VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Magistro baigiamasis darbas

GENETINĖS PAIEŠKOS STRATEGIJŲ TYRIMAS

(Investigation of genetic search strategies)

Atliko: Vaiva Devėnaitė

(parašas)

Darbo vadovas: lekt. V. Dičiūnas

(parašas)

Recenzentas: a. A. Janeliūnas

(parašas)

Vilnius
2006

SUMMARY

The use of genetic algorithms considerably increases. In some research works GA‘s are investigated to optimize graph problems. There are many different strategies for GA optimization. Unfortunately, there are no investigations if a strategy, suitable for a particular graph problem, will be useful solving other graph problems.

In this work I originated, described and developed some GA learning strategy elements. Also I developed some that are available in other research works. These elements are: generation of initial population, selection of individuals, mutation, crossover and some other parameters. All possible strategies (about 300) are tested in this work for three graph problems: shortest path, longest path and traveling salesman problem. Results are summarized and described.

TURINYS

1. ĮVADAS 5

2. LITERATŪROS APŽVALGA 7

2.1. KODAVIMAS 7

2.2. TINKAMUMO FUNKCIJA 11

2.3. SEKANČIOS KARTOS GENERAVIMAS 12

2.3.1. Atranka ir dauginimasis 12

2.3.2. Kryžminimas 16

2.3.3. Mutacija 20

2.4. TESTŲ REZULTATAI 22

2.5. LITERATŪROS APŽVALGOS IŠVADOS 26

3. GRAFŲ UŽDAVINIAI IR TYRIMO DUOMENŲ STRUKTŪROS 27

4. KOMBINATORINIŲ ALGORITMŲ TAIKYMAS GRAFŲ UŽDAVINIAMS 30

4.1.1. Godus algoritmas 30

4.1.2. Atsitiktinės paieškos algoritmas 31

4.1.3. Pilno perrinkimo algoritmas 32

5. GENETINIŲ ALGORITMŲ TAIKYMAS GRAFŲ UŽDAVINIAMS 33

5.1. KODAVIMAS 34

5.2. TINKAMUMO REIKŠMIŲ APSKAIČIAVIMAS 35

5.3. ATRANKA 36

5.4. KRYŽMINIMAS 37

5.5. MUTACIJA 43

5.6. KARTŲ PAKEITIMAS 44

5.7. KITI GENETINIŲ ALGORITMŲ PARAMETRAI 44

6. GENETINĖS PAIEŠKOS STRATEGIJOS 46

6.1. STRATEGIJŲ VERTINIMAS 48

6.2. GENETINĖS PAIEŠKOS STRATEGIJŲ BANDYMAI 49

6.2.1. Trumpiausio kelio uždavinys *49*

6.2.2. Ilgiausio kelio uždavinys *53*

6.2.3. Keliaujančio pirklio uždavinys *55*

6.3. GENETINIO ALGORITMO PALYGINIMAS SU KOMBINATORINIAIS ALGORITMAIS 58

6.3.1. Pilno perrinkimo algoritmas *58*

6.3.2. Atsitiktinės paieškos ir godus algoritmai *60*

7. IŠVADOS IR REZULTATAI 61

LITERATŪROS SĄRAŠAS 62

PRIEDAI 66

I. PROGRAMINIAI MODULIAI 66

I.1. PRADINĖS POPULIACIJOS INICIALIZAVIMAS 66

I.2. ATRANKA 68

I.3. KRYŽMINIMAS 69

I.4. MUTACIJA 72

1. ĮVADAS

Genetinis algoritmas – tai kompiuterinis evoliucionuojančios dirbtinių individų populiacijos modelis. Pradinė genetinio algoritmo populiacija sudaroma iš aibės chromosomų, kurios dar vadinamos individais arba populiacijos nariais. Chromosomos sudarytos iš elementų, vadinamų genais. Kiekviena chromosoma pasižymi tam tikrais požymiais, pagal kuriuos būtų galima apskaičiuoti tinkamumo funkciją. Pagal tinkamumo funkcijos reikšmes, pritaikant atrankos metodus, išrenkami „sveikiausi“ (tinkamiausi) individai, kurie sudarys ateinančias populiacijos kartas. Ateinančios populiacijos kartos sudaromos esamos populiacijos individams pritaikius kryžminimo ir mutacijos operatorius.

Genetinius algoritmus būtų galima pavaizduoti tokiu pseudo-kodu:

```
sugeneruoti pradinę populiaciją;
while (sustojimo sąlygos netenkinamos)
  populiacijos individų apskaičiavimas;
  for kiekvienas individas populiacijoje
    Tinkamiausių individų išrinkimas;
    Kryžminimas;
    ...
    Mutacija;
  End for
End while
```

Genetinis algoritmas gali būti apmokomas daugeliu būdų, priklausomai nuo to, kaip parinkti jo parametrai. Nuo parametrų parinkimo priklauso ir algoritmo mokymosi rezultatai. GA mokymosi parametrai sudaro genetinių algoritmų mokymo (genetinės paieškos) strategiją, kuri gali būti apibrėžiama kaip ketvertas:

$$\Lambda = \{\Sigma, \Pi^M, \Phi, \Omega\}$$

Čia, Σ yra kodavimo formatas, Π^M yra M dydžio populiacija, Φ tinkamumo atrinkimo algoritmas, $\Omega = \{\varpi_1, \varpi_2, \ldots, \varpi_n\}$ – pertvarkymo operatorių (strategijos elementų) rinkinys.

Genetiniai algoritmai grafų uždaviniams plačiai taikomi užsienyje (literatūroje lietuvių kalba tai nėra paplitęs mokslinių tyrimų objektas). Dažniausiai literatūroje (pvz., [Pul06], [JSR01])stengiamasi optimizuoti vieną ar kitą grafų uždavinį, tačiau nepavyko rasti informacijos apie tai, kaip tos pačios genetinės paieškos strategijos gali būti pritaikomos kitiems grafų uždaviniams.

Atliekant šį magistro tezių darbą buvo realizuotas strategijos elementų (metodų) rinkinys, kuris naudojamas tiriant, kaip strategijos elementai įtakoja genetinio algoritmo mokymąsi. Darbe realizuoti šie strategijos elementai:

Pradinės populiacijos generavimas: *atsitiktinis permaišymas, atsitiktinis permaišymas su apribojimais, inicializavimas panaudojant kombinatorinius algoritmus*;

Atranka: *pagal tinkamumo funkciją, eksponentinis metodas, tikimybinė atranka*.

Mutacija: *atsitiktinė, skirtingų genų, sukeitimo, vieno taško*.

Kryžminimas: f*iksuojant pradines viršūnes, perstatos, vieno taško perkoduojant*.

Be šių strategijos elementų, taip pat yra nagrinėjami ir kiti genetinių algoritmų parametrai.

Magistro tezių tyrime atliktas strategijų tyrimas trims grafų uždaviniams: trumpiausio kelio paieškos, ilgiausio kelio paieškos bei keliaujančio pirklio uždaviniams. Jiems spręsti išanalizuotos visos galimos strategijų elementų kombinacijos – apie 300 skirtingų strategijų kiekvienam uždaviniui. Strategijų bandymų rezultatai lyginami tarpusavyje, bei su klasikiniais kombinatoriniais algoritmais.

Toliau šiame darbe pateikiama probleminės srities apžvalga, tyrimo aprašymas bandymų rezultatai ir išvados.

2. LITERATŪROS APŽVALGA

Išnagrinėtuose šaltiniuose išskiriami tokie pagrindiniai genetinio algoritmo etapai:

- Uždavinio užkodavimas;
- Tinkamumo reikšmių apskaičiavimas;
- Sekančios kartos generavimas
  - Atranka ir dauginimasis;
  - Kryžminimas;
  - Mutacija.

Tolesniuose darbo skyriuose apžvelgiami literatūroje pasiūlyti kiekvieno etapo tobulinimo metodai, strategijos.

2.1.Kodavimas

Kodavimo pasirinkimas genetiniam algoritmui gali parenkamas pagal daug faktorių. Pagrindinis jų – algoritmo taikymo sritis. Keletas kodavimo metodų aprašyta žemiau.

[Luk95]Visos paieškos procedūros turi turėti tam tikrą skaičių kintamųjų ar parametrų, kurie keičiami ieškant optimalaus sprendimo.

Kintamųjų skaičius priklauso nuo problemos dimensijos.

Genetinių metodų terminais, šių kintamųjų rinkinys atitinka geną (genetinį vektorių), kuris aprašo sprendimą ar individą. Šis genetinis vektorius atitinka chromosomą ir galimas genetinės abėcėlės reikšmes (A, T, C, G žmogaus chromosomoje), kurias gali įgyti kiekvienas genas.

Grėjaus (Gray) kodavimas

Atvaizdavimas Grėjaus kodu yra efektyvus sprendžiant realių parametrų optimizavimo problemas. Šis kodavimas taip garantuoja, kad sprendimas nebus prarastas susikaupus daugeliui

panašių sprendimų, kurie išvesti iš visiškai skirtingų struktūrų. Pavyzdžiui, jei tokia bitų seka *{10000000}* atitinka globalų optimumą, be Grėjaus kodo, paieška teiks pirmenybę sekoms, kurios turės daugiau vienetų išskyrus pirmąjį. Populiacija greičiausiai turės tokių struktūrų, kaip *{01111110}* arba *{01111100}*, kurios yra netoli optimumo, tačiau jis negali būti pasiektas taikant kryžminimo ar mutacijos operatorius. Grėjaus kodavimas peržengia šias kliūtis ir dažnai naudojamas sprendžiant parametrų optimizavimo problemas. Sąsają tarp dvejetainio kodavimo ir Grėjaus kodo galima atvaizduoti taip:

$$g_i = \begin{cases} b_i, if\ i = 1 \\ b_{i-1} \oplus b_i, if\ i > 1 \end{cases}$$

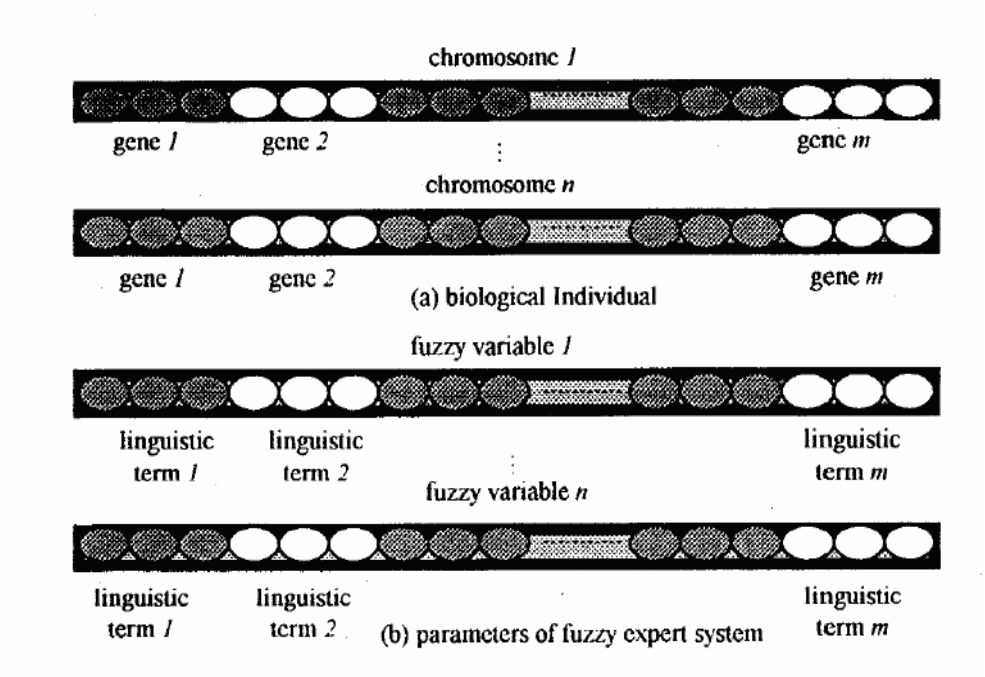
Atvirkštinė sąsaja:

$$b_i = \bigoplus_{j=1}^{i} g_j$$

Čia g_i žymi i-tąjį Grėjaus kodo bitą, o b_i –atitinkamą bitą dvejetainiame kodavime. $\oplus$ žymi sumą moduliu 2.

Parametrais paremtas kodavimo algoritmas. [CC96]

Tarkime turime n kintamųjų, kiekvienas kintamasis sudarytas m lingvistinių termų. Parametrais paremto kodavimo metodo struktūra pavaizduota paveiksle 1. Tikimasi, kad genai atvaizduoja priklausomumo lingvistiniam termui funkcijos parametrus. Taigi, chromosomos geno ilgis priklauso nuo priklausomumo funkcijos parametrų skaičiaus. Reikalaujama, kad chromosoma taip atitiktų visus kintamojo m lingvistinių termų. Taigi individo chromosomų ilgis yra m kartų didesnis už geno ilgį ir individas turės visus n kintamųjų parametrus



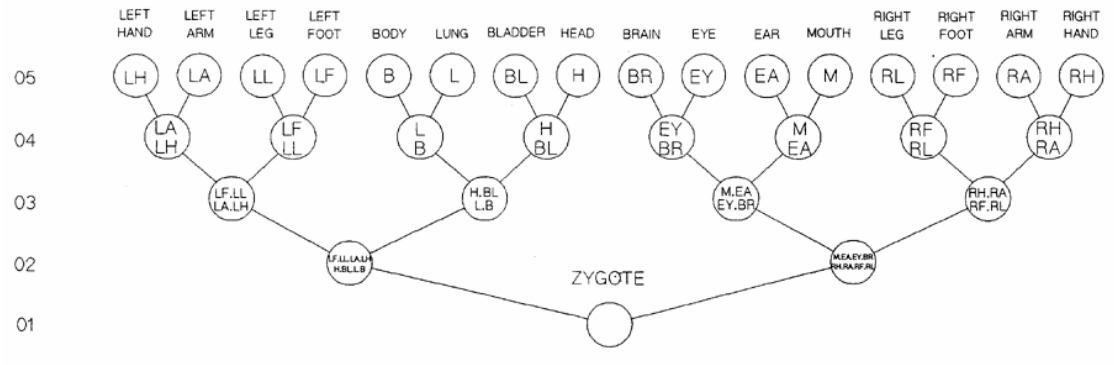
Paveikslas 1. Parametrais paremtas kodavimo algoritmas [CC96]

[Luk95] pateikiami tokie kodavimo metodai: *genetinis kodavimas*, *mazgų kodavimas*, *delta kodavimas*, *sunkus kodavimas*. Jie aprašomi žemiau.

Genetinis kodavimas

Kiekviena genetinio vektoriaus pozicija ar masyvas laikomas kaip nepriklausomas kintamasis. Kiekvienas kintamasis gali turėti reikšmę nepriklausomai nuo kitų kintamųjų reikšmių.

Genetinį kodavimą paaiškintų žemiau pateiktas paveikslas.



Paveikslas 2. [Dar00] Organizmo augimas dalinantis zigotai

Šio supaprastinto žmogaus vystymosi atvaizdavimo genetinis kodavimas žymimas viena ar dviem raidėmis:

```
 H = Head                BR = Brain
EY = Eye                 EA = Ear
 M = Mouth                B = Body
 L = Lung                BL = Bladder
LA = Left Arm            LH = Left Hand
LL = Left Leg            LF = Left Foot
RA = Right Arm           RH = Right Hand
RL = Right Leg           RF = Right Foot
```

Zigotai dalinantis kiekviena ląstelė turės *LH.LA.LL.LF.B.L.BL.H.BR.EY.EA.M.RL.RF.RA.RH.Z* genetinį kodą. Toliau dalinantis, kodas didės.

Mazgų (Node-Based) kodavimas

Genetinis vektorius vaizduoja maršrutą ar kelią. Ši schema būtina keliaujančio pirklio uždaviniui ir poaibių paieškai. Kiekvienas miestas ar atomas kelyje pasikartoti gali tik vieną kartą, taigi, vektoriaus elementai nėra nepriklausomi.

Realių skaičių kodavimas

Kintamųjų reikšmės saugomos genetiniame vektoriuje gali būti tiesiog realieji skaičiai.

Delta kodavimas

Šis kodavimas panašus į genų kodavimą, tačiau, kad būtų išvedamas galutinis sprendimas, kiekvieno sprendimo reikšmės yra įdedamos į „šabloninį“ sprendimą. Optimalus sprendimas rastas, kai randamas toks šablonas, kur geriausias galimas sprendimas susideda iš visų nulių. Čia visos genetinio vektoriaus reikšmės nepriklausomos

Sunkus (angl. messy) kodavimas

Šiame kodavime šablono struktūra taip pat naudojama, tačiau kiek kitu būdu. Yra naudojamas dvigubas indeksas: pirmasis skaičius reiškia poziciją genetiniame vektoriuje, antrasis – jo reikšmę. Šios, galimas daiktas, ne pilnos ir/ar perteklinės eilutės skaitomos pirmas-aptiktas-pirmas-panaudotas būdu. Sprendimo generavimui, visos trūkstamos reikšmės imamos iš šablono. Genetinio vektoriaus reikšmės yra nepriklausomos.

2.2. Tinkamumo funkcija

[CC96] Genetinis algoritmas gali tik optimizuoti tinkamumo funkcijos charakteristikas. Pagrindinis akcentas – minimizuoti gautų rezultatų vidutinį kvadratinį nuokrypį nuo siektų rezultatų. Šaltinyje pateikiama tokia parametrų tinkamumo funkcija:

$$F = \sum_i \left[M - \alpha E_i^m - \beta E_i^M \right]$$

Čia E_i^m: individo i kvadratinio nuokrypio tarp gautos reikšmės ir laukto rezultato vidurkis;

E_i^m: individo i maksimalus kvadratinis nuokrypis tarp gautos reikšmės ir laukto rezultato;

M: didelis skaičius;

α, β: svoriai, $\alpha + \beta = 1; 0 <= \alpha, \beta <= 1$.

Pagal tinkamumo funkcijos reikšmes, generuojamos sekančios kartos, t.y., atrenkami individai dauginimuisi ir palikuonių generavimui.

