KAUNO TECHNOLOGIJOS UNIVERSITETAS

Lygiagretusis programavimas
(P170B328)

Inžinerinis projektas
Funkcijų optimizavimas

Atliko:

IFF-8/12 gr. studentas

Jokūbas Akramas

2020 m. lapkričio 27 d.

Priėmė:

lekt. Dominykas Barisas

lekt. Mindaugas Jančiukas

KAUNAS 2020

Turinys

1. Užduotis

Pagal dėstytojo rekomendacijas užduotis pasirinkta iš skaitinių metodų ir algoritmų modulio (T170B115) antrojo namų darbo optimizavimo užduoties.

Užduotis
Pagal pateiktą uždavinio sąlygą sudarykite tikslo funkciją ir išspręskite ją vienu iš gradientinių metodų (gradientiniu, greičiausio nusileidimo, kvazi-gradientiniu, ar pan.). Gautą taškų konfigūraciją pavaizduokite programoje, skirtingais ženklais pavaizduokite duotus ir pridėtus (jei sąlygoje tokių yra) taškus.

Sąlyga
Plokštumoje ($-10 \leq \boldsymbol{x} \leq 10$, $-10 \leq \boldsymbol{y} \leq 10$) išsidėstę n taškų ($3 \leq \boldsymbol{n}$), vienas jų fiksuotas koordinačių pradžioje (0; 0). Kiekvienas taškas su visais kitais yra sujungtas tiesiomis linijomis (stygomis). Raskite tokias taškų koordinates, kad atstumas tarp taškų būtų kuo artimesnis vidutiniam atstumui, o stygų ilgių suma kuo geriau atitiktų nurodytą reikšmę S ($10 \leq \boldsymbol{S}$).

2. Užduoties analizė ir sprendimo metodas

Norint išspręsti šį uždavinį reikės sudaryti tikslo funkciją. Šiuo atveju tikslo funkcija yra:

$$\Psi(\mathrm{x},\mathrm{y}) = \sum_{i=1}^{n}\sum_{u=i+1}^{n}\left(\sqrt{(x_u - x_i)^2 + (y_u - y_i)^2} - vid\right)^2 + \sum_{i=1}^{n}\sum_{u=i+1}^{n}\sqrt{(x_u - x_i)^2 + (y_u - y_i)^2} - s$$

Pirmasis funkcijos dėmuo yra kiekvieno taško atstumo su kiekvienu tašku atimtis iš vidutinio atstumo tarp taškų, pakelta kvadratu. T.y. kuo atstumai tarp taškų labiau atitiks *‚vid‘* kintamąjį (vidurkį), tuo tikslo funkcija bus mažesnė. Kėlimas kvadratu neatitikima vidurkiui didina eksponentiškai, kas tikslo funkciją daro didesnę, o tai papildomai atitolina funkciją nuo norimo tikslumo. Tokiu būdu pasiekiamas tikslesnis vidurkis.
Antrasis funkcijos dėmuo yra visų stygų suma, o argumentas *‚s‘* atimamas dėl tokios pat priežasties, kaip ir *‚vid‘* argumentas pirmame dėmenyje – tam, kad stygų sumai, labiausiai atitinkant argumentą *‚s‘* tikslo funkcija būtų kaip galima mažesnė.

Iš tikslo funkcijos išvestinės ir normalės santykio išreiškiamas gradientas:

$$\vec{g} = \overrightarrow{\nabla f} \Big/ \left\| \overrightarrow{\nabla f} \right\|$$

Einant gradiento kryptimi funkcija didėja, o kadangi užduotis reikalauja funkciją minimizuoti, bus einama priešinga gradiento kryptimi:

$$x^{i+1} = x^i - \Delta s \cdot [g]^T$$

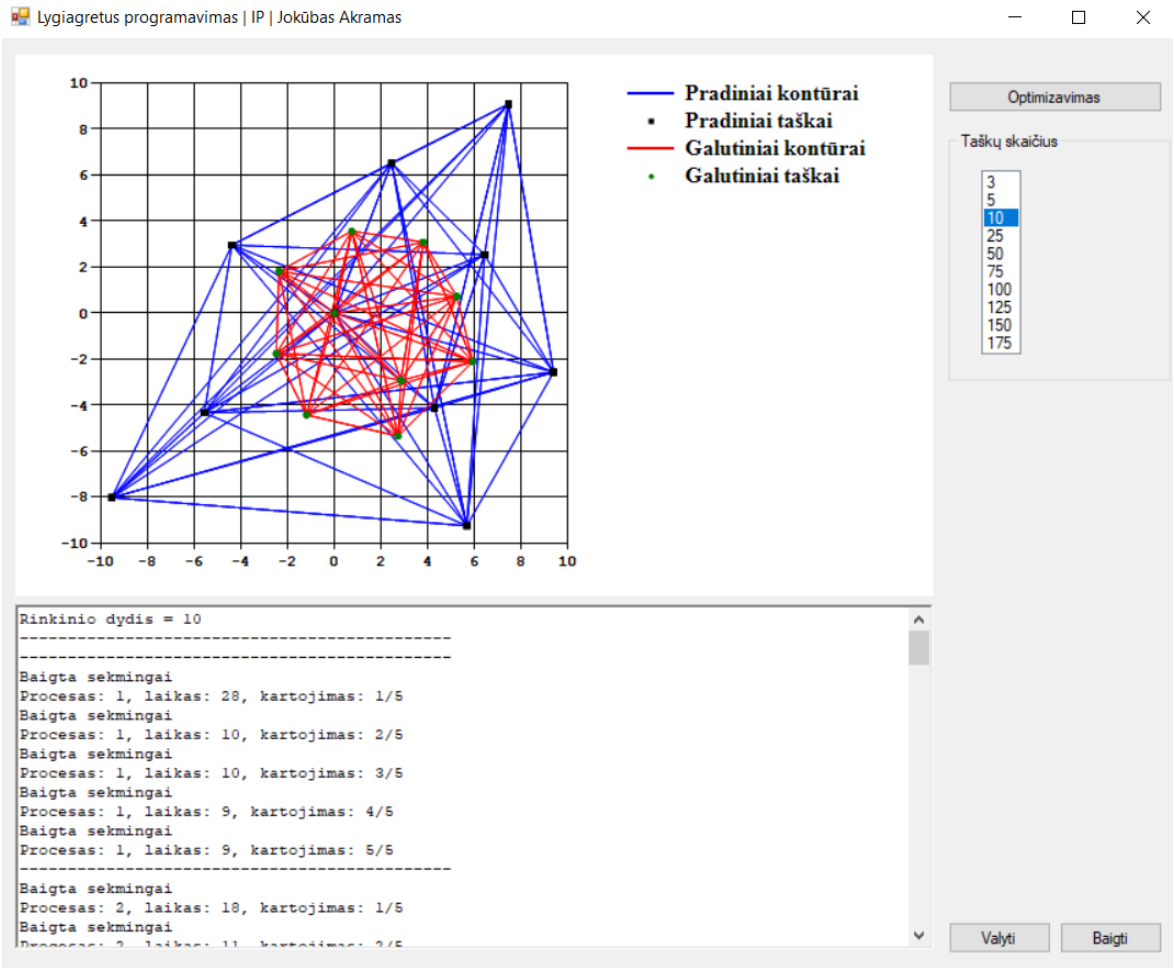
Sekantis argumentas bus gautas esamą argumentą sumažinus gradiento reikšme dauginant iš žingsnio, nurodyto argumentu *‚s‘*. Kol funkcija sėkmingai mažėja, argumentą galima didinti, kad darbas vyktų greičiau. Jei funkcija padidėtų, reiktų grįžti į buvusią padėtį ir mažinti žingsnį tol, kol funkcija toliau mažės ir bus pasiektas norimas tikslumas. Pasiekus tikslumą programa baigia darbą ir nubraižo gautus taškus.

