

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

LYGIAGRETUS PROGRAMAVIMAS

Individualus projektas

Atliko:

IFF-7/2 gr. studentas

Vytenis Kunickas

Priėmė:

lekt. BARISAS Dominykas

doc. BLAŽAUSKAS Tomas

Data:

2019-11-27

KAUNAS
2019

Turinys

1.	Užduoties analizė ir sprendimo metodas.....	3
2.	Programos aprašymas.....	3
3.	Programos pagrindinių dalių tekstai su komentarais.....	4
4.	Testavimas ir programos įrašymo bei vykdymo instrukcija.	5
5.	Vykdymo laiko kitimo tyrimas.....	6
6.	Išvados.....	12
7.	Literatūra.....	12
8.	Paveikslėliai ir lentelės.	12

1. Užduoties analizė ir sprendimo metodas.

Pasirinkta užduotis – Skaitinių metodų ir algoritmų modulio optimizavimo uždavinio – gradiento skaičiavimo išlygiagretinimas. Darbas atliktas c# kalba, naudojant modulio Thread ir Monitor klases.

Uždavinio sąlyga: Duotos n ($3 \leq n$) taškų fiksuotos koordinatės ($-10 \leq x \leq 10$, $-10 \leq y \leq 10$). (Koordinatės gali būti generuojamos atsitiktinai). Srityje ($-10 \leq x \leq 10$, $-10 \leq y \leq 10$) reikia padėti papildomų m ($3 \leq m$) taškų taip, kad jų atstumai nuo visų kitų taškų (įskaitant ir papildomus) būtų kuo artimesni vidutiniam atstumui, o bendra taškų kaina kuo mažesnė. Vieno taško kaina apskaičiuojama pagal funkciją $C(x, y) = x^2 + y^2 + 1,5$.

2. Programos aprašymas.

Funcijos:

- `static double[] Gradient(double[,] nodes, int from, int to)` – skaičiuoja gradientą
- `static double CostFunction(double[,] nodes)` – skaičiuoja visą dabartinę taškų kainą ir ilgių vidurkį
- `static void Uzduotis3()` – skaičiuoja, kur dėti taškus, kad jų kaina ir ilgių vidurkis būtų kuo mažesnis.
- `static void Main(string[] args)` – pagrindinė funkcija

Pagrindiniai kintamieji:

- `nodes` – atsitiktinai suskaičiuoti taškai, kurių gradientas bandomas surasti
- `gradients` – gradientai
- `threads` – gijų masyvas

3. Programos pagrindinių dalių tekstai su komentarais.

lentelė 1 Pagrindinė programos dalis

for (int j = 0; j < threads.Length; j++)	
{	
threads[j] = new Thread(() =>	
{	
int currentThread = int.Parse(Thread.CurrentThread.Name);	Paimamas gijos pavadinimas, kad būtų galima paskirstyti apdorojamus duomenis
int from = currentThread * nodeAmount / 4;	Nustatomi režiai, pagal juos bus nustatoma, kuriuos taškus gija apdoro
int to = from + nodeAmount / 4;	
int fromG = currentThread * nodeAmount / 4 * 2;	
int toG = fromG + nodeAmount / 4 * 2;	
int count = 0;	
double[] grads = Gradient(nodes.getAllNodes(), from, to);	Gaunamas gradientas tik tiems taškams, kurie yra apdorojami gijos
for (int z = fromG; z < toG; z++)	Gradientai yra sudedami į gradient monitoriu
{	
gradients.addGrad(z, grads[count]);	
count++;	
}	
count = 0;	
while (gradients.getCount() != gradients.getMax())	Gija laukia kol visos gijos sudės gradientus
{	
if (gradients.getCount() == gradients.getMax())	
{	
break;	
}	
}	
if (gradients.IsNormalized() == false)	Jei dar gradientai nėra normalizuoti, gija juos normalizuoja
{	
gradients.NormalizeGradientVector();	
}	
grads = gradients.getAllGrads();	Gradientai dėl patogumo yra paimami į lokalią masyvą
if (currentThread == 0)	Pirmai gijai, kuri apdoroja taškų aibės pradžią, nereikia keisti pirmų 4 taškų
{	
from = 4;	
}	
for (int z = from; z < to; z++)	Taškų reikšmės yra pakeičiamos pagal gradiento reikšmes
{	
nodesCopy.changeNode(z, nodesCopy.getNode(z)[0] - (step * grads[count]), nodesCopy.getNode(z)[1] - (step * grads[count + 1]));	
count = count + 2;	
}	
});	