2.3. Sekančios kartos generavimas

[CC96]Populiacijos dydį, kryžminimo ir mutacijos apimtis galima priskirti vidinei genomų evoliucijai, dauginimasis ir tinkamumo parinkimo technika priskiriami išorinei atrankai. Sąveika tarp išorinės atrankos ir vidinės evoliucijos įtakoja genetinio algoritmo atsakymo paiešką ir optimizavimo procedūras.

Vidinė evoliucija stebima keičiant parametrus: populiacijos, kryžminimo, mutacijos dydžius.

Išorinė atranka – dauginimosi ir tinkamumo parinkimo metodai

2.3.1. Atranka ir dauginimasis

Dauginimosi metodas yra procesas kurio metu tėvinės struktūros yra išrenkamos naujų kartų formavimui, kurios vėliau gali pakeisti senų kartų atstovus.

Dažniausiai sutinkami dauginimosi metodai yra: kartų pakeitimas (angl. generational replacement), pastovios būsenos (angl. steady state). [Ric95]

Kartų pakeitimo algoritmas

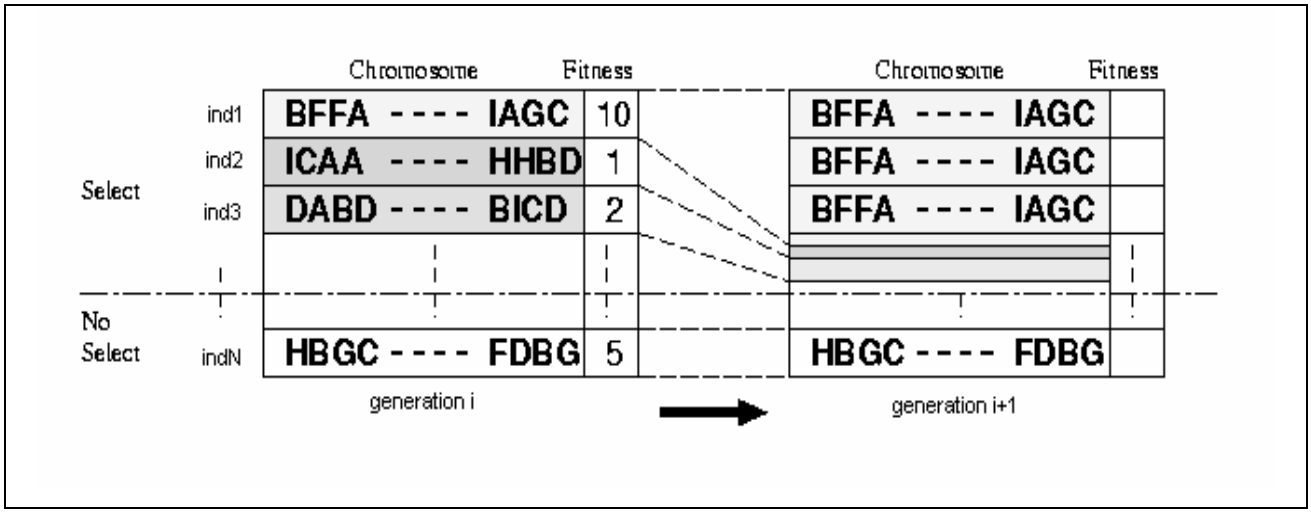
Šis algoritmas kiekvieną kartą generuojant naują generaciją, pakeičia visus populiacijos individus.

Pastovios būsenos algoritmas

Kiekvienoje kartoje pakeičia mažą populiacijos dalį.

[Ren00] Dauginimosi galimybė tiesiogiai priklauso nuo kiekvieno individo tinkamumo (sveikatos). Kuo individas tinkamumo funkcijos reikšmė didesnė, tuo didesnė tikimybė, kad individas išgyvens.

Atranka pagal tinkamumo funkciją (kai kuriuose šaltiniuose dar vadinama *elitine atranka*) pavaizduota žemiau esančiame paveiksle.



Paveikslas 3. [Car01]: atranka pagal tinkamumo funkciją

Čia susiduriama su dviem pagrindinėmis problemomis:

„Super individas“ parenkamas per daug dažnai, taigi visa populiacija linkusi konverguoti į šią genomą. Dėl šios priežasties stabdomas genetinio algoritmo progresas.

Progresuojant genetiniam algoritmui, tinkamumo reikšmių skirtumai sumažėja, taigi tikimybė pasirinkti geriausią individą tampa panaši į kitų populiacijos individų pasirinkimo tikimybę.

Tam, kad būtų galima išvengti tokių problemų, galima transformuoti tinkamumo reikšmes.

Kadravimo metodas

Iš kiekvieno individo tinkamumo reikšmės atimti blogiausio individo tinkamumo reikšmę. Tai leidžia sustiprinti stipriausią individą ir gauti artimą nuliui pasiskirstymą.

Eksponentinis metodas

Iš tinkamumo reikšmių ištraukiamos kvadratinės šaknys ir pridedamas vienetas. Tai leidžia sumažinti didžiausio individo įtaką.

Tiesinė transformacija

Kiekvienai tinkamumo reikšmei pritaikyti tiesinę transformaciją: $f' = a.f + b$. Taip sumažinama stipriausio individo įtaka

Tiesinis normalizavimas (kituose šaltiniuose dar vadinama įverties atranka)

Tinkamumas yra ištiesinamas. Pavyzdžiui, 10 individų populiacijoje pirmasis pažymimas 100, antrasis 90, 80, ... Paskutinis bus pažymimas 10. Atranka vykdoma pagal įvertį, o ne pagal tinkamumo reikšmes. Net jeigu skirtumai tarp individų labai dideli ar labai maži, skirtumai tarp dauginimosi tikimybės priklauso nuo individų sutvarkymo.

[Mar04] išskiriami tokie atrankos metodai:

Proporcionalaus tinkamumo atranka

Labiausiai tinkami individai atrenkami su didele tikimybe, bet nebūtinai.

Ruletės atranka

Proporcionalaus tinkamumo atrankos forma, kur tikimybė, kad individas bus atrinktas dydžiui, už kurį tinkamumas yra didesnis arba mažesnis už konkurentų tinkamumą. Paprastai tariant, šis metodas gali būti atvaizduotas kaip ruletė – kiekvienas individas gauna ruletės rato dalį. Kuo tinkamesnis individas, tuo didesnes dalis gauna. Kai pasukamas rutuliukas, tikimybė, kur jis sustos (kuris individas bus atrinktas) proporcinga individo užimamos dalie dydžiui.

Mastelio parinkimo atranka

Kai vidutinis populiacijos individų tinkamumo dydis išauga, atrankos stiprumas taip pat išauga ir tinkamumo funkcija tampa labiau diskriminuojanti. Šis metodas veiksmingas renkantis optimalų sprendimą vėliau, kai visi individai turi palyginti dideles tinkamumo reikšmes ir tik maži tinkamumo skirtumai skiria vieną individą nuo kito.

Konkurencinė atranka

Iš didesnės populiacijos išrenkami mažesni pogrupiai ir pogrupių nariai priklausantys tam pačiam pogrupiui konkuruoja tarpusavyje. Tik vienas individas iš pogrupio išrenkamas tolesniam dauginimuisi.

Hierarchinė atranka

Toje pačioje kartoje vykdoma keletas individų atrankų. Žemesnių lygių skaičiavimai yra greitesni ir mažiau diskriminuojantys, o tie individai, kurie išgyvena iki aukštesnių kartų diskriminuojami griežčiau. Šio metodo privalumas yra sutaupomas kompiuterio skaičiavimų laikas, naudojant greitesnius, mažiau atskiriančius skaičiavimus nedaug vilčių teikiančių individų eliminavimui ir griežtesnius skaičiavimus atliekant su tais, kurie išgyveno pradinį testą.

2.3.2. Kryžminimas

Kryžminimo operacija yra pasirinktų sekų suporavimas dauginimosi fazėje tam, kad suformuotume naujos kartos individus. Jungiant „gerus“ genus iš dviejų sekų, tikimasi, kad rezultatas bus tik geresnė seka. Žinoma, taip įvyksta ne visada, bet bendru atveju šiuo metodu gaunamos sėkmingos kartos.

Vieno taško (angl. Single-point) kryžminimas

Ko gero paprasčiausias kryžminimo metodas yra vieno taško kryžminimas[Sen00]. Jis generuoja vieną arba dvi vaikų sekas atsitiktinai parinkęs kryžminimo vietą tėvinėje sekoje. Vaikai (ar vaikas) tada yra sugeneruojami keičiant tėvinės chromosomos dalis, kaip pavaizduota paveiksle 4.

Dviejų taškų (angl. Double-point) kryžminimas

Tėvinėse chromosomose parenkami du atsitiktiniai taškai. Taip chromosoma sudalinama į tris dalis. Vaikai generuojami sukeičiant vidurines tėvinių chromosomų dalis.

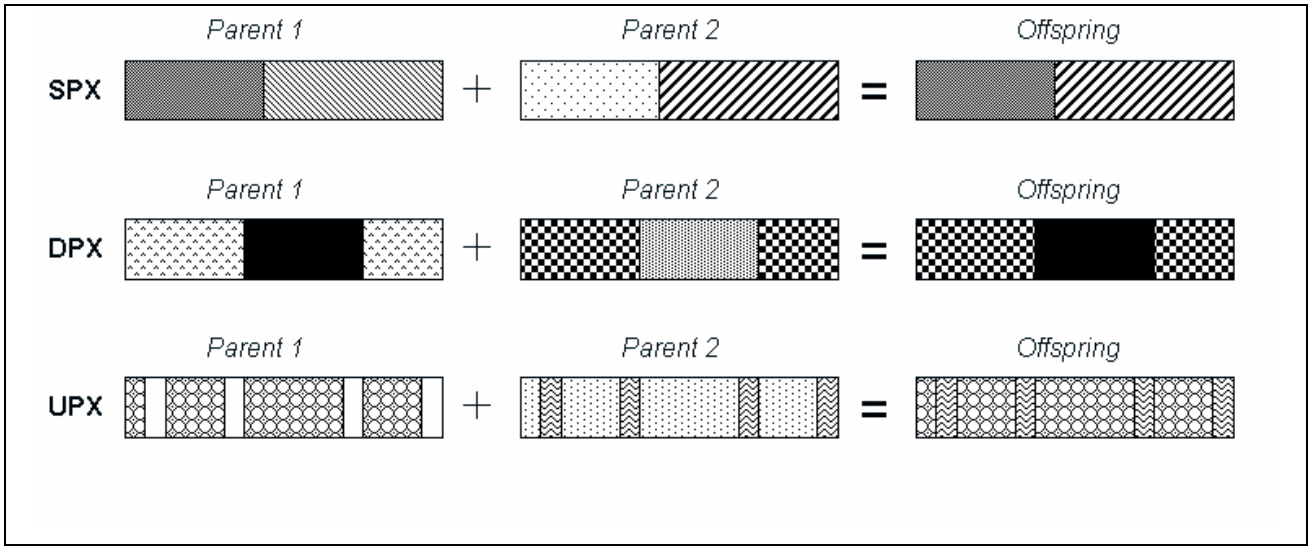
Daugelio taškų kryžminimas

Vykdomas analogiškai kaip ir dviejų taškų kryžminimas, tik atsitiktinai parenkama daugiau kryžminimo taškų.

Universalus kryžminimas

Universalaus kryžminimo operatorius pirmiausia atsitiktinai parenka bitų pozicijų seką genuose ir tada generuoja palikuonis keisdamas atitinkamus bitus tarp tėvinių chromosomų

Žemiau esančiame paveiksle pavaizduoti vieno taško (SPX), dviejų taškų (DPX) ir universalaus (UPX) kryžminimo operatoriai.



Paveikslas 4. [Neo98] kryžminimo operatoriai

Tolygusis kryžminimas

Kita kryžminimo strategija – tolygusis kryžminimas. Šia strategija galima gauti bet kokią dviejų tėvinių chromosomų kombinaciją. Naudojant tolygųjį kryžminimą, kiekvienoje pirmos tėvinės sekos pozicijoje parenkamas atsitiktinis skaičius tarp nulio ir begalybės. Jeigu atsitiktinis skaičius yra didesnis negu kryžminimo tikimybė, paimamas bitas iš antros tėvinės reikšmės, priešingu atveju, jis pasilieka toks pats, kaip pirmoji tėvinė reikšmė.

Schemos kryžminimas [Har00]

Paimama bitų seka (schema) ir pertvarkoma tokiu būdu: tie dviejų individų bitai, kur atitinkamose schemos pozicijose yra vienetas, pasilieka savo vietose, o tie, kur schemoje yra nulis, yra sukeičiami.

Pavyzdžiui:

Tarkime schema yra *1001*

Turime du individus:

0111 ir *1010*

Pritaikius šį kryžminimo metodą, gaunam tokius palikuonis: *0011* ir *1110*

Šiame pavyzdyje du viduriniai bitai buvo sukeisti, kadangi schemoje tose vietose buvo nuliai. Išoriniai bitai pasiliko tie patys.

[YMW03] pateikiami tokie kryžminimo operatoriai:

- Atsitiktinis kryžminimas;
- Pozicijos kryžminimas;
- Sudėtinis kryžminimas.

Atsitiktinis kryžminimas

Panašus į vieno taško kryžminimą. Iš atrinktų tėvų atsitiktinai paimami du individai ir kiekviename jų atsitiktinai pasirenkamas kryžminimo taškas. Abu tėvai apsikeičia jų genetinio kodo dalimis, priklausomai nuo kryžminimo taškų padėties. Sugeneruojami du vaikai, kiekvienas iš jų turi tėvinio kodo dalį.

Pozicijos kryžminimas

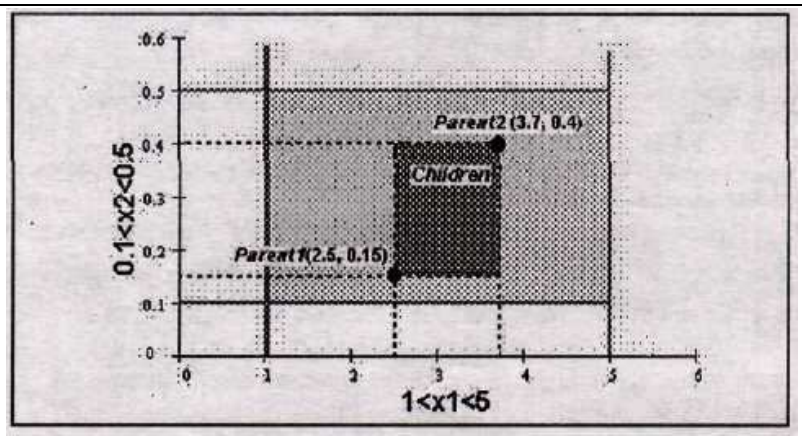
Atsitiktinai parenkamas kryžminimo taškas pirmojo tėvo genuose. Antrojo tėvo genome kryžminimo taškas apribojamas tomis vietomis, kurios atitinka pirmojo tėvo kryžminimo tašką. Dvi pozicijos sutampa, jeigu nuo jų tai pačiai operacijų sekai bus priskirtas pats genetinis kodas. Jeigu nepavyksta rasti tokio kryžminimo taško, kryžminimas nevykdomas ir abu tėvai įtraukiami į sekančią kartą.

Sudėtinis kryžminimas

Šis kryžminimas apima atsitiktinį ir pozicijos kryžminimus. Atsitiktinai parenkamas pirmasis kryžminimo taškas ir patikrinama, ar pozicijos kryžminimas gali būti atliktas, jei taip, atliekamas pozicijos kryžminimas, jei ne – parenkamas kitas kryžminimo taškas ir vykdomas atsitiktinis kryžminimas

Srities kryžminimas [Tak04]

Šį kryžminimo metodą paaiškina žemiau esantis paveikslėlis. Vaizduojamas dvimatis metodo pavyzdys. Tėvų reikšmėmis apribojamos galimų palikuonių reikšmės. Palikuoniai generuojami atsitiktinai. Ši kryžminimo strategija tinka gyvų organizmų kryžminimosi modeliavimui. Pavyzdžiui, jeigu vaiko motina blondinė, o tėvas brunetas, vaiko plaukų spalva gali būti visų atspalvių: nuo šviesios iki tamsios.



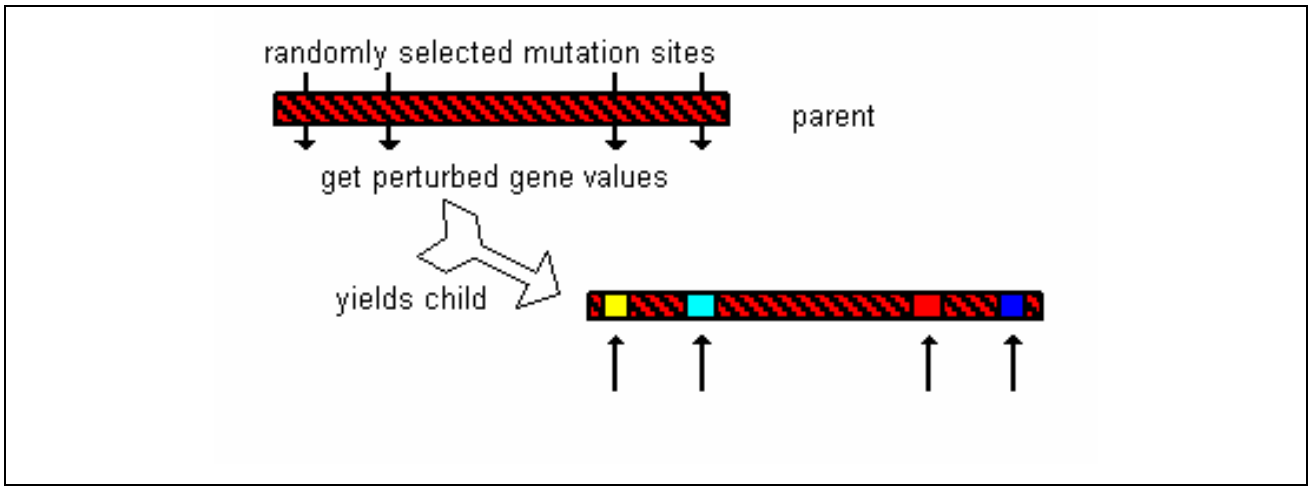
Paveikslas 5. [Tak04] Srities kryžminimas

2.3.3. Mutacija

Atsitiktinė mutacija

Mutacijos operatorius pasirinktiems sprendimams atlieka atsitiktinius pakeitimus su nurodyta tikimybe.