3. Programos aprašymas

3.1. Vartotojo sąsaja

Programa realizuota naudojant rekomenduojamas priemones – C# parallel LINQ.

Paleidus programą yra pateikiamas formos langas (pav. 1), kuriame vartotojas pasirenka duomenų rinkinio dydį (taškų skaičių) ir programa pradeda vykdyti darbą.



pav. 1 Formos langas

Atlikus darbą nubraižomi pradiniai taškai (juoda spalva), juos jungiančios stygos (mėlyna spalva), apskaičiuoti taškai (žalia spalva) ir juos jungiančios stygos (raudona spalva).
Apatinėje formos dalyje esantis tekstinis langas pateikia greitaveikos tyrimo laikus (plačiau 6 skyrius).

3.2. Metodų, klasių, duomenų struktūrų aprašai

Vartotojas, iniciavęs programos darbo pradžią, nukreipia programos vykdymą per *Button3_Click()* metodą.
Šiame metode nurodomi pagrindiniai argumentai – maksimalus gijų skaičius, kartojimų kiekis vidutiniam greitaveikos rezultatui nustatyti, pasirinkto duomenų rinkinio numeris, taip pat grafinio piešimo argumentai. Iš šio metodo yra kviečiamas *GreitaveikosTyrimas()* metodas.
Šiame metode vykdomi duomenų rinkinio rezultatų skaičiavimai skirtingu kiekiu gijų. Matuojami laikai, kiek gijos užtrunka vykdydamos darbą. Laikų rezultatai pateikiami formoje, tekstinėje dalyje. Šiame metode sukuriamas taškų klasės *Dots* objektas. Ši klasė savyje turi užduoties argumentą S taip pat abscisių ir ordinačių vektorius x y. Iš *GreitaveikosTyrimas()* metodo yra kviečiamas *Optimizacija()* metodas.

Šiame metode vykdomas pagrindinis skaičiavimo darbas. Pagrindą sudaro iteracinis ciklas, kurio kiekvienoje iteracijoje yra kviečiami metodai:

- *Ilgis()* – nustatoma stygų ilgių suma;
- ***Gradientas()*** – skaičiuojamas gradientas;
- *Tikslo()* – skaičiuojama tikslo funkcija;
- *Gradiento_norma()* – gradientas pervedamas į normalės vektorių;

Paryškintas metodas *Gradientas()* yra metodas, kuris buvo išlygiagretintas. Taip buvo pasielgta todėl, nes buvo išmatuoti vykdymo laikai ir šis metodas užtruko vidutiniškai 99% viso metodo *Optimizacija()* vykdymo laiko. Kartu bandant lygiagretinti ir kitus metodus buvo pasiektas blogesnis rezultatas laiko atžvilgiu, nei juos vykdant nuosekliai.
Gradientas() metode yra skaičiuojamos tikslo funkcijos išvestinės abscisių ir ordinačių ašims su skirtingais žingsniais. Šios ašys vėliau apjungiamos į 2D masyvą ir grąžinamos kaip gradientas. Užkomentuotose eilutėse pateiktas ankstesnis sprendimo būdas, kai nebuvo naudojamas lygiagretus programavimas.
Visoje programoje plačiai naudojami 1D ir 2D masyvai siekiant didesnės greitaveikos.

