VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Genetinių algoritmų, skirtų neiškiliems daugiakriteriniams uždaviniams spręsti, efektyvumo tyrimas

Investigation of Efficiency of Genetic Algorithms for Non-convex Multiobjective Problems

Magistro baigiamasis darbas

Atliko: Agnė Dzidolikaitė (parašas)
Darbo vadovas: prof. habil. dr. Antanas Žilinskas (parašas)
Recenzentas: doc. dr. Rimantas Vaicekauskas (parašas)

Vilnius – 2012

PADĖKA

Norėčiau padėkoti už visokeriopą pagalbą rengiant magistro darbą savo vadovui prof. habil. dr. Antanui Žilinskui, recenzentui doc. dr. Rimantui Vaicekauskui, tėčiui ir kitiems artimiesiems.

SANTRAUKA

Magistro baigiamajame darbe tiriamas ląstelinių genetinių algoritmų efektyvumas sprendžiant neiškilius daugiakriterius uždavinius. Šie algoritmai santykinai mažai ištirti, bet tikėtinai tinka mus dominančio praktinio uždavinio sprendimui. Tas uždavinys - tai daugiamačių duomenų vizualizavimas daugiamačių skalių metodu. Nagrinėjami trys ląsteliniai genetiniai algoritmai: sMOCell1, sMOCell2, sMOCell3. Trimis skirtingais ląsteliniais genetiniais algoritmais gaunamos Pareto aibės (nedominuojamų sprendinių aibės) ir Pareto aibės taškus atitinkantys penkiamačiai kubai. Gautosios koordinatės atvaizduojamos OpenGL grafine sąsaja. Ištyrus tris ląstelinius genetinius algoritmus, efektyviausias pasirodė sMOCell1, nes pateikė tolygiausiai išsidėsčiusią Pareto aibę. Taip pat šiame magistro darbe taikytas pažangiausias stereoskopijos metodas – autostereoskopija, kai erdvinis vaizdas peržiūrimas tiesiog kompiuterio ekrane, nenaudojant jokių papildomų priemonių. Tam reikalingas specialus ekranas, pritaikytas erdvinių vaizdų peržiūrai ir speciali programinė įranga. Gautieji daugiamačiai vaizdai gali būti peržiūrimi ant erdvinio ekrano.

SUMMARY

This master thesis discusses the efficiency of genetic algorithms for non-convex multiobjective problems. It were chosen cellular genetic algorithms, because they are least investigated. Non-convex multiobjective problems arise in many fields, for instance, in multidimensional data visualization. There are many multidimensional data visualization methods with multidimensional scaling being one of them. Multidimensional scaling is the method of data reduction when objects from higher space are visualized into lower space. The hypercube of five dimensions is visualized from five dimensional space into three dimensional space. Three cellular genetic algorithms are apllied: sMOCell1, sMOCell2, sMOCell3. The fitness functions of these algorithms are quadratic and absolute errors. Pareto sets (non-dominated sets) and five dimensional hypercubes are obtained using three different cellular genetic algorithms. As it was already mentioned, the coordinates of five dimensional hypercubes are received using cellular genetic algorithms. The received coordinates are visualized by OpenGL graphic interface. OpenGL is used for review by stereo display and ordinary display as well. Having investigated three cellular genetic algorithms, sMOCell1 is found to be the most efficient as it gives the most evenly distributed Pareto set. Moreover, in this master thesis has been learned about state of the art stereoscopy method that is called autostereoscopy and requires no extra measures to view images except for computer's display.

TURINYS

ĮVADAS

Optimizavimas yra procesas, kai siekiama ką nors pagerinti, tai yra rasti geriausią sprendinį. Terminas „geriausias sprendinys“ reiškia, kad uždavinys turi daugiau nei vieną sprendinį ir tie sprendiniai yra skirtingos vertės, arba reliatyvūs. Taigi geriausias sprendinys priklauso nuo žmogaus, kuris formuluoja uždavinį. Kai kurie uždaviniai turi konkretų atsakymą. Pavyzdžiui, tiesinė lygtis turi tam tikrą sprendinį. Kiti uždaviniai turi daugybę minimalių ir maksimalių sprendinių, dar vadinamų optimumo taškais, arba ekstremumais.

Optimizavimo tikslas yra turint tam tikrą įeigą arba savybes (įrenginio, matematinio proceso ar eksperimento) rasti minimalią ar maksimalią išeigą (rezultatą). Įeiga susideda iš kintamojo (kintamųjų) - procesas arba funkcija yra vadinama kainos funkcija, tikslo funkcija ar tinkamumo funkcija, o išeiga - kaina ar tinkamumu.

Egzistuoja daugybė optimizavimo uždavinių. Jei yra tik vienas kriterijus, optimizavimas yra vienakriteris. Jei optimizavimo uždavinys turi daugiau nei vieną kriterijų, tai toks uždavinys vadinamas daugiakriteriu optimizavimo uždaviniu. Optimizavimas tampa vis sudėtingesnis augant kintamųjų skaičiui. Dauguma daugiakriterių uždavinių sprendžiami laikantis požiūrio, kad daugiakriteris uždavinys sudarytas iš tam tikro skaičiaus vienakriterių uždavinių.

Daugiakriterių uždavinių iškyla tam tikrose situacijose, kai reikia pasirinkti kurį nors sprendinį iš galimų alternatyvų. Tarkime, egzistuoja projektas, kuriam vykdyti esama skirtingų būdų. Vadinasi, kiekvienas projektuotojas savo praktinėje veikloje turi nagrinėti ir vertinti įvairius projekto variantus. Visi projekto variantai turi savo privalumų ir trūkumų. Paprastai variantas geresnis vieno kriterijaus atžvilgiu, yra blogesnis kurio nors kito kriterijaus atžvilgiu. Ekspertai sprendžia, kokį projekto variantą pasirinkti. Jų darbas yra visapusiškai išnagrinėti esamus projekto variantus ir, remiantis gautais rezultatais, priimti racionalų sprendimą. Tam atlikti reikia optimalumo kriterijų ir daugiakriterio optimizavimo metodų.

Sprendžiant daugiakriterio optimizavimo uždavinius, turinčius keletą konfliktuojančių tikslo funkcijų, paprastai tokie uždaviniai pertvarkomi į uždavinius su viena tikslo funkcija. Egzistuoja įvairių metodų atlikti pertvarkymams. Šie metodai skiriasi savo prigimtimi ir dažnai duoda skirtingus efektyvius sprendinius. Nepaisant to, negalima sakyti, kad vienas iš jų yra geresnis ar blogesnis, nes jie skirti spręsti uždaviniams, kuriems taikomi skirtingi sprendinių nustatymo metodai.

Daugiakriterių uždavinių sprendiniai priimami remiantis matematine teorija ir programine įranga, tačiau sprendimo procesas vis tiek lieka subjektyvus. Minėtosios priemonės skirtos tik palengvinti ekspertų darbą, tai yra pagerinti sprendinių kokybę.

Daugiakriteris optimizavimas taikomas sprendžiant projektavimo, logistikos, ekonomikos, mokslo (bioinformatikos, chemijos, informatikos, matematinio modeliavimo ir t.t.) uždavinius.

Sprendimų priėmimas – dažnai ypatingos svarbos ir atsakomybės veiksmas. Pavyzdžiui, nuo sprendimų sėkmės priklauso ilgametės ekonomikos perspektyvos arba ilgalaikė ekologinė ar socialinė šalies būklė. Dažnai priimant sprendimą tenka lyginti kelias alternatyvas (objektus, planus) ir parinkti geriausią. Gali būti analizuojamas iš anksto pasirinktas alternatyvų rinkinys arba tų alternatyvų gali būti be galo daug.

Sudėtinguose taikomuosiuose uždaviniuose dažnai būna keli ar daugiau kokybės kriterijų. kurie nusako konkrečios alternatyvos savybes ir kuriuos reikia optimizuoti priimant sprendimą. Dažniausiai kriterijai būna prieštaringi – kai vieno kriterijaus reikšmė mažinama, kito gali didėti. Priimant sprendimą kriterijus tenka subalansuoti, pavyzdžiui, pasiekti ir priimtiną kainą, ir kokybę, ir taršą, ir patikimumą. Būna ir situacijų, kai alternatyvą apibūdina ne vien skaitmeniniai kriterijai [DŠT07, 144].

Tarkime, kad turime minimizavimo uždavinį ir reikia priimti sprendimą atsižvelgiant į kelis kriterijus, kurių reikšmes galima apskaičiuoti, t. y. įvertinti kiekybiškai.

Toks uždavinys formuoluojamas taip:

$$\min f(X)$$

esant apribojimui $X \in D \subset R^n$.

$X = (x_1, x_2, \dots, x_n)$ yra n komponenčių vektorius, kuriuo aprašoma alternatyva.