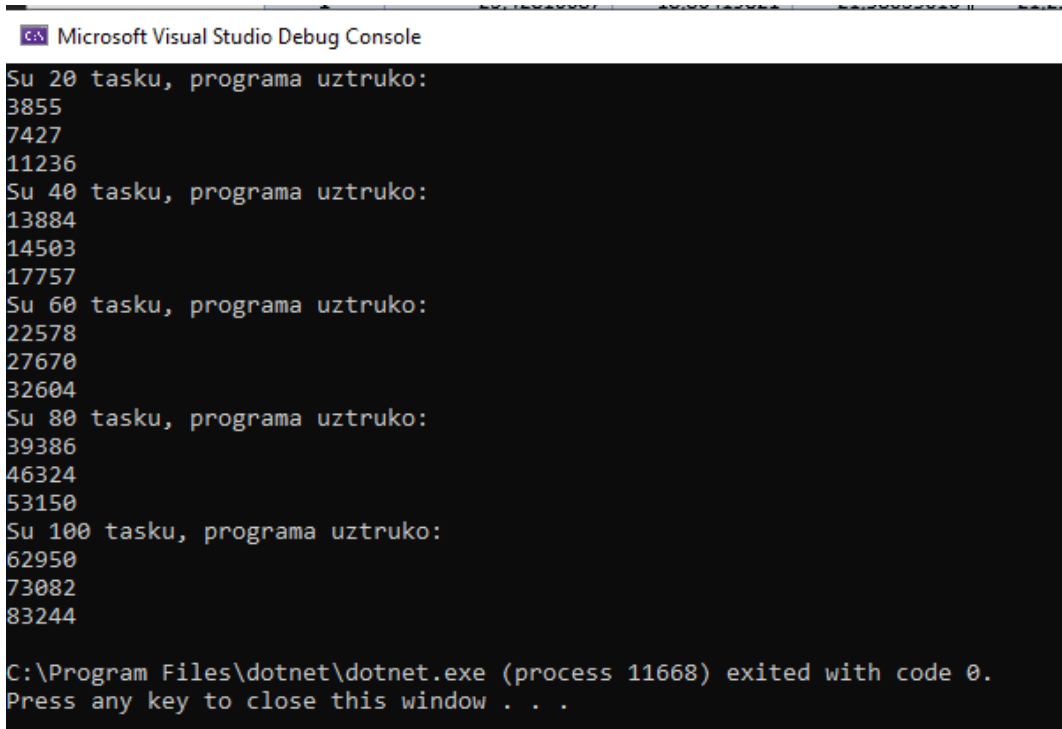
4. Testavimas ir programos įrašymo bei vykdymo instrukcija.

Pasiruošimas darbui:

- Įsidiegti reikalingą aplinką (Visual studio ar kita c# kompiliatoriu turinčią aplinką) ir bibliotekas, jei jų nėra
- Atsisiųsti projektą
- Pasileisti projektą

Projektą sudaro 2 failai:

- Program.cs - taškų skaičiavimo klasė
- Gradients.cs – gradientų talpinimo monitorius
- Nodes.cs – taškų talpinimo monitorius



```
Microsoft Visual Studio Debug Console
Su 20 tasku, programa uztruko:
3855
7427
11236
Su 40 tasku, programa uztruko:
13884
14503
17757
Su 60 tasku, programa uztruko:
22578
27670
32604
Su 80 tasku, programa uztruko:
39386
46324
53150
Su 100 tasku, programa uztruko:
62950
73082
83244

C:\Program Files\dotnet\dotnet.exe (process 11668) exited with code 0.
Press any key to close this window . . .
```

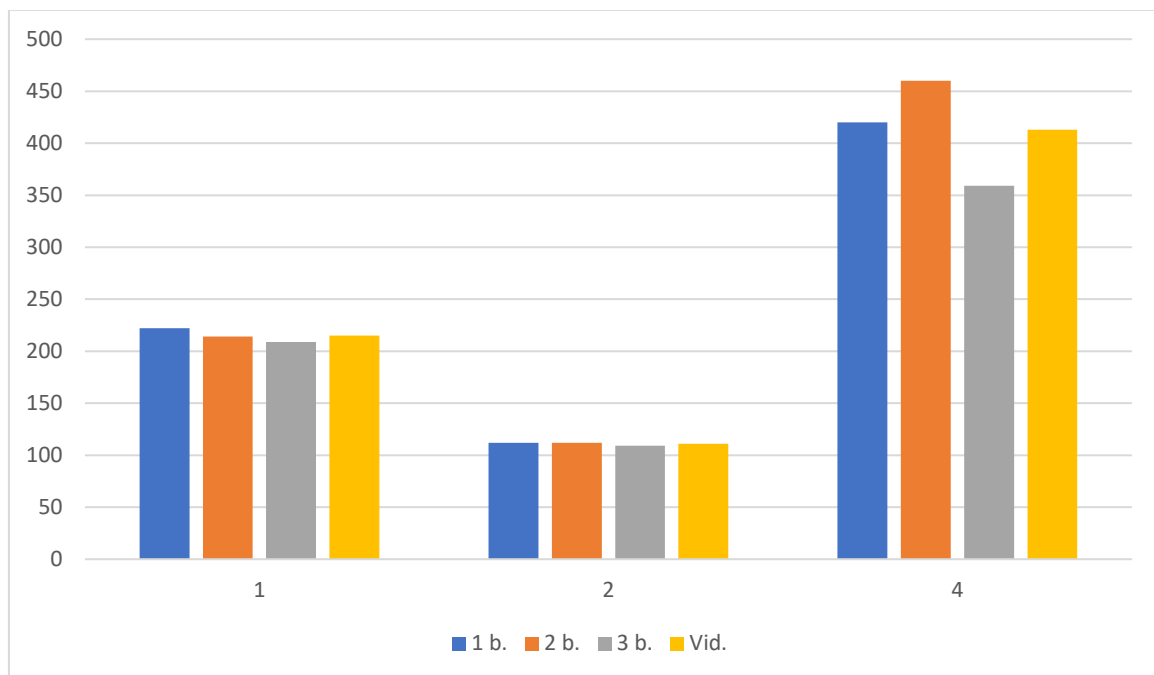
pav. 1 " Program.cs " laikų skaičiavimas

