KAUNO TECHNOLOGIJOS UNIVERSITETAS
Informatikos fakultetas

P170B328 Lygiagretusis programavimas

Inžinerinis projektas — duomenų lygiagretumo priemonių taikymas

Priėmė:
Doc. prakt. Ryselis Karolis
Parengė:
Martynas Kuliešius IFF-1/9

KAUNAS, 2023

Turinys

1. **Užduotis**

Uždavinys 7-10 variantams

Miestas išsidėstęs kvadrate, kurio koordinatės $(-10 \le \boldsymbol{x} \le 10,\ -10 \le \boldsymbol{y} \le 10)$. Mieste yra **n** $(\boldsymbol{n} \ge \mathbf{3})$ vieno tinklo parduotuvių, kurių koordinatės yra žinomos (*Koordinatės gali būti generuojamos atsitiktinai, negali būti kelios parduotuvės toje pačioje vietoje*). Planuojama pastatyti dar **m** $(\boldsymbol{m} \ge \mathbf{3})$ šio tinklo parduotuvių. Parduotuvės pastatymo kaina (vietos netinkamumas) vertinama pagal atstumus iki kitų parduotuvių ir poziciją (koordinates). Reikia parinkti naujų parduotuvių vietas (koordinates) taip, kad parduotuvių pastatymo kainų suma būtų kuo mažesnė (naujos parduotuvės gali būti statomos ir už miesto ribos).

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:

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

Bus panaudota duomenų lygiagretumo programavimo priemonė C# parallel LINQ, kuri gijas valdo automatiškai, tačiau galima apibrėžti paraleliškumą, kiek gijų naudos paraleliškumas. Tam kad galėčiau naudoti PLINQ reikia naudoti AsParallel metodą.

2. Užduoties analizė ir sprendimo metodas

Užduočiai atlikti reikia sudaryti ir išspręsti tikslo funkciją pagal kurią gausime naujų parduotuvių pastatymo kaštus ir jų koordinates. Šios užduoties sėkmė priklauso nuo to, kaip toli yra kitos parduotuvės ir ar nauja parduotuvė yra miesto ar užmiesčio teritorijoje.
Parduotuvių optimizavimui panaudosime duotas formules:

$$(-10 \le \boldsymbol{x} \le 10,\ -10 \le \boldsymbol{y} \le 10).$$

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

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

Čia:
X_1, Y_1, X_2, Y_2 – Analizuojamų parduotuvių koordinatės

Sprendimui naudoju gradiento greičiausio nusileidimo metodą.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

public class Store
{
    public double X { get; set; }
    public double Y { get; set; }
}

class Program
{
    static Random random = new Random();
    static int parallelThreads = 5;


    // Atstumo tarp dvieju parduotuviu kainos apskaiciavimo metodas
    static double DistanceCost(Store store1, Store store2)
    {
        return Math.Exp(-0.3 * ((store1.X - store2.X) * (store1.X -
store2.X) + (store1.Y - store2.Y) * (store1.Y - store2.Y)));
    }

    // Parduotuves vietos kainos skaiciavimo metodas
    static double PlacementCost(Store newStore)
    {
        return (Math.Pow(newStore.X, 4) + Math.Pow(newStore.Y, 4)) /
1000 + (Math.Sin(newStore.X) + Math.Cos(newStore.Y)) / 5 + 0.4;
    }

    // Apskaiciuojama visa kaina sudejus ir naujas ir senas
parduotuves. Panaudojamas AsParallel() lygiagretinimui.
    static double TotalCost(List<Store> existingStores, List<Store>
newStores)
    {
        double totalCost = 0;

        // Apskaiciuojama kaina pridedant naujas parduotuves kartu su
senomis parduotuvemis
        totalCost +=
newStores.AsParallel().WithDegreeOfParallelism(parallelThreads).Selec
tMany(newStore => existingStores, (newStore, existingStore) =>
DistanceCost(newStore, existingStore)).Sum();

