Logo

Description automatically generated­­­

VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

**Našieji skaičiavimai**

Laboratorinių darbų ataskaitos

Atliko: Deividas Roščenkovas

VU el. p.: [deividas.roscenkovas@mif.stud.vu.lt](mailto:deividas.roscenkovas@mif.stud.vu.lt)

Vilnius

2024

Turinys

[Laboratorinis darbas Nr. 1 3](#_Toc163607512)

[Algoritmo analizė 3](#_Toc163607513)

[Bendros atminties lygiagrečiojo algoritmo sudarymas 3](#_Toc163607514)

[Pirma dalis 3](#_Toc163607515)

[Antra dalis 5](#_Toc163607516)

# Laboratorinis darbas Nr. 1

## ­­Algoritmo analizė

Visą algoritmą sudaro trys pagrindinės dalys:

* Duomenų įkėlimas
* Atstumų tarp taškų, suskaičiuotų naudojant Haversino formulę, matricos skaičiavimas
* Sprendinio (*n* taškų aibė *X*) paieška naudojant pilną perrinkimą

Parinkti atitinkami algoritmo parametrai:

* Duomenų aibės dydis (*num\_points*) – 5000
* Ieškomos taškų aibės X dydis (num\_variables) – 3
* Galimų vietų skaičius (*num\_possible\_locations*) – 60

Naudojant šiuos parametrus, atstumų matricos skaičiavimas užtrunka 4,517 sekundžių, o sprendinio paieškos laikas 14,386 sekundžių. Duomenų įkėlimas trunka 0,007 sekundžių.

Taigi, bendras algoritmo vykdymo laikas yra 18,910 sekundžių.

Laikydami duomenų įkėlimą ir atstumų matricos skaičiavimą nuosekliąja algoritmo dalimi (α), o sprendinio paiešką lygiagrečiąja dalimi (β), gauname jų proporcijas – α = 0,239 ir β = 0,761.

Apskaičiuojame teorinius galimus algoritmo pagreitėjimus (Sp):

* S2 = 1,614
* S4 = 2,330
* Smax = 4,184

## Bendros atminties lygiagrečiojo algoritmo sudarymas

### Pirma dalis

Lygiagretiname tik sprendinio paiešką. Iš pradžių sukuriame parelelinį regioną, į jį perkeliame visą sprendimo paieškos algoritmą, išskyrus geriausio sprendimo ir geriausio sprendimo tikslo funkcijos kintamuosius (*best\_solution* ir *f\_best\_solution*). Algoritmą padalinsime tolygiai visoms gijoms. Kiekviena gija apskaičiuoja savo pradinį ir galinį indeksą:

* ***pradžia = gijos\_numeris \* kombinacijų\_kiekis / gijų\_skaičius***
* ***pabaiga = (gijos\_numeris + 1) \* kombinacijų\_kiekis / gijų skaičius – 1***

Kadangi apskaičiuoti tikslias pradines kombinacijas kiekvienai gijai, pagal jų indeksą yra brangu, bandome kiek paprastesnį ir efektyvesnį sprendimą. Kiekvieną giją pradėsime skaičiuoti nuo tos pačios kombinacijos [0,1,2]. Jeigu gija nėra savo pradžios ir pabaigos indeksų rėžyje , ji praleis visus atliekamus skaičiavimus, kol neatsidurs savo režyje. Sukuriame lokalujį kintamąjį *k* su pradine reikšme 0, kuris reikš gijos dabartinės ciklo iteracijos numerį. Kai *k* pasieks gijos apskaičiuotą pradžios indeksą, nuo tada ta gija pradės atlikinėti sprendinio paieškos skaičiavimus su atitinkamomis kombinacijomis. Kai *k* pasieks pabaigos indeksą, gija sustabdys savo ciklą ir baigs darbą. Gija suskaičiavusi sprendimo tikslo funkciją, palygina ja su globaliąja geriausia apskaičiuota tikslo funkcija. Jeigu rezultatas geresnis (mažesnis), gija priskiria globaliajai geriausiąjai reikšmei savo naujai apskaičiuotą reikšmę. Šis palyginimas ir priskyrimas gali vykti vienu metu tik vienoje gijoje (*#pragma omp critical* direktyva).

**Ciklo pseudokodas:**

***kol kitas sprendinys egzistuoja {***

***jei k >= pradžia {***

***jei k == pabaiga:***

***SUSTABDYTI CIKLĄ***

***APSKAIČIUOTI DABARTINĮ SPRENDINĮ***

***omp kritinė zona {***

***jei dabartinis sprendinys geresnis už esamą:***

***GERIAUSIAS SPRENDINYS = DABARTINIS SPRENDINYS***

***}***

***}***

***PADIDINTI k VIENETU***

***}***

Žemiau pateikiami eksperimentiniu būdu nustatyti algoritmo su išlygiagretinta sprendimo paieška pagreitėjimų rezultatai (žr. lentelė 1, pav. 1).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Procesorių sk.** | **Teorinis pagreitėjimas** | **Viso algoritmo pagreitėjimas** | **Lygiagretinamosios dalies pagreitėjimas** |  |
| 1 | 1 | 1 | 1 |  |
| 2 | 1,614 | 1,518 | 1,749 |  |
| 4 | 2,330 | 2,230 | 3,474 |  |

lentelė 1 Pagreitėjimai

pav. 1 Pagreitėjimai

### Antra dalis

Lygiagretiname matricos skaičiavimą. Išoriniam ciklui išlygiagretinti naudojame dinaminį užduočių paskirstymą (*#pragma omp parallel for schedule(dynamic)* direktyva). Matrica yra trikampė, skirtingos išorinio ciklo iteracijos atlieka skirtingą kiekį darbo, todėl statinis užduočių paskirstymas yra netinkamas. Naudojant dinaminį užduočių paskirstymą, iteracijų blokai yra sudaromi bei priskiriami laisvoms gijoms ciklo vykdymo metu.

Žemiau pateikiami eksperimentiniu būdu nustatyti algoritmo su išlygiagretintu matricos skaičiavimu ir išlygiagretinta sprendimo paieška pagreitėjimų rezultatai (žr. lentelė 1, pav. 1).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Procesorių skaičius** | **Teorinis pagreitėjimas** | **Viso algoritmo pagreitėjimas** | **Matricos skaičiavimo pagreitėjimas** | **Sprendimo paieškos pagreitėjimas** |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1,998 | 1,977 | 1,992 | 1,749 |
| 4 | 3,988 | 3,911 | 3,942 | 3,474 |

lentelė 2 Pagreitėjimai

pav. 2 Pagreitėjimai