4. Programos pagrindinių dalių tekstai su komentarais

```
1.  /// <summary>
2.  /// Programos entry point nuo vartotojo lango
3.  /// </summary>
4.  /// <param name="rinkinys">Pasirinkto rinkinio numeris</param>
5.  /// <param name="maxThreads">Maksimalių gijų skaičius tyrime</param>
6.  /// <param name="kartojimaiVidurkiui">Kartojimų skaičius matavimų vidurkiui išvesti</param>
7.  private void GreitaveikosTyrimas(int rinkinys, int maxThreads, int kartojimaiVidurkiui)
8.  {
9.      //Sukuriamas taškų objektas pagal vartotojo pasirinktą rinkinio dydį
10.     Dots dots = new Dots(rinkinys);
11.     double[] x = dots.x;
12.     double[] y = dots.y;
13.     double s = dots.S;
14.     //---
15.     //Brėžiama pradinė taškų seka
16.     PrintPoints(x, y, z1, p1);
17.     //---
18.     Stopwatch stopWatch = new Stopwatch();
19.     string line45 = new string('-', 45);
20.     string line92 = new string('-', 92);
21.     //---
22.     double[] xnew = new double[x.Length];
23.     double[] ynew = new double[y.Length];
24.     double[] benchmarkData = new double[maxThreads];
25.     //---
26.     richTextBox1.AppendText(string.Format("Rinkinio dydis = {0}\n", dots.x.Length));
27.     richTextBox1.AppendText(line45 + "\n");
28.     //Sukamas ciklas kiekvienam kiekiui procesų
29.     for (int i = 1; i <= maxThreads; i++)
30.     {
31.         richTextBox1.AppendText(line45 + "\n");
32.         ThreadCount = i;
33.         double suma = 0;
34.         //Kiekvienam procesui skaičiavimai kartojami kelis kartus, pagal šį ciklą
35.         for (int u = 0; u < kartojimaiVidurkiui; u++)
36.         {
37.             //---
38.             Array.Copy(x, xnew, x.Length);
39.             Array.Copy(y, ynew, y.Length);
40.             //---
41.             //Kviečiamas uždavinio sprendimo metodas ir matuojamas atlikimo laikas
42.             stopWatch.Start();
43.             Optimizacija(xnew, ynew, s);
44.             stopWatch.Stop();
45.             //---
46.             //Laiko reikšmė atspausdinama į ekraną ir išsaugojama vidurkiui išvesti
47.             double ms = stopWatch.ElapsedMilliseconds;
48.             suma += ms;
49.             richTextBox1.AppendText(string.Format("Procesas: {0}, laikas: {1}, kartojimas: {2}/{3}\n", i, ms,
    u + 1, kartojimaiVidurkiui));
50.             stopWatch.Reset();
51.         }
52.         benchmarkData[i - 1] = suma / kartojimaiVidurkiui;
53.     }
54.     richTextBox1.AppendText(line92 + "\n");
55.     richTextBox1.AppendText(line92 + "\n");
56.     //---
57.     //Išvedama duomenų rinkinio greitaveikos informacija pagal panaudotų gijų skaičių
58.     for (int i = 0; i < benchmarkData.Length; i++)
59.     {
60.         richTextBox1.AppendText(string.Format("Procesų skaičius = {0}, Vidutinis laikas = {1}\n", i + 1, bench
    markData[i]));
61.     }
62.     //---
63.     richTextBox1.AppendText(line92 + "\n");
64.     richTextBox1.AppendText(line92 + "\n");
65. }
```

```
/// <summary>
/// Pagrindinis uždavinio sprendimo metodas
/// </summary>
/// <param name="x">Abscisių reikšmių vektorius</param>
/// <param name="y">Oordinačių reikšmių vektorius</param>
/// <param name="s">Užduoties argumentas S</param>
private void Optimizacija(double[] x, double[] y, double s)
{
    //Pradiniai duomenys skaičiavimams (tikslumas, iteracijų sk., zingsnis, ir t.t.)
    double eps = 1e-6;
    int maxIter = 500;
    double zingsnis = 0.1;
    double tikslumas;
    int iteracija = 0;

    //Iteracinis ciklas tolimesnėms reikšmėms skaičiuoti
    for (; iteracija < maxIter; iteracija++)
    {
        int n = x.Length;
        //Vidutinio atstumo tarp taškų radimas
        double vid = Ilgis(x, y, 0, true) / (n * (n - 1) * 1 / 2);
        //Išlygiagretintas metodas, užimantis 99% laiko resursų
        double[,] grad = Gradientas(x, y, vid, s);
        //Tikslo funkcijos apskaičiavimas esamame taške
        double f0 = Tikslo(x, y, vid, s);
        //Gradiento normalės radimas
        double[,] deltaX = Gradiento_norma(grad, zingsnis);
        //Ėjimas prieš gradiento kryptį, t.y tikslo funkcijos mažėjimo kryptimi
        for (int u = 1; u < n; u++)
        {
            x[u] -= deltaX[u, 0];
            y[u] -= deltaX[u, 1];
        }
        //Tikslo funkcijos apskaičiavimas sekančiame taške
        double f1 = Tikslo(x, y, vid, s);
        //Jei tikslo funkcija padidėjo, grįžtama į buvusį tašką mažinamas žingsnis
        if (f1 > f0)
        {
            for (int u = 1; u < n; u++)
            {
                x[u] += deltaX[u, 0];
                y[u] += deltaX[u, 1];
            }
            zingsnis /= 2;
        }
        //Jei tikslo funkcija sumažėjo (to siekiame), žingsnis padvigubinamas
        else
        {
            zingsnis *= 2;
        }
        //Apskaičiuojamas tikslumas
        tikslumas = Math.Abs(f0 - f1) / (Math.Abs(f0) + Math.Abs(f1));
        //Jei tikslumas atitinka nurodytą, ciklas užbaigiamas
        if (tikslumas < eps)
        {
            richTextBox1.AppendText("Baigta sekmingai\n");
            break;
        }
        else if (iteracija == maxIter - 1)
        {
            richTextBox1.AppendText("Baigta nesekmingai\n");
        }
    }
    //Brėžiama gauta taškų seka
    PrintPoints(x, y, z2, p2);
}
```

```
1.  /// <summary>
2.  /// Išlygiagretintas metodas, užimantis 99% laiko resursų
3.  /// </summary>
4.  /// <param name="x">Abscisių reikšmių vektorius</param>
5.  /// <param name="y">Oordinačių reikšmių vektorius</param>
6.  /// <param name="vid">Vidutinis atstumas tarp taškų</param>
7.  /// <param name="s">Užduoties argumentas S</param>
8.  /// <returns></returns>
9.  private double[,] Gradientas(double[] x, double[] y, double vid, double s)
10. {
11.     //Pradiniai duomenys skaičiavimams
12.     int n = x.Length;
13.     double zingsnis = 0.0001;
14.     double[,] grad = new double[n, 2];
15.     //Tikslo funkcija esamame taške
16.     double f0 = Tikslo(x, y, vid, s);
17.     //---
18.     var numbers = Enumerable.Range(0, n);
19.     //---
20.     //Lygiagretinti metodai
21.     //Gradiento skaičiavimas abscisių ašies atžvilgiu
22.     double[] gradX = (from i in numbers.AsParallel().AsOrdered().WithMergeOptions(ParallelMergeOptions.N
    otBuffered).WithDegreeOfParallelism(ThreadCount)
23.                       select ((Tikslo(F1(x, i, zingsnis), y, vid, s) - f0) / zingsnis)).ToArray();
24.     //Gradiento skaičiavimas oordinačių ašies atžvilgiu
25.     double[] gradY = (from i in numbers.AsParallel().AsOrdered().WithMergeOptions(ParallelMergeOptions.N
    otBuffered).WithDegreeOfParallelism(ThreadCount)
26.                       select ((Tikslo(x, F1(y, i, zingsnis), vid, s) - f0) / zingsnis)).ToArray();
27.     //---
28.     //Abscisės ir oordinatės sujungiamos į matricą
29.     for (int i = 0; i < n; i++)
30.     {
31.         grad[i, 0] = gradX[i];
32.         grad[i, 1] = gradY[i];
33.     }
34.
35.     //Originaliai taikytas sprendimo būdas prieš pradedant taikyti lygiagretų sprendimą (eil. 37-41)
36.     /*
37.     for (int i = 0; i < n; i++)
38.     {
39.         grad[i, 0] = (Tikslo(F1(x, i, zingsnis), y, vid, s) - f0) / zingsnis;
40.         grad[i, 1] = (Tikslo(x, F1(y, i, zingsnis), vid, s) - f0) / zingsnis;
41.     }
42.     */
43.
44.     return grad;
45. }
```

