VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMOS

Genetinio algoritmo taikymas objektų vietų parinkimui

Application Of Genetic Algorithm For Facility Location

Bakalaurinis darbas

Atliko: 4 kurso 1 grupės studentas
Lukas Martišius (parašas)

Darbo vadovas: doc. dr. Algirdas Lančinskas (parašas)

Vilnius – 2020

TURINYS

Įvadas

Objektų vietų parinkimo uždavinys dažnai yra formuluojamas kaip matematinio optimizavimo uždavinys, kurio sprendimui naudojami įvairūs matematinio optimizavimo metodai. Tačiau, kai kurie objektų vietų parinkimo uždaviniai negali būti būti aprašyti adekvačia matematine forma, todėl kai kurie matematinio optimizavimo metodai, pavyzdžiui, reikalaujantys tikslo funkcijų diferencijuojamumo, negali būti taikomi.

Kita vertus, sprendžiant praktikoje pasitaikančius uždavinius ne visada yra būtina rasti optimalų sprendinį, bet pakanka žinoti jo artinį su priimtina paklaida. Todėl praktikoje dažnai yra naudojami įvairūs atsitiktinės paieškos metodai, aproksimuojantys optimalų sprendinį. Tokių algoritmų pavyzdžiai gali būti Genetinis Algoritmas (angl. Genetic Algorithm), Dalelių spiečiaus optimizavimas (angl. Particle Swarm Optimization), Modeliuojamo atkaitinimo (angl. Modelling Annealing) algoritmas.

Atsitiktinės paieškos algoritmai paprastai būna lengvai įgyvendinami ir gali būti taikomi įvairiems optimizavimo uždaviniams spręsti - tereikia turėti galimybę surasti bet kurį uždavinio paieškos srities sprendinį.

Šiame darbe objektų vietų parinkimo uždavinius spręsti bus taikomas Genetinis algoritmas, kuris yra paremtas biologijos žiniomis apie evoliuciją, Darvino teorija, t.y. atsitiktiniu būdu parinkus generaciją ir leidžiant jai gyvuoti natūraliu atrankos būdu paliekant geriausius populiacijos elementus gaunamas sprendinys.

Darbe nagrinėjamas objektų vietų parinkimo uždavinys aktualus į rinką įeinančiai naujai įmonei, siekiant maksimizuoti naujai atidarytų objektų pelną atsižvelgiant į konkurenciją su rinkoje jau esančių įmonių objektais.

Tikslas ir uždaviniai

Šio darbo tikslas yra ištirti genetinio algoritmo efektyvumą sprendžiant įvairius objektų vietų parinkimo uždavinius ir nustatyti optimalius genetinio algoritmo parametrų rinkinius ir sprendinių generavimo strategijas efektyviam objektų vietų parinkimui.

Darbo uždaviniai prasideda nuo objektų vietų parinkimo uždavinių aibės sudarymo, kokius tyrimus reikės atlikti. Po to apibrėžiami vertinimo kriterijai. Panagrinėjami egzistuojami algoritmai ir bandoma pasiūlyti naujų, labiau prisitaikiusių šiam uždaviniui strategijų. Vėliau bus atliekami tyrimai norint ištirti įvairių genetinio algoritmo parametrų ir sprendinių generavimo strategijų įtaką objektų vietų parinkimo efektyvumui ir tikslumui. Atlikus eksperimentus bus parenkamos optimalios genetinio algoritmo strategijos, geriausiai sprendžiančios šią užduotį.

Tyrimo metodas naudojamas šiam darbui atlikti bus eksperimentinis tyrimas.

1. Objektų vietų parinkimo uždavinys

1.1. Tyrimo objektas ir aktualumas

Tyrimo objektas – genetinis algoritmas ir jo taikymas objektų vietų parinkimo uždaviniams spręsti. Algoritmas yra plačiai paplitęs įvairiose mokslo srityse, tokiose kaip, matematika, kompiuterių mokslas, gamtos mokslas, finansai ir ekonomika ir kita. Keli jo panaudojimo pavyzdžiai:

- **Ieškant kompiuterinės įrangos defektų** - [HST92] Hitoshi Iba, Sumitaka Akiba, Tetsuya Higuchi, Taisuke Sato: BUGS: A Bug-Based Search Strategy using Genetic Algorithms. PPSN 1992:
- **Vaizdo apdorojimui** - [SBF14] A. dos Santos-Paulino, J.-C. Nebel and F.Florez-Revuelta (2014) Evolutionary algorithm for dense pixel matching in presence of distortions, EvoStar Conference, Granada, Spain, 23–25 April 2014
- **Klimatologijoje** - [SKV15] Karolina Stanislawska; Krzysztof Krawiec; Timo Vihma (July 15, 2015). "Genetic Programming for Estimation of Heat Flux between the Atmosphere and Sea Ice in Polar Regions"
- **Finansų matematikoje** - [BAB11] Zhang, S.X.; Babovic, V. (2011). "An evolutionary real options framework for the design and management of projects and systems with complex real options and exercising conditions". Decision Support Systems. 51 (1): 119–129.

1.2. Objektų vietų parinkimo uždavinys

Šiame ir tolesniuose skyreliuose objektu (angl. Facility) laikysiu įvairius pramonės ir verslo objektus, teikiančius klientams paslaugas ar prekes konkrečioje geografinėje srityje. Tokių objektų vietų parinkimas yra strategiškai svarbus įmonėms, konkuruojančioms tarpusavyje dėl užimamos rinkos dalies. Tad uždavinys sprendžia situaciją, kurioje rinkoje egzistuoja pirmasis žaidėjas ir į ją nori ateiti naujas žaidėjas. Yra žinoma kiekviena konkurento objekto vieta bei jos rinkos vertė, konkrečiu atveju - kiek pritraukia klientų. Antrasis žaidėjas nori patekti į rinką su geriausia įmanoma naujų objektų rinkos verte. Konkretų uždavinį aprašo:

- **Paklausos taškai (angl. Demand Points)** - visų rinkoje egzistuojančių objektų aibė su koordinatėmis bei juose gyvenančių žmonių skaičiumi. Gali būti nurodytas ir objekto paklausumo indeksas.
- **Egzistuojančius objektus (angl. Preexisting Facilities)** - jau egzistuojantys ir gyvuojantys objektai, kurie yra populiacijos dalis, dažniausiai parinkti geriausi populiacijos elementai.
- **Potencialios vietos (angl. Candidate Locations)** - potencialių vietų aibė iš kurios bus galima pasirinkti vietas, kur statyti naujus objektus.
- **Vietų naujiems objektams kintamasis (angl. Decision Variable)** - aibė taškų, iš potencialių objektų aibės. Vienas iš galimų uždavinio sprendinių.

Šie duomenys yra skirti uždaviniui aprašyti. Algoritmo principas remiasi tuo, kad skirtingose vietose pastatyti objektai turi skirtingą savo įvertį kiekvieno paklausos taško atžvilgiu, kurį nulemia atstumas iki taško, bei, jei nurodytas, objekto paklausumo indeksas. Uždavinio sprendimas nėra ypatingai sunkus norint surasti 1 geriausią tašką, tačiau kur kas sudėtingėja didinant norimą naujai pastatytų objektų skaičių.[LFPZ15]

Pavyzdžiui realus uždavinys, kuriame egzistuoja 5000 paklausos taškų ir norima pastatyti 5 naujus objektus pareikalautų

$$\binom{5000}{5} = 25{,}989{,}619{,}781{,}251{,}000$$

funkcijų perskaičiavimų.

1.2.1. Klientų elgsenos modelis

Dar vienas esminis uždavinio komponentas tai klientų elgsenos modelis. Šis kriterijus nurodo, kaip elgsis klientai, atitinkamoje situacijoje ir kas nulems jų pasirinkimą. Klientų elgsenos modelis nulemia tikslo funkciją, kuri bus naudojama skirtingų taškų palyginimui. Šiam darbui atlikti pasirinkti du klientų elgsenos modeliai - binarinis bei proporcinis. Binarinio modelio taisyklės:

- Naujai pastatytas objektas gauna visus klientus, kuriems tas objektas yra arčiau nei bet kuris kitas jau egzistuojantis objektas.
- Jeigu tarp naujai pastatyto objekto ir nagrinėjamo taško bei jau egzistuojančio objekto ir nagrinėjamo taško atstumai yra lygūs, naujasis taškas gauna trečdalį gyventojų nuo nagrinėjamo taško populiacijos.
- Jeigu atstumas iki naujojo objekto yra tolimesnis, nei bet kurio kito arčiau egzistuojančio objekto, naujasis objektas gyventojų iš nagrinėjamo taško negauna.

Proporcinis modelis kiek sudėtingesnis, tačiau labiau pritaikomas praktikoje, nes klientų pasirinkimą nulemia ne tik atstumas, bet ir kokybė. Tad šiam modeliui reikia dar vieno duomens, tai kiekvieno taško kokybės indekso q. Tuomet apskaičiuojant kiekvieno taško paklausą naudojama formulė:

$$a_{\text{ij}} = \frac{q_{\text{j}}}{1 + d_{\text{ij}}}$$

čia q_{j} – j-tojo objekto kokybės parametras, o d_{ij} – atstumas tarp j-jo objekto ir i-tojo paslaugos taško. Tokiu būdu klientai objektą renkasi atsižvelgdami į objekto kokybę ir atstumą iki objekto. Atskiru atveju, esant vienodoms q_{j} vertėms, klientų pasirinkimas tampa grįstas vien tik atstumu – klientai visada renkasi artimiausią objektą.[LAN13]

1.3. Genetinis algoritmas objektų vietų parinkimo uždavinyje

Šio darbo metu spręsti objektų vietų parinkimo uždaviniui naudosiu genetinį algoritmą, kuris yra efektyvus būdas spręsti įvairias sudėtingas problemas. Šis algoritmas nėra naujas, todėl gali kilti dvejonių, ar šis algoritmas tikrai veikia tinkamai ir gali efektyviai spręsti objektų vietų parinkimo problemą. Literatūroje [LFPZ15] galima rasti bandymų, kuriuose buvo sprendžiamas objektų vietų parinkimo uždavinys ir tam naudojamas genetinis algoritmas. Tyrimuose gauti rezultatai parodė, kad genetinis algoritmas geba tinkamai spręsti šį uždavinį bei rasti rezultatus, kurie yra optimalūs arba labai arti optimalaus sprendinio.

Nors genetinis algoritmas siūlo daug efektyvesnį skaičiavimą, tačiau jo sprendinys ne visuomet būna optimalus. Genetinio algoritmo efektyvumas yra rodiklis, nurodantis, kaip dažnai ir kaip arti optimaliausio sprendinio būna genetinio algoritmo rastas sprendinys. Pats efektyvumas tiesiogiai priklauso nuo 3 pagrindinių funkcijų ir joms parinkto algoritmo.

2. Genetinis algoritmas

Genetinis algoritmas (angl. Genetic Algoritm, GA) yra globaliojo optimizavimo algoritmas. Tai nėra nauja mokslo šaka, moksliniai tyrinėjimai vykdomi nuo 1985 metų. Šis algoritmas labai dažnai pagelbėja uždaviniuose, kuriuos išspręsti pilnuoju perrinkimu pareikalautų per daug resursų ir laiko.

Genetinis algoritmas yra populiacinis algoritmas, kurio vykdymas pradedamas nuo populiacijos P, kurią sudaro N atsitiktiniu būdu sugeneruotų sprendinių. Naudojant selekcijos, kryžminimo ir mutacijos operacijas, sudaroma N elementų vaikų populiacija Q. Iš populiacijų P ir Q sąjungos atrenkama N geriausių sprendinių kurie bus naudojami kaip populiacija kitoje algoritmo iteracijoje, kuri vadinama genetinio algoritmo generacija. Toks procesas kartojamas kol nepasiekiamas sustojimo kriterijus, kuris, dažniausiai būna grįstas maksimaliu tikslo funkcijų perskaičiavimų ar generacijų skaičiumi.

Genetinis algoritmas turi keletą savo standartinių parametrų, kurių reikia norint panaudoti genetinį algoritmą. Genetinio algoritmo kintamieji:

- **Generacijos numeris (žym. k)** - kintamasis aprašantis esamos generacijos numerį.
- **Tėvų populiacija (žym. $P^{(k-1)}$)** - N dydžio atsitiktinių sprendinių (individų) aibė, nuo kurios pradedamas sprendimas. Taip pat ir tolimesnių kartų populiacijos dydis.
- **Generacijų skaičius** - skaičius nurodantis, kiek kartų bus vykdomas evoliucijos ciklas.
- **Tikslo funkcija** - kriterijus palyginti vieną sprendinį su kitu.

Šie duomenys yra reikalingi bet kokiam genetinio algoritmo veikimui. Iš paklausos taškų I išrenkama tėvų populiacija $P^{(k-1)}$. Apskaičiuojamas kiekvieno tėvo lyginimo kriterijus pagal nurodytą tikslo funkciją. Tuomet su tėvų populiacija atliekami trys pagrindiniai veiksmai: tėvų parinkimas, kryžminimas bei mutacija. Šie veiksmai skirti iš esamos tėvų populiacijos $P^{(k-1)}$ sukurti naują kartą $P^{(k)}$. Apskaičiuojamas kiekvieno naujos kartos objekto lyginimo kriterijus. Tuomet $P^{(k-1)}$ bei $P^{(k)}$ objektai sudedami į vieną aibę, kurioje yra $2N$ objektų. Iš jos išrenkami N geriausi objektai, pagal lyginimo kriterijų, kurie tuomet tampa naująja tėvų populiacija. Padidinamas iteracijos skaičius k ir ciklas prasideda iš naujo su naujų tėvų rinkimu kryžminimui. Ciklas vykdomas tiek kartų, kiek būna nurodytas generacijų skaičius. [HOL75]

Trys pagrindinės ir tiriamos genetinio algoritmo funkcijos:

- **Tėvų parinkimas** - naudojant lyginimo kriterijų išrinkti objektus (tėvus) kryžminimui.
- **Kryžminimas** - dviejų skirtingų objektų panaudojimas, kuriant naują/naujus objektus.
- **Mutacija** - naujo sprendinio genų pakitimas.

Šios funkcijos yra vykdomos kiekvienos generacijos metu ir turi labai daug skirtingų vykdymo būdų. Šio darbo iškeltieji tyrimų klausimai bus susiję su šiomis funkcijomis, jų panaudojimu ir tyrimu, kokios konstantos geriausios ir koks būdas duoda geriausius rezultatus.

2.1. Tėvų parinkimas

Viena iš pagrindinių genetinio algoritmo užduočių - tinkamai parinkti tėvus. Tėvų parinkimą nulemia lyginimo kriterijus. Jis naudojamas tėvus surikiuoti į gerėjančią arba prastėjančią eilę, su kuria tuomet gali būti atliekami įvairūs atrankos algoritmai, priskiriamos įvairios reikšmės ar kitaip atsitiktiniu būdu vykdoma atranka. Kad ir koks algoritmas būtų pasirinktas, tėvai, su geresniu vertinimo kriterijumi turi didesnę tikimybę būti išrinkti kitos generacijos kūrimui. [DAR94]

Tėvų parinkimo funkcija turi būti labai gerai apgalvota prieš pradedant spręsti konkretų uždavinį. Kai kurie tėvų parinkimo algoritmai specifiniais atvejais gali būti beverčiai, kaip pavyzdžiui situacija, kai vertinimo kriterijaus skirtumai yra labai nedideli, o algoritme vertinimo kriterijus tiesiogiai nulemia jo parinkimo dažnį. Tuomet visi tėvai turės beveik vienodą tikimybę būti išrinkti ir bet koks neapgalvotas tėvų parinkimo funkcijos panaudojimas beveik niekuo nesiskirs nuo atsitiktinio tėvų parinkimo.

2.1.1. „Roulette Wheel”

Šis plačiai naudojamas tėvų parinkimo būdas yra dažnai naudojamas įvairaus tipo uždaviniams spręsti. Jo vykdymas prasideda nuo to, kad kiekvienas kartos objektas gauna savo atitinkamą dalį ruletės rate arba tiesiog atitinkamą atkarpą nuo 0 iki 1 priklausomai nuo to, koks yra jo apskaičiuotas tikslo funkcijos rezultatas. Visų generacijoje dalyvaujančių dalyvių rezultatai sudedami ir kiekvienam dalyviui priskiriama proporcinga dalis, kad išdalinus visiems dalyviams ratas būtų pilnai užpildytas. Tuomet ratas sukamas arba tiesiog generuojamas atsitiktinis skaičius nuo 0 iki 1 ir žiūrima, į kokį intervalą patenka gauta reikšmė. Individas, kuriam priklauso išrinktas intervalas tampa to rinkimo tėvų. Taip iš eilės išrenkamos visos tėvų poros.

2.1.2. „Tournament Selection”

Kitas taip pat gerai žinomas algoritmas, kuris dar dažnai vadinamas „K-Tournament Selection” reikalauja vieno atsitiktinai sugeneruoto arba prieš tai nurodyto skaičiaus K. Algoritmo veikimas prasideda nuo to, kad visi generacijos nariai turi vienodą tikimybę būti išrinkti, kaip vieni iš K narių. Kai pirmoji atranka įvykdoma, tuomet iš likusiųjų narių išrenkamas vienas su geriausiu tikslo funkcijos rezultatu. Šis algoritmas yra mažiau įtakojamas situacijų, kai tėvai turi labai panašius tikslo funkcijos rezultatus, nes prie tikimybių prisiriša tik antrame etape, tad tėvų parinkimas visuomet būna įvairesnis.

2.2. Kryžminimas

Antroji pagrindinė genetinio algoritmo funkcija - objektų kryžminimas ir naujų objektų sukūrimas panaudojant iš tėvų gautus genus. Atrinktos tėvų poros yra įvairiai sukryžminamos norint gauti naujus palikuonis panašius į tėvus. Kryžminimas labai glaudžiai susijęs nuo to, kokie yra tėvai. Dažniausiai, tėvai būna aprašomi kelių genų aibe. Tuomet keli genai yra paimami iš vieno tėvo, keli iš kito ir taip gaunamas naujas individas. Kaip ir kurie genai parenkami iš tėvų tai yra kryžminimo funkcijos uždavinys. [US15]

Kryžminimas, skirtingai nei tėvų parinkimas, blogai veikti negali. Egzistuoja kitos problemos, susijusios su kryžminimu, kad jei genų nėra labai daug, gali gautis identiški vaikai. Kuo daugiau identiškų vaikų, tuo mažiau yra išnagrinėjama sprendinių geriausiojo sprendinio link.

2.2.1. „K-Point Crossover"

Vienas iš labiausiai paplitusių ir turbūt paprasčiausių algoritmų norint sukryžminti du individus. Šiuo būdu iš dviejų tėvų sukuriami du individai. Algoritmas prasideda pasirenkant K taškų, ties kuriais bus apsikeičiama tėvų genais, t.y. naujam individui sukurti yra naudojami pirmojo arba antrojo tėvo genai iki pirmojo taško, tuomet imami kito tėvo genai iki kito taško ir taip tęsiama kol pereinami visi genai. Pirmasis vaikas pradeda nuo pirmojo tėvo, antrasis nuo antrojo, tad jie tiesiog pakaitomis dalinasi tėvų genus. Šiame darbe didžioji dalis eksperimentų bus atliekami su tėvais, kurie turės iki 5 genų, tad bus naudojama paprasčiausia šio algoritmo versija „1-Point Crossover", kai pasirenkamas vienas taškas, ties kuriuo genai bus paimami iš kito tėvo ir taip sukuriami nauji individai.

2.2.2. „Uniform"

Kitas populiarus kryžminimo būdas, kuris dažniausiai labiau sumaišo genus. Šiuo būdu, kaip ir prieš tai aprašytame algoritme, yra sukuriami du individai. Pradedama nuo pirmo geno, ties kuriuo yra generuojamas atsitiktinis skaičius nuo 0 iki 1. Jei tas skaičius didesnis nei 0.5 pirmasis vaikas gauna pirmojo, antrasis antrojo tėvo geną, jei mažesnis - atvirkščiai. Žinoma, tikimybes galima ir apversti, svarbu, kad abu genai turėtų vienodą tikimybę būti išrinkti, ir padalinti skirtingiems vaikams.

2.2.3. „Flat Crossover"

Šis kryžminimo būdas nėra pats efektyviausias atminties atžvilgiu, nes reikalauja dvigubai daugiau tėvų. Šio algoritmo metu iš dviejų tėvų sukuriamas tik vienas vaikas. Algoritmo principas lygiai toks pat, kaip „Uniform" algoritmo, tik šiuo atveju, nekuriamas antrasis individas, o pasirenkamas tik laimėjusio tėvo genas.

2.3. Mutacija

Paskutinis genetinio algoritmo žingsnis - naujosios kartos mutacija. Naujoji individų karta yra leidžiama į mutacijos funkciją, kurioje jau minėti surinkti skirtingi kiekvieno individo genai turi galimybę mutuoti į bet kokį kitą uždavinio kintamuosiuose egzistuojantį geną. Kaip dažnai ir nuo ko priklausys mutacija tai yra mutacijos funkcijos uždavinys.[HOL75]

Mutacija bet kurio uždavinio sprendimui turi didžiulę reikšmę nes leidžia tirti sritis, kurios negali būti gaunamos nei renkant skirtingus tėvus nei bet kaip kryžminant. Mutaciją reikia dalinti į dvi dalis - mutacijos tikimybės strategiją bei pačios mutacijos strategiją. Dažniausiai naudojama mutacijos strategija yra tiesiog mutuoti į bet kokį kitą egzistuojantį geną. Kitos modifikacijos bus nagrinėjamos vėliau.

Mutacijos tikimybės tuo tarpu labiau patyrinėtos, galima rasti ir standartinių ir nestandartinių būdų.

2.3.1. „1/N”

Dažniausias ir turbūt būtų galima vadinti standartinis dydis, parinktas mutacijos tikimybei yra $1/N$, kur N yra genų skaičius. Kitaip tariant algoritmas užtikrina, kad didžiojoje dalyje individų mutuos lygiai vienas genas.

2.3.2. Kitokios tikimybės

Mutacijos tikimybės strategija tai begalinė aibė turinti tiek variantų kiek yra realiųjų skaičių nuo 0 iki 1. Tikrinti visas jas yra beprasmiška ir užtruktų be galo daug laiko. Buvo nuspręsta tyrinėti šiek tiek kitokius literatūroje rastus mutacijos algoritmo būdus, kurie remiasi kintančia mutacijos tikimybe, kaip mažėjanti vis tolimesnėje mutacijoje ar priklausanti nuo tėvų genų [PB14]. Šie įvairesni tikimybių aprašymai, nors nėra nauji, tačiau dėl įvairesnių tikimybių parinkimų bei kitų modifikacijų bus aprašomi vėlesniuose skyriuose kaip tyrimo objektai.

3. Genetinio algoritmo taikymas objektų vietų parinkimo uždavinyje

Pirmajame darbo skyriuje aprašytas uždavinys sprendžia komplikuotą uždavinį, kurį išspręsti tiesioginio perrinkimo būdu kartais gali būti beveik neįmanoma. Genetinis algoritmas anksčiau minėtą 2,59 * 10^{16} funkcijų perskaičiavimų skaičių, esant tokiems patiems parametrams bei pridėjus reikiamus genetinio algoritmo parametrus, pavyzdžiui 100 individų, bei 250 generacijų, rezultatui apskaičiuoti prireiktų „vos"

$$100 * 250 = 25{,}000$$

funkcijų perskaičiavimų. Tai yra apie 1,03 * 10^{12} kartų mažiau, nei skaičiuojant pilno perrinkimo metodu.

