KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PRAKTINĖS INFORMATIKOS KATEDRA

Donatas Šidla

Ruošinių realizacijos optimizavimo procedūrų sudarymas ir tyrimas

Informatikos mokslo magistro baigiamasis darbas

Darbo vadovas
doc. dr. Antanas Lenkevičius

Kaunas, 2004

SANTRAUKA

D. Šidla. Ruošinių realizacijos optimizavimo procedūrų sudarymas ir tyrimas: Magistro darbas/ Vadovas doc.dr. A. Lenkevičius,- - Kaunas: KTU, 2004. 53 p.

Magistro darbo objektas – ruošinių realizacijos optimizavimo procedūrų sudarymas ir tyrimas.

Darbą sudaro keturios pagrindinės dalys: problemos analizė, teoriniai sprendimai, eksperimentai ir jų rezultatai bei išvados ir rekomendacijos.

Pirmoje dalyje nagrinėjami įvairūs optimizavimo metodai, išskiriami šių metodų privalumai ir trūkumai. Parenkamas geriausias optimizavimo metodas nagrinėjamos problemos sprendimui.

Teoriniuose sprendimuose aprašomos sudarytos optimizavimo procedūros: pilnas perrinkimas, tabu paieška bei pasiūlytas algoritmas. Šių procedūrų palyginimui sukurta programa.

Eksperimentinėje dalyje pateikiami atlikto tyrimo rezultatai. Sudarytos optimizavimo procedūros lyginamos pagal surasto maršruto ilgį, proceso trukmę, skaičiavimų laiką ir kitus parametrus.

Išvadose apibendrinami darbo rezultatai ir pateikiamos ruošinių realizacijų optimizavimo procedūrų taikymo rekomendacijos.

Rašant magistro darbą buvo naudota literatūra lietuvių ir anglų kalbomis.

Darbe pateikta 19 lentelių ir 29 paveikslai.

SUMMARY

D. Šidla. Creation and research optimization of blank realization procedures: Master Thesis / Research adviser doc.dr. A. Lenkevičius,-
- Kaunas: KTU, 2004. 53 p.

The object of Master Thesis is the creation and research optimization of blank realization procedures.

Four main parts comprise the thesis: analysis of the problem, theoretical solutions, research and outcomes, conclusions and recommendations.

Optimization methods, their advantages and limitations have been analyzed in the first part. The most appropriate optimization method for problem solving was selected.

Designed optimization methods have been presented in the theoretical solutions part: exhaustive search, tabu search and a new one, presented by the author. A program was created for comparison of these methods.

In the third part of the thesis outcomes of the research are presented. Designed methods were tested and compared by their routes length, process time, computations duration and other parameters.

Conclusions of the thesis and recommendations for implementation of analyzed procedures have been presented in the part of conclusions.

Literature in Lithuanian, English was used in preparing the thesis.

There are 19 tables and 29 pictures in the thesis.

Turinys

ĮVADAS

Optimizacijos uždaviniai yra svarbūs daugelyje taikymo sričių. Tai gali būti telefono tinklų apkrovimas, įvairios transporto schemos ir maršrutai, pavyzdžiui, šiukšlių išvežimas ar prekių pristatymas ir kt. Optimizacijos uždavinių sprendimui siūlomi įvairūs metodai ir algoritmai.

Ruošinio pavyzdžiu galėtų būti spausdinto montažo plokštė, plokštė, skirta baldų gamybai, medžiaga rūbams siūti, stogo danga. Ruošinio realizacija suprantama kaip jau turimas ruošinio projektas. Pavyzdžiui, spausdinto montažo plokštės atveju žinoma, kaip išdėstyti takeliai, reikia su mažiausiomis laiko sąnaudomis nubrėžti šiuos takelius. Sprendžiant ruošinių realizacijos optimizavimo uždavinius galima padidinti įrenginių, pavyzdžiui, braižiklio, darbo greitį, našumą, pailginti eksploatacijos laiką.

Darbo tikslas – optimizuoti braižymo, pjaustymo ar karpymo procesą, t. y. tam tikro įrenginio (pavyzdžiui, braižiklio), atliekančio tam tikrą darbą (pavyzdžiui, braižymą), proceso trukmę. Sudarant ruošinių realizacijos optimizavimo procedūras, reiktų įvertinti šių ruošinių technologinio realizavimo ypatumus, pavyzdžiui, spausdinto montažo plokštės takelių storį (takelis gali būti brėžiamas įvairaus storio plunksnomis), įrenginio darbo parametrus, pavyzdžiui, braižiklio darbo greitį brėžiant liniją ir judant tuščiąja eiga.

Optimalus sprendinys gaunamas atlikus pilną perrinkimą, t. y. peržiūrėjus visus įmanomus sprendinius ir išrinkus patį geriausią. Tačiau toks problemos sprendimo būdas daugeliu atveju neįmanomas. Augant uždavinio dydžiui, skaičiavimų apimtys didėja labai sparčiai. Pilnas perrinkimas gali būti naudingas tik teoriniu požiūriu, lyginant gautą sprendinį po pilno perrinkimo su sprendiniu, gautu kitais metodais, šitaip įvertinant rasto sprendinio tinkamumą.

Ruošinių realizacijos optimizavimui gali būti naudojami algoritmai, kurie neranda optimalaus sprendinio, bet jų rezultatas – pakankamai optimalus sprendinys, gaunamas per polinominį laiką. Šie apytiksliai algoritmai randa sprendinį, kuris yra pakankamai arti optimalaus sprendinio. Naudojant apytikslius algoritmus, gana tikslų sprendinio paieškos laiką galima numatyti iš anksto.

Ruošinių realizacijos optimizavimui gali būti naudojami ir euristiniai algoritmai. Jie atrenka nebūtinai patį geriausią, o tikėtinai geriausią sprendinį. Tačiau euristiniai algoritmai yra efektyvūs laiko prasme. Lyginant su apytiksliais algoritmais, iš anksto nusakyti euristinių algoritmų sprendinio paieškos trukmę yra sudėtingiau.

Praktikoje orientuojamasi tik į trumpiausią maršrutą. Tačiau, surastas trumpiausias maršrutas dar negarantuoja trumpiausios ruošinio realizacijos proceso trukmės. Todėl tema

„Ruošinių realizacijos optimizavimo procedūrų sudarymas ir tyrimas“ yra aktuali ne tik teoriniu, bet ir praktiniu aspektu. Nagrinėjamos problemos sprendimui, siūlomi optimizavimo metodai optimalaus ruošinių realizacijos ciklo rodiklių (ne vien trumpiausio maršruto) paieškai, atliekamas šių metodų teorinis ir modelinis tyrimas.

1. ANALITINĖ DALIS

1.1. Optimizavimo metodai

Telekomunikacijoje, logistikoje, finansiniuose planavimuose, transportavime ar gamyboje susiduriama su gausybe keblių optimizavimo problemų, kurios paskatino efektyvių optimizavimo metodų sukūrimą. Šie metodai yra įvairių idėjų pritaikymo praktiniuose tyrimuose, rezultatas.

Metodologijų struktūrą skatinančias idėjas dažnai lemia neįprastos sąsajos. Pavyzdžiui, tinklų srautų (*Network flow*) programavimas pagrįstas modeliais, kurių pagrindas – elektros ir hidraulikos dėsniai; atkaitinimo (*Simulated annealing*) algoritmas pagrįstas fiziniais procesais metalurgijoje; genetiniai algoritmai (*Genetic algorithms*) atkartoja evoliucinio dauginimosi biologinius reiškinius; tuo tarpu skruzdėlės (*Ant system*) metodas imituoja skruzdžių koloniją, kuri bendradarbiauja sprendžiant bendras problemas. Neuroniniai tinklai (*Neural networks*) kuriami pagal smegenų veiklos mechanizmą. Tabu paieškos (*Tabu search*) esmė – nustatyti ir panaudoti tinkamiausius problemų sprendimų principus. Glover F., Laguna M. (2001) teigia, kad tabu paieška pagrįsta idėjomis, siejančiomis dirbtinio intelekto ir optimizacijos sritis.

Matematiniai optimizavimo metodai dažniausiai įvertinami pagal maksimalią paklaidą. Mockus J., Eddy W., Reklaitis G. ir kt. (1997) tai vadina blogiausio atvejo analize arba *Minimax metodu*. Šio metodo pranašumas praktiniu aspektu, jog užtikrinama, kad paklaida neviršys tam tikros ribos. Minimax metodo pranašumas teoriniu aspektu, kad nebus naudojami subjektyvūs sprendimai arba netikslūs duomenys. Tačiau tokio pasirinkto problemos sprendimo būdo kaina gali būti per didelė.

Praktikoje sprendžiant optimizavimo problemas dažnai nepavyksta išvengti eksponentinio skaičiavimų laiko. To galima išvengti naudojant ne tik griežtai apibrėžtas specifines savybes, bet ir ekspertinį patyrimą. Mockus J., Eddy W., Reklaitis G. ir kt. (1997) subjektyvią ekspertinę patirtį vadina euristine patirtimi ar kitaip – *euristika*.

Nils N. (1971) euristiką apibūdina kaip aibę taisyklių nustatant sprendimų prioritetus, kurie paremti ekspertų patirtimi ir intuicija. Prioritetai nusakomi dviem būdais:

1. sprendimų klasifikavimo taisyklėmis (jos išskiria geresnius sprendimus, bet nenurodo, kiek šie sprendimai yra geresni);
2. sprendinių aibės naudingumo funkcija.

Mockus J., Eddy W., Reklaitis G. ir kt. (1997) skiria šiuos euristikų tipus:

- „godžios“ (*greedy*) euristikos, kurios naudoja lokalią informaciją, siekiant sukurti sistemą iš tinkamiausių elementų;
- „sukeitimų“ (*permutation*) euristikos, kurios paiešką pradeda nuo galimo sprendinio, o naudojantis lokalia informacija, nustato, kurį iš galimų sukeitimų pasirinkti;
- „deterministinės paieškos“ (*deterministic search*) euristikos, kurios naudoja gerai žinomą apytikrį algoritmą. Šių euristikų pavyzdžiu yra Gupta euristikos;
- „atsitiktinės paieškos“ (*stochastic search*) euristikos, kurios sprendinio radimui taiko atsitiktinės paieškos algoritmą, pavyzdžiui, atkaitinimo algoritmą.

Pasak Helman, P., Moret, B., Shapiro, H. (1993) „godžios“ euristikos formuojamą sprendinį pradžioje sudaro tuščia aibė, į kurią vis įtraukiami nauji elementai. Šis procesas vykdomas, kol yra tinkamų elementų. Šiuo atveju sprendimų etapų skaičius lygus sistemos elementų skaičiui.

„Sukeitimų“ euristikos pradinį sprendinį sudaro tam tikra pirminė sistemos būsena, kuri vėliau yra keičiama. Jei duotoje sukeitimų aibėje sprendinio pagerinti nebeįmanoma, šios euristikos baigia sprendinio formavimo procesą. Taikant šią euristiką, sprendimų etapų skaičius gali stipriai viršyti sistemos elementų skaičių. „Sukeitimų“ euristikos privalumas – didelė sukeitimo variantų įvairovė. Kitas privalumas tas, kad pradinio sprendinio parinkimui, galima pritaikyti sukauptą patirtį.

Jeigu nėra tinkamo pradinio sprendinio, jį galima optimizuoti dviem etapais. Pirmajame etape naudojamas „godusis“ algoritmas, kuris pradinį sprendinį formuoja pradėdamas tuščia aibe. Antrajame etape šis sprendinys pagerinamas, atliekant tinkamus sukeitimus. Jei sutelktų žinių pagalba randamas pradinis sprendinys, pirmojo etapo galima atsisakyti.

„Atsitiktinės paieškos“ euristikoms priskiriami anksčiau paminėti atkaitinimo algoritmas, tabu algoritmas, genetiniai algoritmai, taip pat GRASP (*Greedy Randomized Adaptive Search Procedures*) sistemos.

Nagrinėjamos problemos sprendimui galima pritaikyti aukščiau minėtose euristikose naudojamas idėjas.

1.2. Tabu paieška

1.2.1 Tabu paieškos samprata

Tabu paieškos algoritmo idėją pirmieji iškėlė Glover F. (1986) ir Hansen P. (1986). Glover F., Laguna M. (2001) tabu paiešką vadina meta-euristika. Ši meta-euristika valdo lokalią euristinę paieškos procedūrą, kad ištirtų sprendinio erdvę virš lokalaus optimumo. Meta-euristika vadovaujasi kitas euristika valdančia ir modifikuojančia strategija, kuri įgalina surasti geresnius sprendinius, lyginant su surastais, ieškant lokalaus optimumo. Šių meta-strategijų valdomos euristikos gali būti tiek sudėtingos procedūros, tiek procedūros, atliekančios tik šias funkcijas:

- galimų veiksmų, naudojamų sprendinių transformavimui iš vieno į kitą, aprašymą;
- su tuo susijusių įvertinimo taisyklių aprašymą.

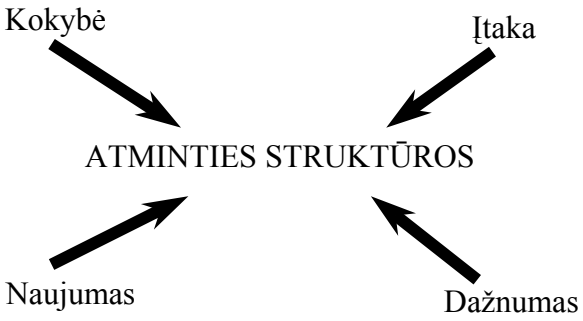
Vienas iš pagrindinių tabu paieškos komponentų yra prisitaikančiosios atminties, sukuriančios daug lankstesnį paieškos funkcionavimą, naudojimas. Todėl atmintimi pagrįstos strategijos yra skiriamasis tabu paieškos metodų požymis.

Tabu paieška grindžiama prielaida, kad sprendžiama problema turi sujungti prisitaikančiosios atminties (*adaptive memory*) principus ir atsakomųjų tyrimų (*responsive exploration*) strategiją. Glover F., Laguna M. (2001) tabu paiešką lygina su kopimu į kalną, kai alpinistas atrankos būdu privalo atsiminti nueitame kelyje paliktus kaiščius (prisitaikančios atminties naudojimas) ir sugebėti pasirinkti tolesnį kelią (atsakomųjų tyrimų naudojimas). Prisitaikančios atminties naudojimas tabu paieškoje leidžia sudaryti procedūras, kurios ekonomiškai ir efektyviai vykdo paiešką sprendinių erdvėje. Kadangi lokalūs pasirinkimai atliekami pagal paieškos metu surinktą informaciją, tabu paieška skiriasi nuo nereikalaujančio atminties projektavimo. Nereikalaujantį atminties projektavimą įtakoja pusiau atsitiktiniai procesai, realizuojantys atrankos modelį. Tokių metodų pavyzdžiu yra pusiau „godžios“ euristikos, tradicinis atkaitinimo ar evoliuciniai metodai. Prisitaikančioji atmintis skiriasi ir nuo nelanksčiosios atminties, kuri būdinga šakų ir ribų (*Branch and Bound*) strategijoms.

Tabu paieška gali atmesti lokalų minimumą ir ieškoti už jo esančiose srityse. Ji taip pat gali rasti globalų minimumą multimodalinėse paieškos erdvėse. Procesas, kurio pagalba tabu paieška įveikia lokalaus optimumo problemą, pagrįstas įvertinimo funkcija. Ši funkcija kiekvienoje iteracijoje išrenka geriausiai įvertintą sprendinį. Naudojant objektų įvertinimus ir tabu apribojimus artėjama prie geriausio leistino sprendinio parinkimo iš einamųjų sprendinių

kaimynystės. Įvertinimo funkcija parenką tokį žingsnį, kuris lemia objekto funkcijos didžiausią pagerinimą arba mažiausią pabloginimą. Tabu sąrašas naudojamas tinkamų žingsnių charakteristikų saugojimui. Vėlesnėse iteracijose šios charakteristikos gali būti panaudotos priskiriant tabu tam tikriems žingsniams, t. y. nurodant, kurių žingsnių reikia vengti. Kitaip sakant, tabu sąrašas apibrėžia, kokie sprendiniai gali būti randami pradedant einamuoju sprendiniu. Kadangi tabu paieškoje leidžiami geresnio sprendinio nerandantys žingsniai, kartais grįžtama į anksčiau aplankytas vietas. Šitaip iškyla ciklo problema. Šios problemos sprendimui taikomas tabu sąrašas. Tabu sąrašui valdyti ir atnaujinti taikoma vadinamoji *draudžiančioji strategija*. Naudojant šią strategiją, vengiama prieš tai aplankyto kelio, o nagrinėjamos tik naujos paieškos erdvės sritys.

Glover F., Laguna M. (2001) tabu paieškoje atminties struktūras skiria pagal keturias pagrindines dimensijas: naujumas, dažnumas, kokybė, įtaka (žr. 1.1 pav.).



1.1 pav. Keturios tabu paieškos dimensijos
(pagal Glover F., Laguna M., 2001)

Naujumu grindžiama ir dažnumu grindžiama atmintys papildo viena kitą. Kokybės dimensija leidžia atskirti paieškos metu rastų sprendinių tinkamumą. Šiuo atveju atmintis gali būti naudojama nustatant elementus, kurie yra bendri tinkamam sprendiniui arba maršrutui, vedančiam prie šių sprendinių. Kokybė sudaro skatinimu grindžiamo mokymosi pagrindą. Čia skatinami veiksmai, kuriais randami tinkami sprendiniai, o draudžiami tie veiksmai, kurie lemia prastesnių sprendinių radimą. Įtakos dimensija įvertina ne tik paieškos metu rastų sprendinių kokybę, bet ir jų struktūrą. Glover F., Laguna M. (2001) teigimu, kokybė gali būti suprantama kaip speciali įtakos forma.

Tabu paieškoje naudojama tiek detalioji (*explicit*), tiek atributinė (*attributive*) atmintis. Detalioji atmintis įrašo galutinius sprendinius, kuriuos dažniausiai sudaro paieškos metu aplankyti geriausi sprendiniai. Šios atminties plėtra įrašo geras, bet dar neištirtas geriausiam sprendiniui artimas reikšmes. Lokalios paieškos plėtrai naudojami įsiminti geriausi

sprendiniai arba jiems artimos reikšmės. Tam tikrais atvejais detalioji atmintis naudojama siekiant išvengiant tų pačių sprendinių lankymo daugiau kaip viena kartą. Tačiau toks atminties taikymas yra ribotas, nes „protingos“ duomenų struktūros turi būti projektuojamos siekiant išvengti pernelyg didelių atminties poreikių.

Be detaliosios atminties tabu paieškos rezultatų valdymui naudojama atributinė atmintis. Šis atminties tipas įrašo informaciją apie sprendinių atributus, kurie keičiasi pereinant nuo vieno sprendinio prie kito. Pavyzdžiui, grafų arba tinklų atveju atributus gali sudaryti viršūnės ar lankai, kurie yra pridedami, atmetami ar išsaugomi kiekviename žingsnyje.

1.2.2 Tabu paieškos strategijos

Glover F. (1989) skiria tris pagrindines paprasto tabu paieškos algoritmo strategijas:

- draudžiančioji strategija;
- išlaisvinančioji strategija;
- trumpalaikė strategija.

Draudžiančioji strategija tikrina, kas patenka į tabu sąrašą. *Išlaisvinančioji strategija* tikrina, kas išmetama iš sąrašo ir kada. *Trumpalaikė strategija* valdo sąveiką tarp pirmosios ir antrosios strategijų, parinkdama bandomąjį sprendinį. Be šių strategijų gali būti nagrinėjama ir strategija, kuri naudoja *vidutinės ir ilgalaikės atminties funkcijas*. Ši strategija tabu paieškos eigoje kaupia informaciją, kuri naudojama paieškos valdymui sekančiuose žingsniuose.

Draudžiančioji strategija. Ši strategija naudojama siekiant išvengti ciklo susidarymo problemos, draudžiant tam tikrus žingsnius, t. y., priskiriant tabu. Šiuo atveju pakanka tikrinti, ar dar kartą nelankomas prieš tai aplankytas sprendinys. Idealiu atveju tabu sąrašas turi saugoti visus prieš tai aplankytus sprendinius ir tikrinti šį sąrašą prieš kiekvieną naują žingsnį. Tačiau šie veiksmai reikalauja daugiau atminties ir skaičiavimų.

Vienas iš paprasčiausių būdų norint išvengti ciklo problemos – negrįžti prie paskutiniame žingsnyje surasto sprendinio. Tačiau šis būdas negarantuoja, kad ciklas nesusidarys. Kitas būdas – negrįžti prie sprendinių, kurie buvo aplankyti per paskutines T_s iteracijų (šie sprendiniai saugomi tabu sąraše). Tokiu būdu sprendinys palaipsniui nutolsta nuo surastų per ankstesnes T_s iteracijas. Pham D.T., Karaboga D. (2000) T_s vadina tabu sąrašo ilgiu arba tabu sąrašo dydžiu. Esant tam tikrai T_s reikšmei ženkliai sumažėja ciklo susidarymo tikimybė. Jei T_s reikšmė per maža, ši tikimybė gali smarkiai išaugti. Esant per didelei T_s

reikšmei, paieška gali nukrypti nuo sričių, į kurias patenka geri sprendiniai (nors šios sritys nebuvo pilnai išnagrinėtos).

Tabu sąrašas įgyvendina vieną iš pagrindinių tabu paieškos trumpalaikės atminties funkcijų. Kaip minėta anksčiau, ji vykdoma registruojant tik paskutiniuosius žingsnius. Jei sąrašas pilnas, kiekvienas naujas žingsnis į sąrašą įrašomas senesnio žingsnio vietoje. Veiksmingiausiai tabu sąrašą galima realizuoti kaip žiedinį masyvą eilės tvarkos (*first-in-first-out*) procedūroje.

Išlaisvinančioji strategija. Ši strategija tikrina, kas išmetama iš sąrašo. Taikant šią strategiją, ištrinami sprendinių tabu apribojimai, todėl jie gali būti peržiūrėti sekančiuose paieškos žingsniuose. Sprendinių tabu atributai tabu sąraše išlieka T_s iteracijų metu. Pham D.T., Karaboga D. (2000) skiria dvi prielaidas, kada sprendinys laikomas priimtinu:

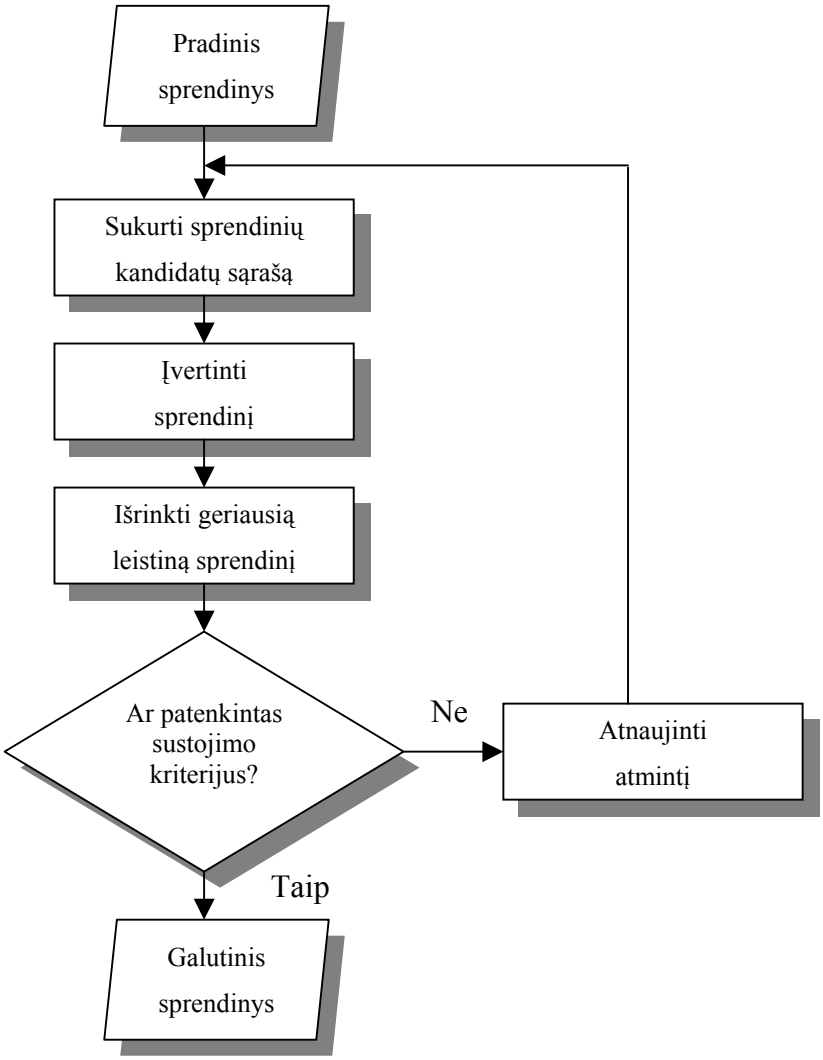
1. jei sprendinio atributai nėra tabu;
2. jei sprendinys išlaiko aspiracijos kriterijaus testą.