$f(X) = (f_1(X), f_2(X), \dots, f_m(X))$ – kriterijų vektorius,

m – kriterijų skaičius.

Sritis D gali būti aprašoma taip:

$$D = \{X: G(X) \geq 0\},$$

čia $G(X) = (g_1(X), g_2(X), \dots, g_k(X))$ – apribojimų funkcijų vektorius, k – apribojimų skaičius.

Kadangi kriterijai gali vienas kitam prieštarauti, todėl sunku pasakyti, kuri alternatyva (sprendinys) geriausia pagal visus kriterijus.

Sprendinių kokybei palyginti įvedamas dominavimo santykis: sprendinys X_1 dominuoja sprendinį X_2, t. y. $X_1 \succ X_2$, jei $f_i(X_1) \leq f_i(X_2)$, $i = 1, 2, \dots, m$ ir egzistuoja bent vienas j, kad $f_j(X_1) < f_j(X_2)$.

Sprendinys X^* yra nedominuojamas, jei jo nedominuoja koks kitas sprendinys. Toks sprendinys X^* vadinamas efektyviu sprendiniu, arba Pareto optimaliu sprendiniu.

Dažniausiai uždavinys turi ne vieną efektyvų sprendinį. Formaliai jie visi tinkami. Efektyvių sprendinių aibė vadinama Pareto aibe.

Tik konkretų sprendimą priimantis ekspertas gali nutarti, kuris Pareto aibės taškas (elementas) priimtinas, o kuris – ne [DŠT07, 144-146].

Sprendžiant daugiakriterio optimizavimo uždavinius, galima taikyti genetinius algoritmus. Genetiniai algoritmai priklauso evoliucinių algoritmų šeimai.Genetinių algoritmų sprendinių kintamieji koduojami baigtinio ilgio kintamųjų sekomis, vadinamomis chromosomomis. Kiekviena chromosoma yra galimas uždavinio sprendinys. Chromosomas sudarantys kintamieji vadinami genais, o skirtingos genų reikšmės – aleliais.

Genetinių algoritmų sudedamosios dalys yra tokios:

1) Inicializacija – atsitiktinai generuojama pradinė galimų sprendinių populiacija.
2) Vertinimas – inicializavus populiaciją arba, kai naujas sprendinys (palikuonis) yra sukurtas, apskaičiuojamos galimų sprendinių tinkamumo reikšmės.
3) Selekcija – vykdant paiešką palaikomi aukščiausias tinkamumo reikšmes įgiję sprendiniai, tai yra sukuriamas mechanizmas, remiantis geriausių individų išlikimą.
4) Rekombinacija – sujungiamos dvi ar daugiau tėvinių sprendinių dalių, kad būtų sukurti nauji sprendiniai (palikuonys), kurie galbūt yra geresni. Efektyvų algoritmo darbą lemia tinkamai parinktas rekombinacijos mechanizmas, kuris gali būti pasiektas įvairiais būdais, geriausia būtų, kad palikuonis, gautas rekombinacijos būdu, nebūtų identiškas nė vienam iš tėvų.
5) Mutacija – atsitiktinai modifikuojamas vienas sprendinys (rekombinacijoje dalyvauja dvi ar daugiau tėvinių chromosomų), egzistuoja daugybė mutacijos variantų, kurie paveikia vieno ar daugiau genų dalį komponenčių.
6) Pakeitimas – vaikinė populiacija, sukurta vykdant selekciją, rekombinaciją ir mutaciją, pakeičia tėvinę populiaciją pagal duotą kriterijų. Populiarus kriterijus yra elitinis pakeitimas, kai evoliucijos metu išsaugomas geriausias sprendinys.

Pagrindinė ypatybė, apibūdinanti ląstelinius genetinius algoritmus yra genetinių operatorių (selekcijos, rekombinacijos, mutacijos ir pakeitimo) taikymas kaimynystės viduje, tai yra individams, priklausantiems skirtingoms kaimynystėms, neleidžiama sąveikauti.

Kaimynystėse individai gali būti įvairiai išsidėstę. Žemiau esančiame paveiksle pateikiama dažniausiai naudojamų kaimynysčių sandara. Juodi ir žali apskritimai žymi kaimynystę sudarančius inidividus, raudoni apskritimai – populiacijai priklausantys kitų kaimynysčių nariai.

1 pav. Įvairios kaimynystės (ląsteliniuose genetiniuose algoritmuose)

Magistro baigiamajame darbe analizuojamas genetinių algoritmų efektyvumas sprendžiant daugiakriterius uždavinius. Šiame darbe siekiama ištirti genetinių algoritmų efektyvumą sprendžiant daugiakriterius uždavinius.

Sprendžiant uždavinį turėtų būti sujungta:

1) Genetinis algoritmas daugiakriteriam uždaviniui spręsti.
2) Daugiamačių duomenų vizualizavimo metodas (pasirinktos daugiamatės skalės - DS);
3) Grafinis vizualizavimas.

Daugiamačių duomenų vizualizavimo metodas (daugiamatės skalės) atvaizduoja duomenis iš didesnės dimensijos erdvės į mažesnės dimensijos erdvę.

Grafiniam vizualizavimui pasirinkta OpenGL grafinė biblioteka.

1. Metaeuristinių metodų taikymas daugiakriteriams uždaviniams spręsti

1.1.Euristikos ir metaeuristikos optimizavime

Dabar labai populiarūs tikslių algoritmų, euristikų ir metaeuristikų tyrimai, skirti spręsti kombinatorinio optimizavimo uždaviniams. Pagrindinis tikslių algoritmų privalumas yra tas, kad jie randa globalų optimumą, bet esminis trūkumas – eksponentiškai didėjantis uždavinio sprendimo laikas augant uždavinio dydžiui. Kita vertus, euristiniai algoritmai dirba labai greitai, tačiau jų sprendinių kokybė nėra labai gera. Priešingai nei euristikos, metaeuristikos duoda ir pakankamai gerą sprendinį (kartais net globalų optimumą), ir baigia darbą per priimtiną laiko tarpą.

Suformuluokime optimizavimo uždavinį minimizavimo atvejui. Optimizavimo uždavinys (S, f), kur $S \neq \emptyset$ žymi leistinąją sritį, o f - tikslo funkciją apibrėžiamas taip:

$$f: S \rightarrow \mathbb{R}.$$

Sprendžiant optimizavimo uždavinį randama tokia kintamųjų reikšmių aibė, kad šios aibės reikšmės $\boldsymbol{i}^* \in S$ tenkina žemiau esančią nelygybę:

$$f(\boldsymbol{i}^*) \leq f(\boldsymbol{i}), \quad \forall\, \boldsymbol{i} \in S.$$

Du sprendiniai yra arti vienas kito, jei jie priklauso tai pačiai kaimynystei leistinojoje srityje. Apibrėžkime kaimynystės sąvoką. Jei (S, f) yra optimizavimo uždavinys, tai kaimynystė gali būti apibrėžta taip:

$$N: S \rightarrow 2^S,$$

kad kiekvienam sprendiniui $\boldsymbol{i} \in S$ apibrėžta aibė $S_i \subseteq S$. Taip pat, jei sprendinys $\boldsymbol{i}$ yra sprendinio $\boldsymbol{j}$ kaimynystėje, tada $\boldsymbol{j}$ yra sprendinio **i** kaimynystėje: $\boldsymbol{j} \in S_i$, jei $\boldsymbol{i} \in S_{\boldsymbol{j}}$.

Sprendžiant sudėtingas optimizavimo problemas, dažnai gaunamas sprendinys, kuris optimalus savo kaimynystėje, bet ne visoje leistinojoje srityje. Taigi paieškos metodas lengvai gali „įstrigti“ optimalioje kaimynystės reikšmėje, arba lokaliajame optimume. Apibrėžkime lokaliojo optimumo sąvoką.

Jei (S, f) yra optimizavimo uždavinys ir $S_{i'} \subseteq S$ yra sprendinio $i' \in S_{i'}$ kaimynystė, tai sprendinys i' yra lokalus optimumas, jei tenkinama žemiau esanti nelygybė (minimizavimo atveju):

$$f(i') \leq f(i), \quad \forall i \in S_{i'}.$$

Taikant tikslius algoritmus randamas globalusis optimumas visiems baigtiniams atvejams. Kadangi augant sudėtingo uždavinio dydžiui, tiksliems metodams reikia eksponentinio laiko, tai tikrovėje šie algoritmai negali išspręsti NP – pilnų uždavinių. Tokiems uždaviniams spręsti naudojamos įvairiausios aproksimacinės technikos, kurios negarantuoja, kad visada bus rastas optimalus sprendinys, tačiau pateikiami pakankamai geri sprendiniai per gerokai trumpesnį laiką nei dirbant su tiksliais metodais.