5. Testavimas ir programos instaliavimo bei vykdymo instrukcija

5.1. Testavimas

Vizualusis
Kadangi užduotis tipas – optimizavimas – savaime neturintis tikslių atsakymų, galime pasikliauti tik vizualiais rezultatais. Užduotis nurodo, kad plokštumos rėžiai yra [-10; 10] abejomis ašimis, todėl jose ganėtinai sunku pavaizduoti didelės apimties duomenų rinkinius. Vizualiniu testavimu galime pasikliauti nagrinėdami 3, 5, 10 taškų rinkinius, didesnio taškų kiekio rinkiniai tampa sunkiai įžiūrimi, tačiau puikiai tinka analizuoti procesų greitaveiką, todėl ir buvo pasirinkti.

Pagal tikslumą
Pagrindinis užduoties sprendimo metodas *Optimizacija()* 54eil. (ciklo pabaigoje) tikrina, ar pasiektas ankščiau nurodytas tikslumas. Tai nėra kardinaliai tikslus atsakymas, bet vis tiek labai efektyvus, atsižvelgiant į tai, kad tikslumą galima nurodyti itin mažą. Pasiekus tikslumą ciklas baigs darbą, į formos tekstinę dalį bus išvedamas pranešimas „`Baigta sekmingai`“.

5.2. Instaliavimas

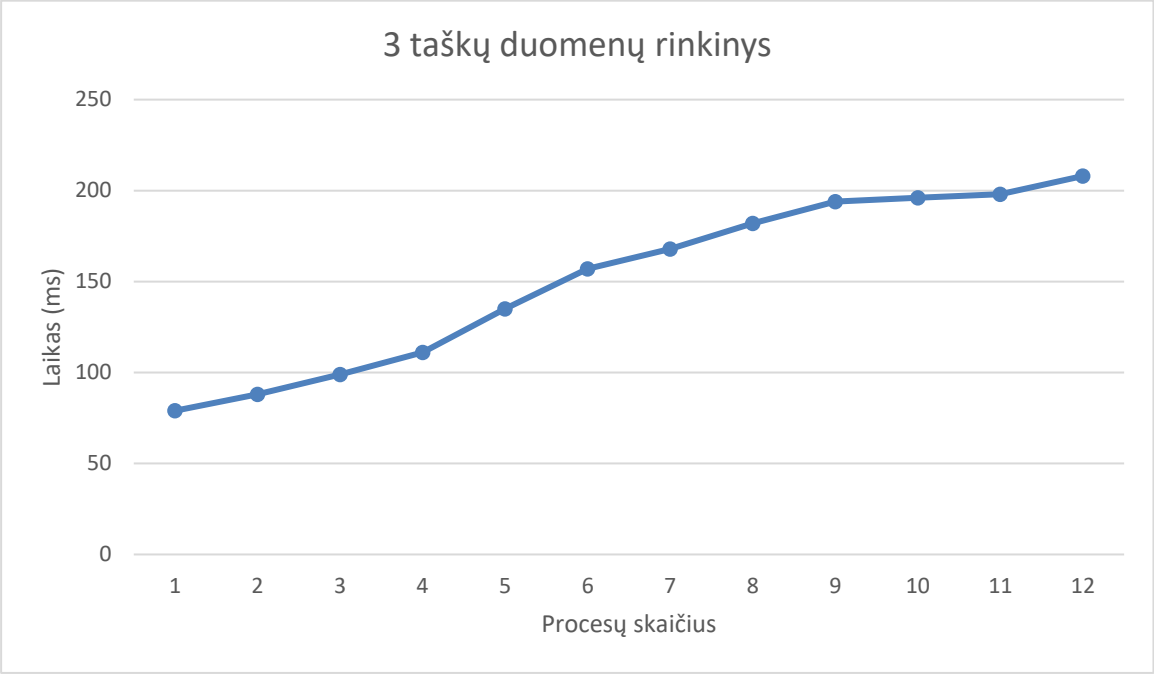
Parsisiuntus .zip paketą ir jį išsiarchyvavus, programą galima pasileisti vienu iš dviejų būdų:

- susikompiliavus kodą per Visual Studio programavimo aplinką;
- pasileidus Pvz1.exe vykdomąjį failą iš *IP/Lygciu sistemos/bin/Release* aplanko.

6. Vykdymo laiko kitimo tyrimas

6.1. Pirmas duomenų rinkinys (3 taškai)

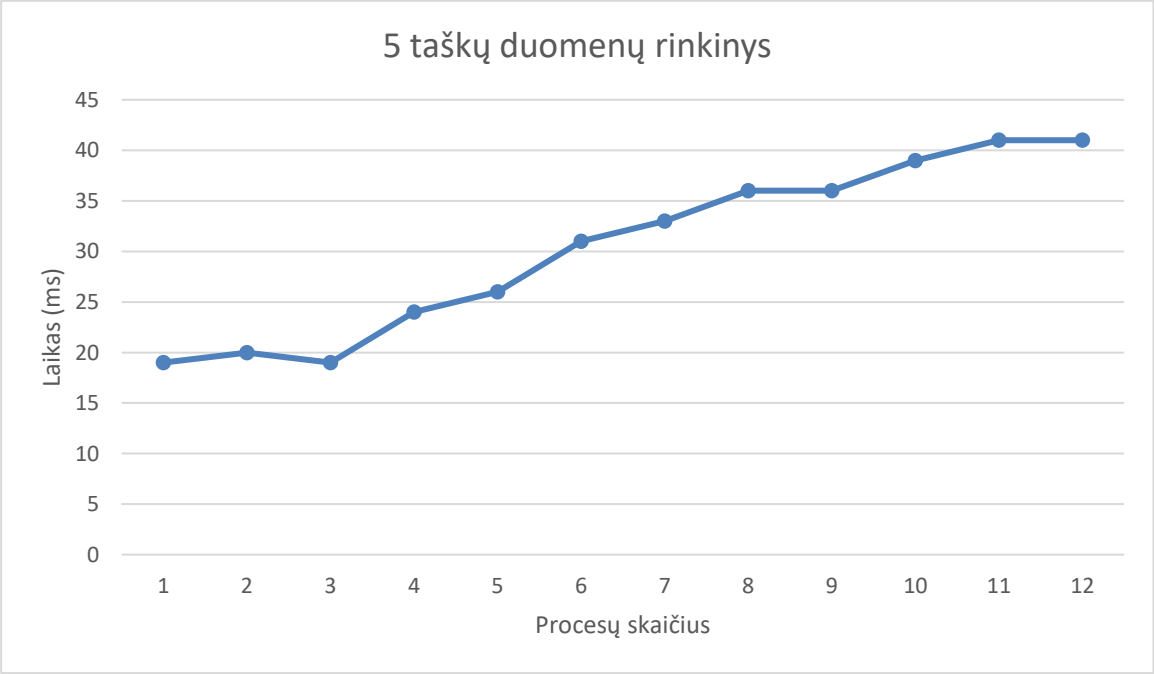
Šį rinkinį sudaro 3 atsitiktinai sudėlioti taškai.