Vidutinio ir ilgalaikio mokymosi strategijos. Šios strategijos realizuojamos, naudojant vidutinės ir ilgalaikės atminties funkcijas. Ji įrašo tinkamas pasirinktų žingsnių, kurie generuojami algoritmo vykdymo metu, savybes. Pham D.T., Karaboga D. (2000) ją vertina kaip mokymosi strategiją, kuria ieškoma naujų sprendinių atskleidžiant panašias į anksčiau įrašytas savybes. Viso to pasiekiama apribojant žingsnius, nepasižyminčius tinkamomis savybėmis.

Trumpalaikė strategija. Trumpalaikė strategija valdo pirmųjų strategijų sąveiką (žr. 1.2 pav.).

Kandidatų sąrašas – tai galimų žingsnių sąrašo dalis. Geriausio sprendinio atrankos metodu iš galimų sprendinių, lemiančių didžiausią pagerinimą arba mažiausią pabloginimą, parenkamas leistinas sprendinys. Šis kriterijus grindžiamas prielaida, kad kuo geriau įvertintas sprendinys, tuo didesnė tikimybė, kad per keletą tolesnių žingsnių jis priartės arba prie optimalaus, arba prie gero sprendinio. Jei sprendinys tenkina aspiracijos kriterijų, tai jis yra leistinas, nepriklausomai nuo to, ar jam priskirtas tabu.

Sustojimo kriterijus tabu paieškos procedūrą užbaigia arba po nustatyto iteracijų skaičiaus, arba radus geriausią sprendinį.



1.2 pav. Standartinio tabu paieškos algoritmo struktūrinė diagrama
(pagal Pham D.T., Karaboga D., 2000)

Aspiracijos kriterijus ir tabu apribojimai. Jei tabu sprendinys yra nepakankamai geras, bet gali apsaugoti nuo ciklo susidarymo, šio sprendinio tabu panaikinimui naudojamas aspiracijos kriterijus. Aspiracijos kriterijus valdo paieškos procesą, o tabu apribojimai sumažina paieškos erdvę. Sprendinys yra priimtinas, jei patenkinti tabu apribojimai. Tačiau nepriklausomai nuo tabu būsenos, sprendinys taip pat laikomas leistinu, jeigu taikomas aspiracijos kriterijus. Žingsnio atributai tabu paieškoje įrašomi ir naudojami apribojimų įvedimui. Šie apribojimai neleidžia atlikti žingsnių, kurie panaikintų atributų sąlygotus pasikeitimus. Tabu apribojimai taikomi norint išvengti pasikartojimų paieškos kelyje, o ne visiškų pasikeitimų. Apribojimai apsaugantys nuo visiškų pasikeitimų skirti siekiant išvengti grįžimo prie ankstesnio sprendinio. Dažniausiai tabu apribojimai naudojami tais atvejais, kai jų atributai pasikartoja per apibrėžtą iteracijų skaičių (naujumu grindžiamas apribojimas) arba

pasikartoja tam tikru dažniu, esant didesniam iteracijų skaičiui (dažniu grindžiamas apribojimas).

Naujumu grindžiamame apribojime visam paieškos laikui nustatoma tabu trukmė ir išsaugomas tabu sprendinys. Taisyklės, pagal kurias nustatoma tabu trukmė, skiriamos į statines ir dinamines. Statinės taisyklės parinkta tabu trukmės reikšmė išlieka pastovi visos paieškos metu. Dinaminės taisyklės leidžia šios reikšmės pokyčius.

Dažniu grindžiamame apribojime naudojamas *dažnio matas*. Pasak Reeves C.R. (1995) šis matas yra santykis, kurio skaitiklis reiškia tam tikro įvykio pasikartojimų skaičių, o vardiklis – vieną iš šių dydžių: skaitiklių sumą, didžiausią skaitiklio reikšmę, vidutinę skaitiklio reikšmę.

Tinkamas aspiracijos kriterijaus pritaikymas tabu paieškoje gali būti labai reikšmingas siekiant gero sprendinio. Aspiracijos kriterijus gali priklausyti arba nepriklausyti nuo laiko. Pradiniams tabu paieškos taikymams buvo naudojamas tik paprasto tipo, nepriklausantis nuo laiko, aspiracijos kriterijus. Šio kriterijaus esmė – tabu klasifikacijos pašalinimas iš bandomojo sprendinio, kuomet sprendinys įgyja geresnes charakteristikas lyginant su iki tol buvusiu geriausiu sprendiniu. Šis kriterijus vis dar plačiai taikomas. Kitas dažnai taikomas aspiracijos kriterijus – *numatytoji aspiracija*. Naudojant šį kriterijų, jei visi galimi žingsniai pažymėti kaip tabu ir kitų aspiracijos kriterijų pagalba negalima gauti leistino sprendinio, tai pasirenkamas mažiausios tabu reikšmės sprendinys. Reeves C.R. (1995) išskiria daugiau aspiracijos kriterijų: aspiracija pagal tikslą, aspiracija pagal paieškos kryptį ir aspiracija pagal įtaką.

1.3. Genetiniai algoritmai

1.3.1 Genetinių algoritmų samprata

Holland J. H. (1975) siūlo genetinių algoritmų idėją. Šie algoritmai yra efektyvūs ir plačiai taikomi stochastinės paieškos ir optimizavimo metodai, pagrįsti evoliucijos teorijos principais. Per pastaruosius kelerius metus genetinių algoritmų bendruomenė daug dėmesio skyrė pramonės inžinerijos optimizavimo problemų sprendimui. Genetinis algoritmas yra atsitiktinės paieškos metodas, įgalinantis sudėtinėse daugiadimensinėse paieškos erdvėse rasti globalų optimalų sprendinį. Genetiniai algoritmai sukurti pagal evoliucijos principus, o juose naudojamos funkcijos „pasiskolintos“ iš evoliucijoje vykstančių procesų. Šios genetinės funkcijos kiekvienos generacijos metu paveikia populiacijos individus, siekdamos palaipsniui gerinti jos tinkamumą.

Genetiniai algoritmai, skirtingai nuo įprastinių paieškos metodų, pradedami pradinės atsitiktinių sprendinių aibės, *populiacijos*, sudarymu. Kiekvienas populiacijos individas vadinamas *chromosoma*. Chromosoma – tai simbolių eilutė, dažniausiai tai dvejetainių bitų eilutė.

Holland J. H. (1975) formuluoja „schemos teoremą“, kuri gali valdyti individų populiacijos evoliuciją. Schema vaizduoja aibę individų, t. y. populiacijos poaibį, šių individų bitų panašumo tam tikrose pozicijose aspektu. Pavyzdžiui, schema 1*0* aprašo aibę individų, kurių pirmas ir trečias bitai atitinkamai yra 1 ir 0. Čia simbolis * reiškia, kad šioje pozicijoje gali būti reikšmė 0 arba 1. Schema charakterizuojama dviem parametrais: ilgiu ir būkle. Ilgis – tai atstumas tarp pirmo ir paskutinio bitų su fiksuotomis reikšmėmis. Schemos būklė – tai bitų skaičius su apibrėžtomis reikšmėmis. „Schemos teorema“ teigia, kad schemos išsidėstymas populiacijoje keičiantis kartoms, priklauso nuo šios schemos būklės, ilgio ir tinkamumo.

Chromosomos evoliucionuoja vykdant nuoseklias iteracijas, kurios vadinamos generacijomis. Kiekvienos generacijos metu chromosomos įvertinamos, naudojant tam tikrą tinkamumo matą. Kitos kartos sukūrimui, naujos chromosomos, vadinamos palikuonimis, suformuojamos dviem būdais:

1. sujungiant dvi chromosomas iš dabartinės kartos, panaudojus kryžminimo (*crossover*) operaciją;
2. modifikuojant chromosomą, panaudojus mutacijos (*mutation*) operaciją.

Norint išlaikanti pastovų populiacijos dydį, nauja karta suformuojama, pagal tinkamumo reikšmę parenkant tam tikrus tėvus bei palikuonis, ir atmetant likusius. Didesnė tikimybė, kad bus parinktos pajėgesnės chromosomos. Atlikus kelias generacijas, algoritmas konverguoja į geriausią chromosomą, kuri tikriausiai reikš optimalų sprendinį. Tegu *P(t)* ir *C(t)* žymi dabartinės generacijos *t* tėvus ir palikuonis (žr. 1.3 pav.).

```
begin
        t ← 0;
        inicializuoti P(t);
        įvertinti P(t);
        while (ne pabaigos sąlyga) do
                sukeisti P(t), kad gauti C(t);
                įvertinti C(t);
                išrinkti P(t+1) iš P(t) ir C(t);
                t ← t + 1;
        end
end
```

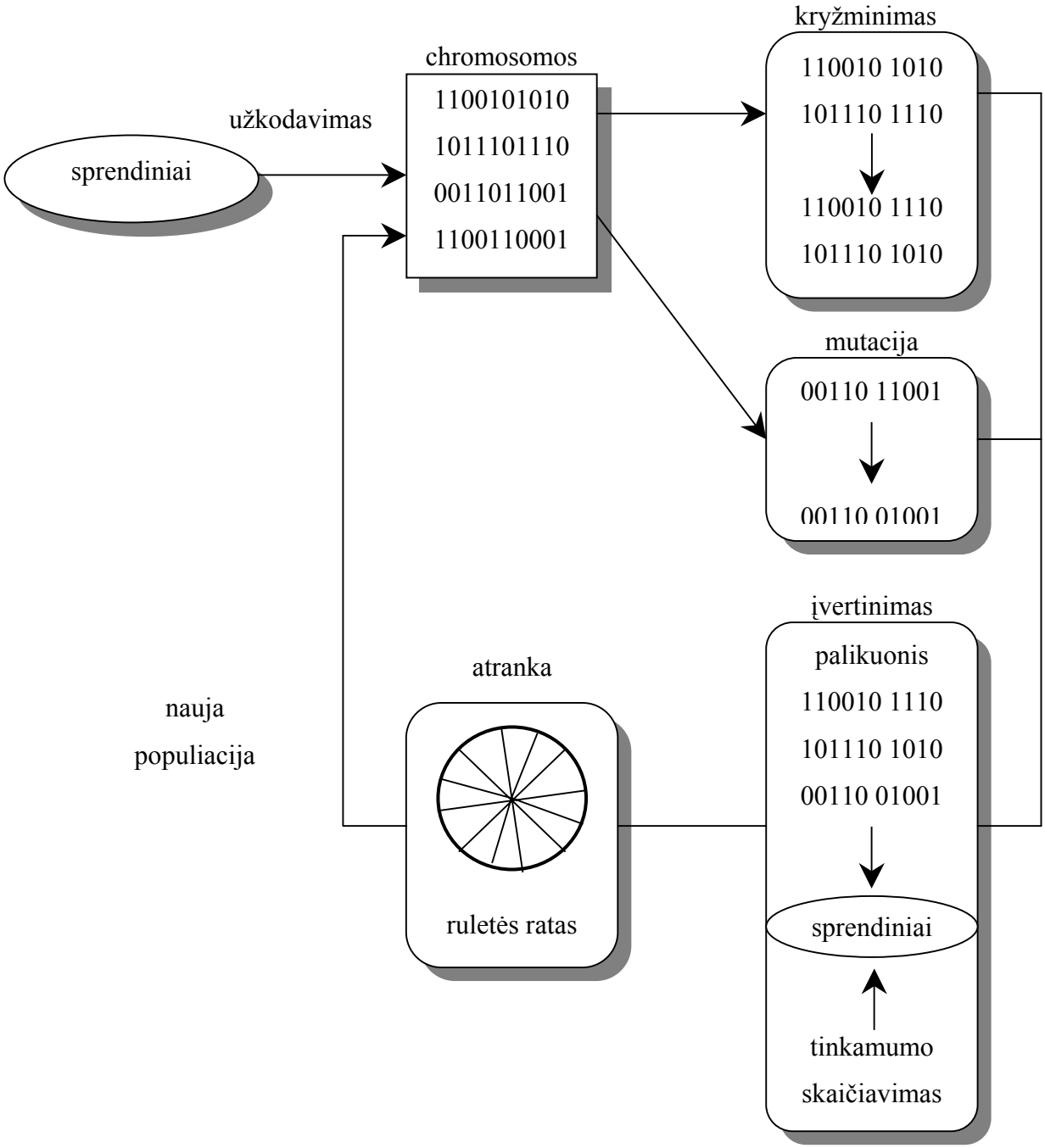
1.3 pav. Genetinių algoritmų procedūra

(pagal Mitsuo G., Runwei C., 1997)

Dažniausiai, pradinės reikšmės priskiriamos (*initialization*) atsitiktinai. Sukeitimo veiksmas siekiant gauti palikuonį, dažniausiai apima kryžminimo ir mutacijos operacijas. Genetiniuose algoritmuose atliekamos dviejų tipų funkcijos:

1. genetinės operacijos – kryžminimas ir mutacija;
2. įvertinimo operacija – atranka.

Apibendrinta genetinių algoritmų procedūra pateikta 1.4 pav.



1.4 pav. Apibendrinta genetinių algoritmų struktūra

(pagal Mitsuo G., Runwei C., 1997)

Genetinės funkcijos imituoja genų paveldėjimo procesą, idant kiekvienos generacijos metu būtų sukurtas naujas palikuonis. Įvertinimo operacija imituoja Darvino evoliucijos procesą, kai populiaciją kuriama iš kartos į kartą.

Kryžminimas yra pagrindinė genetinė funkcija. Ji vienu metu naudoja dvi chromosomas ir generuoja palikuonį, jungdama abiejų chromosomų savybes. Kryžminant paprastuoju būdu, pasirenkamas atsitiktinis atkirtimo taškas ir nuo šio taško jungiant vieno tėvo kairįjį segmentą su kito tėvo dešiniuoju segmentu, generuojamas palikuonis. Genetinių algoritmų efektyvumas labai priklauso nuo naudojamos kryžminimo funkcijos našumo.

Mutacija yra antraeilė funkcija, lemianti atsitiktinius pasikeitimus skirtingose chromosomose. Atliekant mutaciją paprastuoju būdu, reikia pakeisti vieną ar daugiau genų. Genetiniuose algoritmuose mutacija labai svarbi dėl šių priežasčių:

- ji pakeičia išrinkimo proceso metu prarastus populiacijos genus, todėl jie gali būti bandomi naujose situacijose;
- aprūpina pradinėje populiacijoje nebuvusiais genais.

Genetiniai algoritmai skiriasi nuo įprastų optimizavimo ir paieškos procedūrų keletu pagrindinių veiksnių. Goldberg D. E. (1989) šiuos skirtumus apibūdina taip:

1. Genetiniai algoritmai naudoja koduotą sprendinių aibę, o ne pačius sprendinius.
2. Genetiniai algoritmai paiešką vykdo iš sprendinių populiacijos, o ne iš vieno sprendinio.
3. Genetiniai algoritmai naudoja tam tikrą informaciją (tinkamumo funkcija), o ne išvestines ar pagalbines žinias.
4. Genetiniai algoritmai naudoja galimas, o ne nustatytas perėjimo taisykles.

Kaip rašoma anksčiau, genetiniai algoritmai nenaudoja daug informacijos apie optimizuojamą problemą ir tiesiogiai nesusiję su sprendžiamos problemos parametrais. Parametrų vaizdavimui šie algoritmai naudoja kodus. Pham D.T., Karaboga D. (2000) formuluoja keturis klausimus, iškylančius taikant genetinius algoritmus. *Pirmasis klausimas* taikant genetinius algoritmus – kaip užkoduoti nagrinėjamą problemą, t. y., kaip atvaizduoti problemos parametrus? Genetiniai algoritmai naudoja galimų sprendinių populiaciją, o ne vieną galimą sprendinį, todėl *antras klausimas* – kaip sukurti pradinę galimų sprendinių populiaciją? *Trečias klausimas* taikant genetinius algoritmus – kaip išrinkti ar surasti tinkamų genetinių funkcijų aibę? Pagaliau, kaip ir kiti paieškos algoritmai, genetiniai algoritmai turi mokėti nustatyti jau surastų sprendinių kokybę, idant vėliau juos būtų galima pagerinti. Todėl reikalinga nagrinėjamos problemos aplinkos ir paties genetinio algoritmo sąsaja. *Ketvirtasis klausimas* apima šios sąsajos sudarymą.

1.3.2 Atvaizdavimo metodai

Optimizuojami parametrai dažniausiai atvaizduojami eilutės forma, nes šio tipo atvaizdavimui tinka genetinės funkcijos. Atvaizdavimo metodai stipriai įtakoja genetinių algoritmų efektyvumą. Skirtingos atvaizdavimo schemos gali sąlygoti skirtingą genetinių algoritmų tikslumą ir skaičiavimų laiką.

Pasak Michalewicz Z. (1992), Davis L. (1991), sprendžiant skaitmenines optimizavimo problemas, dažniausiai naudojami du atvaizdavimo metodai. Populiariausias atvaizdavimo metodas yra *dvejetainė eilutė*. Literatūroje pateikiama įvairių dvejetainių kodavimo schemų: nekintamas (*Uniform*) kodavimas, Grėjaus (*Gray*) kodavimas ir kt. Antrasis atvaizdavimo metodas naudoja sveikųjų ar realiųjų skaičių vektorių, kur kiekvienas sveikasis ar realusis skaičius atvaizduoja vieną parametrą.

Taikant dvejetainę atvaizdavimo schemą, svarbu nuspręsti, kiek bitų reikės užkoduoti optimizuojamiems parametrams. Kiekvienas parametras turi būti užkoduotas optimaliu bitų skaičiumi, apimant visus galimus sprendinius sprendinių erdvėje. Jei naudojama per mažai ar per daug bitų, gali nukentėti skaičiavimų efektyvumas.

1.3.3 Pradinio sprendinio kūrimas

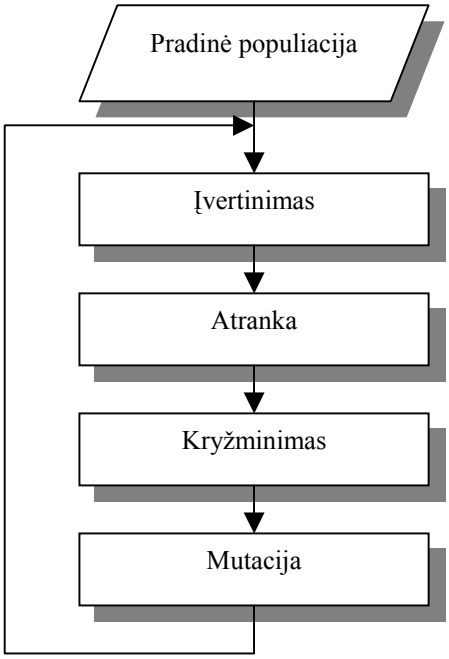
Optimizacijos pradžioje genetiniams algoritmams reikia pradinių sprendinių grupės (populiacijos). Šią pradinę populiaciją galima suformuoti dviem būdais. Pirmasis būdas – naudoti atsitiktinių skaičių generatoriaus sukurtus atsitiktinius sprendinius. Šis metodas dažniau taikomas neturint pradinės informacijos arba įvertinant algoritmo efektyvumą.

Antrasis metodas populiacijos formavimui taiko pradinę informaciją apie sprendžiamą optimizavimo problemą. Naudojant šią informaciją, nustatomi reikalavimai ir parenkami šiuos reikalavimus tenkinantys sprendiniai. Šiuo atveju genetiniai algoritmai optimizavimo procesą pradeda apytikriai žinomų sprendinių aibe, todėl prie optimalaus sprendinio konverguoja greičiau, nei naudojant anksčiau aptartą metodą.

1.3.4 Genetinės funkcijos

1.5 pav. pateikta supaprastinto genetinio algoritmo struktūrinė diagrama. Be anksčiau minėtų pagrindinių genetinių funkcijų (atrankos, kryžminimo ir mutacijos), kartais naudojama papildoma funkcija – inversija. Kai kurios iš šių funkcijų analogiškos gamtoje vykstantiems procesams. Genetiniuose algoritmuose nebūtina naudoti visų šių funkcijų, nes kiekviena iš jų

veikia nepriklausomai nuo kitų. Funkcijų pasirinkimas ar projektavimas priklauso nuo nagrinėjamos problemos ir naudojamos atvaizdavimo schemos. Pavyzdžiui, suprojektuotos dvejetainei eilutei funkcijos negali būti tiesiogiai naudojamos eilutėms, užkoduotoms sveikais ar dvejetainiais skaičiais.



1.5 pav. Supaprastinto genetinio algoritmo struktūrinė diagrama
(pagal Pham D.T., Karaboga D., 2000)

Atranka. Atrankos procedūros tikslas – atkurti kuo daugiau individų kopijų, kurių tinkamumo reikšmė didesnė už individų, turinčių žemą tinkamumo reikšmę. Atrankos procedūra stipriai įtakoja gero sprendinio paiešką per trumpą laiką. Tačiau populiacijos įvairovė turi būti kontroliuojama, siekiant išvengti pirmalaikės konvergencijos ir rasti globalaus optimumo sprendinį. Whitely D., Hanson T. (1989) genetiniuose algoritmuose išskiria dvi pagrindines atrankos procedūras: proporcinę atranką ir rangais pagrįstą atranką.

Proporcinė atranka dažniausiai vadinama „ruletės rato“ atranka, nes jo mechanizmas primena ruletės rato veikimą. Individo tinkamumo reikšmė reiškia rato griovelio plotį. Norint išrinkti naujos kartos individą, atsitiktinai sukamas ratas. Didesnė tikimybė, kad bus išrinkti individai, esantys platesniuose grioveliuose, kurie žymi didesnę tinkamumo reikšmę.

Vienas iš būdų, norint išvengti greito konvergavimo – apriboti kiekvienam individui priskirtą bandymų diapazoną. Šitaip nei vienas individas negeneruos per daug palikuonių. Pasak Baker J. E. (1985), taikant rangais pagrįstą atrankos procedūrą, kiekvienas individas generuoja laukiamą palikuonių skaičių, kuris priklauso nuo ne nuo jo dydžio, o nuo jo tinkamumo reikšmės rango.

Kryžminimas. Ši funkcija skiria genetinius algoritmus nuo kitų, pavyzdžiui, dinaminio programavimo algoritmo. Ji naudojama kuriant du naujus individus (vaikus) iš jau egzistuojančių individų (tėvų), kurie parenkami iš turimos populiacijos naudojant atrankos operaciją. Yra keletas būdų, kaip galima tai atlikti: vieno taško kryžminimas, dviejų taškų kryžminimas, ciklinis kryžminimas ir pastovus kryžminimas.

Vieno taško kryžminimas yra pati paprasčiausia kryžminimo funkcija. Iš individų aibės, kurią suformuoja atrankos procedūra, atitiktinai paimami du individai kaip tėvai ir padalinami atsitiktiniame taške. Gautos dalys sukeičiamos vietomis ir šitaip išvedami du nauji individai (vaikai). Šios operacijos metu bitų reikšmės nekeičiamos (žr. 1.6 pav.).

Tėvas 1	**1 0 0 0 1 0 0 1 1 1 1**
Tėvas 2	0 1 1 0 1 1 0 0 0 1 1
Nauja eilutė 1	**1 0 0 0 1** 1 0 0 0 1 1
Nauja eilutė 2	0 1 1 0 1 **0 0 1 1 1**

1.6 pav. Vieno taško kryžminimo veiksmas

Ciklinio kryžminimo pavyzdys pateiktas 1.7 pav.

Tėvas 1	1 2 3 4 5 6 7 8
Tėvas 2	**a b c d e f g h**
Nauja eilutė 1	1 **b** 3 **d e** 6 **g** 8
Nauja eilutė 2	**a** 2 **c** 4 5 **f** 7 **h**

1.7 pav. Ciklinio kryžminimo veiksmas

Mutacija. Šioje procedūroje visi populiacijos individai tikrinami nuosekliai, o bitų reikšmės atsitiktinai sukeičiamos pagal apibrėžtą rodiklį (žr. 1.8 pav.). Kitaip nei kryžminimas, ši operacija yra „vienaląstė“, t. y. vaiko eilutė sudaroma iš vienos tėvo eilutės. Mutacijos funkcija skatina algoritmą vykdyti paieškos veiksmus naujose srityse. Galiausiai, ji padeda genetiniams algoritmams išvengti pirmalaikio konvergavimo ir padeda surasti globalaus optimumo sprendinį.

Sena eilutė	1 1 0 0 **0** 1 0 1 1 1 0
Nauja eilutė	1 1 0 0 **1** 1 0 1 1 1 0

1.8 pav. Mutacijos veiksmas

Inversija. Ši papildoma funkcija vienu metu taikoma vienam individui. Atsitiktinai parenkami du individo taškai ir tarp šių taškų sukeičiamos eilutės dalys (žr. 1.9 pav.).

Sena eilutė	1 0 **1 1 0 0** 1 1 1 0
Nauja eilutė	1 0 **0 0 1 1** 1 1 1 0

1.9 pav. Dvejetainės eilutės segmento inversija

1.3.5 Valdymo parametrai

Svarbūs genetinių algoritmų valdymo parametrai yra populiacijos dydis (individų skaičius populiacijoje), kryžminimo koeficientas ir mutacijos koeficientas. Schaffer J. D., Caruana R. A. ir kt. (1989), Grefenstette J. J. (1986), Fogarty T. C. (1989) nagrinėja šių parametrų įtaką genetinių algoritmų našumui. Jei populiacija yra didelė, vienu metu apdorojama daugiau sprendinių, bet išauga vienos iteracijos skaičiavimų laiko sąnaudos. Tačiau konvergavimas prie optimalaus sprendinio didesnis, nes naudojama daugiau pavyzdžių iš ieškomos erdvės.

Mitsuo G., Runwei C. (1997) kryžminimo koeficientą p_c apibrėžia kaip santykį kiekvienoje kartoje sukurtų palikuonių skaičiaus su populiacijos dydžiu, kuris paprastai žymimas pop_size. Šis santykis sąlygoja iš kryžminimo operacijos laukiamų chromosomų skaičių $p_c \times$ pop_size. Didesnis kryžminimo koeficientas leidžia paiešką platesnėje sprendinių erdvėje ir sumažina neteisingo optimumo nustatymo tikimybę. Jei šis koeficientas per didelis, daug kompiuterio skaičiavimo laiko iššvaistoma nagrinėjant neefektyvias sprendinių erdves.

Mutacijos koeficientas p_m apibrėžiamas kaip visų genų skaičiaus populiacijoje procentinė dalis. Mutacijos koeficientas nustato lygį, kuriam esant į populiacija įvedami nauji genai. Jei šis koeficientas per mažas, dauguma genų, kurie galėtų būti naudingi, niekada neišbandomi. Esant per didelei koeficiento reikšmei, atsiras daug atsitiktinių trikdymų, dėl

kurių palikuonys gali prarasti savo panašumą į tėvus. Šitaip algoritmas praras galimybę mokytis iš paieškos praeities.

1.3.6 Tinkamumo įvertinimo funkcija

Tinkamumo įvertinimo funkcija sieja genetinius algoritmus ir optimizacijos problemą. Genetiniai algoritmai sprendinių kokybę nustato pagal tinkamumo įvertinimo funkcijos gautą informaciją, nenaudodami tiesioginės informacijos apie jų struktūrą. Inžineriniuose projektavimuose projektuotojui specifikuojami funkciniai reikalavimai. Projektuotojas turi sudaryti struktūrą, kuri realizuotų norimas funkcijas pagal iš anksto nustatytus apribojimus. Dažniausiai pasiūlyto sprendinio kokybė nustatoma pagal tai, kaip sprendinys atlieka norimas funkcijas, bei atitinka nustatytus apribojimus. Genetinių algoritmų atveju šie skaičiavimai turėtų būti vykdomi automatiškai, čia problema iškyla sudarant procedūrą, kuri įvertintų sprendinio kokybę.

Tinkamumo įvertinimo funkcija, priklausomai nuo nagrinėjamos optimizavimo problemos, gali būti sudėtinga arba paprasta. Kai šio uždavinio sprendimui negalima suformuluoti matematinės lygties, tada gali būti naudojama taisyklėmis pagrįsta procedūra; arba abu šie būdai gali būti derinami. Kai tam tikri apribojimai yra labai svarbūs ir negali būti pažeisti, tuomet suprojektuojama atvaizdavimo schema, kurios pagalba iš anksto panaikinami tie sprendiniai, kurie sąlygoja šiuos pažeidimus. Taip pat jiems gali būti priskirtos mažos tikimybės, naudojant specialias bausmių funkcijas.

1.4. Atkaitinimo algoritmas

Atkaitinimo algoritmo idėja kilo iš statistinės mechanikos. Kirkpatrick S., Gelatt C. D. Jr., Vecchi M.P. (1983) siūlo algoritmą, pagrįstą uolienos atkaitinimo ir kombinatorinių optimizavimo problemų sprendimo analogija.

Atkaitinimas yra fizikos procesas, kurio metu uoliena įkaitinama ir po to lėtai atvėsinama iki kristalizacijos. Esant aukštesnei temperatūrai medžiagos atomų energija didesnė ir atomai laisviau juda. Sumažėjus temperatūrai, susilpnėja ir atomų energija. Kai sistemos energijos lygis mažiausias, gaunamas taisyklingos formos kristalas. Jei atvėsinimo procesas vykdomas labai greitai, kristalo struktūroje išryškėja nukrypimai ir defektai. Sistema nepasiekia minimalaus energijos lygio, o išlieka daugiakristalinėje būsenoje, kuri turi daugiau energijos.

Esant nustatytai temperatūrai sistemos energijos pasiskirstymo tikimybė yra:

$$P(E) = e^{[-E/(kT)]},$$

kur E – sistemos energija, k – Bolcmano konstanta, T – temperatūra ir P(E) –tikimybė, kad sistema yra būsenoje, esant energijos lygiui E.

Iš aukščiau pateiktos formulės matyti, jog prie aukštos temperatūros, esant bet kokiam energijos lygiui, P(E) artėja prie 1. Taip pat yra maža tikimybė, kad sistema gali turėti daug energijos net ir esant žemai temperatūrai. Todėl statistinis energijos pasiskirstymas leidžia sistemai išvengti lokalaus energijos minimumo.

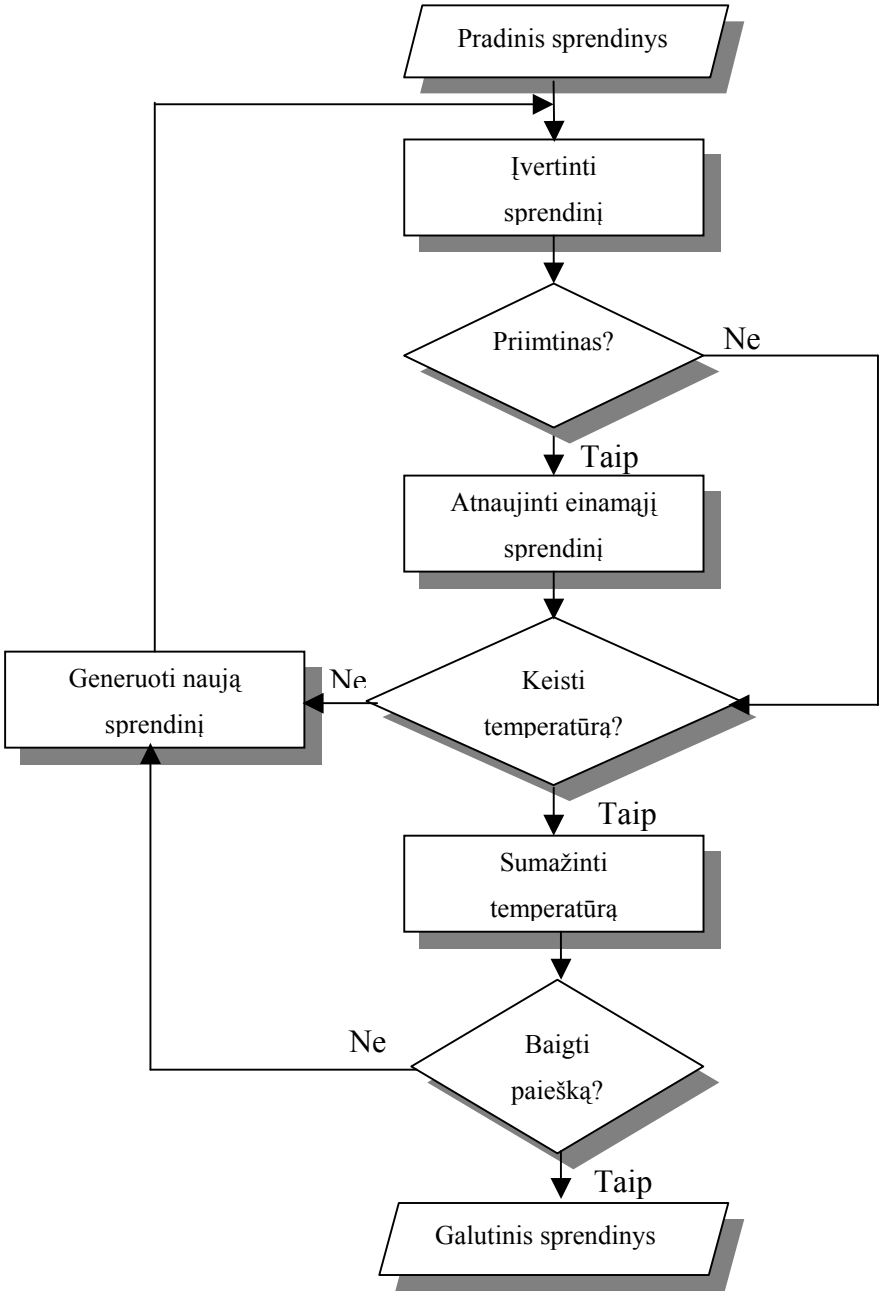
Iš kombinatorinių optimizavimo problemų ir atkaitinimo proceso analogijos, uolienos būsena išreiškia galimus optimizavimo problemos sprendinius; būsenos energija atitinka tikslo funkcijos įvertinimus, suskaičiuotus šiems sprendiniams; minimali būsenos energija atitinka optimalų sprendinį, o greitas atvėsinimas gali būti suprantamas kaip lokalus optimizavimas.

Algoritmas sudarytas iš iteracijų sekos. Naujo sprendinio radimui einamojo sprendinio aplinkoje, kiekvienoje iteracijoje atsitiktinai pakeičiamas einamasis sprendinys. Aplinka apibrėžiama parenkant generavimo mechanizmą. Sukūrus naują sprendinį, skaičiuojamas atitinkamos kaštų funkcijos pokytis, siekiant nuspręsti, ar naujai gautas sprendinys gali būti laikomas einamuoju sprendiniu. Jei kaštų funkcijos pokytis yra neigiamas, tai naujai gautas sprendinys išrenkamas einamuoju. Priešingu atveju, sprendinys parenkamas pagal Metropolis N. ir kt. (1953) pasiūlytą kriterijų, pagrįstą Bolcmano tikimybe. Pagal šį kriterijų, jei skirtumas tarp einamojo ir naujai gauto sprendinio kaštų funkcijos reikšmių didesnis arba lygus nuliui, iš intervalo [0,1] pagal nekintamą pasiskirstymą generuojamas atsitiktinis skaičius δ, ir jei tenkinama sąlyga

$$\delta \le e^{(-\Delta E/T)},$$

tai naujai gautas sprendinys išrenkamas einamuoju. Priešingu atveju einamasis sprendinys nekeičiamas. ΔE yra skirtumas tarp dviejų sprendinių kaštų funkcijos reikšmių.

Standartinio atkaitinimo algoritmo struktūrinė diagrama pateikta 1.10 pav.



1.10 pav. Standartinio atkaitinimo algoritmo struktūrinė diagrama
(pagal Pham D.T., Karaboga D., 2000)

Pham D.T., Karaboga D. (2000) skiria keturis svarbiausius atkaitinimo algoritmo realizavimo etapus:

- sprendinių vaizdavimas;
- kaštų funkcijos nustatymas;
- kaimyninių reikšmių generavimo mechanizmo nustatymas;
- atvėsinimo plano sukūrimas.

Atkaitinimo algoritmuose sprendinių vaizdavimas ir kaštų funkcijos nustatymas yra tokie pat kaip ir genetiniuose algoritmuose. Generavimo mechanizmai gali būti sukurti nauji arba paimti iš genetinių algoritmų. Genetiniuose algoritmuose šie mechanizmai gali būti

mutacija ir perstatymas. Atkaitinimo algoritme sudarant atvėsinimo planą turi būti apibrėžti keturi parametrai:

- pradinė temperatūra;
- temperatūros atnaujinimo taisyklė;
- kiekviename temperatūros lygyje vykdomų iteracijų skaičius;
- paieškos pabaigos kriterijus.

Pasak Osman I. H. (1991), galima sudaryti keletą atvėsinimo planų, kurie naudotų skirtingas temperatūros atnaujinimo schemas. Praktikoje dažnai taikomos nepertraukiamos, nemonotoninės ir nuo žingsnio priklausančios temperatūros mažinimo schemos. Nuo žingsnio priklausančios mažinimo schemos naudoja labai paprastas atvėsinimo taisykles. Pavyzdžiui, temperatūra galėtų būti perskaičiuojama pagal tokią formulę:

$T_{i+1} = cT_i,\ \mathrm{i} = 0,1...,$

kur c – temperatūros daugiklis, kurio reikšmė mažesnė už 1, bet labai artima 1.

1.5. Kiti optimizavimo metodai

GRASP sutrumpinimą pasiūlė Feo T., Resende M. (1995). GRASP sistemose euristika kartojama keletą kartų. Kiekvienos iteracijos metu gaunamas „godus“ atsitiktinis sprendinys, kurio aplinkoje ieškoma lokalaus optimumo. „Godus“ komponentas formuoja sprendinį vienu metu pridedant po vieną elementą. Sprendinio formavimui jis turi atlikti šiuos žingsnius:

- visiems neišrinktiems kandidatams panaudoti godumo funkciją;
- surūšiuoti šiuos kandidatus pagal godumo funkciją;
- išrinkti poaibį gerų (bet ne geriausių) kandidatų ir įtraukti į apribotų kandidatų sąrašą;
- atsitiktinai išrinkti kandidatą iš sąrašo ir įtraukti į sprendinį;
- pakeisti godumo funkciją, atsižvelgiant į kandidato įtraukimą į sprendinį.

Šie žingsniai kartojami, kol gaunamas sprendinys.

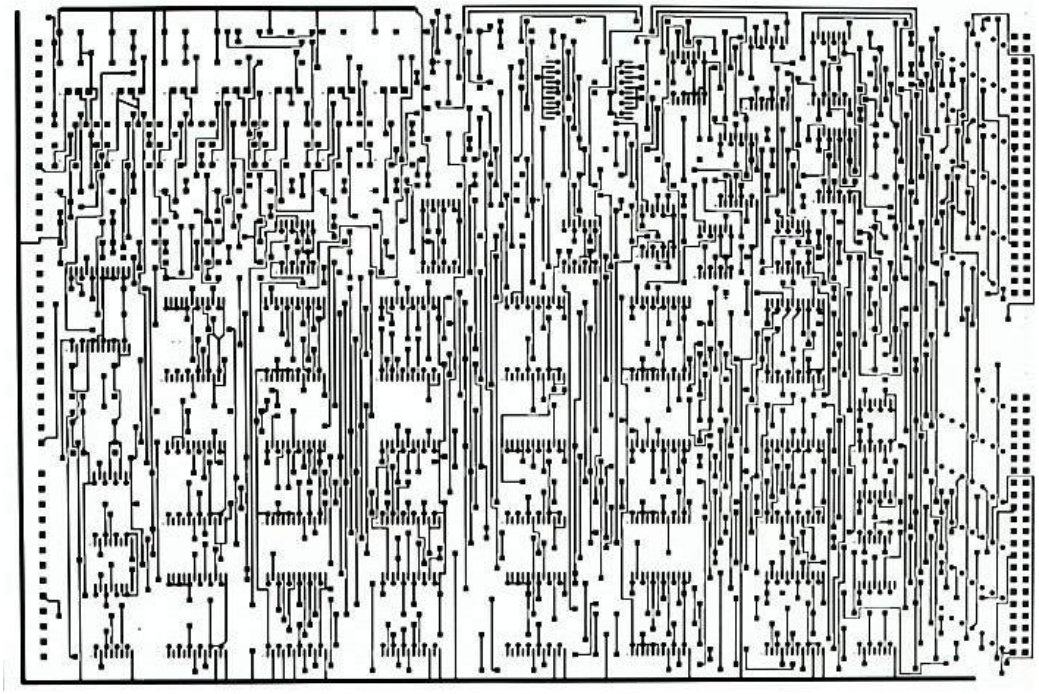
Dauguma autorių, kurie naudoja atsitiktines euristikas, parametrus šioms euristikoms pritaiko intuityviai kaip tyrimų proceso rezultatą. Toks pritaikymas gali būti suprantamas kaip optimizavimo problema. Pasak Mockus J., Eddy W., Reklaitis G. ir kt. (1997) šie parametrai gali būti optimizuojami visais tinkamas metodais, vienas iš jų galėtų būti Bayeso (*Bayesian*) euristinis metodas. Pavyzdžiui, tabu paieškoje taikant Bayeso metodą, galima optimizuoti du svarbius parametrus: geriausią ilgalaikės ir trumpalaikės atminties kombinaciją, geriausią pusiausvyrą tarp stiprėjimo ir įvairinimo strategijų. GRASP sistemose Bayeso euristiniai

metodai gali būti naudojami optimizuojant atsitiktinį kandidato parinkimą. Šie metodai taip pat gali būti taikomi genetinių algoritmų parametrams optimizuoti. Mockus J. (2000) teigia, kad Bayeso euristinio metodo kokybė priklauso nuo konkrečios euristikos, o pačių euristikų kokybė priklauso nuo turimų žinių. Taigi, Bayeso euristinis metodas skirtas euristikų gerinimui, bet ne jų pakeitimui.

2. PROJEKTINĖ DALIS

Spausdinto montažo plokštė, plokštė baldų gamybai, medžiaga rūbams siūti, stogo danga gali būti ruošinio pavyzdžiu. Ruošinio realizacija – tai jau turimas ruošinio projektas. Pavyzdžiui, spausdinto montažo plokštės atveju žinoma, kaip išdėstyti takeliai, reikia su mažiausiom laiko sąnaudomis nubrėžti takelius. Tikslas būtų optimizuoti įrenginio darbo procesą. Spausdinto montažo plokštės atveju, įrenginys galėtų būti braižiklis, tuomet optimizuoti reiktų braižymo procesą. Optimizuojant šį procesą reiktų įvertinti tiek ruošinių technologinio realizavimo ypatumus, pavyzdžiui, spausdinto montažo plokštės takelių storį (takelis gali būti brėžiamas įvairaus storio plunksnomis), tiek braižiklio darbo parametrus (darbo greitį brėžiant liniją ir judant tuščiąja eiga).

Projektavimo eigoje naudojamos grafų teorijos sąvokas. Tuomet, ruošinio realizacija – tai grafo briaunų ir viršūnių aibė. Pavyzdžiui, jei ruošinio realizacija imtume spausdinto montažo plokštės projektą, tai šios plokštės takelius galima interpretuoti kaip plokščiųjų grafų aibę. Spausdinto montažo plokštės pavyzdys pateiktas 2.1 pav.



2.1 pav. Spausdinto montažo plokštė

Plokščiojo grafo briaunos kertasi tik viršūnėse. Visos elektrinės schemos grafas nėra jungusis, t. y. ne visos viršūnių poros sujungtos grandine. Suradus maršrutą (nebūtinai optimalų), gausime grafą, kuris yra jungusis.

Plukas K., Mačikėnas E. ir kt. (2001) išskiria įvairius maršrutai, pavyzdžiui, Oilerio maršrutas, Hamiltono maršrutas, kurie taikomi ne tik sprendžiant įvairius galvosūkius, bet ir praktiniuose uždaviniuose. Optimalaus maršruto paieškai galima naudoti įvairius optimizavimo metodus.

Darbo uždavinį galima suformuluoti taip: aplankyti visas grafo briaunas, kad maršrutas būtų kiek galima trumpesnis. Tačiau maršruto ilgio optimizavimas neužtikrina, kad ruošinio realizacijos proceso trukmė bus optimali. Todėl, kaip minėta anksčiau, reikia įvertinti papildomas sąlygas, t. y. ruošinių technologinio realizavimo ypatumus ir įrenginio darbo parametrus. Taigi, uždavinį reikia suskirstyti į dvi dalis:

- optimizuoti maršruto ilgį;
- optimizuoti proceso trukmę.

Išsprendus šiuos uždavinius, galima palyginti jų rezultatus ir priklausomai nuo konkrečios nagrinėjamos problemos, pasirinkti tinkamiausią sprendinį.

Ruošinių realizacijos optimizavimo problemos spręsti galima pritaikyti patį paprasčiausią sprendimo būdą – *perrinkimo algoritmą*, t. y. sudaryti visus įmanomus maršrutus, paskaičiuoti jų ilgį ar proceso trukmę ir išrinkti optimalų. Tačiau, šio algoritmo sudėtingumas bus faktorialinis. Šį algoritmą galima taikyti tik nedidelių duomenų atveju. Nors perrinkimo algoritmas randa optimalų maršrutą, tačiau yra pats neefektyviausias skaičiavimų trukmės atžvilgiu.

Kaip minėta ankstesniame skyriuje, gali būti naudojami algoritmai, kurie neranda optimalaus sprendinio, bet jų rezultatas – pakankamai optimalus sprendinys, gaunamas per polinominį laiką. Apytiksliai algoritmai duoda sprendinį, kuris yra pakankamai arti optimalaus sprendinio. Apytikslių algoritmų atveju sprendinio paieškos laiką gana tiksliai galima numatyti iš anksto.

Taip pat gali būti naudojami euristiniai algoritmai. Jie pateikia nebūtinai patį trumpiausią maršrutą, o tikėtinai trumpiausią. Tačiau euristiniai algoritmai yra efektyvūs laiko prasme. Lyginant su apytiksliais algoritmais, iš anksto nusakyti euristinių algoritmų sprendinio paieškos trukmę yra sudėtingiau.

Labai dažnai naudojami taip vadinami *godūs euristiniai algoritmai.* Tokio algoritmo pavyzdys yra *artimiausio kaimyno algoritma*s. Euristinių algoritmų klasei priklauso *tabu paieškos* algoritmas. Tabu paieškos algoritmas buvo pasirinktas kaip tinkamiausias ruošinių realizacijos optimizavimo problemoms spręsti. Šių problemų sprendimui taip pat pasiūlytas algoritmas, kuris įvertina ruošinių realizacijos proceso ypatumus.

2.1. Pilno perrinkimo algoritmas ruošinių realizacijos optimizavimui

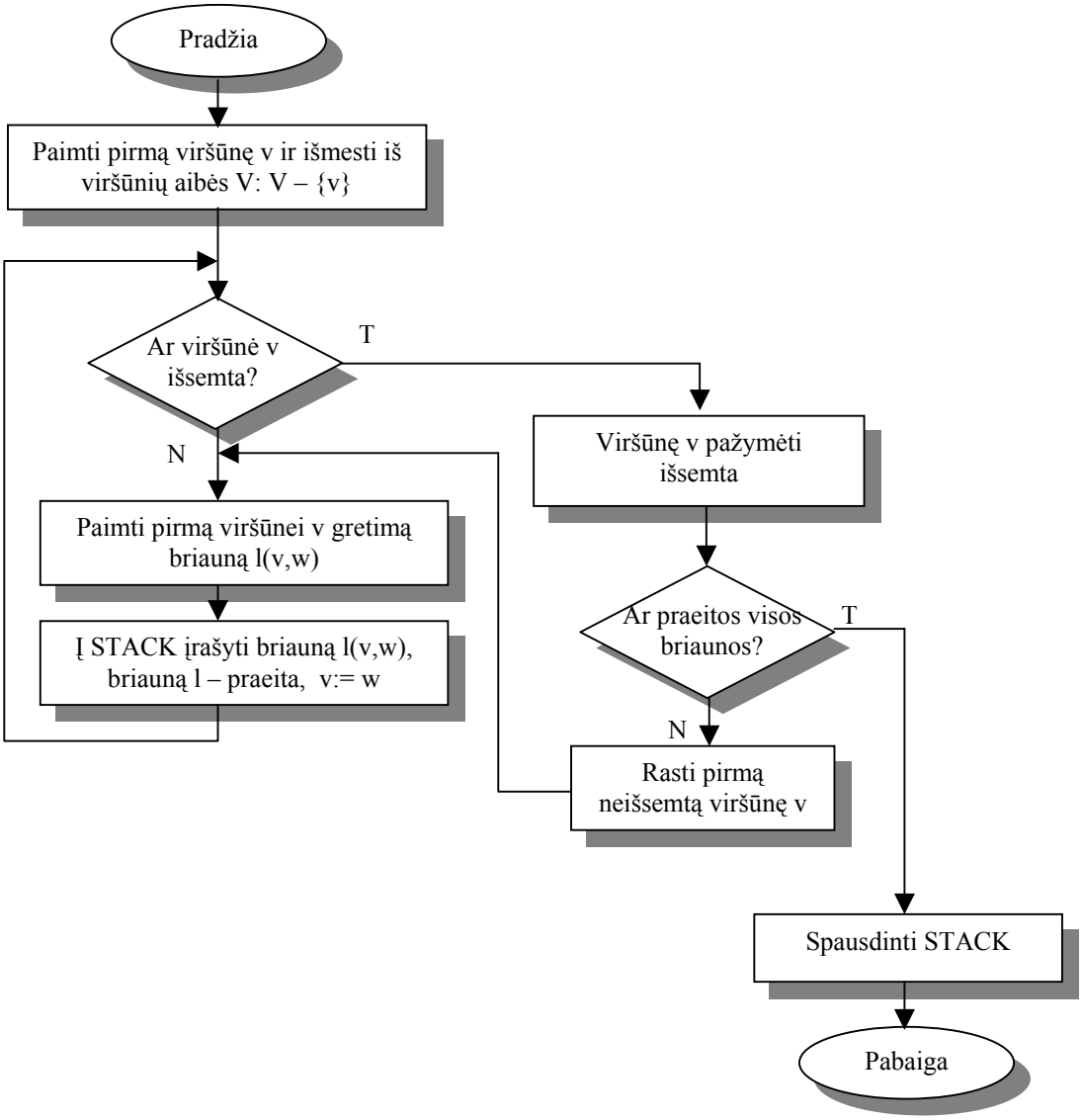
Pilno perrinkimo algoritmo supaprastintos blokinės schemos žymėjimai:

STACK – briaunų, įeinančių į maršrutą stekas.

V – viršūnių aibė, pradžioje aibę sudaro visos grafo viršūnės.

Grafo viršūnė v išsemta, jei praeitos visos briaunos, gretimos šiai viršūnei.

2.2 pav. pateikta tik pirmo maršruto suformavimo blokinė schema.



2.2 pav. Pilno perrinkimo algoritmo supaprastinta blokinė schema

Suradus pirmą maršrutą, iš steko išmetama paskutinė į jį įtraukta briauna. Abi šios briaunos viršūnės pažymimos kaip neišsemtos. Paieška tęsiama toliau, t. y., tikrinama, ar yra neišsemtų viršūnių. Be to, laikomasi sąlygos, kad to paties maršruto kartoti nebegalima. Pilno perrinkimo algoritmas baigia darbą, kai paieška įvykdoma pradedant kiekviena viršūne iš viršūnių aibės V. Pilno perrinkimo metu formuojamas paieškos medis. Iš medžio šaknies į kiekvieną terminalinę viršūnę veda vienintelis kelias. Medis turės tiek terminalinių viršūnių, kiek bus surasta galimų maršrutų. Maršrutas randamas einant nuo terminalinės viršūnės iki šakninės medžio viršūnės. Palyginus visus maršrutus, išrenkamas geriausias. Jei pilno perrinkimo algoritmas optimizuoja maršruto ilgį, tai suradus kiekvieną maršrutą, paskaičiuojamas šio maršruto ilgis. Jei šis algoritmas optimizuoja proceso trukmę, tai paskaičiuojama šio maršruto proceso trukmė.

2.2. Pasiūlytas algoritmas ruošinių realizacijos optimizavimui

Pasiūlyto algoritmo supaprastintos blokinės schemos žymėjimai:

V – viršūnių aibė, pradžioje aibę sudaro visos grafo viršūnės.

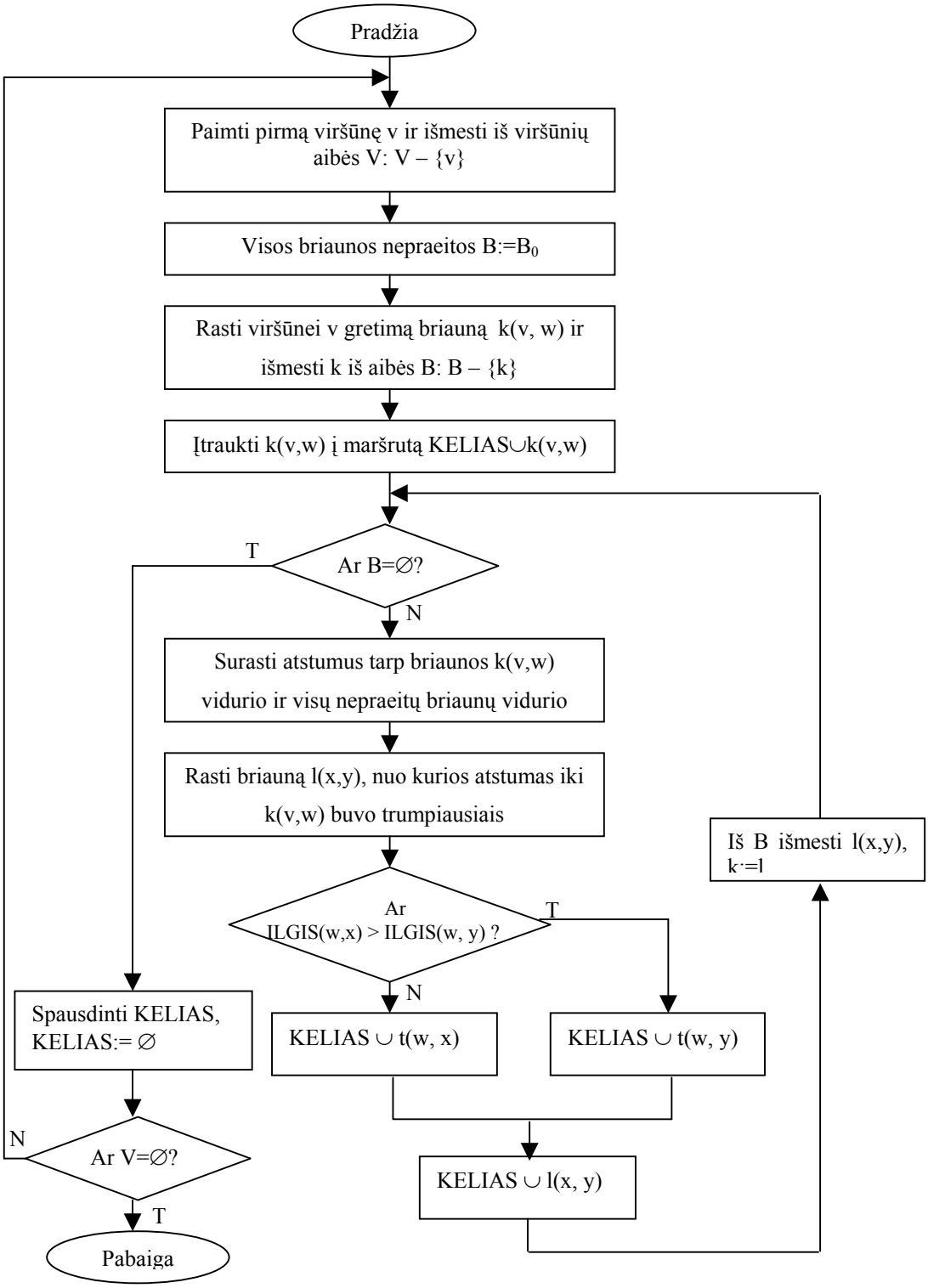
B – nepraeitų briaunų aibė, pradžioje aibę sudaro visos grafo briaunos.

B_0 – grafo briaunų aibė.

KELIAS – ieškomo maršruto briaunų sąrašas, pradžioje KELIAS:=∅.

ILGIS(v,w) – atstumas tarp viršūnių v ir w.

Pasiūlytos paieškos algoritmas vykdomas nuo visų viršūnių. Atstumo tarp briaunų vidurio paieškai naudojamas artimiausio kaimyno metodas. Baigus darbą, išrenkamas geriausias maršrutas, iš surastų maršrutų nuo kiekvienos grafo viršūnės. 2.3 pav. pateikta maršruto formavimo blokinė schema, kuomet optimizuojamas maršruto ilgis.

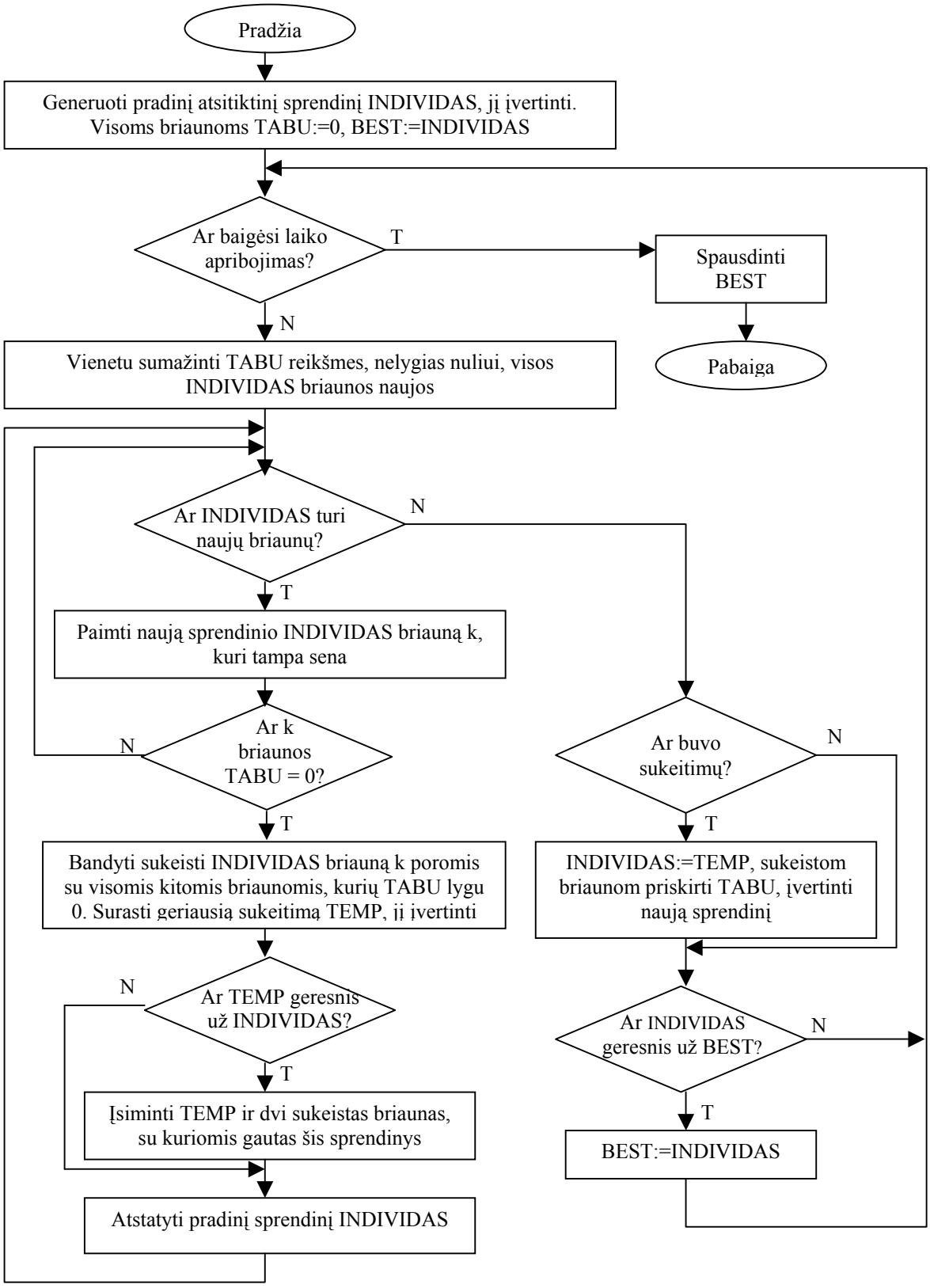


2.3 pav. Pasiūlyto algoritmo blokinė schema

Jei pasiūlytas algoritmas optimizuoja proceso trukmę, tai aukščiau pateiktą schemą reikia modifikuoti. Bloką „*Rasti briauną l(x,y), nuo kurios atstumas iki k(v,w) buvo trumpiausiais*“ reikia pakeisti į „*Rasti briauną l(x,y), nuo kurios proceso trukmė iki k(v,w) buvo mažiausia*“. Šiuo atveju papildomai tikrinama sąlyga *„(LAIKAS(w,x) + LAIKAS(x,y)) > (LAIKAS(w,y) + LAIKAS(x,y))*“, čia žymėjimas LAIKAS(x,y) – tai proceso trukmė tarp viršūnių x ir y.

2.3. Tabu algoritmas ruošinių realizacijos optimizavimui

2.4 pav. Pateikta tabu paieškos maršruto formavimo blokinė schema.



2.4 pav. Tabu paieškos algoritmo blokinė schema

Tabu algoritmo supaprastintos blokinės schemos žymėjimai:

INDIVIDAS – pradinis sprendinys.

BEST – geriausias sprendinys.

TEMP – geriausias tarpinis sprendinys.

TABU – tabu reikšmė, kuri priklauso nuo grafo dydžio.

Tabu paieška nebaigia skaičiavimų, suradus pirmą lokalų minimalų maršrutą. Šio algoritmo darbą galima nutraukti bet kuriuo momentu. Nutraukus greitai, sprendinys bus blogesnis. Kiekvienam surastam sprendiniui skaičiuojamas įvertis, t. y. maršruto ilgis arba proceso trukmė, pagal kurį atliekamas sprendinių lyginimas. Jei tabu paieškos algoritmas optimizuoja maršruto ilgį, tai sprendinio įvertis yra surasto maršruto ilgis. Jei šis algoritmas optimizuoja proceso trukmę, tai sprendinio įvertis – surasto maršruto proceso trukmė.

3. EKSPERIMENTINĖ DALIS

Eksperimentas atliekamas sukurtos programos pagalba. Šios programos trumpas aprašymas pateikiamas prieduose (žr. 5 priedas). Visi eksperimente naudojami duomenys yra kompaktinėje plokštelėje, įsegtoje prieduose (žr. 6 priedas). Duomenų failas – tai ruošinio realizaciją aprašanti informacija. Šios informacijos aprašymas pagrįstas duomenų struktūrizavimo modeliu XML (*eXtensible Markup Language*). XML tipo failo pavyzdys pateiktas prieduose (žr. 1 priedas). XML failo struktūra parodyta lentelėje:

3.1 lentelė. Duomenų failo struktūra

XML elementas	Aprašymas
line	Informacija apie ruošinio briauną
x1	Briaunos pirmos viršūnės x koordinatė
y1	Briaunos pirmos viršūnės y koordinatė
x2	Briaunos antros viršūnės x koordinatė
y2	Briaunos antros viršūnės y koordinatė
curve	Požymis, ar briauna yra tiesė
thick	Briaunos storis

Ruošinių realizacijos optimizavimui sudaryti algoritmai, kurie dėl trumpumo toliau bus vadinami taip:

- **pilnas perrinkimas (ilgis)** – pilno perrinkimo algoritmo realizacija, optimizuojanti maršruto ilgį;
- **pilnas perrinkimas (laikas)** – pilno perrinkimo algoritmo realizacija, optimizuojanti proceso trukmę;
- **tabu paieška (ilgis)** – tabu algoritmo realizacija, optimizuojanti maršruto ilgį;
- **tabu paieška (laikas)** – tabu algoritmo realizacija, optimizuojanti proceso trukmę;
- **pasiūlyta paieška (ilgis)** – pasiūlyto algoritmo realizacija, optimizuojanti maršruto ilgį;
- **pasiūlyta paieška (laikas)** – pasiūlyto algoritmo realizacija, optimizuojanti proceso trukmę;

Šie algoritmai lyginami įvairiais aspektais. Daugelyje eksperimento rezultatų lentelių pateikiamos vidutinės nagrinėjamų dydžių reikšmės. Pavyzdžiui, tiriant pilno perrinkimo

algoritmo skaičiavimų laiko sąnaudų priklausomybę nuo ruošinį sudarančių briaunų kiekio, šis algoritmas vykdomas dešimt kartų ir paimamas gautų skaičiavimo laikų vidurkis.

3.1. Ruošinių realizacijos algoritmų efektyvumo palyginimas

3.1.1 Algoritmų palyginimas, naudojant skirtingus duomenis

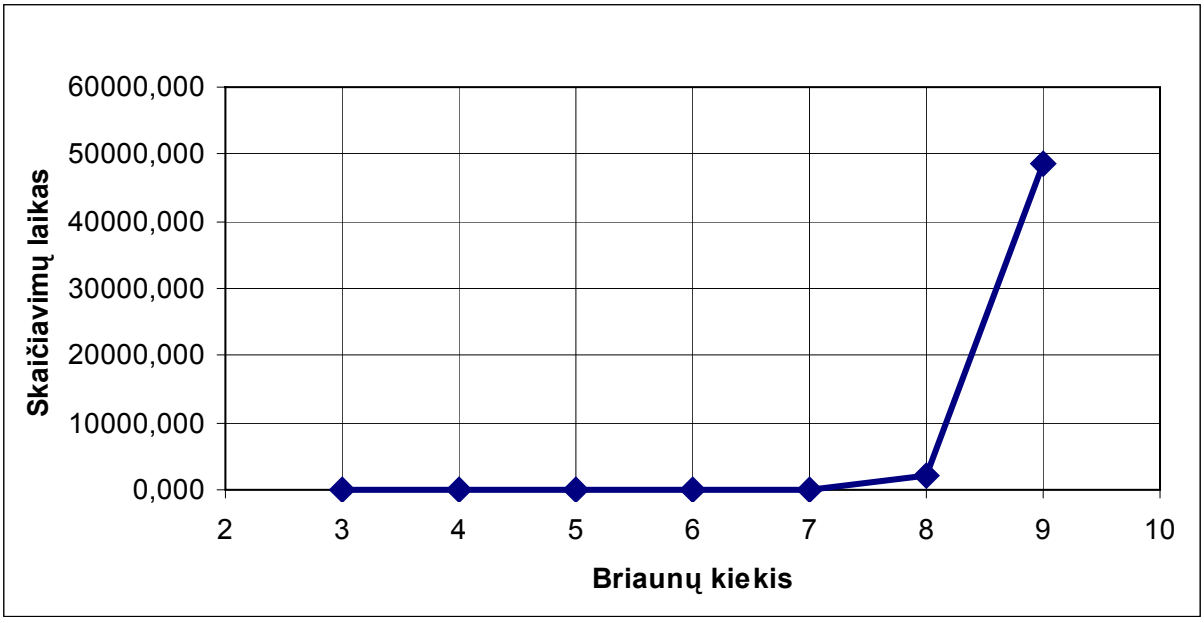
Pilno perrinkimo, tabu paieškos ir pasiūlytos paieškos algoritmų efektyvumas lyginamas pagal skaičiavimų laiko sąnaudas.

Pilno perrinkimo algoritmas tiriamas, naudojant skirtingus duomenis. Šie duomenų failai skiriasi dydžiu, t. y. ruošinį sudarančių briaunų kiekiu. Pilno perrinkimo vidutiniai skaičiavimų laiko rezultatai, naudojant vis didesnius duomenų failus, pateikti 3.2 lentelėje. Šioje lentelėje pateikti tik maršrutą optimizuojančio pilno perrinkimo algoritmo rezultatai, nes optimizuojant proceso trukmę, gaunami panašūs rezultatai.

3.1 pav. parodyta pilno perrinkimo skaičiavimų laiko priklausomybė nuo ruošinio briaunų kiekio. Iš šio grafiko matyti, kad algoritmo skaičiavimų laikas auga eksponentiniu dėsniu.