5. Vykdymo laiko kitimo tyrimas

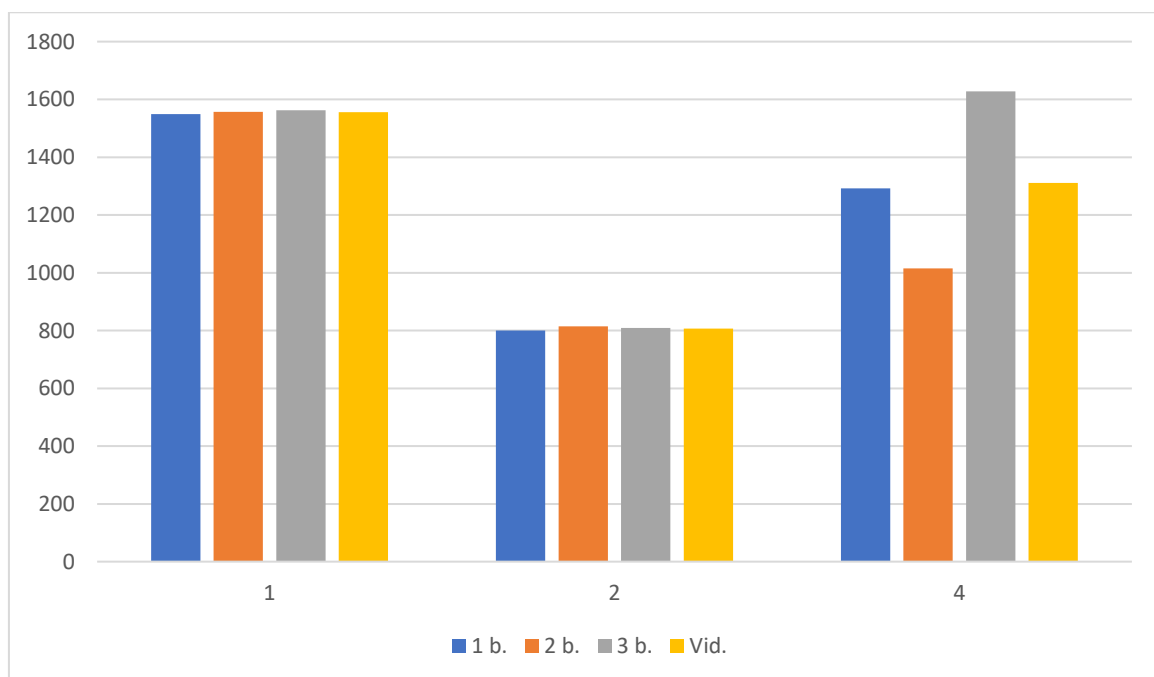
lentelė 2 Bandymų vykdymų laikai

Procesų skaičius	I bandymas	II bandymas	III bandymas	Vidurkis
	20 taškų			
1	222	214	209	215
2	112	112	109	111
4	420	460	359	413
	40 taškų			
1	1549	1557	1562	1556
2	800	814	809	807
4	1292	1015	1628	1311
	60 taškų			
1	5177	5095	5231	5167
2	2611	2602	2599	2604
4	2954	3378	3526	3286
	80 taškų			
1	12068	11838	11606	11837
2	6175	6187	5977	6113
4	5207	5328	5523	5352
	100 taškų			
1	22438	22565	22527	22510
2	11578	11504	11533	11538
4	8886	9117	8736	8913
	120 taškų			
1	38706	38798	38689	38731
2	20266	19744	19657	19889
4	14614	14525	14302	14480
	140 taškų			
1	61353	61313	61394	61353
2	32027	32756	31972	32251
4	22349	21738	21937	22008
	160 taškų			
1	91320	91618	91359	91432
2	46645	46607	46547	46599
4	31651	31422	31803	31625
	180 taškų			
1	124135	128103	125932	126056
2	65423	65441	66914	65926
4	44230	44270	44020	44173
	200 taškų			
1	170634	170643	171182	170819
2	90766	91856	92390	91670
4	61731	61688	60085	61168