Šiam duomenų rinkiniui lygiagretaus programavimo principas **nepasiteisina** – duomenų rinkinys per mažas, kad pasijaustų papildomų procesų jaučiama nauda. Didesnė laiko dalis praleidžiama procesų valdymui, nei pačiam skaičiavimui.

6.2. Antras duomenų rinkinys (5 taškai)

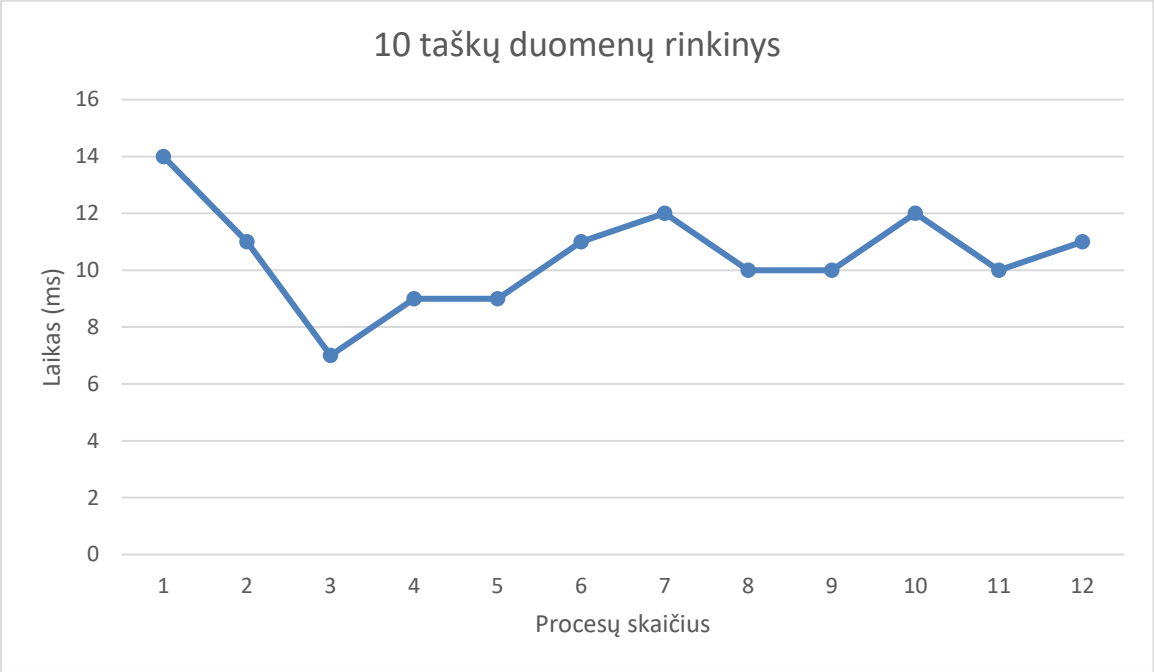
Šį rinkinį sudaro 5 atsitiktinai sudėlioti taškai.



Šiam duomenų rinkiniui lygiagretaus programavimo principas taip pat **nepasiteisina** – duomenų rinkinys per mažas, kad pasijaustų papildomų procesų jaučiama nauda. Didesnė laiko dalis praleidžiama procesų valdymui, nei pačiam skaičiavimui.

6.3. Trečias duomenų rinkinys (10 taškų)

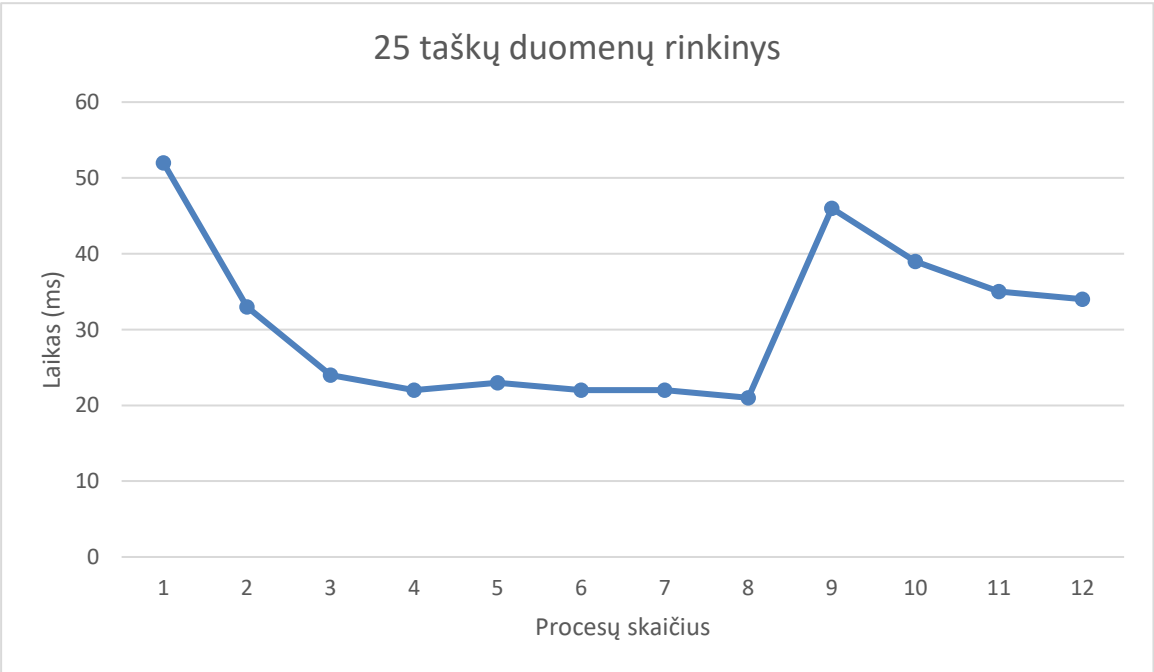
Šį rinkinį sudaro 10 atsitiktinai sudėliotų taškų.



