VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Lygiagretieji skaičiavimai

Laboratorinis darbas nr. 3

Tomas Giedraitis
VU MIF Informatika
3 kursas 3 grupė

Vilnius
2021

Turinys

1. Problemos formulavimas

Įgyvendinta automato „Conways's Life Game" simuliacija. Programa gali būti vykdoma dviem režimais – derinimo režimu, kuriame programos vykdymas gali būti dirbtinai sulėtintas ir į ekraną išvedamas programos vykdymo protokolas, paaiškinantis veikimo principus, bei sparčiuoju režimu, kuriame neturi būti dirbtinių stabdymų, o programa išvestų į ekraną vykdymo trukmę.

Programos pradiniai duomenys – tinklelio matmenys, pradinės ląstelių pozicijos, bei iteracijų skaičius.

Programos rezultatas derinimo režimu – „Life Game" ląstelių keitimasis, pavaizduotas vizualiai. Pasirinktinai galima pavaizduoti galutinį tinklelio vaizdą, arba tinklelio vaizdą po kiekvienos iteracijos (1 pav.). Sparčiuoju režimu programa atspausdina kiek laiko užtruko jos vykdymas.

1 pav. „Glider" šablono iteracijos
(iš kairės į dešinę, iš viršaus į apačią). Periodas lygūs 4.

2. Lygiagretusis algoritmas

Suprojektuotas lygiagretus vykdymo algoritmas JAVA programavimo kalba, naudojant JAVA gijas (Threads).

Paprastumo dėlei nagrinėsime kvadratinius „Life Game“ tinklelius. Gijoms yra paskiriama lygiavertė nepriklausoma užduotis – tinklelio dalis, kurią apibūdina nuosekliai išsidėsčiusių eilučių numeriai, nusakomi pradžios ir pabaigos eilutėmis. Tai gali būti nuo vienos eilutės iki viso tinklelio (vienos gijos atveju). Taip pat, siekiant išlaikyti tinklelio dalybą gijoms kuo paprastesnį, nagrinėsime tokio dydžio tinklelius, kad, esant tam tikram gijų skaičiui, tinklelio eilučių skaičius būtų to gijų skaičiaus kartotinis.

Gijų lygiagrečiai atliekamas veiksmas susideda iš dviejų dalių. Pirma, yra apskaičiuojama kiekvieno gijos tinklelio narvelio kaimynų skaičius, o po to, priklausomai nuo kaimynų skaičiaus, atnaujinami narveliai (buvusi ląstelė narvelyje gali išlikti arba pradingti, o nebuvusi gali atsirasti). Šie du veiksmai nesipriešina vienas kitam tarp gijų, t.y. viena gija gali vis dar skaičiuoti kaimynus, o kita jau atnaujinti ląstelių informaciją, nes kiekviena gija dirba tik su savo tinklelio dalimi. Tačiau prieš pradedant naują iteraciją, visos gijos turi pabaigti abu etapus su savo tinkleliu, tad tam yra naudojamas barjeras (`class Barrier`). Kai visos gijos baigia darbą ir iškviečia barjero metodą `waitBarrier()`, joms visoms yra išsiunčiami pranešimai `notifyAll()`, kad toliau galima tęsti veiklą, ir pereinama prie kitos iteracijos.

3. Vykdymo aplinka

Lygiagretaus sprendimo efektyvumas buvo tiriamas daugiaprocesorinėje MIF klasterio sistemoje („beta“ klasteryje – https://mif.vu.lt/cluster/). Jame yra 12 procesorių po 2 virtualius branduolius (iš viso 24 virtualūs branduoliai), 24 GB operatyviosios atminties.

Įkėlus programos išeities failus į MIF klasterį, papildomai naudojamas komandinis failas `programa.sh` paleidimui, kurio turinys:

```
#!/bin/sh
#SBATCH -p short
#SBATCH -N1
#SBATCH -c12
#SBATCH -C beta
# run these commands:
java Main 1 1000
java Main 2 1000
…
java Main 12 1000
```

Čia yra pavyzdys, kai paleidžiamos 12 komandų `java Main` su skirtingais gijų skaičiaus parametrais (nuo 1 iki 12), ir fiksuotu iteracijų skaičiumi (1000). Norint pakeisti tinklelio dydį ir pradinę jo būseną, tai atliekame pačioje JAVA programoje, ir perkompiliuojame ją.

Testavimas paleidžiamas komanda `sbatch programa.sh`, programos eigą galima stebėti komanda `watch -n1 squeue`, o užduoties išvestis bus išsaugota faile pavadinimu `slurm-užduoties_id.out`, kur `id` yra gautas užduoties identifikatorius.