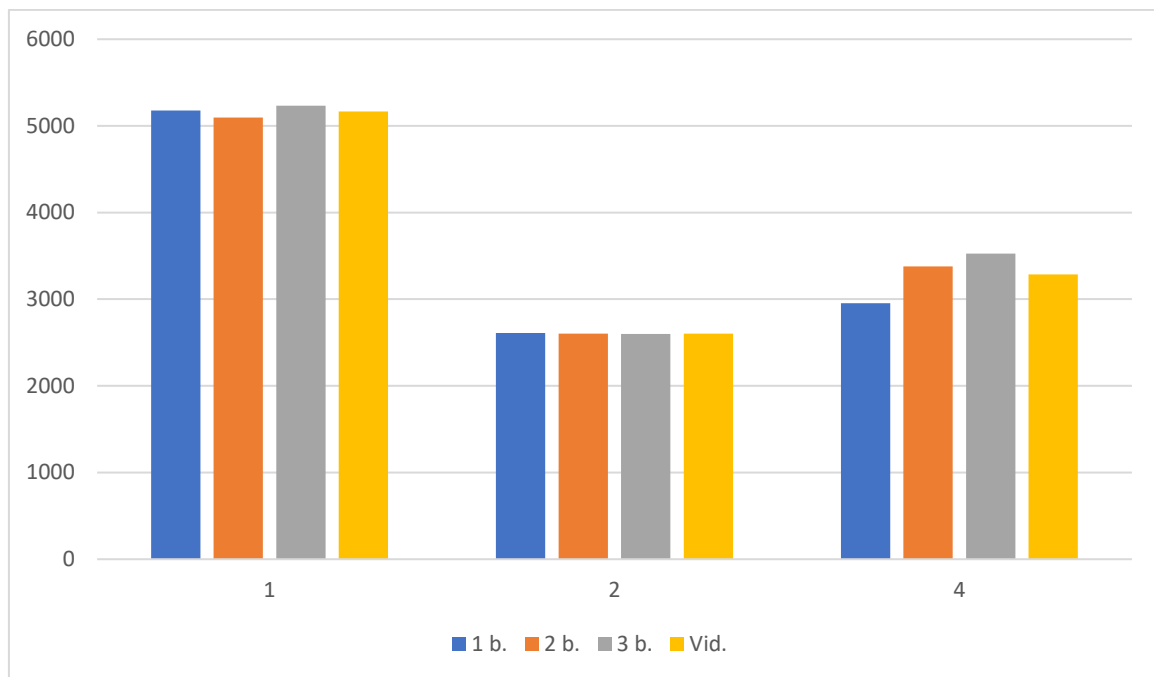
grafikas 3. Procesų darbo laikų palyginimas su 20 taškų



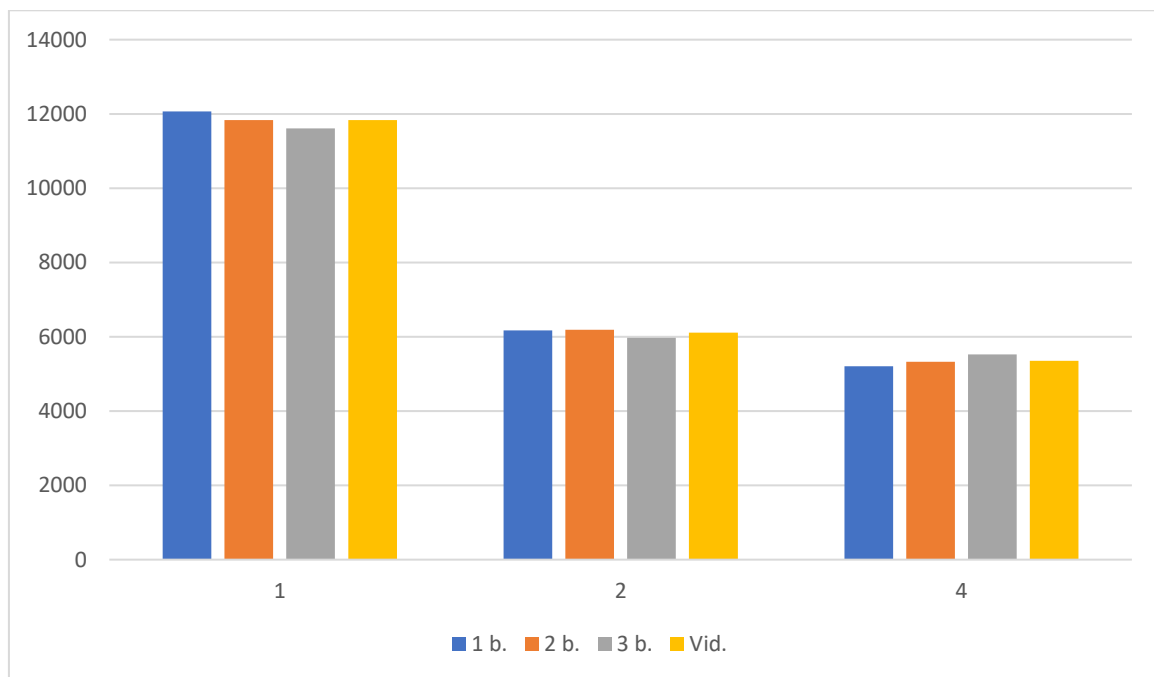
grafikas 4. Procesų darbo laikų palyginimas su 40 taškų