Šiam duomenų rinkiniui lygiagretaus programavimo principas **pradeda pasiteisinti**. Didėjant procesų skaičiui šiek tiek mažėja vykdymo laikas.

6.4. Ketvirtas duomenų rinkinys (25 taškai)

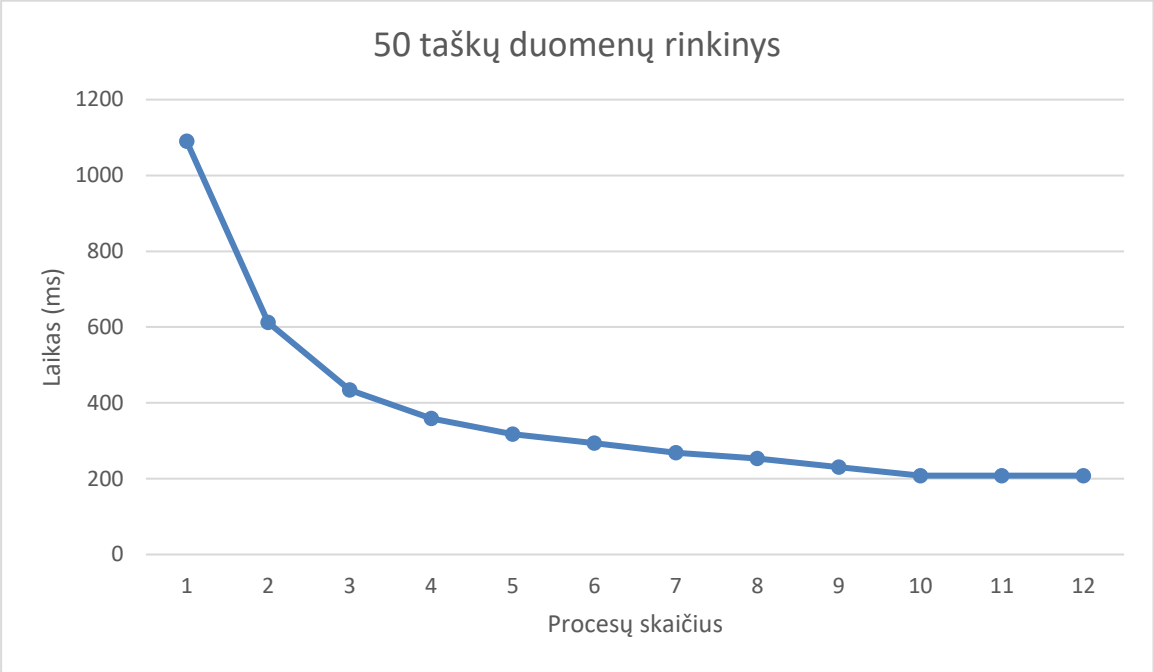
Šį rinkinį sudaro 25 atsitiktinai sudėlioti taškai.



Šiam duomenų rinkiniui lygiagretaus programavimo principas **pasiteisina**. Didėjant procesų skaičiui mažėja vykdymo laikas.

6.5. Penktas duomenų rinkinys (50 taškų)

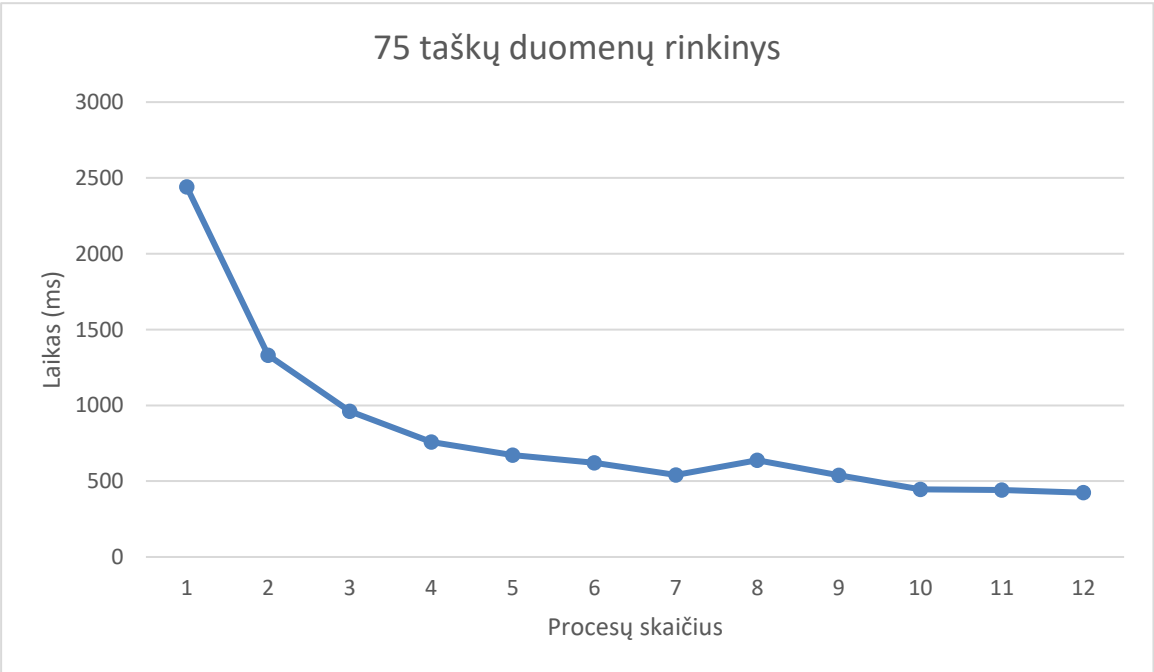
Šį rinkinį sudaro 50 atsitiktinai sudėliotų taškų.



Šiam duomenų rinkiniui lygiagretaus programavimo principas **stipriai pasiteisina**. Didėjant procesų skaičiui stipriai mažėja vykdymo laikas.