3.1. Genetinio algoritmo veikimas objektų vietų parinkimo uždavinyje

Genetinio algoritmo programa nuskaito iš failo visas įmanomas naujas objektų vietas, kurios turi savo koordinates (ilgumą ir platumą), taip pat gyventojų skaičių toje lokacijoje. Iš aibės parenkama E geriausių ir daugiausia gyventojų turinčių vietų ir jos priskiriamos kaip jau egzistuojančios vietos, su kuriomis bus lyginama naujoji generacija. Atsitiktinai iš visų objektų vietų parenkama P naujų grupių, kurios kiekviena turi po N (naujų norimų objektų vietų skaičius) objektų vietų iš visų objektų vietų aibės. Kiekvienai iš P grupei apskaičiuojamas jos vertinimo kriterijus, kiek visos N parinktos vietos surinks gyventojų. Pagal tai jie sustatomi į mažėjimo tvarka einančią eilę. Iš tos eilės, pagal pasirinktą tėvų parinkimo būdą, pasirenkami du tėvai. Tarp jų įvyksta kryžminimas. Kryžminant keičiasi objektų grupės vietos, jei grupėje buvo N skirtingų objektų vietų, naujojoje objektų grupėje bus dalis objektų vietų is vienos tėvinės objektų grupės, dalis iš kitos. Taip gaunamas naujas P grupių sąrašas. Kiekviena iš P grupė turi tikimybę mutuoti vieną ar keletą savo objektų vietų į bet kurį kitą, dar toje grupėje nesančią objektų vietą iš objektų vietų aibės. Užbaigus mutaciją iš naujo apskaičiuojamas naujai gautų objektų vertingumas. Iš viso $2 * P$ objektų - tėvai ir sugeneruoti vaikai. Iš jų pagal lyginimo kriterijų išrenkama P geriausių objektų. Tai galutinis naujosios generacijos sąrašas. Kartos skaičius padidinamas vienetu. Tuomet vėl prasideda genetinio algoritmo ciklas nuo tėvų parinkimo. Taip ciklas tęsiasi tiek kartų, koks buvo nurodytas kartų skaičius. [LFPZ15]

Genetiniam algoritmui reikalingų parametrų sąrašas gali būti matomas 2 skyriuje. Kadangi algoritmas nėra labai paprastas, gali kilti neaiškumų, kaip interpretuoti įvairius uždaviniui bei algoritmui reikalingus parametrus. Čia pateikiu savo uždavinio bei algoritmo parametrų atitikmenis:

- Tėvas - aibė iš N nesikartojančių taškų iš paklausos taškų aibės, kur N yra norimų naujų objektų skaičius. Iš keletos tokių tėvų sudaroma tėvų populiacija.
- Tikslo funkcija - funkcija apskaičiuojanti kiek atitinkamas tėvas surinks gyventojų su savo turimais taškais iš visos tiriamosios taškų aibės.
- Genas - vienas bet kuris taškas/objektas iš tėvo.

- Paklausos taškai - keletas tūkstančių iš failo nuskaitytų objektų, kurie turi savo koordinates, gyventojų skaičių bei kokybės indeksą.

3.2. Genetinio algoritmo modifikacijos

Genetinis algoritmas ypatingas tuo, kad jame nėra konkretaus tikslumo ir visuomet egzistuoja tikimybė rasti kažką dar geresnio. Būtent todėl šis algoritmas leidžia spręsti užduotis, kurios paprastais skaičiavimais gali būti beveik neįmanomos. Labai dažnai standartinės genetinio algoritmo funkcijos nėra prisitaikiusios spręsti konkretų uždavinį. Kaip jau anksčiau buvo minėta, kai tėvų parinkime tikslo funkcijų rezultatai yra labai panašūs, ar kryžminime naudojama tiek mažai genų, kad labai daug vienodų vaikų. Tuomet sprendžiant konkrečią problemą gali būti pasiūlytos įvairios modifikacijos, o kartais net visiškai kitoks funkcijos panaudojimas įvykdyti tėvų parinkimą, kryžminimą ar mutaciją. Šiame darbe bus bandoma rasti potencialių funkcijų modifikacijos būdų, kurie rastų geresnį rezultatą, nei naudojant standartines funkcijas.

3.2.1. Tėvų parinkimo modifikacija

Kaip ir genetinio algoritmo veikimas, analizė buvo pradėta nuo tėvų parinkimo uždavinio. Paanalizavus konkretų uždavinį galima gan greitai pastebėti tam tikrus dalykus apie gautus rezultatus. Objektų vietų parinkimo uždavinyje, individų tikslo funkcijų rezultatai yra gan panašūs, ypač vėlesnėse kartose. Tai nulemia, kad „Roulette Wheel" naudojimas gali tapti ne toks pranašus. Taip pat „Tournament Selection" iš esmės truputį per daug šokinėja generacijų pradžioje, nes tinkamiau panaudojant tikslo funkcijos rezultatus, būtų galima tiksliau ir greičiau rasti geresnį sprendinį. Šios dvi sąlygos padėjo sugalvoti naują algoritmo modifikaciją, kuri turėtų efektyviau ieškoti uždavinio sprendinio. Algoritmas 1 pradedamas parenkant pirmojo nario parinkimo tikimybę q bei mažėjimo konstantą u.Tuomet visi nariai išrikiuojami jų tikslo funkcijos mažėjimo tvarka. Iš eilės traukiamas kiekvienas narys. Apskaičiuojama n-tojo nario tikimybė būti paimtam, t.y. $q * u^{(n-1)}$. Sugeneruojamas atsitiktinis skaičius nuo 0 iki 1 ir jei tas skaičius mažesnis nei n-tojo nario tikimybė, n-tasis narys tampa tėvu. Ciklas iš naujo nepradedamas, o yra tęsiamas ties paskutiniu parinktu tėvu. Šis algoritmas kintamųjų pagalba leidžia prisiderinti prie įvairių sprendžiamo uždavinio rezultatų, jei pradžioje tikslo funkcijos rezultatai yra dideli jis juos sulygina, jei jie per maži, jis juos padidina iki numatyto vidurkio priklausomai nuo to, kokios buvo pasirinktos reikšmės. Šio algoritmo tyrimas tikėtina turėtų atnešti teigiamų rezultatų ieškant geriausiojo sprendinio.

1 algoritmas: Parent Selection

Require: $N \geq 0, q = 0.1, u = 0.05$

1: $ParentPairs \Leftarrow \{ParentIndex, ParentIndex\}$
2: $CurrentParentPair = 0$
3: **for** $i = 1,2,\ldots,N$ **do**
4: $currentOdd = q * u^{(i-1)}$
5: $rand := genRandomDouble[0,1]$
6: **if** $currentOdd < rand$ **then**
7: **if** $ParentPairs\,[CurrentParentPair]\,.first \Rightarrow Null$ **then**
8: $ParentPairs\,[CurrentParentPair]\,.first = N$
9: **else if** $ParentPairs\,[CurrentParentPair]\,.first \neq N$ **then**
10: $ParentPairs\,[CurrentParentPair]\,.second = N$
11: **if** $CurrentParentPair = N/2$ **then**
12: **return**$ParentPairs$
13: **end if**
14: $CurrentParentPair = CurrentParentPair + 1$
15: **end if**
16: **end if**
17: **end for**

3.2.2. Kryžminimo modifikacija

Objektų vietos parinkimo uždavinyje kryžminimas neatlieka labai didelės reikšmės pokyčio, nes kiekvieno individo lygmenyje nėra svarbu, kokia tvarka išdėlioti parinkti taškai, t.y. aibė objektų numerių iš sąrašo 1, 2, 3, 4, 5 duos lygiai tokį patį rezultatą kaip aibė 5, 4, 3, 1, 2. Taip yra todėl, nes kiekvienas populiacijos elementas yra iškart lyginamas su visu objektu ir taip randamas pritrauktų vartotojų skaičius. Genai yra taškai, kaip miestai ar rajonai ir nesvarbu ar jis sąraše buvo pirmas ar trečias, gyventojas renkasi pagal atstumą. Jei atstumai vienodi - taip pat nesvarbu, nes visi gyventojai sėkmės atveju, kai atstumas bus geresnis nei konkurento, vis tiek priklausys to pačio objekto bendrai sumai. Kiekvieno geno rezultatas atskirai nėra aktualus. Tuo tarpu dažniausiai genetiniame algoritme naudojami kryžminimai stipriai siejasi su konkretaus geno pozicija pavyzdžiui:

- Į pasirinktą poziciją parenkamas genas iš pirmo arba antro tėvo tos pačios pozicijos. („Uniform")
- Pasirenkama taškas arba taškai ties kuriais bus apsikeičiama genais. Kiekvienas taškas reiškia, kad nuo to taško pozicijos bus imama kito tėvo genai iki artimiausio kito nurodyto taško.(„K-Point Crossover")
- Imami du genai iš tos pačios tėvų genų pozicijos ir iš jų išvedamas vidurkis.("Average Crossover")

- Tėvų genai sumaišomi atsitiktine tvarka, tačiau tokiu pačiu maišymu, kad kryžminant genai atitiktų. („Shuffle Crossover")

Nors genų tvarkos išsaugojimas sprendinių ieškojimui nekenkia, tačiau yra visiškai neprasmingas, todėl buvo nuspręsta sugalvoti algoritmus, kurie visiškai nesisietų su genų pozicija. Pirmasis algoritmas 2 iš dviejų tėvų būtų sukuriamas vienas vaikas taip, kad būtinai bent vienas, arba nurodytas skaičius atsitiktinių genų g iš tėvo su geresniu tikslo funkcijos rezultatu būtų panaudotas naujojo individo kūrime. Likęs skaičius tuščių genų būtų atsitiktine tvarka paimamas iš kito tėvo. Taip sukurti individai išlaikytų geresniųjų genų kiekį ir galėtų greičiau priartėti prie geriausiojo rezultato.

Kitas labai panašus algoritmas 3 taip pat iš dviejų tėvų sugeneruotų vieną, tačiau šiuo atveju, būtų vykdomas „Flat Crossover", kuriame tikimybė q parinkti geną iš geresniojo tėvo būtų truputį didesnė, nei iš kito tėvo.

Abu atvejai remiasi geresniojo tėvo genų parinkimo prioritetu, todėl tiesiog bus priskirti tai pačiai tyrinėjimo grupei. Jei vienas iš algoritmų pasirodytų veikiantis prasčiau, jo tyrinėjimai nebus tęsiami.

2 algoritmas: First Crossover

Require: $firstParent \Leftarrow \{\}, secondParent \Leftarrow \{\}, g \geq 1$
1: $Child \Leftarrow \{\}$
2: $takenGenes \Leftarrow \{\}$
3: **for** $i = 1,2,\ldots,g$ **do**
4: $randomGene = genRandomInt(1, firstParent.Genes.Count)$
5: **if** $takenGenes.Contains(randomGene)$ **then** i = i - 1
6: **else**takenGenes.insert(randomGene) child.insert(firstParent.Genes[randomGene])
7: **end if**
8: **end for**
9: $takenGenes \Leftarrow \{\}$
10: **for** $i = 1,2,\ldots,firstParent.Genes.Count - g$ **do**
11: $randomGene = genRandomInt(1, secondParent.Genes.Count)$
12: **if** $takenGenes.Contains(randomGene)$ **then** i = i - 1
13: **else**takenGenes.insert(randomGene) child.insert(secondParent.Genes[randomGene])
14: **end if**
15: **end for**

3 algoritmas: Second Crossover

Require: $firstParent \Leftarrow \{\}, secondParent \Leftarrow \{\}, q \geq 0.5$

```
Child ⇐ {}
for i = 1,2,...,firstParent.Genes.Count do
    rand = genRandDouble(0, 1)
    if q ≥ rand then
        Child.insert(firstParent.Genes[i])
    else
        Child.insert(secondParent.Genes[i])
    end if
end for
```

3.2.3. Mutacijos tikimybės

Kaip ir buvo minėta anksčiau, mutacijos skyriuje bus naudojamos jau žinomų strategijų karkasai, tačiau panaudojamos skirtingos originalios tikimybių apskaičiavimų reikšmės. Tai leis kuo labiau pririšti tyrinėjamą uždavinį prie genetinio algoritmo.

Pirmasis kintančios mutacijos tikimybės būdas, kuris bus tiriamas yra prie tėvų prisitaikanti mutavimo tikimybė, pavyzdžiui sudėjus tėvų vertes ir padalinus iš visų tėvų įverčių sumos. Algoritmo 4 veikimas pasikeičia tuo, kad dabar kiekvienas individas kryžminimo metu gauna naują kintamąjį - mutacijos tikimybę, pagal kurią mutacijos metu bus mutuojami jo genai. Bus išbandomos kelios strategijos, kur vidutinė tikimybė gali likti 1/N, bet priklausomai nuo tėvų tikslo funkcijos rezultato mažėti arba didėti. Kitas variantas neprisirišti prie 1/N, tačiau visą tikimybės radimą palikti tiesiogiai priklausomą nuo tėvų. Taip pat reikės parinkti tikimybių rėžius, nes šiuo atveju dažniausiai naudojama tikimybė 1/N nebus kaip rodiklis, dėl kintamos tikimybių reikšmės ir bus bandoma rasti geriausiai veikiančius rėžius ir tėvų nulemiamą pokytį. Vienas ar keli geriausiai rasti sprendiniai bus lyginami su standartiniu 1/N būdu.

Antrasis kintančios mutacijos tikimybės būdas, tai progresyviai mažėjanti (kai kuriais atvejais didėjanti) mutacijos tikimybė. Taip pat kaip ir pirmasis siūlytas algoritmas, šis algoritmas 5 reikalaus tam tikrų rėžių, bei mažėjimo greičio. Bus tiriami tiesinis, kvadratinis bei logaritminis tikimybės pokytis priklausantis nuo einamosios kartos ir taip pat išrinkta vienas ar keli geriausiai veikiantys tyrimams lyginti. Skirtingai nei pirmajame algoritme, čia nereikės saugoti papildomų kintamųjų kiekvieno vaiko atmintyje, tik vieną kartą kartos mutavimo metu apskaičiuoti tikimybę, kuri bus taikoma visiems kartos nariams. Tad jei šis algoritmas veiktų geriau ar net lygiai taip pat nei pirmasis, jis būtų pranašesnis.

Paskutinė, ”Distance dependent” strategija, kurioje mutuojama į genus, kurių tikimybės yra priklausomos nuo atstumo iki geno, kuris mutuos, nėra visiškai žinoma ir ištyrinėta, tačiau galima rasti jos tyrinėjimo pavyzdžių literatūroje [FLPŽ17]. Ši strategija taip pat bus tyrinėjama ir bandoma ją modifikuoti, bandant gauti geresnius rezultatus. Šie tyrimai bus labiau pagrįsti ne tik mutacijos tikimybės, bet ir mutacijos strategijos valdyme, kai naudojama „Distance Dependent” strategija. Pirmuosiuose tyrimuose tiriama tiesiog šios strategijos naudingumas naudojant ją kartu

su „1/N” strategija. Jei ji pasiteisins, bus bandoma ją modifikuoti pritaikant kitas siūlomas strategijas bei stebėti rezultatus, kokią įtaką mutacijos tikimybės parinkimas turi „Distance Dependent” mutacijos strategijai. Pavyzdys, kaip galėtų būti taikoma strategijos modifikacija kartu su kitomis mutacijos tikimybės parinkimo strategijos gali būti matomas algoritmas 6.

4 algoritmas: Parent Dependent Mutation

Require: $Child \Leftarrow \{\}$

1: $Child.MutateProbabilty \Leftarrow CalculateInCrossover()$
2: **for** $i = 1,2,\ldots,Child.Genes.Count$ **do**
3: $rand = genRandDouble(0,1)$
4: **if** $Child.MutateProbabilty \geq rand$ **then**
5: $Child.Genes[i] \Leftarrow Mutate()$
6: **end if**
7: **end for**

5 algoritmas: Generation Dependent Mutation

Require: $Child \Leftarrow \{\}, G \Leftarrow CurrentGenerationNumber$

1: $currentMutationOdd = GetCurrentMutationOdd(G)$
2: **for** $i = 1,2,\ldots,Child.Genes.Count$ **do**
3: $rand = genRandDouble(0,1)$
4: **if** $currentMutationOdd \geq rand$ **then**
5: $Child.Genes[i] \Leftarrow Mutate()$
6: **end if**
7: **end for**

6 algoritmas: Distance Dependent Strategy Modification

Require: $Child \Leftarrow \{\}$

1: $currentMutationOdd = AnyAlgorithmToDecideMutationOdd()$
2: **for** $i = 1,2,\ldots,Child.Genes.Count$ **do**
3: $rand = genRandDouble(0,1)$
4: **if** $currentMutationOdd \geq rand$ **then**
5: $distancedStrategyOdd = getRandDouble(0,1)$
6: **if** $distancedStrategyOdd \geq genRandDouble(0,1)$ **then**
7: $Child.Genes[i] \Leftarrow MutateWithDistancedStrategy()$
8: **else**
9: $Child.Genes[i] \Leftarrow MutateWithStandardStrategy()$
10: **end if**
11: **end if**
12: **end for**

4. Eksperimentiniai tyrimai

Šis skyrius skirtas peržvelgti, palyginti bei analizuoti rezultatus, gautus eksperimentinių tyrimų metu. Visi tyrimai, išskyrus optimaliausio sprendinio paieška, bus atliekami 100 kartų pakartojant leidžiamą scenarijų ir lyginamas vidutinis duomenų rezultatas, kada vidutiniškai buvo rastas optimalus sprendinys bei aprašomos kitos pastabos. Genetinio algoritmo naudojamos konstantos:

- Tėvų populiacija - 100
- Generacijų skaičius - 250

Tyrimų tikslas - rasti geriau veikiančias funkcijas genetiniam algoritmui išspręsti objektų vietų parinkimo uždavinį. Kitos tyrimų konstantos:

- Skaičius norimų naujų objektų vietų - n
- Potencialios vietos - p
- Paklausos taškai - I
- Egzistuojantys objektai - e

Norint tinkamai palyginti skirtingus algoritmus, reikia užtikrinti vienodas sąlygas kiekvieno eksperimento metu. Tam reikia pasirinkti vieną algoritmą, kiekvienai genetinio algoritmo funkcijai. Kaip standartiniai algoritmai pasirinkti šie:

- Tėvų parinkimas - „Roulette Wheel” (2.1.1)
- Kryžminimas - „K-Point Crossover” (2.2.1)
- Mutacija - „1/N” (2.3.1)

Su kiekvienu duomenų komplektu bus vieną kartą atliekama optimaliausio sprendinio paieška. Kadangi kai kurie skaičiavimai gali trukti ypač ilgai, buvo naudojamas paskirstytų skaičiavimų tinklas, kuriame naudojami 16 branduolių. Išlygiagretinus skaičiavimus reikalingas laikas sumažėdavo keliolika kartų, todėl skaičiavimai tapdavo įmanomi. Šie rezultatai bus pateikiami kiekvieno tyrimo pradžioje. Su optimaliausiu sprendiniu bus lyginami genetinio algoritmo gauti sprendiniai, jei pasiekė - kada, jei nepasiekė, koks buvo nuokrypis nuo geriausiojo sprendinio.

Tyrimu metu bus siekiama išlaikyti protingą mastelį t.y. jei nuo tam tikros generacijos sprendiniai nesikeičia arba pirmųjų generacijų rodmenys nerodo naudingos informacijos ji bus praleidžiama ir vaizduojamas tik dalinis informacinis grafikas.

Kiekvienas tyrimas bus išbandytas su abiem aprašytais skyrelyje 1.2.1 klientų elgsenos modeliais - binariniu bei proporciniu.

4.1. Tyrimas nr. 1: $n = 3, p = 100, I = 5000, e = 5$

4.1.1. Optimalus sprendinys

Su pirmaisiais duomenimis optimalus sprendinys pagal klientų elgsenos modelį:

$$Binarinis - 751154, Proporcinis - 881151$$

4.1.2. Tėvų parinkimas

Pirmajame tyrime buvo ištirta, kaip skirtingi tėvų parinkimo algoritmai veikia genetinio algoritmo rezultatą. Ištirti 3 darbe minėti algoritmai. Binarinio bei proporcinio modelio rezultatus galima matyti atitinkamai paveikslėliuose 1a bei 1b.

Kadangi sunku algoritmus įvertinti pagal vieną kriterijų, nes skirtingi algoritmai gali geriau veikti pradžioje arba pabaigoje, jų palyginimui naudosiu lenteles, kuriose galima matyti kokioje generacijoje algoritmas vidutiniškai jau buvo radęs sprendinį su tam tikra paklaida nuo optimalaus sprendinio. Skliausteliuose matomi rėžiai tarp mažiausios bei didžiausios pasitaikiusios reikšmės, kad būtų galima įvertinti rezultatų išsibarstymą.

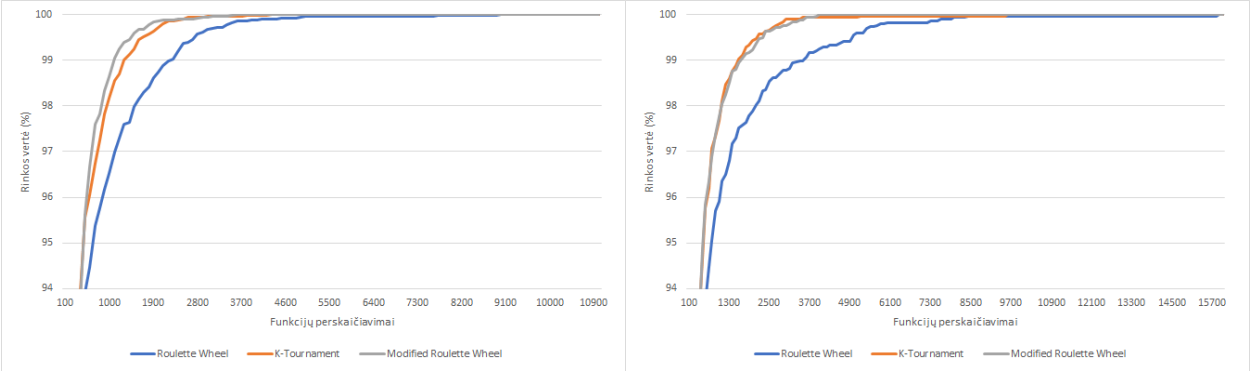
Binarinis

Alg. / Pakl.	**Roulette Wheel**	**K-Tournament**	**Modified Roulette Wheel**
1%	19.8 (3; 48)	12.14 (1; 42)	9.7 (3, 19)
0.1%	33.7 (8; 105)	16.6 (2; 42)	16.76 (3; 42)
0%	33.7 (8; 105)	16.6 (2; 42)	16.76 (3; 42)

Proporcinis

Alg. / Pakl.	**Roulette Wheel**	**K-Tournament**	**Modified Roulette Wheel**
3%	24 (1, 36)	6 (3; 23)	9 (2; 33)
1%	37.22 (6, 158)	17.26 (3; 95)	16.22 (3; 39)
0.1%	37.22 (6; 158)	17.26 (3; 95)	16.22 (3; 39)
0%	37.22 (6; 158)	17.26 (3; 95)	16.22 (3; 39)

Iš rezultatų galima matyti, jog du algoritmai veikė labai panašiai. Binariniame modelyje geriau pradėjo trečiasis algoritmas, tačiau prieš pabaigą susilygino su antruoju. Proporciniame modelyje nors ir antrasis algoritmas laikėsi pirmesnis iki labai artimo sprendinio radimo, galutinį sprendinį daug greičiau vidutiniškai rado trečiasis algoritmas.



(a) Binarinis (b) Proporcinis

1 pav. Tėvų parinkimo palyginimas Nr. 1

4.1.3. Kryžminimas

Ištyrus 5 darbe minėtus kryžminimo algoritmus, gauti binarinio bei proporcinio modelio rezultatai yra matomi atitinkamai paveiksliuose 2a bei 2b.

Algoritmams palyginti vėl bus naudojamos optimaliojo sprendinio paieškos paklaidos lentelės.