4. Eksperimentinis tyrimas

Spartinimo režimo programos parametrai, be jau aptartų pradinių, yra darbinių gijų skaičius (lygiagretinimo parametras), bei du apkrovos parametrai, nusakantys darbo apimtį (uždavinio dydį) – kvadratinio tinklelio dydis, aprašomas eilučių skaičiumi, ir iteracijų skaičius. Priklausomai nuo darbinių gijų skaičiaus ir tinklelio dydžio, vienai gijai teks tam tikras eilučių skaičius, kurį vadinsime darbų dydžio (grain size) charakteristika. Darbų dydis didėja, didinant tinklelį, kai gijų skaičius išlieka pastovus, arba didinant gijų skaičių, kai tinklelis išlieka pastovus.

Yra tikimasi, kad programa demonstruos vykdymo spartinimą, vykdant ją daugiaprocesorinėje sistemoje. Tyrime nepriklausomi kintamieji – gijų skaičius, uždavinio dydis (tinklelio dydis, iteracijų skaičius) bei tuo pačiu automatiškai ir darbų dydis, o priklausomi – programos vykdymo laikas, spartinimas ir plečiamumas.

5. Tyrimo rezultatai

Pirmiausia klasteryje buvo paleista programa `TTest.java`, kuri leidžia patikrinti java programos vykdymo spartinimą. Jos išeitis:

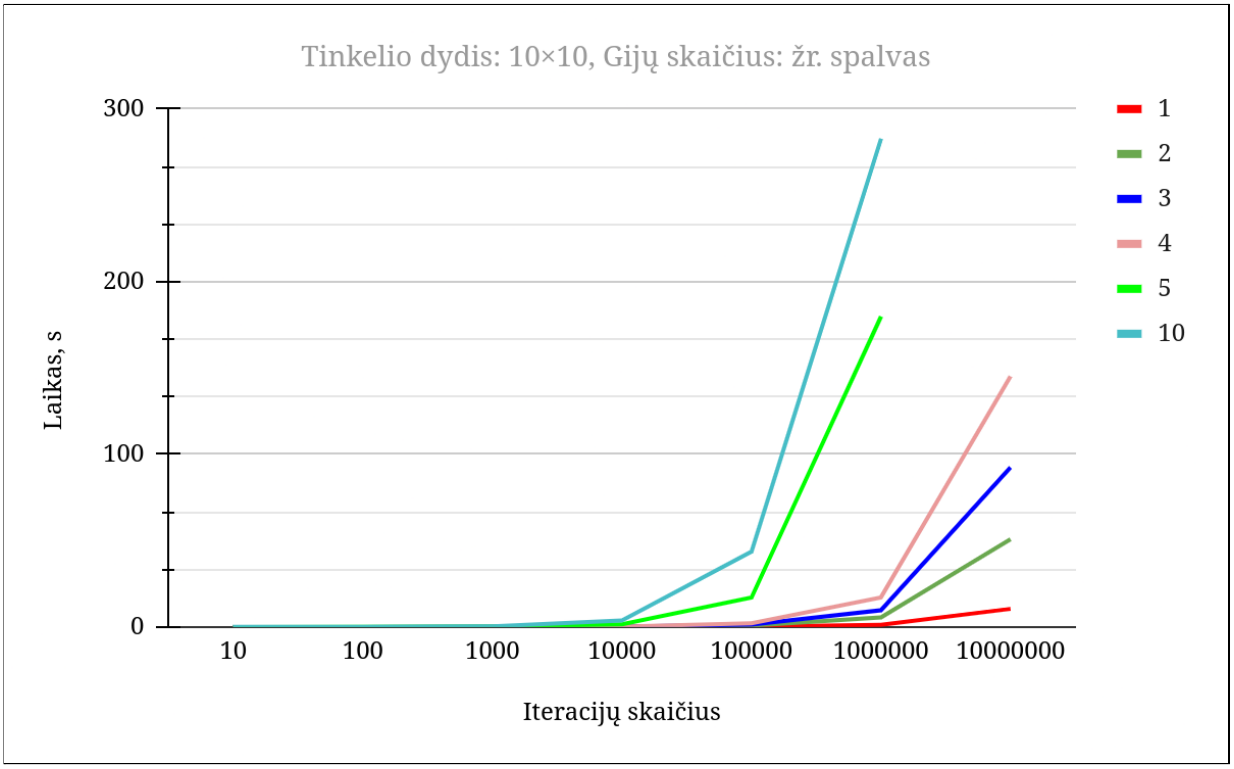
```
Simple system multithreading performance test. Ver 1.3
Parameters: <number threads 1..16> <workload: 1..100000000>
#Make auto test: find workload for > 1 sec…
#nThreads #workload #timeS #speedup
1 1024 0.961 1.0
2 1024 0.594 1.6178451178451179
4 1024 0.353 2.7223796033994336
8 1024 0.164 5.859756097560975
16 1024 0.187 5.13903743315508
32 1024 0.133 7.2255639097744355
#completed
```

Taip pat TTest2.java programos, kuri patikrina atminties prieigos plečiamumą, išeitis:

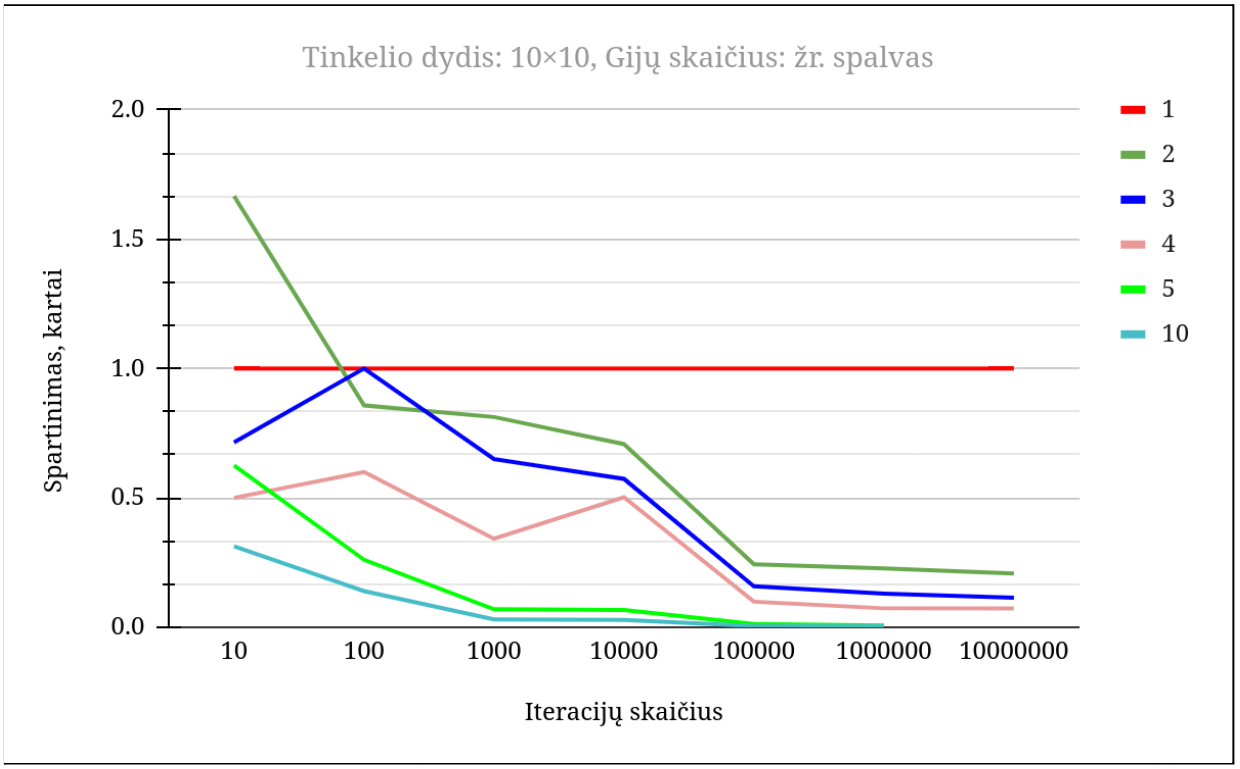
```
Simple system multithreading performance test. Ver 1.3
Parameters: <number threads 1..16> <workload: 1..100000000>
#Make auto test: find workload for > 1 sec…
#nThreads #workload #timeS #speedup
1 128 1.018 1.0
2 128 0.57 1.785964912280702
4 128 0.344 2.9593023255813957
8 128 0.302 3.370860927152318
16 128 0.209 4.8708133971291865
#completed
```

Visi tyrimo duomenys yra pridėtame MS Excel programos faile, todėl šioje ataskaitoje apsiribosime tik grafikais.

Tikriname, kaip iteracijų skaičius daro įtaką tam, kaip gijos atlieka spartinimą. Pasirenkame tinklelio dydį 10×10. Gauname tokius rezultatus:



Grafikas nr. 1



Grafikas nr. 2

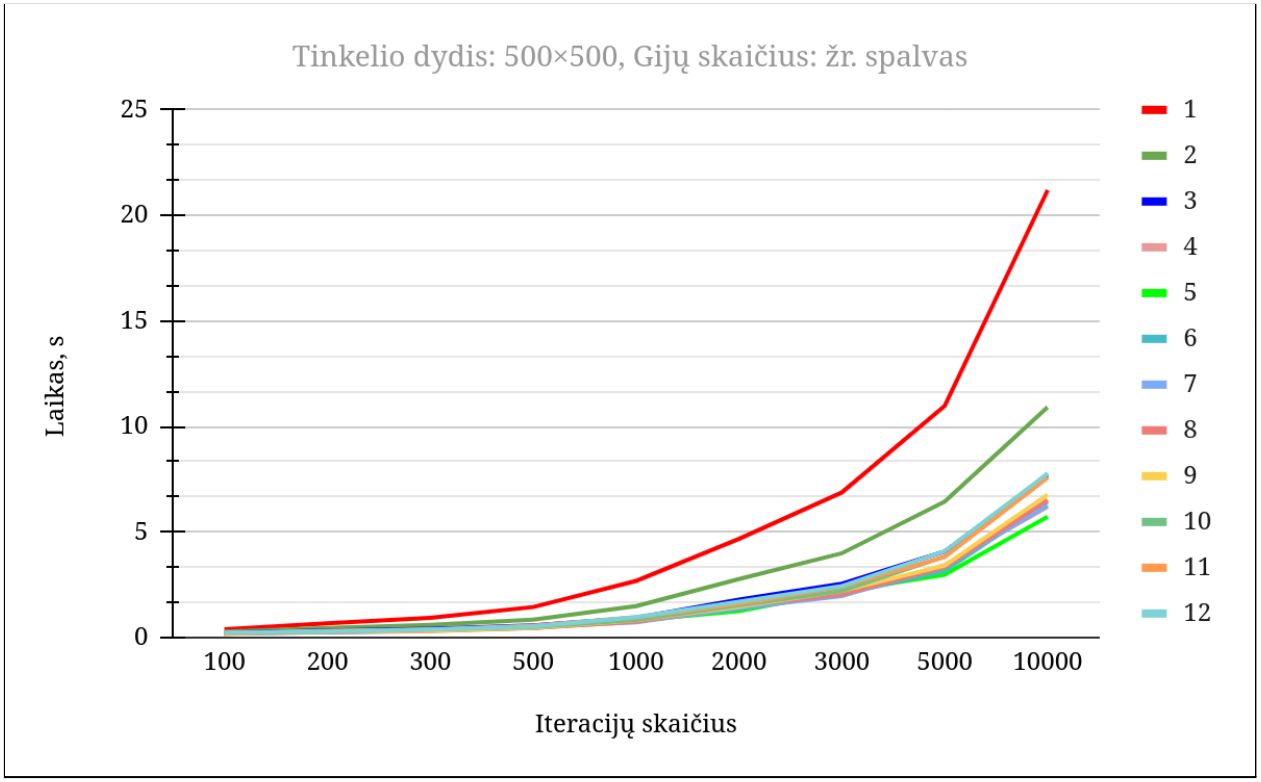
Pirmame grafike matome, kaip kiekvienai gijai, atvaizduojamai skirtingos spalvos kreive, kintant uždavinio iteracijų skaičiui, kinta vykdymo laikas, o antrame – kaip kinta spartinimas. Tiriamos gijos spartinimas yra tiesiog vykdymo laikas vienos gijos atveju, padalintas iš tiriamos gijos vykdymo laiko, taigi tai yra dydis, atvirkščiai proporcingas vykdymo laikui, visada lyginant su vienos gijos (nuosekliu) atveju.

Panašu, kad didėjant gijų skaičiui, spartinimas mažėja. Tai galime paaiškinti tuo, kad pasirinkome mažą tinklelio dydį, kuomet papildomų gijų darbo derinimas užtrunka ilgiau nei vienos gijos nuoseklus darbas. Taip pat, su tokiu mažu tinkleliu net negalime paleisti 12 gijų (12 branduolių).

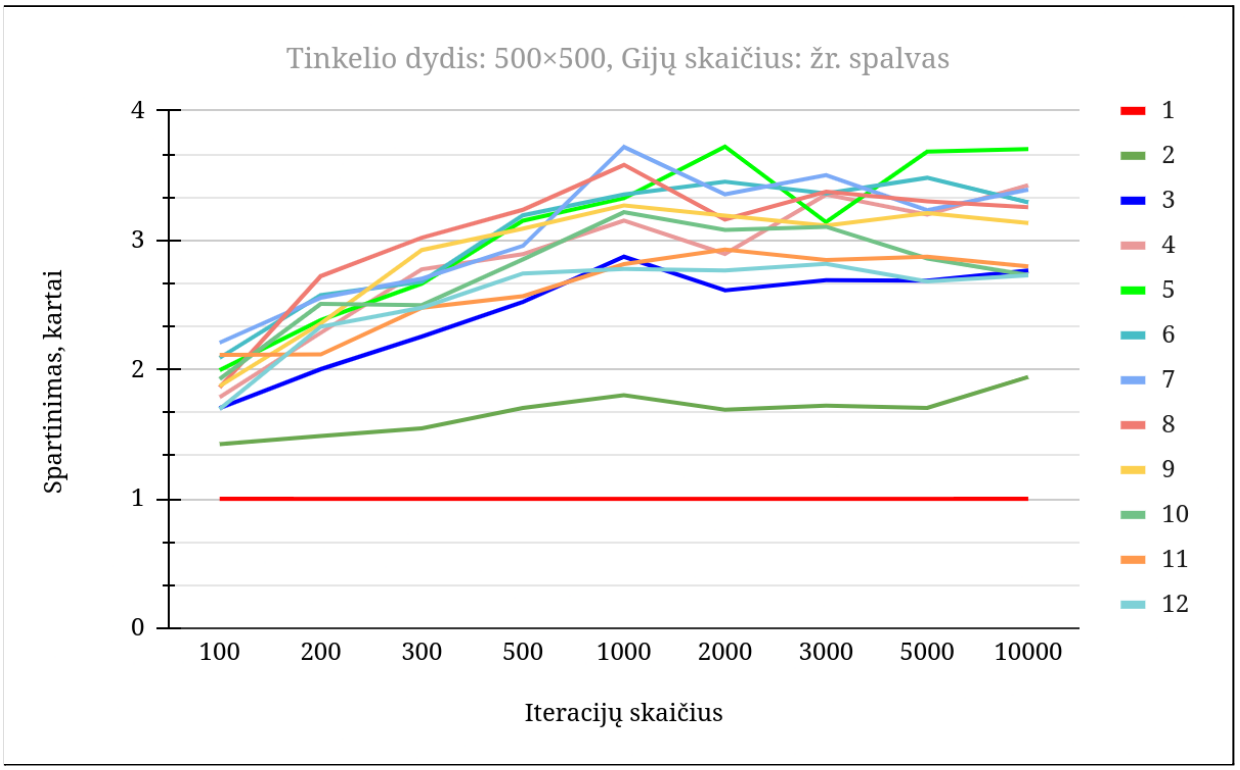
Taigi, panašu, kad su mažu uždaviniu dydžiu išlygiagretinimas tik lėtina programos veikimą. Remiantis šia informacija, galime taip pat daryti prielaidą, kad jei ir kitas uždavinio dydžio parametras – iteracijų skaičius, būtų taip pat pakankamai mažas, tai nors ir pasirinksime didesnį tinklelį, lygiagretus vykdymas spartinimo irgi neduos. Šios situacijos papildomai netirsime, ir nagrinėsime atvejus, kuomet tiek tinklelis, tiek iteracijų skaičius yra pakankamai dideli.