        // Pridedama nauju parduotuviu kaina
        totalCost +=
newStores.AsParallel().WithDegreeOfParallelism(parallelThreads).Selec
t(newStore => PlacementCost(newStore)).Sum();

        return totalCost;
    }

    static void Main()
    {

        Console.Write("Iveskite kiek parduotuviu norite tureti: ");
        int n = Convert.ToInt32(Console.ReadLine());
        Console.Write("Iveskite kiek parduotuviu norite prideti: ");
        int m = Convert.ToInt32(Console.ReadLine());
```

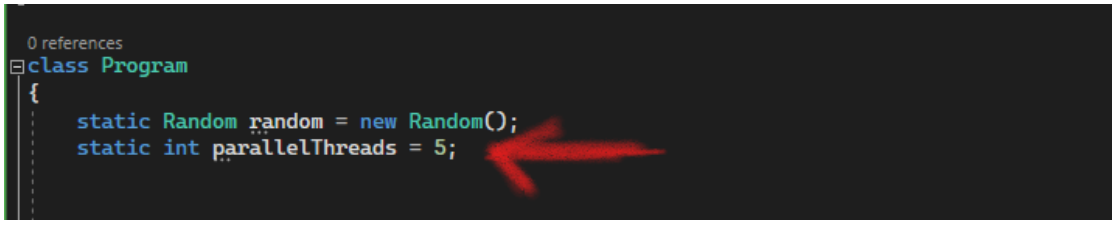
```
        Console.WriteLine("----------------------------------------
----------------------------------------");
        // Sugeneruojamos atitiktines pradines parduotuviu lokacijos
        //int n = 3; // parduotuviu kiekis
        List<Store> existingStores = Enumerable.Range(1, n).Select(_
=> new Store { X = random.NextDouble() * 20 - 10, Y =
random.NextDouble() * 20 - 10 }).ToList();

        // Pradines koordinates nauju parduotuviu.
        //int m = 3; // parduotuviu kiekis
        List<Store> newStores = Enumerable.Range(1, m).Select(_ =>
new Store { X = random.NextDouble() * 20 - 10, Y =
random.NextDouble() * 20 - 10 }).ToList();

        int maxIterations = 100;
        double learningRate = 0.1;

        for (int iteration = 0; iteration < maxIterations;
iteration++)
        {
            // Atnaujina nauju parduotuviu lokacijas naudojant
gradiento nusileidima
            newStores =
newStores.AsParallel().WithDegreeOfParallelism(parallelThreads).Selec
t(newStore =>
            {
                //Apskaiciuoja gradientus
                (double gradientX, double gradientY) =
CalculateGradient(existingStores, newStores, newStore);

                //Atnaujina koordinates pagal gradiento nusileidima
                newStore.X -= learningRate * gradientX;
                newStore.Y -= learningRate * gradientY;

                // apriboja koordinates gautas ir gradientu tarp -10
ir 10
                newStore.X = Math.Max(-10, Math.Min(10, newStore.X));
                newStore.Y = Math.Max(-10, Math.Min(10, newStore.Y));

                return newStore;
            }).ToList();

            // Isveda iteracija ir visa kaina
            Console.WriteLine($"Iteration {iteration + 1}, Total
Cost: {TotalCost(existingStores, newStores)}");
        }
        Console.WriteLine("----------------------------------------
----------------------------------------");