3.2 lentelė. Pilno perrinkimo algoritmo skaičiavimų laiko rezultatai

	Pilnas perrinkimas (ilgis)						
Briaunų kiekis	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Vidutinis laikas sekundėmis	0,041	0,070	0,521	7,681	133,983	2082,032	48582,918



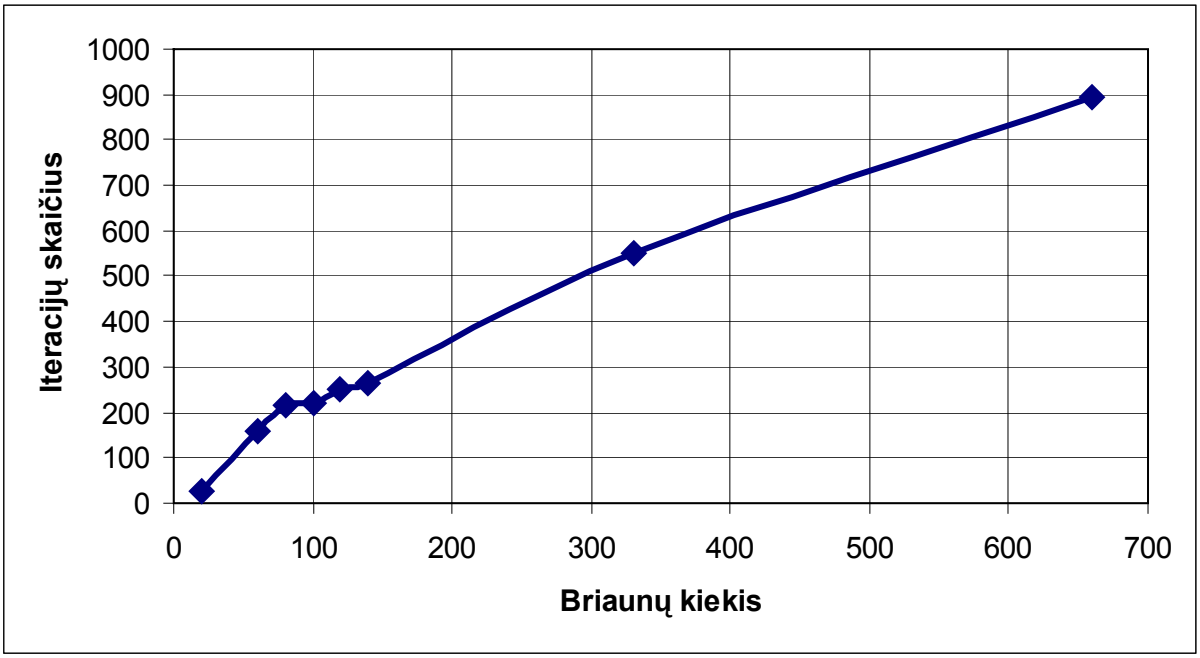
3.1 pav. Pilno perrinkimo skaičiavimų laiko priklausomybė nuo briaunų kiekio

Tabu paieškos algoritmas tiriamas, naudojant skirtingo dydžio duomenų failus. Tabu paieškos algoritmo skaičiavimų pabaiga priklauso nuo iš anksto užduoto skaičiavimų laiko apribojimo. Jei skaičiavimų laiko apribojimas didesnis, tai gaunamas geresnis sprendinys. Kadangi sunku nustatyti tabu paieškos skaičiavimų laiko sąnaudas, nagrinėjamas vidutinis iteracijų skaičius. Iteracijų skaičius rodo, kiek kartų tabu paieškos skaičiavimų metu buvo pagerintas sprendinys. Šios paieškos vidutiniai iteracijų skaičiaus rezultatai, naudojant skirtingo duomenis, pateikti 3.3 lentelėje. Šioje lentelėje pateikti tik maršrutą optimizuojančio tabu algoritmo rezultatai, nes optimizuojant proceso trukmę, gaunami panašūs rezultatai.

3.2 pav. parodyta tabu paieškos iteracijų skaičiaus priklausomybė nuo ruošinio briaunų kiekio. Iš šio grafiko matyti, kad algoritmo iteracijų skaičius auga tiesiškai, didėjant briaunų kiekiui.

3.3 lentelė. Tabu paieškos iteracijų skaičiaus rezultatai

	Tabu paieška (ilgis)							
Briaunų kiekis	**20**	**60**	**80**	**100**	**120**	**140**	**330**	**660**
Vidutinis iteracijų kiekis	26	159	216	222	252	266	549	893



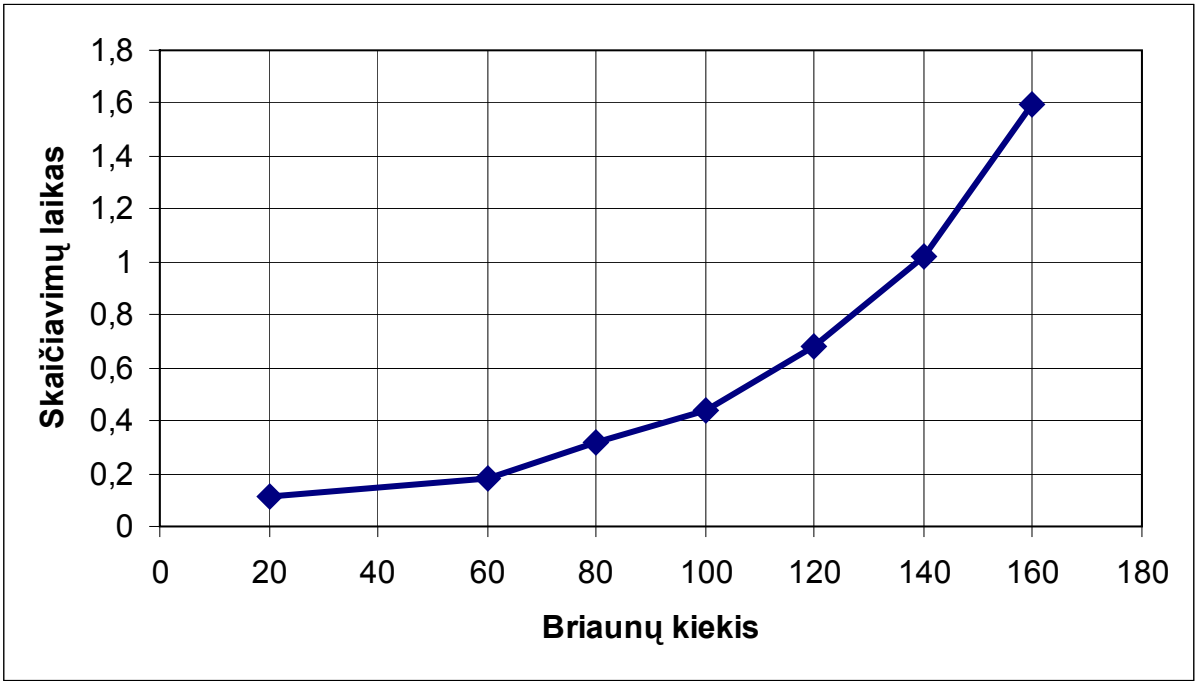
3.2 pav. Tabu paieškos iteracijų skaičiaus priklausomybė nuo briaunų kiekio

Pasiūlytos paieškos algoritmas tiriamas, naudojant skirtingo dydžio duomenų failus. Šios paieškos vidutiniai skaičiavimų laiko rezultatai, naudojant skirtingo duomenis, pateikti 3.4 lentelėje. Šioje lentelėje pateikti tik maršrutą optimizuojančio pasiūlytos paieškos

algoritmo rezultatai, nes optimizuojant proceso trukmę, gaunami panašūs rezultatai. 3.3 pav. parodyta šios paieškos skaičiavimų laiko priklausomybė nuo ruošinio briaunų kiekio.

3.4 lentelė. Pasiūlytos paieškos skaičiavimų laiko rezultatai

	Pasiūlyta paieška (ilgis)						
Briaunų kiekis	**20**	**60**	**80**	**100**	**120**	**140**	**160**
Vidutinis laikas sekundėmis	0,111	0,181	0,321	0,441	0,681	1,021	1,593



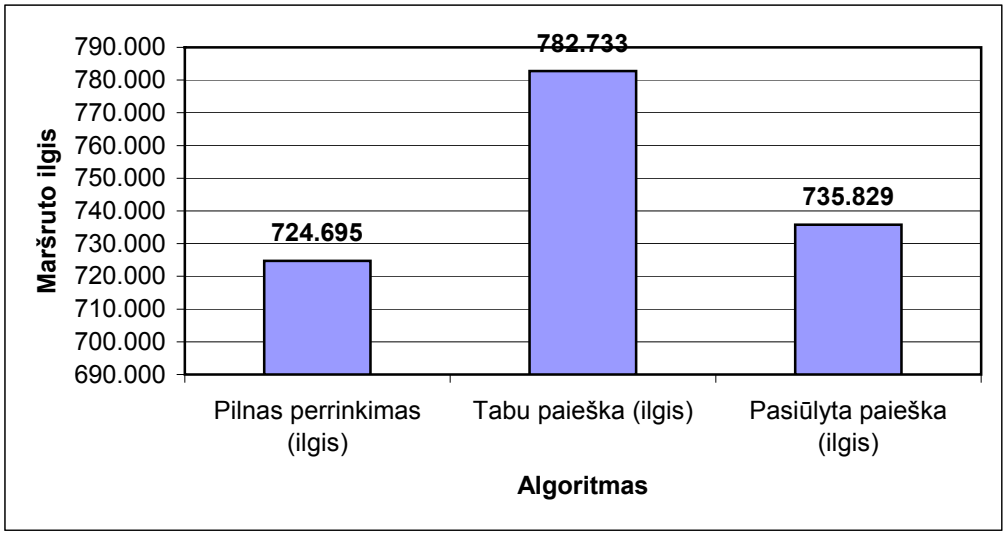
3.3 pav. Pasiūlytos paieškos skaičiavimų laiko priklausomybė nuo briaunų kiekio

3.1.2 Algoritmų palyginimas, naudojant tuos pačius duomenis

Pilno perrinkimo, tabu paieškos ir pasiūlytos paieškos algoritmai tiriami, naudojant tuos pačius duomenis. Šių algoritmų vidutiniai skaičiavimų laiko, surasto maršruto ilgio ir proceso trukmės rezultatai, optimizuojant maršruto ilgį, pateikti 3.5 lentelėje. Šių algoritmų surastų maršrutų ilgių palyginimas pateiktas stulpelinėje diagramoje (žr. 3.4 pav.).

3.5 lentelė. Visų algoritmų rezultatų palyginimas optimizuojant maršruto ilgį

Algoritmas	Briaunų kiekis	Maršruto ilgis	Proceso trukmė	Vidutinis skaičiavimo laikas
Pilnas perrinkimas (ilgis)	8	724.695	73.104	3155,808
Tabu paieška (ilgis)	8	782.733	73.465	0,010
Pasiūlyta paieška (ilgis)	8	735.829	72.073	0,046

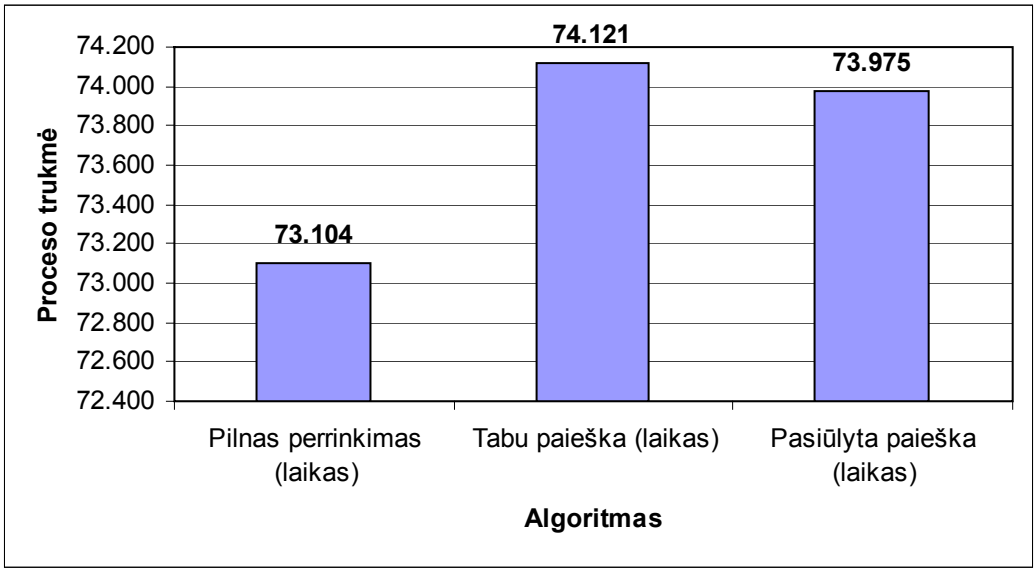


3.4 pav. Maršruto ilgio priklausomybė nuo parinkto algoritmo

Pilno perrinkimo, tabu paieškos ir pasiūlytos paieškos algoritmų vidutiniai skaičiavimų laiko, surasto maršruto ilgio ir proceso trukmės rezultatai, optimizuojant proceso trukmę, pateikti 3.6 lentelėje. Šių algoritmų surastų proceso trukmių palyginimas pateiktas stulpelinėje diagramoje (žr. 3.5 pav.).

3.6 lentelė. Visų algoritmų rezultatų palyginimas optimizuojant proceso trukmę

Algoritmas	Briaunų kiekis	Maršruto ilgis	Proceso trukmė	Vidutinis skaičiavimo laikas
Pilnas perrinkimas (laikas)	8	724.695	73.104	3155,808
Tabu paieška (laikas)	8	805.525	74.121	0,010
Pasiūlyta paieška (laikas)	8	768.224	73.975	0,040



3.5 pav. Proceso trukmės priklausomybė nuo parinkto algoritmo

3.2. Algoritmų palyginimas pagal surasto maršruto ilgį ir proceso trukmę

Pilno perrinkimo algoritmas tiriamas optimizuojant maršruto ilgį ir proceso trukmę. Pilno perrinkimo algoritmo rezultatai, optimizuojant maršruto ilgį, pateikti 3.7 lentelėje. Šio algoritmo rezultatai, optimizuojant proceso trukmę, pateikti 3.8 lentelėje.

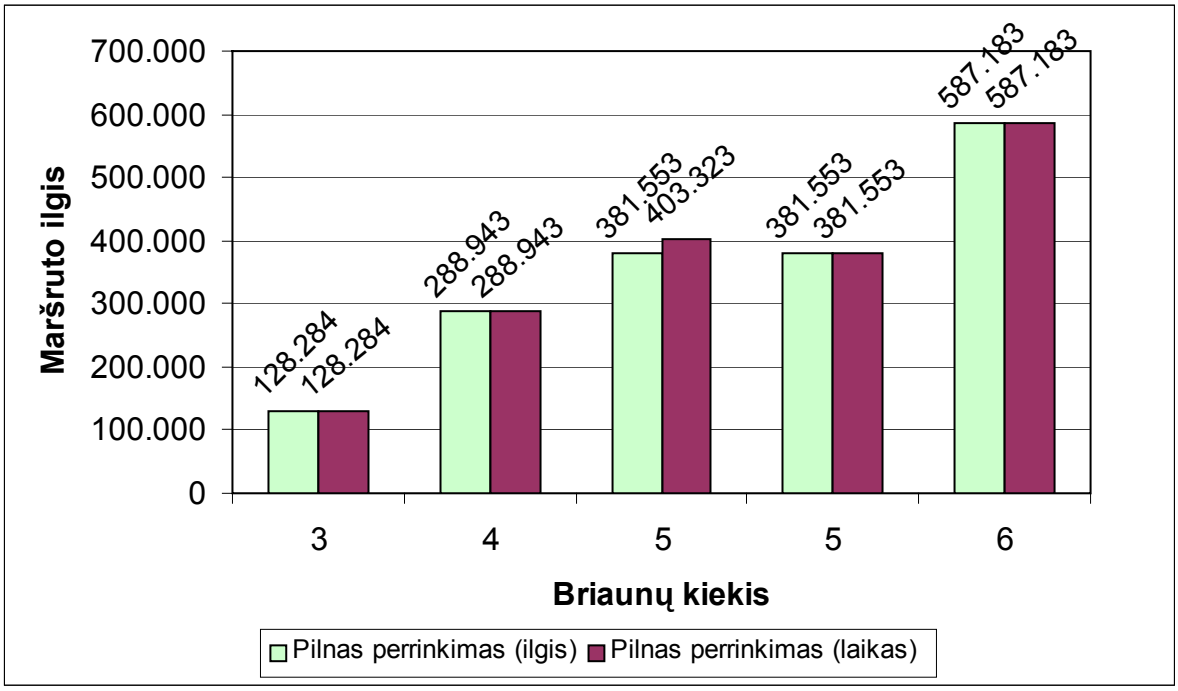
Pilno perrinkimo algoritmo surastų maršrutų ilgių palyginimas, optimizuojant maršruto ilgį ir proceso trukmę, vaizduojamas stulpelinėje diagramoje (žr. 3.6 pav.). Šio algoritmo surastų proceso trukmių palyginimas, optimizuojant maršruto ilgį ir proceso trukmę, vaizduojamas stulpelinėje diagramoje (žr. 3.7 pav.).

3.7 lentelė. Pilno perrinkimo rezultatai, optimizuojant maršruto ilgį

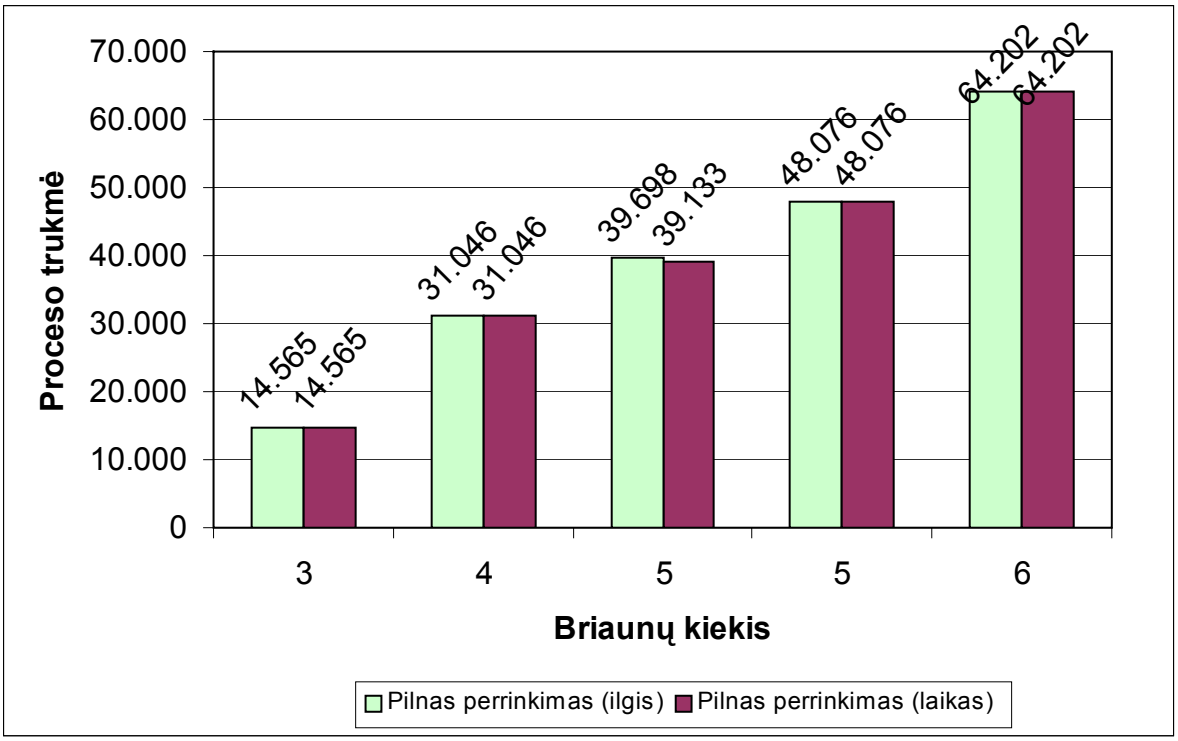
	Pilnas perrinkima (ilgis)				
Briaunų kiekis	**3**	**4**	**5**	**5**	**6**
Failo vardas	**Br3d.xml**	**Br4d.xml**	**Br5d.xml**	**Br5d.xml**	**Br6d.xml**
Maršruto ilgis	128.284	288.943	381.553	381.553	587.183
Proceso trukmė	14.565	31.046	39.698	48.076	64.202

3.8 lentelė. Pilno perrinkimo rezultatai, optimizuojant proceso trukmę

	Pilnas perrinkima (laikas)				
Briaunų kiekis	**3**	**4**	**5**	**5**	**6**
Failo vardas	**Br3d.xml**	**Br4d.xml**	**Br5d.xml**	**Br5d.xml**	**Br6d.xml**
Maršruto ilgis	128.284	288.943	403.323	381.553	587.183
Proceso trukmė	14.565	31.046	39.133	48.076	64.202



3.6 pav. Pilno perrinkimo surastų maršrutų ilgių palyginimas, optimizuojant maršruto ilgį ir proceso trukmę



3.7 pav. Pilno perrinkimo surastų proceso trukmių palyginimas, optimizuojant maršruto ilgį ir proceso trukmę

Tabu algoritmas tiriamas, optimizuojant maršruto ilgį ir proceso trukmę. Tabu algoritmo rezultatai, optimizuojant maršruto ilgį, pateikti 3.9 lentelėje. Šio algoritmo rezultatai, optimizuojant proceso trukmę, pateikti 3.10 lentelėje. Kadangi tabu paieškos algoritmas yra atsitiktinis, šiose lentelėse rodomos vidutinės maršruto ilgio ir proceso trukmės reikšmės.

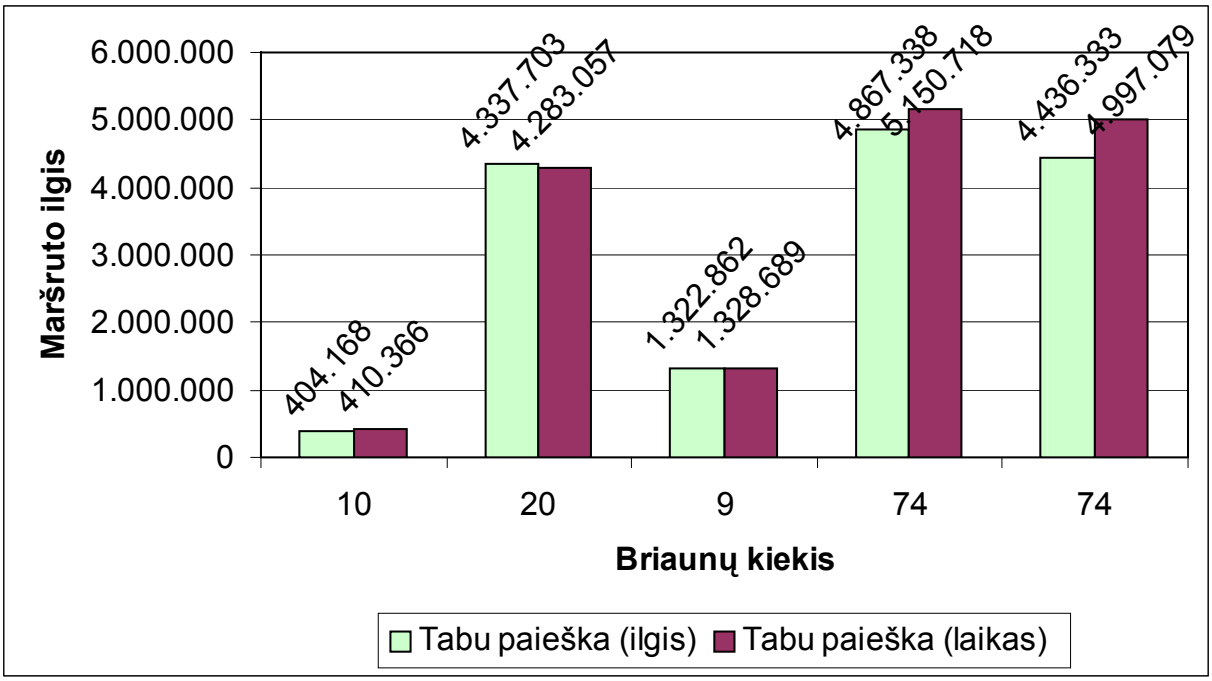
Tabu algoritmo surastų maršrutų ilgių palyginimas, optimizuojant maršruto ilgį ir proceso trukmę, vaizduojamas stulpelinėje diagramoje (žr. **3.8 pav.**). Šio algoritmo surastų proceso trukmių palyginimas, optimizuojant maršruto ilgį ir proceso trukmę, vaizduojamas stulpelinėje diagramoje (žr. 3.9 pav.).

