skirsnis

1.1.1.1. Straipsnis

1.1.2. Skirsnis

2. Skyrius

2.1. Poskyris

2.2. Poskyris

Išvados

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

Literatūra

[BFM06] Michael A Bender, Martin Farach-Colton ir Miguel A Mosteiro. Insertion sort is O (n log n). *Theory of Computing Systems*, 39(3):391–397, 2006.

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

[For64] G. E. Forsythe. Algorithms. *Commun. ACM*, 7(6):347–349, 1964-06. ISSN: 0001-0782. DOI: 10.1145/512274.512284. URL: https://doi.org/10.1145/512274.512284.

[GHR07] Hermann Gruber, Markus Holzer ir Oliver Ruepp. Sorting the Slow Way: An Analysis of Perversely Awful Randomized Sorting Algorithms. Pierluigi Crescenzi, Giuseppe Prencipe ir Geppino Pucci, redaktoriai, *Fun with Algorithms*, p.p. 183–197, Berlin, Heidelberg. Springer Berlin Heidelberg, 2007. ISBN: 978-3-540-72914-3.

[GK72] David Gale ir Richard M. Karp. A phenomenon in the theory of sorting. *Journal of Computer and System Sciences*, 6(2):103–115, 1972. ISSN: 0022-0000. DOI: https://doi.org/10.1016/S0022-0000(72)80016-3. URL: https://www.sciencedirect.com/science/article/pii/S0022000072800163.

[Knu70] Donald E. Knuth. Von Neumann's First Computer Program. *ACM Comput. Surv.*, 2(4):247–260, 1970-12. ISSN: 0360-0300. DOI: 10.1145/356580.356581. URL: https://doi.org/10.1145/356580.356581.

[Knu72] Donald E Knuth. Ancient babylonian algorithms. *Communications of the ACM*, 15(7):671–677, 1972.

[PPS92] C. G. Plaxton, B. Poonen ir T. Suel. Improved lower bounds for Shellsort. *Proceedings., 33rd Annual Symposium on Foundations of Computer Science*, p.p. 226–235, 1992. DOI: 10.1109/SFCS.1992.267769.

[Pra72] Vaughan R Pratt. Shellsort and sorting networks. Tech. atask., STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1972.

[RB13] Irmantas Radavičius ir Mykolas Baranauskas. An empirical study of the gap sequences for Shell sort. *Lietuvos matematikos rinkinys*, 54(A):61–66, 2013-12. DOI: 10.15388/LMR.A.2013.14. URL: https://www.journals.vu.lt/LMR/article/view/14899.

[She59] D. L. Shell. A High-Speed Sorting Procedure. *Commun. ACM*, 2(7):30–32, 1959-07. ISSN: 0001-0782. DOI: 10.1145/368370.368387. URL: https://doi.org/10.1145/368370.368387.

[SS93] R. Schaffer ir R. Sedgewick. The Analysis of Heapsort. *Journal of Algorithms*, 15(1):76–100, 1993. ISSN: 0196-6774. DOI: https://doi.org/10.1006/jagm.1993.1031. URL: https://www.sciencedirect.com/science/article/pii/S019667748371031X.

[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.

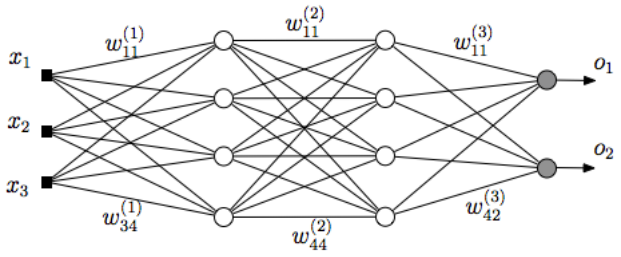
[Tur37] Alan Mathison Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London mathematical society*, 2(1):230–265, 1937.

[Wei91] Mark Allen Weiss. Short Note: Empirical study of the expected running time of Shellsort. *The Computer Journal*, 34(1):88–91, 1991.

[WWG51] Maurice V Wilkes, David J Wheeler ir Stanley Gill. *The Preparation of Programs for an Electronic Digital Computer: With special reference to the EDSAC and the Use of a Library of Subroutines*. Addison-Wesley, 1951.

Priedas Nr. 1

Priedas 1



1 pav. Paveikslėlio pavyzdys