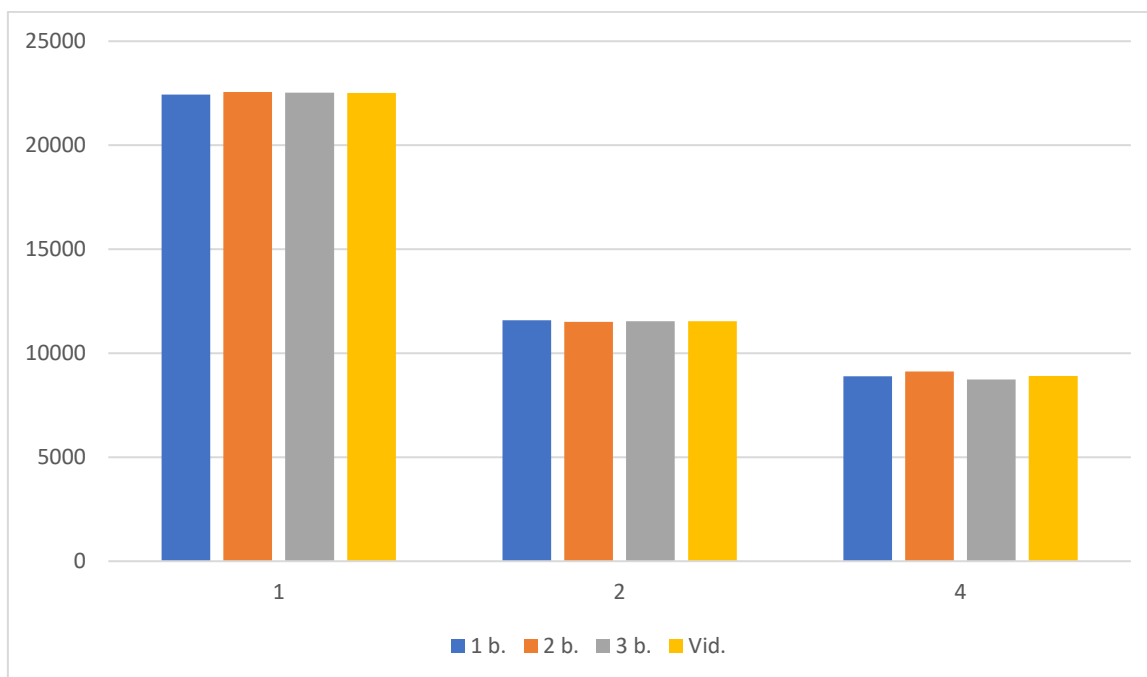
grafikas 5. Procesų darbo laikų palyginimas su 60 taškų