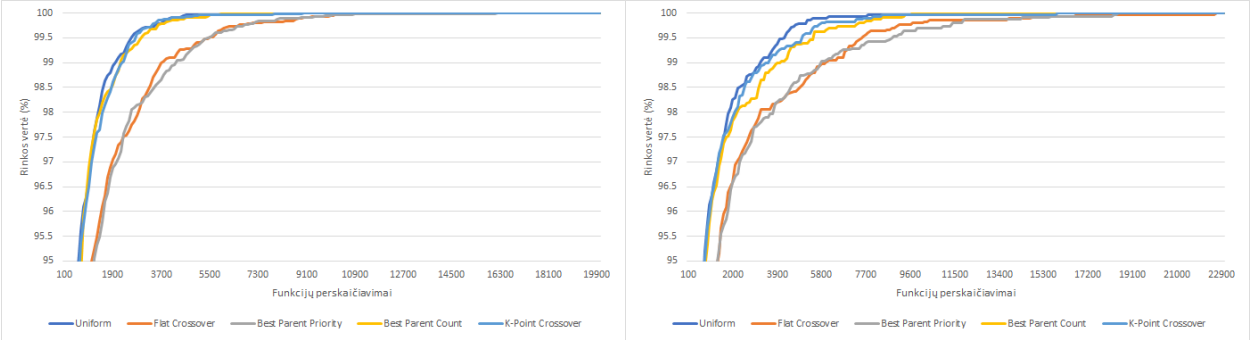
Binarinis

Alg. / Pakl.	**Uniform**	**Flat**	**B.P. Priority**	**B.P. Count**	**K-Point**
1%	18.52 (4; 44)	37.32 (4; 92)	38.64 (7; 88)	19.06 (2; 45)	19.8 (3; 48)
0.1%	30.2 (4; 133)	62.74(19; 173)	60.16 (7; 192)	32.58 (2; 82)	33.7 (8; 105)
0%	30.2 (4; 133)	62.74(19; 173)	60.16 (7; 192)	32.58 (2; 82)	33.7 (8; 105)

Proporcinis

Alg. / Pakl.	**Uniform**	**Flat**	**B.P. Priority**	**B.P. Count**	**K-Point**
3%	12.5 (4; 40)	14 (5; 74)	11 (1; 65)	14 (1; 43)	24 (1; 36)
1%	32.8 (5; 102)	63.44 (7; 226)	68.84 (10; 184))	43.56 (1; 96)	37.22 (6; 158)
0.1%	32.8 (5; 102)	63.44 (7; 226)	68.84 (10; 184))	43.56 (1; 96)	37.22 (6; 158)
0%	32.8 (5; 102)	63.44 (7; 226)	68.84 (10; 184)	43.56 (1; 96)	37.22 (6; 158)

Akivaizdžiai matomi du prasčiausi algoritmai - „Flat Crossover” bei „Best Parent Priority”. Kiti algoritmai veikė panašiai. Beveik visuose tyrimuose pirmavo „Uniform” algoritmas. Tolimesniems tyrimams bus paimti 3 geriausi algoritmai.



(a) Binarinis (b) Proporcinis

2 pav. Kryžminimų palyginimas Nr. 1

4.1.4. Mutacija

Atliktas tyrimas su 3 darbe minėtomis mutacijų strategijomis. Binarinio bei proporcinio bandymo rezultatus galima matyti atitinkamai paveikslėliuose 3a ir 3b.

Algoritmų palyginimui naudojamos optimaliojo sprendinio paieškos paklaidos pagal generaciją lentelės.

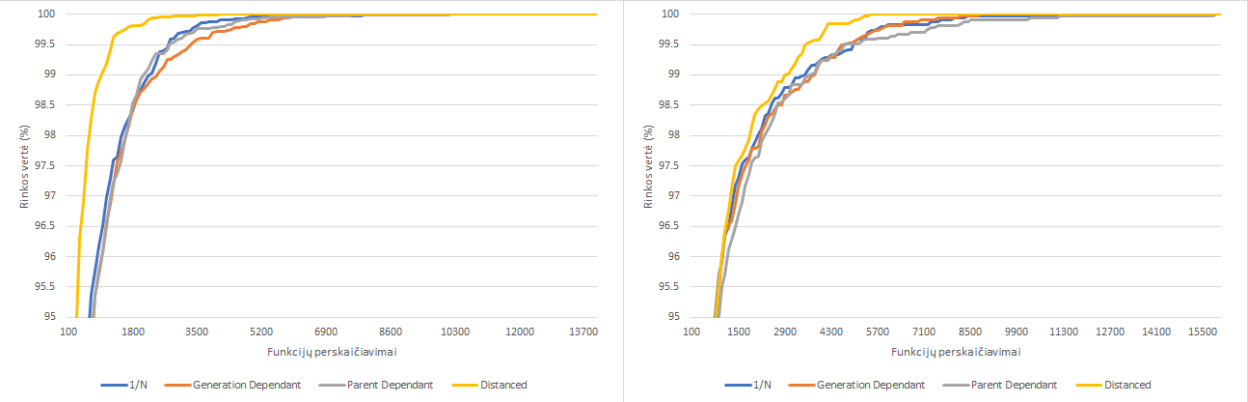
Binarinis

Alg. / Pakl.	**1/N**	**G. Lowering**	**P. Dependent**	**Distance Dependent**
1%	19.8 (3; 48)	22.12 (2; 64)	20.46 (3; 48)	8.85 (1; 21)
0.1%	33.7 (8; 105)	40.2 (5; 100)	38.38 (8; 129)	14.04 (2; 54)
0%	34.7 (9; 106)	40.2 (5; 100)	38.38 (8; 129)	14.04 (2; 54)

Proporcinis

Alg. / Pakl.	**1/N**	**Lowering**	**Dependent**	**Distance Dependent**
3%	24 (1, 36)	30 (3; 39)	14.5 (2; 37)	11.72 (3; 27)
1%	37.22 (6, 158)	38.26 (5; 87)	43.22 (2; 158)	23.88 (4; 79)
0.1%	37.22 (6; 158)	38.26 (5; 87)	43.22 (2; 158)	23.88 (4; 79)
0%	37.22 (6; 158)	38.26 (5; 87)	43.22 (2; 158)	23.88 (4; 79)

Prasčiausiai veikiantis algoritmas su duotais parametrais - „Parent Dependent”. Kiti du algoritmai su standartinėmis mutacijos strategijomis veikė panašiai. Nors „1/N” algoritmas vidutiniškai šiek tiek dažniau rasdavo geresnius sprendinius, tačiau jo reikšmės beveik visada labiau išsimėčiusios. Absoliučiai geriausiai veikė algoritmas, kuriame buvo naudojama „Distance Dependent” strategija - mažiausias reikšmių išsimėtymas bei greičiausiai rastas optimalus sprendinys.



(a) Binarinis

(b) Proporcinis

3 pav. Mutacijų palyginimas Nr. 1

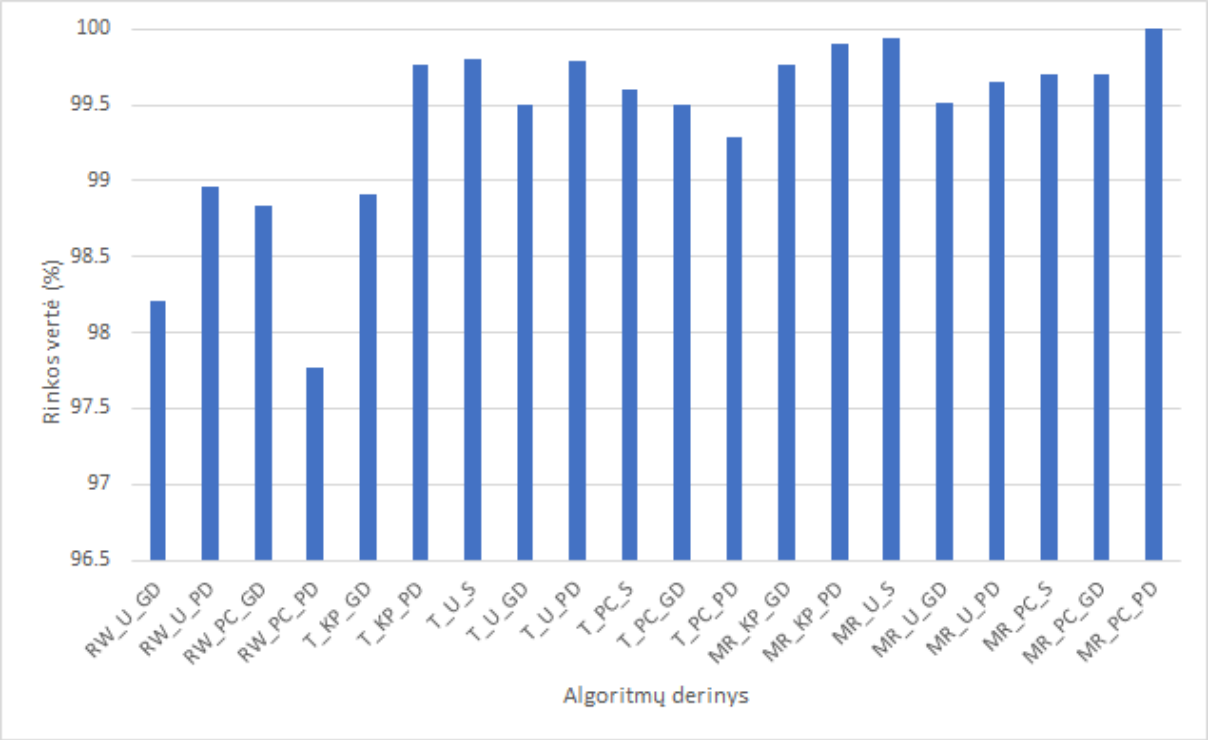
4.1.5. Kiti strategijų deriniai

Atlikus individualius tyrimus ir atrinkus keletą geriausių algoritmų atlikau bandymus su įvairiais geriausių algoritmų deriniais, kad išsiaiškinti, kuris derinys pasiektų geriausius rezultatus. Kol kas įvairių derinių bandymuose nedalyvauja mutacija su „Distance Dependent" strategija, nes ji bus tyrinėjama vėliau.

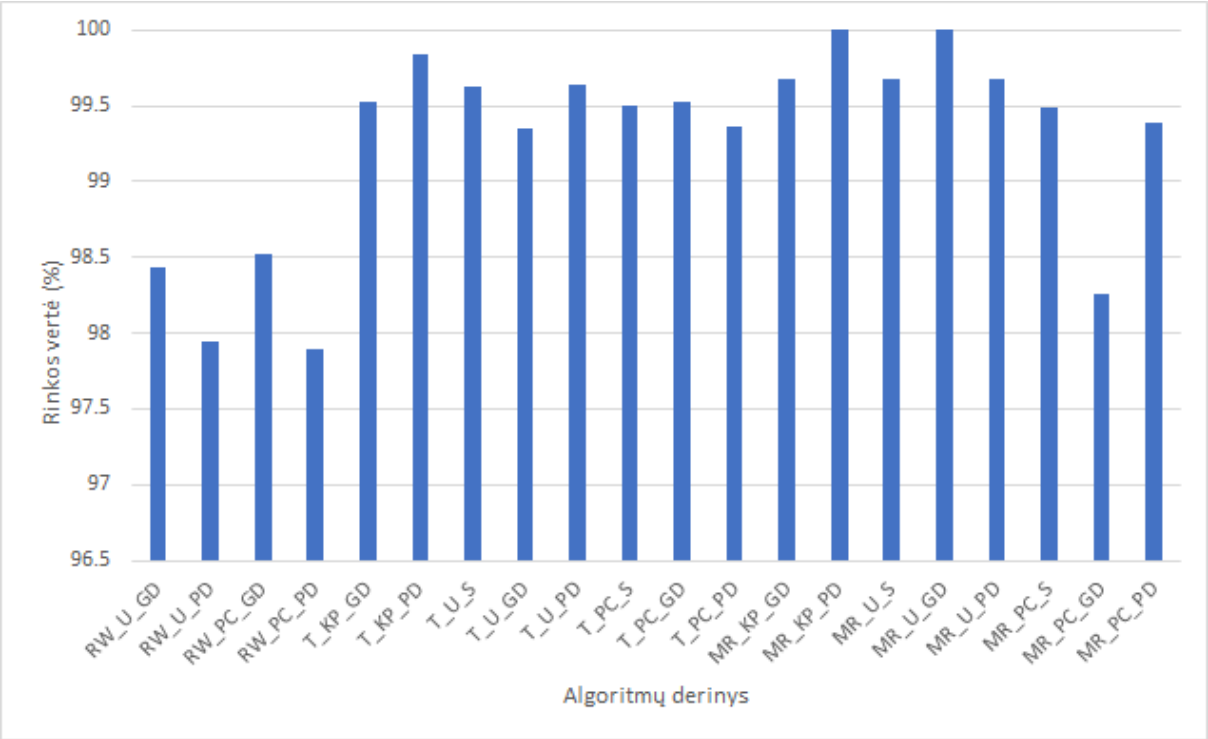
Šie bandymai vėlgi buvo leisti po 100 kartų, surastas jų vidutinis rezultatas. Rezultatai atvaizduoti diagramose, kurias galima matyti prieduose. Kadangi derinių yra nemažai, žymėti strategiją bus naudojami pavadinimų trumpiniai.

- RW - Roulette Wheel
- T - Tournament
- MR - Modified Roulette Wheel
- U - Uniform
- KP - K-Point Crossover
- F - Flat Crossover
- PP - Better Parent Priority
- PC - Better Parent Genes Count
- S - 1/N (Standard)
- GD - Generadion Dependent Mutation
- PD - Parent Dependent Mutation

Gautuose tyrimuose, binarinio (Priedas nr. 1, Priedas nr. 2, Priedas nr. 3, Priedas nr. 4, Priedas nr. 5) ir proporcinio (Priedas nr. 6, Priedas nr. 7, Priedas nr. 8, Priedas nr. 9, Priedas nr. 10), paveikslėliuose esančiuose prieduose, galima matyti įvairių derinių grafikus. Paveikslėliuose (binarinis 4 ir proporcinis 5) galima matyti bendrą algoritmų vaizdą, kai pirmasis derinys buvo suradęs optimalųjį sprendinį.



4 pav. Įvairių derinių tyrimas Nr. 1 (binarinis)



5 pav. Įvairių derinių tyrimas Nr. 1 (proporcinis)

4.2. Tyrimas nr. 2: $n = 5, p = 100, I = 5000, e = 5$

4.2.1. Optimalus sprendinys

Antrajame tyrime optimalus sprendinys yra:

$$Binarinis - 1133054, Proporcinis - 1183861$$

4.2.2. Tėvų parinkimas

Vėl tirti 3 algoritmai. Rezultatai yra matomi paveikslėliuose 6a bei 6b.
Lentelės algoritmų palyginimui:

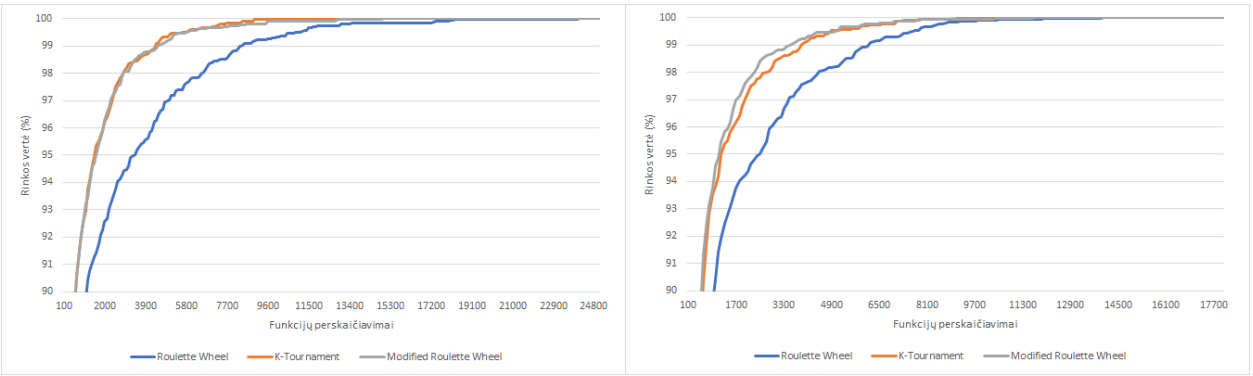
Binarinis

Alg. / Pakl.	**Roulette Wheel**	**K-Tournament**	**Modified Roulette Wheel**
1%	80.52 (30; 182)	37.42 (3; 91)	38.82 (10; 149)
0.1%	99.8 (35; 250)	45.92 (3; 106)	52.14 (10; 152)
0%	99.8 (35; 250)	45.92 (3; 106)	52.14 (10; 152)

Proporcinis

Alg. / Pakl.	**Roulette Wheel**	**K-Tournament**	**Modified Roulette Wheel**
3%	40.6 (9; 78)	20.5 (5; 70)	16.5 (3; 77)
1%	52.6 (9; 104)	33.4 (5; 90)	29.54 (5; 99)
0.1%	75.84 (27; 173)	43.56 (5; 115)	42.7 (5; 139)
0%	75.84 (27; 173)	43.56 (5; 115)	42.7 (5; 139)

Šįkart rezultatai skiriasi binariniame bei proporciniame modelyje. Geriausias rezultatas binariniame - „K-Tournament", kai proporciniame - „Modified Roulette Wheel". Tačiau verta būtų pastebėti, jog antrojo algoritmo rezultatų aibė šiek tiek mažiau išsibarsčiusi.



(a) Binarinis (b) Proporcinis

6 pav. Tėvų parinkimo palyginimas Nr. 2

4.2.3. Kryžminimas

Dar kart tiriami 5 darbe minėti kryžminimo algoritmai. Gauti binarinio bei proporcinio modelio rezultatai yra matomi paveikslėliuose 7a ir 7b.

Lentelės algoritmų palyginimui:

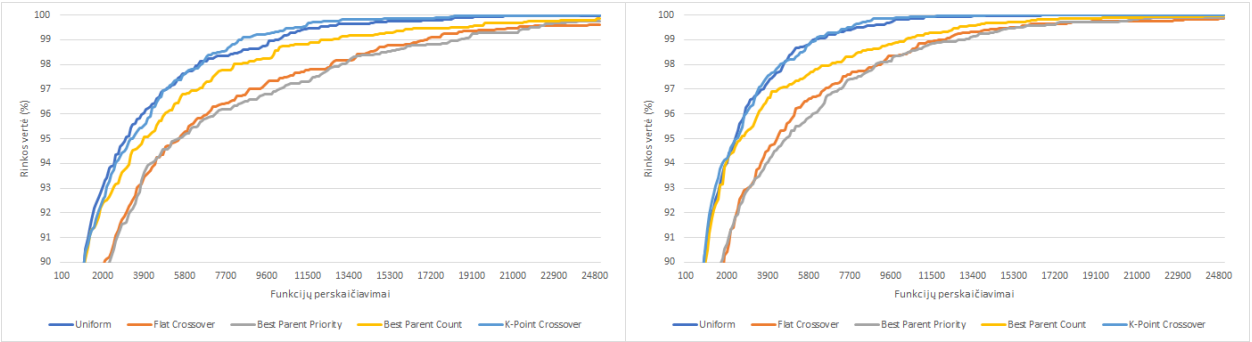
Binarinis

A. / P.	**Uniform**	**Flat**	**B.P. Priority**	**B.P. Count**	**K-Point**
1%	81.58 (11;205)	150.4 (31;250)	150.78 (20;250)	114.22 (27;250)	80.52 (30;182)
0.1%	118.38 (11;250)	196.92 (59;250)	165;96 (20;250)	141.88 (44;250)	99.8 (35;250)
0%	118.38 (11;250)	196.92 (59;250)	165.96 (20;250)	141.88 (44;250)	99.8 (35;250)

Proporcinis

Alg. / Pakl.	**Uniform**	**Flat**	**B.P. Priority**	**B.P. Count**	**K-Point**
3%	12.5 (4; 40)	14 (5; 74)	11 (1; 65)	14 (1; 43)	40.6 (9; 78)
1%	32.8 (5; 102)	63.44 (7; 226)	68.84 (10; 184))	43.56 (1; 96)	52.6 (9; 104)
0.1%	32.8 (5; 102)	63.44 (7; 226)	68.84 (10; 184))	43.56 (1; 96)	75.84 (27; 173)
0%	32.8 (5; 102)	63.44 (7; 226)	68.84 (10; 184)	43.56 (1; 96)	75.84 (27; 173)

Prasčiausiai veikiantys algoritmai nepakito nuo pirmojo tyrimo. Geriausiai veikiantis tyrimas binariniame modelyje yra „K-Point". Tuo tarpu proporciniame „Uniform". Vėlgi verta pastebėti tai, jog proporciniame modelyje „Best Parent Count" algoritmo rezultatai mažiausiai išsimėtę, todėl sunku įvertinti geriausią rezultatą.



(a) Binarinis (b) Proporcinis

7 pav. Kryžminimų palyginimas Nr. 2

4.2.4. Mutacija

Atliktas tyrimas su 3 darbe minėtomis mutacijų strategijomis. Binarinio bei proporcinio bandymo rezultatus galima matyti paveikslėliuose 8a ir 8b.

Lentelės algoritmų palyginimui:

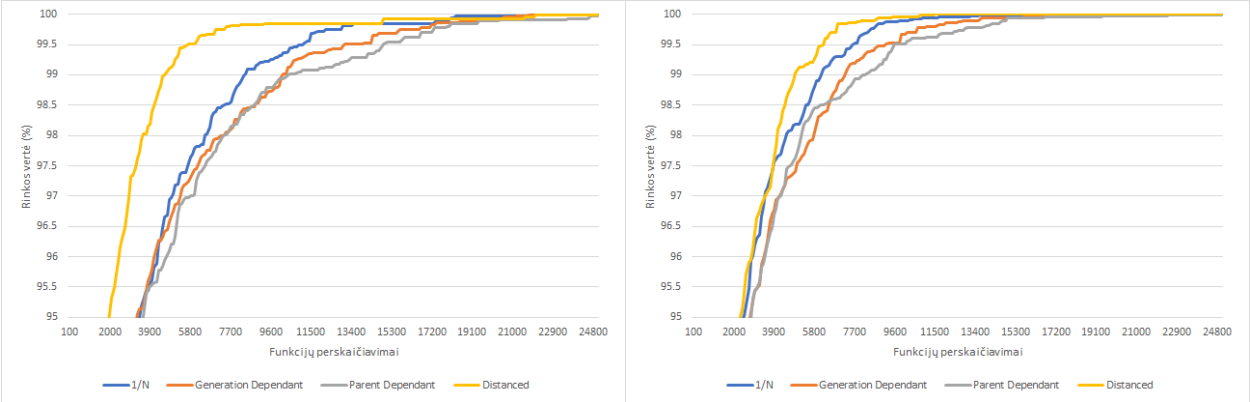
Binarinis

Alg. / Pakl.	**1/N**	**G. Lowering**	**P. Dependent**	**Distance Dependent**
1%	80.52 (30; 182)	97.36 (15; 217)	104.88 (23; 246)	45.72 (16; 220)
0.1%	99.8 (35; 250)	116.96 (15; 250)	129.72 (23; 250)	54.76 (16; 250)
0%	99.8 (35; 250)	116.96 (15; 250)	129.72 (23; 250)	54.76 (16; 250)

Proporcinis

Alg. / Pakl.	**1/N**	**Lowering**	**Dependent**	**Distance Dependent**
3%	40.6 (9; 78)	43 (12; 98)	28.5 (9; 95)	36.32 (15; 68)
1%	52.6 (9; 104)	70.98 (21; 136)	75.18 (22; 148)	44.06 (19; 107)
0.1%	75.84 (27; 173)	96.08 (36; 199)	102.7 (28; 250)	55.68 (19; 114)
0%	75.84 (27; 173)	96.08 (36; 199)	102.7 (28; 250)	55.68 (19; 114)

Šiame tyrime kaip ir ankstesniajame, absoliučiai geriausia buvo strategija su atstumo naudojimu. Lyginant tik mutacijos tikimybės strategijas geriausia buvo ”1/N” strategija.



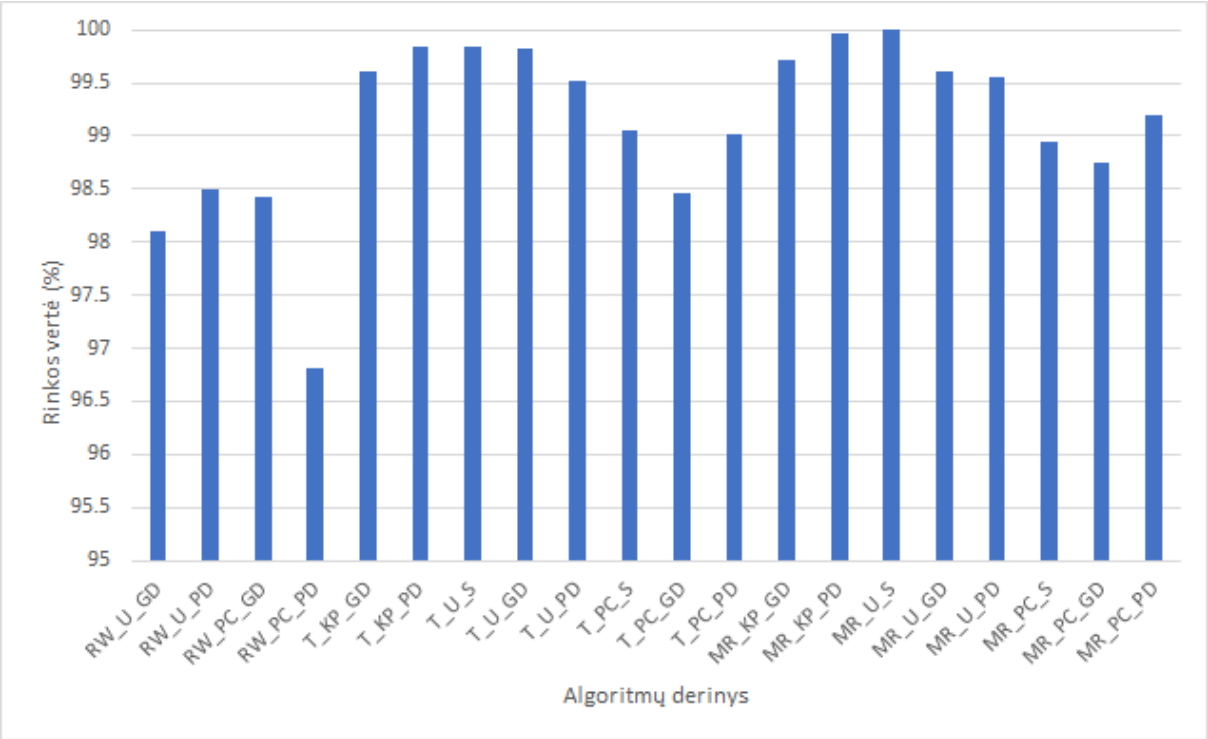
(a) Binarinis

(b) Proporcinis

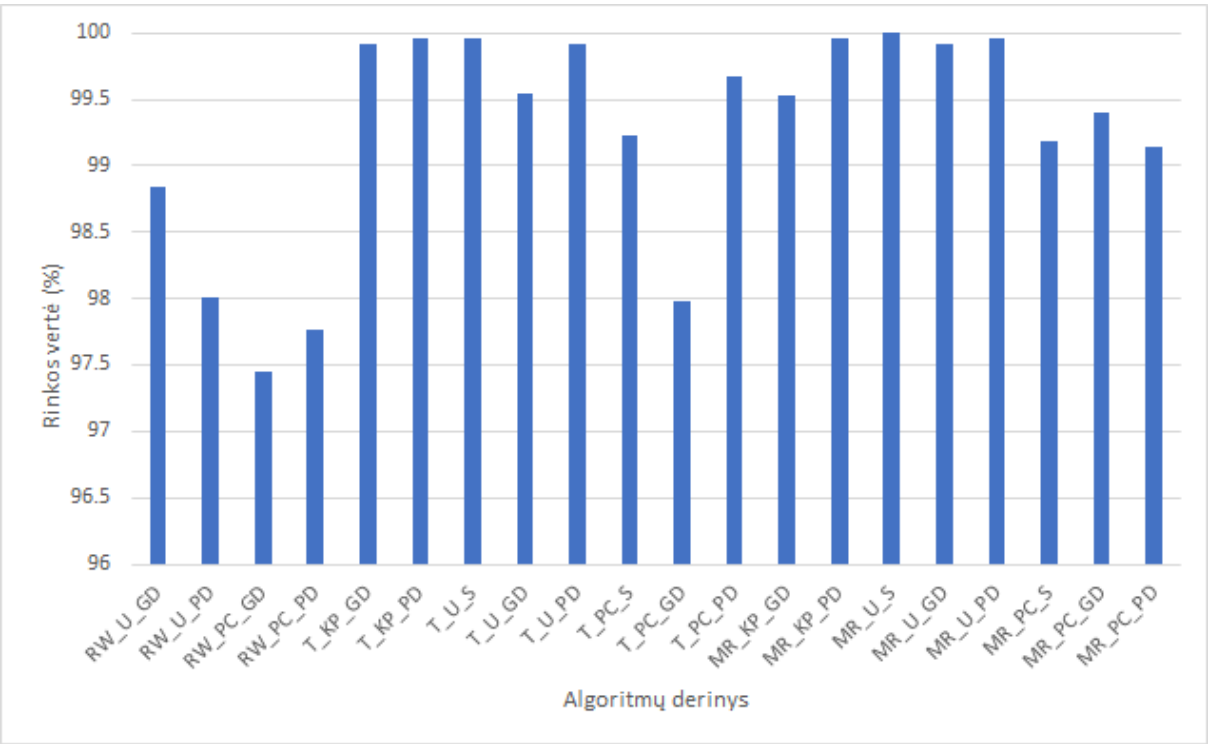
8 pav. Mutacijų palyginimas Nr. 2

4.2.5. Kiti strategijų deriniai

Dar kartą buvo paleistos įvairių derinių kombinacijos nenaudojant mutacijos atstumo strategijos. Gautuose tyrimuose, binarinio (Priedas nr. 11,Priedas nr. 12, Priedas nr. 13, Priedas nr. 14, Priedas nr. 15) ir proporcinio (Priedas nr. 16, Priedas nr. 17, Priedas nr. 18, Priedas nr. 19, Priedas nr. 20), paveikslėliuose esančiuose prieduose, galima matyti įvairių derinių grafikus. Paveikslėliuose (binarinis 9 ir proporcinis 10) galima matyti bendrą algoritmų vaizdą, kai pirmasis derinys buvo suradęs optimalųjį sprendinį.



9 pav. Įvairių derinių tyrimas Nr. 2 (binarinis)



10 pav. Įvairių derinių tyrimas Nr. 2 (proporcinis)

4.3. Tyrimas nr. 3: $n = 3, p = 500, I = 5000, e = 5$

4.3.1. Optimalus sprendinys

Trečiajame tyrime optimalus sprendinys yra:

$Binarinis - 799077, Proporcinis - 1572183$

4.3.2. Tėvų parinkimas

Dar kartą tirti 3 tėvų parinkimo algoritmai. Rezultatai yra matomi paveikslėliuose 11a bei 11b.

Lentelės algoritmų palyginimui:

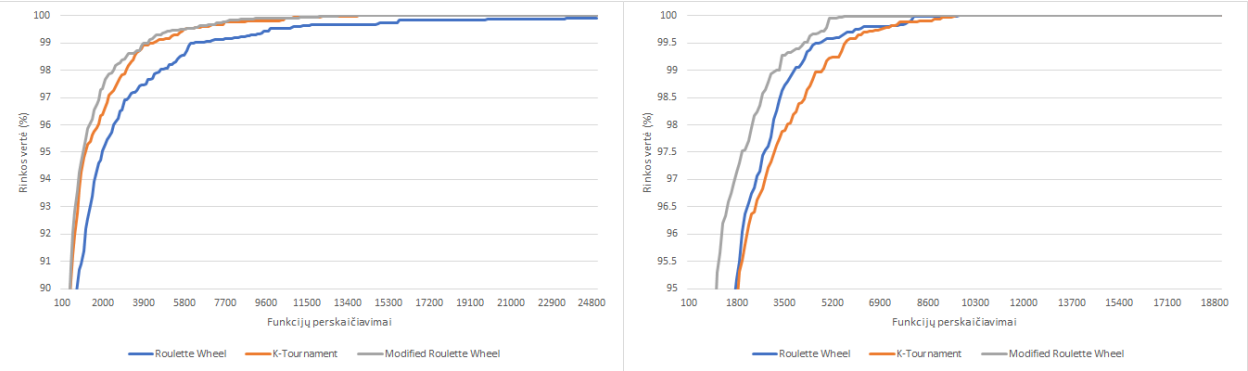
Binarinis

Alg. / Pakl.	**Roulette Wheel**	**K-Tournament**	**Modified Roulette Wheel**
1%	68.66 (12; 250)	39.12 (7; 108)	36.38 (3; 125)
0.1%	130.88 (16; 250)	59.72 (13; 130)	61.22 (9; 136)
0%	168.96 (16; 250)	106.82 (14; 250)	89.96 (19; 250)

Proporcinis

Alg. / Pakl.	**Roulette Wheel**	**K-Tournament**	**Modified Roulette Wheel**
3%	22.32 (6; 80)	19.2 (4; 68)	15.82 (3; 50)
1%	36.38 (14; 80)	24.56 (4; 75)	21.86 (3; 53)
0.1%	41 (15; 91)	27.68 (4; 75)	25.2 (5; 55)
0%	58.68 (15; 166)	40.74 (13; 130)	47,64 (8; 125)

Binariniame modelyje algoritmai veikė panašiai, šiokį tokį pranašumą rezultatų išsimėtyme bei galutiniuose rezultatuose turėjo „Modified Roulette Wheel" algoritmas. Tuo tarpu proporciniame modelyje pradžioje geriau pasirodė vėlgi trečiasis algoritmas, tačiau ieškant optimalaus sprendinio nusileido „K-Tournament" algoritmui.



(a) Binarinis (b) Proporcinis

11 pav. Tėvų parinkimo palyginimas Nr. 3

4.3.3. Kryžminimas

Ištyrus penkis darbe minimus kryžminimo algoritmus su trečiojo tyrimo duomenimis gauti binarinio bei proporcinio modelio rezultatai yra matomi paveikslėliuose 12a bei 12b.

Lentelės algoritmų palyginimui:

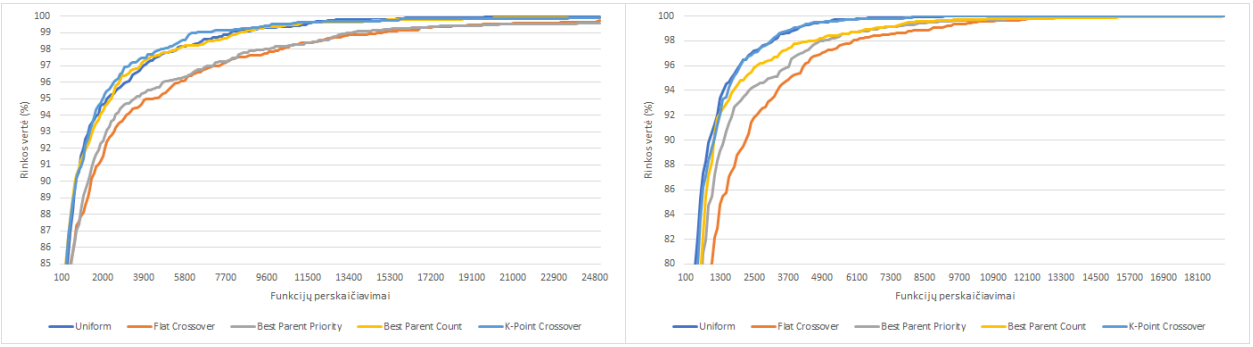
Binarinis

A. / P.	**Uniform**	**Flat**	**B.P. Priority**	**B.P. Count**	**K-Point**
1%	69.56 (12;240)	131.06 (21;250)	126.82 (29;250)	73.94 (14;250)	68.66 (12;250)
0.1%	129.58 (12;250)	184.36 (46;250)	199.62 (37;250)	144.74 (20;250)	130.88 (16;250)
0%	174.7 (46;250)	213.24 (46;250)	227.68 (37;250)	186.96 (60;250)	168.96 (16;250)

Proporcinis

A. / P.	**Uniform**	**Flat**	**B.P. Priority**	**B.P. Count**	**K-Point**
3%	22.12 (3;80)	50.72 (16;136)	36.9 (10;131))	34.26 (5;185)	22.32 (6;80)
1%	34.24 (3;91)	74.34 (23;145)	61.98 (11;146)	56.08 (10;189)	36.38 (14;80)
0.1%	39.78 (3;91)	87.54 (23;241)	72.04 (11;162)	62.06 (12;189)	41 (15; 91)
0%	51.88 (14;119)	120.84 (29;250)	104.14 (14;250)	85.5 (12;236)	58.68 (15;166)

Geriausiai veikiantis algoritmas aprašytomis sąlygomis buvo „Uniform". Kaip ir prieš tai vykusiuose tyrimuose prasčiausiai veikiantys du algoritmai yra antrasis bei trečiasis. Nedaug nuo geriausiojo algoritmo atsiliko ir „K-Point Crossover".



(a) Binarinis (b) Proporcinis

12 pav. Kryžminimų palyginimas Nr. 3

4.3.4. Mutacija

Atliktas tyrimas su 3 darbe minėtomis mutacijų strategijomis. Binarinio bei proporcinio bandymo rezultatus galima matyti paveikslėliuose 13a bei 13b.

Lentelės algoritmų palyginimui:

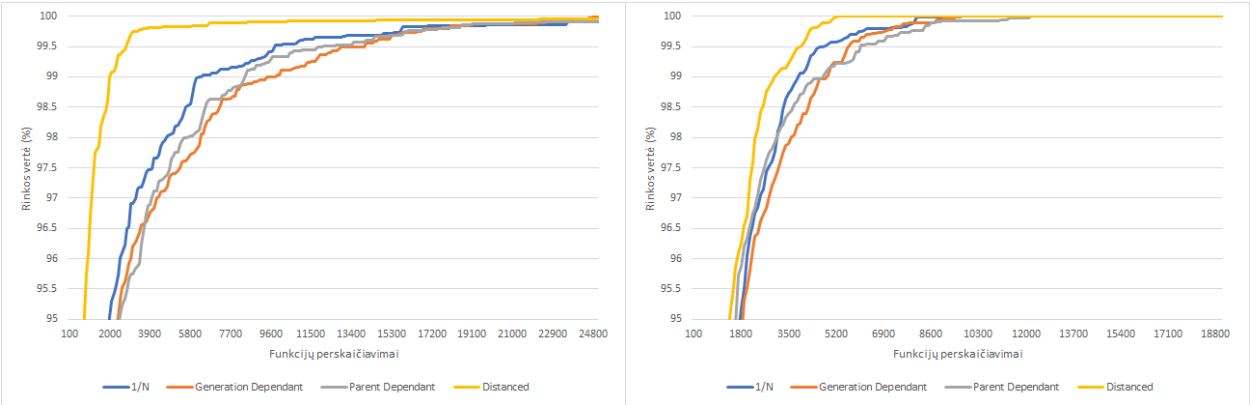
Binarinis

Alg. / Pakl.	1/N	G. Lowering	P. Dependent	Distance Dependent
1%	68.66 (12; 250)	86.24 (9; 245)	75.8 (23; 250)	17.7 (5; 66)
0.1%	130.88 (16; 250)	127.48 (13; 250)	131.84 (33; 250)	137.56 (15; 250)
0%	168.96 (16; 250)	181.68 (33; 250)	183.34 (35; 250)	152 (15; 250)

Proporcinis

Alg. / Pakl.	1/N	G. Lowering	P. Dependent	Distance Dependent
3%	22.32 (6; 80)	25.22 (6; 66)	23.16 (3; 85)	18.42 (4; 50)
1%	36.38 (14; 80)	40.8 (11; 94)	38.14 (7; 121)	25.9 (4; 51)
0.1%	41 (15; 91)	45.94 (11; 94)	46.52 (9; 131)	28.26 (7; 51)
0%	58.68 (15; 166)	68.32 (15; 183)	75.9 (10; 174)	37.76 (10; 64)

Trečiajame mutacijų tyrime dar kartą absoliučiai geriausias buvo algoritmas su atstumo strategija, tačiau galima pastebėti, kad binariniame tyrime jis vidurinėje vertinimo lentelės dalyje nusileidžia visiems algoritmams.



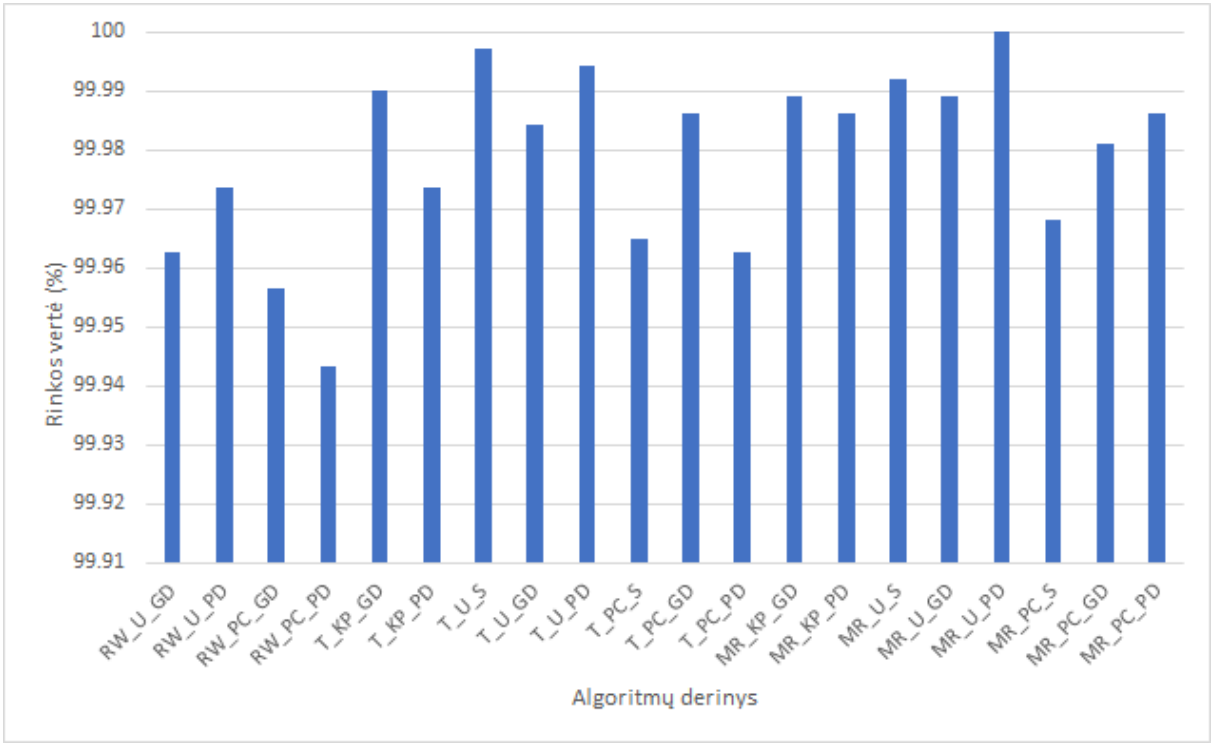
(a) Binarinis

(b) Proporcinis

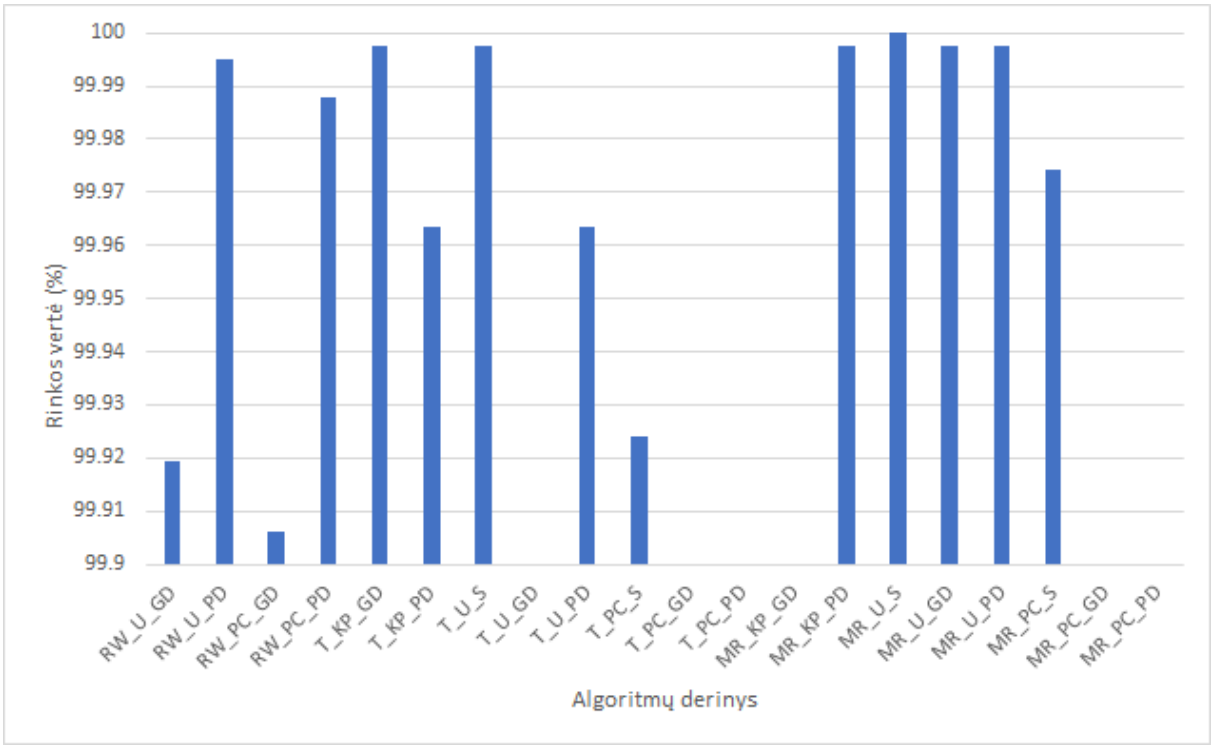
13 pav. Mutacijų palyginimas Nr. 3

4.3.5. Kiti strategijų deriniai

Dar kartą buvo paleistos įvairių derinių kombinacijos nenaudojant mutacijos atstumo strategijos. Gautuose tyrimuose, binarinio (Priedas nr. 21, Priedas nr. 22, Priedas nr. 23, Priedas nr. 24, Priedas nr. 25) ir proporcinio (Priedas nr. 26, Priedas nr. 27, Priedas nr. 28, Priedas nr. 29, Priedas nr. 30), paveikslėliuose esančiuose prieduose, galima matyti įvairių derinių grafikus. Paveikslėliuose (binarinis 14 ir proporcinis 15) galima matyti bendrą algoritmų vaizdą, kai pirmasis derinys buvo suradęs optimalųjį sprendinį. Proporcinio modelio grafike, kai kurie algoritmai nepasiekė pakankamai gero rezultato, todėl nėra matomi grafike. Norint išryškinti algoritmų skirtumus skalė yra sumažinta.



14 pav. Įvairių derinių tyrimas Nr. 3 (binarinis)



s

15 pav. Įvairių derinių tyrimas Nr. 3 (proporcinis)

4.4. Tyrimas nr. 4: $n = 5, p = 5000, I = 5000, e = 5$

4.4.1. Optimalus sprendinys

Šiame tyrime optimalus sprendinys yra:

$$Binarinis - 1208697, Proporcinis - 2644700$$

4.4.2. Mutacija

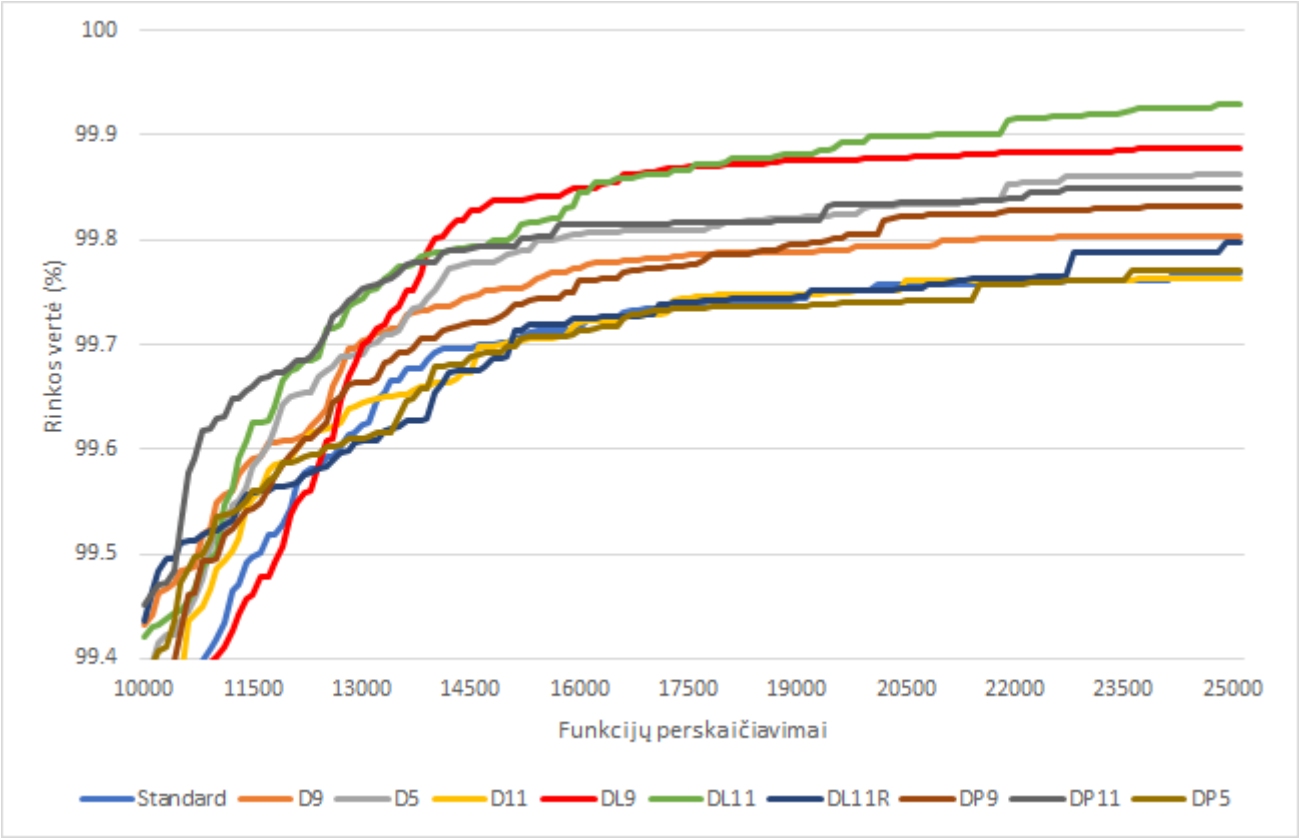
Daug patyrinėjus įvairius algoritmų derinius teko apsispręsti, kuris algoritmas turi daugiausiai potencialo būti geriausias. Absoliučiai geriausius rezultatus visuose tyrimuose nuo paprasčiausio, padidinus naujų objektų skaičių, bei galimų objektų aibę visuomet išlaikė strategija su atstumo priklausomybe. Buvo nuspręsta bandyti tobulinti šią strategiją ieškant absoliučiai geriausio strategijų derinio. Taip pat buvo stipriai pasunkintas uždavinys, tikintis, kad nei vienas algoritmas nepasieks vidutinio maksimumo, kad visa tyrimų aibė prieš pasibaigiant generacijoms pasiektų optimalųjį sprendinį, tačiau net ir taip įvykus, tai būtų tik išskirtiniais atvejais, kai algoritmų derinys tikrai labai gerai veiktų. Pritaikius įvairias strategijas buvo gauti rezultatai, kurie yra matomi paveikslėliuose - binarinis 16 bei proporcinis 17.

