Lygiagretaus programavimo laboratorinių darbų analizė

Pijus Petkevičius

October 13, 2024

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. Ištirti algoritmo pagreitėjimo priklausomybes nuo procesorių skaičiaus, kai matricos skaičiavimas ir sprendinio paieška išlygiagretinti.

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

Lentelė 1: Algoritmo skaičiavimo dalių rezultatai, naudojant **Mac Mini** kompiuterį

1.3. 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

Lentelė 2: Nuoseklaus algoritmo skačiavimo dalių rezultatai

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)

Bandymas išlygiagretinti PRS algoritmą, buvo atliktas 2 būdais.

1 algoritme buvo pasitelkta **Dynamic** scheduling strategija, ir reduction min: f_best_solution. Kievieną karta, kai randamas geresnė sprendinio reikšmė, ji priskiriama f_best_solution kintamajam(jis automatiškai pasiima tik mažesnę reikšmę). Vėliau viskas geriausio sprendinio reikšmės buvo išsaugojamos naudojant critical žymę:

```
int *best_solution;
double f_solution, f_best_solution;
#pragma omp parallel reduction (min: f_best_solution ) private (f_solution)
#pragma omp for schedule(dynamic)
for (int i=0; i<num_iterations; i++) {
    // random find and evaluate solution
    f_solution = evaluate_solution(solution);
    if (f_solution < f_best_solution) {
        f_best_solution = f_solution;
        if(f_best_solution == f_solution){
            #pragma omp critical (DataCollection)
            {
               // copy solution values to the best_solution array
            }
        }
    }
}
```

Pseudokodas 1: PRS pirmas algoritmas

4 antrajame algoritme veikimo principas gana panašus, tik ciklas paskirstomas keliems branduoliams, randamas lokali geriausia f_best_solution_tmp reikšmė ir po ciklo ji priskiriama f_best_solution ir išsaugojamos geriausio sprendinio reikšmės:

```
double f_best_solution;
int *best_solution;
#pragma omp parallel reduction(min: f_best_solution)
{
    int *best_solution_tmp;
    double f_solution, f_best_solution_tmp;
    #pragma omp for schedule(dynamic)
    for (int i=0; i<num_iterations; i++) {
        // random find and evaluate solution
        f_solution = evaluate_solution(solution);
        if (f_solution < f_best_solution_tmp) {
            f_best_solution_tmp = f_solution;
           // copy solution values to the best_solution_tmp array
        }
    }
    f_best_solution = f_best_solution_tmp;
    #pragma omp barrier
    if(f_best_solution == f_best_solution_tmp){
        // copy best_solution_tmp values to the best_solution array
    }
}
```

Pseudokodas 2: PRS antras algoritmas

Abu algoritmai buvo ištestuoti su 2, 4 ir 6 branduoliais ir rezultatai matomi **1** diagramoje:

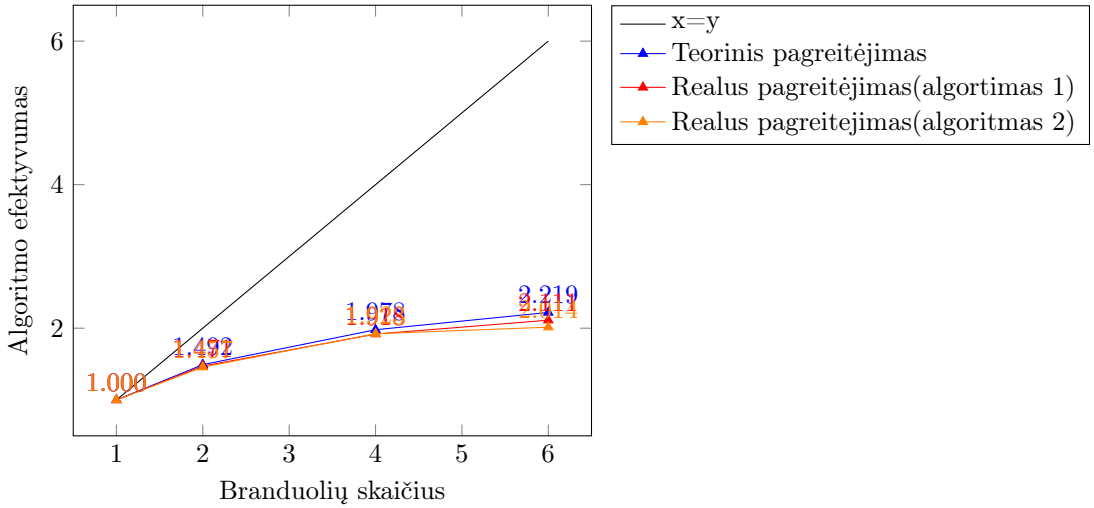

Diagrama 1: PRS algoritmo lygiagretinimo diagrama

Pastebime, kad **1** ir **4** algoritmai su 2 ir 4 branduoliais turėjo ganėtinai panašų efektyvumą:

Branduolių skaicius	1 algoritmas	2 algoritmas
2	1.471	1.457
4	1.918	1.922
6	2.111	2.014

Lentelė 3: Algoritmų efektuvumo priklausomybė nuo branduolių skaičiaus

Tačiau, kai branduolių skaičius pasiekia 6, 4-sis algoritmas nusileidžia efektyvumu 1-jam. **1-jį** algoritmą naudosime tolimesniuose 1 užduoties eksperimentuose.

1.5. Atstumų matricos skaičiavimas

```
#pragma omp parallel for schedule(dynamic)
for (int i=0; i<num_points; i++) {
    ...
    for (int j=0; j<=i; j++) {
        distance_matrix[i][j] = Haversine_distance(...);
    }
}
```

Pseudokodas 3: Atstumų matricos skaičiavimas

3 algoritmas, eksperimentiniu būdu buvo išbandytos įvairios lygiagretinimo strategijos (**Dynamic**,**Guided**, **Static**). Gauti rezultatai matomi **2** diagramoje:

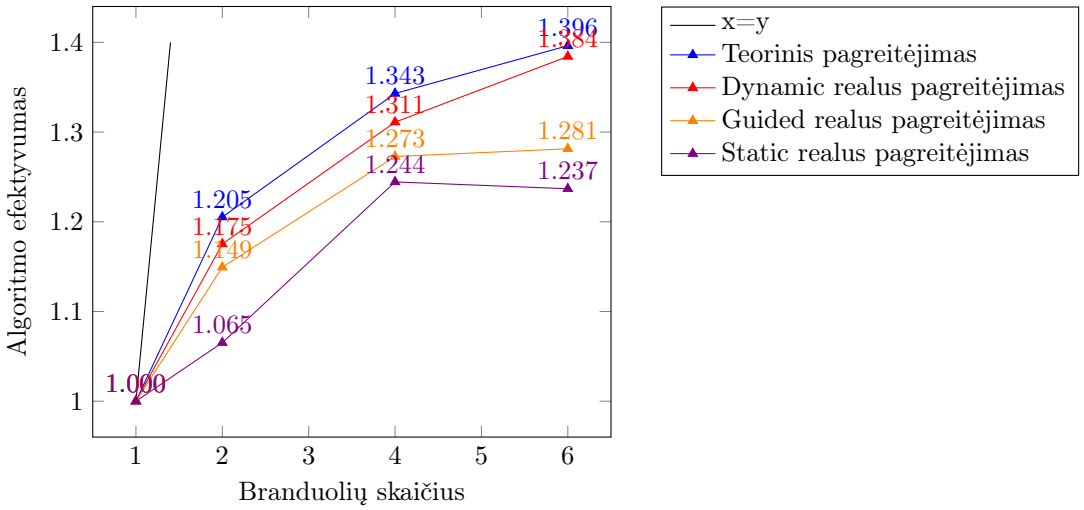

Diagrama 2: Įvairių strategijų efektyvumo diagrama

Pastebime, jog **Dynamic** yra kur kas efektyvesnis, lyginant su **Static** ir **Guided** lygiagretinimo strategijomis ir kur kas labiau priartėja prie teorinio pagreitėjimo.

Dynamic direktyvą naudosime tolimesniuose 1 užduoties eksperimentuose.