3.9 lentelė. Tabu algoritmo rezultatai, optimizuojant maršruto ilgį

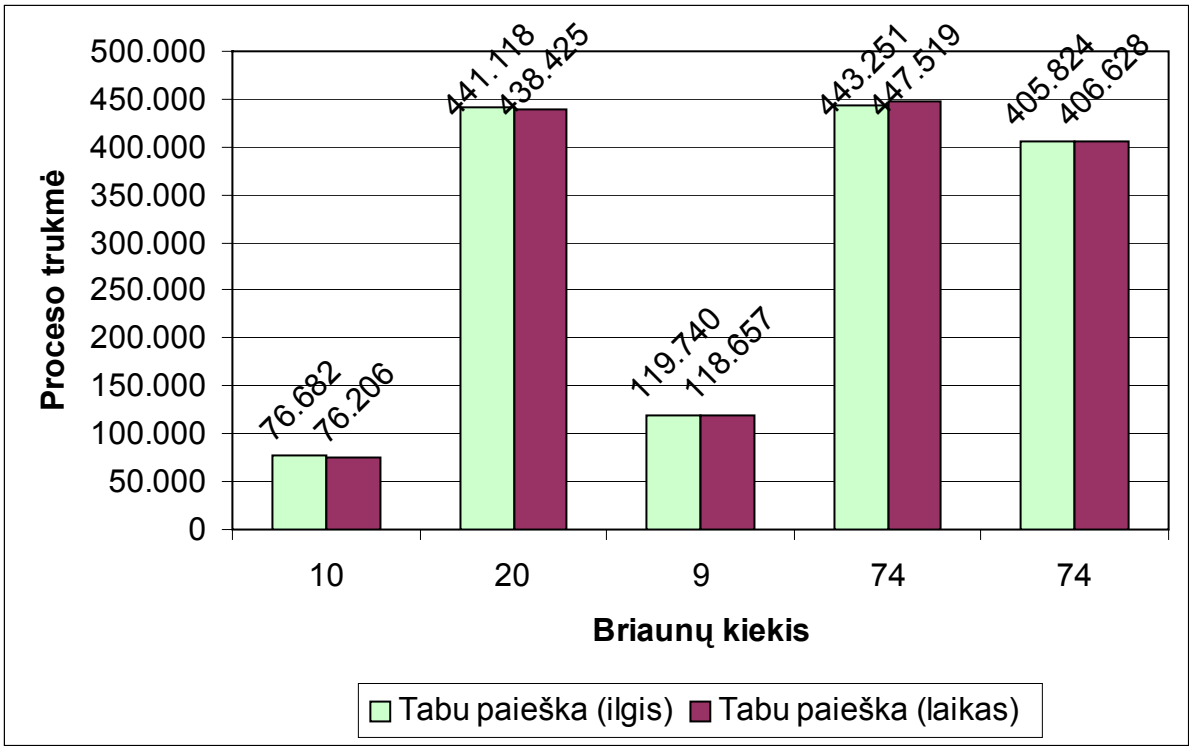
	Tabu paieška (ilgis)				
Briaunų kiekis	**10**	**20**	**9**	**74**	**74**
Failo vardas	**Br10d.xml**	**Br20d.xml**	**Br9d.xml**	**Br74d.xml**	**Br74d.xml**
Maršruto ilgis	404.168	4.337.703	1.322.862	4.867.338	4.436.333
Proceso trukmė	76.682	441.118	119.740	443.251	405.824

3.10 lentelė. Tabu algoritmo rezultatai, optimizuojant proceso trukmę

	Tabu paieška (laikas)				
Briaunų kiekis	**10**	**20**	**9**	**74**	**74**
Failo vardas	**Br10d.xml**	**Br20d.xml**	**Br9d.xml**	**Br74d.xml**	**Br74d.xml**
Maršruto ilgis	410.366	4.283.057	1.328.689	5.150.718	4.997.079
Proceso trukmė	76.206	438.425	118.657	447.519	406.628



3.8 pav. Tabu algoritmo surastų maršrutų ilgių palyginimas, optimizuojant maršruto ilgį ir proceso trukmę



3.9 pav. Tabu algoritmo surastų proceso trukmių palyginimas, optimizuojant maršruto ilgį ir proceso trukmę

Pasiūlytas algoritmas tiriamas, optimizuojant maršruto ilgį ir proceso trukmę. Pasiūlyto algoritmo rezultatai, optimizuojant maršruto ilgį, pateikti 3.11 lentelėje. Šio algoritmo rezultatai, optimizuojant proceso trukmę, pateikti 3.12 lentelėje.

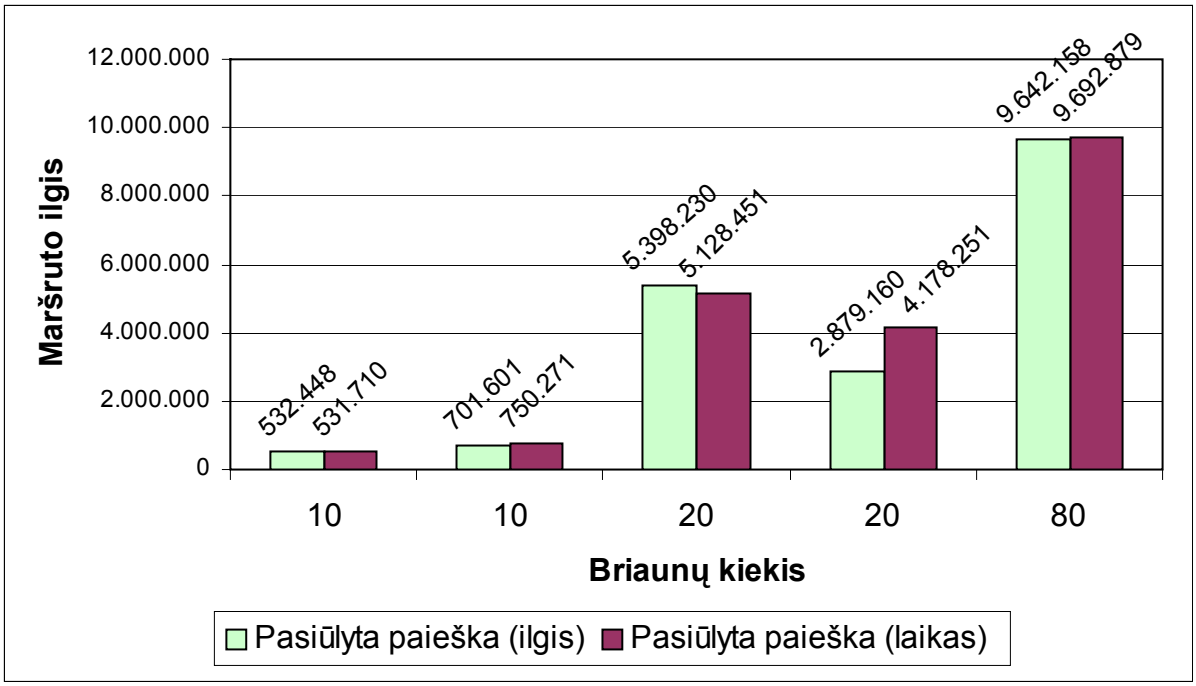
Pasiūlyto algoritmo surastų maršrutų ilgių palyginimas, optimizuojant maršruto ilgį ir proceso trukmę, pavaizduotas stulpelinėje diagramoje (žr. 3.10 pav.). Pasiūlyto algoritmo surastų proceso trukmių palyginimas, optimizuojant maršruto ilgį ir proceso trukmę, pateiktas stulpelinėje diagramoje (žr. 3.11 pav.).

3.11 lentelė. Pasiūlyto algoritmo rezultatai, optimizuojant maršruto ilgį

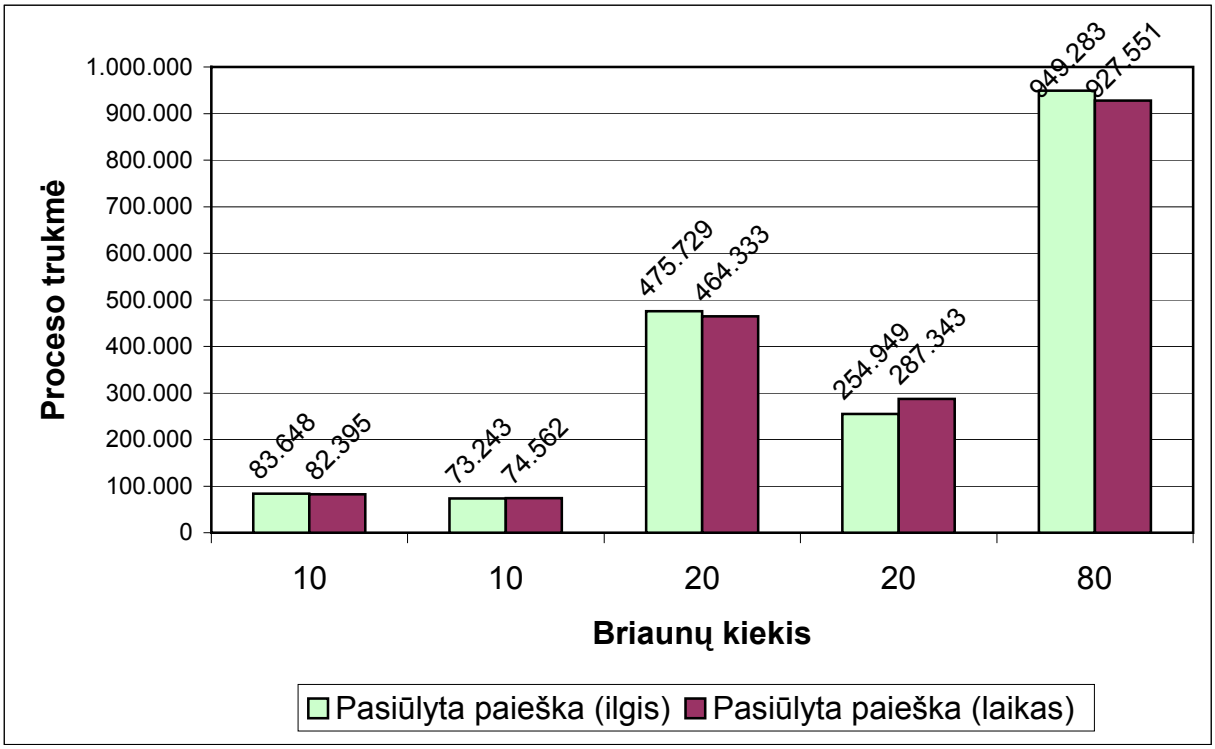
	Pasiūlyta paieška (ilgis)				
Briaunų kiekis	**10**	**10**	**20**	**20**	**80**
Failo vardas	**Br10d.xml**	**Br10_1d.xml**	**Br20d.xml**	**Br20d_2.xml**	**Br80d_1.xml**
Maršruto ilgis	532.448	701.601	5.398.230	2.879.160	9.642.158
Proceso trukmė	83.648	73.243	475.729	254.949	949.283

3.12 lentelė. Pasiūlyto algoritmo rezultatai, optimizuojant proceso trukmę

	Pasiūlyta paieška (laikas)				
Briaunų kiekis	**10**	**10**	**20**	**20**	**80**
Failo vardas	**Br10d.xml**	**Br10_1d.xml**	**Br20d.xml**	**Br20d_2.xml**	**Br80d_1.xml**
Maršruto ilgis	531.710	750.271	5.128.451	4.178.251	9.692.879
Proceso trukmė	82.395	74.562	464.333	287.343	927.551



3.10 pav. Pasiūlyto algoritmo surastų maršrutų ilgių palyginimas, optimizuojant maršruto ilgį ir proceso trukmę



3.11 pav. Pasiūlyto algoritmo surastų proceso trukmių palyginimas, optimizuojant maršruto ilgį ir proceso trukmę

3.3. Parametrų įtaka ruošinių realizacijos algoritmams

Svarbus tyrimo aspektas – kokią įtaka sudarytų algoritmų skaičiavimų rezultatams daro parametrai. Programoje galima keisti šiuos parametrus:

- **Uždelsimas** – įrenginio uždelsimo dydis viršūnėje;
- **Greitis tuščia eiga** – įrenginio greitis judant tuščiąja eiga;
- **Greitis linija** – įrenginio greitis tiese horizontalia ar vertikalia linija;
- **Greitis kampu** – įrenginio greitis tiese nehorizontalia ar nevertikalia linija;
- **Greitis kreive** – įrenginio greitis kreive.

Įrenginio pavyzdys galėtų būti braižiklis, skirtas įvairių schemų, brėžinių, žemėlapių ir pan. braižymui. Taigi, šie parametrai įvertina tam tikro įrenginio darbo proceso savybes. Šis procesas gali būti braižymas, pjaustymas, karpymas ar pan.

3.13 lentelėje pateiktos pradinės parametrų reikšmės. Tarkime, kad tai braižiklio, skirto spausdinto montažo plokštės takelių braižymui, darbo proceso savybes nusakantys parametrai. Šie parametrai tuomet reikštų, kad braižiklio greitis judant tuščiąja eiga yra toks pat, kaip brėžiant liniją.

3.13 lentelė. Pradinės parametrų reikšmės

Parametras	Reikšmė
Uždelsimas	1,000
Greitis tuščia eiga	10,000
Greitis tiesia linija	10,000
Greitis kampu	10,000
Greitis kreive	10,000

Pilno perrinkimo algoritmo rezultatai, esant anksčiau nustatytoms parametrų reikšmėms, pateikti 3.2 lentelėje. Pirmoje eilutėje rodomi rezultatai, optimizuojant maršruto ilgį, antroje – proceso trukmę. Ši lentelė rodo, kad abiem atvejais gaunami tie patys rezultatai.

3.14 lentelė. Pilno perrinkimo rezultatai, esant pradinėms parametrų reikšmėms

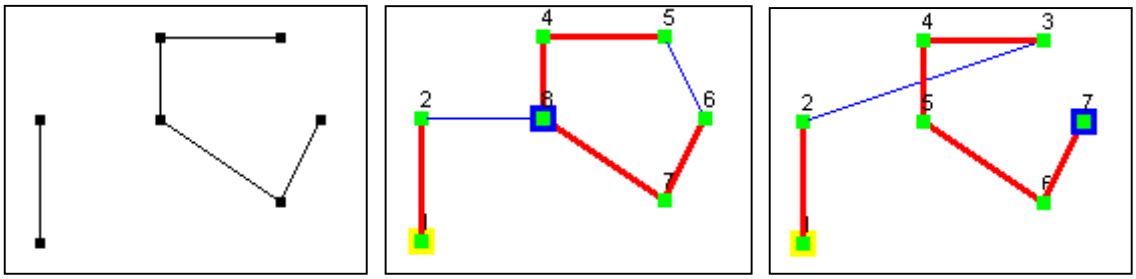
Algoritmas	Briaunų kiekis	Maršruto ilgis	Proceso trukmė
Pilnas perrinkimas (ilgis)	5	381.553	48.076
Pilnas perrinkimas (laikas)	5	381.553	48.076

Tarkime, kad braižiklio darbo greitis tuščiąja eiga yra penkis kartus didesnis, nei brėžiant liniją. Šiuo atveju gauti pilno perrinkimo rezultatai pateikti 3.15 lentelėje.

3.15 lentelė. Pilno perrinkimo rezultatai, pakeitus parametrų reikšmėms

Algoritmas	Briaunų kiekis	Maršruto ilgis	Proceso trukmė
Pilnas perrinkimas (ilgis)	5	381.553	39.698
Pilnas perrinkimas (laikas)	5	403.323	39.133

Ši lentelė rodo, kad optimizuojant proceso trukmę buvo gautas prastesnis maršrutas ilgio prasme, bet geresnis proceso trukmės prasme. Pakeitus parametrų reikšmes, pilno perrinkimo algoritmo darbo rezultatai optimizuojant maršruto ilgį ir proceso trukmę, pavaizduoti 3.12 pav. (šie skaičiavimų rezultatų pavyzdžiai paimti iš sukurtos programos (žr. 6 priedas)).



a) **b)** **c)**

3.12 pav. a) pradinis brėžinys; b) maršrutas optimizuojant ilgį; c) maršrutas optimizuojant proceso trukmę

Imkime pasiūlytą paieškos algoritmą ir duomenų failą „*Br20d.xml*“. Tarkime, kad greitis judant tuščiąja eiga yra penkis kartus didesnis, nei greitis tiesia linija. Šiuo atveju pasiūlyto algoritmo skaičiavimų rezultatai pateikti 3.16 lentelėje.

3.16 lentelė. Pasiūlyto algoritmo rezultatai, padidinus greitį tuščiąja eiga penkis kartus

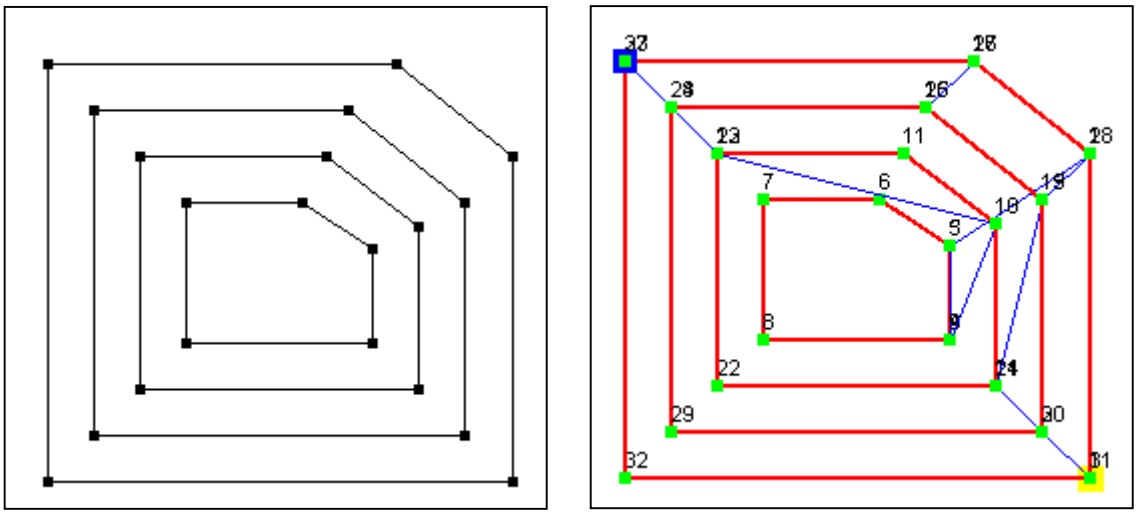
Algoritmas	Briaunų kiekis	Maršruto ilgis	Skaičiavimų laikas
Pasiūlyta paieška (ilgis)	20	5.398.230	475.729
Pasiūlyta paieška (laikas)	20	5.128.451	464.333

Gauti rezultatai grąžinus pradines parametrų reikšmes, pateikti 3.17 lentelėje.

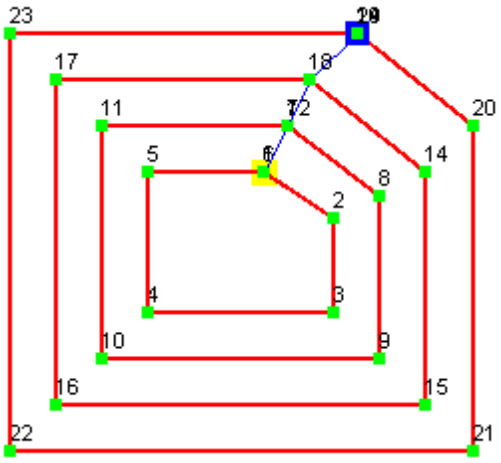
3.17 lentelė. Pasiūlyto algoritmo rezultatai, esant pradinėms parametrų reikšmėms

Algoritmas	Briaunų kiekis	Maršruto ilgis	Skaičiavimų laikas
Pasiūlyta paieška (ilgis)	20	5.398.230	588.528
Pasiūlyta paieška (laikas)	20	4.134.247	447.130

Ši lentelė rodo, kad pakeitus parametrus pasiūlytas algoritmas, kuris optimizuoja proceso trukmę, rado žymiai geresnį maršrutą tiek ilgio tiek trukmės prasme. Pakeitus parametrų reikšmes (žr. 3.17 lentelė), pasiūlyto algoritmo darbo rezultatai optimizuojant maršruto ilgį ir proceso trukmę, pavaizduoti 3.13 pav.



a) **b)**



c)

3.13 pav. a) pradinis brėžinys; b) maršrutas optimizuojant ilgį; c) maršrutas optimizuojant proceso trukmę

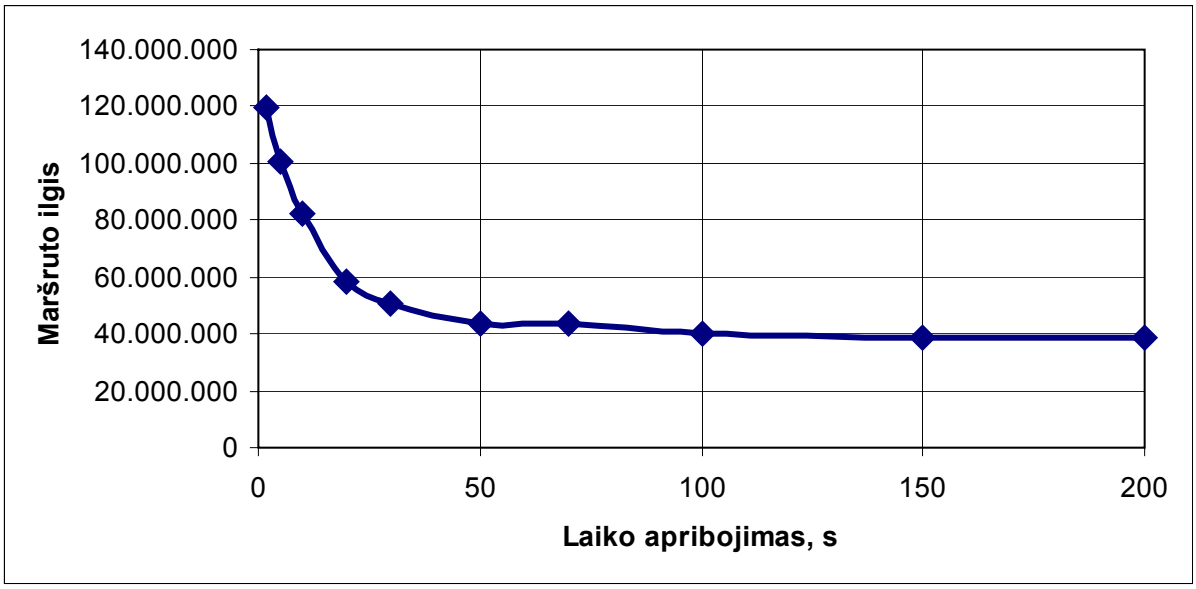
3.4. Tabu paieškos palyginimas pagal laiko apribojimą ir tabu reikšmę

Tabu paieškos algoritmas tiriamas, užduodant skirtingus skaičiavimų laiko apribojimus. Šiame tyrime naudojamas „*Br330d.xml*“ duomenų failas (ruošinį sudaro 330 briaunų). Tabu paieškos surasti vidutiniai maršruto ilgio ir proceso trukmės rezultatai, naudojant skirtingus laiko apribojimus, pateikti 3.18 lentelėje.