Per pastaruosius du dešimtmečius pasirodė įvairių aproksimavimo technikų, jungiančių įvairius euristinius metodus, kad efektyviai atliktų paiešką leistinojoje srityje. Tokie metodai vadinami metaeuristikomis. Tai yra aukšto lygio strategijos, kurios turi tam tikrą struktūrą, kuri suplanuoja tam tikrą operacijų aibę tam, kad tirtų daugiamates ir sudėtingas leistinąsias sritis. Vertėtų apibrėžti ir euristikos sąvoką, nes metaeuristikos gaunamos kombinuojant euristikas.

Euristika – tai praktikos padiktuota pasirinkimo taisyklė (angl. *rule of thumb*), kuri leidžia rasti sprendimą be varginančios paieškos (paprastai suprantamos kaip perrinkimas). Būdvardis euristinis (angl. *heuristic*) apibūdina kiekvieną sprendimo būdą, kuris pagerina uždavinio

sprendimo vidutinę charakteristiką, bet nebūtinai blogiausio atvejo charakteristiką. Euristikos sąvoka kildinama iš graikiško žodžio „heuriskein“ (atrasti). Legenda byloja, kad senovės graikų filosofas Archimedas sušuko „eureka“ (atradau), kai suvokė apie jėgą, kuri išstumia kietą kūną, panardintą į skystį. Dabar ši jėga vadinama Archimedo jėga.

Euristika gali būti grindžiama ir ankstesniu patyrimu. Jeigu panašus uždavinys jau buvo spręstas anksčiau arba netgi ankstesnis sprendimas buvo nesėkmingas, tai patirtis gali padėti.

Euristika skirta pasiūlyti sprendimą, bet be garantijos. Kadangi neatliekama gili analizė, tai gali būti pateikiamas ir neteisingas sprendimas arba jo iš viso nerandama. Nesėkmė pagrįsta tuo, kad euristika pateikia „priimtiną“ variantą, o ne teisingą atsakymą. Euristika gali būti labiau priimtinas variantas negu tikslus sprendimas šiais atvejais:

1) Yra per daug variantų, kuriuos reikia patikrinti.

2) Kiekvieno varianto įvertinimo funkcija yra per daug sudėtinga.

3) Įvertinimo funkcija yra nežinoma ir apytikslis įvertinimas gali atlikti euristikos vaidmenį[Čyr08, 28-29].

1.2.Genetiniai algoritmai

Kadangi klasikiniai metodai daugiakriteriams uždaviniams spręsti ne visada yra tinkami, tai taikomi genetiniai algoritmai bei kiti metaeuristiniai metodai.

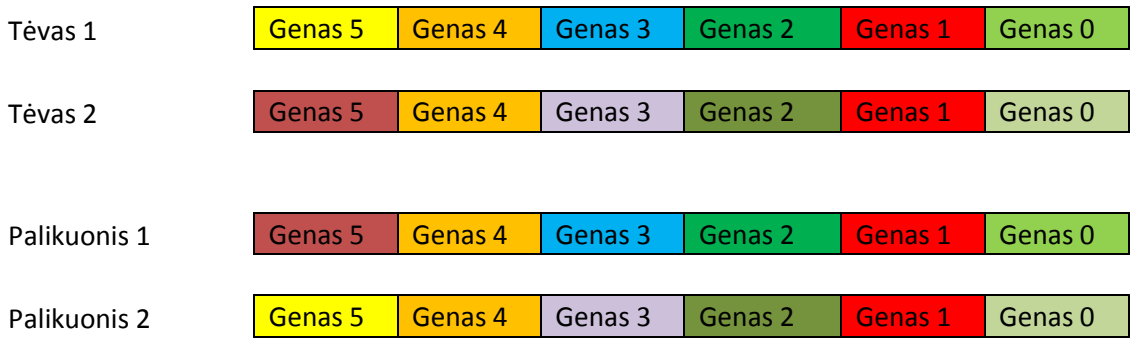
Genetiniai algoritmai pagrįsti natūralia atranka, stebima gamtoje. Šie algoritmai siekia iš senesnių sprendinių (tėvų) gauti geresnius naujus sprendinius (palikuonis). Tai primena gamtą, kur išlieka individai, turintis geresnes savybes, tai yra geriau adaptuojasi prie aplinkos.

Viena iš pagrindinių genetinių algoritmų sąvokų yra populiacija. Gamtoje populiacija laikoma kokios nors vienos rūšies gyvųjų būtybių aibė, gyvenanti tam tikroje vietovėje. Populiacijos individai vienas nuo kito skiriasi tam tikromis savybėmis, pavyzdžiui, akių spalva, ūgiu, ragų aštrumu ir taip toliau. Minėtąsias gyvųjų būtybių savybes lemia organizmų ląstelėse esančios chromosomos, kurios sudarytos iš genų, nustatančių konkrečias individo savybes. Skirtingos genų reikšmės vadinamos aleliais. Nuo alelių ir priklauso, ar individo, tarkime žmogaus, plaukai bus lygūs ar garbanoti, nosis tiesi ar su kuprele.

Individo savybės, kurios koduojamos skirtingomis chromosomose esančių genų reikšmėmis, apibrėžia pastarojo prisitaikymą, arba gebėjimą išlikti ir susilaukti palikuonių. Žinoma, ir labiausiai prisitaikęs (stipriausias) individas gali neišgyventi dėl įvairiausių priežasčių, pavyzdžiui, tigras susižeidžia medžiodamas ir žūsta. Tad reikia suprasti, kad geriausią chromosomų rinkinį turinčiam individui yra didžiausia galimybė išlikti, bet nebūtinai taip ir atsitinka.

Labiausiai prisitaikę organizmai turi didžiausią galimybę išlikti ir perduoti savo genus palikuonims. Prasčiau prisitaikiusiems individams galimybė išlikti mažesnė. Tokiems individams paprastai sunkiau susirasti partnerį poruotis dėl prastesnio prisitaikymo. Kaip jau minėta, taip vyksta natūrali atranka, arba selekcija.

Gamtoje, kai atrenkami geriausi individai poruotis, toliau vykdoma rekombinacija. Rekombinacijos metu imamos tėvų chromosomos, jų genai sukeičiami vienoje ar keliose vietose ir gaunami nauji individai – palikuonys. Žemiau esančiame paveiksle pavaizduota po vieną chromosomą, priklausančią kiekvienam iš dviejų tėvų. Kiekviena tėvinė chromosoma šiame pavyzdyje turi po šešis genus, koduojančius tam tikrą individo požymį. Tarkime, kad „Genas 4“ reiškia plaukų spalvą. Minėtasis genas, pažymėtas oranžiniu stačiakampiu abiems tėvams, vadinasi, šio geno lemiama savybė yra vienoda ir abu tėvai yra šviesiaplaukiai arba tamsiaplaukiai. Tarkime, kad „Genas 2“ lemia nosies ilgį, tačiau šiuo atveju tėvų genų reikšmės skirtingos ir vieno iš individų nosis ilga, o kito – trumpa. Rekombinacijos metu pavaizduotos tėvinės chromosomos padalinamos pusiau ties ketvirtuoju bei trečiuoju genais, o paskui sujungiama: chromosomos atkarpa iš pirmojo tėvo su atkarpa iš antrojo tėvo, kol gaunami du nauji palikuonys. Šiuo atveju palikuonys nėra identiški nė vienam iš tėvų (tai matome iš skirtingų spalvų rinkinių, atitinkančių skirtingas genų kombinacijas). Idealiu atveju, individai, gauti rekombinacijos būdu, pasižymi geresnėmis savybėmis nei jų tėvai, tačiau taip ne visada nutinka, kartais palikuonių prisitaikymas yra prastesnis nei tėvų.



2 pav. Rekombinacija

Rekombinacijos metu gautų palikuonių genuose gali vykti mutacijos. Tai reiškia, kad tam tikros genų komponentės atsitiktinai bus pakeistos, nulemdamos naujas savybes. Mutavę genai gali pagerinti arba pabloginti individo prisitaikymą. Pavyzdžiui, palikuonis turi geną, pavaizduotą geltonu stačiakampiu, reiškiantį ragų aštrumą. Tarkime, kad po atsitiktinės

mutacijos minėtasis genas taps rožiniu stačiakampiu ir lems aštresnius ragus bei geresnį individo prisitaikymą.

Genetinis algoritmas mėgdžioja gyvąją gamtą. Čia svarbiausia dalis yra rekombinacija. Tačiau algoritmo darbas pradedamas nuo inicializacijos, kai sukuriamas atsitiktinis individų sąrašas, kur kiekvienas sąrašo elementas, reiškia tam tikrą individą, arba uždavinio sprendinį. Dažniausiai pradinė populiacija sudaroma iš šimtų ar tūkstančių atsitiktinių individų, kurių prisitaikymas yra prastas.