1.6. Atstumų matricos ir PRS dalių lygiagretinimo rezultatai

Iš ankstesnių eksperimentų radome, jog **1** PRS algoritmas ir **Dynamic** direktyva atstumų matricai skaičiuoti buvo efektyviausi sprendimo būdai. Eksperimentiniu būdu ištestavus progamą, buvo gauti tokie rezultatai:

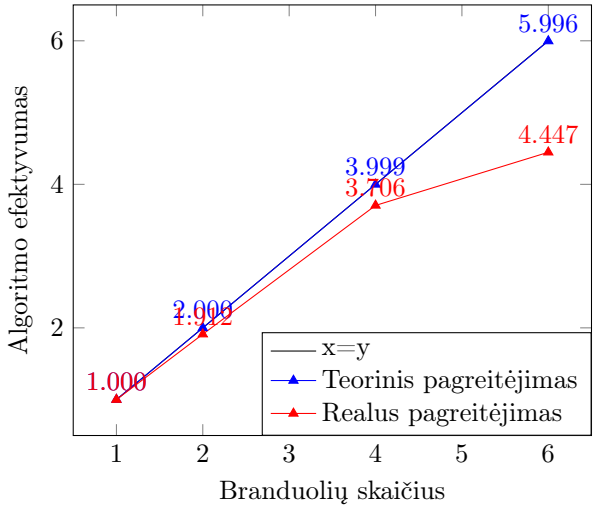

Diagrama 3: Efektyvumo diagrama, kai PRS ir matricos skaičiavimas lygiagretinami

2. 2 laboratorinis darbas

2.1. Aprašymas

2.1.1. Užduotis

Sudarytą bendros atminties lygiagretųjį algoritmą perdarykite į paskirstytos atminties algoritmą ir ištirkite jo pagreitėjimą naudodami 2, 4 ir 8 procesorius (naudokite uosis.mif.vu.lt skaičiavimų resursus). Ataskaitoje aprašykite kokiu būdu paskirstėte darbą skaičiavimo mazgams, kokiu būdu vyko komunikacija tarp jų.

Viename grafike pavaizduokite tiesinį pagreitėjimą ir eksperimentų būdu nustatytą atstumų matricos skaičiavimo pagreitėjimo, sprendinio skaičiavimo pagreitėjimo, bei viso algoritmo (kai matricos skaičiavimas ir sprendinio paieška išlygiagretinti) pagreitėjimo priklausomybes nuo procesorių skaičiaus.

2.2. PRS

2.2.1. Aprašas

Kiekviena gija apskaičiuoja geriausią rezultatą kiekvienoje gijoje, tada 0-gija surenka visų giju rezultatus.

2.2.2. Pseudokodas

```
double f_solution, f_best_solution = 1e10;
int it = 1;
int iterations = num_iterations/ world_size;
int remainder = num_iterations % world_size;
iterations += remainder > world_rank ? 1 : 0;
for (int i=0; i<iterations; i++) {
    // random find and evaluate solution
    f_solution = evaluate_solution(solution);
    if (f_solution < f_best_solution) {
        f_best_solution = f_solution;
        if(f_best_solution == f_solution){
           // copy solution values to the best_solution array
        }
    }
}
if(world_rank == 0){
    // collect f_best_solution value for each thread, save best value and thread
    int rankId = -1;
    for(int j=1;j<world_size;j++){
        double response;
        MPI_Recv(&response, 1, MPI_DOUBLE, j, GET_BEST_SOLUTION_VALUE, MPI_COMM_WORLD, &stat);
        if(response < f_best_solution) {
            rankId = j;
            f_best_solution = response;
        }
    }

    if(rankId != 0 && rankId != -1){
        // retrieve best solution from thread
    }
    for(int j=1;j<world_size;j++){
        // stop worker threads
    }
}
else {
    do{
        MPI_Recv(&buff, 1, MPI_INT, 0, MPI_ANY_TAG, MPI_COMM_WORLD, &stat);
        if(stat.MPI_TAG!=STOP_THREADS){
            if(stat.MPI_TAG == GET_BEST_SOLUTION_VALUE) {
                // returns f_best_solution
            }
            if(stat.MPI_TAG == GET_BEST_SOLUTION){
                // returns best_solution
            }
        }
    } while(stat.MPI_TAG!=STOP_THREADS);
}
```

Pseudokodas 4: PRS mpi pseudokodas

2.3. Atstumų matricos skaičiavimas

2.3.1. Aprašas

Algoritmo principas toks:

1. gija- vadovas skirsto matricą darbininkams, vėliau surenka apskaičiuotus rezultatus, praneša, kad darbininkai gali baigti darbą(dėl to efektyvumas = gijų skaičius - 1).
2. darbininkai gauna matricos eilutę, apskaičiuoja resultatą, gražina vadovui ir laukia, kada galės baigti savo darbą.

2.3.2. Pseudokodas

```
distance_matrix = new double*[num_points];
if(world_rank == 0){
    for (int i=0; i<num_points; i=i+(world_size-1)) {
        for(int j=1;j<world_size;j++){ // send to all threads array item to calculate
            if(i+j <=num_points){
                int temp = i + j - 1;
                MPI_Send(&temp, 1, MPI_INT, j, 0, MPI_COMM_WORLD);
            }
        }

        for(int j=1;j<world_size;j++){
            // receive calculated distance_matrix
        }
    }
    for(int j=1;j<world_size;j++){// stop slave thread work
        MPI_Send(&buff, 1, MPI_INT, j, STOP_THREADS, MPI_COMM_WORLD);
    }
}
else{
    do{
        MPI_Recv(&buff, 1, MPI_INT, 0, MPI_ANY_TAG, MPI_COMM_WORLD, &stat);
        //receive matrix, calculate,ant return
        if(stat.MPI_TAG!=STOP_THREADS){
            double *localDistances = new double[buff + 1];
            localDistances ... = Haversine_distance(...)
            MPI_Send(localDistances, buff + 1, MPI_DOUBLE, 0, stat.MPI_TAG, MPI_COMM_WORLD);
        }
    } while(stat.MPI_TAG!=STOP_THREADS);

}

MPI_Barrier(MPI_COMM_WORLD);

//Broadcast to all threads
```

Pseudokodas 5: Atstumų matricos mpi pseudokodas

2.4. Atstumų matricos ir PRS dalių lygiagretinimos rezultatai

Išlygiagretinus abi dalis gauname tokį grafiką:

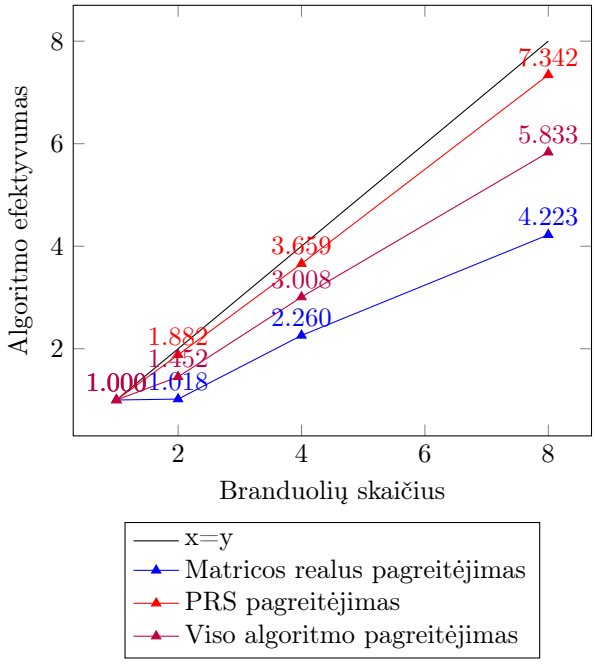

Diagrama 4: Efektyvumo diagrama, kai PRS ir matricos skaičiavimas lygiagretinami

3. 3 laboratorinis darbas

3.1. Aprašymas

3.1.1. Užduotis

Sudarytą paskirstytos atminties lygiagretųjį algoritmą paleiskite (MIF superkompiuteryje) ir atlikite skaičiavimus naudodami 2, 4 ir 8 arba daugiau procesorių.

Viename grafike pavaizduokite tiesinį pagreitėjimą ir eksperimentų būdu nustatytą atstumų matricos skaičiavimo pagreitėjimo, sprendinio skaičiavimo pagreitėjimo, bei viso algoritmo (kai matricos skaičiavimas ir sprendinio paieška išlygiagretinti) pagreitėjimo priklausomybes nuo procesorių skaičiaus.

