VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Magistro baigiamasis darbas

OPTIMALIŲ MARŠRUTŲ PAIEŠKOS ALGORITMAI

Atliko: magistro 2 kurso, 10 grupės studentas
Karolis Šarapnickis

Darbo vadovas:
doc. Tadas Meškauskas

Vilnius
2015

Turinys

1. Įvadas

Su optimalių maršrutų paieška kiekvienas žmogus daugiau ar mažiau susiduria kasdien. Praeituose moksliniuose darbuose [Šar15] buvo apžvelgti heuristiniai algoritmai, naudojami ieškant optimalių maršrutų. Algoritmai, bei jų modifikacijos buvo lyginami tarpusavyje sprendžiant keliaujančio pirklio problemą.

Yra nemažai mokslinių darbų sprendžiančių keliaujančio pirklio problemą naudojant heuristinius algoritmus, per paskutinius 10 metų padaugėjo mokslinių darbų, kuriuose yra naudojami modifikuoti ir sujungti tokie algoritmai kartu. Tačiau didžioji dalis tokių darbų analizuoja pilnus grafus (kiekvienas miestas turi tiesioginį kelią su bet kuriuo kitu miestu), o realiame gyvenime praktiškai nėra sausumos žemėlapių, kuriuose kiekvienas miestas būtų tiesiogiai susietais keliais su visais kitais miestais. Kitaip tariant ne visi žemėlapiai turi pilną Hamiltono ciklą, todėl daugelis mokslinių darbų negali būti tiesiogiai pritaikyti praktiniams uždaviniams spręsti.

Šio magistro baigiamojo darbo tikslas yra įsisavinus naujausias žinias apie optimalių maršrutų algoritmus, sprendžiant tradicinę keliaujančio pirklio problemą, pritaikyti jas sprendžiant praktiškesnę maršruto paieškos problemą dideliuose grafuose ir išanalizuoti parametrų įtaką hibridinio algoritmo rezultatams.

2. Optimalaus maršruto, aplankančio visus paskirties taškus algoritmai

2.1. Skruzdėlių kolonijos sistemos algoritmas

Skruzdėlių kolonijos algoritmas (ACS) (angl. *ant colony system*) yra heuristinis, gamtos įkvėptas algoritmas. Algoritmas yra paremtas skruzdėlių elgesiu gamtoje joms keliaujant miško takais.

Skruzdėlių kolonijos algoritmo veikimo principas [YuHuZha09]:

1. Inicializacija. Nustatomi ACS parametrai, o skruzdėlės yra pastatomos ant atsitiktinių miestų.

2. Kelio sudarymas. Kiekviena skruzdėlė sudaro nuosavą kelią nuolat pritaikydama pseudo-atsitiktinę proporcingumo taisyklę. Kai skruzdė *k* yra mieste *r*, ji pasirenka miestą *s* vadovaudamasi taisykle:

$$s=\begin{cases} arg\,max_{u\in J_k(r)}\{[\tau(r,u)]\cdot[\eta(r,u)]^{\beta}\}, & \text{jei} \quad q\le q_0 \\ S, & \text{kitu atveju} \end{cases} \tag{1}$$

kur $\tau(r,u)$ yra feromono lygis tarp miesto *r* ir *u*, $\eta(r,u)$ yra atstumo tarp miesto *r* ir *u* inversija, $J_k(r)$ yra rinkinys skruzdėlės *k* dar neaplankytų miestų, $\beta(\beta>0)$ yra heuristinis faktorius, kuris nulemia reliatyvę svarbą feromono ir atstumo, *q* yra atsitiktinis skaičius rėžiuose $[0,1]$, $q_0(0\le q_0\le 1)$ yra parametras, *S* yra atsitiktinis kintamasis parinktas pagal tikimybės pasiskirstymą:

$$p_k(r,s)=\begin{cases} \dfrac{[\tau(r,s)]\cdot[\eta(r,s)]^{\beta}}{\sum_{u\in J_k(r)}[\tau(r,u)]\cdot[\eta(r,u)]^{\beta}}, & \text{jei} \quad s\in J_k(r) \\ 0, & \text{kitu atveju} \end{cases} \tag{2}$$

3. Ši procedūra kartojama tol, kol kiekviena skruzdėlė aplanko visus miestus ir tada grįžta į pradinį miestą.

4. Lokalus feromono atnaujinimas. Kelio sudarymo metu skruzdėlės naudoja lokalaus feromono atnaujinimo taisyklę, kad pakeistų feromono lygį ant kraštinės (r,s) , kurią praėjo kaip:

$$\tau(r,s) \leftarrow (1-\rho)\cdot\tau(r,s) + \rho\cdot\tau_0 \tag{3}$$

kur $\rho(0<\rho<1)$ yra lokalaus feromono nykimo koeficientas, τ_0 yra pradinis feromono lygis.

5. Globalus feromono atnaujinimas. Globalus feromono atnaujinimas įvyksta kai visos skruzdėlės baigia savo maršrutus. Tik kraštai (r,s) priklausantys geriausiam maršrutui yra atnaujinti pagal:

$$\tau(r,s) \leftarrow (1-\alpha)\cdot\tau(r,s) + \alpha\cdot\Delta\tau_k(r,s) \tag{4}$$

kur $\alpha(0<\alpha<1)$ yra globalaus feromono nykimo parametras. $\Delta\tau_k(r,s)$ vertė yra nustatoma pagal

$$\Delta\tau_k(r,s) = \begin{cases} \frac{1}{L_{ib}}, & \text{jei } (r,s) \text{ priklauso geriausiam maršrutui} \\ 0, & \text{kitu atveju} \end{cases} \tag{5}$$

kur L_{ib} yra trumpiausio maršruto maršruto ilgis.

Algoritmas kartojamas nuo 2 žingsnio tol kol yra pasiekiama algoritmo sustabdymo sąlyga [YuHuZha09].

ACS algoritmas geba spręsti keliaujančio pirklio problemą ir yra kur kas efektyvesnis nei pilnas perrinkimas, tačiau mokslinės publikacijos [CheChi11] [KatNar01] teigia, kad šio algoritmo hibridinimas su genetiniu algoritmu atneša geresnių rezultatų. Tuo įsitikinome ir praeituose mokslo tiriamuosiuose darbuose [Šar15].

2.2. Simuliuoto atkaitinimo algoritmas

Simuliuoto atkaitinimo (SA) (angl. *simulated annealing*) algoritmas yra heuristinis, gamtos įkvėptas algoritmas. Jis yra paremtas metalo kietinimo procesu, kai jį lėtai vėsinant yra randami absoliutūs minimumai.

1. Sugeneruojamas pradinis maršrutas π ir apskaičiuojamas jo ilgis $f(\pi)$;
2. Nustatoma pradinė temperatūra $T := T_0$;
3. Nustatomas iteracijos indeksas $t := 1$;
4. Kol nesustabdomas ciklas, vykdoma:
    1. Naudojant reprodukcijos operatorių yra sugeneruojamas naujas maršrutas π' ir apskaičiuojamas jo ilgis $f(\pi')$
    2. Nustatoma $\Delta f := f(\pi') - f(\pi)$
    3. Jei $\Delta f < 0$, tada π kelias pakeičiamas π' .
    4. Jei $e^{-\Delta f / T} < rand()$, tada π kelias pakeičiamas π' .
    5. Kitu atveju išmetamas π' kelias.
    6. Jei $t \bmod T_u = 0$, tada atnaujinama temperatūra: $T := \alpha T$
    7. Atnaujinamas iteracijos skaitiklis $t := t + 1$

1. pvz. Standartinė simuliuoto atkaitinimo procedūra [LiZhoGui11].

Standartinė simuliuoto atkaitinimo algoritmo struktūra pavaizduota 1 pav. Kiekvienos SA algoritmo iteracijos metu turimas kelias $\pi = (\pi_1 \pi_2 \cdots \pi_n)$. Algoritmo paaiškinimas [LiZhoGui11]:

- Inicializacija. Jos metu atsitiktinai sugeneruota maršrutas taip: visiems $i=1,2,\cdots,n$ yra rinkinys $\pi_i=i$; tada kiekvienam $i=1,2,\cdots,n-1$ atsitiktinai parenkamas $\pi_j\in\{\pi_i,\pi_{i+1},\cdots,\pi_n\}$ ir sukeičiamas π_i ir π_j .

- Ciklo sustabdymas. Kelio paieškos ciklas gali būti sustabdytas išsipildžius vienai iš dviejų sąlygų: pasiekus tam tikrą kartų skaičių, *maxt* , arba nerastas trumpesnis kelias tam tikrą skaičių kartų, *maxut* .

- Pasirinkimo procedūra. SA algoritmo pasirinkimo taisyklė yra pritaikoma 2 pav. 4.3 – 4.5 žingsniuose. Yra 3 situacijos kaip pasielgiama su naujai gautu maršrutu: 1) naujas kelias yra trumpesnis ir yra pasirenkamas; 2) naujas kelias yra prastesnis, bet yra pasirenkamas remiantis tikimybe kur $rand()$ funkcija gražina atsitiktinį skaičių rėžiuose $[0,1]$; 3) naujas kelias yra prastesnis, todėl yra išmetamas.

- Reprodukcijos operatorius (2 pav.). Standartinio SA algoritmo reprodukcijos operatoriai būna du: 1) parinktas sub-maršrutas yra paimamas priešinga puse; 2) parinktas sub-maršrutas yra įstatomas į kitą maršruto poziciją.

1. Atsitiktinai iš maršruto π yra parenkamas sub-maršrutas ir pažymimas

$$(\pi_{i_1}\cdots\pi_{i_k})$$

kur π_{i_1} yra sujungtas su π_{i_1-1} , π_{i_k} yra sujungtas su π_{i_k+1} ir $1<k<n-1$

2. Jei $rand()<0.5$

   Apsukamas sub-maršrutas ir gaunamas naujas maršrutas π' pavidalu:

$$(\pi_1\cdots\underline{\pi_{i_1-1}\pi_{i_k}}\cdots\underline{\pi_{i_1}\pi_{i_k+1}}\cdots\pi_n)$$

3. Jei $rand()\geq 0.5$

   Atsitiktinai parenkamas kraštas $\pi_j\pi_{j+1}$ iš galimų, ne sub-maršrute esančių, kraštų. Įterpiamas sub-maršrutas tarp π_j ir π_{j+1} ir gaunamas naujas maršrutas π' pavidalu:

$$(\pi_1\cdots\underline{\pi_{i_1-1}\pi_{i_k+1}}\cdots\underline{\pi_j\pi_{i_1}}\cdots\underline{\pi_{i_k}\pi_{i+1}}\cdots\pi_n)$$

2. pvz. Standartinė simuliuoto atkaitinimo reprodukcijos procedūra [LiZhoGui11].

SA algoritmo pagalba galima tikrai efektyviai ieškoti optimalių maršrutų keliaujančio pirklio problemoje, tačiau šio algoritmo panaudojimas kaip mutacijos operatoriumi genetiniuose algoritmuose duoda kur kas geresnių rezultatų [CheChi11] [Šar15].

2.3. Genetiniai algoritmai

Genetiniai algoritmai (GA) (angl. *genetic algorithm*) – vieni iš heuristinių algoritmų, įkvėptų gamtos. Jie yra paremti evoliuciniu gamtos modeliu, kai didėjant kartų skaičiui individai tampa vis tobulesni ir labiau prisitaikę prie aplinkos sąlygų. Genetiniam algoritmui yra svarbi fitneso funkcija, kurios pagalba galima nuspręsti individo tinkamumą.

Dėl keliaujančio pirklio problemos paprastos formuluotės yra bandoma įvairiausius algoritmus panaudoti sprendžiant šią problemą. Ne išimtis yra ir šie algoritmai, tačiau „grynieji“ genetiniai algoritmai, kurti J. H. Holland'o ir jo studentų Mičigano universitete 60-taisiais, 70-taisiais, nebuvo sugalvoti spręsti kombinatorinio optimizavimo problemas. Tais laikais genetiniai algoritmai dažniausiai buvo naudojami sprendžiant įvairias skaitines funkcija, todėl norint rasti keliaujančio pirklio problemos sprendimą, reikia atlikti tam tikrus genetinio algoritmo pakeitimus, kuriuos pradėjo taikyti jau 90-asiais metais [Pot96].

Genetinis algoritmas iš esmės operuoja su baigtine chromosomų arba bitų sekų populiacija. Paieškos mechanizmas susideda iš trijų skirtingų fazių: kiekvienos chromosomos tinkamumo (angl. *fitness*) įvertinimas, tėvinių chromosomų parinkimas ir mutacijos bei rekombinacijos operatorių pritaikymas tėvinėms chromosomoms. Naujos chromosomos, gautos pritaikius genetinius operatorius, dalyvauja tolimesnėje revoliucijos iteracijoje ir pati sistema tobulėja augant kartų skaičiui [Pot96]. Sistemos supaprastintas pseudo-kodas pavaizduotas 3 pavyzdyje.

1. Sukuriama pradinė P chromosomų populiacija (0 karta).
2. Įvertinamas kiekvienos chromosomos tinkamumas.
3. Pasirenkama P tėvų iš esamos populiacijos pasitelkiant proporcingumo taisykle (angl. *proportional selection*) (pasirinkimo tikimybė priklauso nuo tinkamumo vertės).
4. Dauginimuisi atsitiktinai pasirenkama pora chromosomų. Atliekama rekombinacijos operacija apkeičiant bitus pasirinktame taške, taip sukuriant vaikines chromosomas.
5. Kiekviena vaikinė chromosoma apdorojama mutacijos operatoriais ir grąžinama į populiaciją.
6. Kartojami 4 ir 5 žingsniai kol visos chromosomos būna parinktos ir paveiktos genetiniais operatoriais.
7. Sena chromosomų populiacija pakeičiama nauja.
8. Įvertinamas kiekvienos chromosomos tinkamumas.
9. Kartojama viskas nuo 3 žingsnio kol pasiekiamas atitinkamas kartų skaičius ar kitokia sąlyga.

