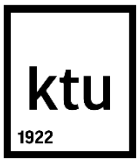


Kauno technologijos universitetas
Informatikos fakultetas

Lygiagretusis programavimas (P170B328)

Inžinerinio projekto ataskaita

Vytenis Kriščiūnas IFF-1/1

Studentas

Lekt. Barisas Dominykas

Doc. Vasiljevas Mindaugas

Dėstytojai

Kaunas 2023

TURINYS

1. Užduoties analizė ir sprendimo metodas

Sprendžiama problema – optimizavimo uždavinys. Yra duotos taškų koordinatės, kurios nurodo parduotuvių vietas mieste, reikia pastatyti naujas parduotuves taip, kad pastatymo kaina būtų kuo mažesnė. Ši kaina yra skaičiuojama pagal naujų parduotuvių pastatymo mieste koordinates ir atstumus tarp parduotuvių.

Atstumo tarp dviejų parduotuvių, kurių koordinatės $(\boldsymbol{x_1}, \boldsymbol{y_1})$ ir $(\boldsymbol{x_2}, \boldsymbol{y_2})$, kaina apskaičiuojama pagal formulę:

$$C(x_1, y_1, x_2, y_2) = \exp(-0.3 \cdot ((x_1 - x_2)^2 + (y_1 - y_2)^2))$$

Parduotuvės, kurios koordinatės $(\boldsymbol{x_1}, \boldsymbol{y_1})$, vietos kaina apskaičiuojama pagal formulę:

$$C^P(x_1, y_1) = \frac{x_1^4 + y_1^4}{1000} + \frac{\sin(x_1) + \cos(y_1)}{5} + 0.4$$

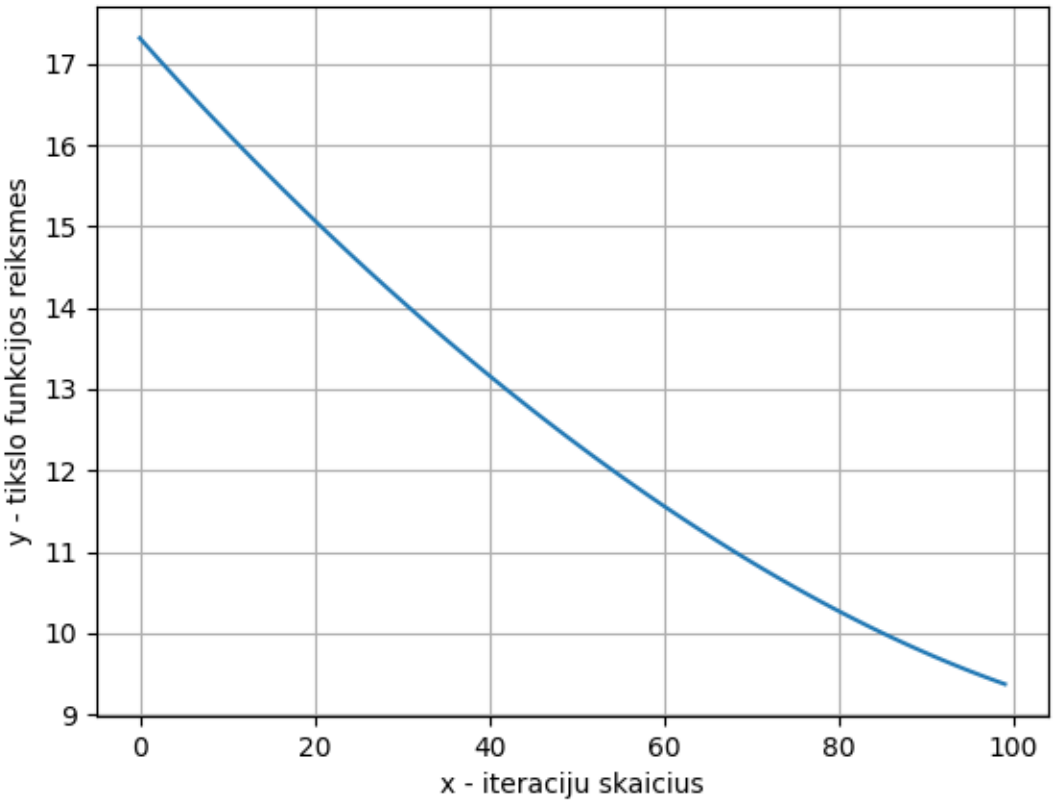
Optimalios parduotuvių koordinatės yra randamos pagal tikslo funkciją, kuri yra sudaroma iš duotų formulių. Reikia rasti kuo įmanoma mažesnes tikslo funkcijos reikšmes, kurios indikuotų, kad bendrą mažiausią parduotuvių mieste pastatymo kainą. Optimizavimas programos vykdymo metu įvykdytas korektiškai tada, kai gaunama tikslo funkcijos reikšmė yra mažesnė už pradinę reikšmę. Yra naudojamas ciklas, kuris veikia tam tikrą iteracijų skaičių ir skaičiuoja naujas tikslo funkcijos reikšmes.