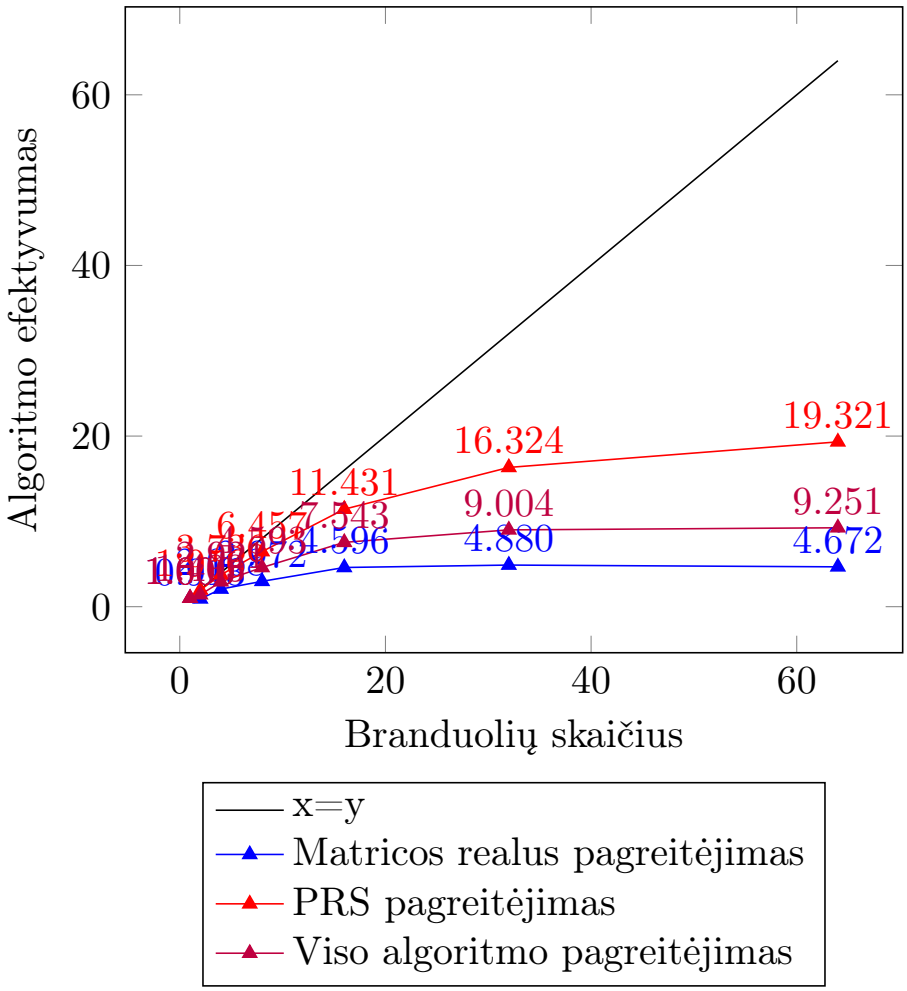

Diagrama 5: Efektyvumo diagrama, kai PRS ir matricos skaičiavimas lygiagretinami (leista ant MIF superkompiuterio)

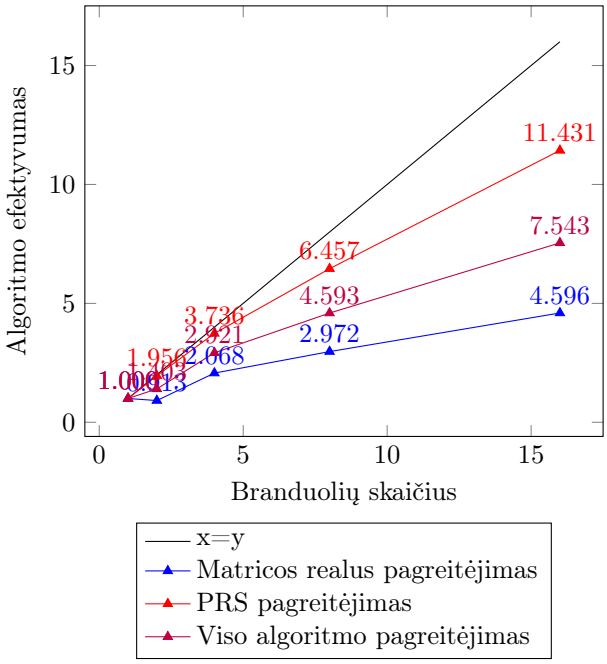

Diagrama 6: Efektyvumo diagrama, kai PRS ir matricos skaičiavimas lygiagretinami iki 16 branduolių (leista ant MIF superkompiuterio)