        Console.WriteLine("Pradiniu parduotuviu lokacijos:");
        foreach (var store in existingStores)
        {
            Console.WriteLine($"X: {store.X}, Y: {store.Y}");
            Console.WriteLine("Parduotuves kaina: " +
PlacementCost(store));
        }
        Console.WriteLine("----------------------------------------
----------------------------------------");
        Console.WriteLine("Nauju parduotuviu lokacijos:");
        foreach (var store in newStores)
        {
```

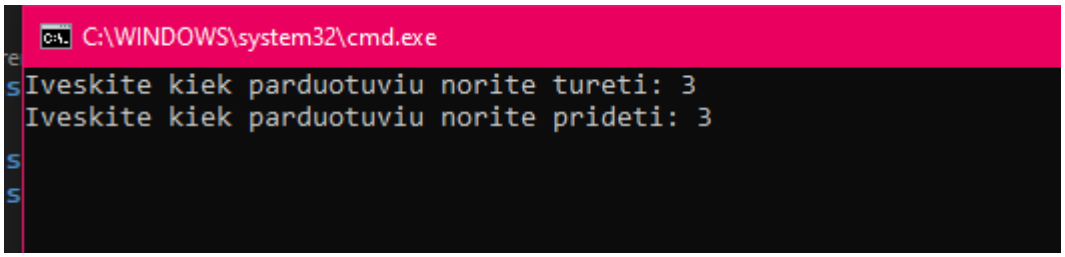
```
            Console.WriteLine($"X: {store.X}, Y: {store.Y}");
            Console.WriteLine("Parduotuves kaina: " +
PlacementCost(store));
        }
    }
    static (double, double) CalculateGradient(List<Store>
existingStores, List<Store> newStores, Store currentStore, double
epsilon = 1e-6)
    {
        //issisaugau pradines koordinates
        double originalX = currentStore.X;
        double originalY = currentStore.Y;

        //apsiskaiciuoju pradine kaina parduotuves
        double originalCost = TotalCost(existingStores, newStores);

        //Pakeiciu koordinates reiksme sudejus su epsilon
        currentStore.X = originalX + epsilon;
        double perturbedXCost = TotalCost(existingStores, newStores);

        // Atstatau x koordinate
        currentStore.X = originalX;

        //Pakeiciu koordinates reiksme sudejus su epsilon
        currentStore.Y = originalY + epsilon;
        double perturbedYCost = TotalCost(existingStores, newStores);

        // Atstatau y koordinate
        currentStore.Y = originalY;

        // Apskaiciuojama gradientu skaitine aproksimacija
        double gradientX = (perturbedXCost - originalCost) / epsilon;
        double gradientY = (perturbedYCost - originalCost) / epsilon;

        return (gradientX, gradientY);
    }
}
```

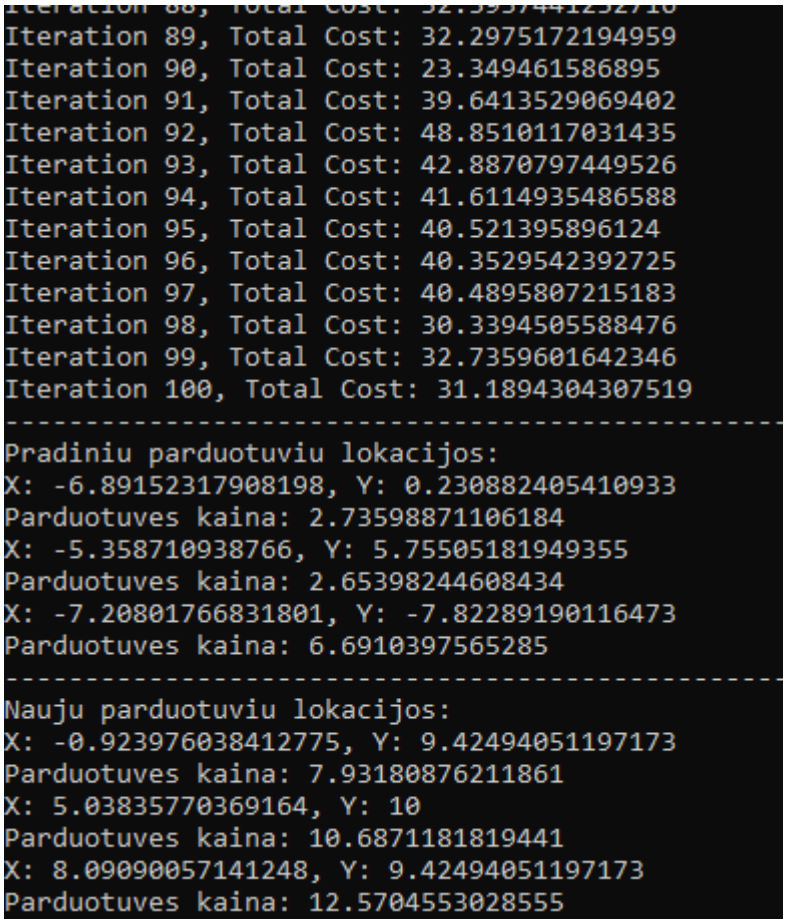
Skaičiuojant kiekvienos parduotuvės kainą ir gradientą yra naudojamas AsParallel().WithDegreeOfParallelism() metodas sulygiagretinti skaičiavimus ir suteikti tam tikrą kiekį gijų AsParallel() metodui.

3. Testavimas ir programos vykdymo instrukcija

Testuoti programą ganėtinai paprasta. Prieš paleidžiant, kode reikia pakeisti norimą gijų kiekį, o visų kitų kintamųjų programa paprašo paleidus ją.







Baigus skaičiavimus išveda kiekvienos iteracijos parduotuvių kainų sumas, iteracijų skaičių, pradinių parduotuvių informaciją ir naujai sukurtų parduotuvių informaciją.

4. Vykdymo laiko tyrimas

Kiekvienam pirmos dalies testui bus gijų kiekis apribotas iki 5.
Pirmas testas bus su 1 pradine parduotuve ir 1 nauja parduotuve:

```
Iteration 98, Total Cost: 0.279698578580411
Iteration 99, Total Cost: 0.271353766186389
Iteration 100, Total Cost: 0.263303195730489
-----------------------------------------------
Uztruko laiko 64ms
-----------------------------------------------
Pradiniu parduotuviu lokacijos:
X: 1.53625515826803, Y: 1.07380748310769
Parduotuves kaina: 0.702136399138426
-----------------------------------------------
Nauju parduotuviu lokacijos:
X: -2.74284584545485, Y: -3.0161533348557
Parduotuves kaina: 0.263275975374096
Press any key to continue . . .
```

Antras testas bus su 3 pradinėm parduotuvėm ir 3 naujomis parduotuvėmis.

```
Iteration 97, Total Cost: 39.1589920824915
Iteration 98, Total Cost: 32.6396650119749
Iteration 99, Total Cost: 36.0700636573918
Iteration 100, Total Cost: 32.4213360606554
-----------------------------------------------------
Uztruko laiko 70ms
-----------------------------------------------------
Pradiniu parduotuviu lokacijos:
X: 3.20777598452185, Y: 3.00253645191087
Parduotuves kaina: 0.375858363140655
X: -4.09676084951347, Y: -6.06589759516804
Parduotuves kaina: 2.39414389131655
X: -6.28373163579206, Y: 5.74301634716942
Parduotuves kaina: 3.21833063133355
-----------------------------------------------------
Nauju parduotuviu lokacijos:
X: -8.74859660979802, Y: 7.98069914831444
Parduotuves kaina: 10.1642317719509
X: -1.86836884343734, Y: 10
Parduotuves kaina: 10.0531611849189
X: 6.58202286170194, Y: 10
Parduotuves kaina: 12.1679520813064
Press any key to continue . . .
```

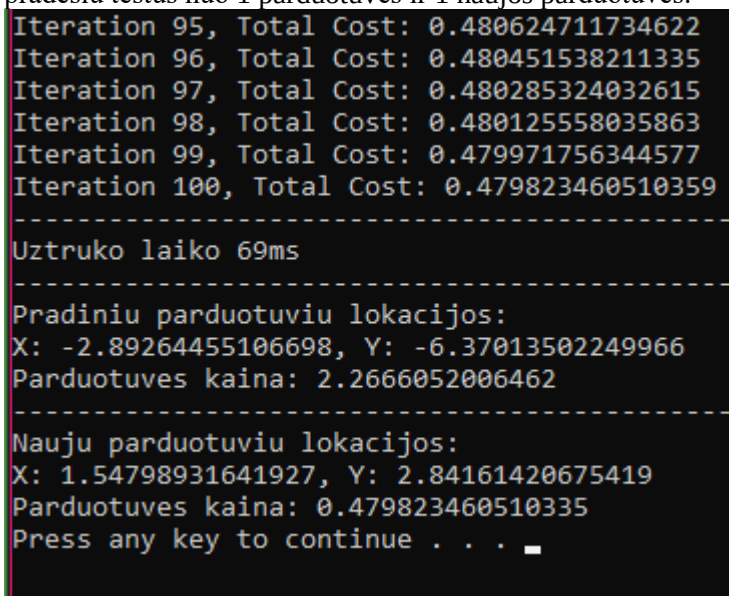
Trečias testas bus atliktas su 3 pradinėm parduotuvėm ir 20 naujų parduotuvių.

```
Iteration 99, Total Cost: 362.006267337228
Iteration 100, Total Cost: 376.785395581119
-----------------------------------------------------------
Uztruko laiko 127ms
-----------------------------------------------------------
Pradiniu parduotuviu lokacijos:
X: 4.3993364714083, Y: 0.0492915185397909
Parduotuves kaina: 0.784061064715725
X: -0.987119903316312, Y: -1.03816020816479
Parduotuves kaina: 0.336783830976576
X: -1.72000966580585, Y: -5.1321802172494
Parduotuves kaina: 0.986245969824946
-----------------------------------------------------------
Nauju parduotuviu lokacijos:
X: 9.05013186776159, Y: -9.43583677008974
Parduotuves kaina: 14.9088453481389
X: -10, Y: -10
Parduotuves kaina: 20.3409899163626
X: -10, Y: -10
Parduotuves kaina: 20.3409899163626
X: -10, Y: -10
Parduotuves kaina: 20.3409899163626
X: 6.93148460551129, Y: 10
Parduotuves kaina: 12.6613200568332
X: -9.96843307599192, Y: 10
Parduotuves kaina: 20.2099681382075
X: -10, Y: -10
Parduotuves kaina: 20.3409899163626
X: 9.33970181018594, Y: -10
Parduotuves kaina: 17.8582585227385
X: 10, Y: 10
Parduotuves kaina: 20.1233814720068
X: -10, Y: -10
Parduotuves kaina: 20.3409899163626
X: -6.56797819511667, Y: 10
Parduotuves kaina: 12.0369101250674
X: -10, Y: -10
Parduotuves kaina: 20.3409899163626
X: 10, Y: 10
Parduotuves kaina: 20.1233814720068
X: -9.70084033407147, Y: 8.82342858086304
Parduotuves kaina: 15.2066683384709
X: -10, Y: -10
Parduotuves kaina: 20.3409899163626
X: 10, Y: -10
Parduotuves kaina: 20.1233814720068
X: 10, Y: 10
Parduotuves kaina: 20.1233814720068
X: -10, Y: -10
Parduotuves kaina: 20.3409899163626
X: -10, Y: -10
Parduotuves kaina: 20.3409899163626
X: -10, Y: -10
Parduotuves kaina: 20.3409899163626
Press any key to continue . . .
```

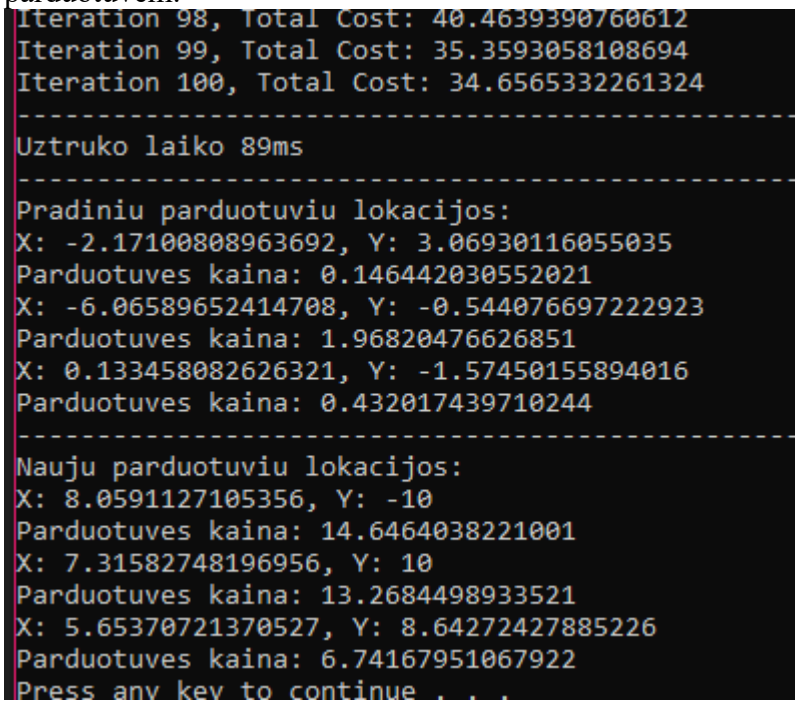
Ketvirtam testui naudosiu 20 senų parduotuvių ir 20 naujų parduotuvių:

```
----------------------------------------------------------------
Uztruko laiko 155ms
----------------------------------------------------------------
Pradiniu parduotuviu lokacijos:
X: 7.19202184918896, Y: -3.68835297119261
Parduotuves kaina: 3.24747660985938
X: -8.40039490647632, Y: 2.31269479371267
Parduotuves kaina: 5.10224049587627
X: 1.04748546660295, Y: -1.50975779234886
Parduotuves kaina: 0.591833418676707
X: 9.83741536728918, Y: 1.95964712275176
Parduotuves kaina: 9.62406737918482
X: 7.9732909975449, Y: 0.700940457499094
Parduotuves kaina: 4.79324009436692
X: 2.55846175018626, Y: 6.99612030619575
Parduotuves kaina: 3.09994514176327
X: 3.44085081175009, Y: -2.98156512574366
Parduotuves kaina: 0.362793125622898
X: 5.93136700612091, Y: -0.866236421682983
Parduotuves kaina: 1.69889252689108
X: 4.6520622515362, Y: 4.60109243849343
Parduotuves kaina: 1.09468397558172
X: -1.31711065364867, Y: 5.34548637240449
Parduotuves kaina: 1.14422591498062
X: 9.95285038834105, Y: -3.76509547874569
Parduotuves kaina: 10.1505471077715
X: 9.80645994646775, Y: -1.68160552702919
Parduotuves kaina: 9.55940939025543
X: 1.53616847076275, Y: -9.24598716164287
Parduotuves kaina: 7.71688270883137
X: -8.3013193301397, Y: 1.17883328868953
Parduotuves kaina: 5.0468617227641
X: 5.22598148101288, Y: -3.72620704291677
Parduotuves kaina: 0.997684930307892
X: -1.28545024957762, Y: -5.62370647472502
Parduotuves kaina: 1.36908843012745
X: -6.38950751926261, Y: 3.14222821646474
Parduotuves kaina: 1.94301004292827
X: -6.09059550617384, Y: -0.749581358744567
Parduotuves kaina: 1.96105608480373
X: -1.13593773503599, Y: -2.97721489936915
Parduotuves kaina: 0.101542118203095
X: 5.56007043251771, Y: -5.47798101579676
Parduotuves kaina: 2.26244386081233
----------------------------------------------------------------
Nauju parduotuviu lokacijos:
X: 3.85337472972424, Y: 10
```

Antrai testų daliai, pašalinsiu `WithDegreeOfParallelism` metodą. Be šio metodo, teoriškai programa turėtų visas įmanomas gijas panaudoti užduoties atlikimui. Taip pat pradėsiu testus nuo 1 parduotuvės ir 1 naujos parduotuvės.



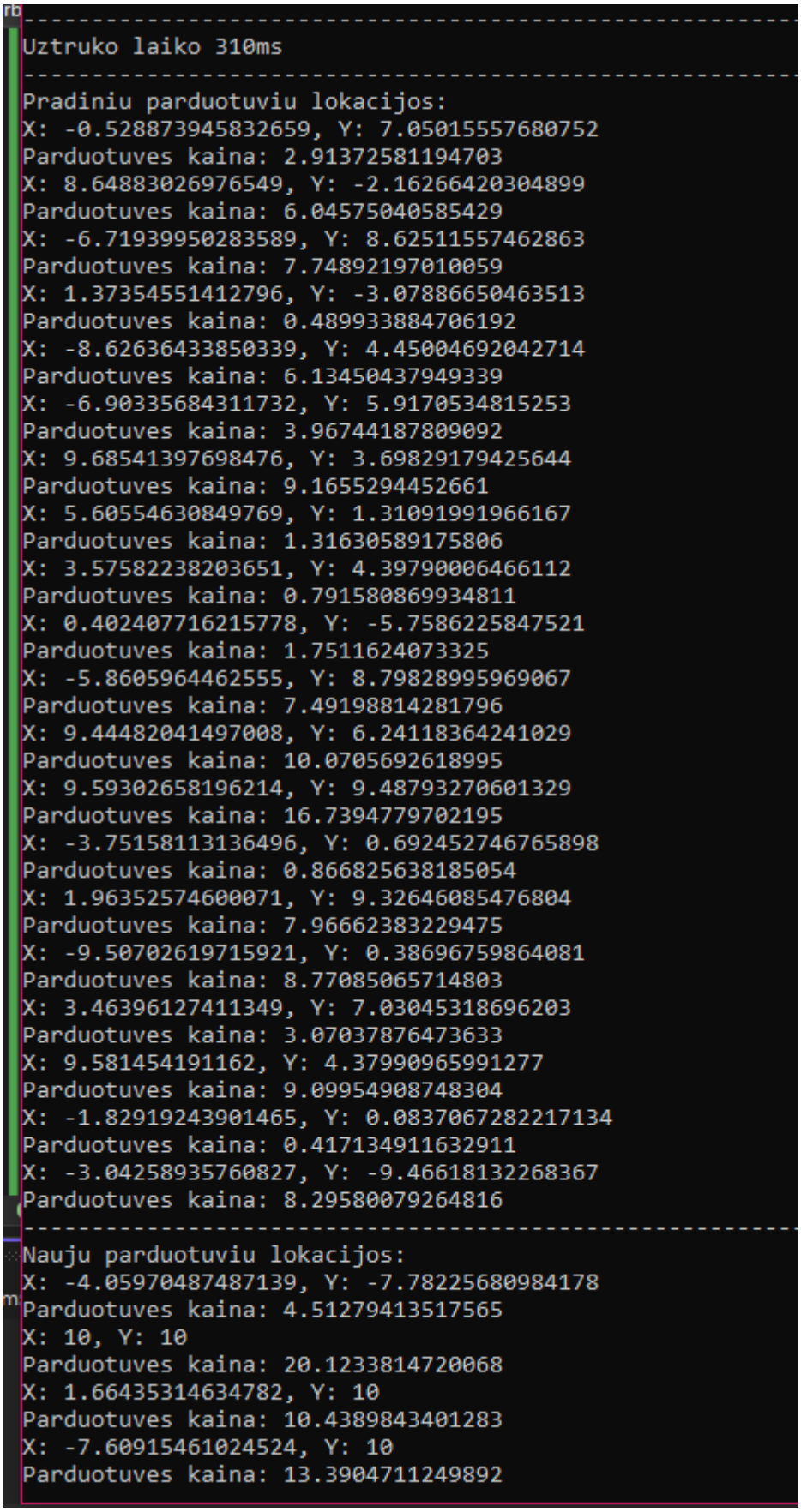
Antrasis testas bus taip pat toks pat su 3 senom parduotuvėm ir 3 naujom parduotuvėm.



Trečiasis testas bus su 3 senom parduotuvėm ir 20 naujų parduotuvių.