6.6. Šeštas duomenų rinkinys (75 taškai)

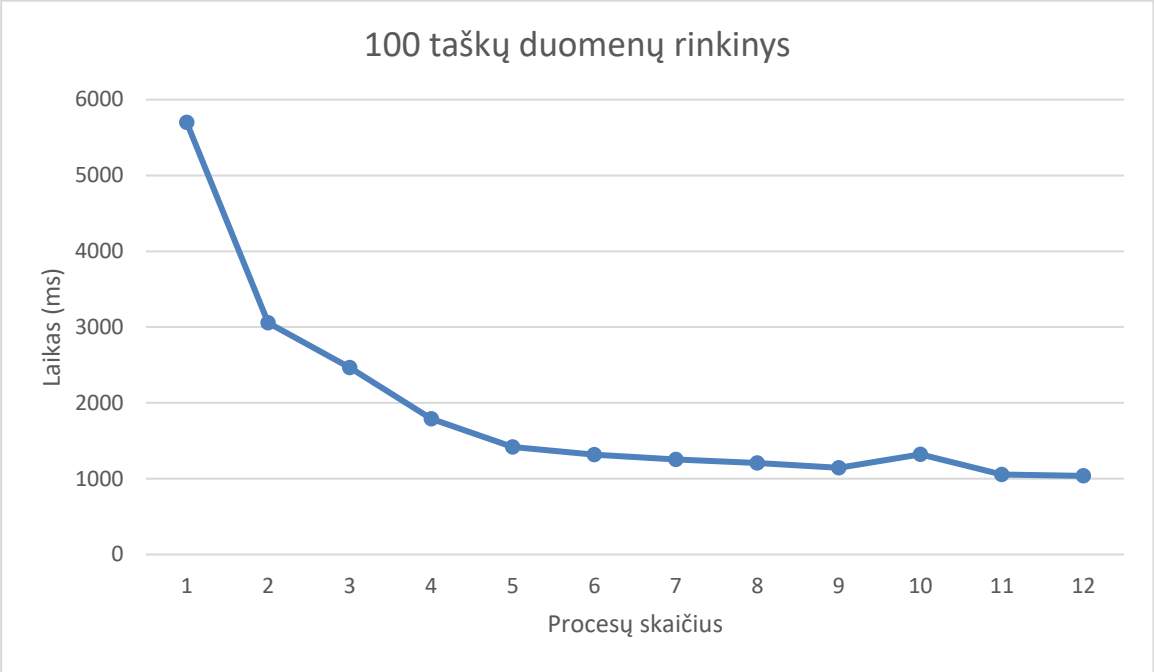
Šį rinkinį sudaro 75 atsitiktinai sudėlioti taškai.



Šiam duomenų rinkiniui lygiagretaus programavimo principas **stipriai pasiteisina**. Didėjant procesų skaičiui stipriai mažėja vykdymo laikas.

6.7. Septintas duomenų rinkinys (100 taškų)

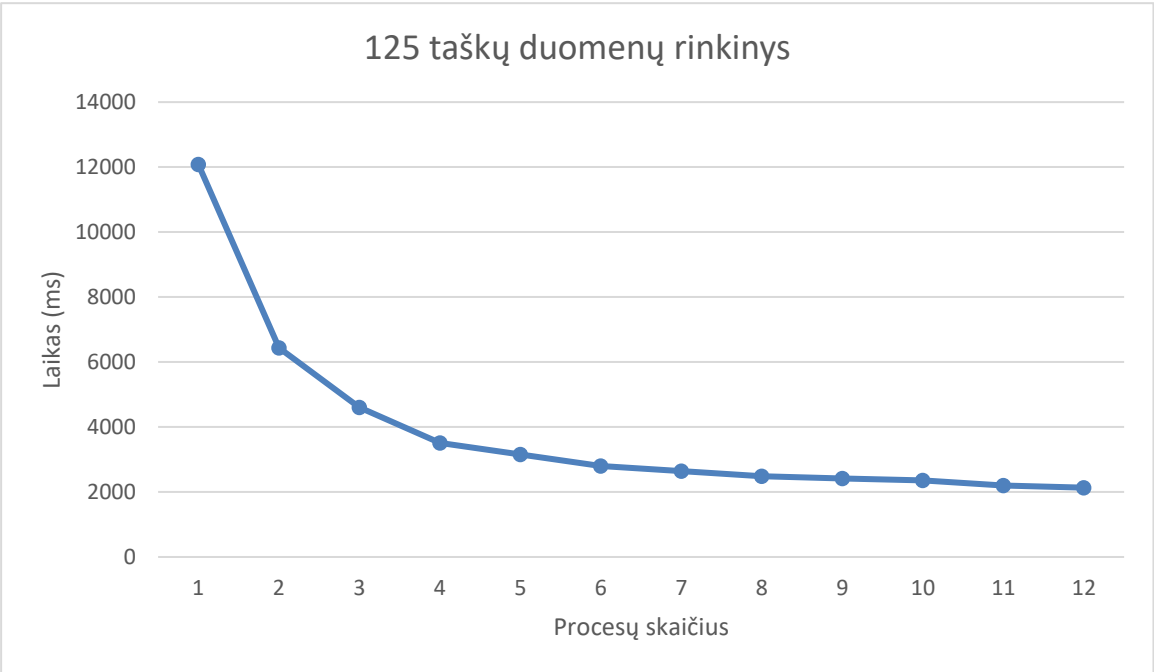
Šį rinkinį sudaro 100 atsitiktinai sudėliotų taškų.



Šiam duomenų rinkiniui lygiagretaus programavimo principas **stipriai pasiteisina**. Didėjant procesų skaičiui stipriai mažėja vykdymo laikas.