Problemos optimizavimui yra naudojama greičiausio nusileidimo gradiento funkcija, kuri randa naujas koordinačių reikšmes parduotuvėms, vėliau šios koordinatės yra naudojamas apskaičiuoti tikslo funkciją. Gradiento metodo ir jo viduje apskaičiuojamos tikslo funkcijos bendras asimptotinis sudėtingumas yra $\theta(n^2)$. Kadangi gradientas yra skaičiuojamas visoms koordinatėms atskirai, šį metodą galima išlygiagretinti – paleisti kelis procesus. Kiekvienas procesas apdoroja tam tikrą kiekį duomenų ir galiausiai yra gražinamas bendras apskaičiuotų koordinačių rinkinys.

2. Testavimas ir programos vykdymo instrukcija

Programos veikimo korektiškumą galima patikrinti pagal grafiką sudarytą iš iteracijų skaičiaus ir tikslo funkcijos reikšmės pokyčio. Jis parodo vis mažėjančios tikslo funkcijos reikšmę. Taigi, programos

optimizavimas yra sėkmingas.



Nusprendžiau naudoti python multiprocessing.Pool ir kurti atitinkamą kiekį procesų.

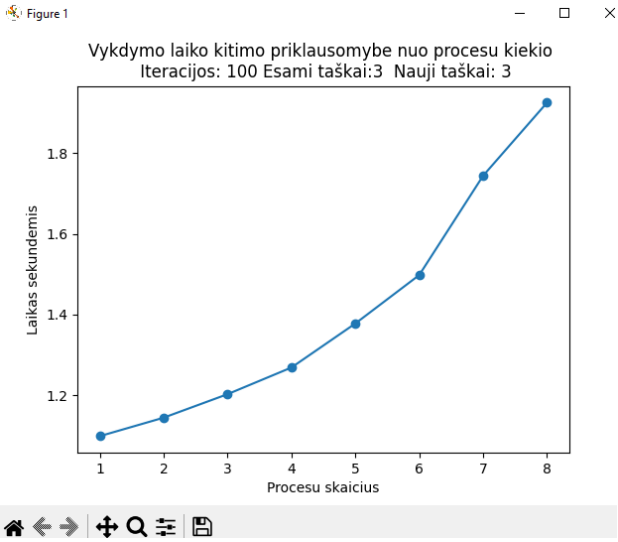
Programa yra paleidžiama iškvietus vieną ar kelis iš pasirinktų aštuonių programos kvietinių, testavimo variantų:

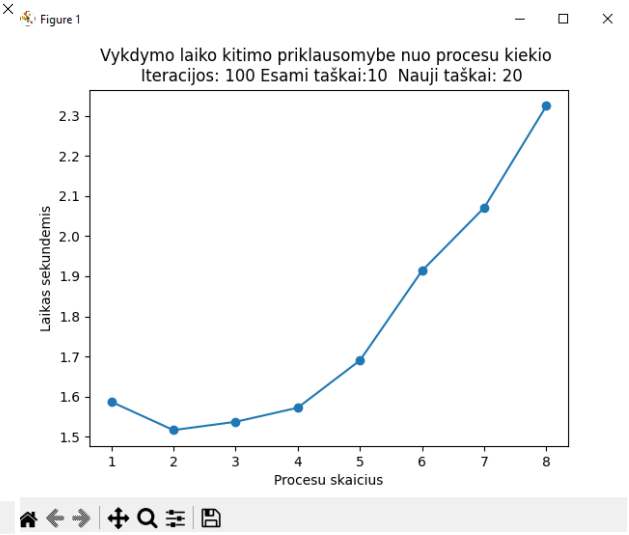
```
if __name__ == "__main__":
    #test(procesuKiekis, zingsnis, iteracijuSkaicius, failoPavadinimas)

    test(8, 0.01, 100, "exsisting3_new3.json") #beveik tiese
    test(8, 0.01, 100, "exsisting10_new20.json") #beveik tiese
    test(8, 0.01, 100, "exsisting20_new20.json") #beveik tiese
    test(8, 0.01, 100, "exsisting12_new47.json") #Parabole
    test(8, 0.01, 100, "exsisting10_new50.json") #Parabole
    test(8, 0.01, 100, "exsisting60_new60.json") #Parabole
    test(8, 0.01, 100, "exsisting100_new100.json") #Hiperboles desine puse
    test(8, 0.01, 100, "exsisting120_new120.json") #Hiperboles desine puse
```