```
Iteration 99, Total Cost: 240.678523330701
Iteration 100, Total Cost: 260.286297813524
--------------------------------------------------
Uztruko laiko 236ms
--------------------------------------------------
Pradiniu parduotuviu lokacijos:
X: -6.28093077627985, Y: 9.12998134229797
Parduotuves kaina: 8.71370434795214
X: -2.35821239294401, Y: 3.77028816089513
Parduotuves kaina: 0.330100093922285
X: 5.031065398376, Y: -2.91204499682041
Parduotuves kaina: 0.727904489279474
--------------------------------------------------
Nauju parduotuviu lokacijos:
X: -8.13663756924598, Y: 10
Parduotuves kaina: 14.4232070161105
X: -10, Y: 4.37066679490727
Parduotuves kaina: 10.8066964090508
X: -7.33131756158855, Y: -10
Parduotuves kaina: 12.9477580697627
X: -9.85156265154274, Y: -10
Parduotuves kaina: 19.7343153583765
X: 3.70069645578951, Y: 10
Parduotuves kaina: 10.3136576090243
X: 10, Y: 10
Parduotuves kaina: 20.1233814720068
X: -5.39515445405868, Y: 10
Parduotuves kaina: 11.2346095708919
X: 10, Y: 10
Parduotuves kaina: 20.1233814720068
X: 4.37210948087795, Y: 10
Parduotuves kaina: 10.4090496838433
X: 6.16525277303936, Y: 10
Parduotuves kaina: 11.653439903564
X: 3.32612739522119, Y: 10
Parduotuves kaina: 10.317880555774
X: 8.2811997455201, Y: 10
Parduotuves kaina: 15.1171890879199
X: 3.55561075162768, Y: 10
Parduotuves kaina: 10.3115569218762
X: 10, Y: 10
Parduotuves kaina: 20.1233814720068
X: -2.61353287134l7, Y: 10
Parduotuves kaina: 10.1780704916553
X: 4.76405375276045, Y: -10
Parduotuves kaina: 10.5475704244917
X: 0.727280507485375, Y: 10
Parduotuves kaina: 10.3654336035336
X: -0.754552562275421, Y: 10
Parduotuves kaina: 10.095517304335
X: -0.701149147962497, Y: 10
Parduotuves kaina: 10.1034081389969
X: 1.32355163453667, Y: -10
Parduotuves kaina: 10.4291725420452
Press any key to continue . . .
```

Ketvirtas testas bus su 20 senų parduotuvių ir 20 naujų parduotuvių.



Pagal testavimo rezultatus manau, kad galiu skirti išvadą, kad davus gijų ribojimą, duomenys būna apdorojami optimaliau ir skaičiavimai atliekami greičiau, tačiau taip pat reikia nepamiršti įvertinti duomenų kiekį. Kuo mažiau duomenų yra paduodama apdoroti, tuo sprendimo laikas labiau susivienodina, o kai daugėja duomenų, apribojus paralelizmą darbas atliekamas vis greičiau.

5. Išvados

Atliekant inžinerinį projektą buvo išbandytos C# parallel LINQ PLINQ lygiagretinimo priemonės. Sužinojau, kad su šiomis priemonėmis labai paprasta išlygiagretinti kodą, jeigu kode yra naudojami enumeratoriai. Taip pat išsiaiškinau kad ne visada didelis kiekis gijų yra daugiau.