Bandymuose buvo bandoma plėsti bei mažinti tikimybės sritį, kiek daug padidės ar sumažės tikimybė pagal generaciją. Kita strategija bandyti nuo pradžios visuomet mažinti mutacijos tikimybę ir tuo pačiu naująją strategiją arba atvirkščiai - mutacijos tikimybę nuolat didinti.

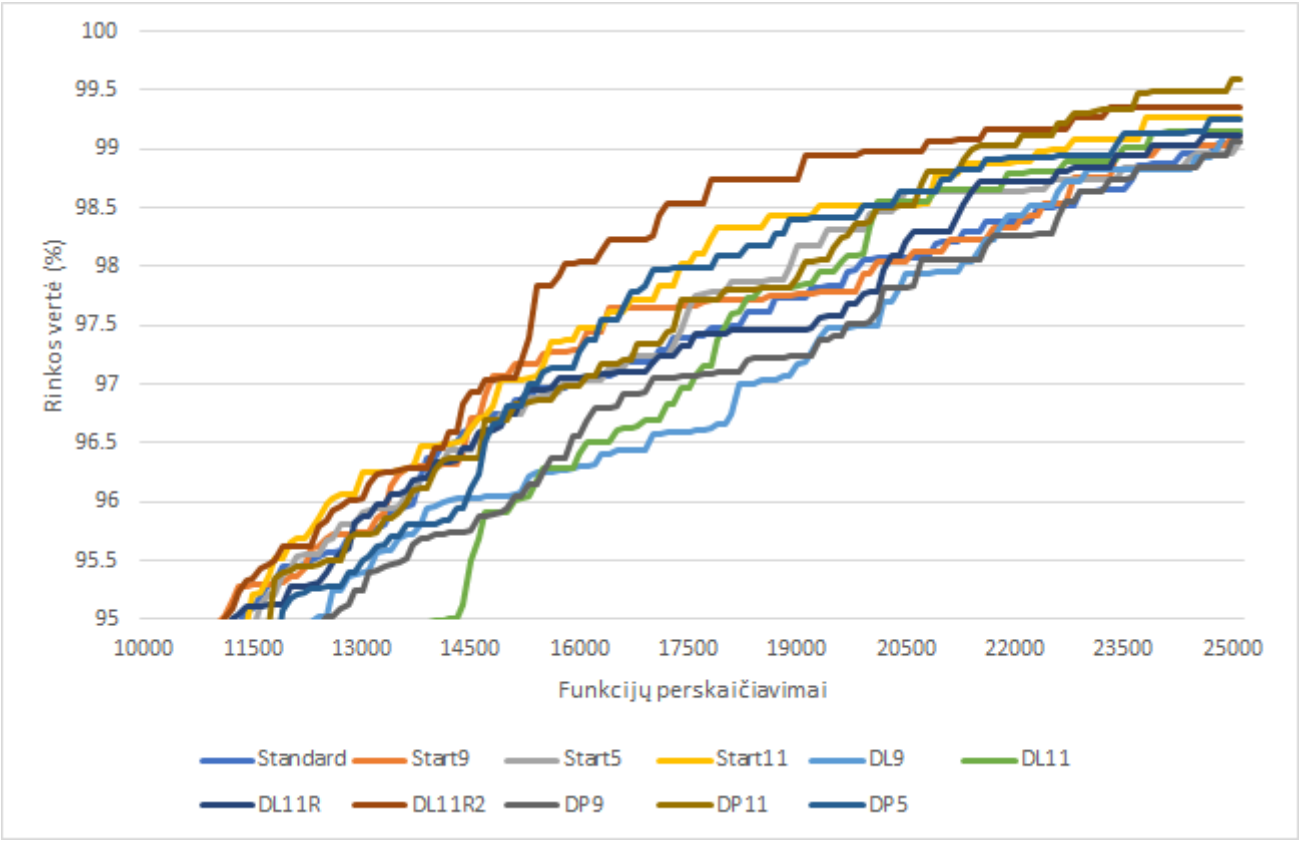
Paveikslėliuose matomi trumpiniai, kurie reiškia įvairių strategijų pasirinkimą. Kaip atskaitos taškas naudojama standartinė strategija, kai atstumo taikymas yra naudojamas nuolatos. Kitos strategijos žymimos StartX, žymi, kad pradedama algoritmas taip, kad atstumų strategija bus naudojama tik su tikimybe 0.5, bet su kiekviena generacija vis didės. Skaičius X žymi didėjimo greitį, pavyzdžiui, žymėjimas 5 reiškia, kad pradinė tikimybė, kad bus naudojama „Distance Dependent" mutacija yra 0.5, ji su kiekviena ateinančia generacija didės po tam tikrą nustatytą dydį, kuris šiuo atveju prasideda nuo 0.05 ir mažėja su kiekviena ateinančia generacija, kol tikimybė tampa 100%. Algoritmai prasidedantys DL reiškia, kad buvo naudojama jau nebe standartinė mutacijos tikimybė, tačiau tokia, kuri priklauso nuo generacijos (GD alg. 5). Algoritmai prasidedantys DP buvo naudojama mutacijos tikimybės strategija, kurioje tikimybė priklauso nuo tėvų (PD alg. 4). Šios strategijos aprašytos anksčiau darbe 3.2.3. Raidė R reiškia, kad buvo naudojamas bandymas, kuriame GD strategija iš nuolat mažėjančios mutacijos tikimybės pakeičiama į nuolat didėjančią. Numeracija (R2) žymi kitokį didėjimo greitį, šiuo atveju greitis buvo sumažintas.

Geriausiai veikiantis algoritmas binariniame tyrime buvo DL11, tai nuo atstumo priklausanti strategija su nuo generacijos priklausančia (DL) mutacijos tikimybe, kuri nuolat mažėja bei greičiausiai kylančiu atstumo strategijos taikymu (11).

Geriausiai veikiantis algoritmas proporciniame tyrime buvo DP11, tai nuo atstumo priklausanti strategija su nuo tėvų priklausančia (DP) mutacijos tikimybe bei greičiausiai kylančiu atstumo strategijos taikymu (11).



16 pav. Mutacijos strategijų palyginimas (binarinis)



17 pav. Mutacijos strategijų palyginimas (proporcinis)

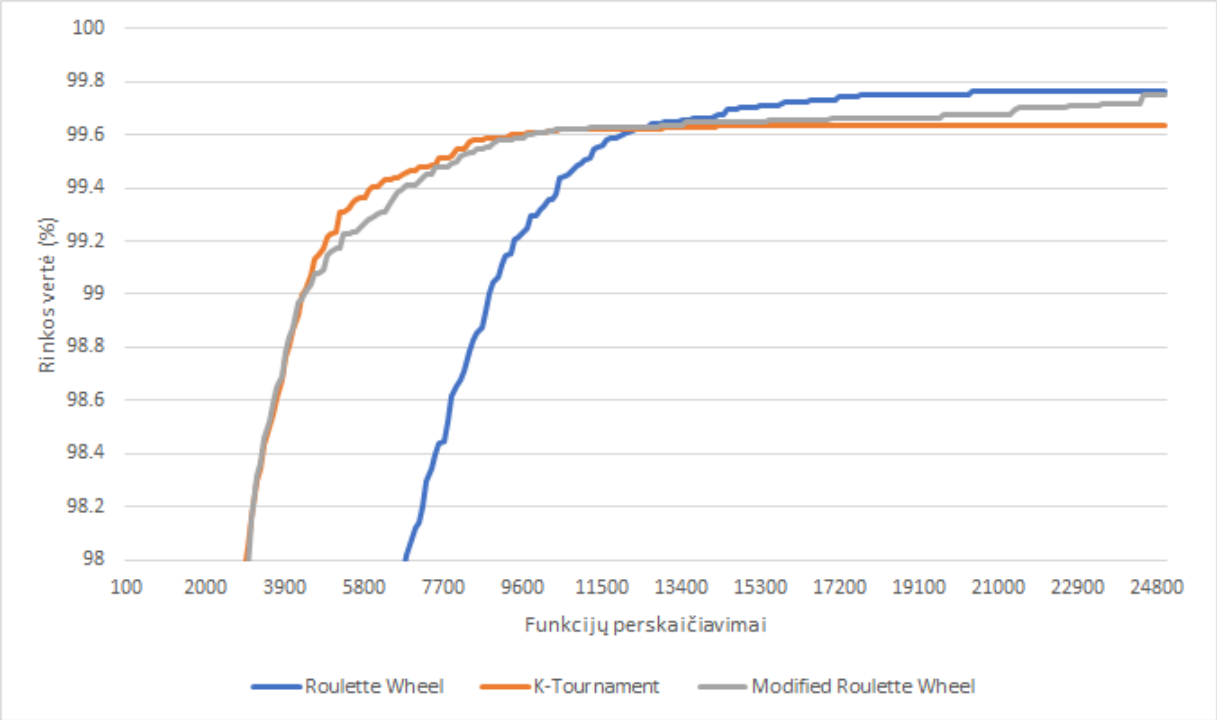
4.4.3. Kiti tyrimai

Suradus absoliučiai geriausią sprendinį mutacijai, buvo nuspręsta dar kartą paleisti ir išbandyti kaip geriausia strategija veiks su skirtingomis tėvų parinkimo bei kryžminimo strategijomis. Šiuo atveju, bus paleisti tyrimai binariniam modeliui su įvairiomis tėvų parinkimo bei kryžminimo strategijomis ir mutacijos strategija DL11. Atitinkamai ir proporciniam, kur mutacijos tikimybė bus DP11.

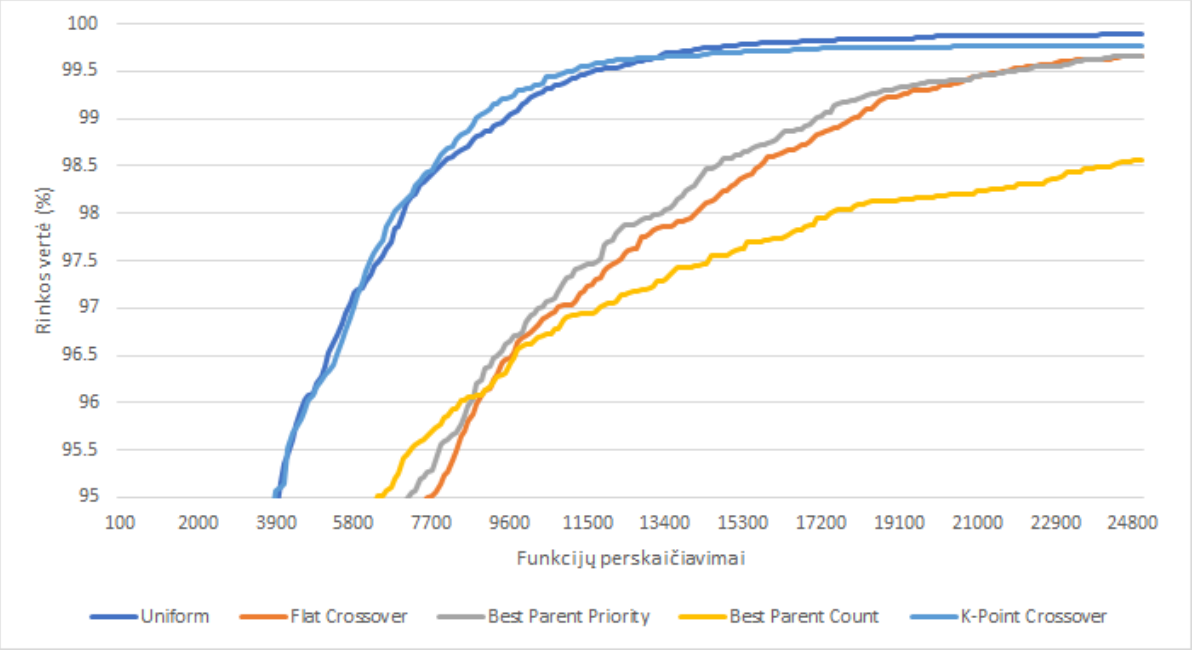
4.4.3.1. Binarinis modelis

Buvo atlikti tyrimai su įvairiomis strategijomis, bei mutacija, kurioje naudojama „Distance Dependent" strategija, su nuo generacijos priklausančia mažėjančia tikimybe mutuoti. Rezultatai yra matomi paveikslėliuose - tėvų parinkimo 18 bei kryžminimo 19.

Iš grafikų matyti, kad geriausia tėvų parinkimo strategija šiame derinyje yra „Modified Roulette Wheel", o geriausia kryžminimo „K-Point Crossover". Kyla klausimas, tai kuris iš šių dviejų derinių geriausias. Tačiau šis tyrimas jau buvo atliktas ir gali būti matomas paveikslėlyje 19, nes būtent algoritmas, kuris naudoja K-Point Crossover ir yra tas pats derinys, kuris buvo geriausias tėvų parinkimo paveikslėlyje - „Roulette Wheel", „K-Point" bei DL11. Tad geriausias derinys spręsti binarinį uždavinio modelį yra „Roulette Wheel", „Uniform" bei nuo atstumo priklausančia mutacija, kurioje tikimybė priklauso nuo esamos generacijos bei nuolat mažėja.



18 pav. Tėvų parinkimo strategijos su DL11 mutacija



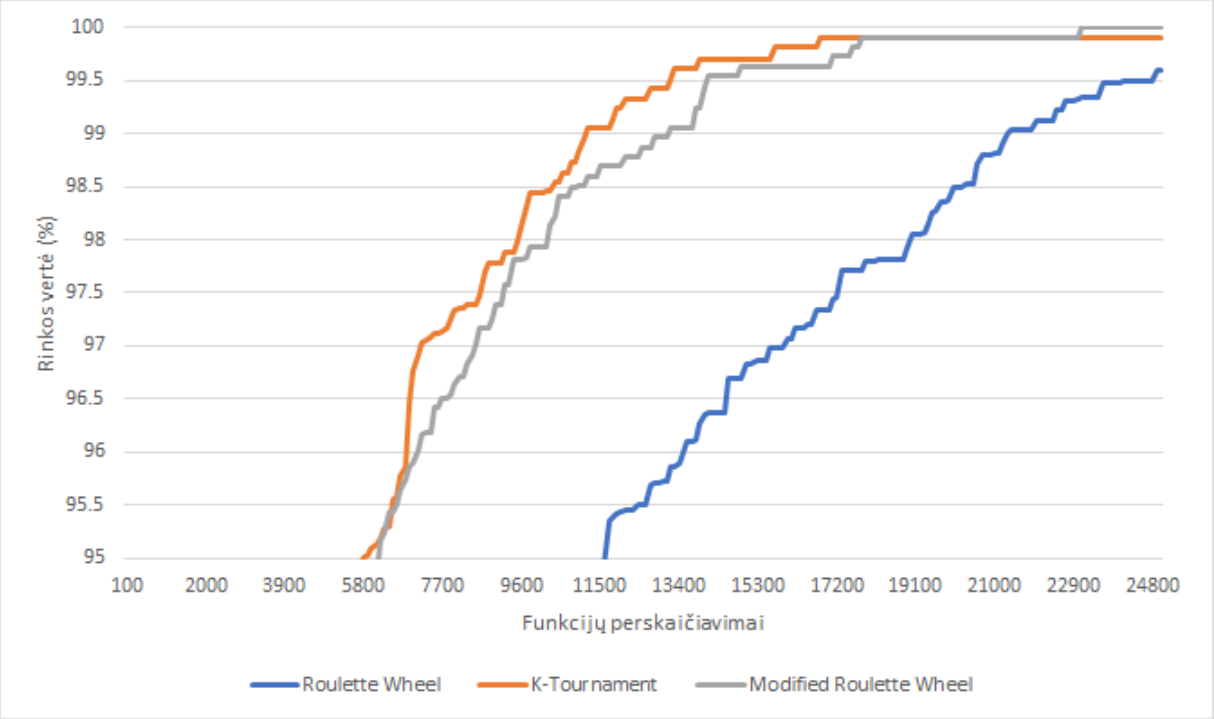
19 pav. Kryžminimo strategijos su DL11 mutacija

4.4.3.2. Proporcinis modelis

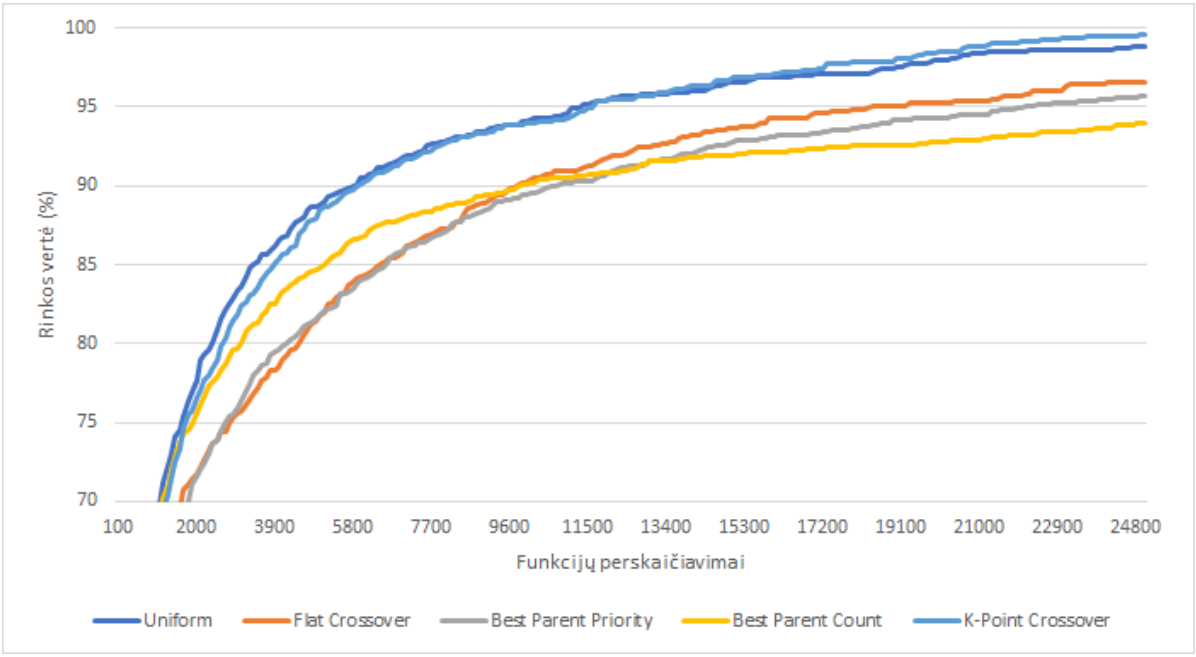
Buvo atlikti tyrimai su įvairiomis strategijomis, bei mutacija, kurioje naudojama „Distance Dependent" strategija, su nuo tėvų priklausančia mutacijos tikimybe. Rezultatai yra matomi paveikslėliuose - tėvų parinkimo 20 bei kryžminimo 21.

Šiame modelyje rezultatai atvirkščiai nei binariniame išsidėstė taip, kad geriausias tėvų parinkimas kitoks, nei su kuriuo būdavo atliekami tyrimai, o kryžminime geriausias ir išliko „K-Point Crossover", kuris ir buvo naudojamas tyrimuose.

Kaip ir prieš tai atliktame tyrime įdomu, kuris derinys geresnis. Tačiau šis palyginimas taip pat jau atvaizduotas paveikslėlyje 20. Antroji funkcija ir yra ta pati, kuri geriausia buvo kryžminimo tyrime, „Roulette Wheel", „K-Point Crossover" bei DP11. Todėl absoliučiai geriausias derinys, kuris be to ir vienintelis šitame pasunkintame uždavinyje sugebėjo jau nuo 233 generacijos vidutiniškai visuomet būti radęs optimalųjį sprendinį yra „Modified Roulette Wheel", „K-Point Crossover" bei nuo atstumo priklausanti mutacijos strategija kurios greitis didžiausias, su nuo tėvų priklausančia mutacijos tikimybe.



20 pav. Tėvų parinkimo strategijos su DP11 mutacija



21 pav. Kryžminimo strategijos su DP11 mutacija

Rezultatai ir išvados

Išsprendus darbe suformuluotus uždavinius gauti rezultatai:

1. Sudaryti ir ištirti skirtingo sudėtingumo objektų vietų parinkimo uždaviniai bei nustatyti genetinio algoritmo vertinimo kriterijai skirtingiems klientų elgsenos modeliams.

2. Įgyvendintos ir ištirtos kelios naujos kryžminimo, mutacijos bei tėvų parinkimo strategijos. Įvertintas skirtingų algoritmų derinių pranašumas įvairaus sudėtingumo objektų vietų parinkimo uždaviniuose.

3. Tyrimo eigoje pamodifikuota geriausiai veikusi mutacijos strategija, kuri po modifikacijos gavo geresnius rezultatus abiejuose klientų elgsenos modeliuose.

4. Sukurta modifikuota „Roulette Wheel" strategijos versija tėvų parinkimui, kuri taip pat davė geriausius rezultatus proporcinio tipo uždavinyje ir buvo vieninteliame derinyje, kuris surado optimalųjį sprendinį visuose tyrimuose anksčiau nei buvo pasibaigusios visos generacijos.

5. Tyrimų pagalba surastas geriausiai binariniame modelyje veikiantis strategijų derinys, kuris yra „Roulette Wheel", „Uniform" bei „Distance Dependent" mutacijos greičiausiai vykdoma strategija su nuo generacijos priklausančia mutacijos tikimybe. Sunkiausiame tirtame uždavinyje vidutiniškai suranda sprendinį kuris yra daugiau nei 99.89% arti optimalaus sprendinio.

6. Tyrimų pagalba surastas geriausiai proporciniame modelyje veikiantis strategijų derinys, kuris yra „Modified Roulette Wheel", „K-Point Crossover" bei „Distance Dependent" mutacijos greičiausiai vykdoma strategija su nuo tėvų priklausančia genų mutavimo tikimybe. Sunkiausiame tirtame uždavinyje optimalųjį sprendinį vidutiniškai randa visuomet 231 generacijoje.

Atliekant eksperimentinius tyrimus pastebėtos išvados:

1. Naujai pasiūlyta tėvų parinkimo strategija „Modified Roulette Wheel" tyrimuose, kuriuose buvo sprendžiami paprastesni uždaviniai, pasirodė šiek tiek geriau, nei kiti žinomi algoritmai. Vėliau ši strategija buvo išrinkta kaip geriausia tėvų parinkimo strategija sunkiausiame proporcinio modelio uždavinio tyrime.

2. Ištyrus skirtingas kryžminimo algoritmų modifikacijas, deja, teigiamų rezultatų gauta nebuvo, nes abi siūlomos strategijos buvo sukurtos „Flat Crossover" pagrindu, kur iš dviejų tėvų sukuriamas tik vienas vaikas, vietoje dviejų. Tai lėmė, kad siūlomos strategijos nesugebėjo aplenkti „Uniform" bei „K-Point Crossover" strategijų. Tuo tarpu modifikacijos daugumoje atvejų pasirodė geriau nei standartinė „Flat Crossover" strategija.

3. Pasiteisino mutacija naudojant „Distance Dependent" mutacijos būdą, kas leido tęsti gilesnius tyrimus šį būdą modifikuojant.

4. Modifikuojant „Distance Dependent” mutacijos strategija pastebėta, kad šios strategijos taikymas aktualiausias vėlesnėje algoritmo stadijoje. Algoritmo efektyvumas padidėjo pradžioje naudojant kitą strategiją.

5. Keičiant mutacijos tikimybės strategiją į „Parent Dependent” arba „Generation Dependent” pagerėjo „Distance Dependent” mutacijos strategijos rezultatai.

Šaltiniai

[US15] A.J. Umbarkar and P.D. Sheth *CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW*. ICTACT JOURNAL ON SOFT COMPUTING. VOLUME: 06, OCTOBER 2015, 1083-1092p.

[LFPZ15] Algirdas Lančinskas, Pascual Fernández, Blas Pelegín, Julius Žilinskas *Optimization Letters* "Improving solution of discrete competitive facility location problems". Volume 11. 2015.

[DAR94] Whitley, Darrell *A genetic algorithm tutorial* 1994

[HOL75] John H. Holland *Genetic Algorithms and Adaptation* Adaptive Control of Ill-Defined Systems pp 317-333, 1975

[PB14] Suvarna Patil, Manisha Bhende *Comparison and Analysis of Different Mutation Strategies to improve the Performance of Genetic Algorithm* Suvarna Patil et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 4669-4673

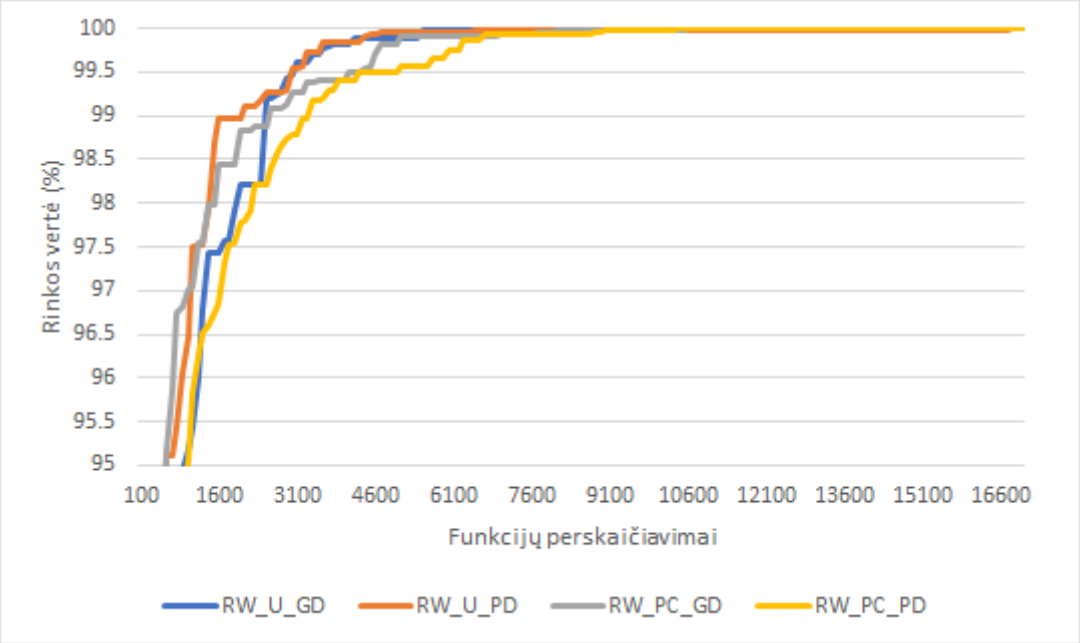
[LAN13] Algirdas Lančinskas *ATSITIKTINĖS PAIEŠKOS GLOBALIOJO OPTIMIZAVIMO ALGORITMŲ LYGIAGRETINIMAS* Doctoral Dissertation, 2013

[FPLŽ17] Pascual Fernández, Blas Pelegín, Algirdas Lančinskas, Julius Žilinskas *Computers & Operations Research* New heuristic algorithms for dicrete competitive location problems with binary and partially binary customer behavior, Article, 2017

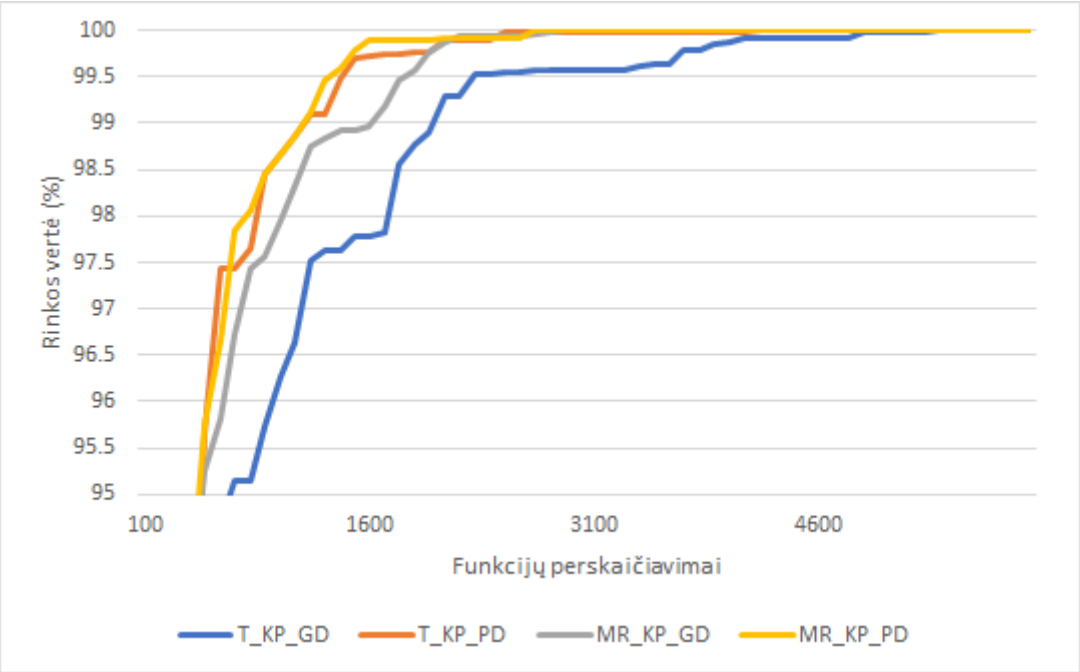
Priedai

Priedai suskirstyti į tris tyrimus atitinkamai pagal darbe vykdytus pirmuosius tris tyrimus. Kiekvieno tyrimo metu buvo leisti 2 skirtingi klientų elgsenos modeliai - binarinis bei proporcinis. Kiekvieno tyrimo ir modelio yra po 5 grafikus, suskirstytus į grupes:

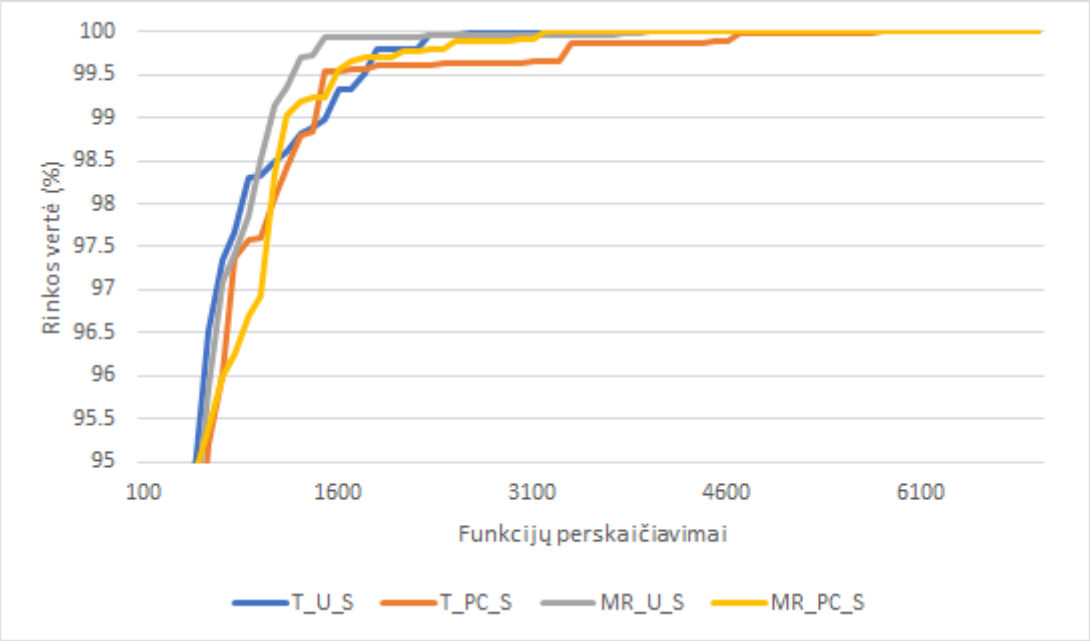
- 1 grupė - likę neištirti „Roulette Wheel" deriniai.
- 2 grupė - likę neištirti „K-Point Crossover" deriniai.
- 3 grupė - likę neištirti „1/N" mutacijos deriniai.
- 4 grupė - likę neištirti „Uniform" kryžminimo deriniai.
- 5 grupė - likę neištirti „Parent Count" kryžminimo deriniai.



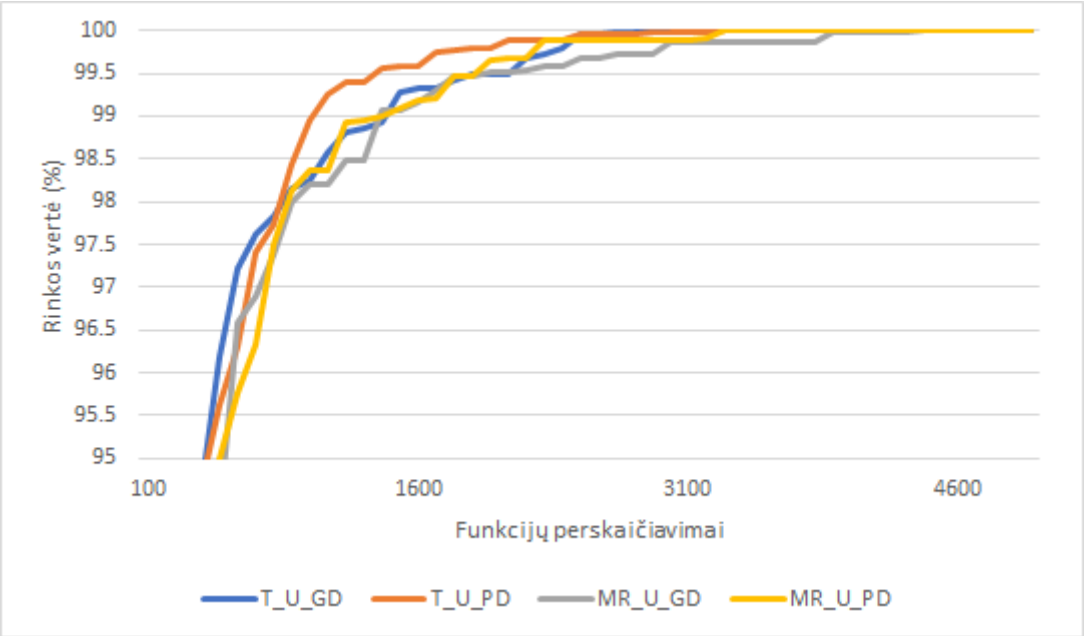
Priedas Nr. 1. Įvairių derinių I tyrimas. Grupė 1. Binarinis modelis.



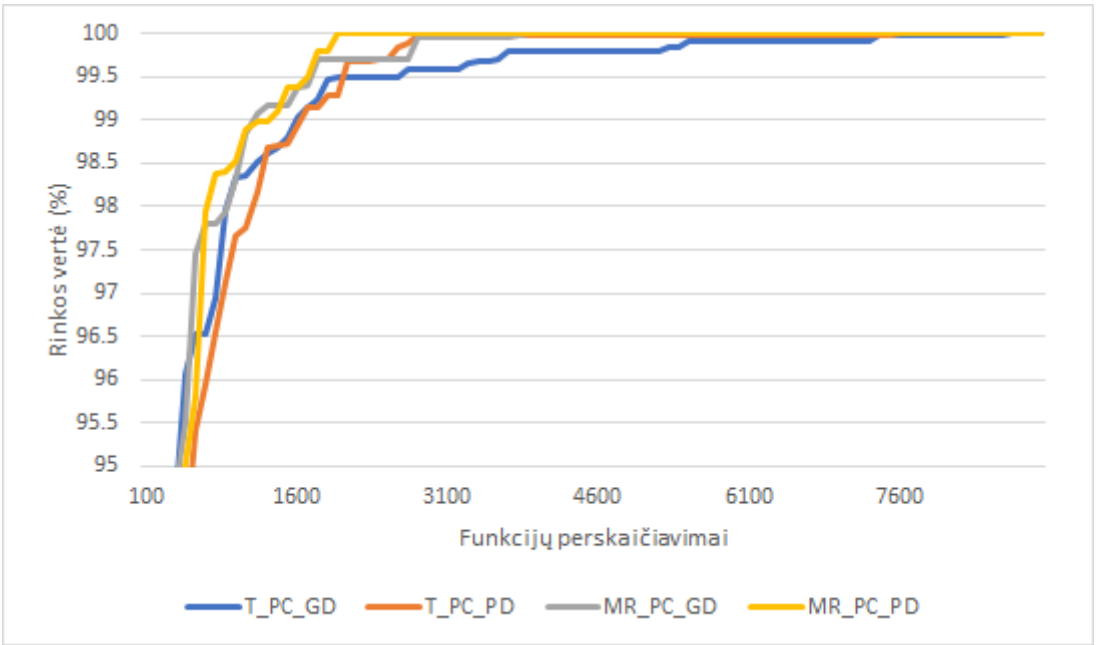
Priedas Nr. 2. Įvairių derinių I tyrimas. Grupė 2. Binarinis modelis.



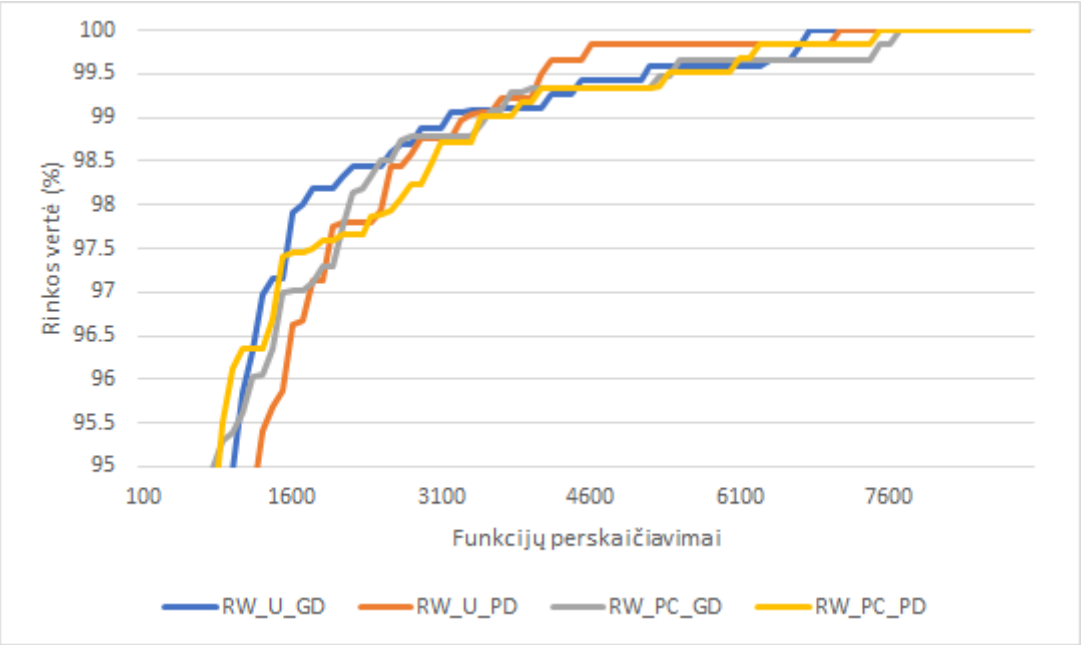
Priedas Nr. 3. Įvairių derinių I tyrimas. Grupė 3. Binarinis modelis.



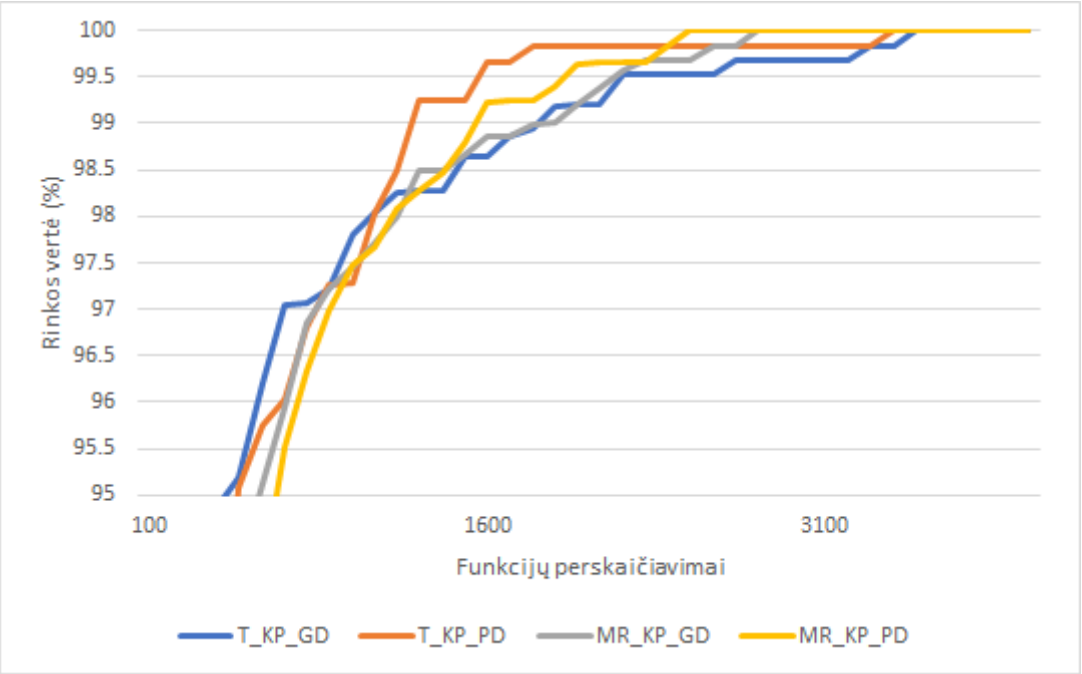
Priedas Nr. 4. Įvairių derinių I tyrimas. Grupė 4. Binarinis modelis.



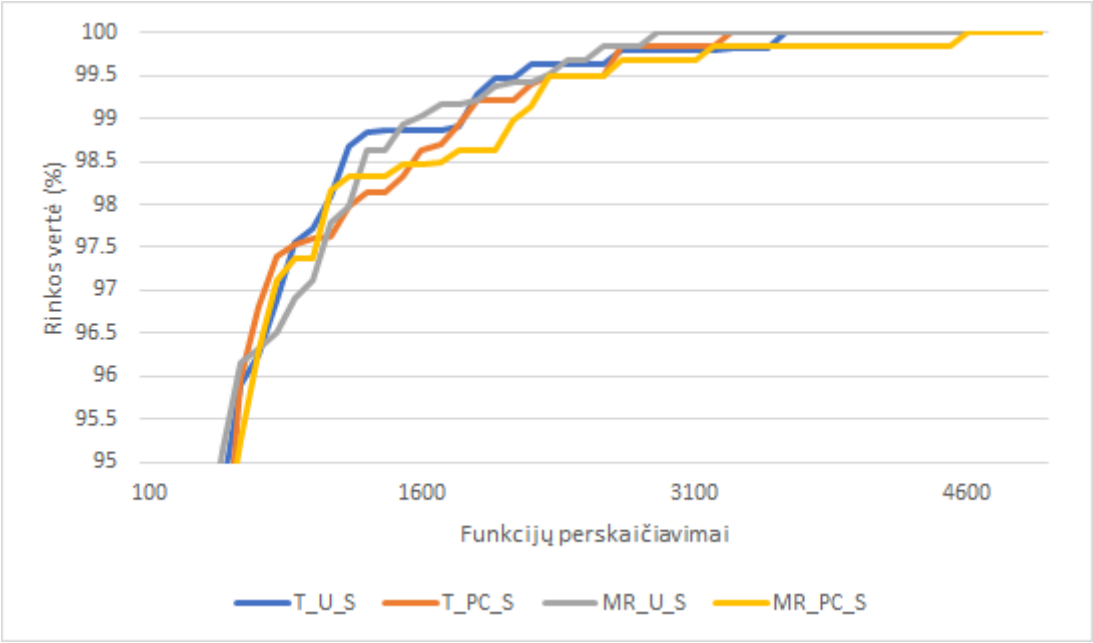
Priedas Nr. 5. Įvairių derinių I tyrimas. Grupė 5. Binarinis modelis.



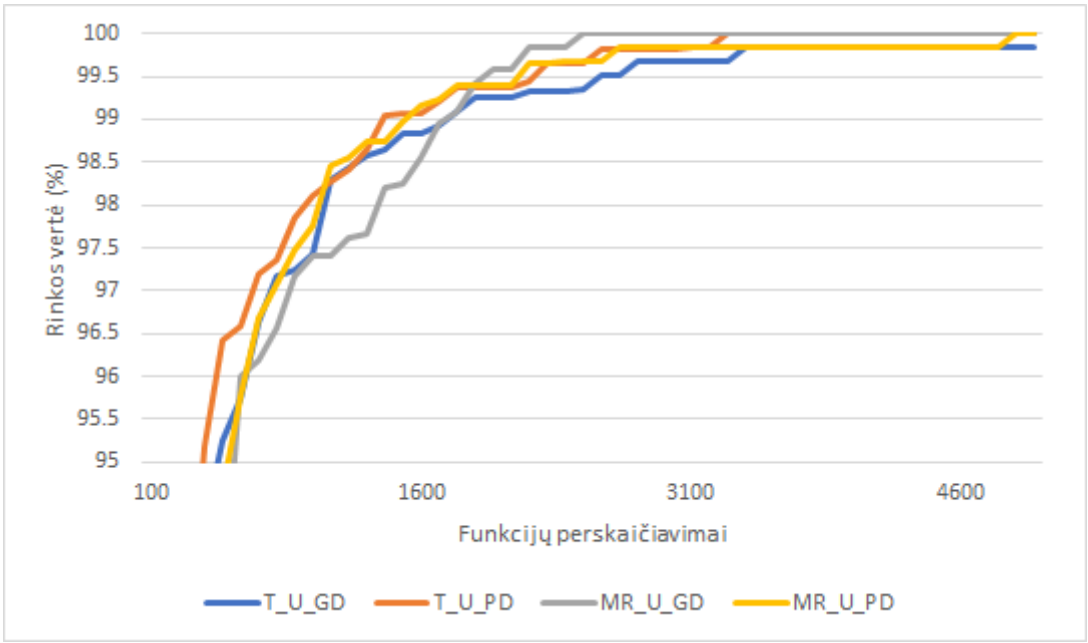
Priedas Nr. 6. Įvairių derinių I tyrimas. Grupė 1. Proporcinis modelis.



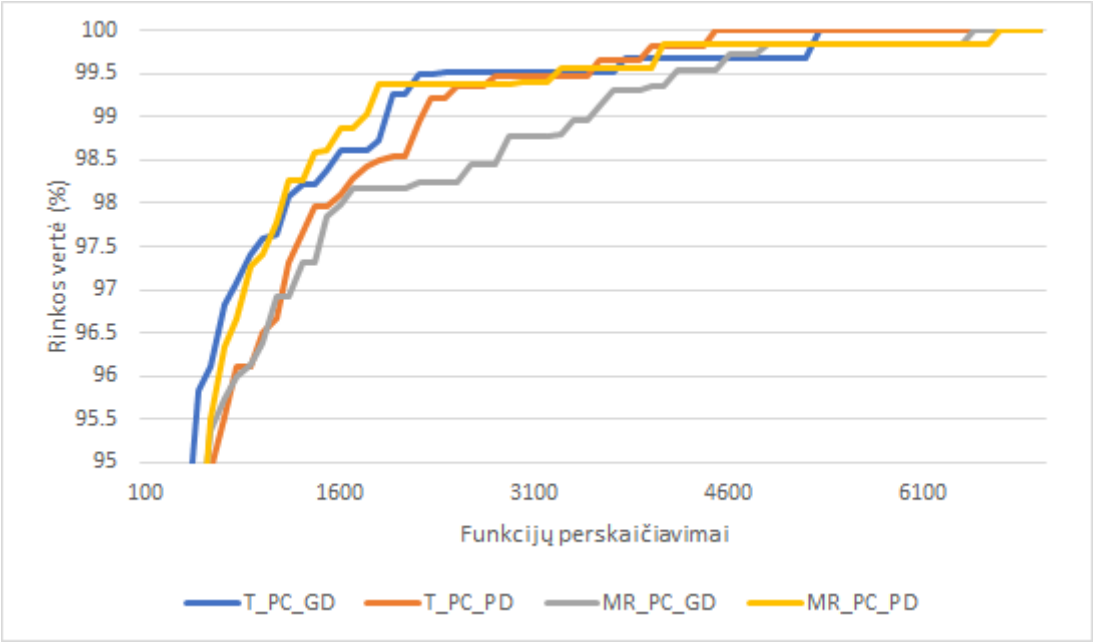
Priedas Nr. 7. Įvairių derinių I tyrimas. Grupė 2. Proporcinis modelis.



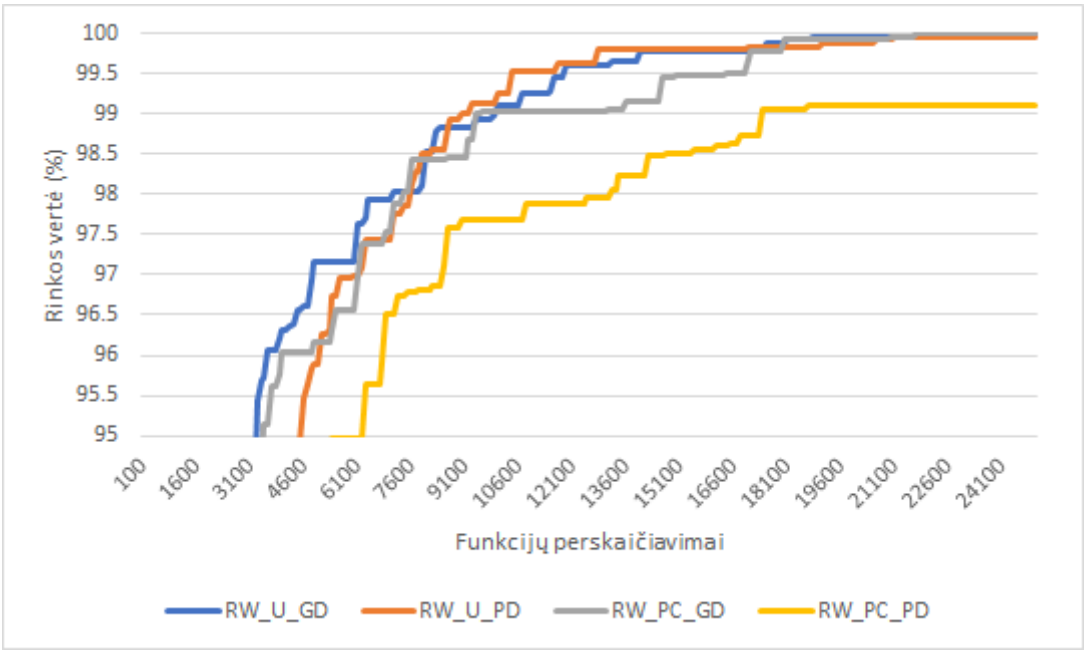
Priedas Nr. 8. Įvairių derinių I tyrimas. Grupė 3. Proporcinis modelis.



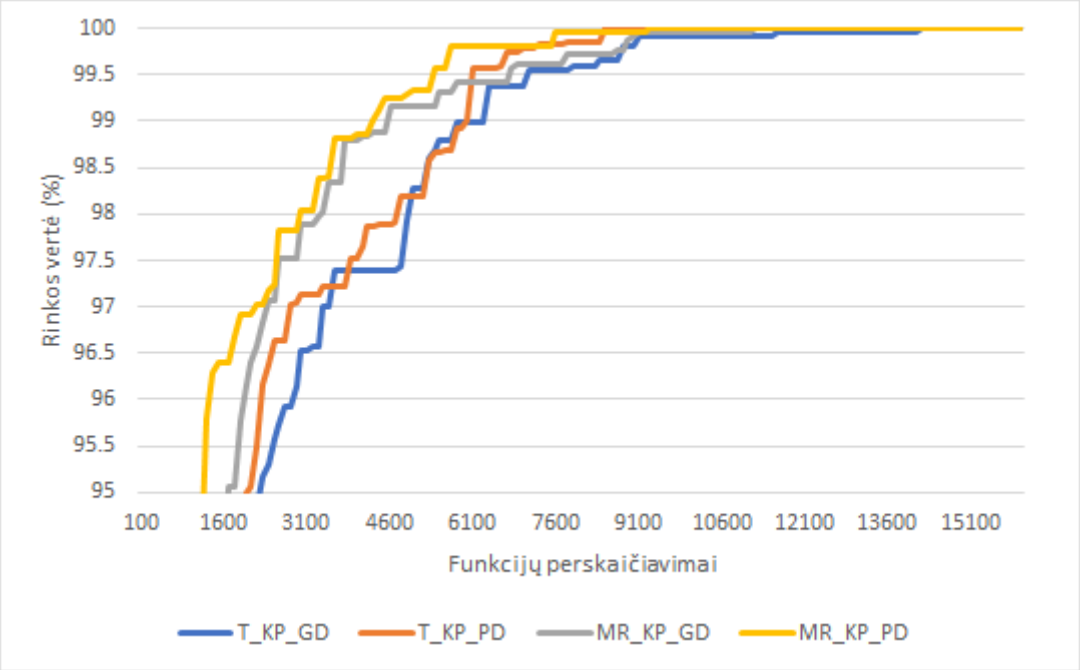
Priedas Nr. 9. Įvairių derinių I tyrimas. Grupė 4. Proporcinis modelis.



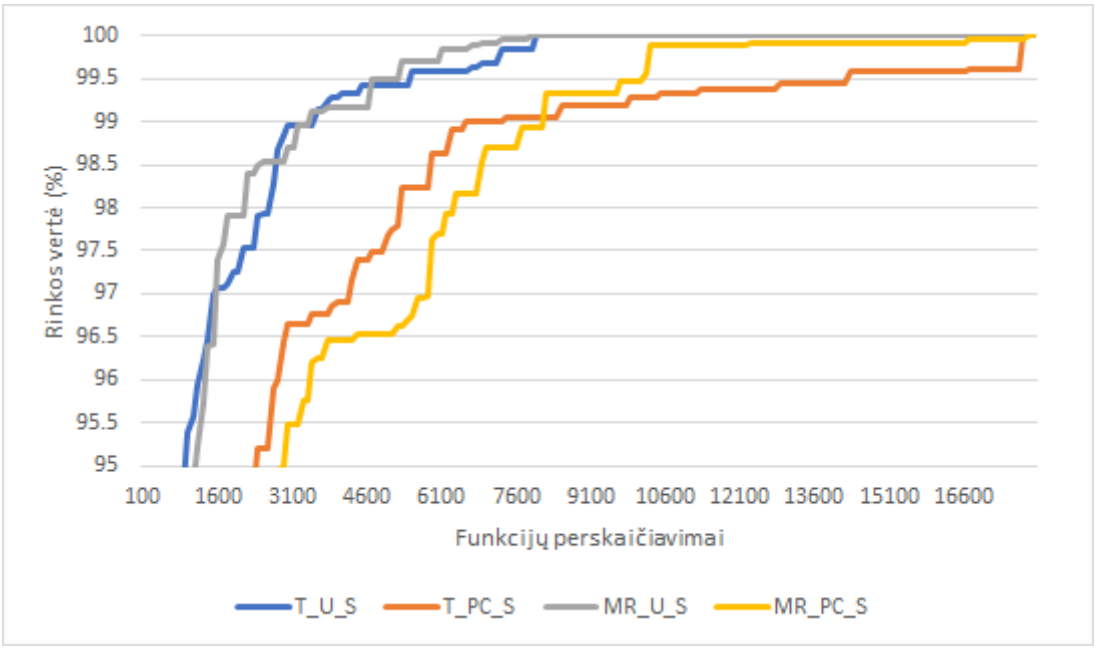
Priedas Nr. 10. Įvairių derinių I tyrimas. Grupė 5. Proporcinis modelis.



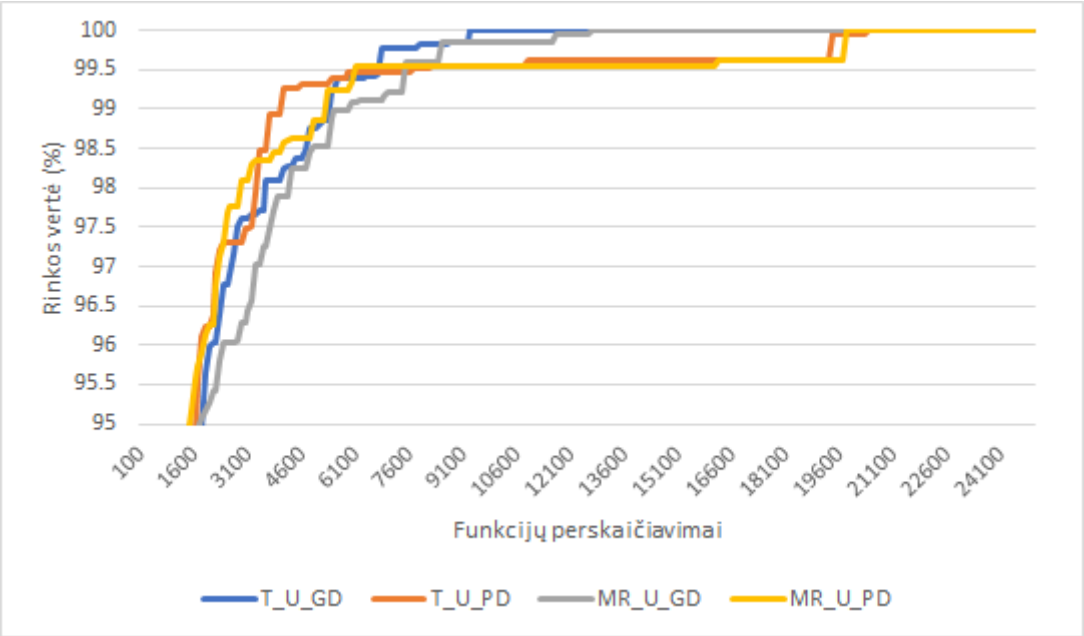
Priedas Nr. 11. Įvairių derinių II tyrimas. Grupė 1. Binarinis modelis.



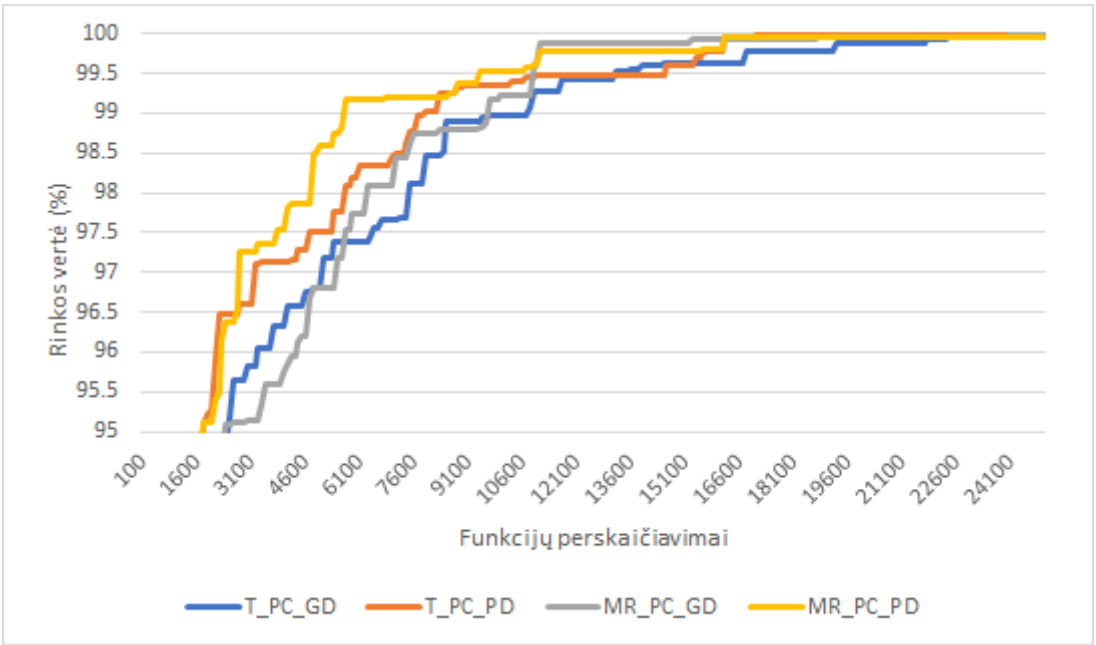
Priedas Nr. 12. Įvairių derinių II tyrimas. Grupė 2. Binarinis modelis.



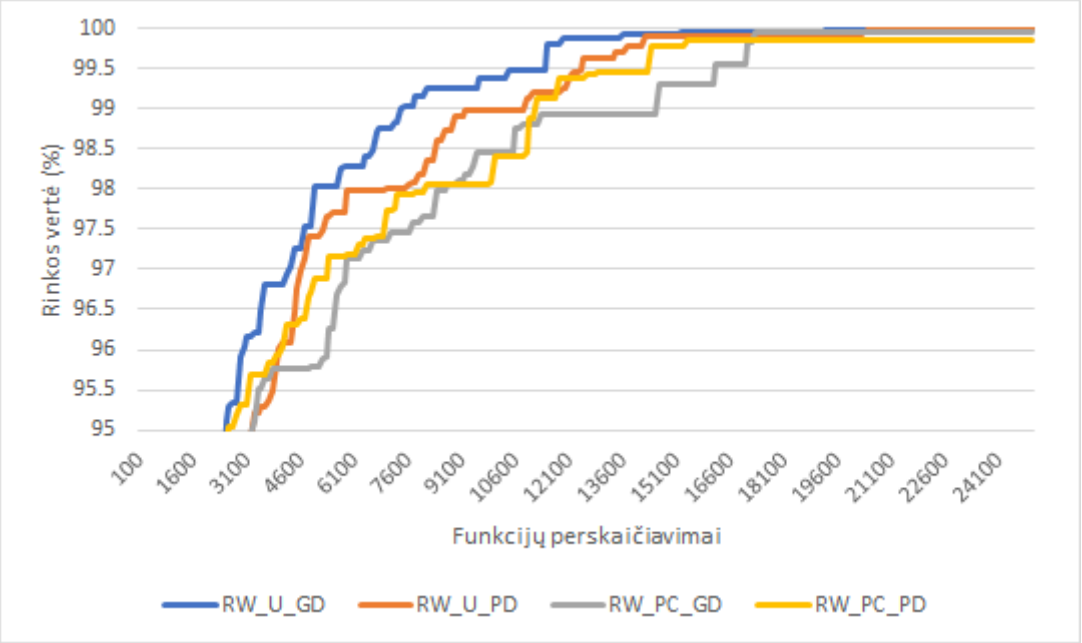
Priedas Nr. 13. Įvairių derinių II tyrimas. Grupė 3. Binarinis modelis.



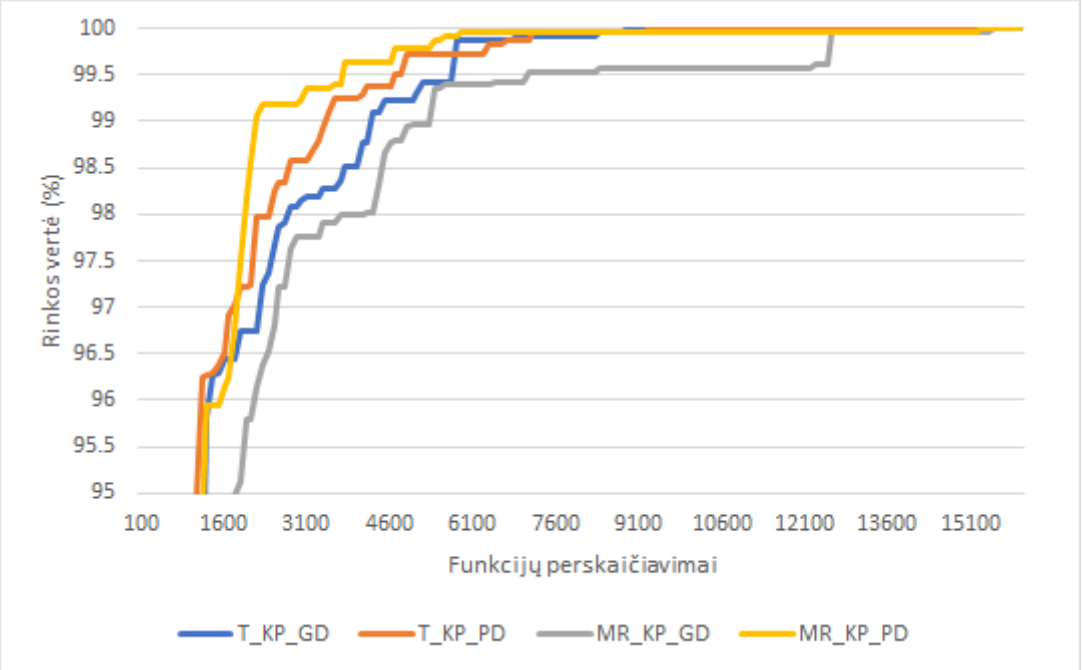
Priedas Nr. 14. Įvairių derinių II tyrimas. Grupė 4. Binarinis modelis.



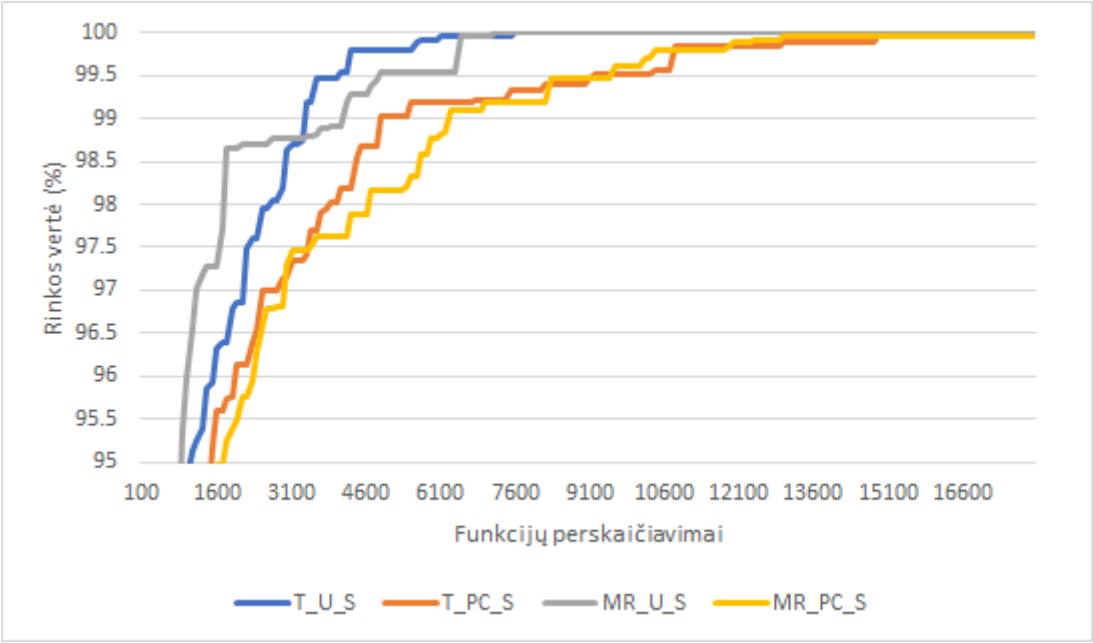
Priedas Nr. 15. Įvairių derinių II tyrimas. Grupė 5. Binarinis modelis.



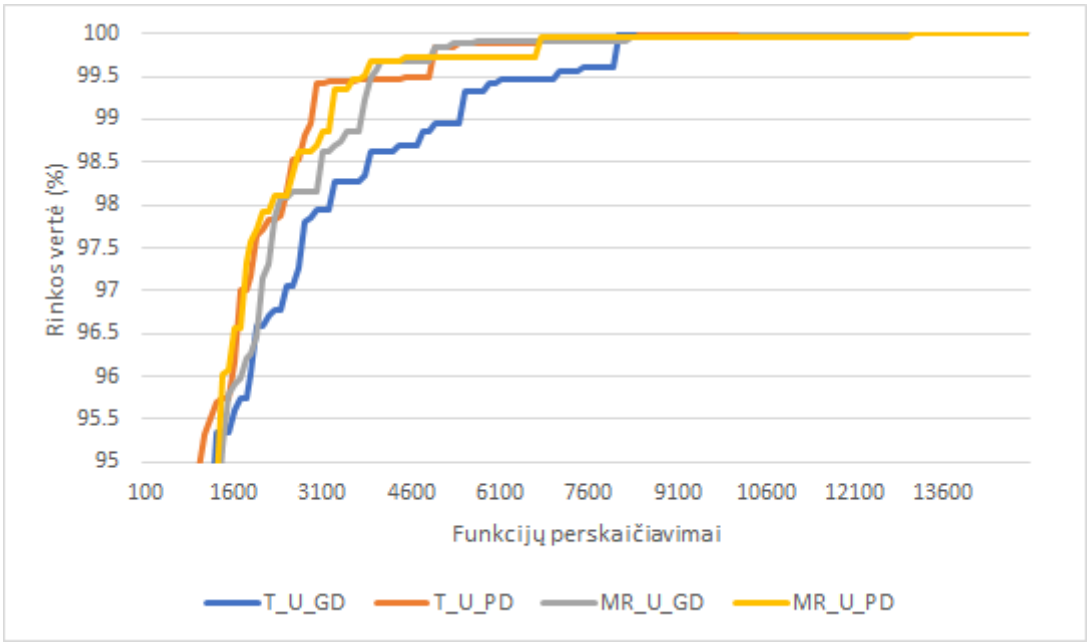
Priedas Nr. 16. Įvairių derinių II tyrimas. Grupė 1. Proporcinis modelis.



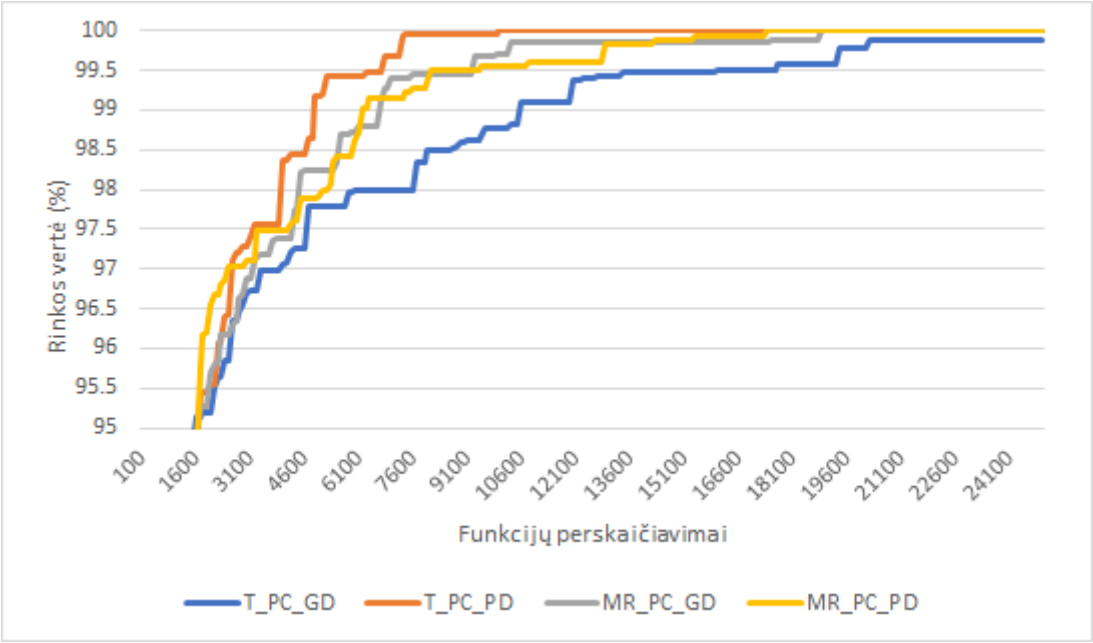
Priedas Nr. 17. Įvairių derinių II tyrimas. Grupė 2. Proporcinis modelis.



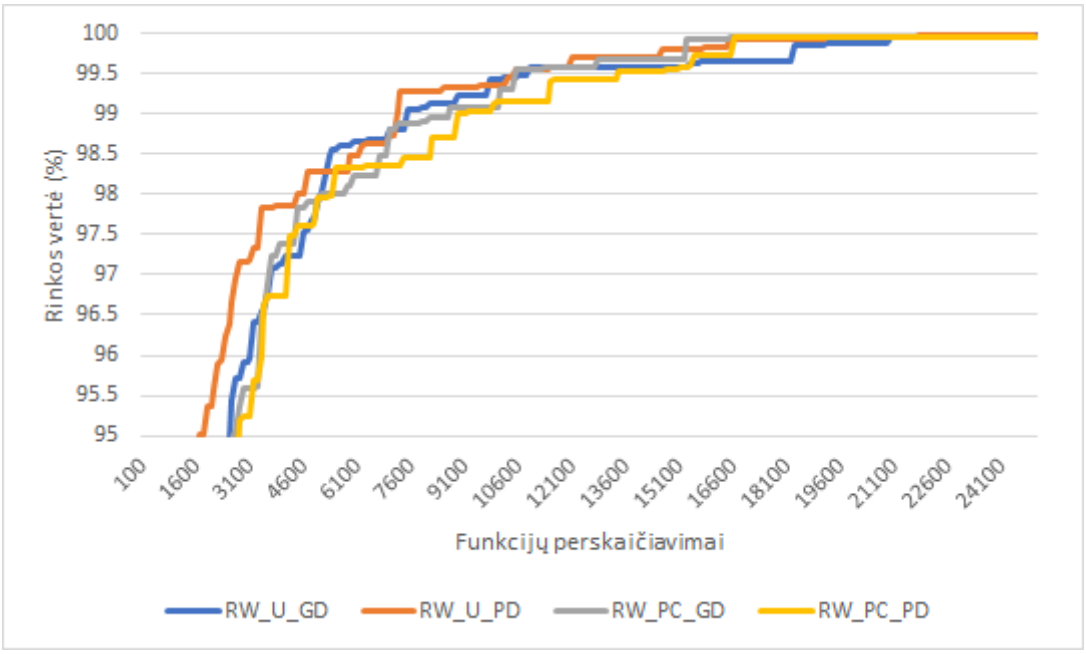
Priedas Nr. 18. Įvairių derinių II tyrimas. Grupė 3. Proporcinis modelis.



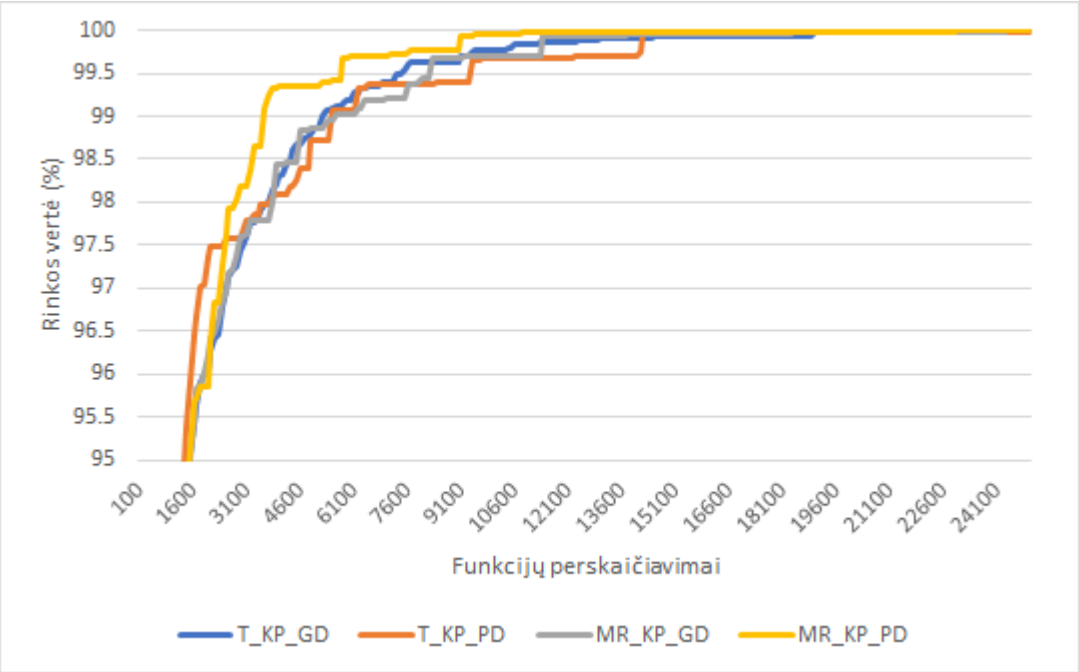
Priedas Nr. 19. Įvairių derinių II tyrimas. Grupė 4. Proporcinis modelis.



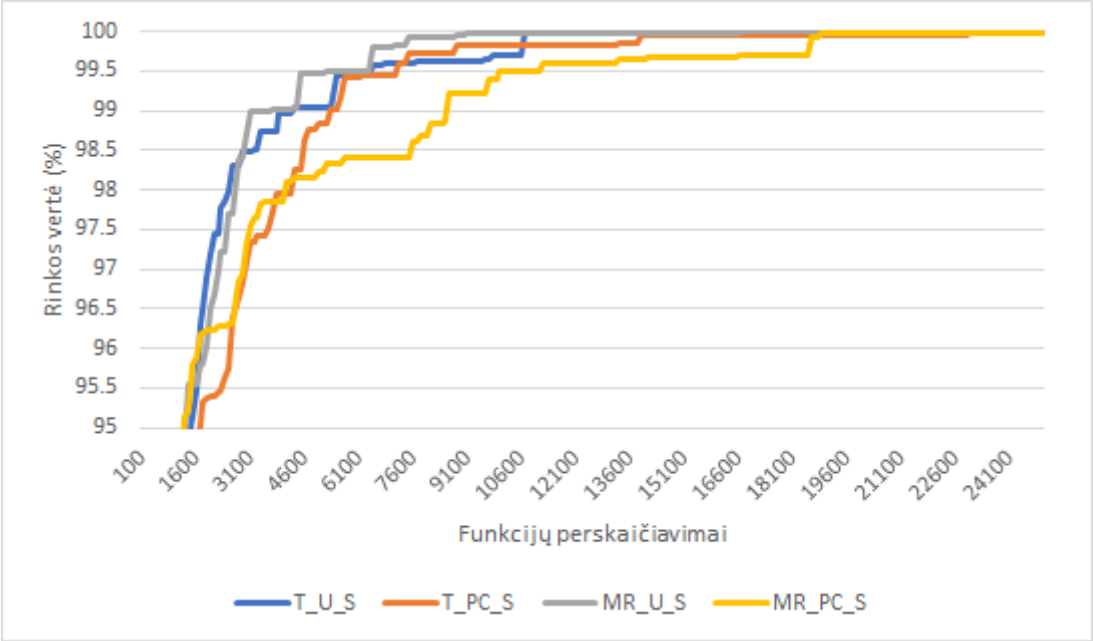
Priedas Nr. 20. Įvairių derinių II tyrimas. Grupė 5. Proporcinis modelis.



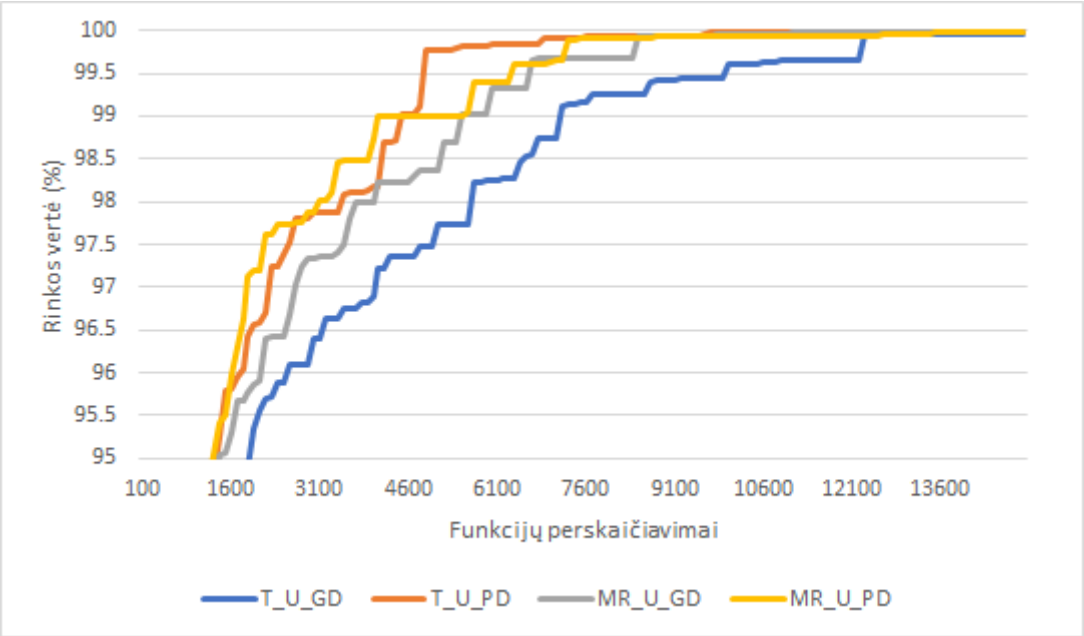
Priedas Nr. 21. Įvairių derinių III tyrimas. Grupė 1. Binarinis modelis.



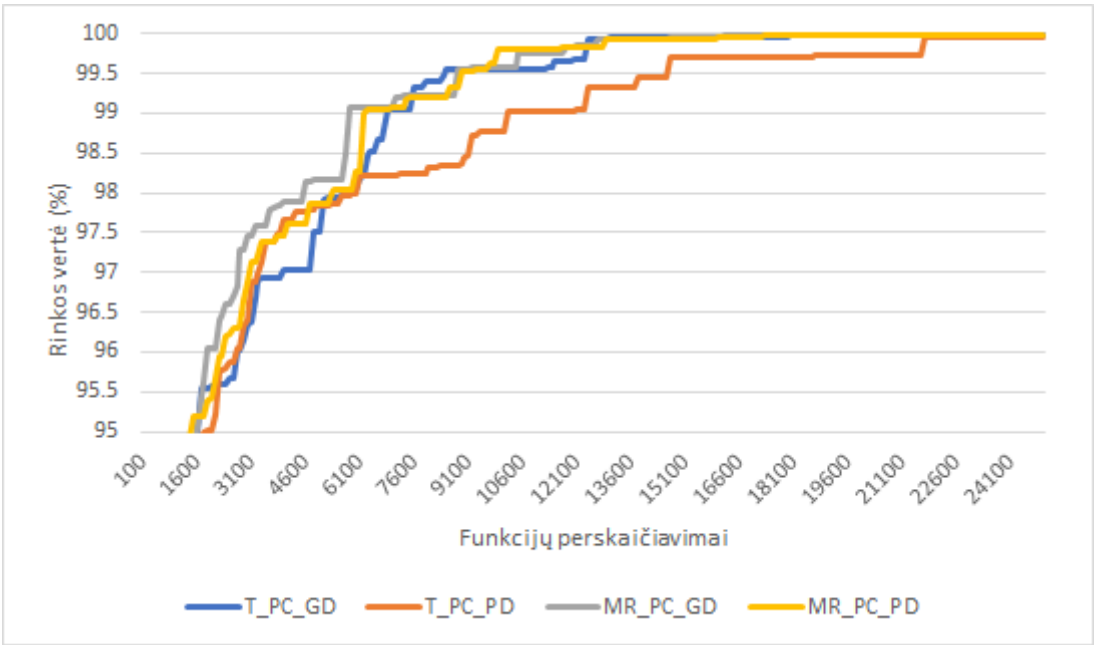
Priedas Nr. 22. Įvairių derinių III tyrimas. Grupė 2. Binarinis modelis.



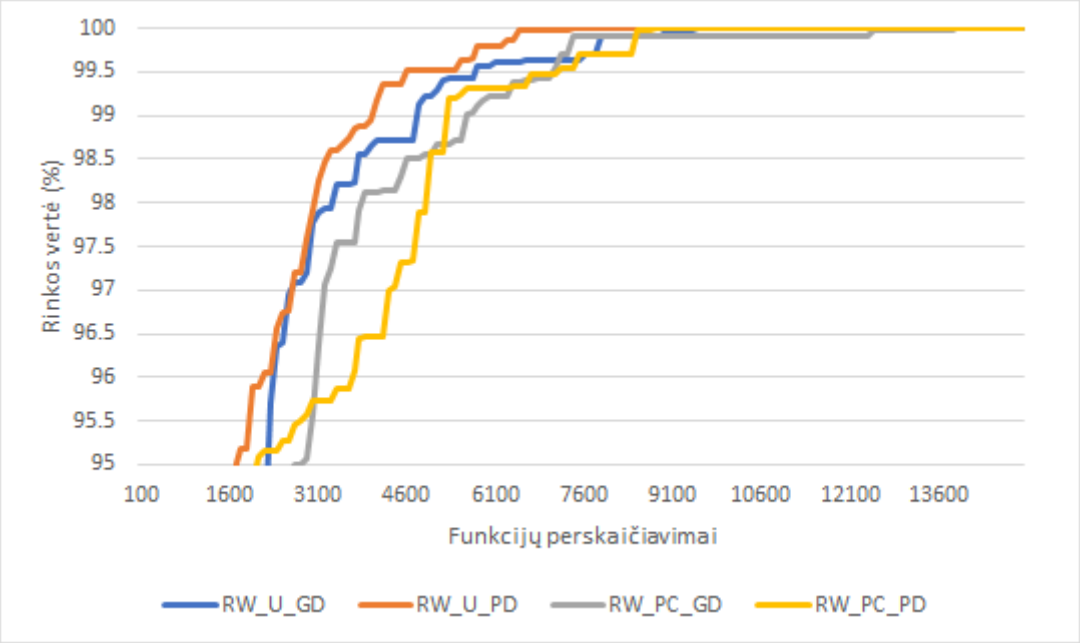
Priedas Nr. 23. Įvairių derinių III tyrimas. Grupė 3. Binarinis modelis.



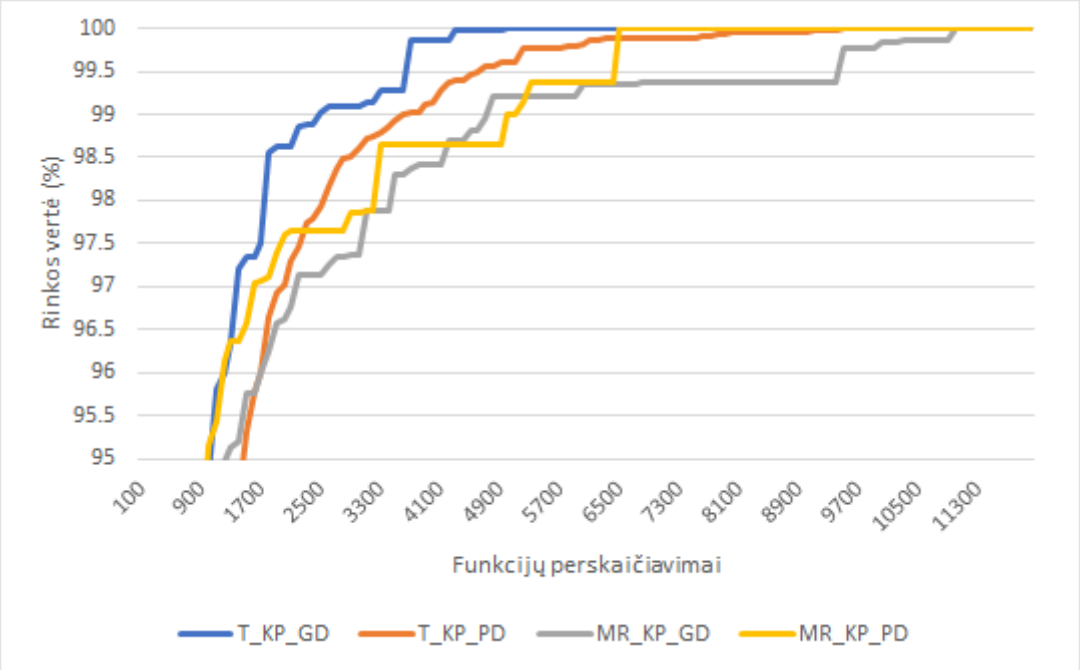
Priedas Nr. 24. Įvairių derinių III tyrimas. Grupė 4. Binarinis modelis.



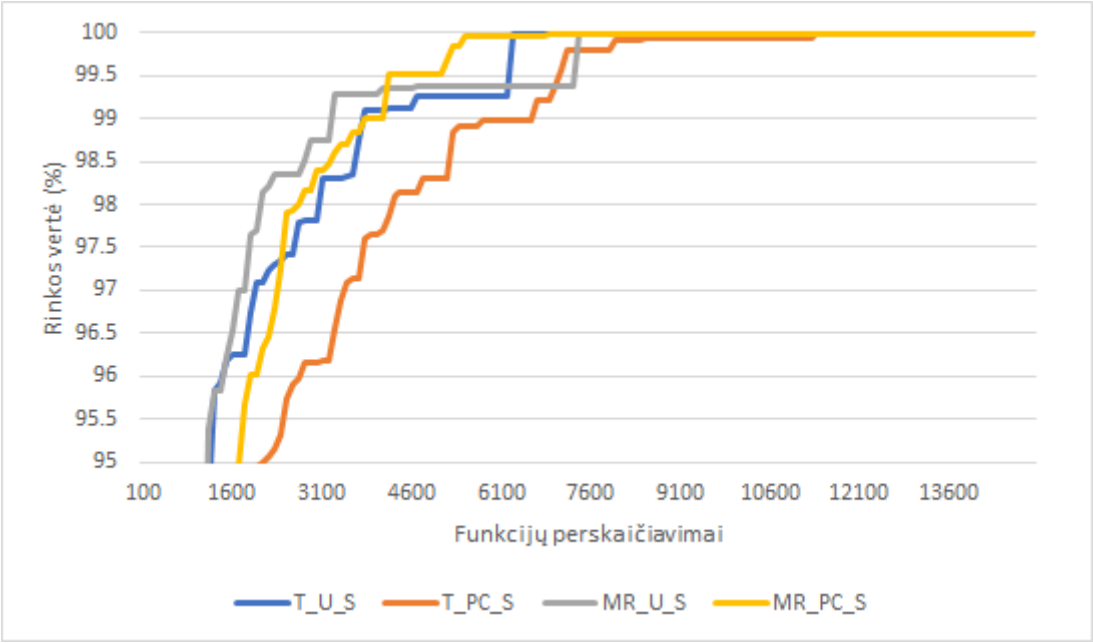
Priedas Nr. 25. Įvairių derinių III tyrimas. Grupė 5. Binarinis modelis.



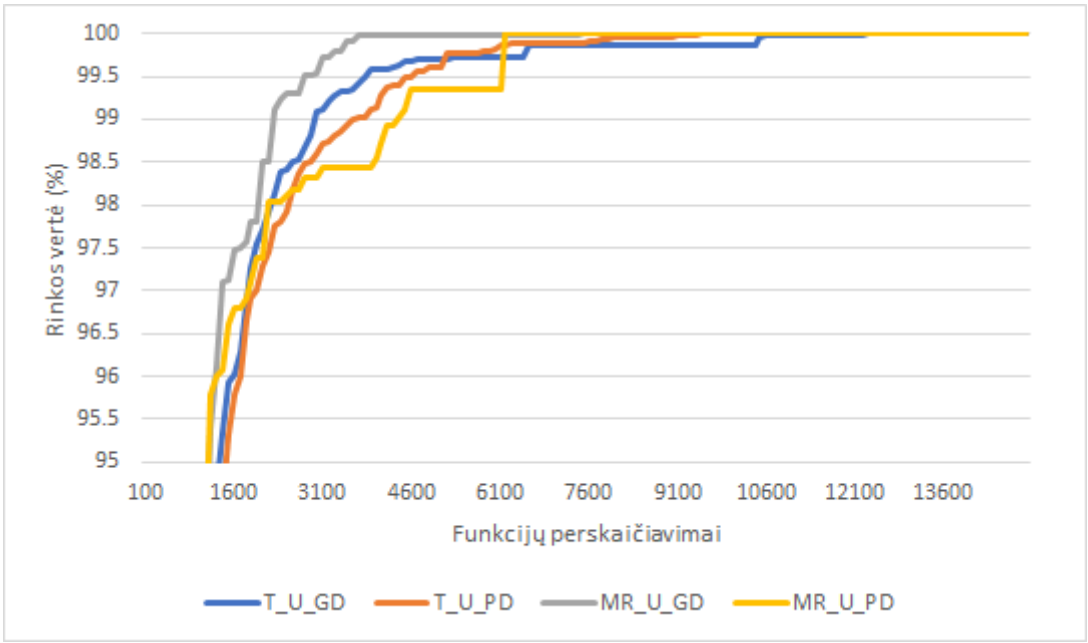
Priedas Nr. 26. Įvairių derinių III tyrimas. Grupė 1. Proporcinis modelis.



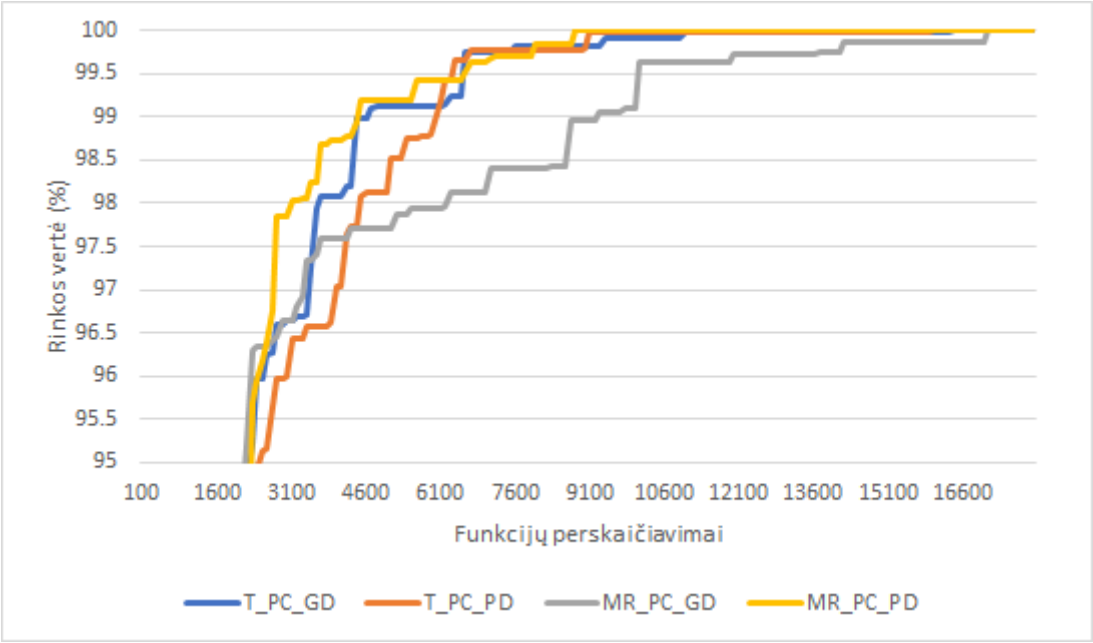
Priedas Nr. 27. Įvairių derinių III tyrimas. Grupė 2. Proporcinis modelis.



Priedas Nr. 28. Įvairių derinių III tyrimas. Grupė 3. Proporcinis modelis.



Priedas Nr. 29. Įvairių derinių III tyrimas. Grupė 4. Proporcinis modelis.



Priedas Nr. 30. Įvairių derinių III tyrimas. Grupė 5. Proporcinis modelis.