Lygiagretaus programavimo laboratorinių darbų analizė

Pijus Petkevičius

October 23, 2022

Contents

1. 1 laboratorinis darbas

1.1. Aprašymas

1.1.1. Įžanga

Aibę A sudaro geografiniai taškai, nurodant platumos ir ilgumos koordinates. Iš šios aibės reikia parinkti taškų aibę X tokią, kad atstumų nuo kiekvieno aibės A taško iki jam artimiausio aibės X taško suma būtų minimali $X \subset A$.

Faile lab_data.dat pateikti 50000 geografinių taškų, kur viena eilutė aprašo vieno geografinio taško koordinates.

Faile lab_01_2_algorithm.cpp pateikta programa, kuri randa nurodyto n taškų aibę X, atitinkančią uždavinio sąlygą, naudojant paprastosios atsitiktinės paieškos (angl. Pure Random Search, PRS) algoritmą.

Pagrindiniai algoritmo parametrai (globalūs kintamieji):

- num_points: duomenų aibės A dydis (max 50000)
- num_variables: ieškomos taškų aibės X dydis
- num_iterations: sprendinio paieškai skirtų iteracijų skaičius (kuo daugiau, tuo didesnė tikimybė rasti geresnį sprendinį).

Algoritmų vykdymo pradžioje sudaroma atstumų matrica, kurioje saugomi atstumai kilometrais tarp taškų, suskaičiuoti pagal Haversino formulę. Atsižvelgiant į tai, kad atstumas nuo taško a iki taško b yra lygus atstumui nuo taško b iki taško a, yra užpildoma tik pusė matricos. Šioje matricoje saugomi atstumai yra naudojami vykdant aibės X taškų paiešką.

1.1.2. Užduotis

1. Pasirinkti duomenų aibės dydį ir algoritmo iteracijų skaičių, kad atstumų matricos skaičiavimas užtruktų ne mažiau 10 sekundžių, o sprendinio paieškos laikas būtų nemažesnis nei 20 sekundžių.
2. Duomenų įkėlimą ir atstumų matricos skaičiavimą laikyti nuosekliąja algoritmo dalimi, o sprendinio paiešką - lygiagretinama dalimi, įvertinti teorinius galimus algoritmo pagreitėjimus naudojant 2 ir 4 procesorius, bei didžiausią galimą pagreitėjimą.
3. Duomenų įkėlimą ir atstumų matricos skaičiavimą laikyti nuosekliąja algoritmo dalimi, sudarykite lygiagretųjį bendros atminties algoritmą ir eksperimentiniu būdu ištirkite jo pagreitėjimą naudodami 2 ir 4 procesorius.
4. Sudarykite lygiagretų bendros atminties algoritmą atstumų matricos skaičiavimui ir eksperimentiniu būdu ištirkite jo pagreitėjimą naudodami 2 ir 4 procesorius.
5. Pananalizuoti, kai matricos reikšmių suskaičiavimą lygiagrečiąja dalimi, o pure random search (PRS), nuosekliąja.

1.2. Kompiuterinės įrangos ir parametrų pasirinkimas

Algoritmo analizei buvo naudojama **Apple Mac Mini Desktop Computer, 3.2GHz 6-Core Intel Core i7** kompiuteris, kurio dėka, buvo galima paleisti ant 2, 4 ir 6 procesorių. Kad įgyvendinti **1** nurodymą, buvo pasirinkta:

- num_points = 12000
- num_iterations = 30000

Duomenų nuskaitymas (s)	Atstumų matricos skaičiavimas (s)	PRS skaičiavimas (s)
0.003	10.312	19.955
0.004	10.315	19.993
0.003	10.321	19.967

1.3. 2 nurodymo teoriniai įverčiai

Paleidus programą 3 kartus, gauti skaičiavimo dalių rezultatai:

Duomenų nuskaitymas (s)	Atstumų matricos skaičiavimas (s)	PRS skaičiavimas (s)
0.004	10.316	19.972

Pagal 2 nurodymą, nuosekliąja dalimi (α) laikoma duomenų nuskaitymas ir atstumų matricos skaičiavimas, o lygiagrečiąja (β)- PRS skaičiavimas.

$$\alpha = \frac{\text{nuoseklioji dalis}}{\text{visas laikas}}$$

$$\beta = \frac{\text{lygiagrečioji dalis}}{\text{visas laikas}}$$

Gauname kad:

$$\alpha = 0.341$$

$$\beta = 0.659$$

Teorinis pagreitėjimas naudojant p procesorių:

$$S_p = \frac{1}{\alpha + \frac{\beta}{p}}$$

$$S_2 = \frac{1}{0.341 + \frac{0.659}{2}} = 1.492$$

$$S_4 = \frac{1}{0.341 + \frac{0.659}{4}} = 1.978$$

Teorinis maksimumas pagal Andalo Dėsnį:

$$S_{max} = \lim_{p \to \infty} \frac{1}{\alpha + \frac{\beta}{p}} = \frac{1}{\alpha} = \frac{1}{0.341} = 2.935$$