3. pvz. Genetinio algoritmo pseudo-kodas [Pot96].

2.3.1. Parinkimo tikimybė

Genetinio algoritmo vykdymo metu yra parenkamos tėvinės chromosomos. Problema yra kokias chromosomas yra geriausia parinkti, kad iš jų gautos vaikinės chromosomos būtų geresnės. Yra nemažai metodų kaip yra parenkamos chromosomos atsižvelgiant į jų tinkamumo koeficientus ir kiekvienas metodas pasižymi tam tikrais privalumais ir trūkumais.

Vienas iš metodų parinkti tėvines chromosomas yra ruletės metodas. Ruletės parinkimo metodo principas:

1. Susumuojami visų populiacijos chromosomų tinkamumo koeficientai.
2. Sugeneruojamas atsitiktinis skaičius tarp 0 ir tinkamumo koeficientų sumos.
3. Atsitiktine tvarka sumuojami chromosomų tinkamumo koeficientai, kol suma yra lygi arba viršija 2 punkte gautą skaičių. Paskutinė sumuota chromosoma ir yra gražinama.

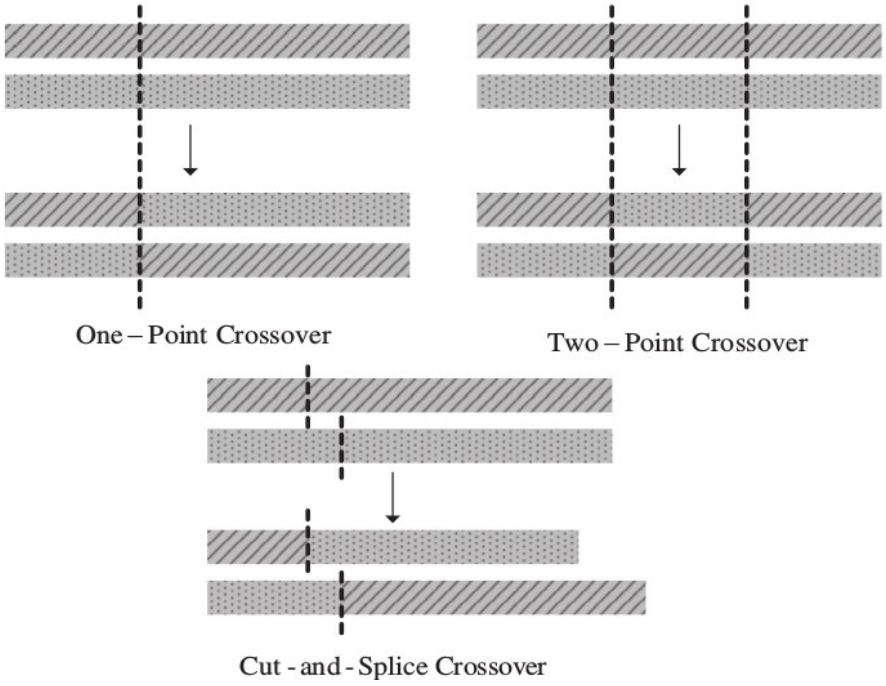
Šis metodas pasižymi vienos super-chromosomos trūkumu. Pavyzdžiui, esant tinkamumo koeficientams - [20, 10, 5, 847, 42, 1], dominuojanti chromosoma bus su koeficientu 847 ir praktiškai visada bus parinkta ruletės metodu. Todėl keičiantis kartoms populiacija praktiškai nesikeičia, kiekviena chromosoma būna panaši ir evoliucija toliau nebevyksta [Pot96].

Galima alternatyva ruletės metodui yra rango parinkimo metodas. Šis metodas labai panašus į ruletės, tačiau vietoj tinkamumo koeficientų yra sumuojami tinkamumo koeficientų rangai. Tačiau šis metodas taip pat nėra idealus, jis pasižymi lėta konvergencija, nes geriausios chromosomos ne taip stipriai skiriasi nuo prastesnių chromosomų [Pot96].

2.3.2. Rekombinacija

Rekombinacijos metu yra generuojamos naujos chromosomos tol, kol atnaujinama visa populiacija. Tam tikru parinkimo metodu iš chromosomų populiacijos yra parenkamos 2 tėvinės chromosomos. Atsitiktinai sugeneruojamas skaičius tarp 0 ir 1. Jei tas skaičius yra didesnis nei nurodytas rekombinacijos koeficientas *CR* (kur $CR \in (0,1]$), tada yra vykdoma rekombinacija. Jei rekombinacija yra įvykdoma, tada yra gaunamos 2 vaikinės chromosomos. Šis metodas yra vykdomas tol, kol yra atnaujinama visa populiacija [CheChi11].

Rekombinacijos metodų taip pat yra ne vienas (4 pav). Vienas iš metodų yra vieno taško (angl. *one point*) rekombinacija, kurio metu yra apkeičiamos bitų eilutės tarp tėvinių chromosomų. Yra parenkamas atsitiktinis skaičius nuo 1 iki $L-1$, kur L yra chromosomos ilgis. Tada chromosomos yra perskiriamos parinktame taške ir jų galai yra apkeičiami tarp tėvinių chromosomų, taip sukuriant 2 vaikines chromosomas [Pot96].



4. pvz. Vieno taško, dviejų taškų ir supjaustymo ir sujungimo rekombinacijos metodų [CheChi11].

Rekombinacijos tikimybė siejama su indeksu *CR*, kuris nulemia paieškos „agresyvumą“. Jei nėra pritaikomas rekombinacijos metodas, tada tėvinės chromosomas keliauja į naują populiaciją, kitu atveju tai daroma su naujai gautomis chromosomomis. Didelis *CR* indeksas sukuria daugiau naujų chromosomų, tačiau padidina tikimybę prarasti geresnes tėvines chromosomas [Pot96].

Keliaujančio pirklio problemos sprendimo metu įprastiniai vieno taško ir panašūs rekombinacijos metodai netinkami taikant juos tiesiogiai galimiems maršrutams. Šie operatoriai yra skirti bitų manipuliacijai, o juos pritaikius miestų sekoms yra nemaža tikimybė, kad gauti vaikai nebus validžios permutacijos. Pavyzdžiui turint dvi chromosomas 12564387 ir 14236578, ir atlikus vieno taško rekombinacijos operatorių antroje pozicijoje yra gaunami vaikai: 12236578, bei 14564387. Tokios chromosomos nėra validžios, nes kartojasi arba trūksta tam tikrų miestų. Šią problemą galima spręsti pakeitus kelio reprezentacinę išraišką, arba pasirinkus kitokį rekombinacijos metodą [Pot96].

2.3.2.1. Kelintinė išraiška

Viena iš galimų išraiškų yra kelintinė (angl. *ordinal*) (5 pav.). Kelintinę išraišką paveikus rekombinacijos operatoriumi visada yra gaunamos validžios vaikinės chromosomos, kurios yra sistemos perturbacijos. Atlikus vieno taško ar panašaus tipo rekombinacijos operaciją ant chromosomų, išreikštų kelintine išraiška, yra gaunamos vaikinės chromosomos, kurias galima iš kelintinės išraiška konvertuoti atgal į skaitinę [Pot96].

Kelio išraiška	Likę miestai	Kelintinė išraiška
1 2 5 3 4	1 2 3 4 5	1
1 2 5 3 4	2 3 4 5	1 1
1 2 5 3 4	3 4 5	1 1 3
1 2 5 3 4	3 4	1 1 3 1
1 2 5 3 4	4	1 1 3 1 1

5. pvz. Kelintinė išraiška.

2.3.2.2. Eilės (OX) rekombinacija

Viena iš alternatyvų tradiciniams rekombinacijos metodas (vieno taško ir t.t.) yra eilės (angl. *order crossover*) rekombinacija (OX). OX rekombinacija paremta modifikuotos rekombinacijos principu (6 pav.), tik vietoj 1 taško yra imami 2 taškai. Modifikuotos rekombinacijos metu vaikinė chromosoma yra sukuriama prie pirmosios chromosomos dalies pridedant antrosios chromosomos galą, eliminuojant dublikatus [Pot96].

Chromosoma 1 : **1 2** | 5 6 4 3 8 7
Chromosoma 2 : 1 4 | **2 3 6 5 7 8**

Vaikas 1 : 1 2 4 3 6 5 7 8

6. pvz. Modifikuota rekombinacija.

OX rekombinacija sukuria vaikines chromosomas padarydama 2 pjūvio taškus (7 pav.). Padalinus abi chromosomas į 3 dalis, į pirmą vaikinę chromosomą yra nukeliama 2-oji pirmosios chromosomos dalis. Tada iš antrosios chromosomos yra paimama pradžia ir pabaiga ir sujungiama su vaikinės vidurine dalimi. Panaikinus dublikatus yra gaunama nauja chromosoma [Pot96].

Chromosoma 1 : **1 2** | **5 6 4** | **3 8 7**
Chromosoma 2 : 1 4 | 2 3 6 | 5 7 8

Vaikas 1
1 žingsnis : - - **5 6 4** - - -
2 žingsnis : 2 3 **5 6 4** 7 8 1

7. pvz. OX rekombinacija.

2.3.3. Mutacija

Rekombinacijos metu gautų vaikinių chromosomų bitai, su labai maža tikimybe yra paveikiami mutacijos operatoriumi. Ši tikimybė apibrėžiama naudojant koeficientą *MR*, kur $MR \in (0,1]$. Jei atsitiktinai sugeneruotas skaičius nuo 0 iki 1 yra didesnis, nei *MR*, tada yra taikoma mutacijos operacija [CheChi11].

Tam tikroje vietoje pritaikius šį operatorių bitas iš 0 paverčiamas į 1, arba iš 1 į 0. Mutacijos operatoriaus tikslas yra į procesą įtraukti atsitiktinių perturbacijų tikimybę. Yra naudinga į genetinio

algoritmo veikimo procesą įtraukti įvairovės, kuri sukurti chromosomas, kurios nebūtų gražinamos į ankstesnes būsenas vien rekombinacijos metodais. Taip pat yra naudinga didinti mutacijos tikimybę augant kartų skaičiui, norint išlaikyti tinkamą įvairovės lygį [Pot96].

Sprendžiant keliaujančio pirklio problema genetinio algoritmo pagalba galima naudoti apkeitimo (angl. *swap*) mutacija. Apkeitimo mutacijos metu chromosomoje atsitiktinai parinkti du miestai yra apkeičiami vietomis [Pot96].

2.3.4. Elitizmas

Paprasto genetinio algoritmo veikimo metu, kiekviena nauja chromosomų karta pakeičia senąją. Tačiau yra ir kitų galimų būdų kaip užbaigti algoritmo kartą. Galima pakeisti tik dalį senosios kartos naujomis chromosomomis. Elitizmo metodas kiekvienos kartos generavimo pabaigoje išsaugo geriausią senosios kartos chromosomą [Pot96]. Taip panaikinama tikimybė, kad keičiantis kartoms gali būti sunaikinta labai tinkama chromosoma.

3. Praktinė dalis

Hibridiniu genetiniu algoritmu buvo sprendžiama modifikuota keliaujančio pirklio problema grafuose, kurie neturi pilno Hamiltono ciklo. Dėl pastarosios sąlygos keliaujančio pirklio problemos taisyklė, kad kiekvienas miestas turi būti aplankytas tik vieną kartą, buvo pakeista leidžiant kiekvieną miestą aplankyti daugiau negu vieną kartą. Tokios analizės krypties išvados yra plačiau pritaikomos sprendžiant praktinius optimaliausių maršrutų apieškos uždavinius.

Daugelis heuristinių algoritmų turėdami neribotą laiką galėtų rasti optimaliausią maršrutą. Tačiau skaičiavimo laikas yra ne ką mažiau svarbesnis matavimo vienetas už patį maršruto ilgį. Taip pat tokie algoritmai pasižymi įsisotinimu ties tam tikromis ribomis, ties kuriomis maršruto trumpėjimas labai sustoja. Todėl magistro baigiamojo darbo metu buvo ieškoma ne tik optimalių parametrų su kuriais būtų gaunamas geriausias vidutinis rezultatas, bet ir su kokiais parametrais maršrutas trumpėja greičiausiai.

Remiantis praeitais mokslo tiriamaisiais darbais [Šar15] genetiniame hibridiniame algoritme yra naudojama OX rekombinacija bei SA algoritmas kai mutacijos operatorius. Pradinė populiacija parenkama naudojantis 8 pav. pateikta ACS algoritmo modifikacija, kuri nusako tikimybę esant mieste *r* pasirinkti miestą *s*. Pradžioje mėginama parinkti miesto *r* kaimynus, ir tik tada iš likusių miestų. Jei (6) algoritmo gauta tikimybė yra didesnė nei $rand(1,0)$, tada miestas yra aplankomas.

$$p_k(r,s)=\begin{cases}\dfrac{\tau(r,s)}{\sum_{u\in J_k(r)}\tau(r,u)}, & \text{jei} \quad s\in J_k(r)\\ 0, & \text{kitu atveju}\end{cases} \tag{6}$$

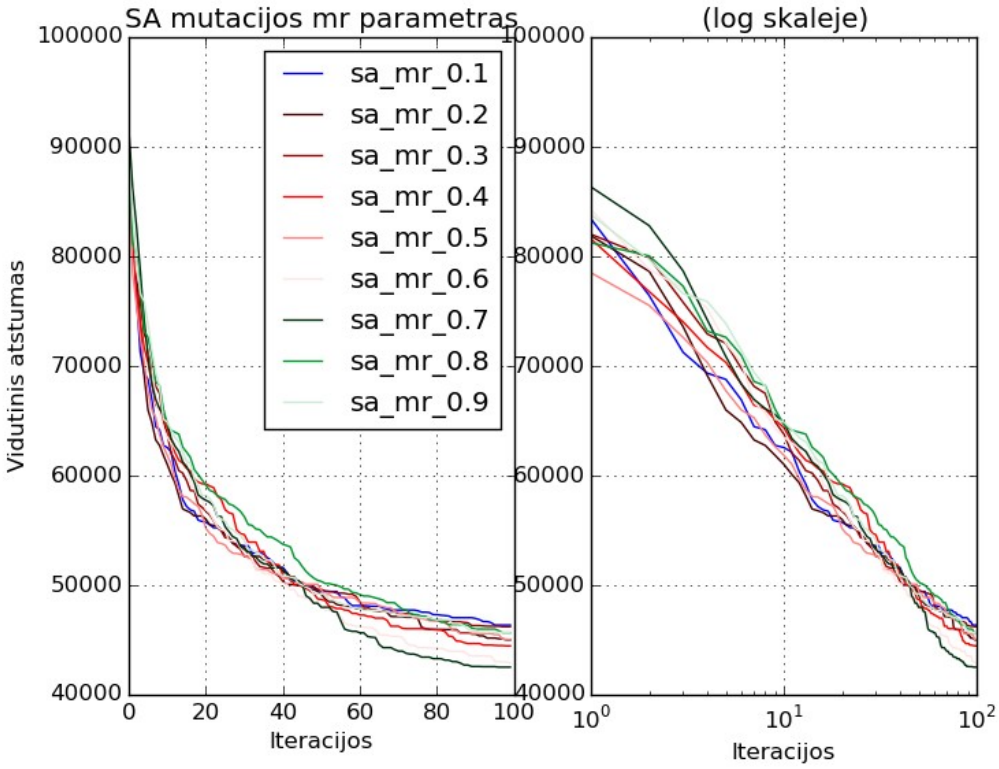
8. pvz. Hibridinio genetinio algoritmo pradinės populiacijos parinkimo algoritmas, paremtas ACS algoritmu, kur $J_k(r)$ yra rinkinys miestų, kuriuos dar reikia aplankyti skruzdėlei *k* iš miesto *r*, o $\tau(r,s)$ yra didžiausias įmanomas atstumas tarp grafo taškų koordinačių padalintas iš kelio ilgio nuo *r* iki *s* miesto.

Galutinė algoritmo implementacija turi 4 parametrus: chromosomų skaičių, iš jų (6) algoritmo generuotų chromosomų skaičius, rekombinacijos koeficientas *cr* ir mutacijos koeficientas *mr.* Pradžioje visi analizuoti grafai buvo sintetiniai – atsitiktinai generuotos viršūnės, o kraštinės nubrėžtos ranka.

Programinio kodo implementacija parašyta naudojant Python 2.7 programinę kalbą ir vykdytas Ubuntu operacinėje sistemoje.

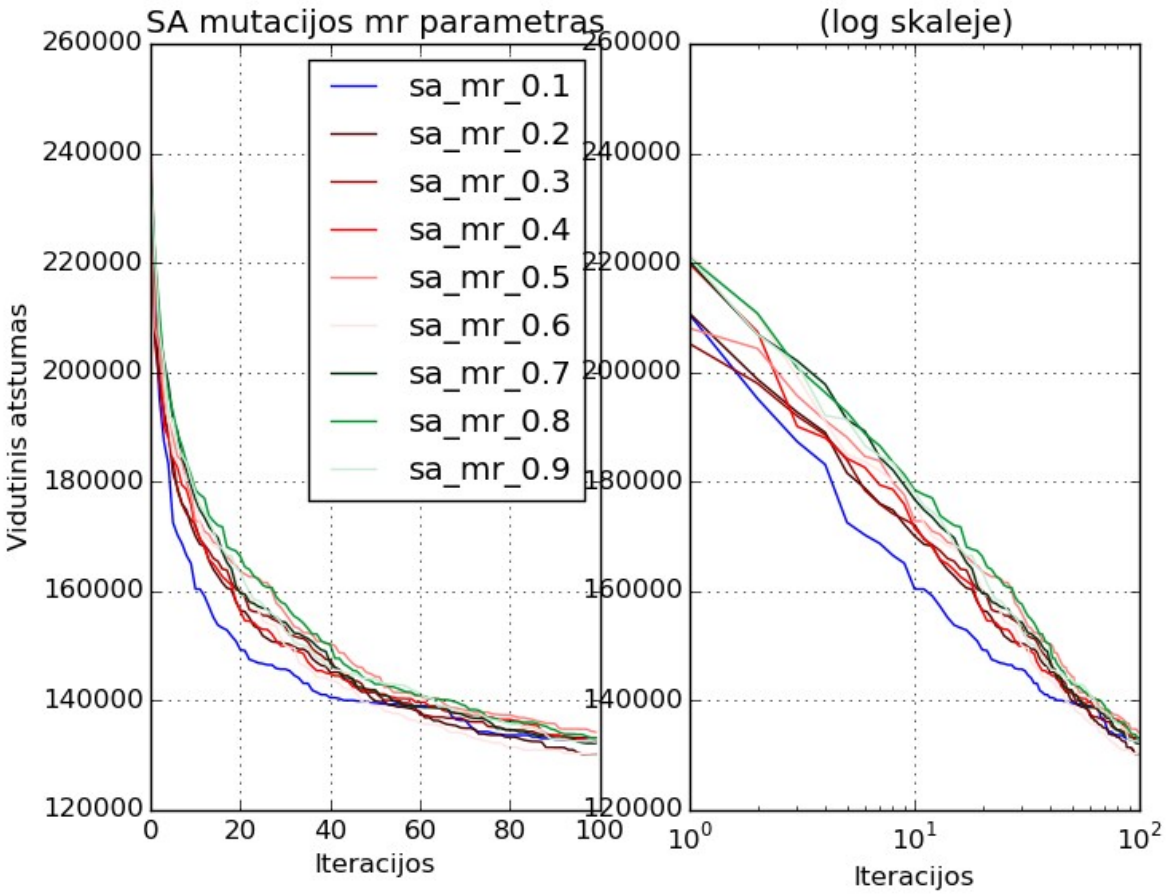
3.1. Mutacijos koeficientas

Iš pradžių, naudojant 3 skirtingus grafus (48, 66, 100 miestų dydžio), buvo siekiama išanalizuoti *mr* koeficiento įtaką algoritmui (9 pav., 10 pav.).



9. pvz. 48 miestų dydžio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: 40 chromosomų, atsitiktinės populiacijos generavimas, 0.3 rekombinacijos koef. su OX rekombinacija, SA mutacija, 10 matavimo pakartojimų.

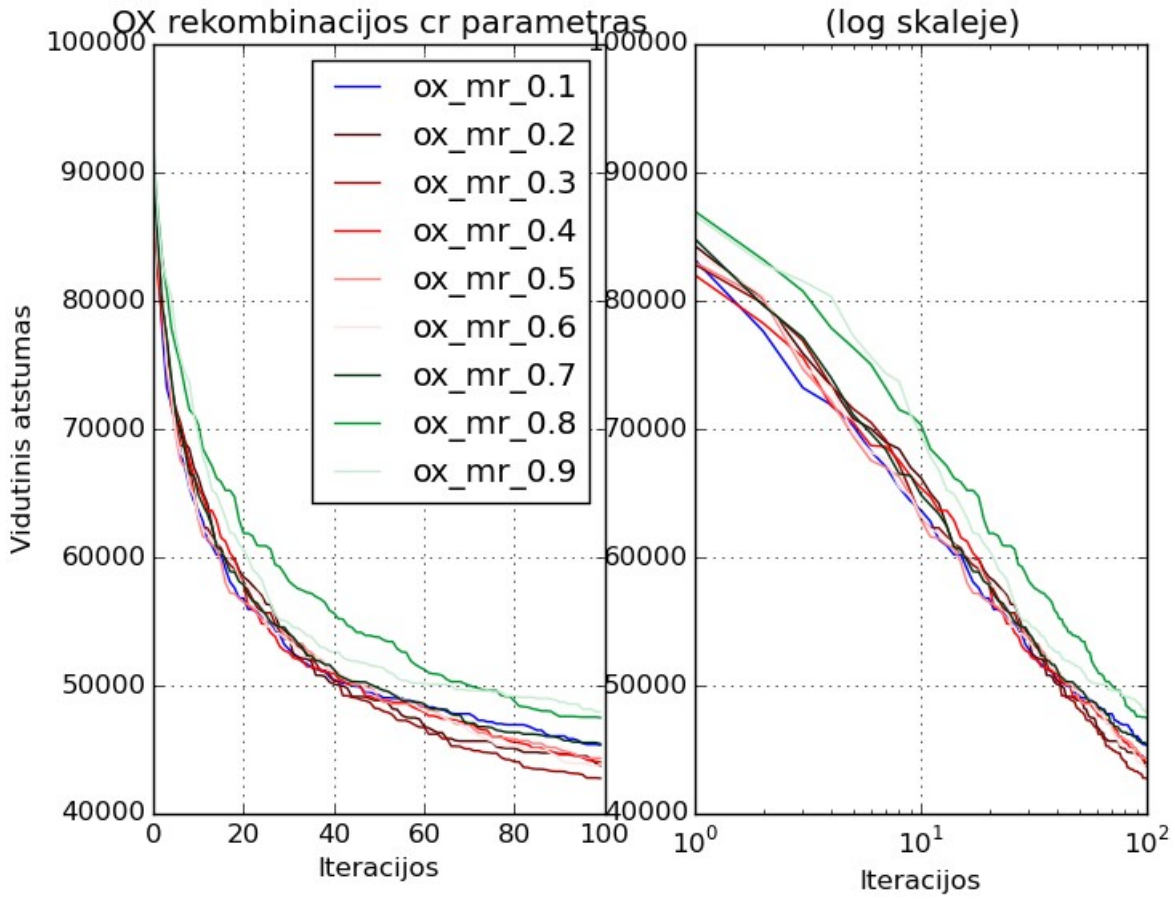
Iš 9 pav. ir 10 pav. bei kitų atliktų matavimų siekiant įžvelgti *mr* koeficiento tendencijas to padaryti nepavyko. Prie skirtingų grafų skirtingi *mr* koeficientai davė geriausius rezultatus.



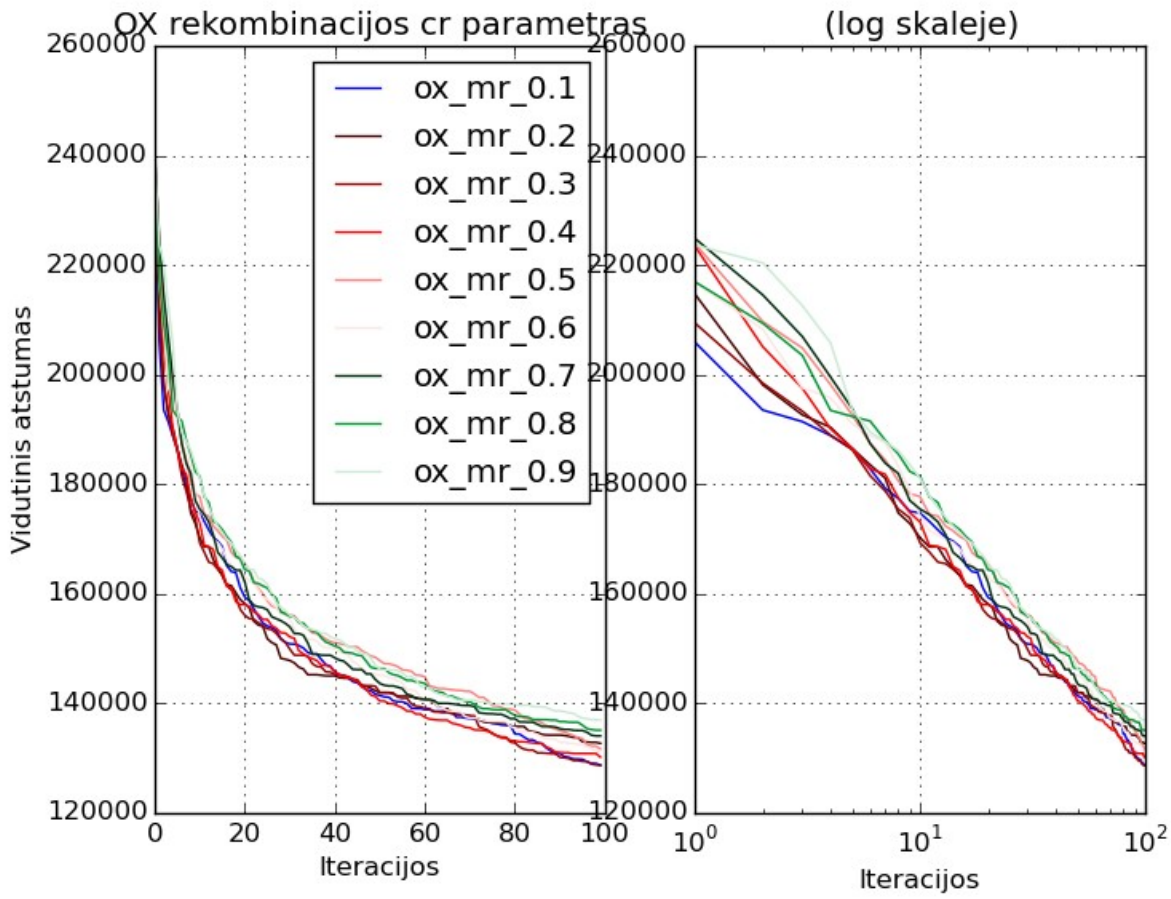
10. pvz. 100 miestų dydžio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: 80 chromosomų, atsitiktinės populiacijos generavimas, 0.3 rekombinacijos koef. su OX rekombinacija, SA mutacija, 10 matavimo pakartojimų.