Iteracijų keitimas

Apibrėžkime tinklelio dydį 500×500, ir keiskime iteracijų skaičių intervale nuo 100 iki 10000. Gauname tokius laiko bei spartinimo grafikus:



Grafikas nr. 3



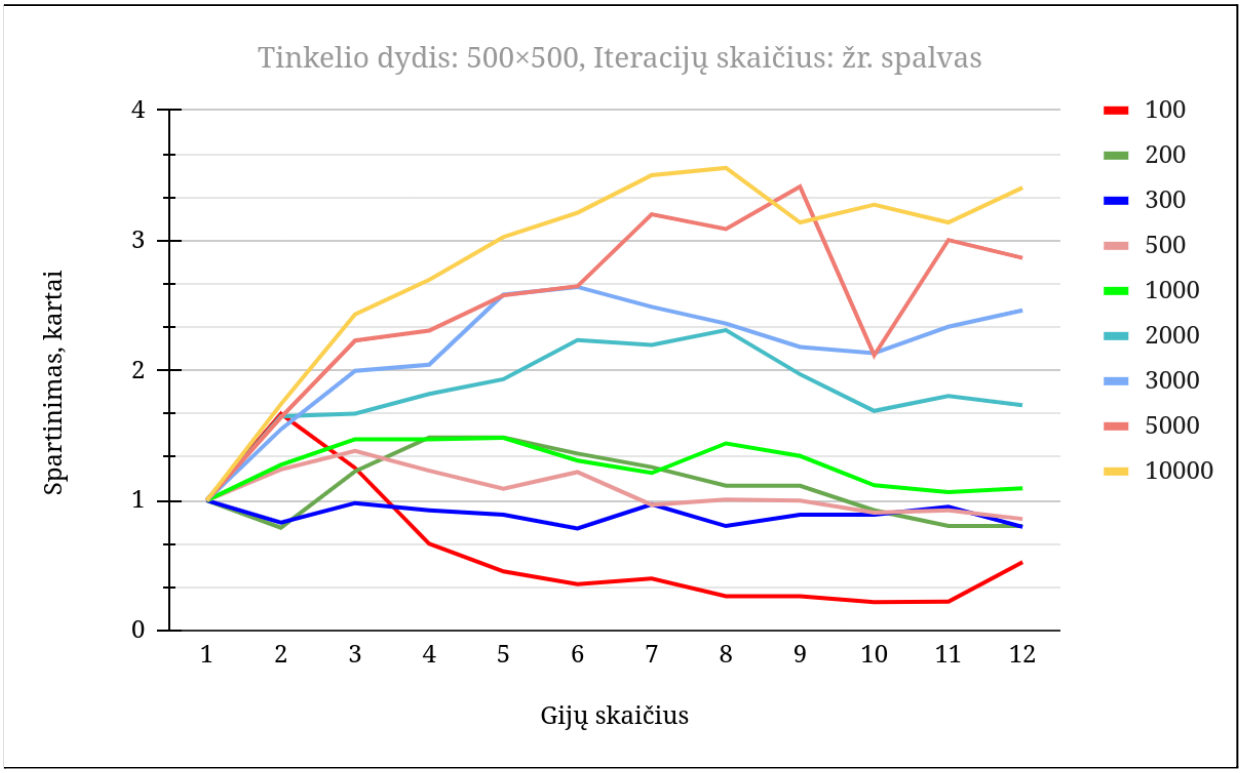
Grafikas nr. 4

Darbų dydis (tinklelio eilučių skaičius, tenkantis vienai gijai) šiuo atveju keitėsi nuo 500 (vienos gijos atveju) iki 42 (12 gijų atveju). Taigi, nepriklausomai nuo iteracijų skaičiaus, plečiamumas šiuo atveju sutampa su maksimaliu gijų skaičiumi, ir yra lygus 12.