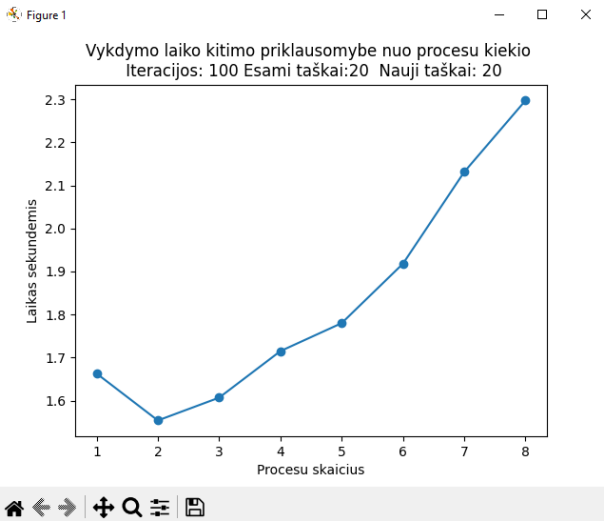
Pirmojo skaičiaus reikšmė – procesų kiekis, antrojo – žingsnis (naudojamas taškų koordinačių skaičiavimams), trečiojo – iteracijų skaičius ir ketvirtojo – sugeneruoto taškų koordinačių .json failo pavadinimas.

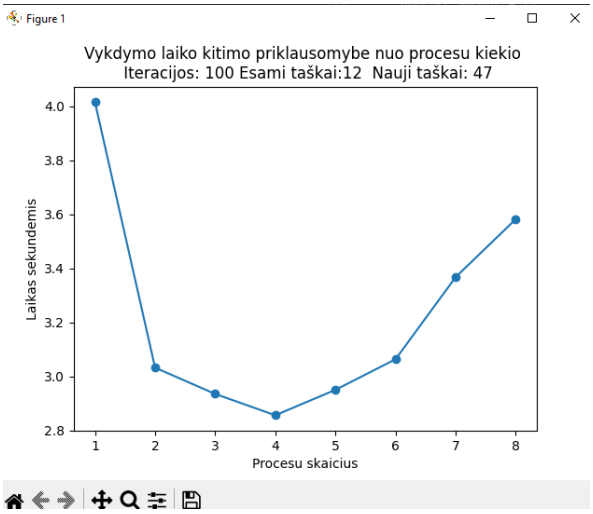
3. Vykdymo laiko kitimo tyrimas

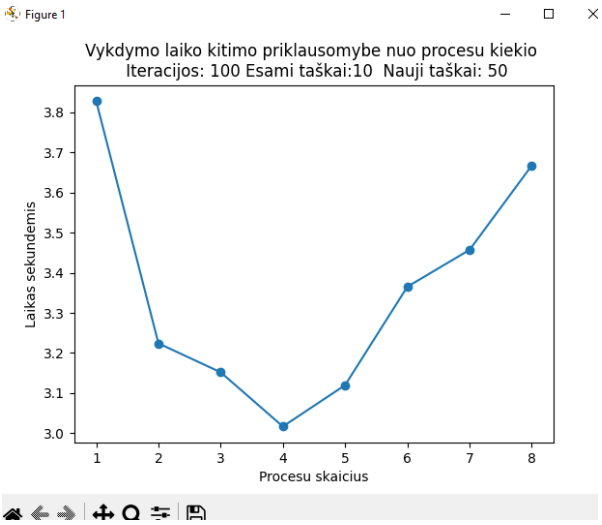
Pirmo tyrimo metu yra bandoma keisti taškų koordinačių kiekius norint stebėti kaip kinta vykdymo laikas priklausomai nuo procesų kiekio. Visa tai yra atvaizduojama grafiškai:

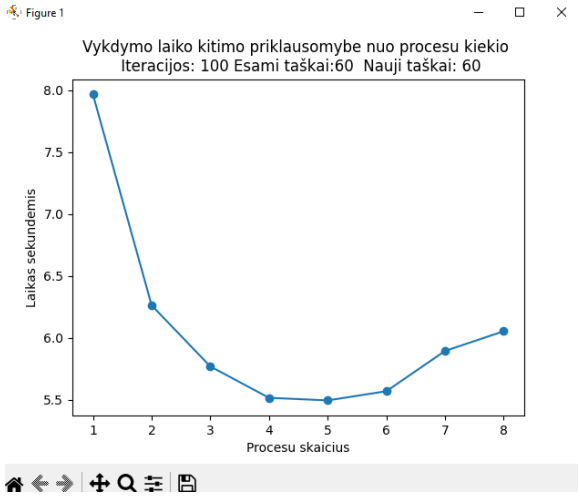


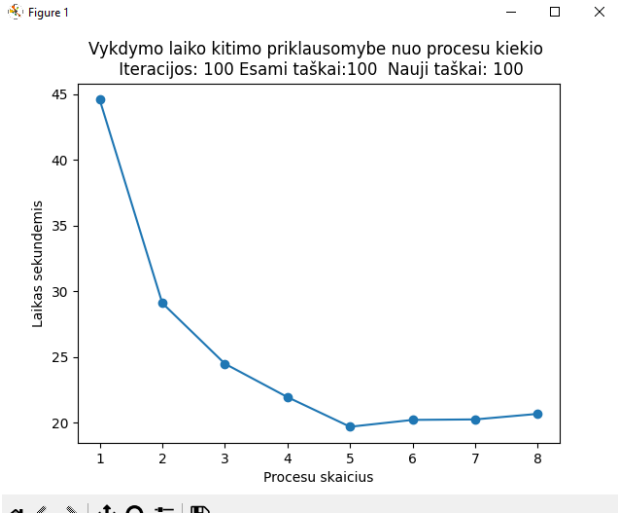


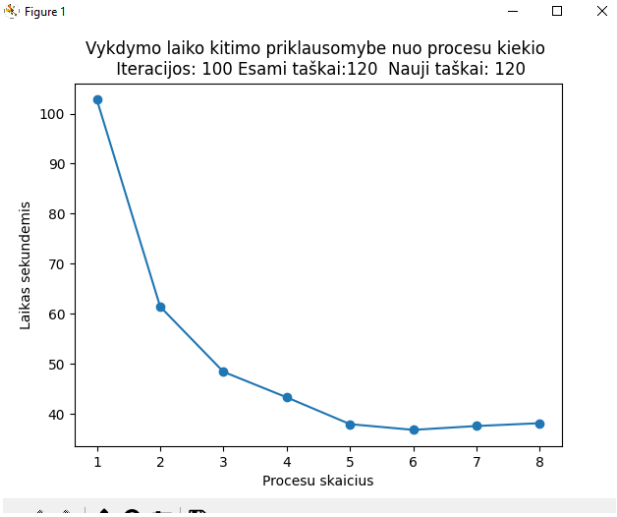












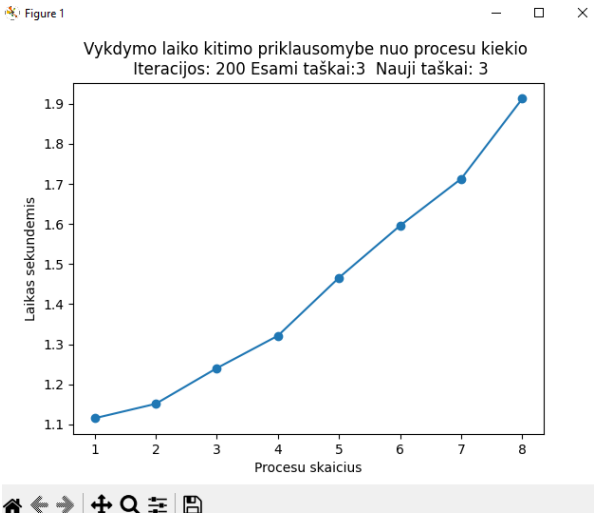
Vidutiniai programos vykdymo laikai, kai naudojama 100 iteracijų:

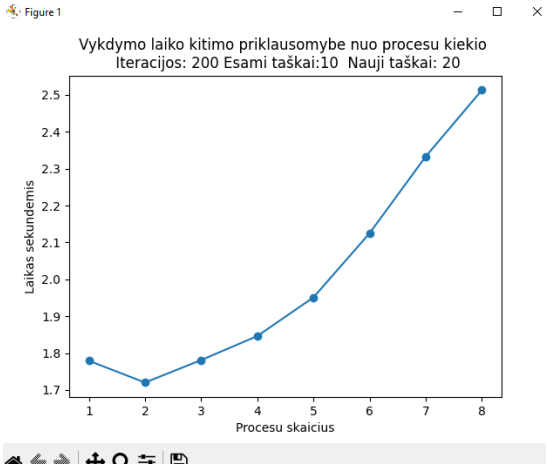
- Esami taškai: 3, nauji taškai: 3 - 1.4s
- Esami taškai: 10, nauji taškai: 20 - 1.7s
- Esami taškai: 20, nauji taškai: 20 - 1.8s
- Esami taškai: 12, nauji taškai: 47 - 3.2s
- Esami taškai: 10, nauji taškai: 50 – 3.3s
- Esami taškai: 60, nauji taškai: 60 – 6.1s
- Esami taškai: 100, nauji taškai: 100 – 25.1s
- Esami taškai: 120, nauji taškai: 120 – 50.1s

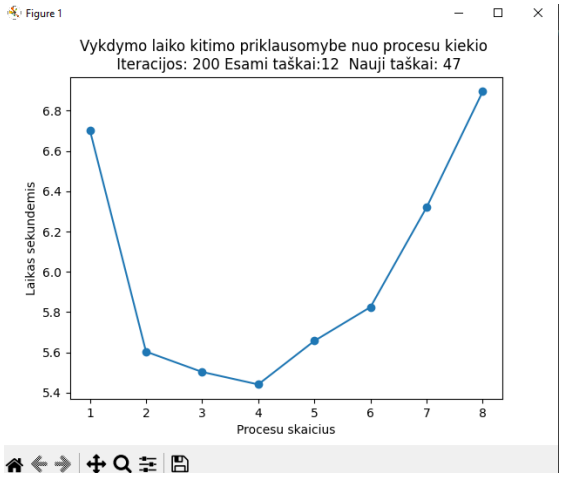
Iš pirmųjų grafikų su nedideliais duomenų kiekiais galima manyti, kad bandymas išlygiagretinti programą naudojant vis daugiau procesų kenkia jos greitaveikai, tačiau vėlesni grafikai rodo visai kitokią tendenciją. Kuo daugiau duomnų programa turi apdoroti, tuo didesnis procesų kiekis gerina

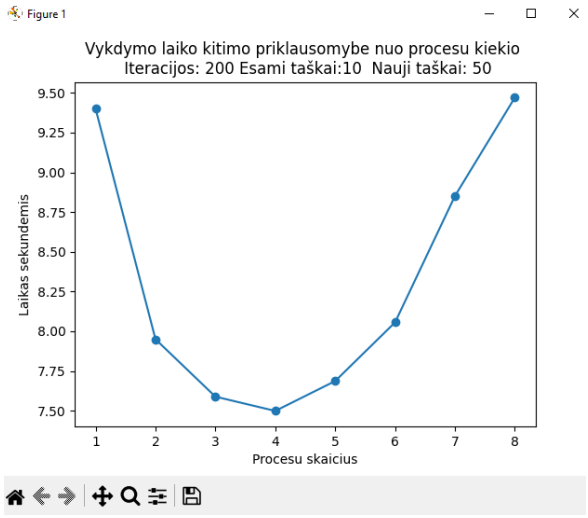
greitaveiką. Kadangi procesų paleidimo laikas yra didesnis nei paprastų gijų, todėl su mažais duomenų kiekiais nepastebime laiko pagerėjimo. Taip pat galima pastebėti, kad procesų kiekis neprivalo būti labai didelis, kad pagerėtų greitaveika, reikia ieškoti optimalaus jų kiekio, nes per didelis procesų kiekio panaudojimas taip pat kenkia programos vykdymo laikui.

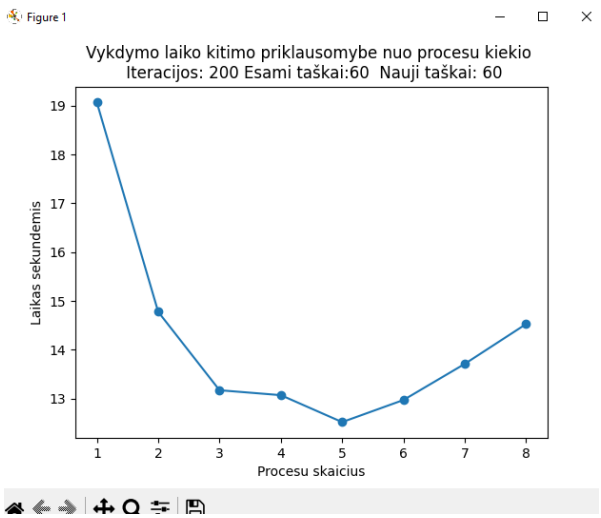
Antrojo tyrimo metu naudojamis tokie patys taškų koordinačių kiekiai, tačiau yra pakeičiamas iteracijų skaičius.