3.2. Rekombinacijos koeficientas

Toliau buvo siekiama nustatyti prie kokių rekombinacijos *cr* koeficientų hibridinis algoritmas gautų geriausius rezultatus visuose grafuose (11 pav., 12 pav.).



11. pvz. 48 miestų dydžio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: 40 chromosomų, atsitiktinės populiacijos generavimas, OX rekombinacija, 0.3 mutacijos koef. su SA mutacija, 10 matavimo pakartojimų.

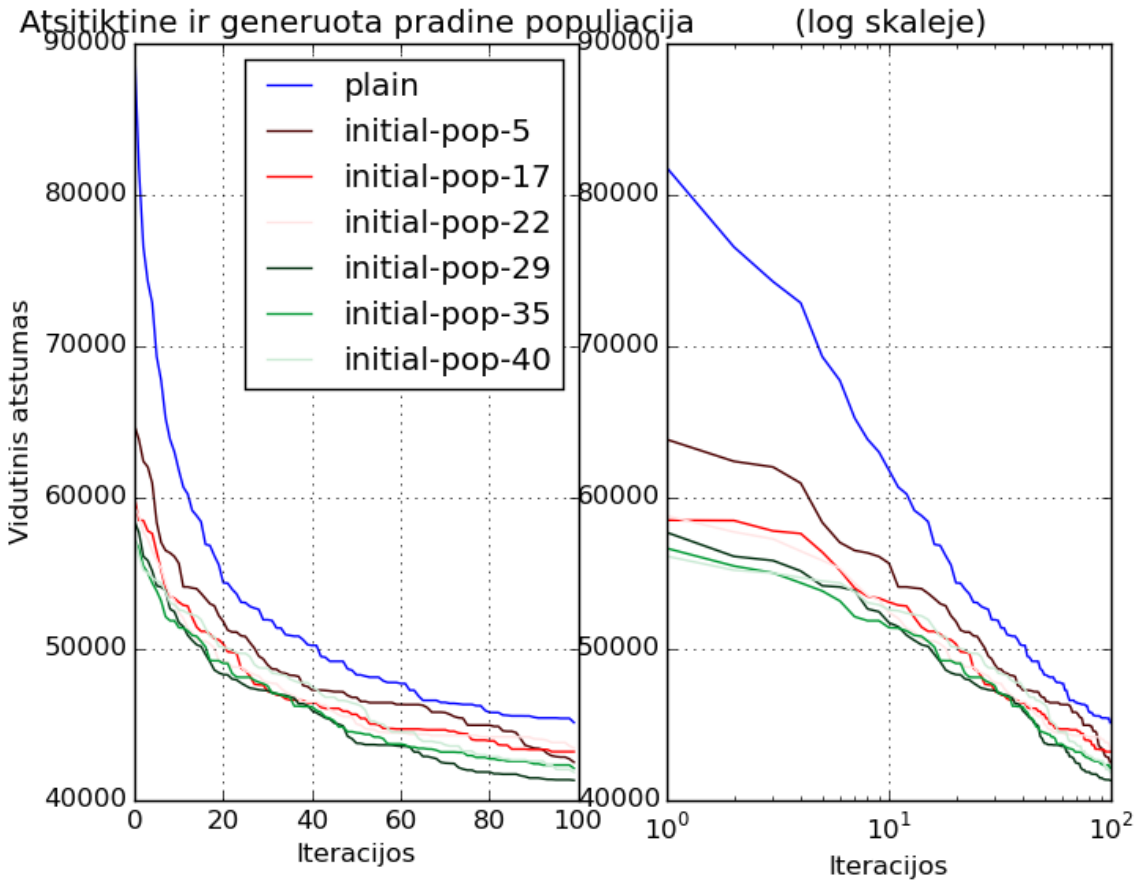


12. pvz. 100 miestų dydžio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: 80 chromosomų, atsitiktinės populiacijos generavimas, OX rekombinacija, 0.3 mutacijos koef. su SA mutacija, 10 matavimo pakartojimų.

Iš 11 pav. ir 12 pav. bei kitų matavimų ryškaus, dominuojančio bendru atveju mutacijos koeficiento įžvelgti nepavyko. Tik matyti kad prie mažesnių koeficientų, intervale (0,1 – 0,4), algoritmas duoda geresnius rezultatus nei kitais atvejais.

3.3. Pradinės populiacijos parinkimas

Sekantis žingsnis yra tikrinimas, ar yra duomenų koreliacija tarp skirtingų grafų keičiant pradinės populiacijos generavimo (6) algoritmu skaičius. 48 miestų dydžio optimalių maršrutų paieškos kreivės matomos 13 pav.



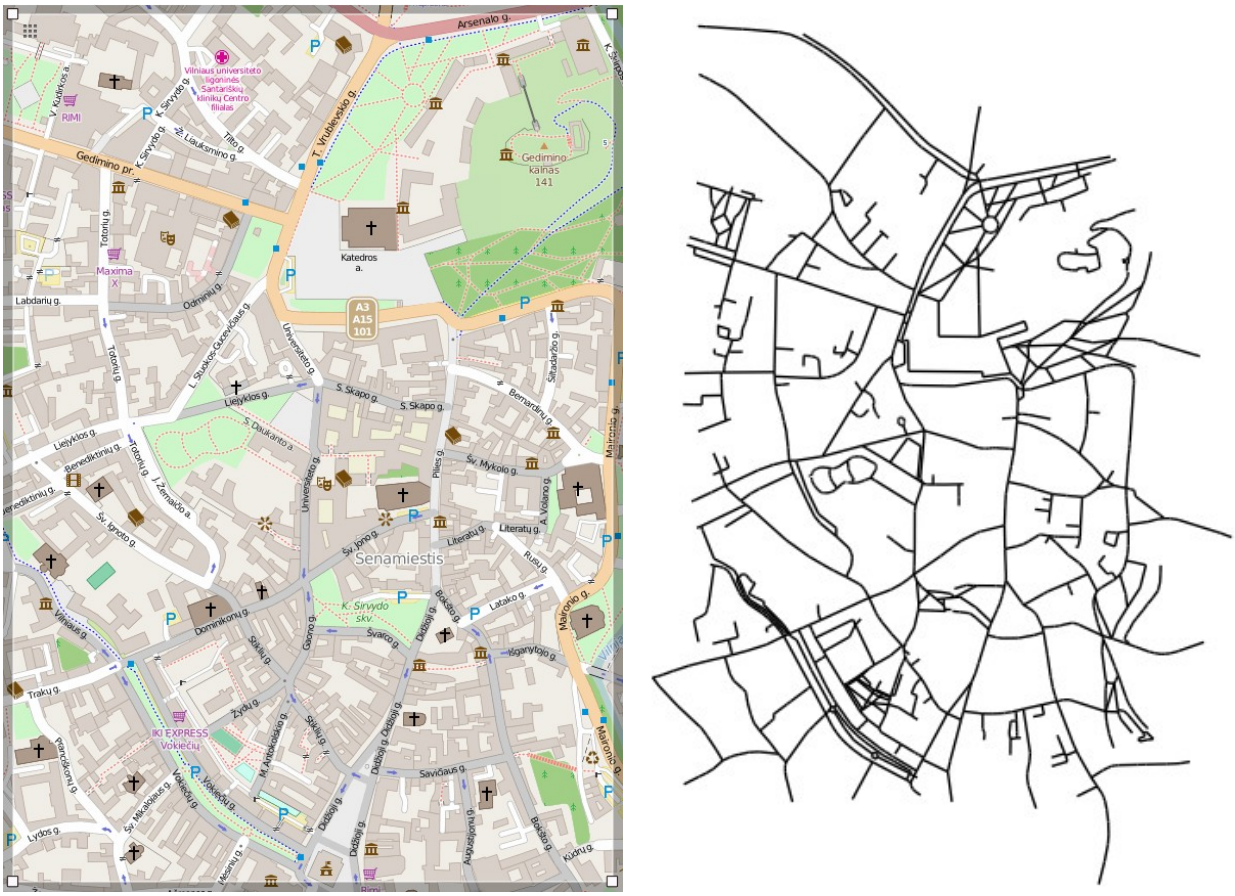
13. pvz. 48 miestų dydžio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: 40 chromosomų, 0.3 rekombinacijos koef. su OX rekombinacija, 0.7 mutacijos koef. su SA mutacija, 10 matavimo pakartojimų.

Iš gautų kreivių (13 pav.) matyti, kad (6) algoritmu gautos pradinės populiacijos chromosomos tikrai pagerina bendrą algoritmo veikimą. Panašu, kad apie 70% populiacijos sugeneravus (6) algoritmu rezultatai pagerėja.

3.4. Praktinis uždavinys

Kadangi iki šiol analizuoti grafai buvo sintetiniai (atsitiktinai sugeneruoti ir modifikuoti ranka) ir gauti parametrų rezultatai yra pastebimai įtakojami grafų pavidalo, buvo nuspręsta išanalizuoti parametrų įtaką nagrinėjant praktinį uždavinį, kur grafų duomenys būtų imti iš esamų žemėlapių.

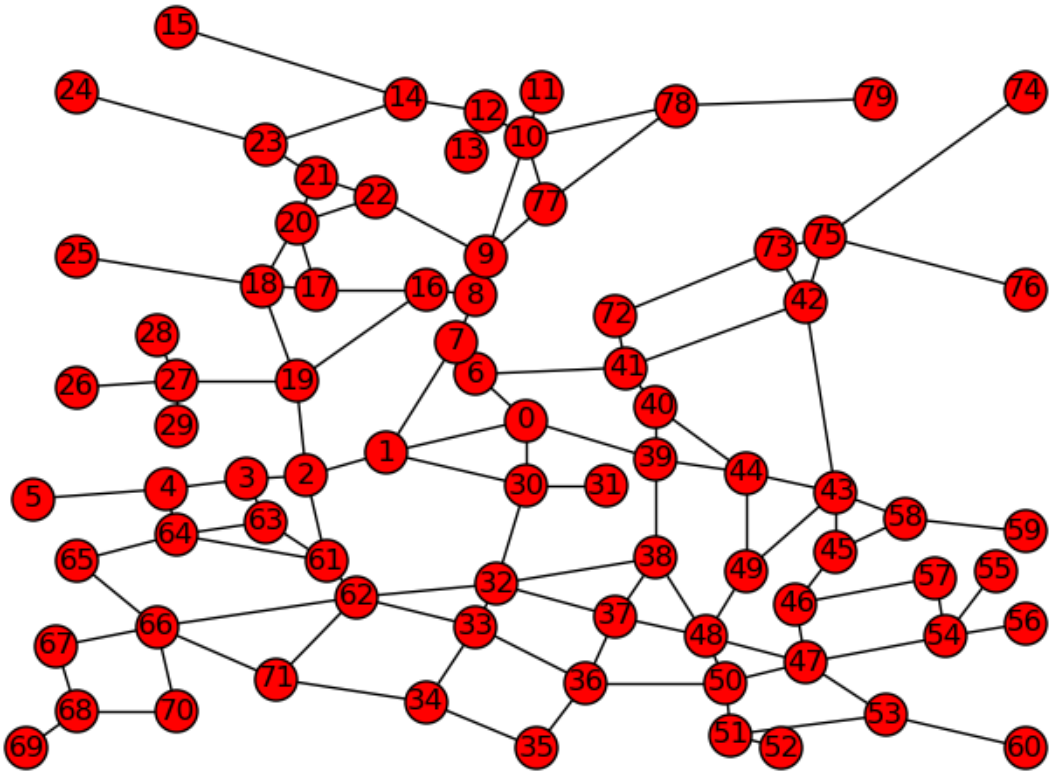
Sukurtas genetinio algoritmo hibridas ir jo implementacija buvo pritaikyta ieškant trumpiausio maršruto siekiant apvaikščioti visas Vilniaus senamiesčio gatveles (14 pav.).



14. pvz. Vilniaus senamiesčio žemėlapis (kairėje) ir jo pavidalas grafe (dešinėje).

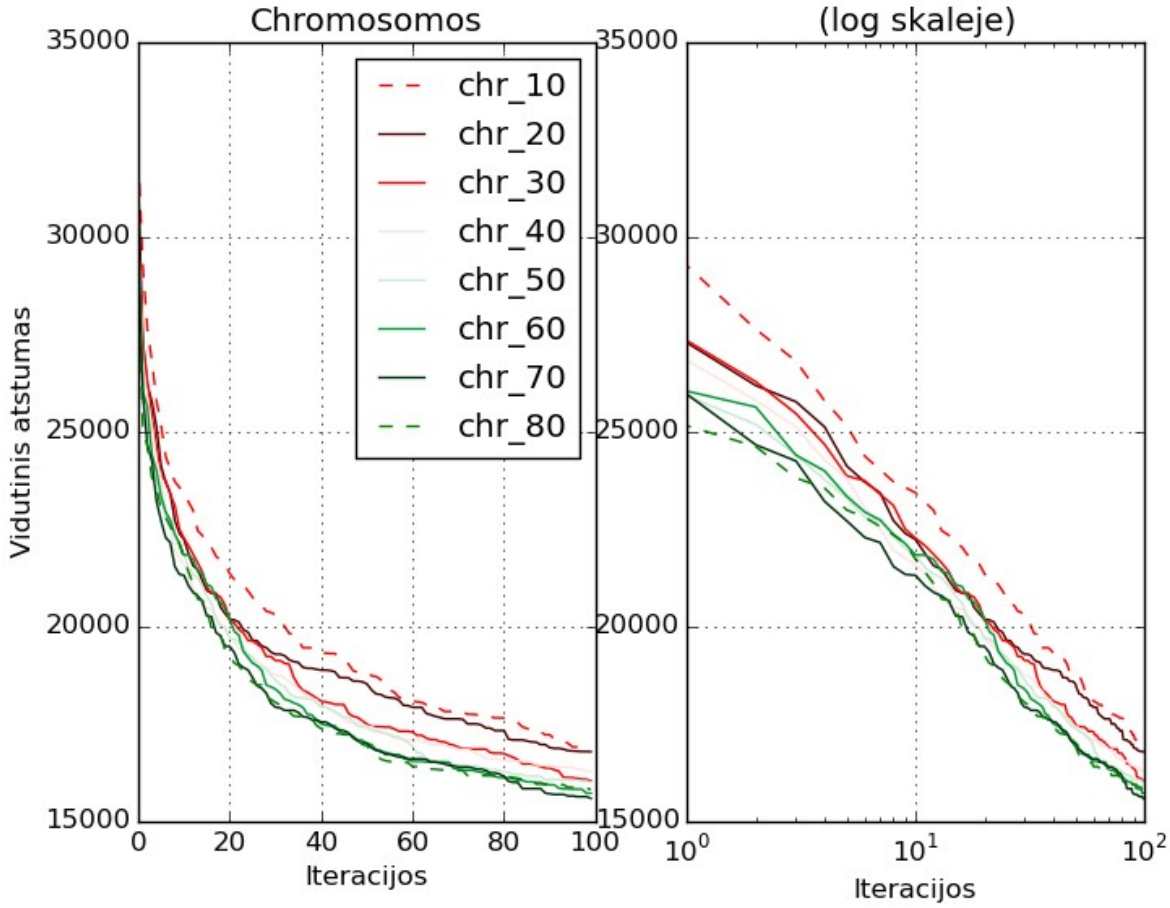
Eksportuoti žemėlapio duomenys iš https://www.openstreetmap.org/ (14 pav.) ir konvertuoti juos į analizuojamą grafą buvo ganėtinai nesudėtinga, tačiau buvo susidurta su kita problema –

grafe yra pernelyg daug perteklinių duomenų. 14 pav. pavaizduotas grafas turi apie 1300 grafo viršūnių, iš kurių nemažai yra sujungtos grandine ir pats grafas yra per didelis plataus spektro analizei atlikti, todėl buvo atliekamas grafo supaprastinimas. Supaprastinus buvo gautas 80 viršūnių grafas, reprezentuojantis pagrindines Vilniaus senamiesčio gatvių jungtis (15 pav.).



15. pvz. Supaprastinta Vilniaus senamiesčio gatvelių reprezentacija grafe.

Turint minėtą grafą buvo tiriama kokia 4 parametrų kombinaciją duotų geriausius rezultatus sprendžiant optimalaus maršruto per visus grafo taškus uždavinį, kai grafas yra paremtas tikrais duomenimis ir yra kur kas didesnis nei praeituose mokslo tiriamuosiuose darbuose [Šar15] (praeituose darbuose didžiausias analizuotas grafas yra su 21 viršūne).



16. pvz. 80 taškų dydžio Vilniaus senamiesčio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: chromosomų skaičius keistas intervale (10, 20, 30, 40, 50, 60, 70, 80) atsitiktinės populiacijos generavimas, 0.3 rekombinacijos koef. su OX rekombinacija, 0.1 mutacijos koef. su SA mutacija, 10 matavimo pakartojimų.

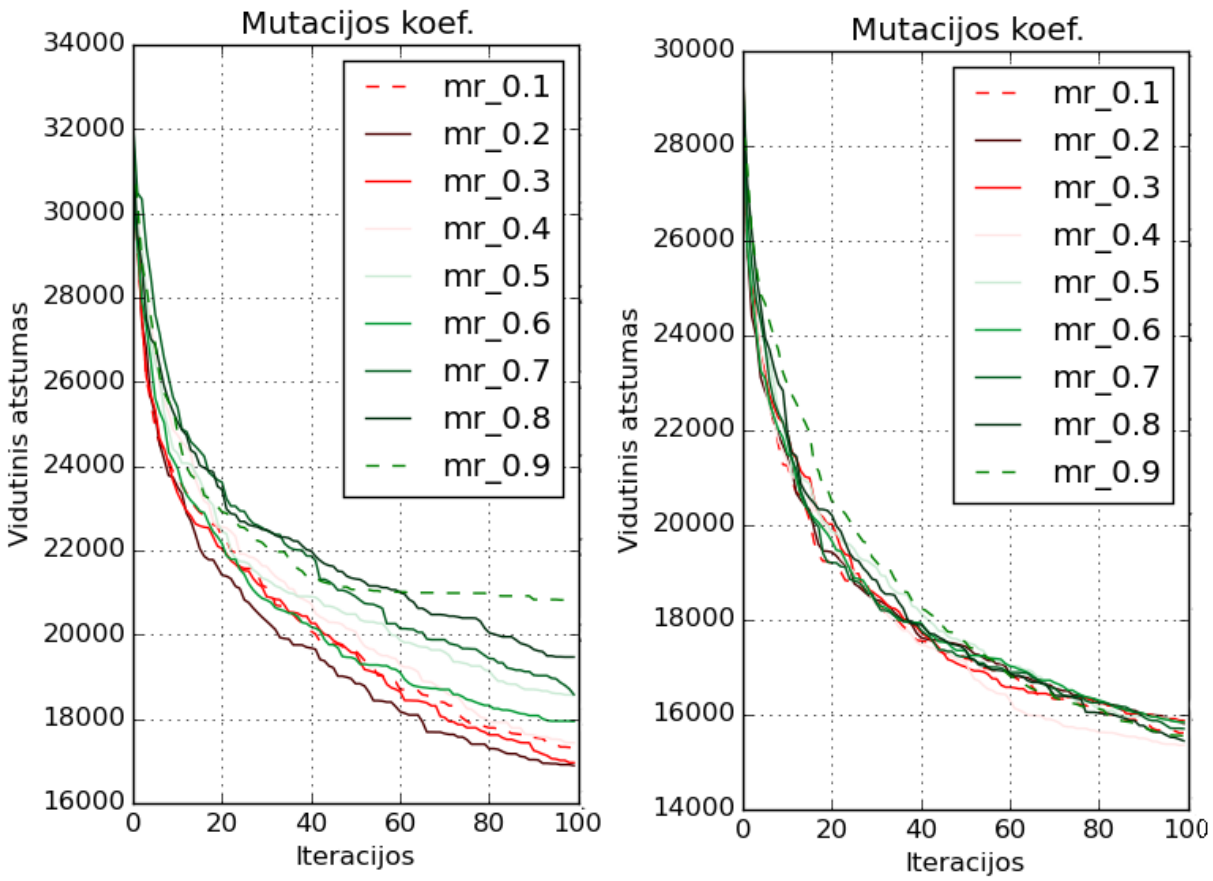
Chromosomų skaičius. Analizuojant chromosomų skaičiaus poveikį algoritmo veikimui (16 pav.) suprantama, kad kuo daugiau chromosomų parinksime tuo mažiau iteracijų reikia pasiekti norimą maršruto ilgį. Tačiau savaime suprantam, kad didėjantis chromosomų skaičius stipriai įtakoja ir skaičiavimo laiką (17 pav.).

	Geriausias laikas (s)	Vidutinis laikas (s)	Blogiausias laikas (s)
10 chromosmų	10	13	16
20 chromosmų	33	40	43
30 chromosmų	51	58	67
40 chromosmų	75	79	82
50 chromosmų	97	102	109
60 chromosmų	133	140	148
70 chromosmų	170	144	124
80 chromosmų	142	148	159

17. pvz. 80 taškų dydžio Vilniaus senamiesčio optimaliausio maršruto 100 iteracijų skaičiavimo laikai. Parametrai: atsitiktinės populiacijos generavimas, 0.3 rekombinacijos koef. su OX rekombinacija, 0.1 mutacijos koef. su SA mutacija, 10 matavimo pakartojimų.

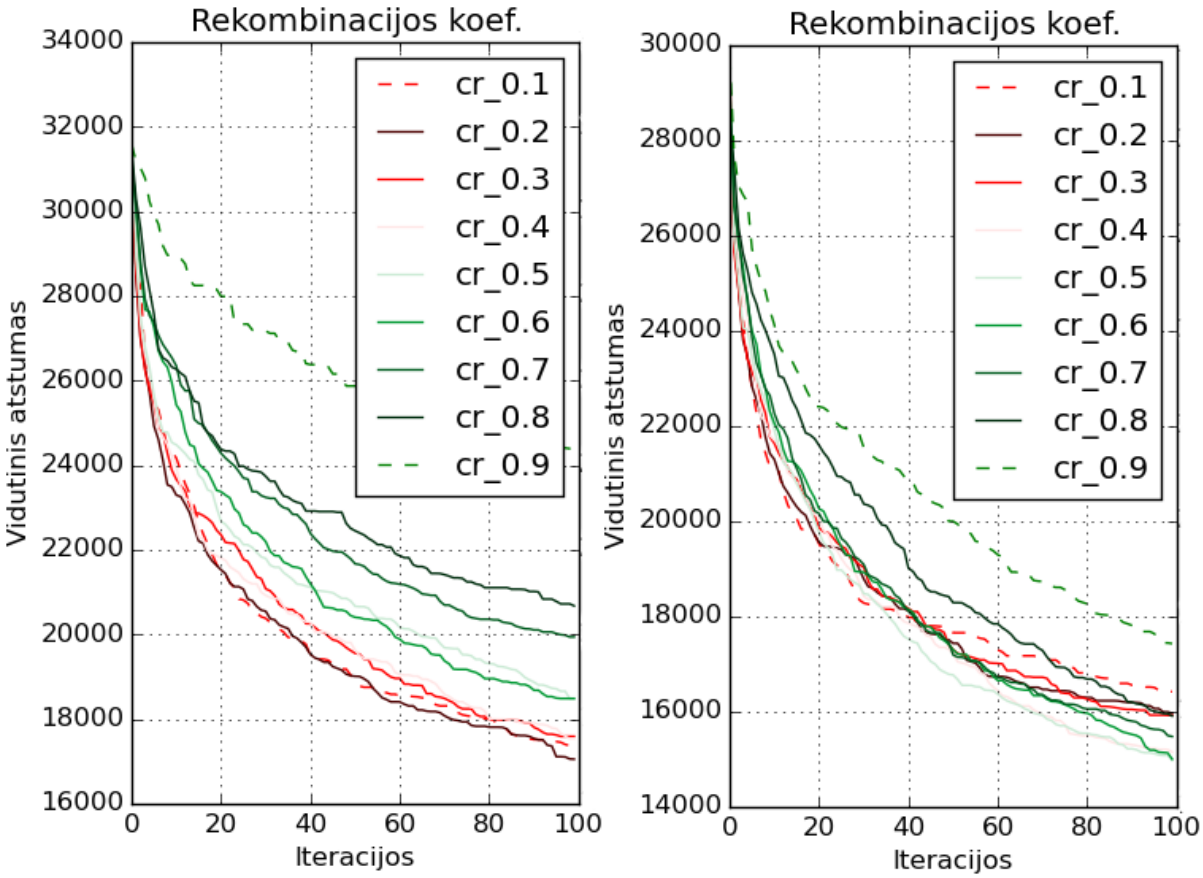
17 pav. įrodo, kad didinant chromosomų skaičių algoritmo skaičiavimo laikas gerokai išauga. Žiūrint į 16 pav. grafiką matyti, kad visų chromosomų kreivės yra panašios formos ir nėra pastebimų ir ryškesnių šuolių kintant jų skaičiui. Todėl būtų galima teigti, kad didelis chromosomų skaičius negarantuoja trumpiausią maršrutą, per trumpiausią laiką, tačiau yra reikalinga pakankamai didelė populiacijos įvairovė, kad ji tobulėtu. Siekiant įsitikinti šiuo teiginiu, reikia patikrinti chromosomų skaičiaus rezultatų priklausomybę ir nuo kitų parametrų (mutacijos, rekombinacijos koeficientų, bei (6) algoritmu generuotų pradinių chromosomų skaičiaus).

Chromosomų skaičius ir mutacijos koeficientas. Buvo atlikta mutacijos koeficientų (intervale 0,1 – 0,9) ir chromosomų įtaka algoritmo skaičiavimams. Iš gautų grafikų (18 pav.) matyti, kad didėjant chromosomų skaičiui mutacijos koeficientų įtaka mažėja galutiniam algoritmo rezultatui. Pastebima, tik kad prie mažesnio chromosomų skaičiaus rezultatai geresni esant mažesniam mutacijos koeficientui.



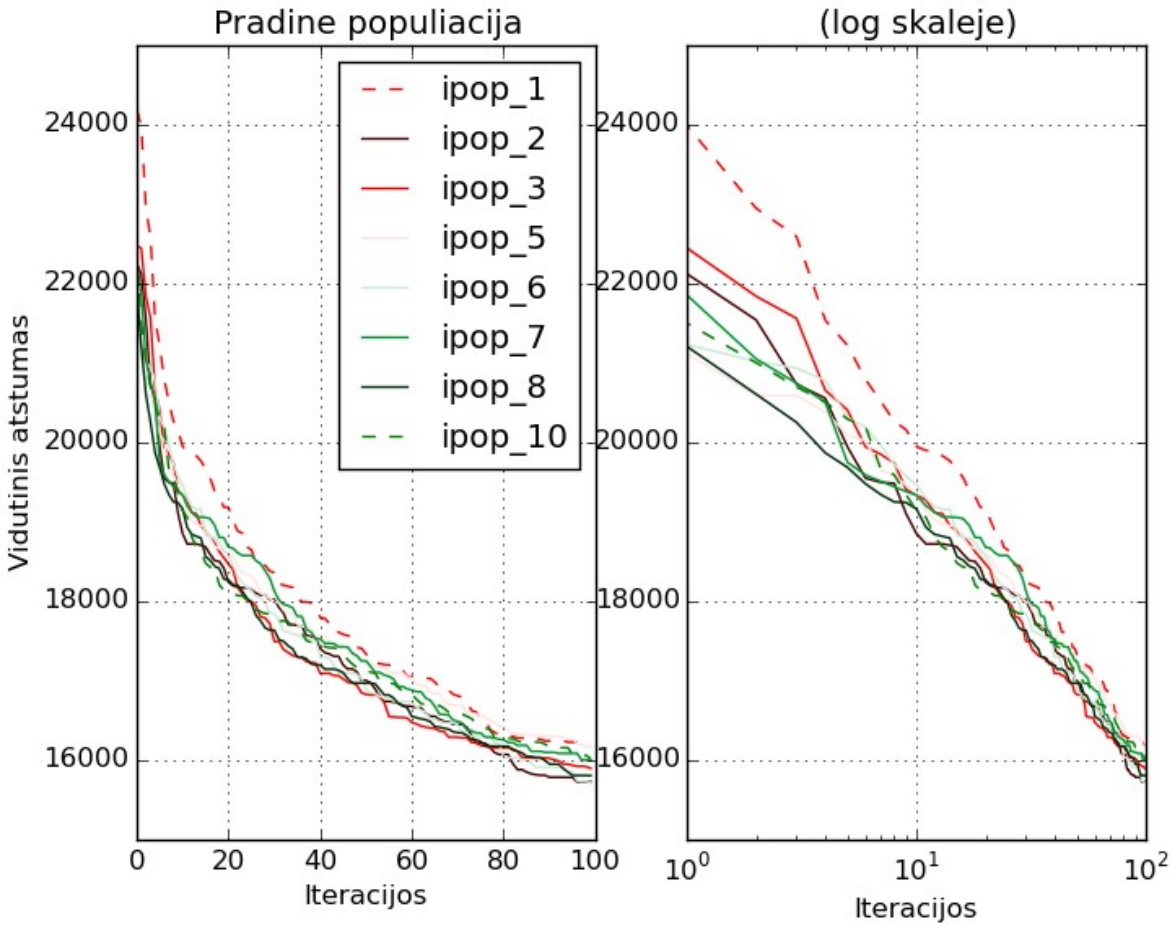
18. pvz. 80 taškų dydžio Vilniaus senamiesčio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: 10 chromosomų kairiajame ir 80 chromosomų dešiniajame grafike, atsitiktinės populiacijos generavimas, 0.3 rekombinacijos koef. su OX rekombinacija, mutacijos koef. Intervale (0,1 - 0,9) su SA mutacija, 10 matavimo pakartojimų.

Chromosomų skaičius ir rekombinacijos koeficientas. Buvo atlikta rekombinacijos koeficientų (intervale 0,1 – 0,9) ir chromosomų įtaka algoritmo skaičiavimams. Iš gautų grafikų 19 pav. matyti, kad rekombinacijos koeficientų įtaka algoritmo rezultatams mažėja didėjant chromosomų skaičiui, tačiau įtakos intervalas nesumažėja taip greitai kaip kad mutacijos koeficientui (18 pav.).



19. pvz. 80 taškų dydžio Vilniaus senamiesčio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: 20 chromosomų, atsitiktinės populiacijos generavimas, rekombinacijos koef. Intervale (0,1 – 0,9) su OX rekombinacija, 0.3 mutacijos koef. su SA mutacija, 10 matavimo pakartojimų.

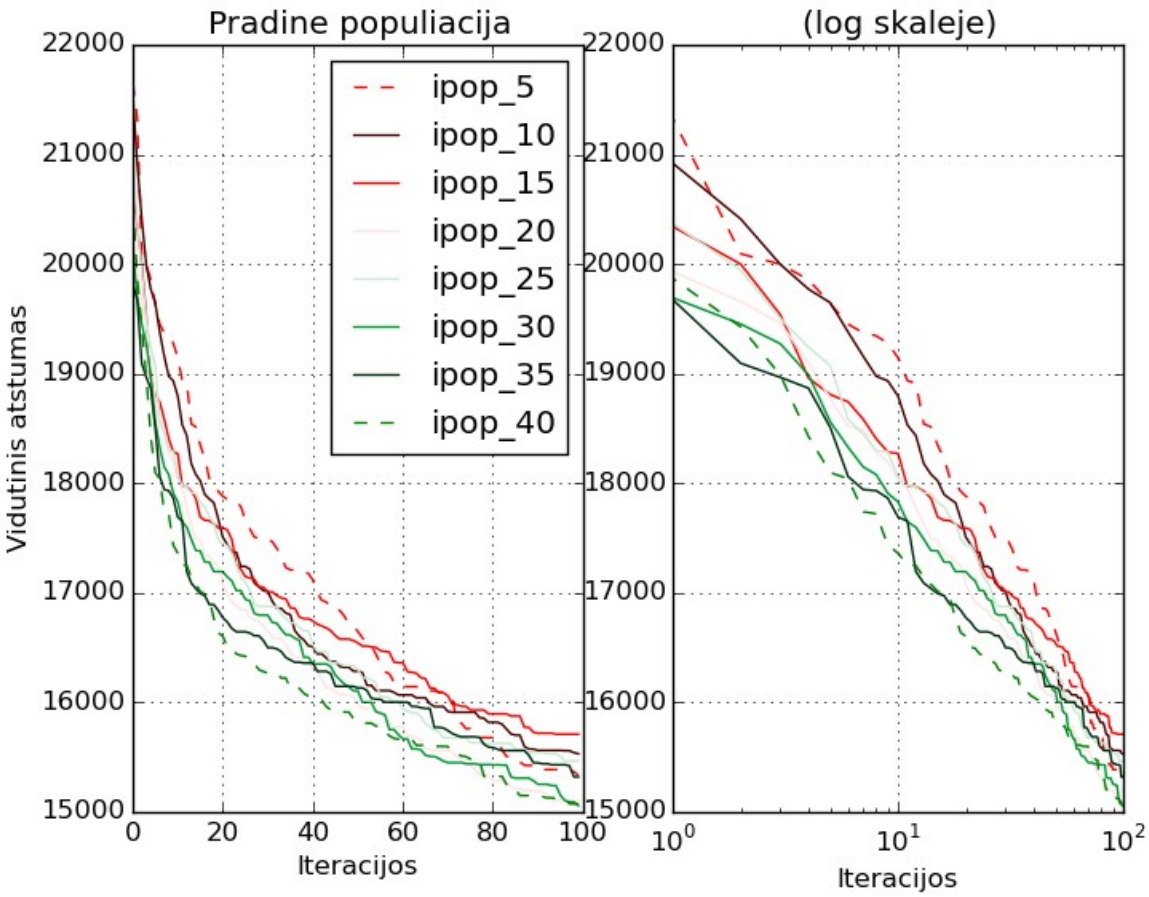
Chromosomų skaičius ir (6) algoritmu generuotos pradinės populiacijos skaičius. Taip pat buvo atlikta neatsitiktinai generuotos populiacijos skaičius su galutiniu chromosomų skaičiumi įtaka hibridinio algoritmo rezultatams analizė (20 pav., 21 pav.).



20. pvz. 80 taškų dydžio Vilniaus senamiesčio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: 10 chromosomų, pradinė populiacija generuota (6) algoritmu intervale (1 - 10), 0.3 rekombinacijos koef. su OX rekombinacija, 0.3 mutacijos koef. su SA mutacija, 10 matavimo pakartojimų.

Iš 20 pav. matyti, kad apie 40% pradinės populiacijos, sugeneravus modifikuotu ACS algoritmu genetinio algoritmo, rezultatai būna geriausi. Tačiau 21 pav. ir kituose grafikuose pradinės populiacijos generavimo grafikuose pastebėta, kad didėjant chromosomų skaičiui geriausi algoritmo rezultatai būna su vis didesniu (6) algoritmu pradinės populiacijos generuotu chromosomų

skaičiumi.

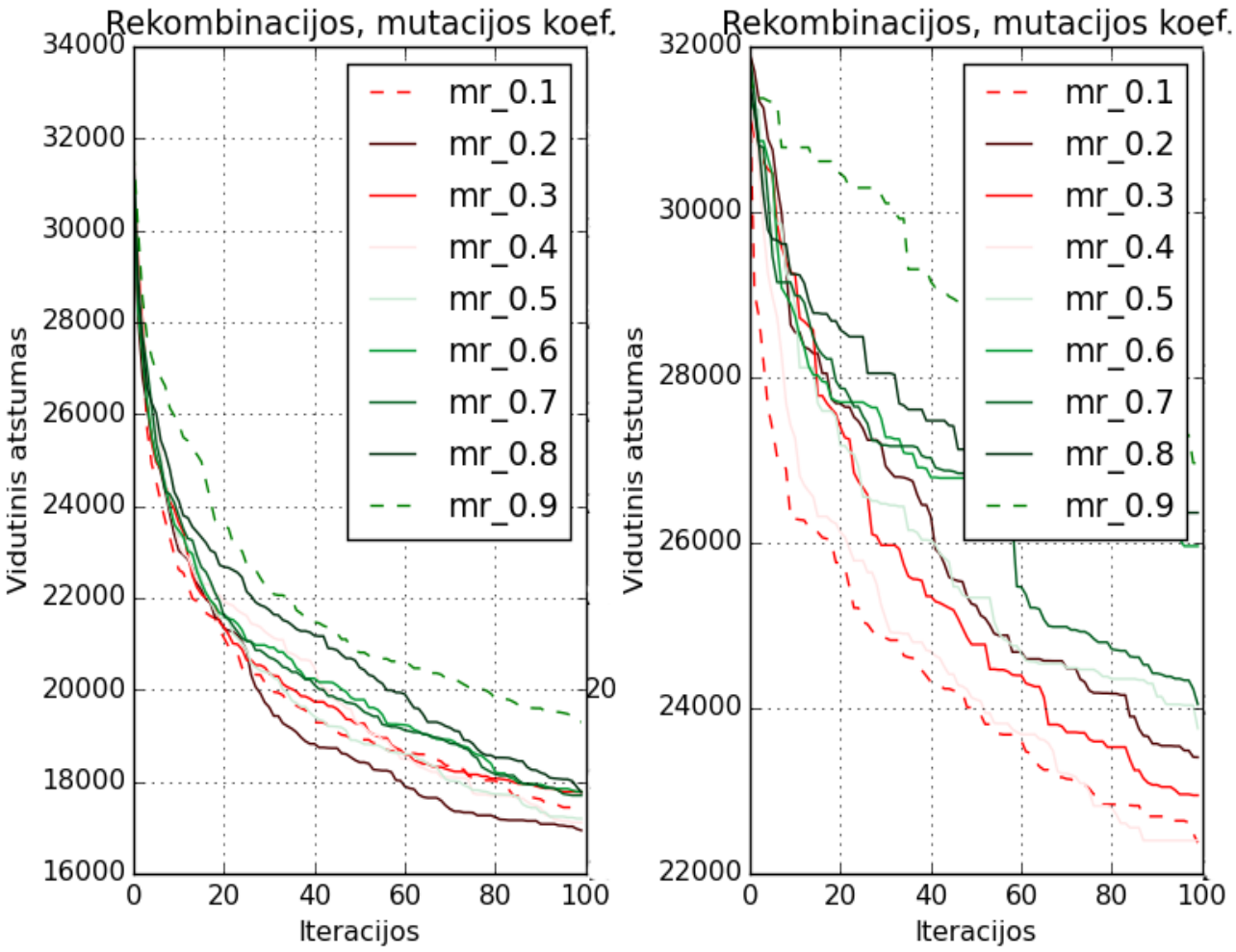


21. pvz. 80 taškų dydžio Vilniaus senamiesčio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: 40 chromosomų, pradinė populiacija generuota (6) algoritmu intervale (5 - 40), 0.3 rekombinacijos koef. su OX rekombinacija, 0.3 mutacijos koef. su SA mutacija, 10 matavimo pakartojimų.

Būtų galima visas pradines chromosomas generuoti ACS algoritmo paremtu metodu, tačiau tai tiesiogiai įtakoja skaičiavimo laiką ir didėjant chromosomų skaičiui bendras algoritmo veikimo laikas taip pat didėja kur kas sparčiau.

Atsižvelgiant į pastarąją chromosomų skaičiaus priklausomybę nuo kitų parametrų, būtų galima daryti prielaidą, kad nedidelis chromosomų skaičius, apie 20, optimaliai tinka tiek skaičiavimo laiko, tiek randamo atstumo atžvilgiu.

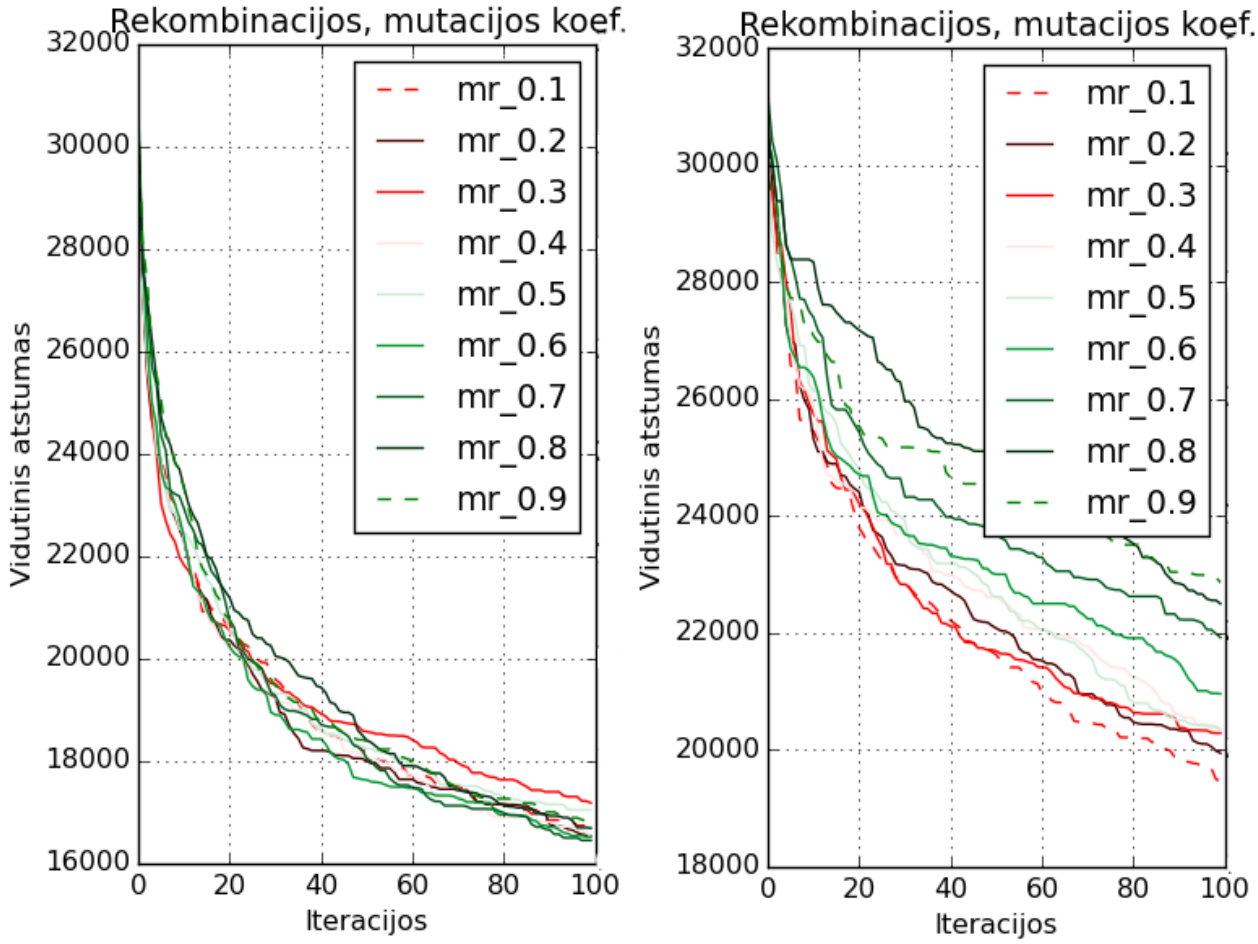
Rekombinacijos ir mutacijos koeficientas. Siekta išsiaiškinti mutacijos ir rekombinacijos koeficientų tarpusavio kombinacijų poveikį hibridinio algoritmo veikimui (22 pav., 23 pav.).



22. pvz. 80 taškų dydžio Vilniaus senamiesčio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: 10 chromosomų, pradinė populiacija generuota atsitiktinai, 0.1 kairėje ir 0.9 rekombinacijos koef. dešinėje su OX rekombinacija, SA mutacija intervale su (0,1 – 0,9) koef., 10 matavimo pakartojimų.

Iš 22 pav. ir 23 pav. matyti, kad didėjant rekombinacijos koeficientui mutacijos koeficiento įtaka tampa vis ryškesnė (įtakos intervalas tarp 0,1 ir 0,9 *mr* koef.), arba kitaip tariant – kuo mažesnis rekombinacijos koeficientas, tuo mažesnė skirtingų mutacijos koeficientų įtaka. Šis pastebėjimas

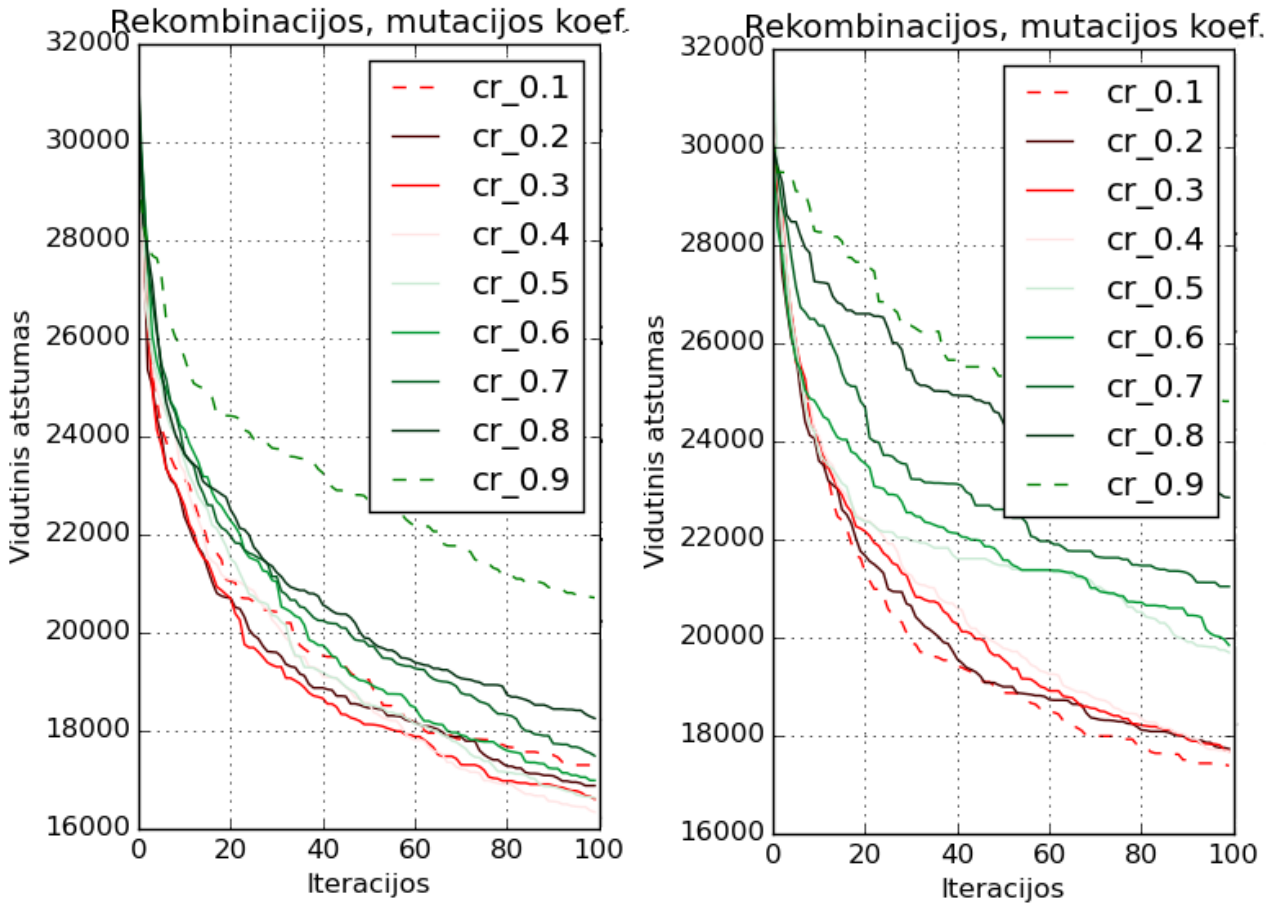
galioja įvairiam chromosomų skaičiui. Tačiau didėjant chromosomų skaičiui matoma, kad mutacijos koeficiento įtakos intervalas siaurėja – prie vis didesnio chromosomų skaičiaus, vis mažiau skiriasi mutacijos koeficientų rezultatai.



23. pvz. 80 taškų dydžio Vilniaus senamiesčio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: 30 chromosomų, pradinė populiacija generuota atsitiktinai, 0.1 kairėje ir 0.9 rekombinacijos koef. dešinėje su OX rekombinacija, SA mutacija intervale su (0,1 – 0,9) koef., 10 matavimo pakartojimų.

Iš 24 pav. ir kitų grafikų pastebėta, kad prie visų mutacijos koeficientų rekombinacijos koeficientų įtaka pasiskirsto vienodai – prie mažesnių *cr* koeficientų gaunami trumpesni atstumai ir atvirkščiai. Skirtingam chromosomų skaičiui optimalus rekombinacijos koeficientas kinta, tačiau tas

kitimas nėra didelis ir pasiskirsto intervale (0,1 – 0,4).



24. pvz. 80 taškų dydžio Vilniaus senamiesčio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: 20 chromosomų, pradinė populiacija generuota atsitiktinai, OX rekombinacija intervale su (0,1 – 0,9) koef., 0.1 kairėje ir 0.9 mutacijos koef. dešinėje su SA mutacija, 10 matavimo pakartojimų.

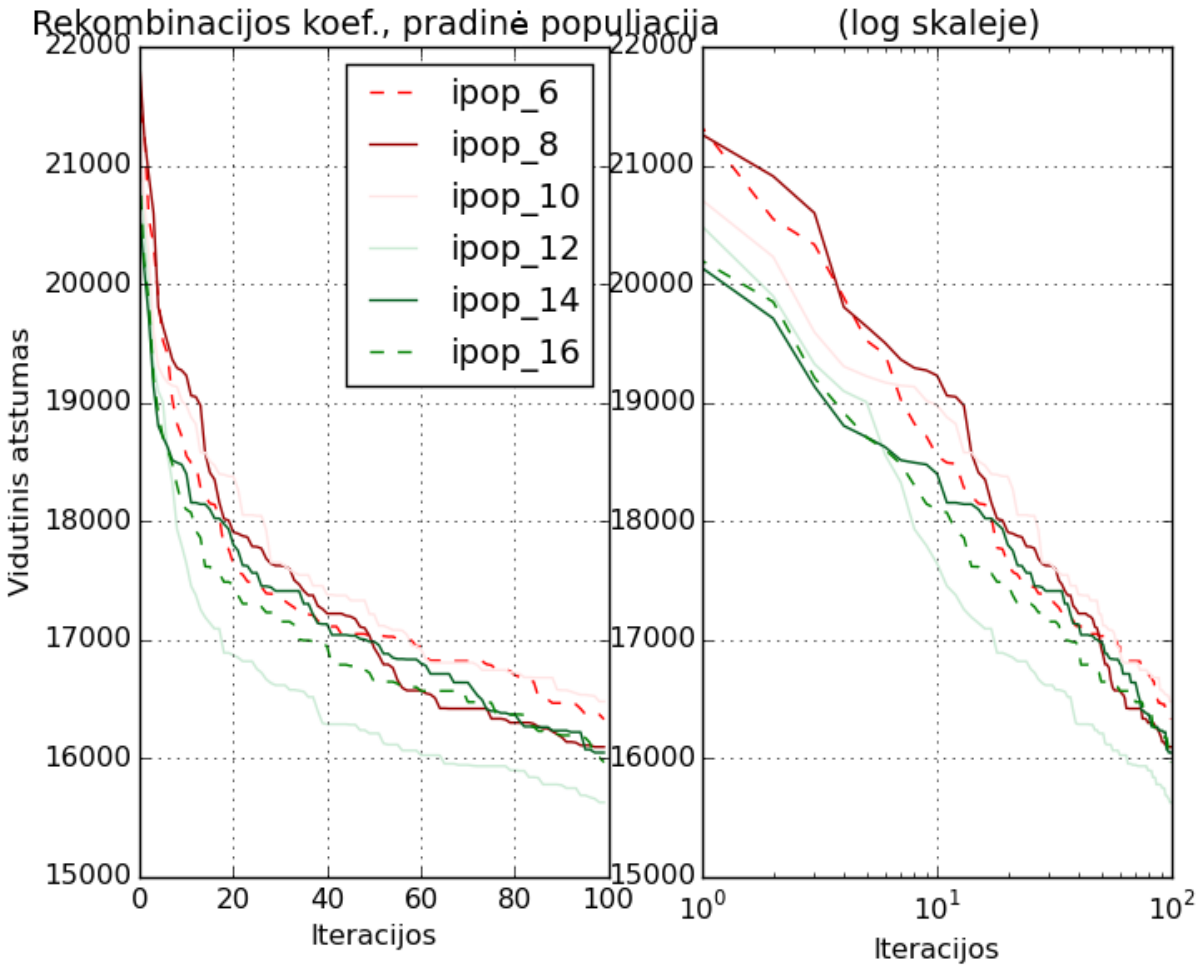
22 Pav., 23 pav, 24 pav. ir kituose grafikuose optimaliausi maršrutai randami naudojant mažą mutacijos koeficientą. Mutacijos koeficiento optimalumo priklausomybė nuo rekombinacijos koeficiento ir chromosomų skaičiaus yra ganėtinai maža ir bendru atveju optimaliausias *mr* koeficientas yra 0,15.

	mr 0.1	mr 0.2	mr 0.3	mr 0.4	mr 0.5	mr 0.6	mr 0.7	mr 0.8	mr 0.9
cr 0.1	41	42	37	39	34	31	28	24	22
cr 0.2	29	27	25	23	22	20	18	15	13
cr 0.3	24	23	21	19	18	15	14	12	11
cr 0.4	19	18	16	14	14	12	11	10	9
cr 0.5	14	13	12	11	10	9	8	8	7
cr 0.6	10	9	9	8	8	7	6	6	5
cr 0.7	7	7	7	6	5	5	4	4	4
cr 0.8	4	4	4	4	3	3	3	3	2
cr 0.9	2	2	2	2	2	1	1	1	1

25. pvz. 80 taškų dydžio Vilniaus senamiesčio optimaliausio maršruto skaičiavimų laikai keičiant mutacijos ir rekombinacijos koeficientus. Parametrai: 20 chromosomų, pradinė populiacija generuota atsitiktinai, OX rekombinacija intervale su (0,1 – 0,9) koef., SA mutacija intervale su (0,1 – 0,9) koef., 10 matavimo pakartojimų.

Iš 14 pav. matyti, kad esant dideliems mutacijos ir rekombinacijos koeficientams bendras algoritmo veikimo laikas sumažėja. Ši įžvalga savaime suprantama, nes kuo mažesni šie koeficientai, tuo didesnė tikimybė kad mutacijos ar rekombinacijos operatorius bus įvykdytas. Trumpiausio atstumo atžvilgiu algoritmo rezultatai gaunami geresni kai šie koeficientai yra arčiau 0,1. Kadangi skaičiavimo laikai nekinta taip drastiškai (kinta tiesiškai), ir bendrai algoritmo skaičiavimo laiką nemažai įtakoja pati jo implementacija, pasirenkant optimalius mutacijos ir rekombinacijos koeficientus trumpiausias atstumas yra svarbesnis nei 14 pav. lentelėje atvaizduoti skaičiavimo laikai.

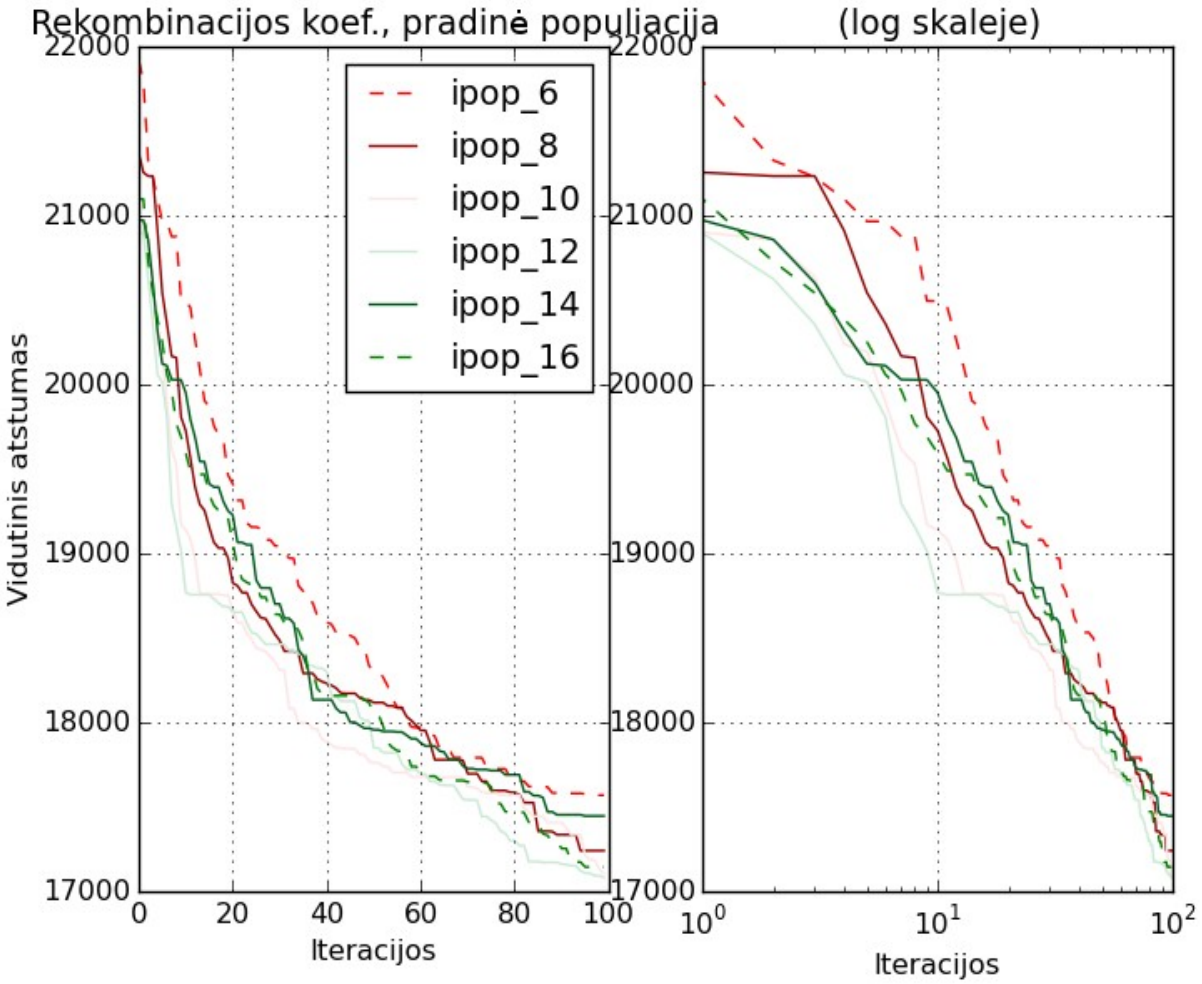
Rekombinacijos koeficientas ir (6) algoritmu generuotos pradinės populiacijos skaičius. Analizuojant rekombinacijos koeficiento įtaką algoritmo rezultatams taip pat buvo tirtas galutinis rezultatas kintant (6) algoritmu parinktų populiacijos pradinių chromosomų skaičius (26 pav., 27 pav.).



26. pvz. 80 taškų dydžio Vilniaus senamiesčio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: 20 chromosomų, pradinė populiacija generuota (6) algoritmu intervale (6 - 16), 0.1 rekombinacijos koef. su OX rekombinacija, 0.15 mutacijos koef. su SA mutacija, 10 matavimo pakartojimų.

Iš 26 pav., 27 pav. bei kitų grafikų analizės pastebėta, kad kintant rekombinacijos koeficientams optimaliausias (6) algoritmu parinktos pradinės populiacijos skaičius kinta intervale (10 – 14).

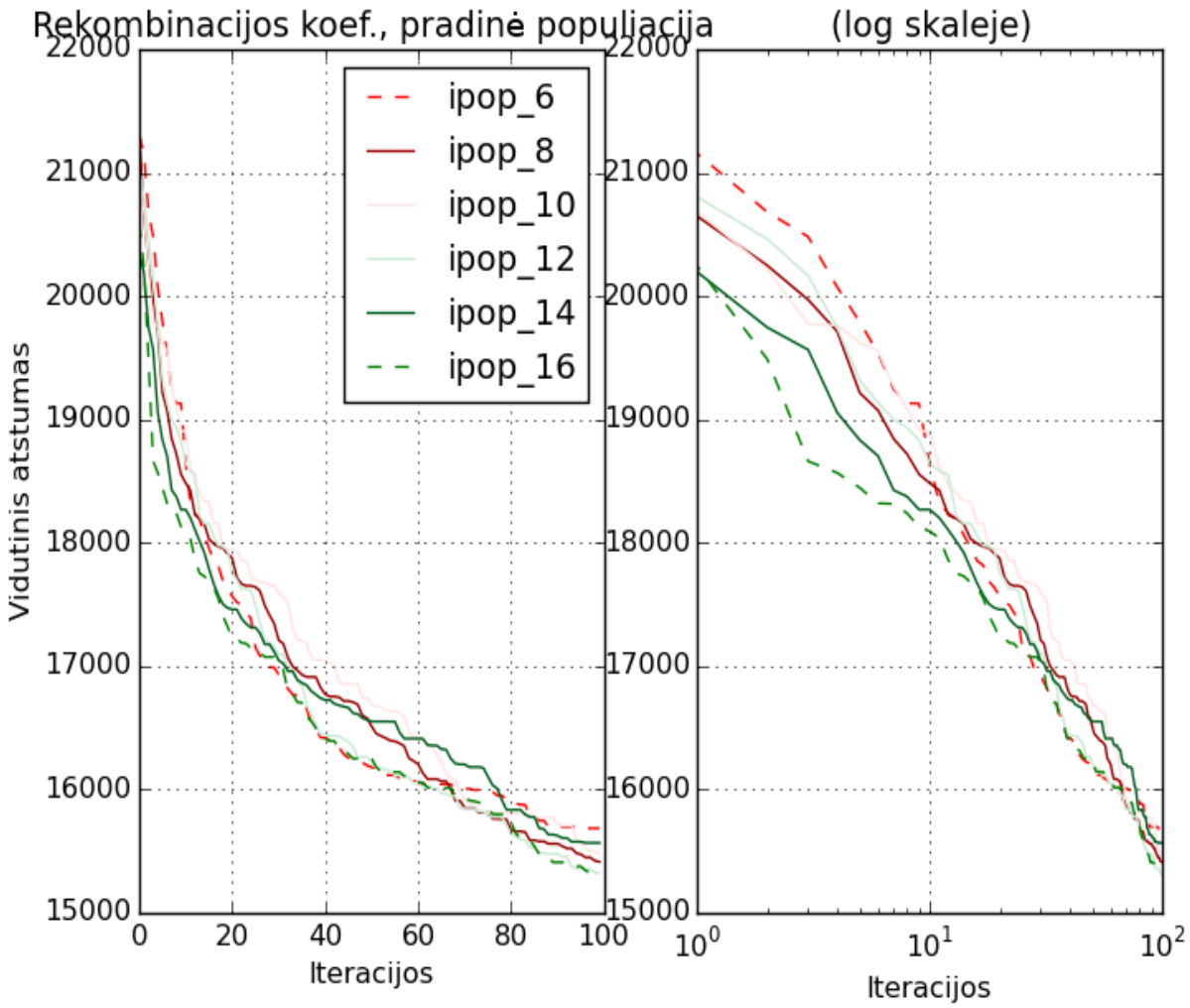
Todėl galima teigti, kad apie 60% pradinės populiacijos sugeneravus ACS algoritmo hibridu yra gaunami optimaliausias rezultatai su hibridiniu genetiniu rezultatu.



27. pvz. 80 taškų dydžio Vilniaus senamiesčio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: 20 chromosomų, pradinė populiacija generuota (6) algoritmu intervale (6 - 16), 0.9 rekombinacijos koef. su OX rekombinacija, 0.15 mutacijos koef. su SA mutacija, 10 matavimo pakartojimų.

Analizuojant rekombinacijos įtaką pradinės populiacijos generavimo skaičiui pastebėta, kad *cr* koeficientui artėjant prie 0,4 (6) algoritmu generuotos pradinės populiacijos įtakos rėžiai mažėja, tai matyti 28 pav. grafike. Prie maksimalaus, arba minimalaus rekombinacijos koeficiento ACS algoritmo generuotos pradinės populiacijos skaičius turi didesnės įtakos galutiniai hibridinio

genetinio algoritmo veikimui, tačiau galima teigti kad visais atvejais optimalus neatsitiktinai generuotų chromosomų skaičius yra 12.



28. pvz. 80 taškų dydžio Vilniaus senamiesčio optimaliausio maršruto ilgio kitimas didėjant iteracijų skaičiui. Parametrai: 20 chromosomų, pradinė populiacija generuota (6) algoritmu intervale (6 - 16), 0.3 rekombinacijos koef. su OX rekombinacija, 0.15 mutacijos koef. su SA mutacija, 10 matavimo pakartojimų.

Iš 26 pav., 27 pav., 28 pav. ir kitų grafikų analizės pastebėta, kad optimaliausia rekombinacijos koeficientas visais atvejais yra apie 0,3.

Apibendrinant atliktus matavimus tiriant 4 parametrų (mutacijos, rekombinacijos koeficientai,

chromosomų skaičiaus, modifikuotu ACS algoritmu generuotos pradinės populiacijos skaičius) galima teigti, kad sprendžiant pasirinktą praktinį uždavinį (Vilniaus senamiesčio gatvelių aplankymą 14 pav.) optimaliausias maršrutas randamas naudojant 20 chromosomų, 12 pradinių sugeneruojant (6) algoritmu, 0.15 mutacijos, bei 0.3 rekombinacijos koeficientus

4. Išvados

5. Anotacija

6. Summary

7. Literatūros sąrašas

[CheChi11]	Shyi-Ming Chen, Chih-Yao Chien, Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. 2011
[Pot96]	Jean-Yves Potvin, Genetic Algorithms for the Traveling Salesman Problem, 1996
[KatNar01]	Kengo Katayama,Hiroyuki Narihisa: An Efficient Hybrid Genetic Algorithm for the Traveling Salesman Problem, 2001
[LiZhoGui11]	Yang Li, Aimin Zhou, Guixu Zhang, Simulated Annealing with Probabilistic Neighborhood for Traveling Salesman Problems, 2011
[YuHuZha09]	Wei-jie Yu, Xiao-min Hu, Jun Zhang, Self-Adaptive Ant Colony System for the Traveling Salesman Problem, 2009
[Šar15]	Karolis Šarapnickis, Mokslo tiriamasis darbas I/II: optimalių maršrutų paieškos algoritmai, 2014/2015