3.14 pav. parodyta šios paieškos surasto maršruto ilgio priklausomybė nuo skaičiavimų laiko apribojimo. Iš šio grafiko matyti, kad nuo tam tikros laiko apribojimo reikšmės, tabu paieškos surasto maršruto ilgis nesikeičia.

3.18 lentelė. Tabu paieškos rezultatai, keičiant laiko apribojimą

	Tabu paieška				
Laiko apribojimas, s	**2**	**5**	**10**	**20**	**30**
Maršruto ilgis	119.771.440	100.735.260	82.112.540	58.434.137	50.860.530
Realizacijos laikas	4.265.343	2.596.397	2.497.084	3.020.596	2.326.024
Laiko apribojimas, s	**50**	**70**	**100**	**150**	**200**
Maršruto ilgis	43.892.477	43.763.367	39.927.598	38.486.250	38.426.324
Realizacijos laikas	2.260.183	2.681.181	2.244.359	2.571.638	2.222.853



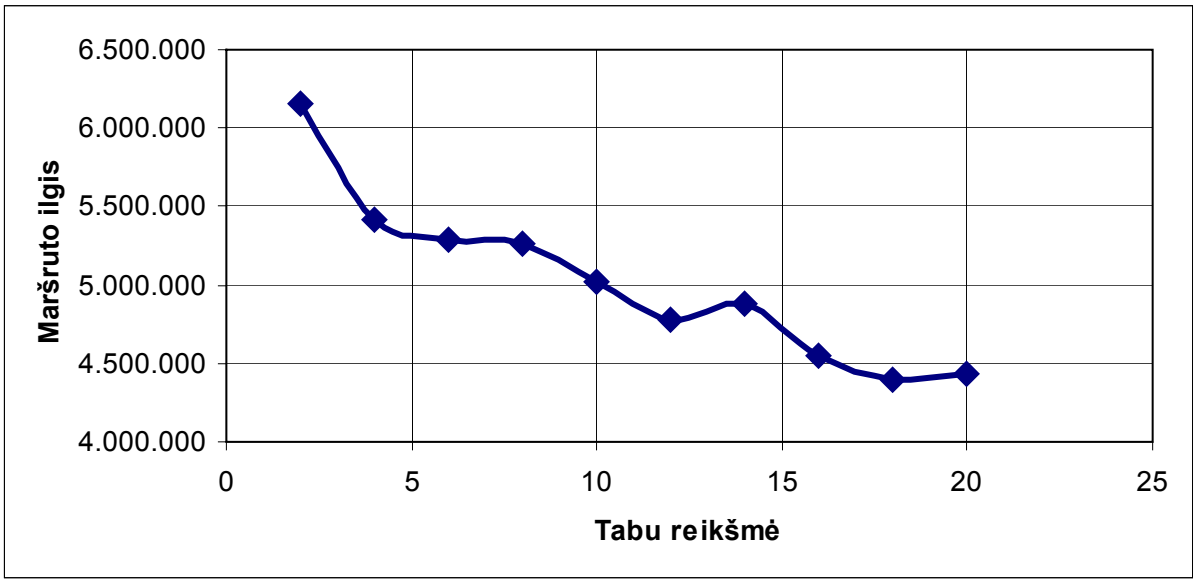
3.14 pav. Tabu paieškos maršruto ilgio priklausomybė nuo skaičiavimų laiko apribojimo

Tabu paieškos algoritmas tiriamas, užduodant skirtingas tabu reikšmes. Šios paieškos surasti vidutiniai maršruto ilgio ir proceso trukmės rezultatai, naudojant skirtingus tabu

apribojimus, pateikti 3.19 lentelėje. Didinant tabu reikšmę, tabu paieška randa geresnį maršrutą (žr. 3.15 pav.).

3.19 lentelė. Tabu paieškos rezultatai, keičiant tabu reikšmę

	Tabu paieška				
Tabu reikšmė	**20**	**18**	**16**	**14**	**12**
Maršruto ilgis	4.437.018	4.397.382	4.550.625	4.884.329	4.781.138
Realizacijos laikas	433.045	431.252	435.317	441.991	440.927
Tabu reikšmė	**10**	**8**	**6**	**4**	**2**
Maršruto ilgis	5.015.596	5.261.948	5.289.746	5.419.667	6.157.374
Realizacijos laikas	444.616	452.543	452.099	452.698	467.452



3.15 pav. Tabu paieškos maršruto ilgio priklausomybė nuo tabu reikšmės

Šiame skyriuje pateikti skaičiavimų laiko apribojimo ir tabu reikmės įtaka tabu paieškos algoritmo rezultatams optimizuojant maršruto ilgį. Tabu paieškos algoritmui optimizuojant proceso trukmę, šių parametrų įtaka rezultatams yra tokia pati.

IŠVADOS

1. Darbe nagrinėti ruošinių realizacijos optimizavimui tinkami metodai, išskirti šių metodų privalumai ir trūkumai.
2. Ruošinių realizacijos optimizavimui pasiūlyti ir realizuoti trys algoritmai:
    - pilnas perrinkimas;
    - tabu paieška;
    - siūlomas algoritmas.

   Kiekvienas šių algoritmų gali optimizuoti ruošinio realizacijos maršruto ilgį arba proceso trukmę. Jei algoritmas optimizuoja maršruto ilgį, galutiniam sprendiniui yra paskaičiuojama proceso trukmė. Jei algoritmas optimizuoja proceso trukmę, galutiniam sprendiniui yra paskaičiuojamas maršruto ilgis. Šiuo atveju, pasirinkus bet kurį iš sudarytų algoritmų, gaunami du ruošinių realizacijos įverčiai – maršruto ilgis ir proceso trukmė.
3. Ruošinių realizacijos optimizavimui sudaryti algoritmai palyginti pagal skaičiavimų laiko sąnaudas, pagal surasto maršruto ilgį ir proceso trukmę. Taip pat, tyrinėta ruošinių realizacijos technologinių ypatumų ir įrenginių darbo parametrų, tabu paieškos parametrų įtaka rezultatams.
4. Pilnas perrinkimas visiškai netinkamas ruošinių realizacijos optimizavimo problemoms spręsti, nors randamas optimalus sprendinys. Didėjant duomenų kiekiui, skaičiavimų apimtys auga labai sparčiai. Šio algoritmo rezultatai įdomūs tik teoriniu aspektu, lyginant juos su kitų algoritmų rezultatais ir stebint, kokią įtaką šiems rezultatams daro ruošinių realizacijos parametrai.
5. Naudojant tabu paiešką, sprendinio gerumas priklauso nuo skaičiavimams skirto laiko apribojimo. Daugiau laiko skiriant skaičiavimams, gaunamas geresnis sprendinys. Taikant šį paieškos būdą ruošinių realizacijos optimizavimui, gauti pakankamai geri rezultatai maršruto ilgio ir proceso trukmės atžvilgiu.
6. Daugeliu atveju, skaičiavimų trukmės, surasto maršruto ilgio ir proceso trukmės atžvilgiu, geriausi rezultatai gaunami veikiant pasiūlytam algoritmui.
7. Iš eksperimento metu gautų rezultatų, galima teigti, kad įvertinus ruošinių realizacijos technologinius ypatumus ir įrenginio darbo parametrus, galima sumažinti ruošinių realizacijos laiko sąnaudos. Ši nauda išryškėja prie tam tikrų duomenų.
8. Sukurtos programos pagalba galimas paprastas sudarytų algoritmų skaičiavimų rezultatų palyginimas ir jų darbo eigos (maršruto paieškos) stebėjimas.

LITERATŪRA

[1] Baker, J. E. Adaptive selection methods for genetic algorithms. Proc. 1st Int. Conf. On Genetic Algorithms and their Applications, Lawrence Erlbaum Associates, Hillsdale, NJ, p. 101-111, 1985.

[2] Davis, L. Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, NY, 1991.

[3] Feo, T., Resende, M. Greedy Randomized Adaptive Search Procedures. Journal of global optimization. 6 (2). p. 109-133, 1995.

[4] Fogarty, T. C. Varying the probability of mutation in the genetic algorithm, Proc. 3rd Int. Conf on Genetic Algorithms and Their Applications, George Mason University, p. 104-109, 1989.

[5] Goldberg, D. E. Genetic Algorithms in Search, Optimizations and Machine Learning. – Addison-Wesley: Boston, MA, USA, 1989.

[6] Glover, F., Laguna, M. Tabu search. - Boston : Kluwer Academic Publishers, 2001.

[7] Glover, F. Future paths for integer programming and links to artificial intelligence, Computers and Operation Research, Vol.13, p. 533-549, 1986.

[8] Glover, F. Tabu Search I. ORSA Journal on Computing, Vol.1., p. 190-206, 1989.

[9] Glover, F. Tabu Search II. ORSA Journal on Computing, Vol.2., p. 14-32, 1990.

[10] Grefenstette, J. J. Optimisation of control parameters for genetic algorithms, IEEE Trans. On Systems, Man and Cybernetics, Vol.SMC-16, No.1, p. 122-128, 1986.

[11] Hansen, P. The steepest ascent mildest descent heuristic for combinatorial programming, Conf. On Numerical Methods in Combinatorial Optimisation, Capri, Italy, 1986.

[12] Helman, P., Moret, B., Shapiro, H. An exact characterization of greedystructures. *SIAM Journal of Discrete Mathematics*, p. 274-283, 1993.

[13] Holland, J. H. Adaptation in Natural and Artificial Systems. – Ann Arbor : University of Michigan Press, 1975.

[14] Kirkpatrick, S., Gelatt, C. D. Jr. Vecchi, M.P. Optimization by simulated annealing, Science, Vol.220, No. 4598, p. 671-680, 1983.

[15] Lenkevičius, A., Matickas, J. Kompiuterinė grafika. – K.: Technologija, 2002.

[16] Mavridou, T., Pardalos, P., Pitsoulis, L., Resende, M. A GRASP for the biquadratic assignment problem. *European Journal of Operations Research*, p. 613–621, 1998.

[17] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N. Teller, A. H., Teller, E. Equation of state calculations by fast computing machines, J. of Chem. Phys., Vol.21, No.6, p. 1087-1092, 1953.

[18] Mitsuo, G., Runwei, C. - Genetic algorithms and engineering design. - New York : John Wiley & Sons, 1997.

[19] Mockus, J. A set of exemples of global and discrete optimization 2. - Kluwer Academic Publishers. - Dordrecht-Boston-London, 2000. [žiūrėta 2004-03-15]. Prieiga per internetą: http://www.soften.ktu.lt/~mockus/docj/stud2.pdf.

[20] Mockus, J., Eddy, W., Reklaitis, G., Mockus, A., Mockus, L. Bayesian Heuristic Approach to Discrete and Global Optimization. - Kluwer Academic Publishers.- Dordrecht-Boston-London, 1997. [žiūrėta 2003-11-30]. Prieiga per internetą: http://www.soften.ktu.lt/~mockus/docj/book.pdf.

[21] Michalewicz, Z. Genetic Algorithms + Data Structes = Evolution Programs, - New York : Springer-Verlag, 1992.

[22] Nils, N. Problem - Solving Methods in Artificial Intelligence. McGraw-Hill, 1971.

[23] Osman, I. H., Metastrategy Simulated Annealing and Tabu Search Algorithms for Combinatorial Optimisation Problems, PhD Thesis, University of London, Imperial College, UK, 1991.

[24] Pham, D.T., Karaboga, D. Intelligent optimisation techniques : genetic algorithms, tabu search, simulated annealing and neural networks. - London : Springer, 2000.

[25] Plukas, K., Mačikėnas, E., Jarašiūnienė, B., Mikuckienė, I. Taikomoji diskrečioji matematika: vadovėlis. – K.: Technologija, 2001.

[26] Reeves, C.R. Modern Heuristic Techniques for Combinatorial Problems. – McGraw-Hill : Maidenhead, UK, 1995.

[27] Schaffer, J.D., Caruana, R. A., Eshelman, L. J., Das, R. A study of control parameters affecting on-line performance of genetic algorithms for function optimization, Proc. 3rd Int. Conf on Genetic Algorithms and Their Applications, George Mason University, p. 51-61, 1989.

[28] Whitely, D., Hanson, T. Optimising neural networks using faster, more accurate genetic search. Proc. 3rd Int. Conf. On Genetic Algorithms and their Applications, George Mason University, p. 370-374, 1989.

PRIEDAI

1 priedas. Duomenų failo pavyzdys

```
<?xml version='1.0' encoding='utf-8'?>
<schemeData name='Br4d.xml' dimemsion='4'>
<line>
    <x1>160</x1> <y1>120</y1> <x2>180</x2> <y2>80</y2> <curve>0</curve> <thick>nil</thick>
</line>
<line>
    <x1>100</x1> <y1>80</y1> <x2>160</x2> <y2>120</y2> <curve>0</curve> <thick>nil</thick>
</line>
<line>
    <x1>100</x1> <y1>80</y1> <x2>100</x2> <y2>40</y2> <curve>0</curve> <thick>nil</thick>
</line>
<line>
    <x1>40</x1> <y1>140</y1> <x2>40</x2> <y2>80</y2> <curve>0</curve> <thick>nil</thick>
</line>
</schemeData>
```

2 priedas. Rezultatų failo pavyzdys

```
<?xml version='1.0' encoding='utf-8'?>
<result
    name='Br4r.xml'
    dimension='4'
    routeLength='288'
    routeTime='31'
>
    <line>
        <x>180.0</x> <y>80.0</y> <state>true</state> <curve>0</curve> <thick>0</thick>
    </line>
    <line>
        <x>160.0</x> <y>120.0</y> <state>true</state> <curve>0.0</curve> <thick>0.0</thick>
    </line>
    <line>
        <x>100.0</x> <y>80.0</y> <state>true</state> <curve>0.0</curve> <thick>0.0</thick>
    </line>
    <line>
        <x>100.0</x> <y>40.0</y> <state>true</state> <curve>0.0</curve> <thick>0.0</thick>
    </line>
    <line>
        <x>40.0</x> <y>80.0</y> <state>false</state> <curve>0</curve> <thick>0</thick>
    </line>
    <line>
        <x>40.0</x> <y>140.0</y> <state>true</state> <curve>0.0</curve> <thick>0.0</thick>
    </line>
</result>
```

```
(defun C:get()
(setq sar (ssget "X" (list (cons 0 "LINE"))))
(setq failoVardas (getstring "Iveskite failo varda: "))
(setq failas (open (strcat "C:\\" failoVardas) "w"))
(cond ((null sar)
		(setq ilgis 0)
		(print "Nera liniju !!!")
	)
	(( = (sslength sar) 1)
		(setq ilgis 0)
		(print "Turi buti maziausiai dvi linijos !")
	)
	(t
		(setq ilgis (sslength sar))
		(princ "<?xml version='1.0' encoding='utf-8'?>" failas)(princ "\n" failas)
		(princ "<schemeData name='" failas)
		(princ failoVardas failas)(princ "'" failas)
		(princ " dimemsion='" failas)
		(princ ilgis failas)(princ "'" failas)(princ ">" failas)(princ "\n" failas)
		(setq i 0)
		(while(< i ilgis)
			(setq vardas (ssname sar i))
			(setq elem (entget vardas))
			(setq x1y1 (cdr (assoc 10 elem)))
			(setq x2y2 (cdr (assoc 11 elem)))
			(setq x1 (car x1y1))
			(setq y1 (cadr x1y1))
			(setq x2 (car x2y2))
			(setq y2 (cadr x2y2))
			(setq sluoksnis (cdr (assoc 8 elem)))
			(setq storis (cdr (assoc 39 elem)))
			(princ "<line>" failas)(princ "\n" failas)
			(princ "\t" failas)
			(princ "<x1>" failas)(princ x1 failas)(princ "</x1> " failas)
			(princ "<y1>" failas)(princ y1 failas)(princ "</y1> " failas)
			(princ "<x2>" failas)(princ x2 failas)(princ "</x2> " failas)
			(princ "<y2>" failas)(princ y2 failas)(princ "</y2> " failas)
			(princ "<curve>" failas)(princ 0 failas)(princ "</curve> " failas)
			(princ "<thick>" failas)(princ storis failas)(princ "</thick>" failas)(princ "\n"
failas)
			(princ "</line>" failas)(princ "\n" failas)
			(setq i (+ i 1))
		)
		(princ "</schemeData>" failas)
	)

);COND
(close failas)
)
```

JavaDoc – standartizuoti pagalbos failai, aprašantys klases, paketus ir interfeisus. Jie yra generuojami pagal šablonus, klasių kūrėjui pateikus visą reikalingą informaciją.

Kiekvieną programos klasę atitinka atskiras *html* failas, kuriame yra visa su klasės kintamaisiais, metodais, konstruktoriais susijusi informacija. Pavyzdys gali būti klasę **Data** aprašantis failas *Data.html*. Jo vidinės struktūros fragmentai pateikiami žemiau.

Antraštė:

SchemeData

Class Data

```
java.lang.Object
  |
  +--SchemeData.Data
```

All Implemented Interfaces:

java.lang.Cloneable

public class **Data**
extends java.lang.Object

implements java.lang.Cloneable

Nuskaito duomenis iš duomenų failo. Suformuoja briaunų ir viršūnių sąrašus.

Kintamieji:

listLine

Schemos briaunų sąrašas

Konstruktorius:

Data()

Data objekto konstruktorius

Metodas:

isSelected(double x,double y)

Tikrina, ar viršūnė su koordinatėmis (x,y) jau yra sąraše

Kadangi pagalbos failai yra *html* formate, tai spragtelėjus pelės klavišą ties metodo, kintamojo ar konstruktoriaus pavadinimu, pateikiama detalesnė informacija. Pavyzdžiui, spragtelėjus ties **isSelected** metodo vardu, bus gautas detalesnis aprašymas:

isSelected

```
public boolean isSelected(double x,double y)
```

Tikrina, ar viršūnė su koordinatėmis (x,y) jau yra sąraše

Parameters:

x - Viršūnės x koordinatė

y - Viršūnės y koordinatė

Returns:

true, jei viršūnė yra sąraše, false - priešingu atveju

Tokia informacija bus pateikta apie kiekvieną konstruktorių ir *public* modifikatoriumi pažymėtą metodą ar kintamąjį.

5 priedas. Vartotojo vadovas

Programos sudėtis. Programa pateikiama suspausta į vieną failą *Magistrinis.jar*. Išskleidus šį failą automatiškai sukuriama katalogų ir failų struktūra. Šios struktūros keisti negalima. Plėtinys *jar* reiškia, jog tai specializuotas kompaktiškas Java failas, kuriame suarchyvuotos ir patalpintos visos programos klasės bei *manifest.mf* failas, o taip programos klasių dokumentacija JavaDoc pavidalu.

Bendri reikalavimai. Norint pasinaudoti programa, reikia, kad kompiuteryje būtų įdiegta JDK 1.4.2.* ar aukštesnė versija. Programos išskleidimui kietajame diske reikia apie 200 KB laisvos vietos. Sistemos minimalūs reikalavimai - Pentium 166 MHz ar greitesnis procesorius. Kadangi programa turi grafinę vartotojo sąsają (GUI), mažiausiai reikės 32 MB operatyvinės atminties. Dokumentacijos failų peržiūrai reikalinga bet kokia interneto naršyklė arba teksto redaktorius, suprantantis *html* formato failus.

Programos paleidimas. Programos paleidimas gali būti valdomas iš komandinės eilutės. Norint pradėti darbą su programa, nebūtina išarchyvuoti *Magistrinis.jar* failo. Esant kataloge, kuriame yra *Magistrinis.jar* failas, programa paleidžiama komandinėje eilutėje, įvedus:

```
java -jar Magistrinis.jar
```

Norint išarchyvuoti visą *jar* failo turinį, komandinėje eilutėje reikia įvesti:

```
jar xf Magistrinis.jar
```

Įvykdžius šią komandą bus išarchyvuoti programos **.class* failai, standartizuoti klasių dokumentacijos failai ir sukurta archyve esanti katalogų struktūra. Jeigu reikalingi tik klasių dokumentacijos failai *html* formatu, esantys API kataloge, komandinėje eilutėje reikia įvesti:

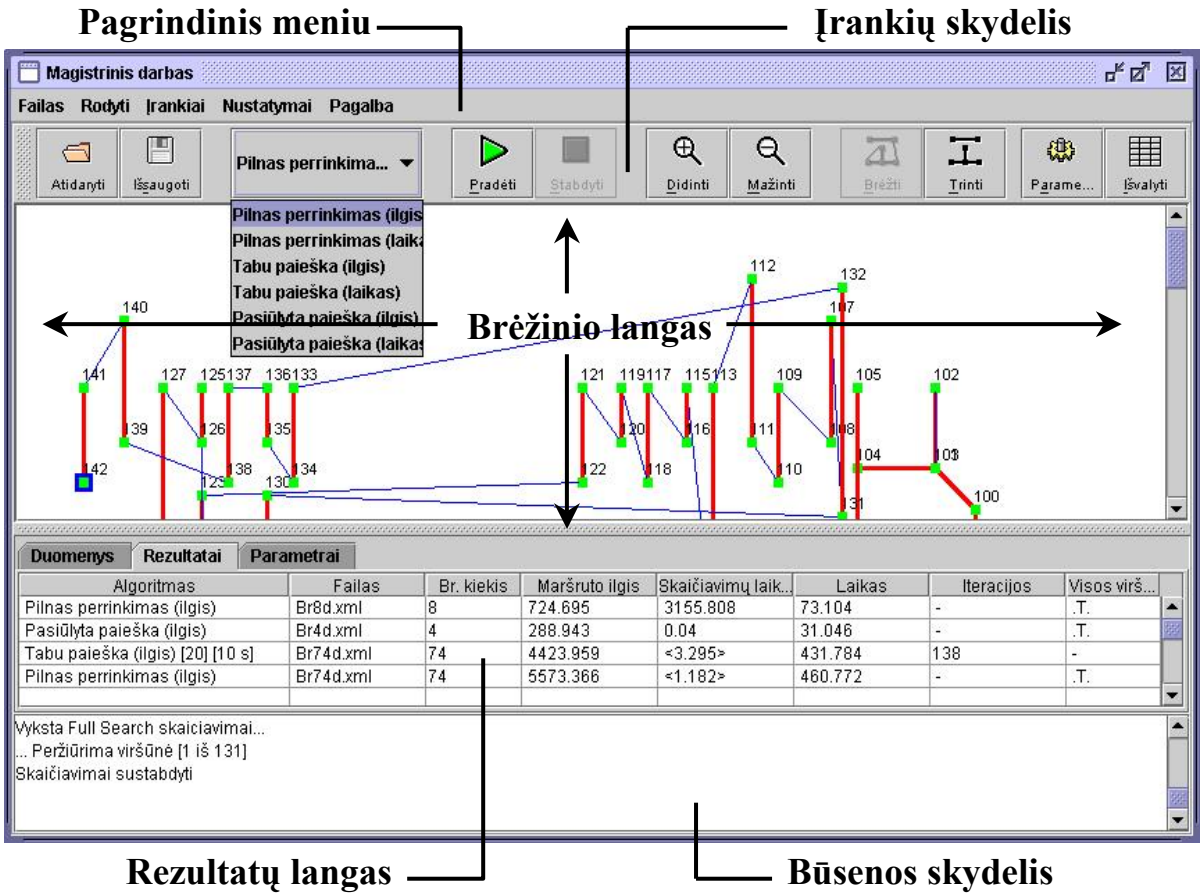
```
jar xf Magistrinis.jar API
```

Programos valdymas labai paprastas, visus veiksmus galima atlikti pelės ar klaviatūros

pagalba.