Atsitiktinė mutacija pavaizduota žemiau esančiame paveiksle.



Paveikslas 6. [HTC98] atsitiktinė mutacija

Keletas mutacijos žingsnio dydžio nustatymo realiai užkoduotiems genetiniams algoritmams variantų: išorinės taisyklės ir vidiniai faktoriai, tokie kaip prisitaikantis žingsnio dydis Gauso mutacijoje.

Išorinių taisyklių pavyzdys galėtų būti „*1/5 Rechenberg*“ taisyklė. Ši taisyklė skirta konvergavimo pagreitinimui Ji gali būti suformuluojama taip: Sėkmingų mutacijų santykis φ su visomis mutacijomis turėtų būti 1/5. Jeigu φ yra didesnis, reikia padidinti mutacijos dispersiją. Priešingu atveju, dispersiją mažinti.

Iš kitos pusės, bitų karkasas kryžminimo ir mutacijos kontroliavimui, paremtas induktyviu postūmiu turi patvirtintų pliusų evoliucionavimui, dažniausiai: klaidų atpažinimas ir uždraudimas (kryžminimo ir mutacijos pritaikymas palikuoniams), kad jie neišgyventų kitos atrankos.

[SSR97] koncentruojamasi į mutacijos operatorius R^n. Praktikoje, realiai koduotų vektorių mutacija yra dažnai pasiekiama pridedant Gauso triukšmą. Pagrindinė problema lieka pasirinkti

pageidaujamą Gauso triukšmo kitimą, t.y., mutacijos žingsnį. Šis žingsnis idealiu atveju turėtų priklausyti nuo konkretaus individo ir dabartinės evoliucijos padėties.

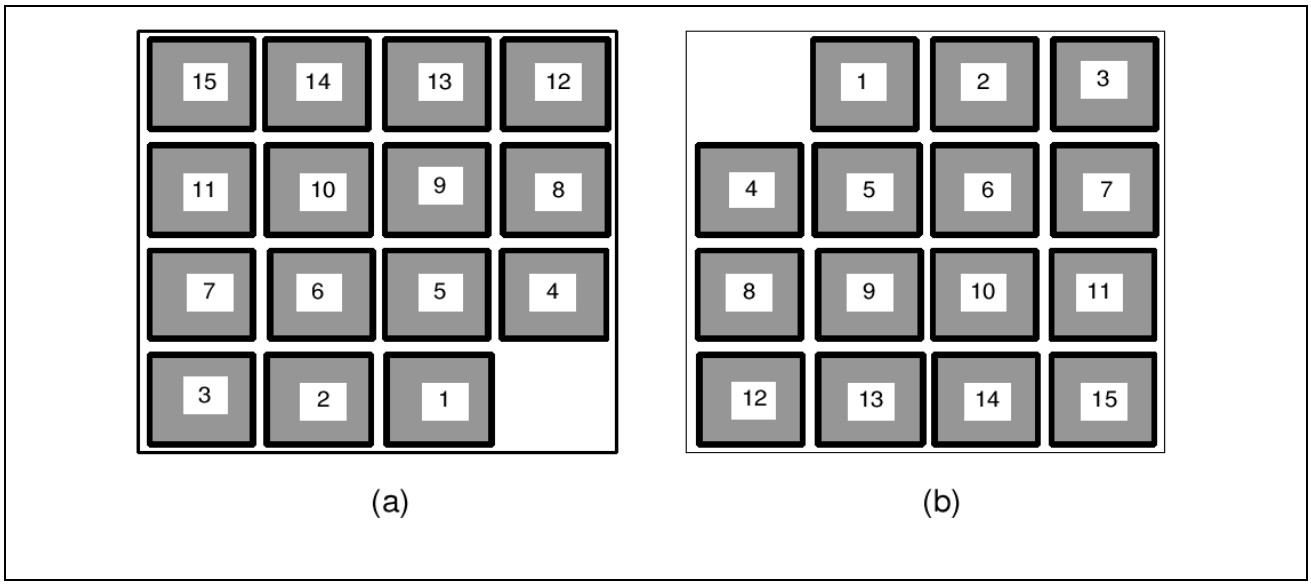
Idėja: stebėti evoliuciją ir realiu laiku mokytis iš tų stebėjimų. Pagal bandymų ir klaidų pavyzdžius, mokymasis išskiria taisykles sėkmingų bandymų atskyrimui nuo klaidų. Šios taisyklės gali būti panaudotos, kad evoliucionuojant būtų išvengta senų klaidų, bei būtų galima filtruoti evoliuciją griaunančius veiksnius sekančiose generacijose.

2.4. Testų rezultatai

Tas pats, kaip ir prieš tai buvusiame paveiksle, tik naudojamas konkurencinės atrankos metodas.

[YMW03] pateikiami rezultatai, kai genetinis algoritmas panaudojamas sprendžiant galvosūkį.

Problema: kaip n x n ploto lentoje, turint vieną laisvą vietą, kvadratėlius su skaičiais (jų yra n x n - 1) suslankioti didėjimo tvarka? Viena iš galimų pradinių uždavinio kombinacijų, kai n = 4 (a), bei siekiamas galutinis tikslas (b) pavaizduoti paveiksle 7.



Paveikslas 7. Genetinio algoritmo taikymo uždavinys

Žemiau pateikta genetinio algoritmų parametrų lentelė.

Parametras	Reikšmė
Populiacijos dydis	200
Kartų skaičius	500
Kryžminimo tipas	Atsitiktinis/pozicijos/sudėtinis
Kryžminimo dydis	0,9
Mutacijos tikimybė	0,1
Atrankos metodas	Konkurencinė
Lentos dydis (n)	3 ir 4

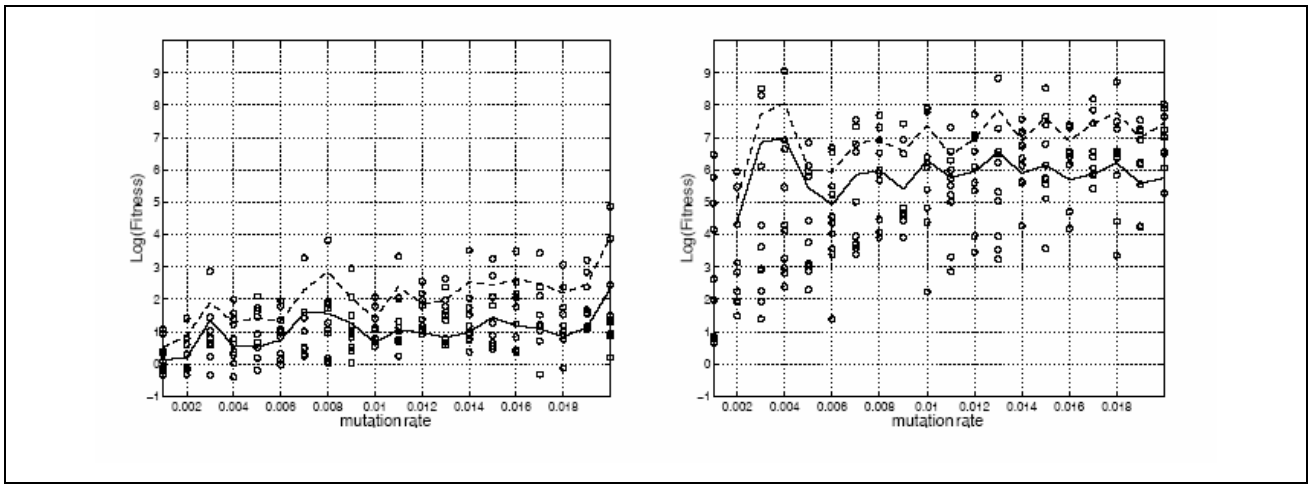
Eksperimentiniai uždavinio sprendimo rezultatai pateikti žemiau esančioje lentelėje.

Kryžminimo tipas	Dalių skaičius (n)	Vidutinė rezultato tinkamumo reikšmė	Sprendimo dydžio vidurkis	Vykdymų kiekis, tinkamam sprendimui rasti	Vidutinis laikas (sekundėmis)
Pozicijos	9	0,995	10,6,96	48	57,70
	16	0,927	865,40	0	415,27
Atsitiktinis	9	0,995	182,52	48	82,35
	16	0,935	831,70	1	408,86
Sudėtinis	9	0,995	131,32	48	60,33
	16	0,928	922,06	1	434,13

[Fro97] Testavimai atliekami su algoritmu, kurį būtų galima atvaizduoti tokiu pseudo kodu:

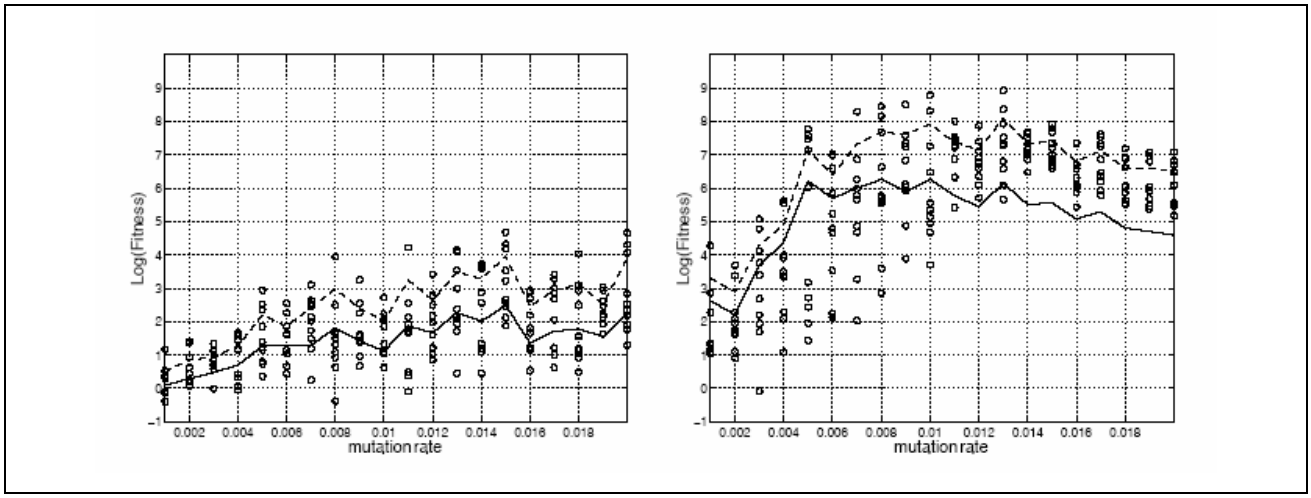
```
I : maksimalus iteracijų skaičius;
Inicializuoti Populiaciją P(0);
i=0;
while(i<I or nutraukimo sąlyga = true) {
  Apskaičiuoti Populiacijos Individus P(i);
  Išrūšiuosti Populiaciją P(i);
  do {
     Išrinkti duindividus  ind1,ind2 iš P(i);
     Rval k[0,...,1], čia k žymi atsitiktinį kintamąjį;
     if(rval< tikimybė pc) {
        kryžminimas(ind1,ind2) => vaikas1, vaikas2;
     else {
        vaikas1= ind1;
        vaikas2= ind2;
     }
     Mutuoti(vaikas1);
     Mutuoti(vaikas2);
     Nukopijuoti vaikas1 ir vaikas2 į sekančią populiaciją P(i+1)
  } until P(i+1) pilna
  Nukopijuoti geriausią P(i)individą į P(i+1);
  i=i+1;
}
```

8 paveiksle pateikiami trumpo (kairėje) ir ilgo (dešinėje) genetinio algoritmo mokymosi rezultatai, naudojant proporcionalaus tinkamumo atranką: apskritimai žymi geriausių individų tinkamumo reikšmes visuose algoritmo vykdymuose, brūkšninė linija – vidutinę tinkamumo reikšmę, o ištisinė linija – visų dešimties vykdymų našumą, palygintą su mutacijos dydžiu



Paveikslas 8. Mokymosi rezultatai naudojant proporcionalaus tinkamumo atranką

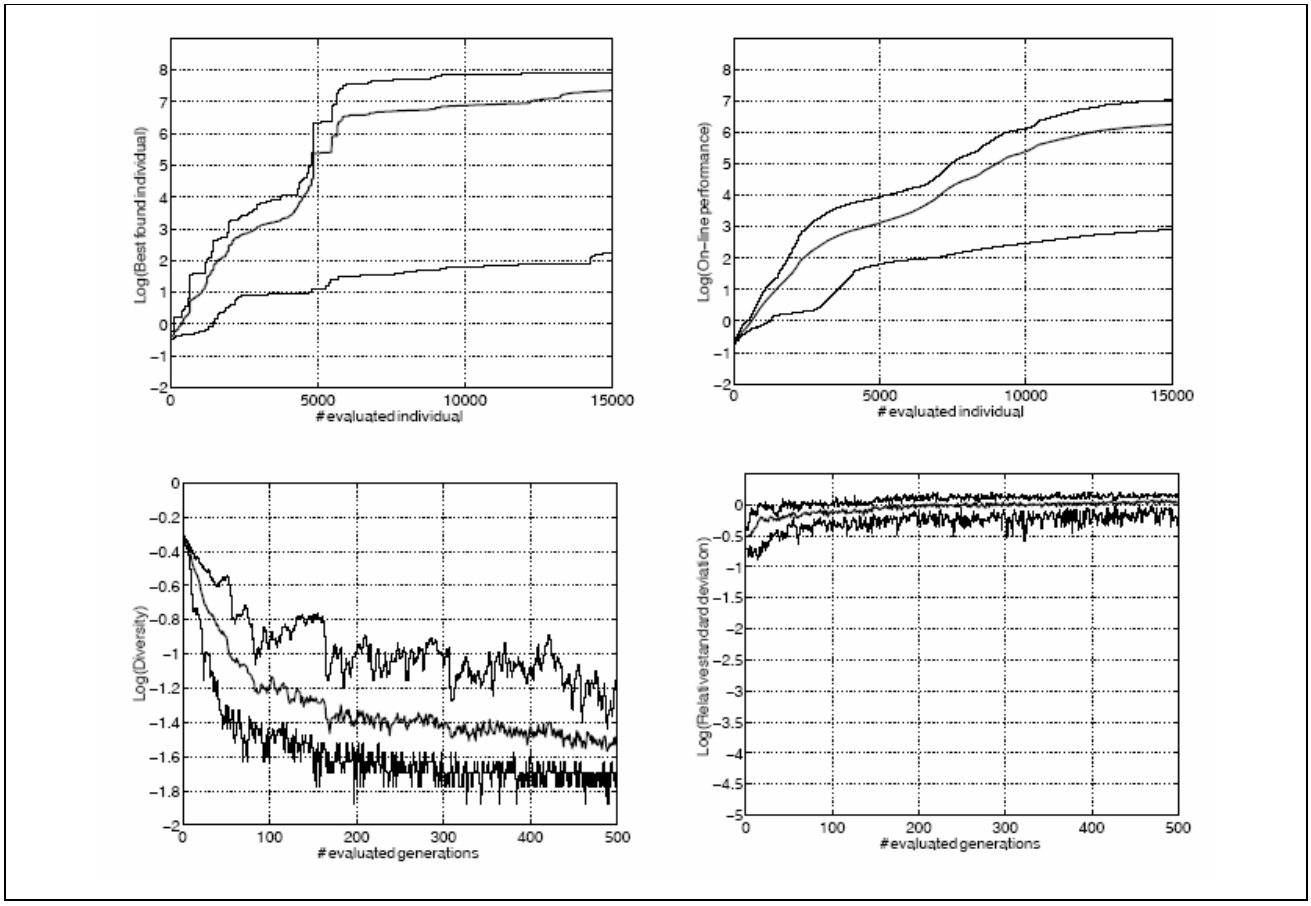
Paveiksle 9 pavaizduoti Trumpo (kairėje) ir ilgo (dešinėje) genetinio algoritmo mokymosi rezultatai, naudojant konkurencinę atranką (konkurencijos dydis: 6.



Paveikslas 9. Genetinio algoritmo mokymosi rezultatai, naudojant konkurencinę atranką

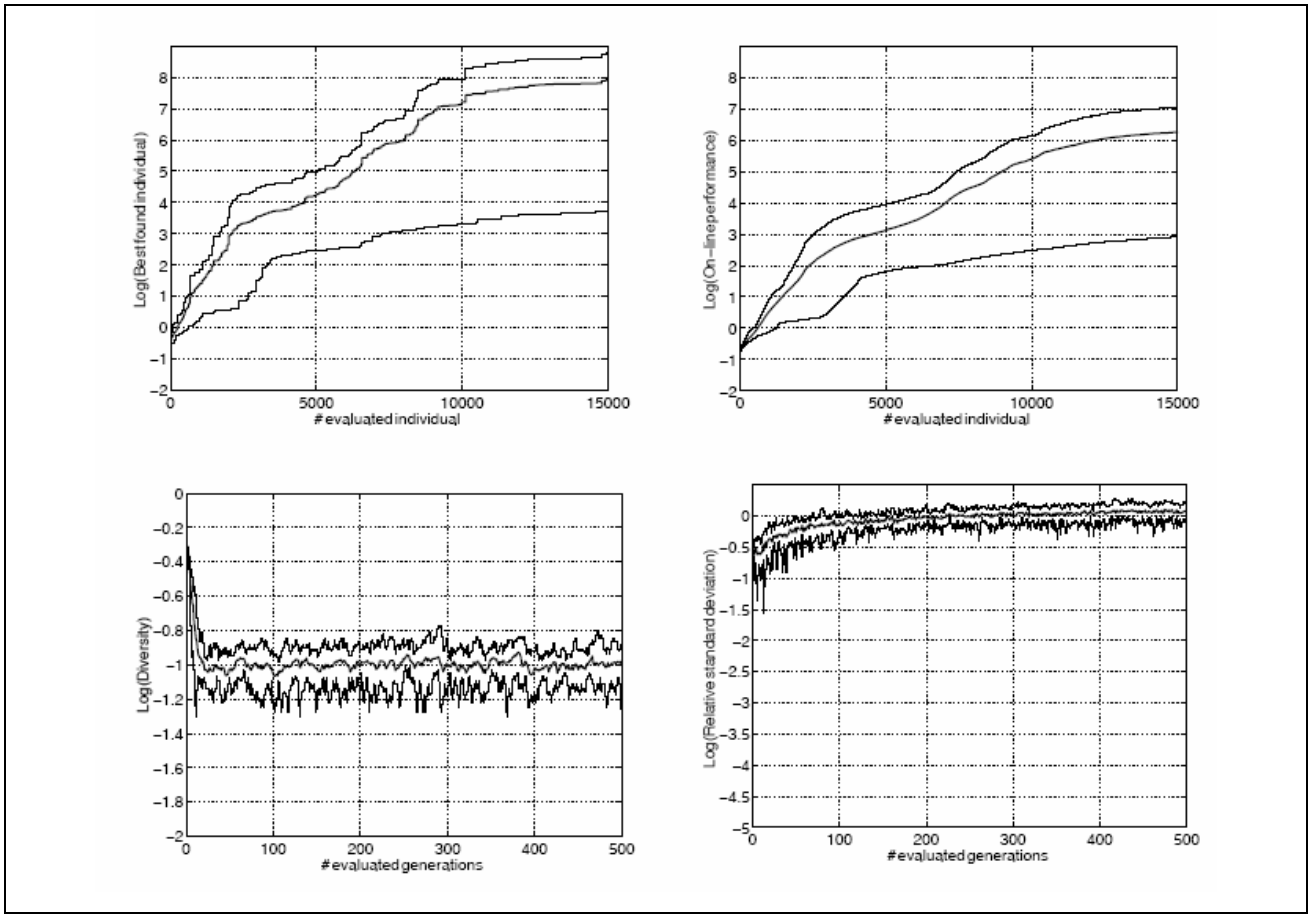
10 paveiksle pateikiama surastų geriausių individų evoliucija (viršus, kairė), našumas (viršus, dešinė), vidutinis Hemingo atstumas tarp visų populiacijos genų (apačia, kairė) ir santykinis tinkamumo reikšmių nuokrypis (apačia, dešinė) palyginus su sugeneruotų individų skaičiumi.