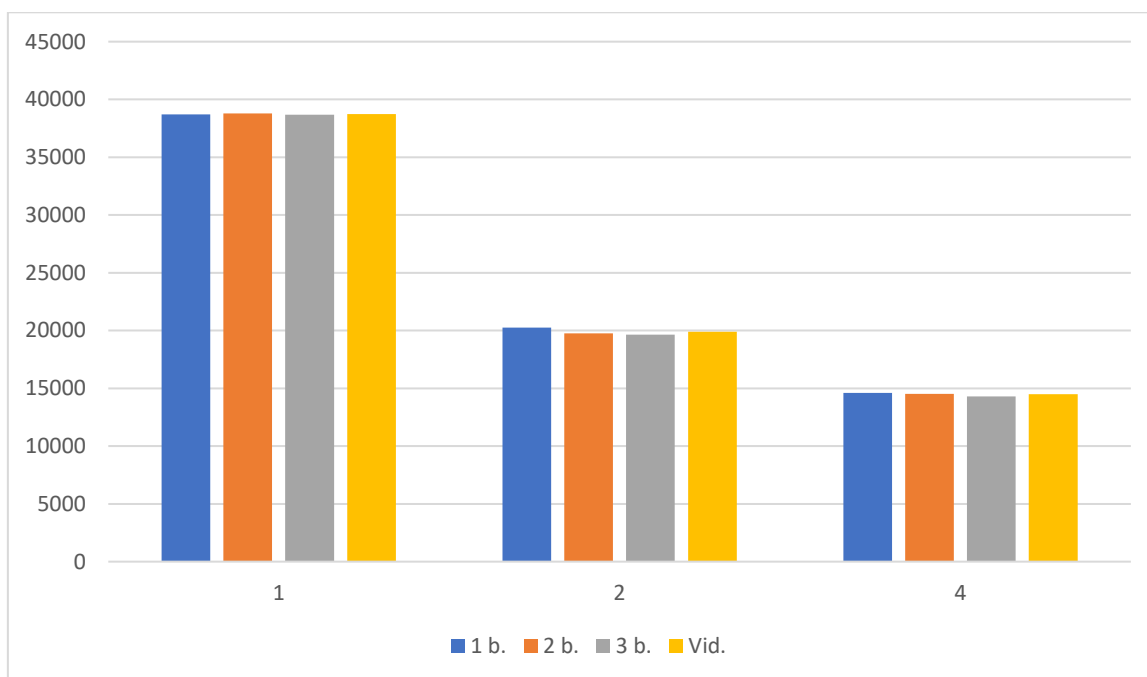
grafikas 6. Procesų darbo laikų palyginimas su 80 taškų



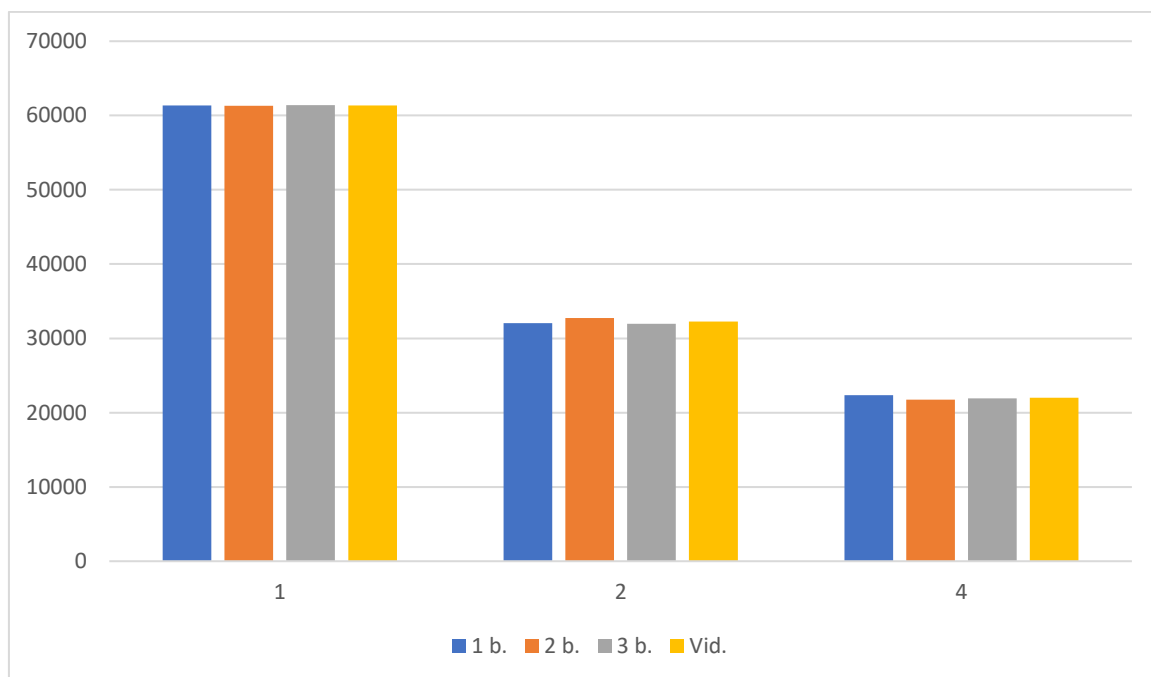
grafikas 7. Procesų darbo laikų palyginimas su 100 taškų