Pagrindinis langas. Paleidus programą pasirodys pagrindinis langas, kuriame dalis meniu punktų ir komandų mygtukų bus neaktyvūs. Pagrindinis programos langas sudarytas iš kelių dalių (žr. 5.1 pav.):

- pagrindinio meniu;
- įrankių skydelio;
- brėžinio lango;
- skaičiavimų rezultatų lentelės lango;
- būsenos skydelio.



5.1 pav. Pagrindinis langas

Pagrindinis meniu. Pagrindinio lango viršuje esantis meniu leidžia pasirinkti norimą komandą (žr. 5.2 pav.).

5.2 pav. Pagrindinis meniu

- **Failas** – pasirinkti naują duomenų failą, išsaugoti rezultatus ir baigti darbą su programa.
- **Rodyti** – padidinti ar sumažinti brėžinį, paslėpti ar vėl parodyti įrankių ir būsenos skydelius.
- **Įrankiai** – sustabdyti ar pradėti vykdyti algoritmų skaičiavimus, brėžti ar ištrinti surastą maršrutą.
- **Nustatymai** – pakeisti parametrus, išvalyti rezultatų lentelę.
- **Pagalba** – parodyti informaciją apie programos autorių.

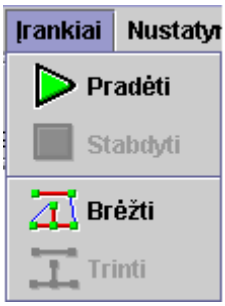
5.3 pav. Meniu punktas „Failas“

- **Atidaryti** – atidaromas duomenų failas iš dialoginio failų pasirinkimo lango. Yra galimybė atlikti failų filtravimą pagal plėtinį **.xml*. Klaviatūros komanda - *Ctrl+O*.
- **Išsaugoti** – išsaugomi skaičiavimų rezultatai faile, kurio vardą įveda vartotojas. Rezultatų failo katalogas pasirenkamas iš dialoginio failų išsaugojimo lango. Klaviatūros komanda - *Ctrl+S*.
- **Baigti** – baigiamas darbas su programa. Klaviatūros komanda - *Ctrl+X*.

5.4 pav. Meniu punktas „Rodyti“

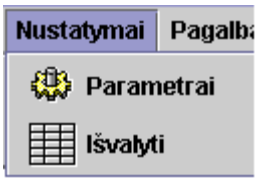
- **Didinti** – padidinamas brėžinys. Tą patį daro įrankių skydelio mygtukas „*Didinti*“.
- **Mažinti** – sumažinamas brėžinys. Tą patį daro įrankių skydelio mygtukas „*Mažinti*“.
- **Įrankių skydelis** – rodomas arba paslepiamas įrankių skydelis.

- **Būsenos skydelis** – rodomas arba paslepiamas būsenos skydelis.



5.5 pav. Meniu punktas „Įrankiai“

- **Pradėti** – pradedami pasirinkto algoritmo skaičiavimai. Tą patį daro įrankių skydelio mygtukas „*Pradėti*“.
- **Stabdyti** – sustabdomi pasirinkto algoritmo skaičiavimai. Tą patį daro įrankių skydelio mygtukas „*Stabdyti*“.
- **Brėžti** – nubrėžiamas surastas maršrutas. Tą patį daro įrankių skydelio mygtukas „*Brėžti*“.
- **Trinti** – ištrinamas surastas maršrutas. Tą patį daro įrankių skydelio mygtukas „*Trinti*“.



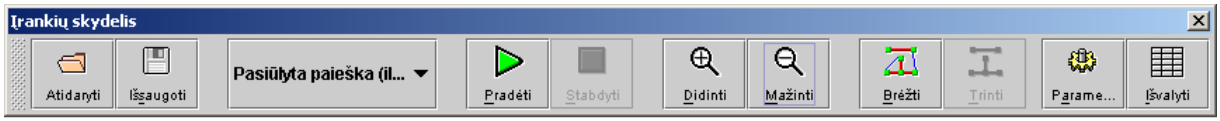
5.6 pav. Meniu punktas „Nustatymai“

- **Parametrai** – iškviečiamas programos parametrų keitimo langas. Tą patį daro įrankių skydelio mygtukas „*Parametrai*“.
- **Išvalyti** – išvaloma rezultatų lentelė. Tą patį daro įrankių skydelio mygtukas „*Išvalyti*“.

5.7 pav. Meniu punktas „Pagalba“

- **Apie** – parodomas pranešimas apie programos autorių.

Įrankių ir būsenos skydelis. Vartotojo patogumui dažniausiai naudojamos komandos iškviečiamos paspaudus mygtukus, esančius įrankių skydelyje (žr. 5.8 pav.).

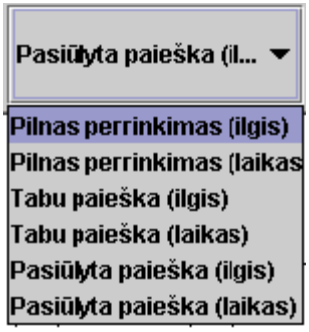


5.8 pav. Įrankių skydelis

- **Atidaryti** – atidaromas duomenų failas iš dialoginio failų pasirinkimo lango. Tą patį daro meniu punktas „*Atidaryti*“.
- **Išsaugoti** – išsaugomi skaičiavimų rezultatai faile, kurio vardą įveda vartotojas. Rezultatų failo katalogas pasirenkamas iš dialoginio failų išsaugojimo lango. Tą patį daro meniu punktas „*Išsaugoti*“.
- **Pradėti** – pradedami pasirinkto algoritmo skaičiavimai. Tą patį daro meniu punktas „*Pradėti*“.
- **Stabdyti** – sustabdomi pasirinkto algoritmo skaičiavimai. Tą patį daro meniu punktas „*Stabdyti*“.
- **Didinti** – padidinamas brėžinys. Tą patį daro meniu punktas „*Didinti*“.
- **Mažinti** – sumažinamas brėžinys. Tą patį daro meniu punktas „*Mažinti*“.
- **Brėžti** – nubrėžiamas surastas maršrutas. Tą patį daro meniu punktas „*Brėžti*“.
- **Trinti** – ištrinamas nubrėžtas maršrutas. Tą patį daro meniu punktas „*Trinti*“.
- **Parametrai** – iškviečiamas programos parametrų keitimo langas. Tą patį daro meniu punktas „*Parametrai*“.
- **Išvalyti** – išvaloma rezultatų lentelę. Tą patį daro meniu punktas „*Išvalyti*“.

Šias komandas galima įvykdyti klaviatūros pagalba. Reikia paspausti pageidaujamos komandos pavadinimo pabrauktą raidę, tuo pat metu laikant paspaustą *Alt* klavišą.

Iš iškrentančio sąrašo galima pasirinkti norimą paieškos algoritmą (žr. 5.9 pav.).

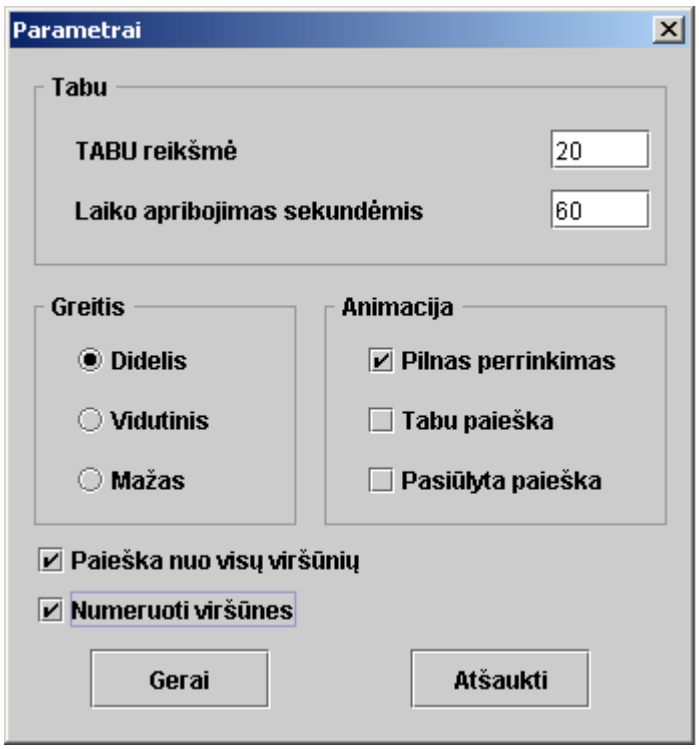


5.9 pav. Algoritmų sąrašas

Būsenos skydelis yra skirtas padėti vartotojui sekti atliekamus ar atliktus veiksmus (žr. 5.10 pav.).

5.10 pav. Būsenos skydelio pavyzdys

Parametrų langas. Programos parametrus galima keisti pasirinkus „*Nustatymai“* ir „*Parametrai“* iš pagrindinio meniu arba paspaudus mygtuką „*Parametrai“* iš įrankių skydelio. Keičiant parametrus, kuomet skaičiavimai nevyksta, matomas pilnai aktyvus dialoginis parametrų langas (žr. 5.11 pav.).



5.11 pav. Parametrų nustatymo langas

Parametrų langas suskirstytas į dalis:

- **Tabu** – prašoma įvesti tabu reikšmę ir skaičiavimų laiko apribojimą. Šie parametrai skirti tabu paieškos algoritmui.
- **Greitis** – pasirenkamas norimas animacijos greitis („*Didelis*“, „*Vidutinis*“, „*Mažas*“). Galioja visiems algoritmams.

- **Animacija** – varnele pažymimi algoritmai, kurių darbą norima animuoti. Galioja visiems algoritmams.

Varnelė ties užrašu "*Paieška nuo visų viršūnių"* nurodo, kad maršruto paieška bus vykdoma nuo visų viršūnių. Jei varnelė nuimta, paieška bus vykdoma nuo atsitiktinai parinktos vienos viršūnės. Šis pasirinkimas negalioja tabu paieškai.

Varnelė ties užrašu "*Numeruoti viršūnes"* nurodo, kad suradus maršrutą bus numeruojamos viršūnės. Viršūnės numeruojamos pagal viršūnių lankymo tvarką.

Parametrų lango mygtukai:

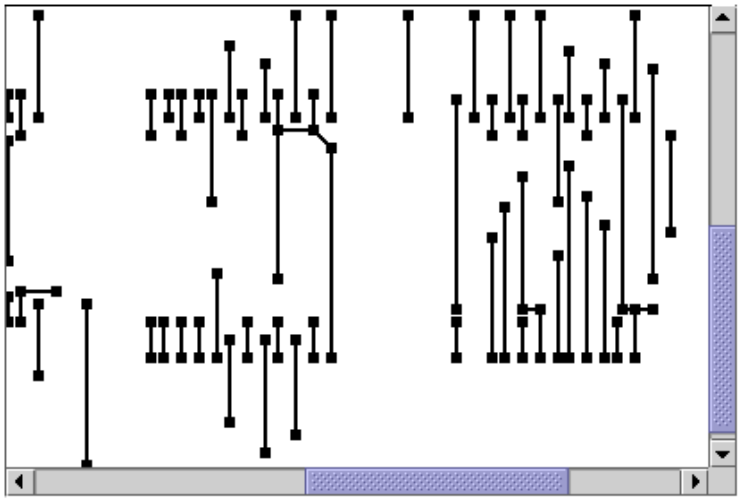
- **Gerai** – pakeitimai išsaugomi.
- **Atšaukti** – pakeitimų atsisakoma.

Keičiant parametrus, kuomet vyksta skaičiavimai, matomas dalinai aktyvus dialoginis parametrų langas (žr. 5.12 pav.).

5.12 pav. Dalinai aktyvus parametrų nustatymo langas

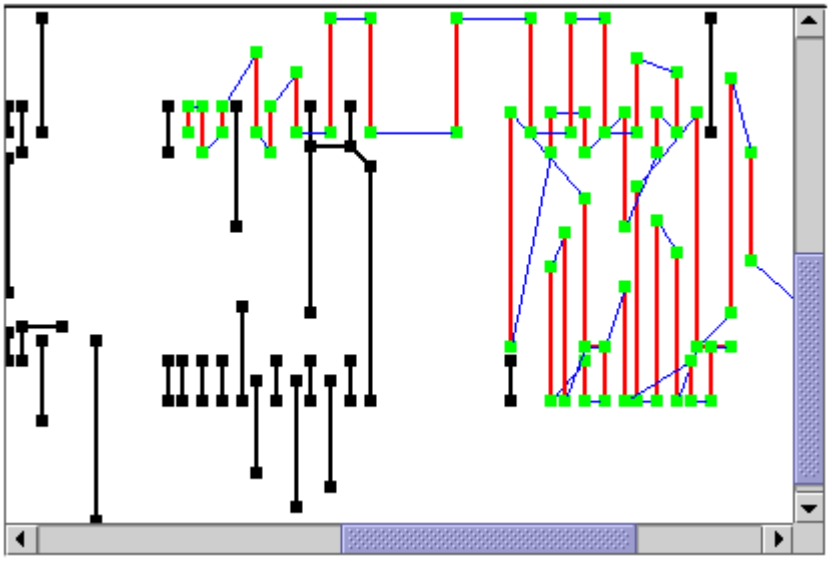
Šiuo atveju galima keisti tik animacijos greitį ir viršūnių numeracijos požymį.

Brėžinio langas. Pagrindiniame meniu pasirinkus „*Failas"* – „*Atidaryti"* ir atidarius duomenų failą, brėžinio lange nubraižoma schema (žr. 5.13 pav.). Jei duomenų failas netinkamas, apie tai bus pranešta būsenos skydelyje. Brėžinį galima padidinti ir sumažinti mygtukų ar meniu punktų „*Didinti"* ir „*Mažinti"* pagalba.



5.13 pav. Brėžinio pavyzdys

Jei parametrų lango animacijos dalyje ties algoritmo vardu uždėta varnelė, tai brėžinio lange galima stebėti šio algoritmo darbo eigą (žr. 5.14 pav.). Raudonos linijos žymį jau peržiūrėtas brėžinio briaunas, o mėlynos – kaip šios briaunos sujungtos tarpusavyje.



5.14 pav. Brėžinio pavyzdys, programai vykdant skaičiavimus

Rezultatų langas. Rezultatų lange pateikiami algoritmų darbo rezultatai. Algoritmui baigus darbą arba vartotojui nutraukus skaičiavimus, rezultatai išvedami į šią lentelę (žr. 5.15 pav.).

Duomenys | Rezultatai | Parametrai

Algoritmas	Failas	Br. kiekis	Maršruto ilgis	Skaičiavimų laik...	Proceso trukmė	Iteracijos	Visos viršū...
Pasiūlyta paieška (ilgis)	Br100d.xml	100	5566.296	0.631	569.603	-	.T.
Pasiūlyta paieška (ilgis)	Br100d.xml	100	5860.746	0.02	577.492	-	.N.
Tabu paieška (ilgis) [2] [10 s]	Br10d.xml	10	399.442	(<0.701>)	75.988	8	-
Pilnas perrinkimas (ilgis)	Br10d.xml	10	391.224	<0.2>	75.823	-	.N.
Pilnas perrinkimas (ilgis)	Br74d.xml	74	5671.164	<0.311>	463.728	-	.N.
Pilnas perrinkimas (ilgis)	Br74d.xml	74	---	(---)	---	-	.N.

5.15 pav. Rezultatų lentelė

Lentelę sudaro aštuoni stulpeliai:

- **Algoritmas** – pasirinkto algoritmo vardas.
- **Failo vardas** – duomenų failo vardas.
- **Br. kiekis** – briaunų kiekis brėžinyje.
- **Maršruto ilgis** – surasta maršruto ilgis.
- **Skaičiavimų laikas** – pasirinkto algoritmo darbo laikas.
- **Proceso trukmė** – surasta proceso trukmė.
- **Iteracijos –** tabu paieškos iteracijų skaičius.
- **Visos viršūnės** – požymis, ar paieška vykdoma nuo visų viršūnių.

Žymėjimai stulpelyje *"Algoritmas"*:

[…] – tabu reikšmė (pvz. [2]).

[… s] – tabu paieškos laiko apribojimas sekundėmis (pvz. [10 s]).

Žymėjimai stulpelyje *"Skaičiavimų laikas"*:

(…) – skaičiavimai buvo animuota.

<…> – skaičiavimai buvo nutraukti, bet bent vienas maršrutas buvo surastas.

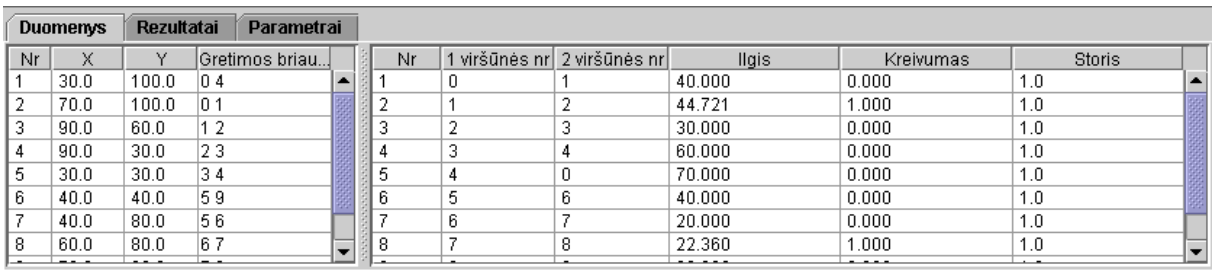
--- – skaičiavimai buvo nutraukti, nerastas nei vienas maršrutas.

Žymėjimai stulpelyje *"Visos viršūnės"*:

.T. – paieška įvykdyta nuo visų viršūnių.

.N. – paieška įvykdyta nuo atsitikinės viršūnės

Duomenų langas. Duomenų lange pateikiama ruošinį aprašanti informacija. Vartotojui atidarius duomenų failą, ši informacija išvedami į duomenų lentelę (žr. 5.16 pav.).



5.16 pav. Duomenų langas

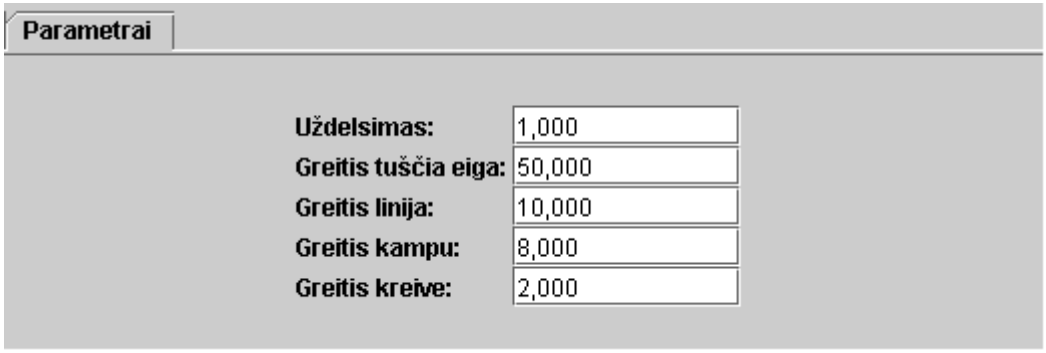
Kairėje lango dalyje pateikiama ruošinio viršūnes aprašanti informacija:

- **Nr** – viršūnės indeksas;
- **X** – viršūnės x koordinatė;
- **Y** – viršūnės y koordinatė;
- **Gretimos briaunos** – viršūnei gretimų briaunų indeksai.

Dešinėje lango dalyje pateikiama ruošinio briaunas aprašanti informacija:

- **Nr** – briaunos indeksas;
- **1 viršūnės nr** – briaunos pirmosios viršūnės indeksas;
- **2 viršūnės nr** – briaunos antrosios viršūnės indeksas;
- **Ilgis** – briaunos ilgis;
- **Kreivumas** – briaunos kreivumas;
- **Storis** – briaunos storis.

Parametrų langas. Parametrų lange pateikiami parametrai, skirti proceso trukmei optimizuoti. Vartotojas šiuos parametrus gali keisti (žr. 5.17 pav.).



5.17 pav. Parametrų langas

- **Uždelsimas** – uždelsimo dydis viršūnėje;

- **Greitis tuščia eiga** – įrenginio greitis judant tuščiąja eiga;
- **Greitis linija** – įrenginio greitis tiese horizontalia ar vertikalia linija;
- **Greitis kampu** – įrenginio greitis tiese nehorizontalia ir nevertikalia linija;
- **Greitis kreive** – įrenginio greitis kreive.

Instrukcijos vartotojui

1. **Programos paleidimas:**

   Esant kataloge, kuriame yra *Magistrinis.jar* failas, komandinėje eilutėje reikia įvesti:

   ```
   java -jar Magistrinis.jar
   ```

2. **Duomenų failo atidarymas:**

   Iš įrankių skydelio pasirinkti mygtuką „*Atidaryti“* arba iš pagrindinio meniu pasirinkti „*Failas“* ir „*Atidaryti“,* arba klaviatūros kombinacija *CTRL+O*.

3. **Schemos didinimas ekrane:**

   Iš įrankių skydelio pasirinkti mygtuką „*Didinti“* arba iš pagrindinio meniu pasirinkti „*Rodyti“* ir „*Didinti“,* arba klaviatūros kombinacija *Alt+D*.

4. **Schemos mažinimas ekrane:**

   Iš įrankių skydelio pasirinkti mygtuką „*Mažinti“* arba iš pagrindinio meniu pasirinkti „*Rodyti“* ir „*Mažinti“,* arba klaviatūros kombinacija *Alt+M*.

5. **Algoritmo pasirinkimas:**

   Iš įrankių skydelio iškrentančio sąrašo pasirinkti vieną iš galimų algoritmų: „*Pilnas perrinkimas (ilgis)“,* „*Tabu paieška (ilgis)“*, „*Pasiūlyta paieška (ilgis)“,* „*Pilnas perrinkimas (lakas)“,* „*Tabu paieška (laikas)“* arba „*Pasiūlyta paieška (laikas)“*.

6. **Pasirinkto algoritmo vykdymas:**

   Iš įrankių skydelio pasirinkti mygtuką „*Pradėti“* arba iš pagrindinio meniu pasirinkti „*Įrankiai“* ir „*Pradėti“,* arba klaviatūros kombinacija *Alt+P*.

7. **Algoritmo darbo sustabdymas:**

   Iš įrankių skydelio pasirinkti mygtuką „*Stabdyti“* arba iš pagrindinio meniu pasirinkti „*Įrankiai“* ir „*Stabdyti“,* arba klaviatūros kombinacija *Alt+S*.

8. **Surasto maršruto nubraižymas ekrane:**

   Iš įrankių skydelio pasirinkti mygtuką „*Brėžti“* arba iš pagrindinio meniu pasirinkti „*Įrankiai“* ir „ *Brėžti “,* arba klaviatūros kombinacija *Alt+B*.

9. **Nubraižyto maršruto ekrane ištrynimas:**

   Iš įrankių skydelio pasirinkti mygtuką „*Trinti“* arba iš pagrindinio meniu pasirinkti „*Įrankiai“* ir „ *Trinti “,* arba klaviatūros kombinacija *Alt+T*.

10. **Surasto maršruto išsaugojimas rezultatų faile:**

Iš įrankių skydelio pasirinkti mygtuką „*Išsaugoti“* arba iš pagrindinio meniu pasirinkti „*Failas“* ir „*Išsaugoti“,* arba klaviatūros kombinacija *CTRL+S.*

11. Programos parametrų pakeitimas:

Iš pagrindinio meniu pasirinkti „*Nustatymai“* ir „*Parametrai“* arba iš įrankių skydelio mygtuką „*Parametrai“,* arba klaviatūros kombinacija *CTRL+A*.

12. Baigti darbą su programa:

Iš pagrindinio meniu pasirinkti „*Failas“* ir „*Baigti“* arba klaviatūros kombinacija *CTRL+X.*

6 priedas. Kompaktinė plokštelė