Vėliau sugeneruota populiacija įvertinama pagal kiekvieno nario prisitaikymą. Individo prisitaikymą galima skaičiuoti įvairiai. Laikykime, kad duotasis individas koduoja lygties sprendinį. Vadinasi, kuo individo pateiktas sprendinys artimesnis lygties sprendiniui, tuo didesnė jo prisitaikymo reikšmė ir atvirkščiai.

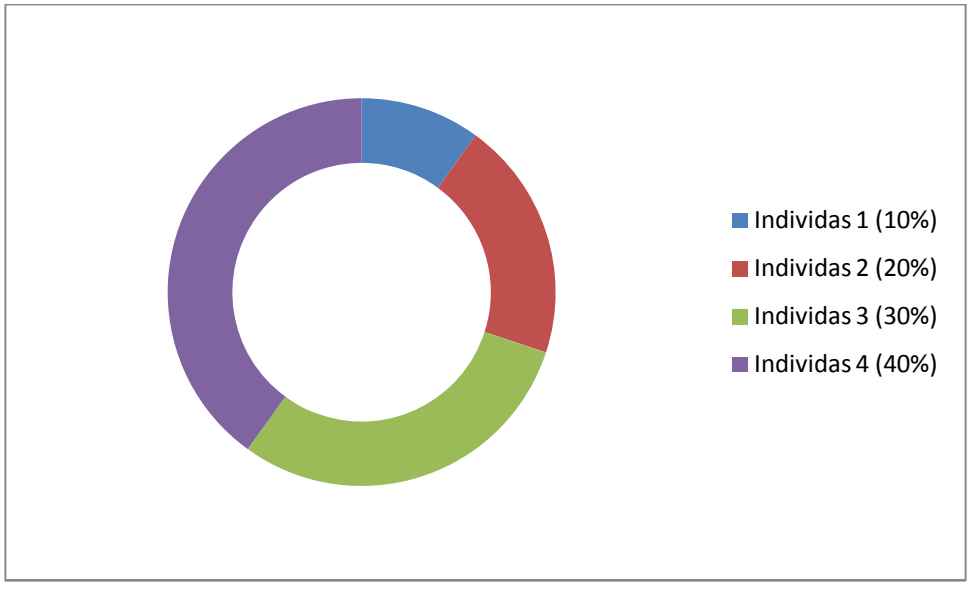
Įvertinus populiacijos individus vykdoma selekcija – atrenkami tėvai rekombinacijai. Dažniausiai naudojami šie selekcijos metodai:

1) Ruletės selekcija.
2) Turnyro selekcija.

Ruletės selekcijos metodu įvertinamas kiekvieno individo prisitaikymas. Tada apskaičiuojama, kokią bendro prisitaikymo dalį (pocentais) sudaro šio individo prisitaikymas. Kuo didesnis individo prisitaikymas, tuo didesnė tikimybė jam būti atrinktam. Ruletės selekcija vykdoma atsitiktinai, o tikimybė konkrečiam individui būti parinktam tėvu lygi:

$$P = \frac{f_i}{\sum_{j=1}^{n} f_j}.$$

Čia f_i - konkretaus individo prisitaikymas, $\sum_{j=1}^{n} f_j$ – visų populiacijos individų prisitaikymas.



3 pav. Ruletės selekcija

Turnyro selekcijos metodu selekcijai atsitiktinai parenkama k individų. Geriausias individas parenkamas su tikimybe p. Antras geriausias individas – su tikimybe $p * (p - 1)$. Trečias geriausias individas – su tikimybe $p * (p - 1) * (p - 1)$ ir taip toliau.

Atlikus selekciją, sukuriami palikuonys iš tėvinių individų. Šis procesas vadinamas reprodukcija ir yra sudarytas iš rekombinacijos ir mutacijos.

Egzistuoja įvairių rekombinacijos variantų, tačiau dažniausiai taikomos šie:

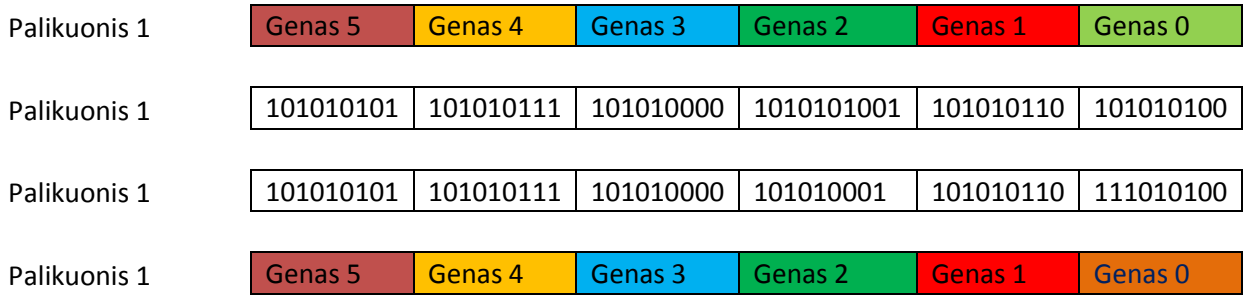
1) Rekombinacija, kai tėvinės chromosomos padalinamos į dvi dalis.
2) Rekombinacija, kai tėvinės chromosomos padalinamos į daugiau nei dvi dalis.

Žemiau esančiame paveiksle pavaizduoti palikuonys gauti tėvines chromosomas padalinus viename taške ir dviejuose taškuose. Kiekviena tėvinė chromosoma sudaryta iš šešių genų, pavaizduotų mėlynais ir raudonais stačiakampiais, kad būtų lengviau pastebėti, kuriose vietose buvo padalytos chromosomos. Kiekvienas genas koduoja skirtingą individo savybę.

4 pav. Rekombinacijos būdu gauti palikuonys, padalinus tėvines chromosomas viename taške ir dviejuose taškuose

Mutacijos metu atsitiktinai pakeičiama viena ar daugiau geno komponenčių ir gaunama nauja individo savybė. Galima geno komponentes vaizduoti nulių ir vienetų sekomis. Tada mutacijos atveju nulis keičiamas vienetu, o vienetas – nuliu. Žemiau pavaizduotame paveiksle matoma rekombinacijos būdu gauto individo chromosoma, sudaryta iš šešių genų. Kiekvienas

genas – skirtingos spalvos stačiakampis. Šie genai taip pat pavaizduoti nulių ir vienetų sekomis. Mutacijos metu pakeičiama „Genas 0“ antra komponentė – nulis tampa vienetu. Gaunama nauja individo savybė, kurią rodo kitokia spalva pažymėtas genas. Tarkime, kad ankstesnė geno reikšmė kodavo trumpas kojas, o nauja – ilgas kojas. Individas, turintis ilgas kojas turi didesnę tikimybę išlikti ir susilaukti palikuonių. Taigi šiuo atveju mutacija buvo naudinga. Pasitaiko mutacijų dėl kurių individo prisitaikymas suprastėja (degraduoja).



5 pav. Mutacija

Po mutacijos vykdomas pakeitimas, tai yra palikuonių populiacija pakeičia dalį arba visą tėvų populiaciją. Populiarus elitistinis pakeitimas, kai išsaugomi labiausiai prisitaikę individai. Dažnai atrenkama apie 10% prasčiausiai prisitaikiusių individų iš tėvinės populiacijos ir jie pakeičiami individais iš palikuonių populiacijos.

Genetinis algoritmas baigia darbą, kai tenkinama pabaigos sąlyga, pavyzdžiui pasiektas užsiduotas pakankamai geras sprendinys arba maksimalus kartų skaičius.

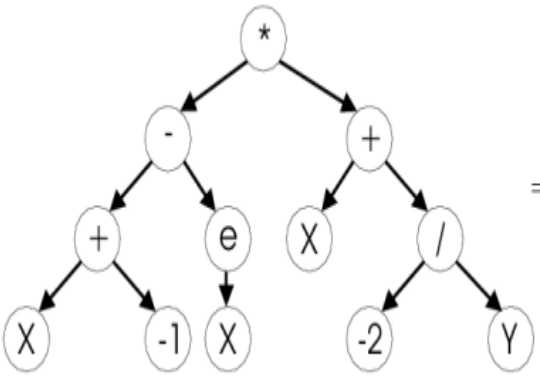
Toliau pateikiamas genetinio algoritmo pseudokodas, kurio žingsniai buvo išsamiai aprašyti:

```
SukurtiPradinęPopuliaciją();
ĮvertintiPrisitaikymą (populiacija);
  while ! PabaigosSąlyga do
     for individas ← 1 to populiacijos_dydis do
          tėvai ← Selekcija (populiacija);
          palikuonis ← Rekombinacija (P′, tėvai);
          palikuonis ← Mutacija (P″, palikuonis);
          ĮvertintiPrisitaikymą (palikuonis);
          Pakeitimas (pozicija (individas), papildoma_populiacija, palikuonis)
       end for
       populiacija ← papildoma_populiacija
```

end while

1.3.Genetinis programavimas

Genetinis programavimas yra evoliucinių algoritmų šaka. Pagrindinis skirtumas tarp genetinio programavimo ir genetinių algoritmų yra sprendinių vaizdavimas. Genetiniuose algoritmuose sprendiniai vaizduojami eilutės pavidalo chromosomomis, o genetiniame programavime sprendinius atitinka medžiai.



$$\Rightarrow (x - 1 - e^x) \cdot (x + \frac{-2}{y})$$

6 pav. Medžio pavyzdys [Zel11]

Genetiniame programavime naudojami keturi žingsniai, kad išspręsti uždavinį:

1) Generuojama pradinių sprendinių populiacija (kompiuterio programas atitinkantys medžiai).
2) Vykdoma kiekviena sugeneruota programa ir jai priskiriama prisitaikymo reikšmė pagal tai, kaip gerai ji sprendžia uždavinį.
3) Sukuriama naujų kompiuterio programų populiacija.
    i. Kopijuojamos geriausios egzistuojančios programos.
    ii. Sukuriamos naujos kompiuterio programos atliekant mutaciją.
    iii. Sukuriamos naujos kompiuterio programos atliekant rekombinaciją.
4) Geriausia iš visų sukurtų programų, pasirodžiusi kurioje nors kartoje, laikoma genetinio programavimo rezultatu.

Sudėtingiausia ir svarbiausia genetinio programavimo sąvoka yra prisitaikymo funkcija. Prisitaikymo funkcija apibrėžia, kaip gerai programa sprendžia uždavinį. Prisitaikymo funkcija skirtingoms programoms gali labai skirtis. Pavyzdžiui, jei reikia sukurti programą, kuri skirta

nustatyti laikui, prisitaikymo funkcija bus lygi laikui, kai laikrodis rodo neteisingai. Žinoma, daugumos programų prisitaikymo funkcijos gerokai sudėtingesnės.

Funkcijos ir terminalai yra svarbūs genetinio programavimo komponentai. Terminalų ir funkcijų aibės yra programų, kurios turi būti sukurtos, abėcėlė. Terminalų aibė susideda iš programos kintamųjų ir konstantų. Funkcijų aibė susideda iš programos funkcijų. Funkcijos yra kelios matematinės funkcijos, tokios kaip sudėtis, atimtis, daugyba, dalyba ir kitos sudėtingesnės funkcijos.

Genetiniame programavime egzistuoja dvi pagrindinės operacijos, skirtos modifikuoti struktūroms – tai rekombinacija ir mutacija. Rekombinacija yra svarbiausia genetinio programavimo operacija. Rekombinacijos metu iš dviejų tėvinių sprendinių gaunami du nauji sprendiniai – palikuonys. Tėvai iš populiacijos atrenkami pagal jų prisitaikymą. Egzistuoja trys metodai atrinkti sprendinius rekombinacijos operacijai.

Pirmasis metodas (ruletės metodas) naudoja tikimybę, paremtą sprendinio prisitaikymu. Jei $f(s_i(t))$ yra sprendinio prisitaikymas ir $\sum_{j=1}^{M} f(s_j(t))$ yra visų populiacijos narių prisitaikymų suma, tada tikimybė, kad sprendinys s_i atsiras kitoje kartoje lygi:

$$\frac{f(s_i(t))}{\sum_{j=1}^{M} f(s_j(t))}$$

Kitas selekcijos metodas yra turnyro selekcija. Šiuo atveju atsitiktinai parenkami du sprendiniai. „Laimi“ tas sprendinys, kurio didesnė prisitaikymo reikšmė. Turnyro selekcijos metodas simuliuoja gyvojoje gamtoje stebimą poravimąsi, kur du tos pačios lyties populiacijos nariai varžosi, kad galėtų poruotis su trečiu priešingos lyties populiacijos nariu.

Trečiasis selekcijos metodas yra selekcija pagal rangą. Rango selekcijos atveju, kiekvienam sprendiniui suteikiamas rangas pagal jo prisitaikymą. Pavyzdžiui, sprendinys, kurio prisitaikymo reikšmė mažiausia, įgaus rangą lygų vienetui. Suskirsčius sprendinius į rangus, atsitiktinai parenkami tėviniai sprendiniai.

Rekombinacijos metu palikuonys gaunami iš vieno tėvo (medžio) pašalinus fragmentą ir įterpus jį į kitą tėvą.

Svarbi genetinio programavimo ypatybė yra ta, kad iš to paties sprendinio galima sukurti du visiškai naujus sprendinius. To negalima padaryti taikant genetinius algoritmus.

Genetiniame programavime mutacijas galima atlikti dviem būdais. Pirmuoju atveju funkcija gali pakeisti funkciją arba terminalas terminalą. Antruoju atveju visas pomedis gali būti pakeistas atsitiktinai sugeneruotu pomedžiu.

Genetinis programavimas yra daug galingesnis metodas nei genetiniai algoritmai. Genetinių algoritmų išvestis yra kiekybinė reikšmė, o genetinio programavimo – nauja kompiuterio programa. Tai gali būti pradžia kompiuterių programų, kurios pačios kuria naujas programas – programuoja.

Genetinis programavimas geriausiai tinkamas spręsti kelių rūšių uždaviniams. Pirmasis uždavinių tipas yra uždaviniai, kuriems nėra idealaus sprendinio, pavyzdžiui programa, kuri vairuoja automobilį. Neegzistuoja tokio sprendinio, kuris nurodytų, kaip vairuoti automobilį. Pagal kai kuriuos sprendinius vairuojama saugiai, bet sugaištama daugiau laiko, pagal kitus sprendinius vairuojama greitai, bet nesaugiai. Taigi vairuojant automobilį daromi kompromisai tarp greičio ir saugumo, taip pat ir kompromisai tarp kitų kintamųjų. Genetinis programavimas padeda rasti sprendinį, kuris ir eina į kompromisą, ir yra efektyviausias iš visų galimų sprendinių.

Genetinis programavimas taip pat tinka spręsti uždaviniams, kur kintamieji nuolat keičiasi. Imkime automobilį vairuojančios programos pavyzdį. Programa ras vienokį sprendinį, kai važiuojama lygia automagistrale ir kitokį esant negrįstam keliui.

1.4.Metrikos, skirtos palyginti genetiniams algoritmams

Lyginant genetinius optimizavimo algoritmus, naudinga įsivesti papildomų sąvokų iš statistikos, tokių kaip vidutinis tinkamumas, populiacijos entropija, taip pat keletą matematinių funkcijų, tokių kaip standartinis nuokrypis, mediana ir kvartilių skirtumas.

Pirmiausia apibrėžkime vidutinį tinkamumą $\bar{f}$. Vidutinis tinkamumas individų iš N dydžio populiacijos P apibrėžiamas kaip vidurkis tinkamumo reikšmių sumos, sudarytos iš visų individų, sudarančių populiaciją, tinkamumo reikšmių:

$$\bar{f} = \frac{\sum_{i=1}^{N} f_i}{N},$$

kur f_i yra individo i tinkamumo reikšmė.

Kita svarbi sąvoka – populiacijos entropija H_p. Populiacijos entropija H_p yra apskaičiuojama kaip entropijų vidurkis visų individų visų chromosomų genui:

$$H_p = \frac{\sum_{i=1}^{L} H_i}{L},$$

kur H_i yra reikšmių, sudarančių geną, vidurkis L ilgio chromosomos pozicijoje i.

Dar viena sąvoka, kurią reikia apibrėžti yra standartinis nuokrypis. Tačiau prieš apibūdinant standartinį nuokrypį, turime žinoti, ką reiškia dispersija.

Dispersija, arba imties, sudarytos iš n stebėjimų $X_1, X_2, \dots, X_n$, vidutinis kvadratinis nuokrypis, apibūdinamas taip:

$$S = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1},$$

kur $\bar{X}$ yra nagrinėjamos imties aritmetinis vidurkis.

Tada standartinis kvadratinis nuokrypis išreiškiamas, naudojantis dispersijos apibrėžimu. Standarinis kvadratinis nuokrypis lygus kvadratinei šakniai iš dispersijos:

$$stdv = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}}.$$

Dar viena sąvoka – mediana. Mediana (M) – tai skaičius, padalijantis imtį į dvi dalis: apatinę ir viršutinę. Norėdami ją rasti, pirmiausia išrikiuojame imties elementus didėjimo tvarka (imtį sutvarkome), paskui randame skaičių (ar du tokius skaičius), esantį sutvarkyto sąrašo viduryje:

a) kai imties tūris (imties tiriamų individų arba objektų skaičius) n - nelyginis skaičius, vidurinio skaičiaus numeris yra $\frac{n+1}{2}$, todėl tas skaičius ir yra mediana;

b) kai imties tūris n - lyginis skaičius, imami du viduriniai skaičiai $\frac{n}{2}$ ir $\frac{n}{2}+1$. Mediana yra tų dviejų skaičių aritmetinis vidurkis.

Pavyzdžiui, sutvarkytos imties $3, 4, 5, 7, 8$ mediana $M = 5$, o imties $2, 4, 6, 8, 10, 12$ mediana $M = \frac{6+8}{2} = 7$.

Pirmasis kvartilis (K_1) yra imties apatinės pusės mediana, trečiasisis kvartilis (K_3) – viršutinės pusės mediana.

Pavyzdžiui, sutvarkytos imties $1, 2, 3, 4, 5, 6, 7, 8, 9, 10$ mediana $M = \frac{5+6}{2} = 5{,}5$, pirmasis kvartilis $K_1 = 3$, trečiasis kvartilis $K_3 = 8$.

Kvartilių skirtumas taikomas apskaičiuoti imties reikšmių dispersijai. Jis paprastai apskaičiuojamas kaip trečiojo ir pirmojo kvartilių skirtumas:

$$IQR = K_3 - K_1.$$

1.5.Genetinių algoritmų praktinis pritaikymas

Genetinis algoritmas – tai metaeuristinis paieškos metodas, taikomas informatikoje, kad būtų rasti optimalūs sprendiniai. Šis metodas įkvėptas evoliucinės biologijos. Yra daugybė genetinių algoritmų panaudojimo sričių.

Taikant genetinius algoritmus projektuojant lenktyninių automobilių ir įprastų transporto priemonių (įskaitant lėktuvus) medžiagas bei aerodinamines formas, galima gauti greitesnes, lengvesnes, mažiau kuro naudojančas ir saugesnes transporto priemones. Užuot praleidus ištisus metus laboratorijose, dirbant su polimerais, vėjo tuneliais ir balzos medienos modeliais, procesą galima pagreitinti kompiuteriu modeliuojant sprendinius genetiniais algoritmais.

Genetinių algoritmų taikymas pastatų, gamyklų, įvairiausių mechanizmų projektavimui, sparčiai plečiasi. Optimizuojamas robotizuotų rankų, gervių, smagračių, turbinų projektavimas. Genetiniai algoritmai taikomi ne tik minėtiesiems objektams projektuoti, bet ir analizuoti jų silpnosioms vietoms, kad būtų galima išvengti įvairiausių gedimų.

Robotų projektuotojai ir inžinieriai išbando daugybę variantų, kad sukurtų naudingas mašinas, galinčias atlikti darbą už žmones. Kiekvieno roboto projektas priklauso nuo darbo ar darbų, kuriuos jis turi atlikti. Genetiniai algoritmai gali būti programuojami ieškoti optimalių projektų arba pateikti visiškai naujų robotų projektus, kurie atlitktų daugybę užduočių vienu metu.

Evoliucionuojanti techninė įranga yra elektros grandinės, sukurtos taikant genetinius algoritmus, kurios naudoja stochastinius (statistiškai atsitiktinius) operatorius, kad gautų naujas konfigūracijas iš senų. Algoritmui dirbant, gaunama tokia konfigūracija, kokios pageidauja projektuotojas. Apie persikonfigūruojančias grandines galima mąstyti kaip apie kosmoso robotą. Jis gali naudotis genetinių algoritmų biblioteka ir simuliatoriumi, kad persiprojektuotų, kai pažeidžiama normali konfigūracija arba, kai susiduriama su situacija, kai reikalinga funkcija, kurios nėra. Genetiniai algoritmai įgalintų adaptaciją ir savęs taisymą.

Yra programų, skirtų kurti kalambūrams ir juokams. Jas naudoja komikai, žmonės, dirbantys su vaikais, turinčiais bendravimo problemų. Tokios programos vadinamos „dirbtine kūryba“. Pavyzdžiui, vartotojas įveda žodį ir programa sukuria kalambūrą (žodžių žaismą), naudodama duotąjį žodį. Čia irgi galima panaudoti genetinius algoritmus.

Biomimikrija, arba biomimetika – disciplina, kurianti technologijas, kurių projektai mėgdžioja gamtą. Čia ir vėl pasitarnauja genetiniai algoritmai, analizuodami gamtos pavyzdžiu paremtus projektus ir jungdami juos, kad būtų gauta kažkas visiškai naujo ir turinčio praktinį pritaikymą.

Genetiniai algoritmai gali būti taikomi spręsti keliaujančio pirklio uždaviniui. Taip pat gerai žinomas maršrutų uždavinys, kai norima tam tikram miestų skaičiui pristatyti krovinius ar prekes. Praktikoje kelionių agentūros gali taikyti genetinį algoritmą, kad sudarytų optimalų turistinės kelionės maršrutą.

Genetiniai algoritmai gali būti taikomi kompiuteriniuose žaidimuose, kur žmogus žaidžia su kompiuteriu. Genetiniais algoritmais pagrįstos programos „mokosi“ iš geriausių žaidimų partijų žaisdamos su žmogumi.

Naujų cheminių molekulių projektavimas yra besiplečianti taikomosios chemijos, pramonės ir medicinos šaka. Genetiniai algoritmai gali būti taikomi suprasti molekulių sandarai ir funkcijoms, kurios taikomos kuriant tam tikroms ligoms gydyti reikalingus vaistus. Taip pat, taikant genetinius algoritmus, gali būti nustatomi įvairūs neigiami poveikiai. Tai turi didžiulę įtaką kuriant naujas chemines medžiagas ir vaistus.

Šiuolaikinės technologijos leidžia padaryti momentines genų nuotraukas ir analizuoti genų reikšmes. Šiai analizei galima taikyti genetinius algoritmus ir, pavyzdžiui, nustatyti genus, lemiančius įvairias ligas. Tai turėtų platų pritaikymą medicinoje.

Genetinius algoritmus galima panaudoti kuriant finansinio prognozavimo programoms, kurios taikomos, pavyzdžiui, numatyti situacijai akcijų biržoje.

2.Genetiniai algoritmai daugiakriteriams uždaviniams spręsti

2.1. Daugiakriterių algoritmų efektyvumo tyrimo metodikos

Algoritmai, skirti optimizuoti daugiakriteriams uždaviniams, vadinami daugiakriteriais algoritmais. Tikslo funkcijos, sudarančios daugiakriterį uždavinį, dažniausiai konfliktuoja tarpusavyje, tai reiškia, kad „pagerinus“ vieną iš tikslo funkcijų, gali būti „pabloginta“ kuri nors kita funkcija ar funkcijos. Tokių uždavinių sprendiniai yra ne viena reikšmė, bet nedominuojamų taškų aibė. Kad galima būtų palyginti skirtingus šio tipo uždavinių rezultatus, naudojamos metrikos. Šių metrikų taikymas leidžia ne tik palyginti skirtingus daugiakriterio optimizavimo algoritmus, bet ir daryti patikimas išvadas atliekant statistinę analizę.

Šiuo metu literatūroje nėra aprašyta vienintelės metrikos, kuri visiškai nusakytų vieno daugiakriterio algoritmo pranašumą kito algoritmo atžvilgiu. Literatūroje siūloma taikyti keletą metrikų. Štai keletas metrikų:

1) Generacinis atstumas (GD) – tai metrika, skirta nustatyti, kiek nedominuojamų vektorių aibės elementai nutolę nuo optimalios Pareto aibės; apibrėžiama taip:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n},$$

kur n - vektorių skaičius nedominuojamų sprendinių aibėje, d_i - Euklido atstumas tarp kiekvieno iš sprendinių ir jam artimiausio nario optimalioje Pareto aibėje. Iš pateikto apibrėžimo aišku, kad visi nedominuojami vektoriai yra optimalioje Pareto aibėje, kai GD = 0.

2) Dispersija (Δ) – ši metrika skirta nustatatyti reikšmei, kurią dispersija pasiekia tarp gautų sprendinių, užrašoma taip:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}},$$

kur d_i yra Euklido atstumas tarp greta esančių sprendinių, $\bar{d}$ yra šių atstumų vidurkis, d_f ir d_l yra Euklido atstumai iki Pareto fronto ekstremalių sprendinių. Šios metrikos reikšmė yra lygi nuliui esant idealiam sprendinių pasiskirstymui Pareto fronte.