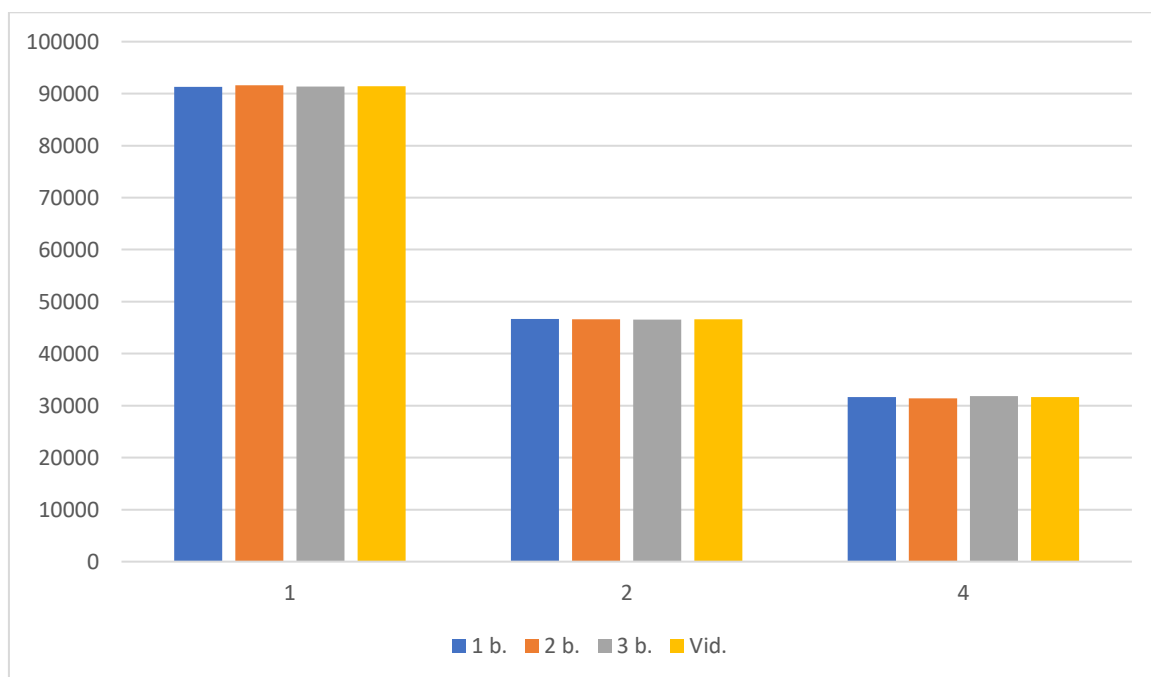
grafikas 8. Procesų darbo laikų palyginimas su 120 taškų



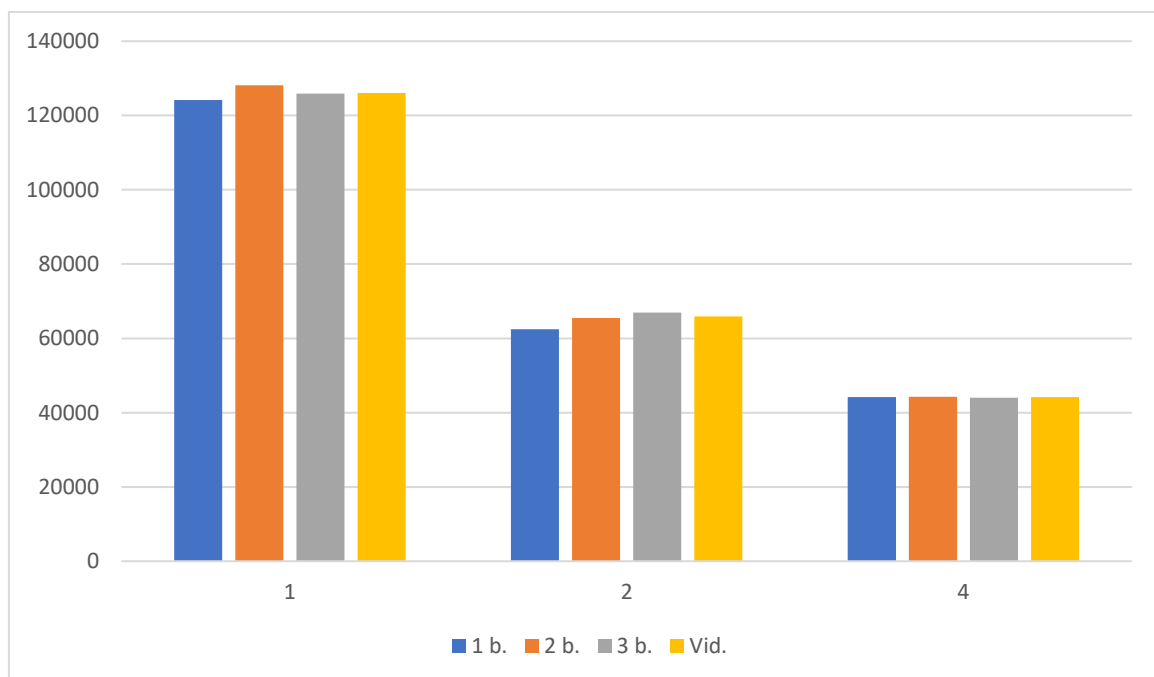
grafikas 9. Procesų darbo laikų palyginimas su 140 taškų