Mutacija lygi 0,01, naudojama proporcionalaus tinkamumo atranka. Plona linija žymi atstumą tarp kreivių dešimties vykdymų grafike, o stora linija – dešimties vykdymų vidurkį.



Paveikslas 10. Individų charakteristikos

Palyginimui, paveiksle 11 pateikiama tas pats, kaip ir 10 paveiksle, tik naudojamas konkurencinės atrankos metodas.



Paveikslas 11. Individų charakteristikos, naudojant konkurencinės atrankos metodą

2.5. Literatūros apžvalgos išvados

Genetinių algoritmų panaudojimo galimybės ir paplitimas nuolat didėja. Daugelyje nagrinėtų mokslinių darbų, genetiniai algoritmai yra naudojami uždavinių optimizavimui. Optimizavimui naudojama daug skirtingų metodų. Sprendžiant konkretų uždavinį mokslinėje literatūroje paprastai pritaikoma keletas metodų tam, kad būtų pagerinti gauti rezultatai, t.y., išbandoma keletas strategijų. Deja, nepavyko rasti tyrimų, kaip tos pačios genetinės paieškos strategijos gali būti pritaikytos kitoms analogiškoms problemoms spręsti.

Remiantis literatūros apžvalga, matyti, kad genetinės paieškos strategijų tyrimą tikslinga vykdyti toliau. Sekančiuose šio magistro tezių darbo skyriuose pateikiamas detalus tyrimo aprašymas ir rezultatai.

3. GRAFŲ UŽDAVINIAI IR TYRIMO DUOMENŲ STRUKTŪROS

Apibrėžimas

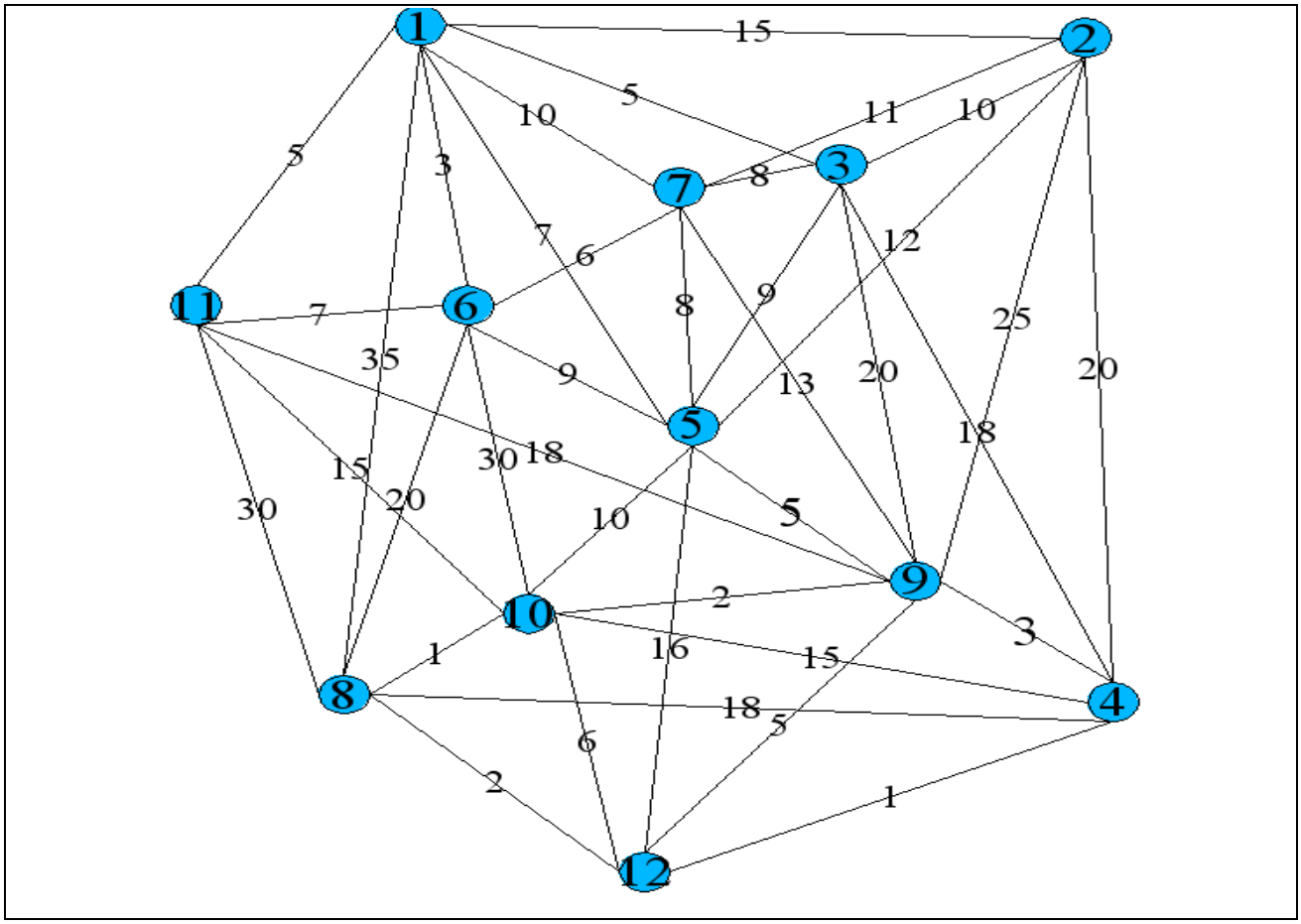
Tarkime, V - netuščia aibė (viršūnių), E - visų aibės V galimų dvielemenčių poaibių aibė (visų įmanomų briaunų aibė), U - bet koks aibės E poaibis (briaunų aibė).

Grafu vadinama pora (V, U), t.y. $G = (V, U)$.

Siekiant maksimaliai ištirti probleminę sritį, magistro tezių darbe buvo analizuojami tokie grafų uždaviniai:

- Trumpiausio kelio tarp dviejų viršūnių uždavinys;
- Ilgiausio kelio tarp dviejų viršūnių uždavinys;
- Keliaujančio pirklio uždavinys.

Kad būtų galima objektyviai palyginti rezultatus, visi uždaviniai buvo realizuoti panaudojant vienodą duomenų struktūrą. Pavyzdinis grafas, su kuriuo buvo atliekami korektiškumo testavimai realizuojant sprendimo metodus, pateikiamas 12 paveiksle.



Paveikslas 12. Pavyzdinis grafas

Šį grafą aprašanti duomenų struktūra, pavaizduota 13-ame paveiksle.

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	15	5	1000	7	3	10	35	1000	1000	5	1000
2	15	0	10	20	12	1000	11	1000	25	1000	1000	1000
3	5	10	0	18	9	1000	8	1000	20	1000	1000	1000
4	1000	20	18	0	1000	1000	1000	18	3	15	1000	1
5	7	12	9	1000	0	9	8	1000	5	10	1000	16
6	3	1000	1000	1000	9	0	6	20	1000	30	7	1000
7	10	11	8	1000	8	6	0	1000	13	1000	1000	1000
8	35	1000	1000	18	1000	20	1000	0	1000	1	30	2
9	1000	25	20	3	5	1000	13	1000	0	2	18	5
10	1000	1000	1000	15	10	30	1000	1	2	0	15	6
11	5	1000	1000	1000	1000	7	1000	30	18	15	0	1000
12	1000	1000	1000	1	16	1000	1000	2	5	6	1000	0

Paveikslas 13. Duomenų struktūra, aprašanti pavyzdinį grafą

Su šia pavyzdine grafo duomenų struktūra atliekami pirminiai testavimai, skirti nustatyti realizuotų algoritmų darbo korektiškumą. Lentelės celių reikšmės lygios 1000 (ilgiausio kelio tarp dviejų taškų uždavinio atveju, reikšmė yra -1000) žymi, kad tarp tą celę indeksuojančių viršūnių briaunų nėra, t.y., palyginus su kitomis, jos sąlyginai labai brangios. Jeigu algoritmai pasirenka šias briaunas, vadinasi kelio nerado.

Pagrindiniam magistro tezių tyrimui tokios pačios struktūros grafai generuojami automatiškai, panaudojant tam skirtą programinį modulį.

4. KOMBINATORINIŲ ALGORITMŲ TAIKYMAS GRAFŲ UŽDAVINIAMS

Vienas iš tikslų atliekant magistro tezių tyrimą buvo realizuoti keletą kombinatorinių algoritmų, dažnai taikomų spręsti grafų uždaviniams.

Magistro tezių darbe realizuoti tokie algoritmai:

- Godus algoritmas;
- Atsitiktinės paieškos algoritmas;
- Pilno perrinkimo algoritmas.

Šie algoritmai bandymuose naudojami palyginimui, kaip efektyviai dirba genetiniai algoritmai. Tiriant genetinės paieškos strategijas, kombinatoriniai algoritmai buvo panaudoti pradinės individų populiacijos parinkimui.

4.1.1. Godus algoritmas

Godus algoritmas yra euristinis algoritmas, kiekviename žingsnyje pasirenkantis lokaliai optimalų pasirinkimą. Taip tikimasi rasti globalų optimumą. Grafų uždavinių atveju, godaus algoritmo veikimo principą trumpai galima būtų aprašyti taip: kiekvieną kartą renkantis sekantį miestą kelyje, rinktis arčiausiai dabartinio miesto esantį miestą (ilgiausio kelio atveju – toliausiai esantį).

Godų algoritmą trumpiausio kelio ir keliaujančio pirklio uždaviniams būtų galima pavaizduoti tokiu pseudo-kodu:

```
%godus algoritmas
kelias = [pradzios_virsune];
kaina = 0;
esama_virsune = pradzios_virsune;

while (esama_virsune != tikslo_virsune)
    optimali_briauna = min(keliu_matrica);
    kaina = kaina + optimali_briauna;
    kelias = [kelias, virsune(optimali_briauna);
    esama_virsune = virsune(optimali_briauna);
    keliu_matrica = keliu_matrica / {optimali_briauna};
end;
```

Trumpiausio kelio atveju, godus algoritmas renkasi lokalius minimumus, ilgiausio kelio atveju - lokalius maksimumus, taigi pseudo-kodas atrodo taip:

```
%godus algoritmas
kelias = [pradzios_virsune];
kaina = 0;
esama_virsune = pradzios_virsune;

while (esama_virsune != tikslo_virsune)
    optimali_briauna = max(keliu_matrica);
    kaina = kaina + optimali_briauna;
    kelias = [kelias, virsune(optimali_briauna);
    esama_virsune = virsune(optimali_briauna);
    keliu_matrica = keliu_matrica / {optimali_briauna};
end;
```

4.1.2. Atsitiktinės paieškos algoritmas

Dar vienas euristinis algoritmas – atsitiktinės paieškos algoritmas. Šis algoritmas paremtas atsitiktinių sprendinių generavimu. Tikėtina, kad per tam tikrą kartojimų skaičių, algoritmas ras pakankamai gerą sprendimą. Atsitiktinės paieškos algoritmą galima užrašyti tokiu pseudo-kodu:

```
%atsitiktines paieskos algoritmas
minkaina = sum(keliu_matrica)
<'atsitiktiniu permaisymu inicializuoti pirma galima sprendima'>
for i = 1 : kartojimu_skaicius
    kaina = <'apskaiciuoti galimo sprendimo kaina'>
    if kaina < minkaina
        kaina = minkaina;
        <'isiminti kelia'>
    end;
    <'atsitiktinaisugeneruoti sekanti galima sprendima'>
end;
```

4.1.3. Pilno perrinkimo algoritmas

Magistro tezėse nagrinėjamuose uždaviniuose sprendinių skaičius yra baigtinis ir teoriškai juos galima išspręsti pilnai perrenkant visus galimus sprendinius. Tai daro pilno perrinkimo (brutalios jėgos) algoritmas. Struktūrizuotai jį galima pavaizduoti taip:

```
%pilnas perrinkimas
function kelias = brute_force_search(pradzios_virsune, tikslo_virsune,
                                     keliu_matrica, perejimu_matrica,
                                     blogiausias_atvejis)

if pradzios_virsune == tikslo_virsune
    kelias = [0];
    return
end

kaina = blogiausias_atvejis;
kelias = [];

for i = 1 : miestu_kiekis
    if perejimu_matrica(i) == 0 %buvome virsuneje i
        perejimu_matrica(i) == 1; %nebuvome virsuneje i - vykdome grizima
        R = brute_force_search(i, tikslo_virsune, keliu_matrica,
                               perejimo_matrica, blogiausias_atvejis);
        naujas_kelias = [i, R];
        nauja_kaina = kainos_funkcija(naujas_kelias)
        perejimu_matrica(i) == 0 %buvome virsuneje i
        if nauja_kaina < kaina
            kaina = nauja_kaina;
            kelias = naujas_kelias;
        end
    end
end
```

5. GENETINIŲ ALGORITMŲ TAIKYMAS GRAFŲ UŽDAVINIAMS

Magistro tezių darbe nagrinėjamas genetinės paieškos strategijos keliems iš ko gero labiausiai paplitusių ir turinčių gana platų panaudojimo spektrą (pavyzdžiui, įvairiuose labirintų, žemėlapių uždaviniuose, tinklų maršrutizavimo problemose ir pan.) grafų uždavinių: trumpiausio kelio paieškos, ilgiausio kelio paieškos bei keliaujančio pirklio uždaviniai.

Magistro tezių darbe, genetinės paieškos strategiją sudaro tokie genetinio algoritmo elementai:

- Kodavimas

  Šis elementas nusako, kaip atvaizduojami populiacijos individai. Nuo kodavimo būdo priklauso, kokius atrankos, kryžminimo, mutacijos operatorius galima taikyti populiacijos individams.

- Tinkamumo funkcija

  Šios funkcijos reikšmė nusako, kaip „geras“ yra individas. Kuo geresnis individas, tuo jis arčiau optimalaus uždavinio sprendimo.

- Atranka

  Atranka yra procesas kurio metu tėvinės struktūros yra išrenkamos naujų kartų formavimui. Naujos kartos vėliau gali pakeisti senų kartų atstovus.
  Populiacijos individai pagal tinkamumo funkcijos reikšmes atrenkami sekančios kartos generavimui. Geriausi individai turi didžiausią tikimybę patekti į sekančią kartą. Iš jų pritaikant kryžminimo ir mutacijos metodus, generuojami palikuonys.

- Kryžminimas

  Atrinkti individai kryžminami tarpusavyje. Taip sudaromos naujos chromosomos – palikuonys.

- Mutacija

  Tam, kad būtų galima išlaikyti natūraliems evoliucijos metodams būdingą įvairovę, o taip pat neleisti populiacijai sukonverguoti į stipriausius individus, įvedamas genų individuose pakeitimas – genų mutacija.

- Kartų pakeitimas

  Šiuo elementu nurodoma, kaip generuojama sekanti individų karta: kiek individų išgyvena iš ankstesnės kartos ir kiek jų pakeičiama palikuoniais.

Be strategijose naudojamų kodavimo, tinkamumo funkcijos paskaičiavimo, atrankos, kryžminimo, mutacijos ir kartų pakeitimo metodų, genetinio algoritmo mokymuisi taip pat didelės įtakos turi ir tai, kokia tvarka ir aplinkybėmis (pvz., su kokia tikimybe) metodai taikomi. Šie faktoriai aprašomi pateikiant bandymų rezultatus.

Remiantis literatūroje prieinamais moksliniais tyrimais, genetinių algoritmo rezultatus labiausiai įtakoja atrankos, kryžminimo ir mutacijos metodai. Dėl šios priežasties, šiems metodams magistro tezių tyrime skiriamas didžiausias dėmesys. Šiems metodams magistro tezėse realizuota po keletą algoritmų, aprašomų toliau šiame skyriuje. Programinis kodas, kuriuo realizuoti algoritmai, pateiktas I priede „Programiniai moduliai“.

5.1. Kodavimas

Tam, kad būtų galima skirtingiems uždaviniams taikyti vienodus tinkamumo reikšmių apskaičiavimo, atrankos, kryžminimo ir mutacijos metodus, reikalingas visiems uždaviniams tinkantis kodavimas. Vieningas kodavimas reikalingas taip pat ir dėl rezultatų palyginamumo.

Magistro tezių tyrimo metu genetinio algoritmo individai buvo koduojami naudojant „Mazgų” kodavimą. Populiacijos individus (chromosomas) sudaro grafo viršūnių aibė (genai). Arba kitaip tariant – populiacijos individu vaizduojamas kelias grafe.

Apibrėžimas

Turimas $G = (V, U)$ grafas , čia V – viršūnių aibė, U – briaunų aibė.

Tuomet, chromosomą C galima apibrėžti taip:

$$C = c_1 c_2 ... c_N,\ kur\ c_i \in V\ ir\ c_i \neq c_j,\ jei\ i \neq j$$

Kadangi uždavinių specifika tokia, kad kelyje negali buti ciklų, chromosomos genai yra tarpusavyje priklausomi. Nagrinėjamos chromosomos, turinčios lygiai N genų (N – grafo viršūnių skaičius).

5.2. Tinkamumo reikšmių apskaičiavimas

Tinkamumo funkcija, priklausomai nuo konkretaus uždavinio, gali būti apskaičiuojama dviem būdais.

Tinkamumo funkcija trumpiausio ir ilgiausio kelio paieškos uždaviniams

Kadangi chromosomose kiekviena viršūnė gali pasikartoti tik vieną kartą (priešingu atveju turėtume ciklą ir taip negautume trumpiausio atstumo) ir chromosomų ilgis fiksuotas, kiekvieno populiacijos individo genuose išrenkamas poaibis tarp pradinės viršūnės ir tikslo. Šiam poaibiui (o tuo pačiu ir individui) apskaičiuojama ir kelio tinkamumo (kainos) funkcija. Atrenkami individai turintys mažiausias (ilgiausio kelio paieškos atveju - didžiausias) tinkamumo funkcijos reikšmes.

Pavaizduosiu šį algoritmą struktūrizuotai.

Tarkime K – kelių (kainų) matrica; S, F – pradinė ir galinė viršūnės; tuomet kainos funkciją galima išreikšti formule:

$$cost = \sum_{i=a..b-1} K(c_i, c_{i+1});$$

Čia

$$a = min(e, d),\ b = max(e, d),\ e : c_e = S,\ d : c_d = F$$

Keliaujančio pirklio uždavinys

Keliaujančio pirklio uždaviniui tinkamumo funkcija gali būti paskaičiuota analogiškai kaip ir ilgiausio ir trumpiausio kelių atveju. Pagal atstumų tarp miestų matricą, paskaičiuojamas kelias, kurį žymi visi chromosomoje esantys genai. Tai atvaizduoja ši formulė:

$$cost = \sum_{i=1..N-1} K(c_i, c_{i+1}),\ kur\ N = |C|$$

5.3. Atranka

Atliekant magistro tezių tyrimą buvo išbandytos tokie atrankos metodai:

- Atranka pagal tinkamumo funkciją;
- Eksponentinis metodas;
- Tikimybinė atranka.

Atranka pagal tinkamumo funkciją

Tai ko gero pats paprasčiausias atrankos metodas: išrenkami geriausią tinkamumo funkcijos reikšmę turintys individai.

Iš aibės M į aibę N atrenkam s elementų. $cost$ – tinkamumo funkcijos reikšmė.

$$N = (l \in M, cost_l > h);$$

$$h : M_1 = N; M_2 = M/M_1; |N| = s$$

Eksponentinis metodas

Iš tinkamumo funkcijos reikšmių ištraukiamos kvadratinės šaknys ir pridedamas vienetas. Tai leidžia sumažinti geriausią tinkamumo funkcijos reikšmę turinčio individo įtaką. Atranka vykdoma analogiškai kaip ir atrankos pagal tinkamumo funkciją atveju, tik tinkamumo funkcijos reikšmės prieš vykdant atranką yra transformuojamos pagal tokį algoritmą:

$$f_{cost} = \sqrt{cost} + 1$$

Tikimybinė atranka

Didžiausią tinkamumo funkcijos reikšmę turintys individai atrenkami su didele tikimybe.

Atranka vykdoma kaip ir atrankos pagal tinkamumo funkciją atveju, tačiau individai iš aibės M į N patenka su tikimybe P.

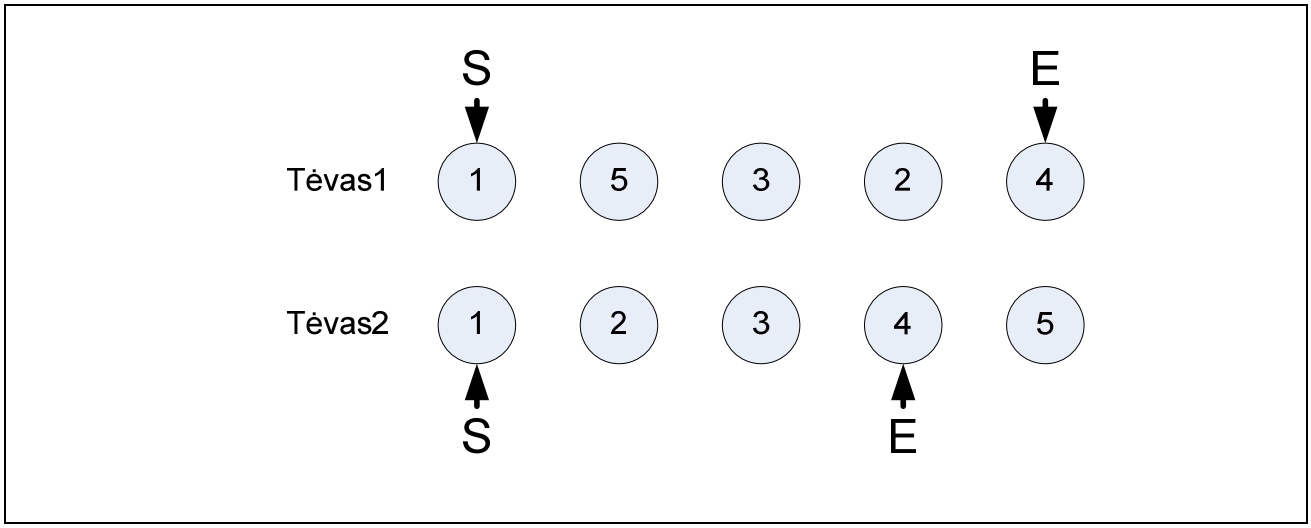
5.4. Kryžminimas

Kadangi chromosomos genai yra priklausomi vienas nuo kito (kiekvienas genas sekoje gali pasikartoti tik vieną kartą), standartiniai kryžminimo metodai (tokie kaip vieno, dviejų, daugelio taškų kryžminimas, atsitiktinis kryžminimas ir pan.) negalimi. Dėl šios priežasties, magistro tezių tyrimo metu sugalvojau keletą kryžminimo metodų, kurie tinka nagrinėjamai probleminei sričiai:

- Kryžminimas fiksuojant pradžios ir tikslo viršūnes;
- Kryžminimas fiksuojant atsitiktines viršūnes;
- Perstatos kryžminimas;
- Vieno taško kryžminimas perkoduojant.

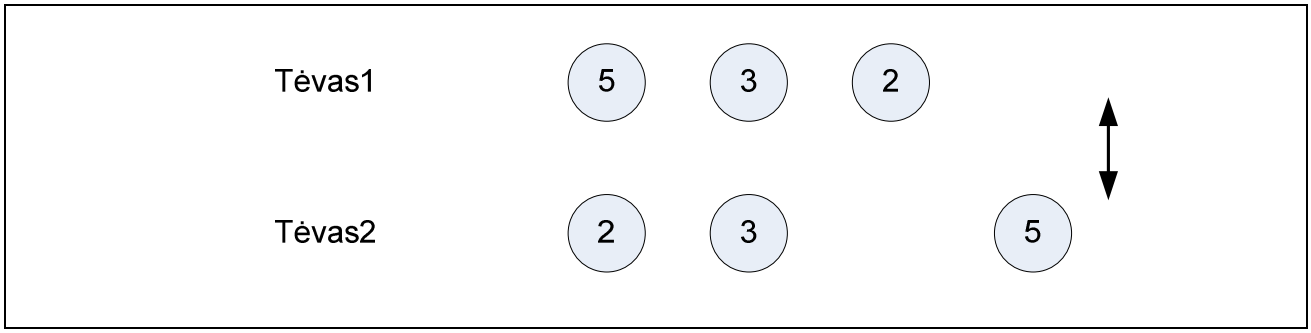
Kryžminimas fiksuojant pradžios ir tikslo viršūnes

Tėvinėse chromosomose fiksuojamos pozicijos, kuriose yra pradžios ir tikslo viršūnės, kaip parodyta paveiksle 14 (pradžios viršūnė yra 1, tikslo viršūnė - 4).



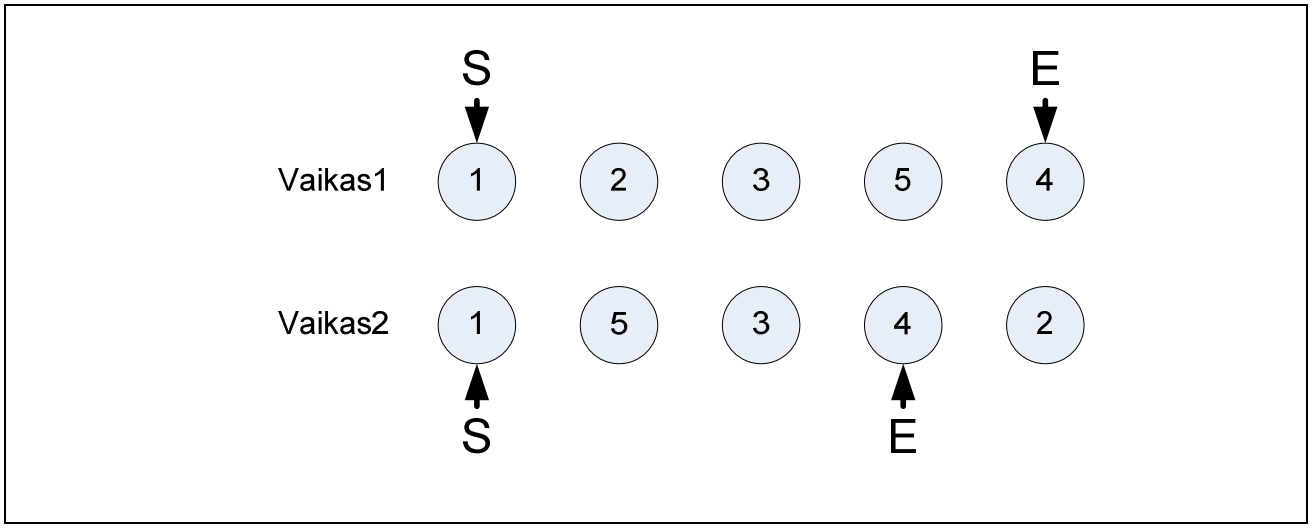
Paveikslas 14. Pozicijų fiksavimas tėvinėse chromosomose

Atrenkamos kitos viršūnės, kaip pavaizduota paveiksle 15.



Paveikslas 15. Viršūnių atrinkimas

Atrinktos viršūnės apkeičiamos tarp tėvinių individų. Tai iliustruoja žemiau esantis paveikslas.



Paveikslas 16. Kryžminimas

Visą šį algoritmą būtų galima aprašyti formalizuotai:

Tėvinės chromosomos: T^1,T^2; vaikai: V^1,V^2. S – pradinė, F – galinė viršūnės.

$$\bar{S}^i : T^i_{\bar{S}^i} = S$$

$$\bar{F}^i : T^i_{\bar{F}^i} = F$$

$$K^i = (T^i_p : p \neq \bar{S}^i, p \neq \bar{F}^i)$$

$$V^i = (K^i_1, K^i_2 ... K^i_{\bar{S}^{3-i}-1}, S, K^i_{\bar{S}^{3-i}} ... K^i_{\bar{F}^{3-i}-2}, F, K^i_{\bar{F}^{3-i}-1} ... K^i_n), n = |K|$$

<u>Kryžminimas fiksuojant atsitiktines viršūnes</u>

Šis kryžminimo metodas analogiškas pirmajam, tik šiuo atveju dvi viršūnės apsikeitimo taškų fiksavimui parenkamos atsitiktinai.

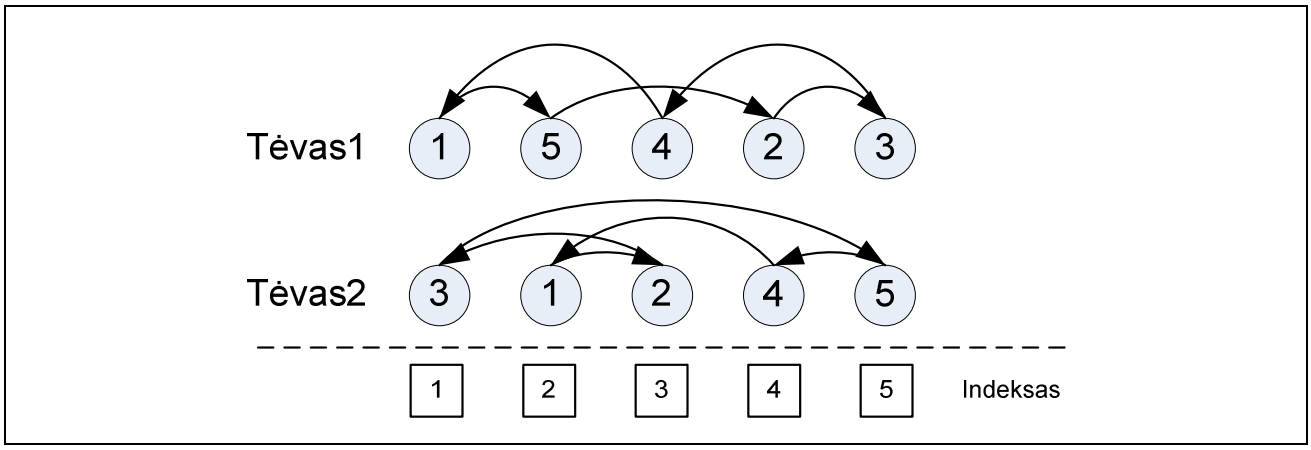
Kiti magistro tezių tyrimo metu naudoti kryžminimo parametrai (tokie kaip tikimybė ir pan.) atspindimi bandymų aprašymuose.

Perstatos kryžminimas

Šiame kryžminimo metode kiekvienas iš tėvų apibrėžia perstatą, pagal kurią perstatomas kitas tėvas. Perstatą galima aprašyti tokia formule:

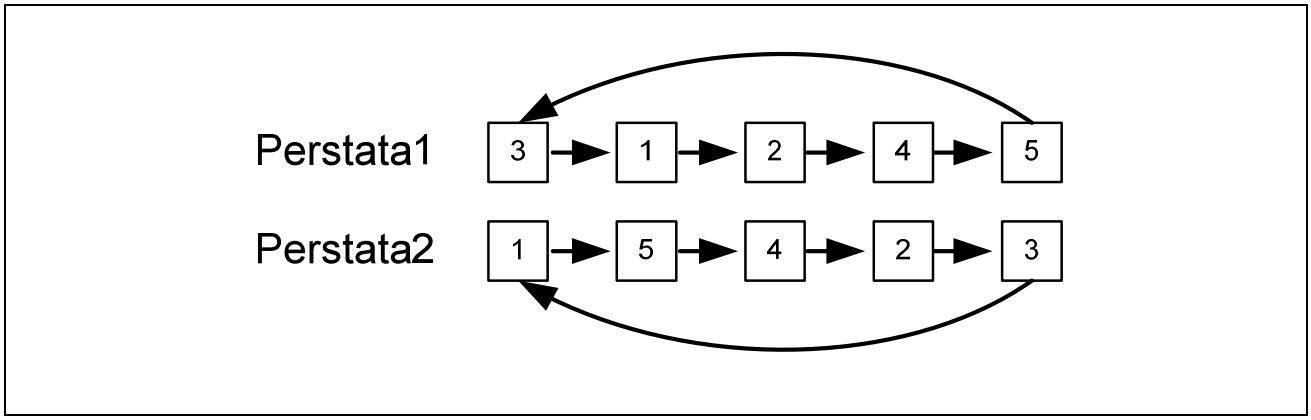
$$V_p^i = T^i_{T_p^{3-i}}$$

Pavyzdžiui, turime tokias tėvines chromosomas, kaip parodyta žemiau esančiame paveiksle. Rodyklės žymi, kaip bus perstatytas kiekvienas tėvas.



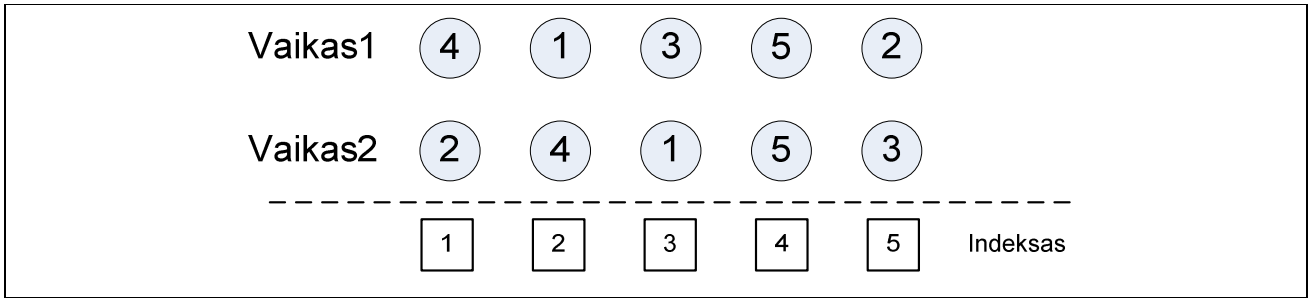
Paveikslas 17. Tėvinės chromosomos

Perstatymas vykdomas pagal genų indeksus chromosomose. Žemiau esančiame paveiksle pateiktos tėvams sudarytos perstatos. Pirmoji perstata sudaryte pagal antrąjį tėvą ir taikoma pirmajam tėvui, antroji – pagal pirmąjį it taikoma antrajam.



Paveikslas 18. Perstatos schema

Žemiau esančiame paveiksle pavaizduoti vaikai, kurie gaunami iš perstatytų tėvinių chromosomų.



Paveikslas 19. Kryžminimo rezultatai

Vieno taško kryžminimas perkoduojant

Kryžminimo operacijas riboja tai, kad genetinio algoritmo populiacijos individai žymi kelią (atlikus netinkamą kryžminimą, gali būti gautas nevalidus kelias). Kad būtų galima pritaikyti įprastinius kryžminimo metodus, reikalingas kodavimas, kuriuo užkodavus individą, jo genai taptų nepriklausomi. Dėl šios priežasties panaudojamas nepriklausomų genų kodavimas. Su perkoduotais individais atliekamas vieno taško kryžminimas. Gauti palikuonys atkoduojami atgal ir įtraukiami į sekančias kartas.

Kodavimas vykdomas tokia tvarka:

Turime aibę A.

$$A = (i : i < n, i \in \mathbb{N})$$

C – individas, kurį užkoduosime (paprastumo dėlei, aiškinant šį algoritmą, vadinu vektoriumi).

$$|C| = n$$

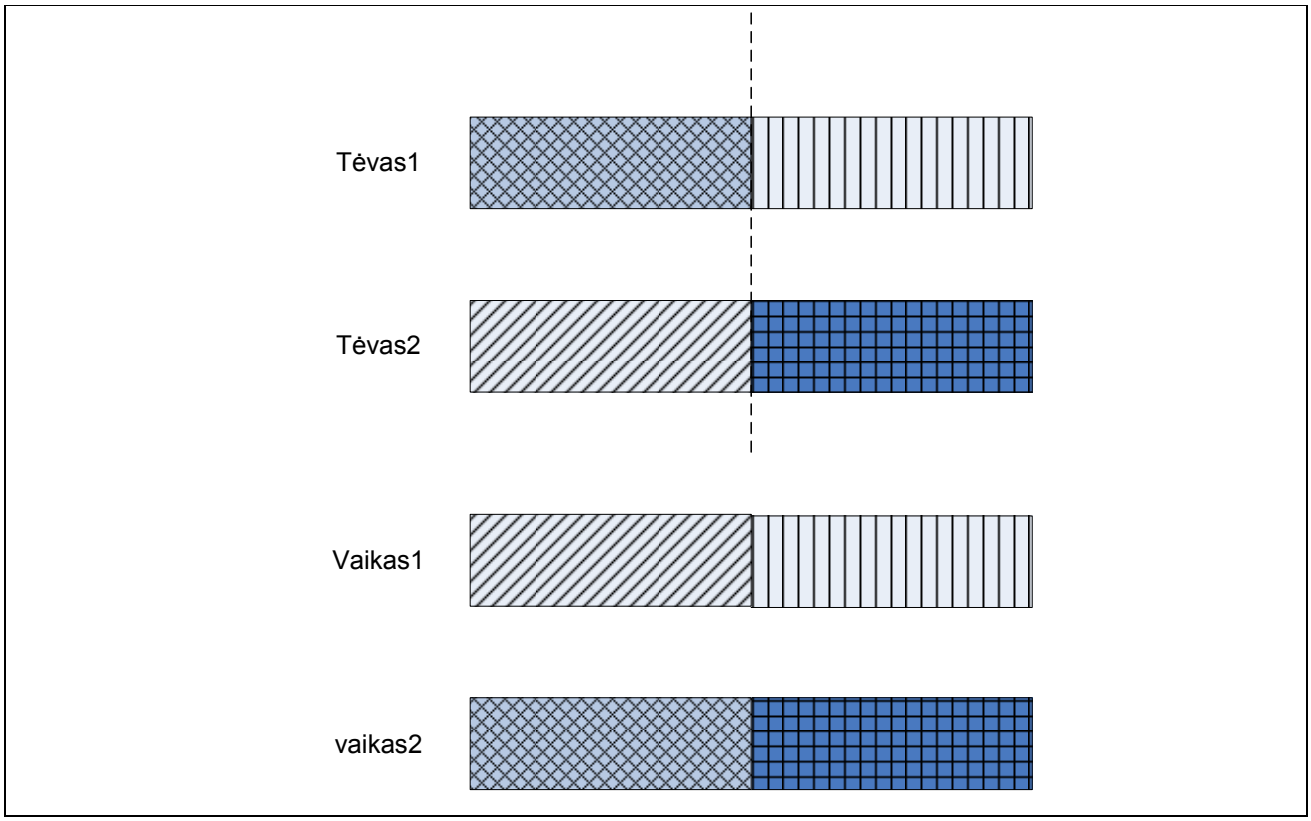
$$^{i}C' = (c_k : k \geq i)$$

$$^{i}A' = A/(C/^{i}C').$$

Užkodavus gaunamas vektorius *K*, kur

$$K_i = p \text{ , toks, kad } c_i = {}^{i}A'_p.$$

Su koduotu vektoriumi galima vykdyti klasikinius kryžminimo metodus. Magistro tezių rengimo metu realizavau vieno taško kryžminimą. Šis kryžminimas atrodo taip:



Paveikslas 20. Vieno taško kryžminimas

Vektoriai, gauti atlikus kryžminimą atkoduojami pagal algoritmą:

$${}^{1}A' = A;$$

$${}^{i}A' = ({}^{i-1}A'_k : k \neq c_i); k \in \mathbb{N}; i \in \mathbb{N}; i > 1$$

Atkodavus, gaunamas vektorius K:

$$K_i = {}^{i}A'_{c_i}$$

5.5.Mutacija

Kadangi chromosomos genai yra tarpusavyje priklausomi, be to, gali įgyti reikšmes iš nustatytos aibės, mutacijos metodų aibė susiaurėja.

Magistro tezių bandymams atlikti buvo realizuoti tokie mutacijos algoritmai:

- Atsitiktinės;
- Skirtingų genų;
- Sukeitimo;
- Vieno taško.

Atsitiktinė mutacija

Tai mutacija, kai chromosomoje atsitiktinai permaišomi visi genai.

Skirtingų genų mutacija

Mutuojamas vaikas. Chromosomoje atsitiktinai permaišomi tik tie genai, kurie buvo skirtingi tėvinėse chromosomose.

Sukeitimo mutacija

Sukeičiami vietomis atsitiktinai parinkti du chromosomos genai. Algoritmą būtų galima formalizuoti taip:

Atsitiktinai parenkami du indeksai: $p_1, p_2 \in \mathbb{N}, p_1, p_2 \leq n$. Vykdomas indeksų apkeitimas

$$c'_i = c_i, i \neq p_1, i \neq p_2$$

$$c'_{p_1} = c_{p_2}, c'_{p_2} = c_{p_1}$$

Kur C – chromosoma prieš mutaciją, C' – po mutacijos.

Vieno taško mutacija

Atsitiktinai parenkamas chromosomos dalinimo taškas. Chromosoma padalinama į dvi dalis. Gautos dalys sukeičiamos tarpusavyje.

Algoritmas:

Kaip ir sukeitimo mutacijoje, atsitiktinai parenkamas skaičius p. C – chromosoma prieš mutaciją, C' – po mutacijos.

$$k = n - p;$$

$$c'_i = c_{i+p}, i \in \mathbb{N}, i \leq p$$

$$c'_i = c_{i-k}, i \in \mathbb{N}, i > k, i \leq n$$

5.6.Kartų pakeitimas

Tyrimo metu buvo naudoti du kartų pakeitimo metodai: pilnas pakeitimas ir pastovios būsenos metodas.

Pilnas pakeitimas

Kiekvienoje kartoje pakeičiama visa populiacija.

Pastovios būsenos metodas

Šis metodas kiekvienoje kartoje pakeičia mažą populiacijos dalį.

5.7.Kiti genetinių algoritmų parametrai

Genetinių algoritmų mokymosi rezultatai labai priklauso ir nuo pradinių populiacijos narių reikšmių parinkimo.

Tiriant genetinės paieškos strategijas buvo nagrinėti atsitiktinio permaišymo, atsitiktinio permaišymo su apribojimais ir pradinės populiacijos inicializavimo panaudojant kombinatorinius algoritmus metodai.

Atsitiktinis permaišymas

Pradinė populiacija generuojama atsitiktinai permaišius visas grafo viršūnes.

Atsitiktinis permaišymas su apribojimais

Pradinė populiacija generuojama užfiksavus vieną viršūnę, o kitas permaišant atsitiktinai.

Inicializavimas panaudojant kombinatorinius algoritmus

Dalis populiacijos inicializuojama panaudojant godų paieškos algoritmą. Tam, kad būtų gaunami skirtingi individai (godus algoritmas tai pačiai pradinei viršūnei randa vieną galimą sprendimą), keičiama pradinė viršūnė. Likusi populiacijos dalis generuojama atsitiktinai permaišant grafo viršūnes.

6. GENETINĖS PAIEŠKOS STRATEGIJOS

Rengiant magistro tezes, išnagrinėti ir realizuoti genetinės paieškos strategijų elementai aprašyti ankstesniuose skyriuose. Atliekant bandymus buvo išbandytos įvairios elementų kombinacijos, tam, kad būtų galima nustatyti, kurie elementai tinkamiausi grafų uždaviniams, su kuriais iš jų gauti geriausi rezultatai. Genetiniai algoritmai taip pat buvo palyginti ir su klasikiniais algoritmais, taikomais grafų uždaviniams.

Norint ištirti visas realizuotas strategijas, reikia ištestuoti visas galimas strategijos elementų kombinacijas. Taigi, strategijų skaičius yra

$$|S| = |ini| \star |atr| \star |kry| \star |mut| \star |pak|$$

$$|ini| = 3, \ |atr| = 3, |kry| = 4, |mut| = 4, |pak| = 2.$$

Čia

$|ini|$ - inicializacijos metodų skaičius;

$|atr|$ - atrankos metodų skaičius;

$|kry|$ - kryžminimo metodų skaičius;

$|mut|$ - mutacijos metodų skaičius;

$|pak|$ - kartų pakeitimo metodas;

Mūsų atveju kiekvienam uždaviniui gaunama apie 300 strategijų. Strategijų tyrime taip pat naudojami ir kiti parametrai. Tokie, kaip:

- Mutacijos tikimybė;
- Kryžminimo tikimybė;
- Populiacijos narių skaičius;
- Generacijų skaičius.

Visi šitie parametrai dar padidina strategijų skaičių. Savaime suprantama, visų galimų variantų su skirtingais mutacijos, kryžminimo tikimybių, bei populiacijos narių skaičiais, ištirti neįmanoma, taigi šie parametrai genetiniam algoritmui mokantis keičiami priklausomai nuo dėsningumų.

Bandymai atliekami su atsitiktinai sugeneruota grafų aibe. Tam, kad gautųsi palyginami rezultatai, visos strategijos tiriamos tai pačiai grafų aibei. Grafų aibė bendra trumpiausio kelio ir keliaujančio pirklio uždaviniams. Kadangi ilgiausio kelio tinkamumo funkcija turi būti maksimizuota (trumpiausio kelio ir keliaujančio pirklio uždaviniams - minimizuota), tai reikalingas kitaip aprašytas grafas. Dėl šios priežasties, ilgiausio kelio uždavinio strategijoms tos pačios grafų aibės struktūra pakeičiama taip, kad būtų galima taikyti ilgiausio kelio uždaviniui.

Visiems uždaviniams buvo tirtos visos galimos strategijų kombinacijos, kai strategiją sudaro inicializacijos, atrankos, kryžminimo, mutacijos elementai. Šios strategijos buvo lygintos tarpusavyje, vertinant, kaip keičiasi rezultatai priklausomai nuo kartų pakeitimo metodo.

Buvo analizuotos dvi genetinio algoritmo struktūros.

Standartinė struktūra su pilnu kartų pakeitimu

Šią struktūrą būtų galima pavaizduoti tokiu pseudo kodu:

```
for i = 1 : generaciju_skaicius
    <'apskaiciuoti kainos funkcija'>;
    <'isrinkti k individu (tevai)'>;
    if rand <= tikimybe
        <'kryzminti x individu'>;
    end;
    <'sugeneruoti sekancia karta'>;
    if rand <= tikimybe
        <'mutuoti individus su tikimybe p'>;
    end;
end;
```

Pritaikyta struktūra su pastovios būsenos kartų pakeitimo algoritmu

Vykdant bandymus pastebėta, kad genetinių algoritmų rezultatams didelės įtakos turi tai, kiek individų pakeičiama sekančioje kartoje, bei kiti vykdymo parametrai priklausomai nuo individų savybių. Pastebėta, kad daugiausia įtakos turi tai, kaip parenkami tėvai kryžminimui, bei su kokiomis iš anksto nustatytomis sąlygomis atliekami kryžminimas ir mutacija.

Genetinio algoritmo struktūrą, naudotą pilnam kartų pakeitimo metodui atvaizduoja pseudo kodas:

```
for i = 1 : generaciju_skaicius
    <'apskaiciuoti kainos funkcija'>;
    <'isrinkti k individu (tevai)'>;
    if <'tevai pakankamai toli vienas nuo kito'>
        if rand <= tikimybe
            <'mutuoti individus su tikimybe p'>;
        end;
    else
        if rand <= tikimybe
            <'mutuoti individus su tikimybe p'>;
        end;
        if rand <= tikimybe
            <'kryzminti individus'>;
        end;

    end;
    <'sugeneruoti sekancia karta'>;
end;
```

Apibendrinti bandymų rezultatai pateikti sekančiuose šio skyriaus poskyriuose.

6.1. Strategijų vertinimas

Genetinės paieškos strategijos buvo vertinamos lyginant laiką, kurį užtrunka genetinio algoritmo vykdymas, bei kelio ilgį (vidutinę individo kainos funkcijos reikšmę). Strategijos pirmiausia lyginamos laiko atžvilgiu, po to – kelio ilgio. Kai algoritmo vykdymo laikas lyginant skirtingas strategijas skiriasi žymiai, laikas yra pagrindinis kriterijus. Panašų vykdymo laiką turinčios strategijos vertinamos pagal kelio ilgį.

Atlikus bandymus nustatyta, kad laiko atžvilgiu genetinės paieškos strategijos žymiai skiriasi lyginant skirtingus kartų pakeitimo algoritmus. Labai dideli vykdymo laiko skirtumai ir tada, kai genetinių algoritmų vykdymas lyginamas su pilno perrinkimo algoritmu.

Tarp strategijų su tuo pačiu kartų pakeitimo algoritmu laiko skirtumai nedideli. Todėl renkant geriausias strategijas, naudojančias vieną kartų pakeitimo metodą, laikas nėra lyginamas – strategijos „gerumas“ nustatomas pagal gautą kelio ilgį. Trumpiausio kelio paieškos ir keliaujančio pirklio uždaviniams strategija tuo geresnė, kuo trumpesnis kelias gaunamas genetiniam algoritmui baigus darbą, ilgiausio kelio atveju – atvirkščiai.

6.2. Genetinės paieškos strategijų bandymai

Genetinės paieškos strategijų bandymai buvo atlikti 1.7GHz 320Mb RAM kompiuteriu su Windows 2000 SP4 operacine sistema.

Kad būtų paprasčiau pateikti bandymų rezultatus, sunumeruoju strategijų elementus:

Nr.	**Pradinės populiacijos inicializavimas**	**Atranka**	**Kryžminimas**	**Mutacija**
1.	Atsitiktinis permaišymas	Pagal tinkamumo funkciją	Fiksuojant konkrečias viršūnes	Atsitiktinė
2.	Atsitiktinis permaišymas su apribojimais	Eksponentinė	Fiksuojant atsitiktines viršūnes	Skirtingų genų
3.	Inicializavimas panaudojant godų algoritmą	Tikimybinė	Perstatos	Sukeitimo
4.	-	-	Vieno taško perkoduojant	Vieno taško

6.2.1. Trumpiausio kelio uždavinys

Atlieku po 10 kiekvienos strategijos bandymų su automatiškai generuota grafų, turinčių 20 viršūnių aibe.

Pagrindiniai genetinio algoritmo parametrai:

Viršūnių skaičius	20
Populiacijos narių skaičius	20
Generacijų skaičius	800
Individų kiekis populiacijoje	20
Pradžios viršūnė	1
Pabaigos viršūnė	20
Mutacijos tikimybė	0.3
Kryžminimo tikimybė	0.9

Pilnas kartų pakeitimas

Atlikus šiuos bandymus, gauti rezultatai:

Išbandyta strategijų	144
Vidutinė minimali kelione vmin<= 60	32
Vykdymo trukmė (10 kartų)	[190s ; 330s]
Vidutinė minimali kelionė vmin >= 100	17
Vidutinė minimali kelionė vmin priklauso intervalui (60; 100)	97

Geriausių strategijų rezultatai pavaizduoti žemiau esančioje lentelėje. Reikšmės surūšiuotos pagal vidutinę kainą didėjančiai.

Lentelė 1. Geriausi pilno kartų pakeitimo metodo rezultatai trumpiausio kelio uždaviniui

Nr.	Rezultatai		Inicializacija	Atranka	Kryžminimas	Mutacija
	Vidutinė kaina	Laikas				
1.	51	249 s	3	1	3	3
2.	51	251 s	2	3	3	3
3.	51	285 s	1	3	2	1
4.	51	288 s	1	2	1	4
5.	52	283 s	1	2	2	1
6.	52	327 s	1	3	4	1
7.	54	206 s	3	1	2	2
8.	54	258 s	3	3	1	2
9.	54	261 s	3	1	2	4
10.	54	261 s	3	2	2	3
11.	54	265 s	3	2	2	4
12.	54	306 s	1	2	2	4
13.	54	318 s	3	1	4	3
14.	56	206 s	3	2	2	2
15.	56	240 s	1	1	1	1
16.	56	257 s	3	1	2	3
17.	56	258 s	3	1	1	3
18.	56	265 s	3	2	2	1
19.	56	315 s	3	2	4	1
20.	56	316 s	3	1	4	4

Nr.	Rezultatai		Inicializacija	Atranka	Kryžminimas	Mutacija
	Vidutinė kaina	Laikas				
21.	58	198 s	3	3	1	2
22.	58	200 s	3	2	1	2
23.	58	205 s	3	1	1	2
24.	58	256 s	3	1	4	2
25.	58	258 s	3	1	2	1
26.	58	258 s	3	2	1	3
27.	58	259 s	3	3	2	3
28.	58	263 s	3	2	4	2
29.	58	265 s	3	3	2	4
30.	58	314 s	3	2	4	4
31.	58	319 s	3	1	4	1
32.	58	319 s	3	2	4	3
Apibendrinimas						
Daugiausia įtakos turėjo:						
Pirmas metodas			6(19%)	12(38%)	8(25%)	8(25%)
Antras metodas			1(3%)	13(41%)	13(41%)	8(25%)
Trečias metodas			25(78%)	7(22%)	2(6%)	9(28%)
Ketvirtas metodas			-	-	9(28%)	7(22%)

Blogiausių strategijų rezultatai pavaizduoti žemiau esančioje lentelėje. Reikšmės surūšiuotos pagal vidutinę kainą mažėjančiai.

Lentelė 2. Blogiausi pilno kartų pakeitimo metodo rezultatai trumpiausio kelio uždaviniui

Nr.	Rezultatai		Inicializacija	Atranka	Kryžminimas	Mutacija
	Vidutinė kaina	Laikas				
1.	560	247 s	1	3	3	1
2.	280	276 s	1	2	3	3
3.	188	342 s	2	2	4	1
4.	147	230 s	2	1	3	2
5.	142	263 s	2	3	2	3
6.	140	269 s	2	3	4	2
7.	140	203 s	2	3	3	2
8.	138	331 s	2	1	4	1
9.	123	259 s	3	1	1	4
10.	117	191 s	3	3	3	2
11.	116	220 s	2	1	2	2
12.	112	347 s	2	2	4	3

Nr.	Rezultatai		Inicializacija	Atranka	Kryžminimas	Mutacija
	Vidutinė kaina	Laikas				
13.	110	258 s	2	2	3	4
14.	107	312 s	2	3	4	1
15.	104	295 s	2	1	3	3
16.	104	273 s	2	1	3	4
17.	103	291 s	2	2	3	1
Apibendrinimas						
Daugiausia įtakos turėjo:						
Pirmas metodas			2(12%)	6(35%)	1(6%)	5(29%)
Antras metodas			13(76%)	5(29%)	2(12%)	5(29%)
Trečias metodas			2(12%)	6(35%)	9(53%)	4(24%)
Ketvirtas metodas			-	-	5(29%)	3(18%)

Pastovios būsenos kartų pakeitimas

Atlikus bandymus, gauti rezultatai:

Išbandyta strategijų	144
Vidutinė minimali kelione vmin<= 60	144
Vykdymo trukmė (10 kartų)	[10s ; 30s]
Vidutinė minimali kelionė vmin >= 100	0
Vidutinė minimali kelionė vmin priklauso intervalui (60; 100)	0

Kaip matome iš rezultatų, genetinis algoritmas su pastovios būsenos kartų pakeitimo strategija veikia gerokai stabiliau ir greičiau, negu pilno pakeitimo atveju.

Išskirsiu geriausias strategijas, kai vidutinė minimali kelionė < 50. Rezultatai pateikti žemiau esančioje lentelėje

Lentelė 3. Geriausi pastovios būsenos kartų pakeitimo metodo rezultatai trumpiausio kelio uždaviniui

Nr.	Rezultatai		Inicializacija	Atranka	Kryžminimas	Mutacija
	Vidutinė kaina	Laikas				
1.	38	21 s	2	3	3	2
2.	42	20 s	1	3	3	2

6.2.2. Ilgiausio kelio uždavinys

Pagrindiniai genetinio algoritmo parametrai visoms ilgiausio kelio uždavinio sprendimo strategijoms:

Viršūnių skaičius	20
Populiacijos narių skaičius	20
Generacijų skaičius	800
Individų kiekis populiacijoje	20
Pradžios viršūnė	1
Pabaigos viršūnė	20
Mutacijos tikimybė	0.3
Kryžminimo tikimybė	0.9

Pilnas kartų pakeitimas

Atlikus bandymus, gauti rezultatai:

Išbandyta strategijų	144
Vykdymo trukmė (10 kartų)	[100s ; 200s]
Nerasta sprendinio	144

Kaip matyti iš lentelės, pilno kartų pakeitimo metodą ilgiausio kelio uždaviniui taikyti nėra tikslinga.

Pastovios būsenos kartų pakeitimas

Atlikus bandymus, gauti rezultatai:

Išbandyta strategijų	144
Vidutinė maksimali kelione vmax >= 450	15 kartų
Vykdymo trukmė (10 kartų)	[8s ; 15s]
Vidutinė maksimali kelionė vmax <= 250	10 kartų
Vidutinė maksimali kelionė vmax priklauso intervalui (250; 450)	119 kartų

Geriausių strategijų rezultatai pavaizduoti žemiau esančioje lentelėje. Reikšmės surūšiuotos pagal vidutinę kainą didėjančiai.

Lentelė 4. Geriausi pastovios būsenos kartų pakeitimo metodo rezultatai ilgiausio kelio uždaviniui

Nr.	Rezultatai		Inicializacija	Atranka	Kryžminimas	Mutacija
	Vidutinė kaina	Laikas				
1.	454	9 s	3	2	3	2
2.	455	8 s	3	1	2	1
3.	469	9 s	2	3	3	2
4.	471	10 s	1	2	3	3
5.	473	12 s	1	2	4	4
6.	477	9 s	3	2	3	3
7.	479	8 s	1	3	3	1
8.	488	9 s	1	3	3	2
9.	499	9 s	3	1	4	4
10.	511	9 s	3	2	3	4
11.	515	8 s	3	1	3	3
12.	531	12 s	3	2	4	4
13.	543	8 s	3	3	3	4
14.	559	10 s	3	3	4	4
15.	562	9 s	2	3	3	3
Apibendrinimas						
Daugiausia įtakos turėjo:						
Pirmas metodas			4(27%)	3(20%)	0(0%)	2(13%)
Antras metodas			2(13%)	6(40%)	1(7%)	3(20%)
Trečias metodas			9(60%)	6(40%)	10(67%)	4(27%)
Ketvirtas metodas			-	-	4(27%)	6(40%)

Blogiausių strategijų rezultatai pavaizduoti žemiau esančioje lentelėje. Reikšmės surūšiuotos pagal vidutinę kainą mažėjančiai.

Lentelė 5. Blogiausi pastovios būsenos kartų pakeitimo metodo rezultatai ilgiausio kelio uždaviniui

Nr.	Rezultatai		Inicializacija	Atranka	Kryžminimas	Mutacija
	Vidutinė kaina	Laikas				
1.	272	8 s	2	1	2	3
2.	248	8 s	2	1	4	2
3.	246	10 s	1	2	2	2
4.	237	12 s	2	3	4	2
5.	232	8 s	2	1	2	2
6.	228	8 s	1	1	3	2

Nr.	Rezultatai		Inicializacija	Atranka	Kryžminimas	Mutacija
	Vidutinė kaina	Laikas				
7.	216	14 s	2	2	4	2
8.	207	15 s	1	2	4	2
9.	204	12 s	1	3	4	2
10.	171	10 s	2	2	2	2
Apibendrinimas						
Daugiausia įtakos turėjo						
Pirmas metodas			4(40%)	4(40%)	0 (0%)	0(0%)
Antras metodas			6(60%)	4(40%)	4(40%)	9(90%)
Trečias metodas			0(0%)	2(20%)	1(10%)	1(10%)
Ketvirtas metodas			-	-	5(50%)	0(0%)

6.2.3. Keliaujančio pirklio uždavinys

Pagrindiniai genetinio algoritmo parametrai visoms keliaujančio pirklio uždavinio sprendimo strategijoms:

Viršūnių skaičius	20
Populiacijos narių skaičius	20
Generacijų skaičius	800
Individų kiekis populiacijoje	20
Pradžios viršūnė	1
Pabaigos viršūnė	20
Mutacijos tikimybė	0.3
Kryžminimo tikimybė	0.9

Geriausi keliaujančio pirklio uždavinio rezultatai gauti strategijomis, kai pradinė populiacija buvo generuojama panaudojant godų algoritmą.

Naudojant kitas genetinio algoritmo mokymosi strategijas, dauguma atvejų algoritmas nerasdavo sprendimo – būdavo pasirenkamos neegzistuojančios briaunos. Tarp šių strategijų buvo keletas, kurios rasdavo sprendimus, tačiau jie buvo blogesni už rastus naudojant pradinės populiacijos inicializavimą panaudojant godų algoritmą. Dėl šios priežasties, lentelėse pateikiami tik naudojančių inicializavimą su godžiu algoritmu strategijų rezultatai.

Pilnas kartų pakeitimas

Atlikus bandymus, gauti rezultatai:

Išbandyta strategijų (iš viso)	144
Strategijos, kai inicializuojama panaudojant godų algoritmą (3_ini_strat)	48
Vykdymo trukmė (10 kartų)	[100s ; 205s]
Vidutinė minimali kelionė 3_ini_strat vmin <= 382	13
Vidutinė minimali kelionė 3_ini_strat vmin priklauso intervalui (382; 713)	23
Nerasta sprendinio	12

Geriausių strategijų rezultatai pavaizduoti 6 lentelėje.

Lentelė 6 Geriausi pilno kartų pakeitimo metodo rezultatai keliaujančio pirklio uždaviniui

Nr.	Rezultatai		Inicializacija	Atranka	Kryžminimas	Mutacija
	Vidutinė kaina	Laikas				
1.	381	128 s	3	1	2	2
2.	381	138 s	3	1	1	3
3.	382	121 s	3	3	1	2
4.	382	127 s	3	2	1	2
5.	382	140 s	3	1	1	1
6.	382	140 s	3	3	1	3
7.	382	141 s	3	2	1	3
8.	382	141 s	3	2	4	2
9.	382	157 s	3	2	1	1
10.	382	173 s	3	1	4	2
11.	382	180 s	3	2	4	3
12.	382	201 s	3	1	4	1
13.	382	203 s	3	1	4	3
Apibendrinimas						
Daugiausia įtakos turėjo:						
Pirmas metodas			0(0%)	6(46%)	7(54%)	3(23%)
Antras metodas			0(0%)	5(38%)	1(8%)	5(38%)
Trečias metodas			13(100%)	2(15%)	0(0%)	5(38%)
Ketvirtas metodas			-	-	5(38%)	0(0%)

Blogiausių strategijų rezultatai pavaizduoti žemiau esančioje lentelėje. Reikšmės surūšiuotos pagal vidutinę kainą mažėjančiai.

Lentelė 7. Blogiausi pilno kartų pakeitimo metodo rezultatai keliaujančio pirklio uždaviniui

Nr.	**Rezultatai**		**Inicializacija**	**Atranka**	**Kryžminimas**	**Mutacija**
	Vidutinė kaina	**Laikas**				
1.	1881	140 s	3	3	3	4
2.	1817	140 s	3	3	3	1
3.	1642	179 s	3	1	3	1
4.	1642	139 s	3	2	3	1
5.	1642	113 s	3	3	3	2
6.	1635	169 s	3	2	3	3
7.	1632	150 s	3	3	3	3
8.	1589	119 s	3	2	3	2
9.	1582	151 s	3	1	3	3
10.	1570	134 s	3	1	3	2
11.	1567	164 s	3	1	3	4
12.	1427	159 s	3	2	3	4
Apibendrinimas						
Daugiausia įtakos turėjo						
Pirmas metodas			0(0%)	4(33%)	0(0%)	3(25%)
Antras metodas			0(0%)	4(33%)	0(0%)	3(25%)
Trečias metodas			12(100%)	4(33%)	12(100%)	3(25%)
Ketvirtas metodas			-	-	0(0%)	3(25%)

Pastovios būsenos kartų pakeitimas

Išbandyta strategijų (iš viso)	144
Strategijos, kai inicializuojama panaudojant godų algoritmą (3_ini_strat)	48
Vykdymo trukmė (10 kartų)	[18s ; 26s]
Vidutinė minimali kelionė 3_ini_strat vmin priklauso intervalui (378; 382)	48

Kaip matome iš gautų rezultatų, strategijų su inicializavimu panaudojant godų algoritmą atveju, keliaujančio pirklio uždavinys su pastovios būsenos kartų pakeitimu veikė labai stabiliai. Gautos minimalios kelionės kito intervale nuo 378 iki 382.

Dėl itin mažo gautų reikšmių pasiskirstymo, netikslinga išskirti atskirų strategijų. Ir galime prieiti išvados, kad didžiausią įtaką šiuo atveju darė inicializacijos metodas

6.3. Genetinio algoritmo palyginimas su kombinatoriniais algoritmais

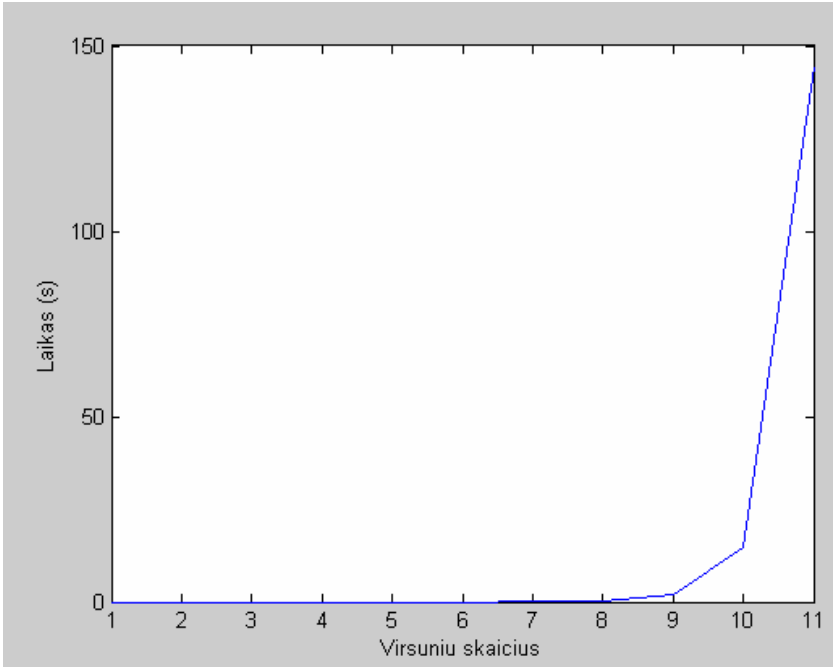
Magistro tezių darbo metu gauti genetinio algoritmo mokymosi rezultatai buvo palyginti su kombinatoriniais algoritmais. Palyginimas buvo atliktas su tokiais algoritmais:

- Pilno perrinkimo algoritmu;
- Atsitiktinės (randomized) paieškos;
- Godžiu algoritmu;

6.3.1. Pilno perrinkimo algoritmas

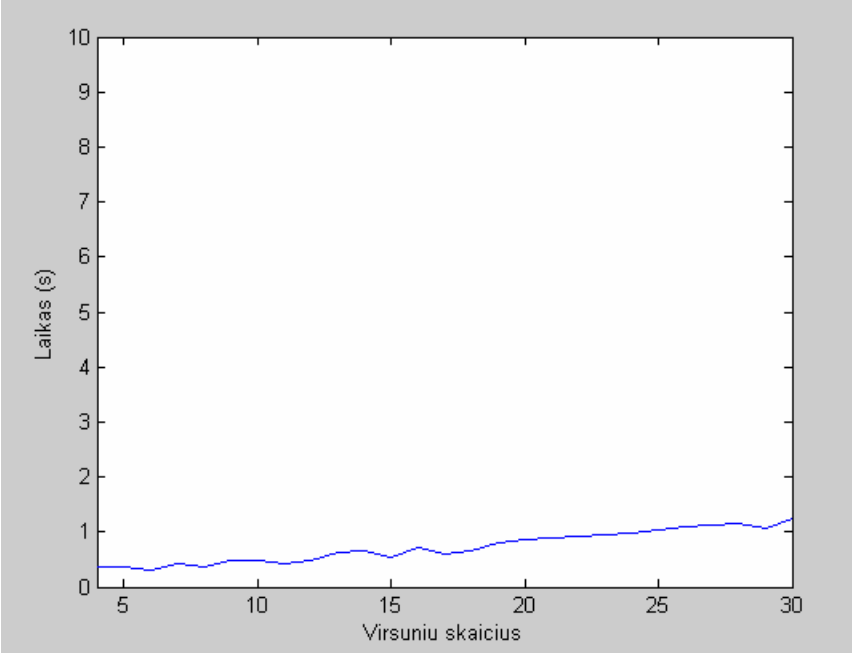
Pilno perrinkimo algoritmas randa globaliai optimalų sprendinį grafe. Tačiau pagrindinis šio algoritmo minusas yra tai, kad jis reikalauja labai daug laiko paieškai, taigi šio algoritmo pritaikyti praktiškai neina.

Paveiksle 21 pateikiama pilno perrinkimo algoritmo vykdymo laiko priklausomybė nuo viršūnių skaičiaus.



Paveikslas 21. Vykdymo laiko priklausomybė nuo viršūnių skaičiaus pilno perrinkimo algoritme

Žemiau esančiame paveiksle pateikiama genetinio algoritmo vykdymo laiko priklausomybė nuo viršūnių skaičiaus (naudojant pastovios būsenos kartų pakeitimo algoritmą).



Paveikslas 22. genetinio algoritmo vykdymo laiko priklausomybė nuo viršūnių skaičiaus

Kaip matyti iš grafikų, genetiniai algoritmai greičio požiūriu gerokai lenkia pilno perrinkimo algoritmą.

6.3.2. Atsitiktinės paieškos ir godus algoritmai

Magistro tezių tyrimo metu buvo atlikti palyginimai su klasikiniais algoritmais, taikomais grafų uždaviniams:

- Atsitiktinės paieškos algoritmu;
- Godžiu algoritmu.

Atsitiktinės paieškos atveju buvo atliekama tiek pat bandymu, kaip ir kiekvienai genetinių algoritmų strategijai: 10 bandymų po 1000 vykdymų. Bandymai buvo vykdomi su ta pačia grafų aibe.

Atsitiktinės paieškos algoritmas nė karto nerado sprendinio.

Tyrimo metu taip pat buvo atliktas genetinio algoritmo palyginimas su godžiu algoritmu. Buvo atlikta 10 bandymų su skirtingomis viršūnėmis. Godus algoritmas veikė šiek tiek greičiau, bet genetiniai algoritmai 50% dažniau rado geresnį rezultatą.

7. IŠVADOS IR REZULTATAI

Kaip matome iš bandymų rezultatų, tos pačios genetinės paieškos strategijos gali būti pritaikytos skirtingiems grafų uždaviniams. Tendencijos, kurios mokymosi strategijos geriausios yra bendros visiems nagrinėtiems grafų uždaviniams.

Svarbiausią įtaką turėjo kartų pakeitimo metodas ir pradinės populiacijos generavimo algoritmas.

Analizuojant rezultatus, visais atvejais pirmauja pastovios būsenos kartų pakeitimo metodas. Tam tikriems uždaviniams gali būti geresnis pilno kartų pakeitimo metodas – juo galima apriboti stipriausių individų dominavimą, tačiau, kaip parodė tyrimai, grafų uždavinių atveju, tinkamesnis pastovios būsenos metodas.

Tarp kitų strategijų elementų didžiausią įtaką turėjo pradinės populiacijos generavimo metodas. Labiausiai pasiteisino pradinės populiacijos generavimas panaudojant klasikinius euristinius algoritmus.

Tarp likusių strategijos elementų tinkamiausius grafų uždavinių sprendimui išrinkti sunkiau. Bet pagrindines tendencijas išskirti galima. Naudojamas atrankos metodas didelės įtakos genetinių algoritmų mokymuisi neturėjo: visi atrankos metodai darė apylygę įtaką mokymosi rezultatams. Nors kryžminimo įtaka taip pat nėra didelė, būtų galima išskirti fiksuotų viršūnių (tiek iš anksto numatytų, tiek atsitiktinai fiksuojamų kryžminimą). Tinkamiauios mutacijos metodą išskirti sunku.

Darbe išanalizuotas strategijas galima tirti toliau, realizuojant naujus strategijų elementus, tiriant jų sąveiką su dabar išskirtomis geriausiomis strategijomis.

LITERATŪROS SĄRAŠAS

[AH99] K. S. Anderson, Y. Hsu. Genetic crossover strategy using an approximation concept, 576KB, 1999

[Bro04] J. Brownlee. Parallel niching genetic algorithms a crowding perspective, http://www.it.swin.edu.au/personal/jbrownlee/thesis/Minor%20Thesis%20-%20Release%20Version%201.6.pdf, 2004

[Car01] Adaptation, http://www.carc.aist.go.jp/~kiyoshi/paper/dron/node38.html, 2001

[CC96] Mei-Shiang Chang and Huey-Kuo Chen. A New Encoding Method of Genetic Algorithmtowards Parameter Identification of Fuzzy Expert System, 1996

[Deb05] K. Deb. Evolutionary Algorithms for Optimization. http://www.iitk.ac.in/kangal/EA-course.html, 2005

[EHM00] E. Eiben, R. Hinterding, Z. Michalewicz. Parameter control in evolutionary algorithms, 327KB, 2000.

[Fro97] J. Froelich. Evolutionary Optimization for Computational Electromagnetics, http://e-collection.ethbib.ethz.ch/ecol-pool/diss/fulltext/eth12232.pdf, 1997

[Har00] S. Harmeling. Solving Satisfiability Problemswith Genetic Algorithms, http://ida.first.fraunhofer.de/~harmeli/pubs/genetic_sat.pdf, 2000

[Htc98] Genetic algorithms http://www.htc.honeywell.com/projects/neurogen/Genetic_Algorithms.htm, 1998

[YL99] X. Yao, Y. Liu. Evolving artificial neural networks, IEEE, 1999

[YMW03] Han Yu, Dan C. Marinescu, Annie S. Wu. A Genetic Approach to Planning in Heterogeneous Computing Environments,http://www.cs.ucf.edu/~ecl/papers/0304.hcw.pdf, 2003

[Luk95] B. T. Luke. Overview of Genetic Methods, http://members.aol.com/btluke/gmovr01.htm, 19KB, 1995

[Mar04] A. Marczyk. Genetic Algorithms and Evolutionary Computation http://www.talkorigins.org/faqs/genalg/genalg.html, 2004

[Neo98] Networking and emerging optimization, http://neo.lcc.uma.es/TutorialEA/semEC/cap02/cros_ops.gif, 1998

[Ren00] J. F. Renard. Genetic Algorithm Viewer : Demonstration of a Genetic Algorithm, www.rennard.org/alife/english/gavgb.pdf, 61,5KB,2000

[Ric95] R. A. Richards. Glossary, http://www.stanford.edu/~buc/SPHINcsX/bkhm15.htm#RndMat, 1995

[Sen00] P.K.Senegal. Numerical optimization using the GEN4 micro-genetic algorithm code, http://www.erc.wisc.edu/~hessel/research/manuals/manualGaWrittenByKelly.pdf, 613KB, 2000

[SSR97] M. Sebag, M.Schoenauer, C.Ravise. Inducrive learning o mutation step-size in evolutionary parameter optimization, 562KB, 1997.

[Tak04] A. Takahaki.Genetic algorithms in complex optimization tasks, www3.acadlib.lv/greydoc/ Takahasi_disertacija/Takahasi_ang.doc, 327Kb, 2004

[Dar00] R. D. Darby. Basic Normal Growth of an Organism, http://www.hctc.com/~darby/CHAP2.HTM, 2000.

[Pey99] M. Peysakhov. Genetic algorithms fot the sub-graph isomorphism problem, http://edge.cs.drexel.edu/GICL/people/peysakhov/GraphGa/Graph_ga.pdf, 336KB, 1999.

[MSP04] V. Muntes Mulero, J. Aguilar Saborit, J. Lluis Larriba Pey, C.Zuzarte. A study of execution plan aware mutations for genetic cyclic query optimization, http://www.dama.upc.edu/downloads/mutation04.pdf, 209KB, 2004.

[AW04] Y. Akkhalifah, R. L. Wainwright. A genetic algorithm applied to graph problems involving subsets of vertices, http://www.mcs.utulsa.edu/~rogerw/papers/Yaser-CEC-2004.pdf, 159KB, 2004.

[MBB05] A. Misevičius, J. Blonskis, V. Bukšnaitis. Kombinatorinio optimizavimo ir genetinių algoritmų aspektai. http://www.ik.ku.lt/kodi2005/prezentacijos/16.ppt, 313KB, 2005.

[Kov05] V. Kovaliovas. Genomų palyginimo algoritmų tyrimas, http://submit.library.lt//ETD-afiles/KTU/etd-LABT20050523-134443-96340/unrestricted/kovaliovas.pdf, 524KB, 2005

[Bal04] M. Balnys. Genetinių algoritmų pritaikymo klasifikavimo uždaviniams spręsti tyrimas, http://submit.library.lt//ETD-afiles/KTU/etd-LABT20040528-155652-64676/unrestricted/Mantas_Balnys_GAinClassification2.pdf, 765KB, 2004

[Mis03] A. Misevičius. Intelektualieji optimizavimo metodai. Informacijos mokslai, 26, http://www.tzc.vu.lt/get.php?f.936, 2003, pp. 160-166.

[Pul06] W. Pullan. Adapting the genetic algorithm to traveling salesman problem. http://www.int.gu.edu.au/courses/2008int/ga_tsp1.pdf, 412KB, 2006.

[JSR01] G. Andal Jayalakshmi, S. Sathiamoorthy, R. Rajaram. A hybrid genetic algorithm – a new approach to solve traveling dalesman problem, 147KB, 2001.

PRIEDAI

I. PROGRAMINIAI MODULIAI

I.1. Pradinės populiacijos inicializavimas

```
function population = Initialize (kk, greed, npop, N, X, s, e)
%pradines populiacijos inicializavimo funkcija.
%k - populiacijos inicializavimo metodas
%npop - populiacijos dydis
%N - miestu skaicius (genu skaicius chromosomoje)
%X - keliu ilgiai tarp miestu
%s - pradzios virsune
%e - tikslo virsune

% k == 1: atsitiktinis permaisymas
% k == 2: atsitiktinis permaisymas su apribojimais
% k == 3: viena chromosoma parenkama greedy algoritmu, kitos - atsitiktinai

population = zeros(npop, N);
%atsitiktinis permaisymas
if kk == 1
    for i = 1 : npop
        population(i, :) = randperm (N);
    end;
end;

%atsitiktinis permaisymas su apribojimais
if kk == 2
    for i = 1 : npop
        population(i, :) = [1 randperm(N-1)+1];
    end;
end;

%greedy algoritmu parenkamas vienas populiacijos narys, kiti
%atsitiktiniu permaisymu.
if kk == 3
    V = [];
    V2 = [];
    [c, V] = greedy_search (greed, X, s, e);
    [c, V2] = greedy_search (greed, X, e, s);
    %prie greedy metodu gautu vektoriu galo prijungiam trukstamas
    %virsunes
    tmp = [1:N];
    tmp2 = sort (V);

    %pirmas individas
    k = 1;
    for j = 1 : N
        if k <= size(tmp2, 2)
```

```
            if tmp2(k) == j
                tmp = [0, tmp(1:j-1), tmp(j+1:N)];
                k = k + 1;
            end;
        end;
    end;
    V = [V, tmp(size(tmp2,2)+1:size(tmp, 2))];
    %antras individas
    k = 1;
    tmp = [1:N];
    tmp2 = sort (V2);
    for j = 1 : N
        if k <= size(tmp2, 2)
            if tmp2(k) == j
                tmp = [0, tmp(1:j-1), tmp(j+1:N)];
                k = k + 1;
            end;
        end;
    end;
    V2 = [V2, tmp(size(tmp2,2)+1:size(tmp, 2))];
    %Inicializuojama visa populiacija
    population (1, :) = V(1, :);
    population (2, :) = V2(1, :);
    for i = 3 : npop
        population(i, :) = randperm (N);
    end;
end;
```

I.2. Atranka

```
function parents = selection (k, pop, cost, sk);
%funkcija tevu isrinkimui
%k - atrankos metodas
%pop - populiacija
%rid - indeksai pagal kainos funkcijos reiksmes
%sk - skaicius, nurodantis kiek reikia atrinkti tevu
% k == 1 atranka pagal tinkamumo funkcija
% k == 2 eksponentinis metodas
% k == 3 tikimybine atranka
% k == 4 ruletes atranka

%atranka pagal tinkamumo funkcija
if k == 1
%     [costmin, id] = min(cost);
%     pathmin = pop(id, :);
    [csort, rid] = sort(cost); % isrusiuojama didejanciai
    for i = 1 : sk
        parents(i, :) = pop(rid(i), :);
    end;
end;

%eksponentinis metodas
if k == 2
    ecost = sqrt(cost)+1;
    [ecsort, erid] = sort(ecost);
    for i = 1 : sk
        parents(i, :) = pop(erid(i), :);
    end;
end;

%tikimybine atranka
if k == 3
    pt = 0.9;
    [csort, rid] = sort(cost); % isrusiuojama didejanciai
    i = 1; ind = 1;
    while (ind <= sk) & (i <= size(pop, 1))
        if rand <= pt
            parents(ind, :) = pop(rid(i), :);
            ind = ind + 1;
        end;
        i = i + 1;
    end;
    i = 1;
    if ind < sk
        while (ind <= sk)
            parents (ind, :) = pop(rid(i), :);
            i = i + 1;
            ind = ind + 1;
        end;
    end;
end;
```

I.3. Kryžminimas

```
function childs = Crossover (k, N, vstart, vend, parents, sk)
    %individu kryzminimo funkcija.
    %k - populiacijos inicializavimo metodas
    %alg - kuriam uzdaviniui vykdomas kryzminimas
    %N - miestu skaicius (genu skaicius chromosomoje)
    %vstart - pradzios virsune
    %vend - tikslo virsune
    %parents - tevai
    %sk - kiek populiacijos nariu dalyvauja kryzminime

    %k = 1: kryzminimas fiksuojant pradzios ir pabaigos virsuniu pozicijas
    %k = 2: kryzminimas fiksuojant dvieju atsitiktiniu virsuniu pozicijas
    %k = 3: perstatos kryzminimas
    %k = 4: vieno tasko kryzminimas perkoduojant

    j = 1;
    cnt = 0;
    childs = [];
    while (j < size(parents, 1))& (cnt < sk)

    %kryzminimas poromis
    %     mother = parents(j, :)
    %     father = parents(j+1, :)
    %geriausio individo kryzminimas su visais
        mother = parents(1, :);
        father = parents(j+1, :);
        %kryzminimas fiksuojant pradzios ir pabaigos virsuniu pozicijas
        if k == 1
            sameid=[father==mother];
            diffid=find(sameid==0); %randami skirtingi genai

            saf = find ((father(1, :) == vstart)|(father(1, :) == vend));
            sam = find ((mother(1, :) == vstart)|(mother(1, :) == vend));
            dif = find (not(father(1, :) == vstart)&not(father(1, :) == vend));
            dim = find (not(mother(1, :) == vstart)&not(mother(1, :) == vend));

            ii = 1; boy=father;
            for i = 1:N
                if(not(i==saf(1))&&(not(i==saf(2))))
                    boy(i) = mother(dim(ii));
                    ii = ii + 1;
                end
            end
            if cnt < sk
                cnt = cnt + 1;
                childs(cnt, :) = boy;
            end;
            ii = 1; girl=mother;
            for i = 1:N
                if(not(i==sam(1))&&(not(i==sam(2))))
                    girl(i) = father(dif(ii));
```

```
            ii = ii + 1;
        end;
    end;
    if cnt < sk
        cnt = cnt + 1;
        childs(cnt, :) = girl;
    end;
end;
%kryzminimas fiksuojant atsitiktiniu 2 virsuniu pozicijas
if k == 2
    tmp = randperm(N);
    vstart = tmp(1);
    vend = tmp(2);
    sameid=[father==mother];
    diffid=find(sameid==0); %randami skirtingi genai

    saf = find ((father(1, :) == vstart)|(father(1, :) == vend));
    sam = find ((mother(1, :) == vstart)|(mother(1, :) == vend));
    dif = find (not(father(1, :) == vstart)&not(father(1, :) == vend));
    dim = find (not(mother(1, :) == vstart)&not(mother(1, :) == vend));

    ii = 1; boy=father;
    for i = 1:N
        if(not(i==saf(1))&&(not(i==saf(2))))
            boy(i) = mother(dim(ii));
            ii = ii + 1;
        end
    end
    if cnt < sk
        cnt = cnt + 1;
        childs(cnt, :) = boy;
    end;
    ii = 1; girl=mother;
    for i = 1:N
        if(not(i==sam(1))&&(not(i==sam(2))))
            girl(i) = father(dif(ii));
            ii = ii + 1;
        end
    end
    if cnt < sk
        cnt = cnt + 1;
        childs(cnt, :) = girl;
    end;
end

%perstatos kryzminimas
if k == 3
    for i = 1 : N
        ind = i + 1;
        if ind > N
            ind = 1;
        end;
        boy(1, mother(ind)) = father (1, mother(i));
        girl(1, father(ind)) = mother (1, father(i));
    end;
    if cnt < sk
```

```
            cnt = cnt + 1;
            childs(cnt, :) = boy;
        end;
        if cnt < sk
            cnt = cnt + 1;
            childs(cnt, :) = girl;
        end;

    end;
    %vieno tasko kryzminimas perkoduojant
    if k == 4
        f1 = perkoduoti(1, N, father);
        m1 = perkoduoti(1, N, mother);
        splt = randperm(N);
        pnt = splt(1);
        i = 1;
        while ((pnt == 1) | (pnt == N))
            pnt = splt(i);
            i = i + 1;
        end;
        f2 = [f1(1:pnt), m1(pnt+1:N)];
        m2 = [m1(1:pnt), f1(pnt+1:N)];

        if cnt < sk
            cnt = cnt + 1;
            childs(cnt, :) = perkoduoti(2, N, f2);
        end;
        if cnt < sk
            cnt = cnt + 1;
            childs(cnt, :) = perkoduoti(2, N, m2);
        end;
    end;
    j = j + 1;
end;
```

I.4. Mutacija

```
function pop = Mutation(k, N, pop, pm)
%funkcija individu mutavimui. Mutuojami individai mutuojami atsitiktinai su
tikimybe pm
%k - mutacijos metodas
%N - genu skaicius chromosomoje (miestu skaicius)
%pop - chromosomos, kurias numatoma mutuoti
%pm - mutacijos tikimybe

% k == 1 individai mutuojami atsitiktinai permaicius genus
% k == 2 atsitiktinai permaisomi tik skirtingi dvieju parinktu individu genai
% k == 3 atsitiktinai parinkti du genai chromosomoje sukeiciami vietomis
% k == 4 atsitiktinai parenkamas chromosomos dalinimo taskas ir chromosomos
%        dalys sukeiciamos tarpusavyje

%atsitiktinai permaisomi genai
if k == 1
    for i = 1 : N
        if rand <= pm
            pop (i, :) = randperm (N);
        end;
    end;
end;
%atsitiktinai permaisomi tik skirtingi dvieju individu genai
if k == 2
    if rand <= pm
        tmp = randperm(N);
        mother = pop(tmp(1), :);
        father = pop(tmp(2), :);

        sameid=[father==mother]; %randami vienodi genai
        diffid=find(sameid==0);  %randami skirtingi genai

        boy=father.*sameid;
        boy(diffid)=randomize(father(diffid));

        girl=mother.*sameid;
        girl(diffid)=randomize(mother(diffid));

        pop(tmp(1), :) = girl;
        pop(tmp(2), :) = boy;
    end;
end;
%atsitiktinai parenkami du genai chromosomoje ir jie sukeiciami vietomis
if k == 3
    for i = 1 : N
        if rand <= pm
            tmp = randperm(N);
            ch = pop(tmp(1), :);
            tmp2 = randperm (N);
            temp = ch(tmp2(1));
            ch(tmp2(1)) = ch(tmp2(2));
```

```
            ch(tmp2(2)) = temp;
            pop(tmp(1), :) = ch(1, :);
        end;
    end;
end;

%atsitiktinai parenkamas chromosomos dalinimo taskas ir chromosomos dalys
%sukeiciamos tarpusavyje
if k == 4
    for i = 1 : N
        if rand <= pm
            splt = randperm(N);
            f = pop(1, :);
            ch = [f(splt(1)+1:N), f(splt(1)), f(1:splt(1)-1)];
            pop(1, :) = ch(1, :);
        end;
    end;
end;
```