6.8. Aštuntas duomenų rinkinys (125 taškai)

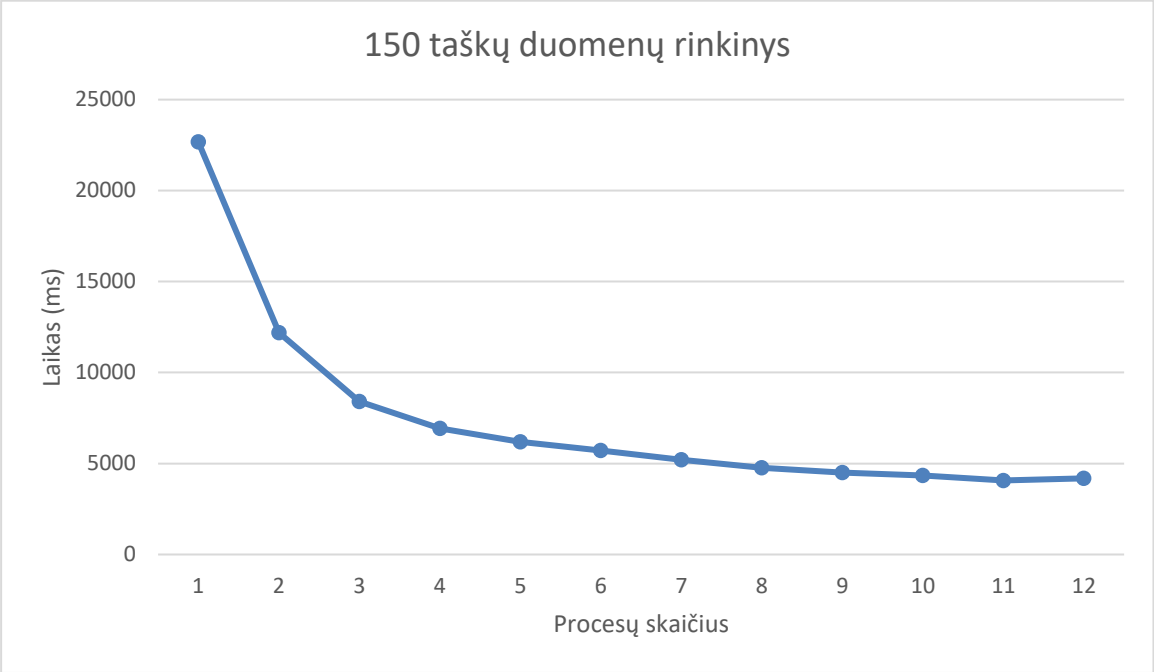
Šį rinkinį sudaro 125 atsitiktinai sudėlioti taškai.



Šiam duomenų rinkiniui lygiagretaus programavimo principas **stipriai pasiteisina**. Didėjant procesų skaičiui stipriai mažėja vykdymo laikas.

6.9. Devintas duomenų rinkinys (150 taškų)

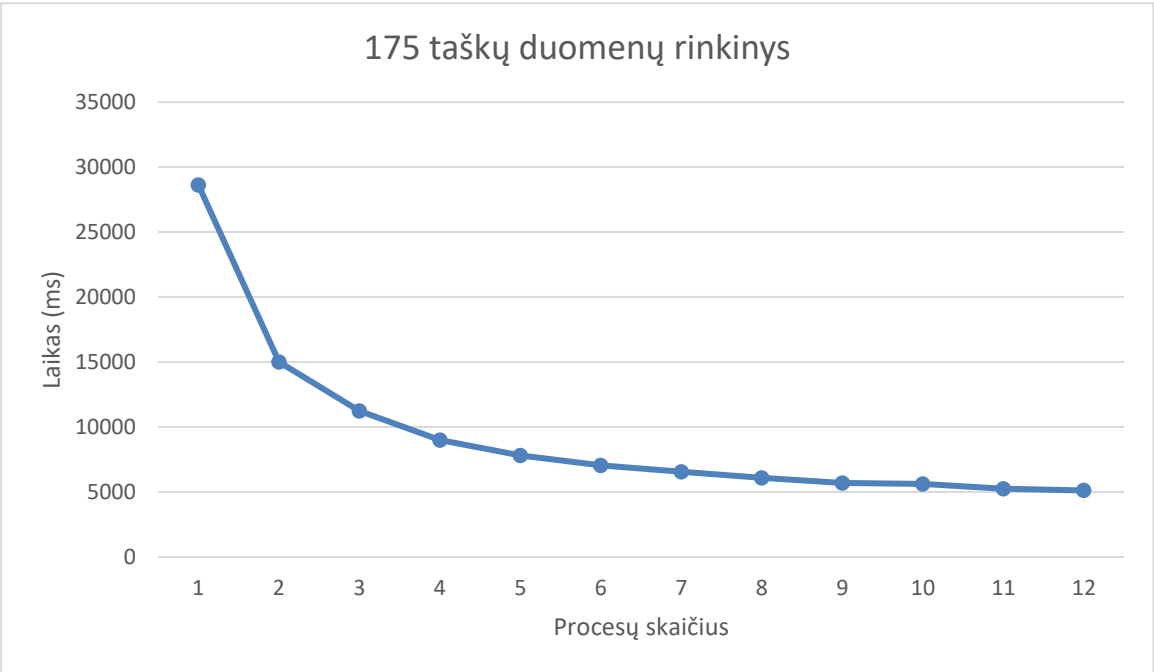
Šį rinkinį sudaro 150 atsitiktinai sudėliotų taškų.



Šiam duomenų rinkiniui lygiagretaus programavimo principas **stipriai pasiteisina**. Didėjant procesų skaičiui stipriai mažėja vykdymo laikas.

6.10. Dešimtas duomenų rinkinys (175 taškai)

Šį rinkinį sudaro 175 atsitiktinai sudėlioti taškai.



Šiam duomenų rinkiniui lygiagretaus programavimo principas **stipriai pasiteisina**. Didėjant procesų skaičiui stipriai mažėja vykdymo laikas.

7. Išvados ir literatūra

Pagrindinė išvada

Metodų išlygiagretinimas gali smarkiai paspartinti programos veikimą, bet gali ir sulėtinti. Viskas priklauso nuo duomenų apimties ir metodų sudėtingumo. Esant mažiems rinkiniams (3, 5 taškai) lygiagretus sprendimas užtruko ilgiau, nei nuoseklus; taip pat lygiagretinant paprastus metodus laiko charakteristikos tapo prastesnės. Tačiau su didesniais duomenų rinkiniais programos darbas kaip taisyklė pagreitėjo iki 6 kartų. Galiu daryti prielaidą, kad tai susiję su mano kompiuterio procesoriaus 6 fiziniais branduoliais. Tą patvirtina ir kitas mano pastebėtas dalykas – metode *Gradientas()*, 22 ir 25 eilutėse atributui `.WithDegreeOfParallelism(threadCount)` nurodant gijų skaičių `threadCount` programos darbas pastebimai greitėjo iki 12 gijų skaičiaus, su didesniu laikėsi toks pat, arba letėjo. Iš to peršasi išvada, kad mano kompiuterio 12 virtualių branduolių buvo pilnai išnaudoti programos vykdymo metu ir didesnis gijų skaičiaus nurodymas programiniame kode neturėjo jokios įtakos.

Literatūra

- **https://moodle.ktu.edu/pluginfile.php/351670/mod_resource/content/7/SMA_06_funkciju_optimizavimas.pdf**
- **https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/introduction-to-plinq**
- **http://www.planetb.ca/syntax-highlight-word**