Akivaizdu, kad spartinimas įvyksta, ypač gijų skaičiui esant didesniam nei 2, ir ypač padidėja nuo 100 iteracijų perėjus pamažu prie 1000, tačiau ir po to kyla, ir ties 10 000 jau pasiekia 3-3.5 karto.

Kadangi *x*-ašyje šiuose grafikuose yra ne gijų skaičius, o iteracijos, tai šis grafikas labiau atvaizduoja plečiamumą, kur kiekviena kreivė parodo, kaip didėja spartinimas priklausomai nuo iteracijų skaičiaus, gijų skaičiui esant pastoviam toje kreivėje.

Tačiau iš tų pačių duomenų (žr. MS Excel dokumentą) galime nubraižyti ir spartinimo priklausomybę nuo gijų skaičiaus:

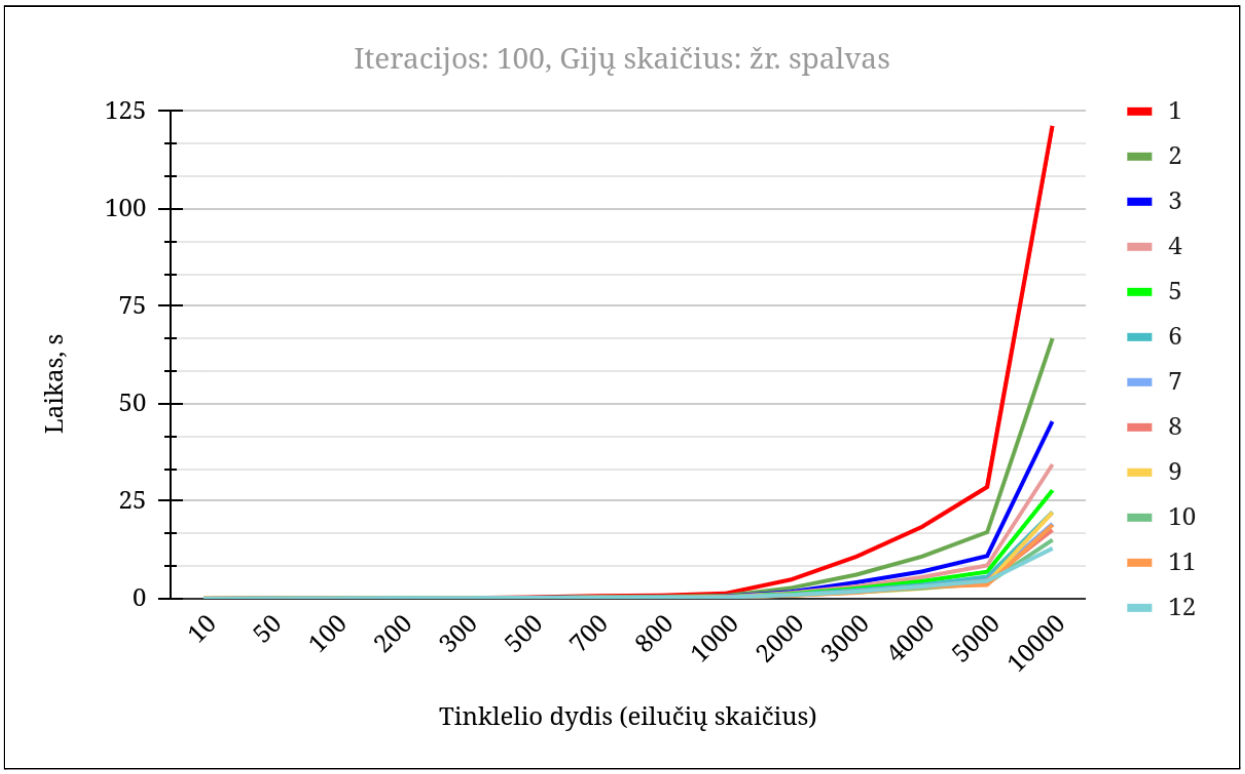


Grafikas nr. 5

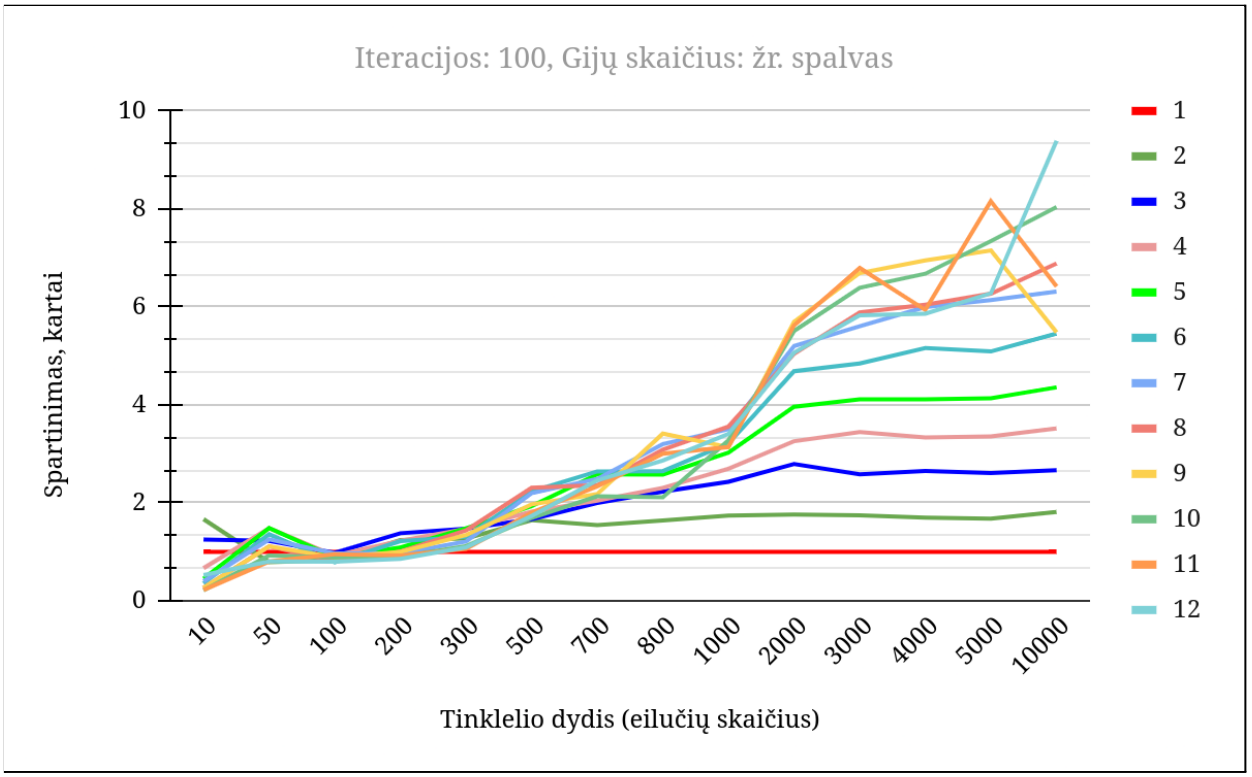
Matome, kad su mažu iteracijų (100-500) skaičiumi spartinimas mažėja didinant gijas, o pradedant nuo 1000 iteracijų jau gauname kiek didesnį spartinimą, tačiau didinant gijas, po kurio laiko įvyksta lūžis, ir spartinimas vėl mažėja, nors ir išlieka teigiamas. Esant 10 000 iteracijų, šis spartinimas lūžta mažiau ir išlaiko panašią vertę. Nors negavome, kad 12 branduolių (gijų) duoda didžiausią spartinimą, kaip tikėjomės, bet iš šio aptarto grafiko galime teigti, kad dar padidinus iteracijų skaičių, galėtumėme gauti didžiausią spartinimą. Testuojant su daugiau nei 12 gijų rezultatai negerėjo, kadangi testavimo aplinkoje buvo naudojama 12 branduolių.

Tinklelio dydžio keitimas

Apibrėžkime iteracijų skaičių lygų 100, ir keiskime tinklelio dydį nuo 10×10 iki 10 000×10 000. Gauname tokius laiko bei spartinimo grafikus:



Grafikas nr. 6

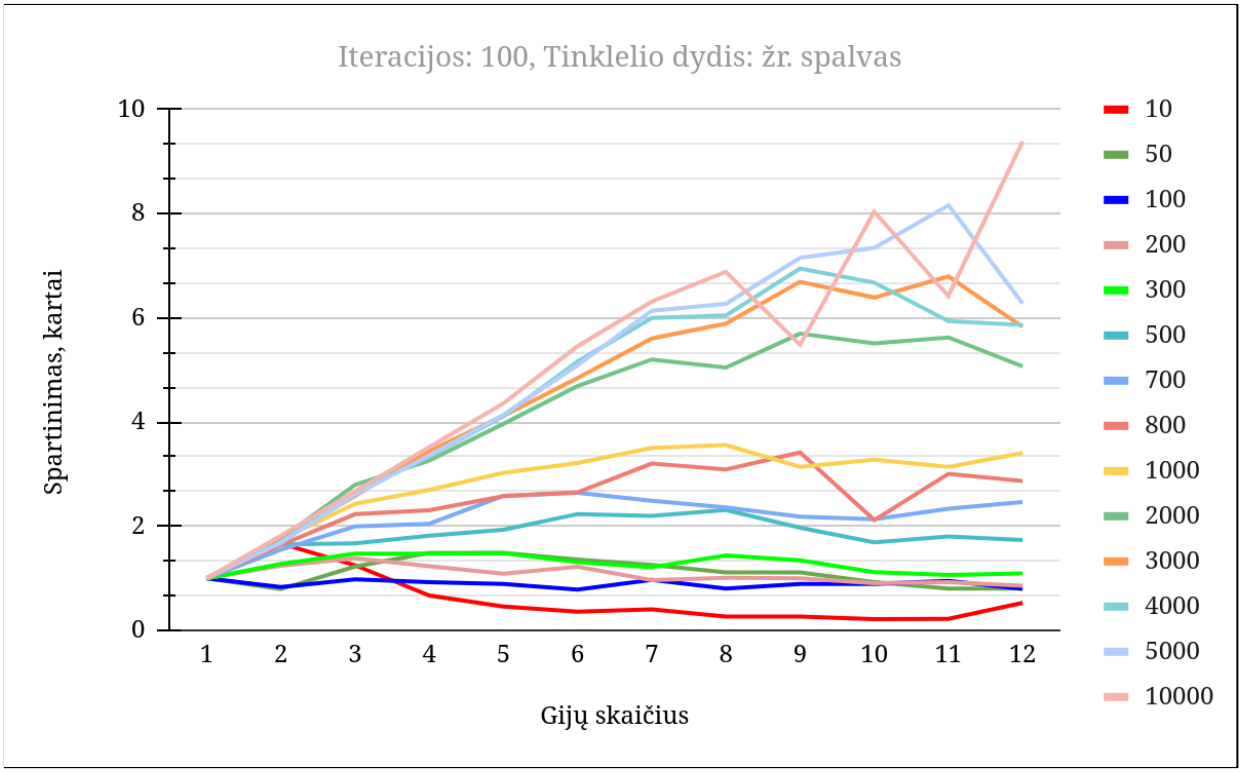


Grafikas nr. 7

Čia darbų dydis kito ne tik priklausomai nuo gijų skaičiaus, bet ir nuo tinklelio dydžio. Kiekvienu atveju darbų dydis randamas tinklelio dydį padalinus iš gijų skaičiaus.

Matome, kad spartinimas įvyksta, ir nuosekliai didėja, didinant tinklelį, ir didinant gijų skaičių. Iš šių grafikų, nagrinėdami atskiras kreives, taip pat matome uždavinio plečiamumą.

Taip pat nubrėžiame spartinimo, priklausančio nuo gijų skaičiaus, grafiką:



Grafikas nr. 8

Iš grafiko matome, kad šiuo atveju taip pat uždavinio dydis turi būti pakankamas (nuo 500×500), kad gautumėme spartinimą. Tuomet spartinimas didėja ir pasiekia 2-6 karto, tačiau su 12 gijų vis tiek jis būna mažesnis, išskyrus vienintelį atvejį, kada iteracijų skaičius yra 10 000 – tuomet 12 gijų spartinimas yra didžiausias (viršija 9 kartus). Panašu, kad su šiuo ir dar didesniu iteracijų skaičiumi gauname (ir gausime) tokį spartinimo rezultatą, kokio ir tikėjomės, t.y. maksimalų, esant maksimaliam gijų skaičiui.

6. Išvados

Galime daryti išvadą, kad algoritmą išlygiagretinus, spartinimas pasiekiamas, kai uždavinio dydis yra pakankamai didelis, kitaip gijų darbo derinimo laiko išlaidos kainuos brangiau nei nuoseklus vienos gijos darbas.

Suradę reikiamą uždavinio dydį, tiek iteracijų, tiek tinklelio dydžio atžvilgiu, atlikome spartinimo ir plečiamumo tyrimus. Pastebime, kad plečiant uždavinio dydį, spartinimas visada didėja. Tačiau tiriant spartinimą priklausomai nuo gijų skaičiaus, pastebime, kad su mažesnio dydžio uždaviniais, nors ir turime spartėjimą, tačiau jis turi tam tikrus lūžius su didesniu gijų skaičiumi ir pradeda mažėti (nors visumoje teigiamas spartinimas išlieka). Tai įvyksta todėl, kad uždavinio dydis dar nėra pakankamai didelis, kad gautumėme laukiamą rezultatą panaudojus visas 12 gijų. Tik paskutiniais tirtais atvejais, tiek su iteracijų skaičiumi, o ypač su tinklelio

dydžiu, gauname aiškesnį spartinimo padidėjimą, didinant gijas iki 12. Šio tyrimo patobulinimui galima būtų abiem atvejais paimti dar didesnį uždavinio dydį, ir, tikėtina, gausis dar aiškesnis spartinimo didėjimas, didinant gijas iki maksimalaus skaičiaus.