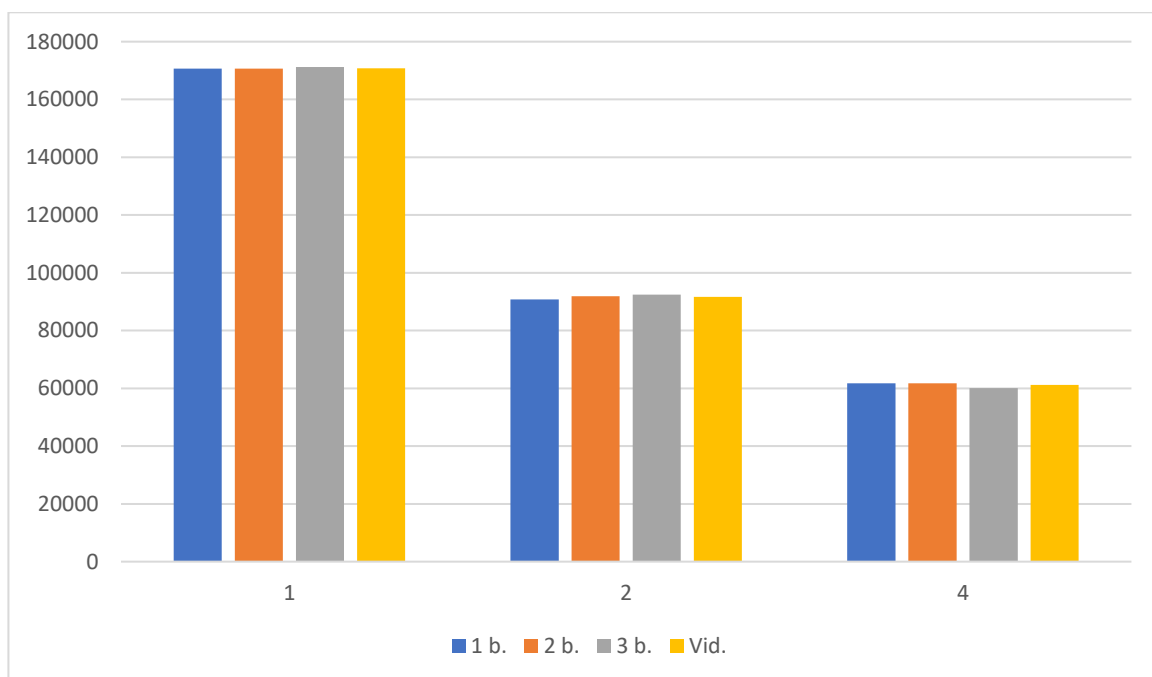
grafikas 10. Procesų darbo laikų palyginimas su 160 taškų



grafikas 11. Procesų darbo laikų palyginimas su 180 taškų



grafikas 12. Procesų darbo laikų palyginimas su 200 taškų



6. Išvados

Su mažai duomenų, vienas procesas arba mažas jų kiekis yra efektyviau nei daug procesų, bet didėjant duomenų kiekiui, didesnis procesų kiekis mažina darbo laiką

Taigi, jei turimas didelis duomenų kiekis, didesnis procesų kiekis padidina programos efektyvumą, bet esant mažiems duomenų kiekiams, lygiagretinti programos neapsimoka, nes tai taip pat užtrunka laiko.

7. Literatūra.

- <https://docs.microsoft.com/en-us/dotnet/csharp/>

8. Paveikslėliai ir lentelės.

LENTELĖ 1 PAGRINDINĖ PROGRAMOS DALIS	4
LENTELĖ 2 BANDYMŲ VYKDYMŲ LAIKAI	6
GRAFIKAS 3. PROCESŲ DARBO LAIKŲ PALYGINIMAS SU 20 TAŠKŲ.....	7
GRAFIKAS 4. PROCESŲ DARBO LAIKŲ PALYGINIMAS SU 40 TAŠKŲ.....	7
GRAFIKAS 5. PROCESŲ DARBO LAIKŲ PALYGINIMAS SU 60 TAŠKŲ.....	8
GRAFIKAS 6. PROCESŲ DARBO LAIKŲ PALYGINIMAS SU 80 TAŠKŲ.....	8
GRAFIKAS 7. PROCESŲ DARBO LAIKŲ PALYGINIMAS SU 100 TAŠKŲ	9
GRAFIKAS 8. PROCESŲ DARBO LAIKŲ PALYGINIMAS SU 120 TAŠKŲ	9
GRAFIKAS 9. PROCESŲ DARBO LAIKŲ PALYGINIMAS SU 140 TAŠKŲ	10
GRAFIKAS 10. PROCESŲ DARBO LAIKŲ PALYGINIMAS SU 160 TAŠKŲ	10
GRAFIKAS 11. PROCESŲ DARBO LAIKŲ PALYGINIMAS SU 180 TAŠKŲ	11
GRAFIKAS 12. PROCESŲ DARBO LAIKŲ PALYGINIMAS SU 200 TAŠKŲ	11