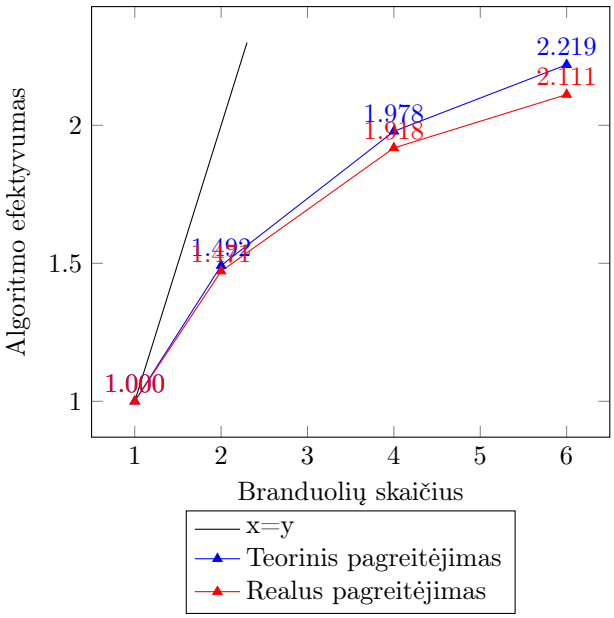
1.4. Pure random search (PRS)

pakeisti i pseudo koda

```
int *best_solution = new int[num_variables];
double f_solution, f_best_solution = 1e10;
#pragma omp parallel reduction (min: f_best_solution ) private (f_solution)
#pragma omp for schedule(dynamic)
for (int i=0; i<num_iterations; i++) {
    int *solution = new int[num_variables];
    random_solution(solution);
    f_solution = evaluate_solution(solution);
    if (f_solution < f_best_solution) {
        (mazesnis) uz geriausia zinoma
        f_best_solution = f_solution;
        if(f_best_solution == f_solution){
            #pragma omp critical (DataCollection)
            {
                for (int j=0; j<num_variables; j++) {
                    best_solution[j] = solution[j];
                }
            }
        }
    }
}
```

```
  double f_best_solution = 1e10;
int *best_solution= new int[num_variables];
#pragma omp parallel reduction(min: f_best_solution)
{
int *best_solution_tmp = new int[num_variables];
double f_solution, f_best_solution_tmp = 1e10;
#pragma omp for schedule(dynamic)
for (int i=0; i<num_iterations; i++) {
int *solution = new int[num_variables];
random_solution(solution);
f_solution = evaluate_solution(solution);
if (f_solution < f_best_solution_tmp) {
f_best_solution_tmp = f_solution;
for (int j=0; j<num_variables; j++) {
best_solution_tmp[j] = solution[j];
}
}
}
f_best_solution = f_best_solution_tmp;
#pragma omp barrier
if(f_best_solution == f_best_solution_tmp){
for (int j=0; j<num_variables; j++) {
best_solution[j] = best_solution_tmp[j];
}
}
}
```

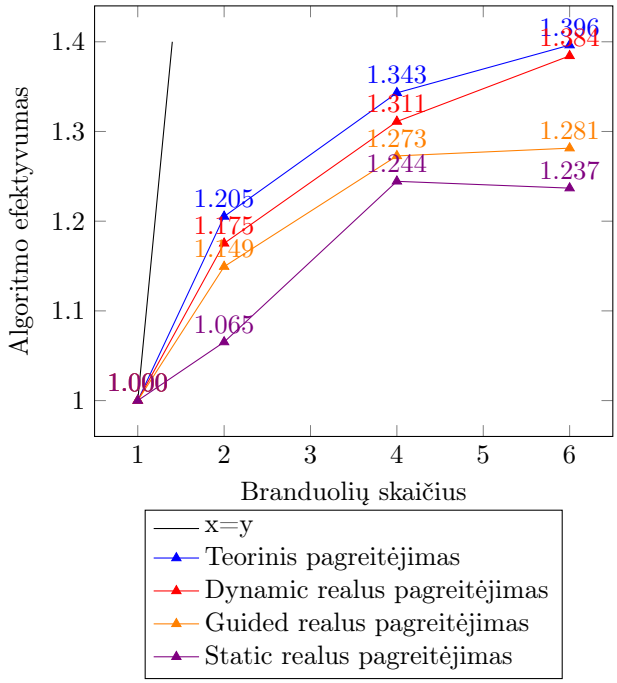
PRS lygiagretinimas:


Algoritmo efektyvumas
Branduolių skaičius
1
1.5
2
1
2
3
4
5
6
2.219
2.111
1.978
1.918
1.493
1.471
1.000
x=y
Teorinis pagreitėjimas
Realus pagreitėjimas

1.4.1. Atstumų matricos skaičiavimas

```
#pragma omp parallel for schedule(dynamic)
for (int i=0; i<num_points; i++) {
    distance_matrix[i] = new double[i+1];
    for (int j=0; j<=i; j++) {
        distance_matrix[i][j] = Haversine_distance(points[i][0], points[i][1], points[j][0], poi
    }
}
```

Matricos skaiciavimo laikai:



1.5. Rezultatų analizė

Abu sulygiagrerinti: