

**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**Daugardas Lukšas**

**P170B328 Lygiagretusis programavimas**

Inžinerinis projektas

**Dėstytojai**

lekt. Dominykas Barisas

lekt. Mindaugas Jančiukas

**KAUNAS, 2023**

Turinio lentelė

[1 Užduotis 3](#_Toc1)

[2 Užduoties analizė ir sprendimo metodas 3](#_Toc2)

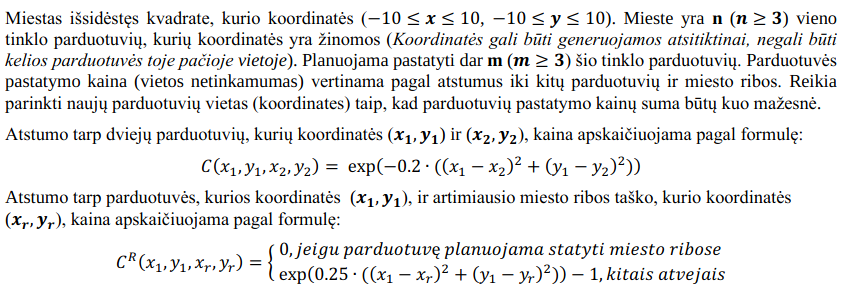
[3 Testavimas ir programos vykdymo instrukcija 3](#_Toc3)

[4 Vykdymo laiko kitimo tyrimas: 3](#_Toc4)

[5 Išvados 3](#_Toc5)

[6 Literatūra 4](#_Toc6)

# Užduotis



# Užduoties analizė ir sprendimo metodas

Turime *n* parduotuvių koordinates, kurios jau yra pastatytos mieste. Reikia pastatyti dar *m* parduotuvių optimaliausiose vietose mieste arba aplink jį. Kad tam padaryti naudosime gradiento metodą, kuri optimizuos naujos parduotuvės pastatymo kainą pagal vietą.

Taikomos duomenų lygiagretinimo priemonė yra C++ su OpenMP parallel for. Šią priemonę naudojame išlygiagretinti ciklus, kurie daro kokius nors veiksmus su parduotuvėmis.

# Testavimas ir programos vykdymo instrukcija

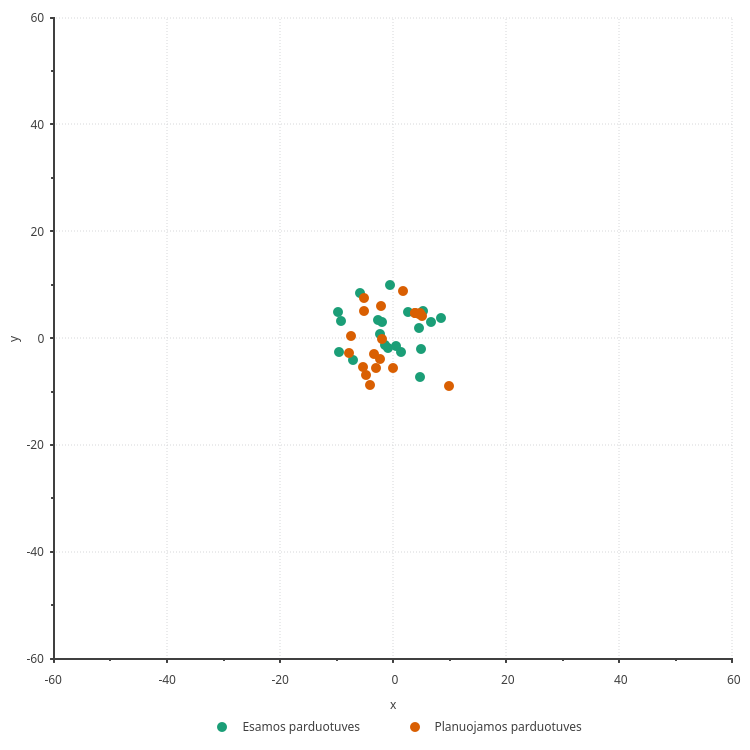
## Testavimas

Programa gauna teisinga rezultatą, nes naujos parduotuvės yra pakankamai viena nuo kitos nutolusios, kartu su tinkamu atstumu nuo jau pastatytų parduotuvių.

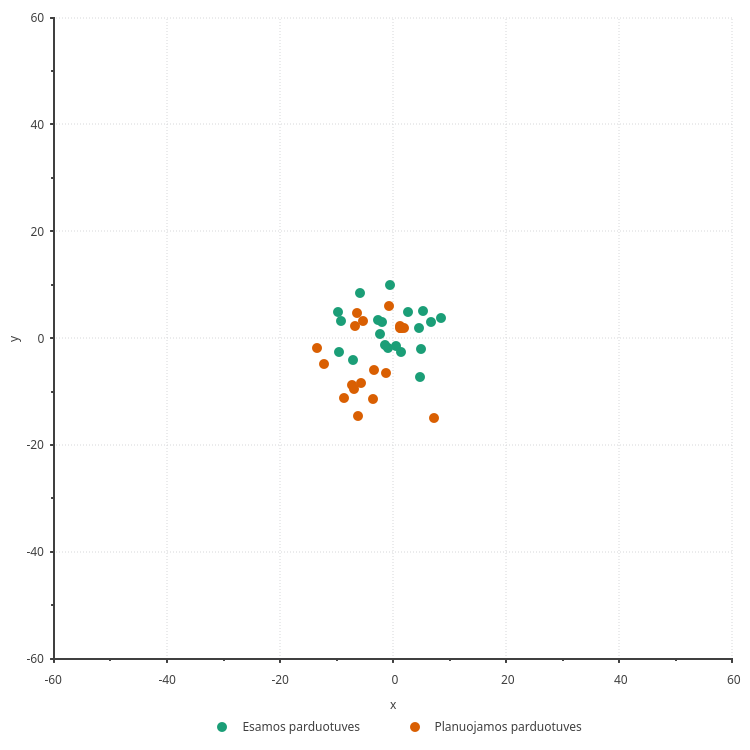
Testavimo rezultatai su skirtingais jau pastatytų ir naujų parduotuvių skaičiais:

Esamų parduotuvių skaičius: 20.

Naujų parduotuvių skaičius: 20.



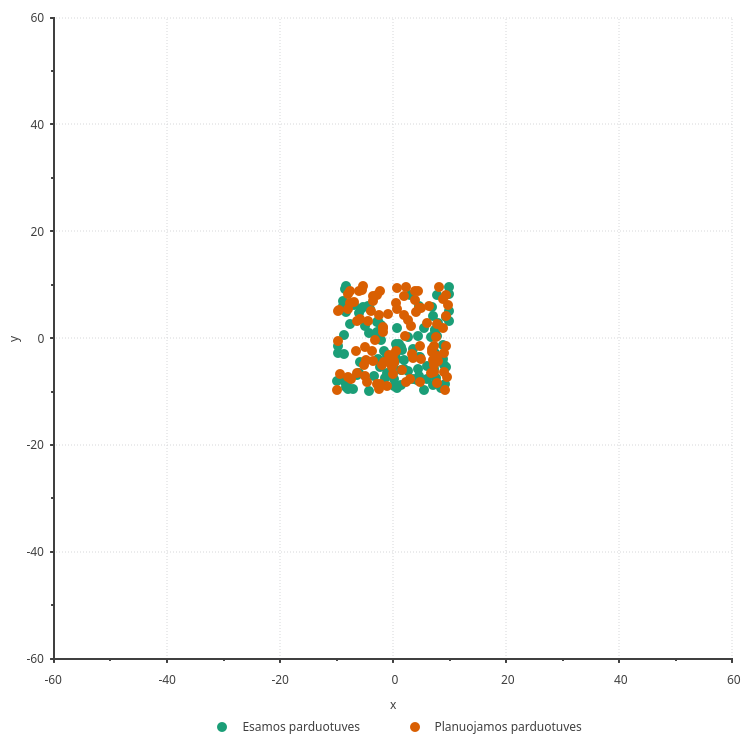
Pav. 1 pradinės planuojamų statyti parduotuvių koordinatės, kartu su jau pastatytomis parduotuvėmis



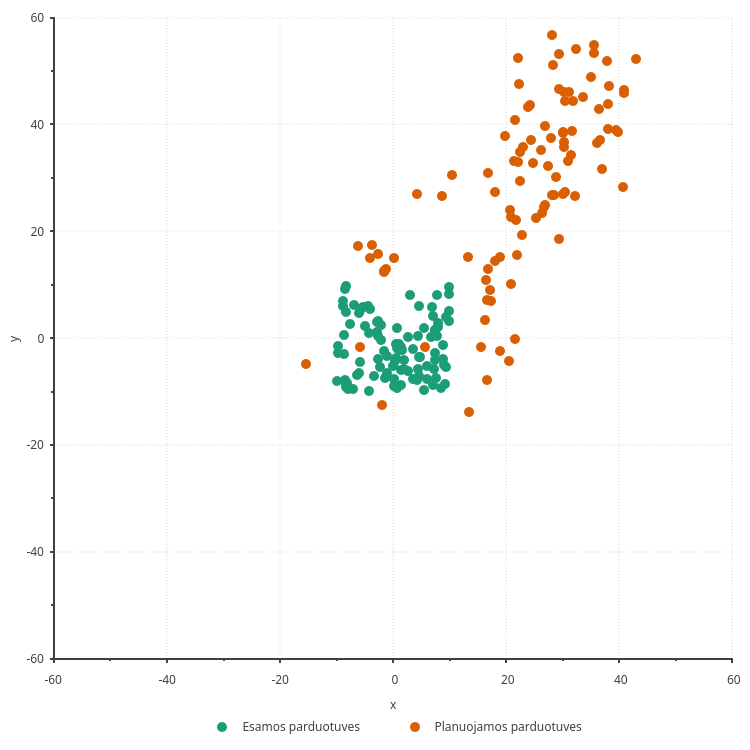
Pav. 2 Optimizuotos planuojamų statyti parduotuvių koordinatės, kartu su jau pastatytomis parduotuvėmis

Esamų parduotuvių skaičius: 100.

Naujų parduotuvių skaičius: 100.



Pav. 3 Pradinės planuojamų statyti parduotuvių koordinatės, kartu su jau pastatytomis parduotuvėmis



Pav. 4 Optimizuotos planuojamų statyti parduotuvių koordinatės, kartu su jau pastatytomis parduotuvėmis

## Programos vykdymo instrukcija

Programa kompiliuojama su *cmake* įrankiu. Direktorijoje, kurioje yra *main.cpp* failas reikia įvykdyti šias komandas:

mkdir build && cd build && cmake ../ && cmake —build .

Įvykdžius šias komandas *build* aplankale turėtų atsirasti *IP* paleidimo failas, kuris ir paleidžia programą.

Kad programa apskaičiuotų jau sukurtus duomenų rinkinius, reikia, kad jie būtų įkelti į *build* direktoriją. Jei ten nebus kode nurodyto varianto duomenų rinkinio, tai tuomet jis bus sugeneruotas automatiškai.

# Vykdymo laiko kitimo tyrimas:

Kiekviena diagrama parodo vidutinę programos sprendimo trukmę tam tikram gijų skaičiui, su tam tikru duomenų rinkiniu.

Iš pirmos – trečios diagramos matome, kad kai gijų skaičius pasiekia 18, programos vykdymo laikas padidėja eksponentiškai. Tai yra dėl to kad duomenų kiekis yra per mažas tokiam gijų skaičiui, todėl daugiau laiko užtrunkama operacinei sistemai keičiant vykdančiąją giją, nei spręsti pačiai problemai (kai naudojamų gijų skaičius yra didesnis nei procesorius turi fizinių gijų).

Iš diagramų, kurios sudarytos su didelės apimties duomenų rinkiniais, duomenų užtenka visoms gijoms pilnai dirbti, įskaitant ir nefizinėms gijoms.

# Išvados

Kadangi mano testavimo įranga turi 16 fizinių gijų, buvo įdomu sužinoti kaip pasikeičia programos vykdymo trukmė, kai programa pradeda naudoti daugiau gijų, nei yra fiziškai ir kaip tą laiką įtakoja duomenų rinkinio dydis.

OpenMP naudojimas C++ programavimo kalboje yra labai patogus su tokio tipo problemomis, kai yra daug ciklų su konkrečiais jų apdorojimo kartais. Buvo labai patogu beveik kiekvieną ciklą išlygiagretinti ir išlygiagretinimas paspartino programos vykdymo laiką tūkstančiais kartu su pakankamai dideliais duomenų rinkiniais.

# Literatūra

1. OpenMP dokumentacija (https://www.openmp.org/resources/refguides/)
2. C++ dokumentacija (https://en.cppreference.com/w/)
3. P170B115 Skaitiniai metodai ir algoritmai, 2 inžinerinio darbo, 3 užduoties, 4 variantas.