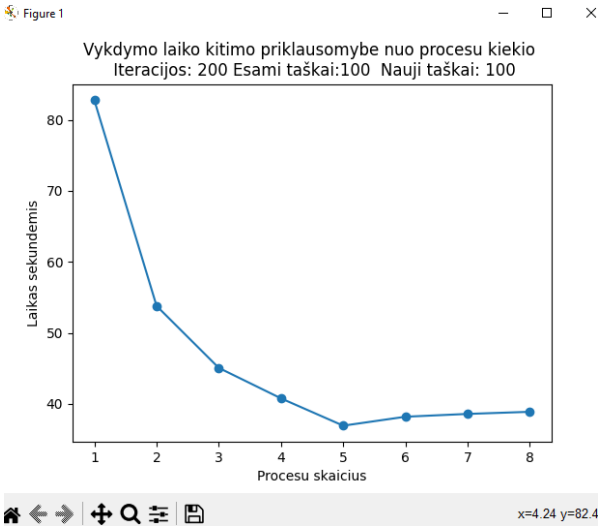


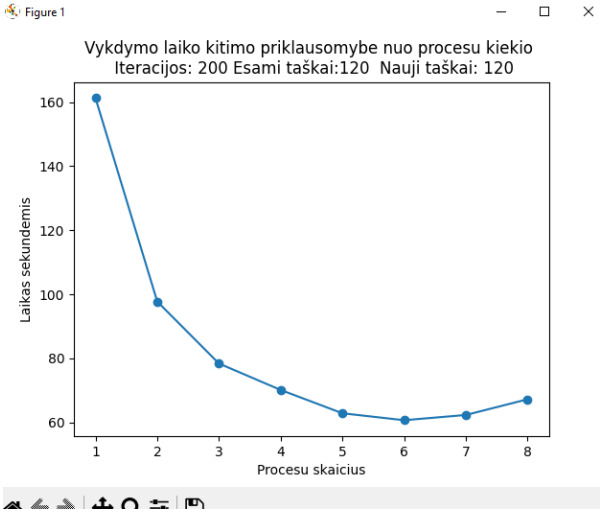












Vidutiniai programos vykdymo laikai, kai naudojama 200 iteracijų:

- Esami taškai: 3, nauji taškai: 3 - 1.4s
- Esami taškai: 10, nauji taškai: 20 - 2s
- Esami taškai: 20, nauji taškai: 20 - 3.7s
- Esami taškai: 12, nauji taškai: 47 - 5.9s
- Esami taškai: 10, nauji taškai: 50 – 8.3s
- Esami taškai: 60, nauji taškai: 60 – 14.2s
- Esami taškai: 100, nauji taškai: 100 – 46.8s
- Esami taškai: 120, nauji taškai: 120 – 82.6s

Kaip matome grafikai mažai pakito, taigi iteracijų skaičius neturi įtakos kokiam procesų kiekiui esant pakis vykdymo laikas., tai priklauso nuo duomenų kiekio. Pakito tik vykdymo laikas, kadangi

iteracijų kiekis padvigubėjo, todėl ir programos vykdymo laikas su daugiau duomenų beveik padvigubėjo.

4. Išvados ir literatūra

Python programavimo kalba yra pakankamai nesudėtinga, todėl pasirinkau naudoti jos siūlomą multiprocessing.Pool, kad išlygiagretinčiau programą procesų pagalba. Taip pat mano rašytas kodas jau buvo naudojamas nuosekliam vykdymui, todėl tiesiog reikėjo jį išlygiagretinti. Pagrindinis sunkumas pasirodė procesų paleidimo laikas su nedideliais duomenų kiekiais. Iš pradžių atrodė, kad programos veikimas labai sulėtėja naudojant šiuos lygiagretinimo įrankius, tačiau atlikus veiksmus su daugiau duomenų kelių procesų paleidimo nauda pasireiškė.

Pagrindinis šaltinis - https://stackoverflow.com