3) Hipertūris (HV) – metrika, naudojama apskaičiuoti aibės Q nedominuojamų sprendinių dengiamam tūriui (uždaviniams, kurių visos tikslo funkcijos turi būti minimizuotos); kiekvienam sprendiniui $i \in Q$, hiperkubas v_i sudaromas iš taško W (šis taškas gali būti gautas, pavyzdžiui, iš blogiausio sprendinio kiekvienam tikslui) ir taško i, atitinkančio hiperkubo įstrižainę. Hipertūris yra apskaičiuojamas kaip visų hiperkubų sąjungos tūris:

$$HV = volume\left(\bigcup_{i=1}^{Q} v_i\right).$$

4) Pareto optimumų skaičius – sprendžiant kai kuriuos ypač sudėtingus daugiakriterinio optimizavimo uždavinius, duotam daugiakriteriniam algoritmui gali būti sunku rasti didelį nedominuojamų sprendinių skaičių, todėl lyginant skirtingus daugiakriterius algoritmus, galima naudoti surastų Pareto optimumų skaičių kaip metriką.

5) Aibės denginys ($C(A,B)$) – metrika, skirta apskaičiuoti, kiek aibės B sprendinių yra dominuojami aibės A sprendinių:

$$C(A,B) = \frac{|\{b \in B | \exists a \in A : a \preccurlyeq b\}|}{|B|};$$

jei $C(A,B) = 1$, vadinasi, visi aibės B nariai yra dominuojami aibės A narių, jei $C(A,B) = 0$, tai nė vienas aibės B elementas nėra dominuojamas aibės A [AD08, 79-80].

Darbo autorė pasirinko algoritmus lyginti pagal hipertūrio (dvimačiu atveju – hiperploto metriką), nes šios metrikos reikšmę paprasčiausia apskaičiuoti turint Pareto aibę. Kitų metrikų reikšmių radimas gerokai sudėtingesnis.

2.2.Daugiakriteriai genetiniai algoritmai

Praktikoje taikoma daugybė įvairiausių genetinių algoritmų. Darbo autorė pasirinko analizuoti NSGA II, SPEA 2 ir MOCell algoritmus. Pirmieji du algoritmai aptariami teoriškai, o MOCell algoritmui ir dviems jo variacijoms sukurtos programos ir atliktas eksperimentinis tyrimas.

Nedominuojamo rūšiavimo genetinis algoritmas (NSGA II) yra daugiakriterio optimizavimo genetinis algoritmas. NSGA II siekiama pagerinti galimų populiacijos sprendinių prisitaikymą. Algoritme naudojami evoliuciniai operatoriai – selekcija, rekombinacija mutacija ir pakeitimas. Populiacija surūšiuojama į subpopuliacijas, remiantis Pareto dominavimo tvarka. Įvertinamas panašumas Pareto fronte tarp kiekvienos tokios grupės narių ir panašumo matai naudojami sudaryti įvairiapusiam Pareto frontui iš nedominuojamų sprendinių.

Toliau pateikiamas NSGA II algoritmo pseudokodas:

Įvestis: $Populiacijos_{dydis}$, UždavinioDydis, $P_{rekombinacija}$, $P_{mutacija}$
Išvestis: Palikuonys

Populiacija ← InicializuotiPopuliaciją ($Populiacijos_{dydis}$, UždavinioDydis)
ĮvertintiTikslоFunkcijas (Populiacija)
GreitasNedominuojamasRūšiavimas (Populiacija)
Selekcija ← AtrinktiTėvusPagalRangą (Populiacija, $Populiacijos_{dydis}$)
Palikuonys ← RekombinacijaIrMutacija (Selekcija, $P_{rekombinacija}$, $P_{mutacija}$)
while !PabaigosSąlyga() **do**
  ĮvertintiTikslоFunkcijas (Palikuonys)
  Sąjunga ← Sulieti (Populiacija, Palikuonys)
  Frontai ← GreitasNedominuojamasRūšiavimas (Sąjunga)
  Tėvai ← ∅

$Frontas_L \leftarrow \emptyset$

foreach $Frontas_i \in$ Frontai **do**

AtstumoTarpTaškųPriskyrimas ($Frontas_i$)

if Dydis (Tėvai) + Dydis ($Frontas_i$) > $Populiacijos_{dydis}$ **then**

$Frontas_L \leftarrow i$

Break ();

else

Tėvai ← Sulieti (Tėvai, $Frontas_i$)

end

end

if Dydis (Tėvai) < $Populiacijos_{dydis}$ **then**

$Frontas_L \leftarrow$ RūšiuotiPagalRangąIrAtstumą ($Frontas_L$)

for P_1 **to** $P_{Populiacijos_{dydis} - Dydis\ (Frontas_L)}$ **do**

Tėvai ← P_i

end

end

Selekcija ← AtrinktiTėvusPagalRangąIrAtstumą (Tėvai, $Populiacijos_{dydis}$)

Populiacija ← Palikuonys

Palikuonys ← RekombinacijaIrMutacija (Selekcija, $P_{rekombinacija}$, $P_{mutacija}$)

end

return Palikuonys

Greito nedominuojamo rūšiavimo metodu dydžio N populiacija surūšiuojama pagal nedominavimą (kiekvienas populiacijos narys palyginamas su visais likusias populiacijos nariais). Randami individai pirmajame nedominuojamame fronte. Paskui, neatsižvelgiant į pirmojo fronto individus, kartojama ši procedūra, kol randamas antrasis frontas. Taip randami ir visi kiti frontai.

Ieškant frontų, kiekvienam sprendiniui i pirmiausia apskaičiuojamos dvi esybės:

1) n_i - sprendinių, kurie dominuoja sprendinį i, skaičius.

2) S_i - aibė sprendinių, kuriuos dominuoja sprendinys i.

Toliau ieškome taškų, kur $n_i = 0$ ir patalpiname juos į sąrašą F_1. Sąrašą F_1 vadiname einamuoju frontu. Dabar kiekvienam einamojo fronto sprendiniui aplankome kiekvieną jo narį (j) aibėje S_i ir sumažiname jo n_j vienetu. Jei kuriam nors nariui j jo n_j tampa lygus nuliui, patalpiname jį į atskirą sąrašą H. Kai visi einamojo fronto nariai patikrinti, sąrašo F_1 nariai

paskelbiami pirmuoju frontu. Toliau šis procesas tęsiamas su naujai gautu frontu H kaip einamuoju frontu.

Greitas nedominuojamas rūšiavimas, pritaikius jį populiacijai P, grąžina nedominuojamų frontų sąrašą F. Toliau pateikiamas greito nedominuojamo rūšiavimo algoritmo pseudokodas:

foreach $p \in P$
  foreach $q \in P$
    if $(p \prec q)$ **then**
      $S_p = S_p \cup \{q\}$
    elseif $(q \prec p)$ **then**
      $n_p = n_p + 1$
  if $n_p = 0$ **then**
    $F_1 = F_1 \cup \{p\}$
$i = 1$
while $F_1 \neq \emptyset$
  $H = \emptyset$
  foreach $p \in F_1$
    foreach $q \in S_p$
      $n_q = n_q - 1$
      if $n_q = 0$ **then**
        $H = H \cup \{q\}$
    $i = i + 1$
    $F_i = H$

Kad įvertinti sprendinių, supančių konkretų tašką, tankį, imamas vidutinis atstumas tarp taškų, esančių kiekvienoje taško pusėje. Gautas $i_{atstumas}$ yra didžiausias stačiakampis gretasienis, gaubiantis tašką i neįtraukiant jokių kitų taškų. Tankio atstumas (angl. crowding distance) kiekvienam taškui aibėje L:

$l = |L|$ // sprendinių skaičius aibėje L
foreach i, aibės $L[i]_{atstumas} = 0$
foreach tikslui m
  $L = rikiuoti\ (L, m)$
  $L[1]_{atstumas} = L[l]_{atstumas} = \infty$

for $i = 2\ to\ (l-1)$

$L[i]_{atstumas} = L[i]_{atstumas} + (L[i+1].m - L[i-1].m)$

Stiprusis Pareto evoliucinis algoritmas (SPEA 2), kaip ir NSGA II, yra daugiakriterio optimizavimo evoliucinis algoritmas. Šiuo algoritmu siekiama rasti ir išlaikyti nedominuojamų sprendinių frontą, idealiu atveju – Pareto optimalių sprendinių aibę. Kad tai būtų atlikta, naudojama rekombinacija ir mutacija siekiant ištirti paieškos erdvę, ir selekcija, kuri rodo, kiek galimas sprendinys yra nedominuojamas. Nedominuojama aibė laikoma archyve atskirai nuo galimų sprendinių populiacijos, taip sukuriant tam tikrą elitizmą. Žemiau pateikiamas algoritmo pseudokodas:

Įvestis: $Populiacijos_{dydis}$, $Archyvo_{dydis}$, UždavinioDydis, $P_{rekombinacija}$, $P_{mutacija}$
Išvestis: Archyvas

Populiacija ← InicializuotiPopuliaciją ($Populiacijos_{dydis}$, UždavinioDydis)
Archyvas ← ∅
while !PabaigosSąlyga() **do**
  for S_i ∈ Populiacija **do**
    $Si_{tikslai}$ ← ApskaičiuotiTikslus (S_i)
  end
  Sąjunga ← Populiacija + Archyvas
  for S_i ∈ Sąjunga **do**
    Si_{raw} ← ApskaičiuotiRawTinkamumą (S_i, Sąjunga)
    Si_{tankis} ← ApskaičiuotiSprendinioTankį (S_i, Sąjunga)
    $Si_{tinkamumas} \leftarrow Si_{raw} + Si_{tankis}$
  end`
  Archyvas ← GautiNedominuojamą (Sąjunga)
  if Dydis (Archyvas) < $Archyvo_{dydis}$ **then**
  SujungtiSuLikusiaisGeriausiais (Sąjunga, Archyvas, $Archyvo_{dydis}$)
else if Dydis (Archyvas) > $Archyvo_{dydis}$ **then**
  PašalintiPanašiausią (Archyvas, $Archyvo_{dydis}$)
end
Selekcija ← AtrinktiTėvus (Archyvas, $Populiacijos_{dydis}$)
Populiacija ← RekombinacijaIrMutacija (Selekcija, $P_{rekombinacija}$, $P_{mutacija}$)

```
end
return NedominuojamasArchyvas
```

Galiausiai aptarkime MOCell algoritmą. MOCell yra ląstelinis genetinis algoritmas, skirtas daugiakriteriams uždaviniams spręsti. Pateikiamas šio algoritmo pseudokodas:

```
proc Evoliucionuoti(mocell)
Pareto_frontas = Sukurti_PFront() // sukuriamas tuščias Pareto frontas
while !PabaigosSąlyga() do
  for individas ← 1 until mocell.populiacijosDydis do
    kaimynai ← RastiKaimynus(mocell, pozicija(individas));
    tėvai ← Selekcija(kaimynai);
    palikuonis ← Rekombinacija(mocell.Pc, tėvai);
    palikuonis ← Mutacija(mocell.Pm, palikuonis);
    Įvertinti(palikuonis);
    Įterpti(pozicija(individas), palikuonis, mocell, pagalbinė_populiacija);
    ĮterptiĮParetoFrontą(individas, Pareto_frontas);
 end for
 mocell.populiacija ← pagalbinė_populiacija;
 mocell.populiacija ← GrįžtamasisRyšys(mocell, Pareto_Frontas);
end while
end proc Evoliucionuoti;
```

MOCell algoritmas pradeda darbą, sukurdamas tuščią Pareto frontą. Individai yra surikiuojami dvimačiame toro formos tinklelyje ir jiems taikomi genetiniai operatoriai (rekombinacija, mutacija), kol sutinkama pabaigos sąlyga. Taigi kiekvienam individui algoritmas susideda iš: dviejų tėvų parinkimo iš jo kaimynystės, rekombinacijos, kad būtų gautas palikuonis, palikuonio mutacijos, naujai gauto individo įvertinimo ir įterpimo į pagalbinę populiaciją (jei nėra dominuojama esamojo individo) ir Pareto frontą. Pagaliau po kiekvienos kartos sena populiacija pakeičiama pagalbine populiacija, taip pat atliekama grįžtamojo ryšio procedūra, kuria tam tikras skaičius iš populiacijos atsitiktinai parinktų individų pakeičiamas sprendiniais iš archyvo. Šis algoritmas nuo anksčiau aptartųjų skiriasi tuo, kad čia populiacija struktūrizuota – sudaro kaimynystes. Kaimynystėse individai gali būti įvairiai išsidėstę.

2.3.Ląstelinių genetinių algoritmų testavimas

Dauguma tyrėjų sprendžia vienakriterius optimizavimo uždavinius. Tačiau daug realaus pasaulio ir kombinatorinių uždavinių sudaryti iš dviejų ir daugiau tikslo funkcijų, kurios dažnai konfliktuoja viena su kita.

Jei pasirenkamas daugiakriteris uždavinio formulavimas, tada esama daug privalumų. Vienas iš privalumų yra tas, kad tuo pačiu metu vienam uždaviniui galima apskaičiuoti daugiau nei vieną sprendinį, kurie vadinami nedominuojamais, arba efektyviaisiais sprendiniais. Tokie Pareto optimalūs sprendiniai gali būti pavaizduoti erdvėje ar plokštumoje, taip sudarydami Pareto frontą. Pareto fronto radimas yra pagrindinis daugiakriterio optimizavimo uždavinių tikslas.

Dauguma sėkmingiausių metodų, taikomų spręsti daugiakriteriams uždaviniams yra aproksimaciniai. Tarp šių metodų evoliuciniai algoritmai yra vieni iš populiariausių, kaip, pavyzdžiui, NSGA-II ir SPEA2.

Daugiakriterio optimizavimo uždavinius galima spręsti ir taikant ląstelinius genetinius algoritmus. Dar kartą aptarkime daugiakriterio optimizavimo ląstelinį genetinį algoritmą MOCell. Šis algoritmas pradeda darbą sukurdamas tuščią Pareto frontą. Populiacijos individai išdėstyti dvimatėje toro formos gardelėje. Individai pereina reprodukcijos ciklą, kol pasiekiama pabaigos sąlyga. Kiekvienam individui, taikant algoritmą parenkami du tėvai iš kaimynystės, vykdoma rekombinacija, mutacija, kurių metu gaunamas vienas palikuonis, kuris įvertintas, įterpiamas į pagalbinę populiaciją (jei nedominuoja einamojo individo) ir Pareto frontą. Galiausiai einamoji populiacija vienu žingsniu pakeičiama pagalbine populiacija (sinchroninis ląstelinis genetinis algoritmas). Taip pat atliekama grįžtamojo ryšio procedūra, kai atsitiktinai parenkamas tam tikras skaičius individų iš populiacijos ir pakeičiami atitinkamu skaičiumi geriausių individų iš archyvo (Pareto fronto). Ką tik aprašytas daugiakriteris ląstelinis genetinis algoritmas yra sinchroninis ir vadinamas MOCell, arba sMOCell1. Taip pat eksperimentuota ir su sMOCell2, kai grįžtamojo ryšio procedūra atliekama vieną tėvinį sprendinį parenkant iš archyvo, o kitą – iš kaimynystės. Dar vienas algoritmas, su kuriuo eksperimentuota, yra sMOCell3, kai pakeitimo proceso metu imama visa kaimynystė ir pakeičiamas prasčiausias individas iš kaimynystės.

Toliau verta paminėti analizuojamų daugiakriterių ląstelinių genetinių algoritmų parametrus. Populiacija yra stačiakampio formos gardelė, kurią sudaro 100 individų (10 * 10). Algoritmas baigia darbą po penkiasdešimties kartų. Įvade buvo pavaizduoti įvairūs kaimynysčių pavidalai. Čia pasirinkta kaimynystė $L5$. Abu tėvai parenkami taikant binarinio turnyro metodą. Rekombinacija vykdoma dalinant chromosomas vieną kartą trūkio taške. Mutacijos yra

polinominės su tikimybe $p_m = 1.0/L$, čia L atitinka individo ilgį. Pakeitimo metu, individas pakeičiamas, jei jis yra geresnis. Pakeitimas atitinka NSGA-II tankį (crowding distance). Archyvo dydis lygus dešimčiai individų. Grįžtamojo ryšio procedūros metu pakeičiama dvidešimt atsitiktinai parinktų individų.

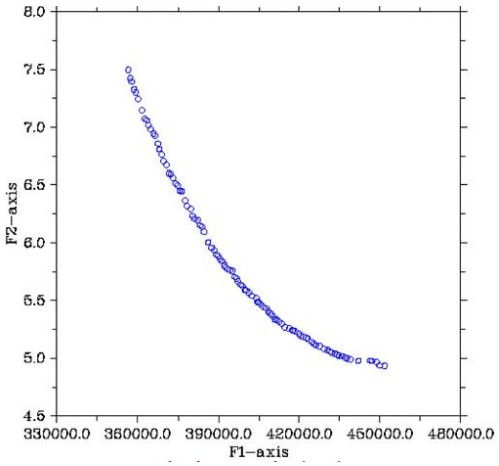
sMOCell1, sMOCell2 ir sMOCell3 algoritmams tirti pasirinktos trys testinės funkcijos ir apskaičiuotas vidutinis hiperplotas, gautas taikant šias funkcijas. Duomenys pateikiami žemiau esančioje lentelėje:

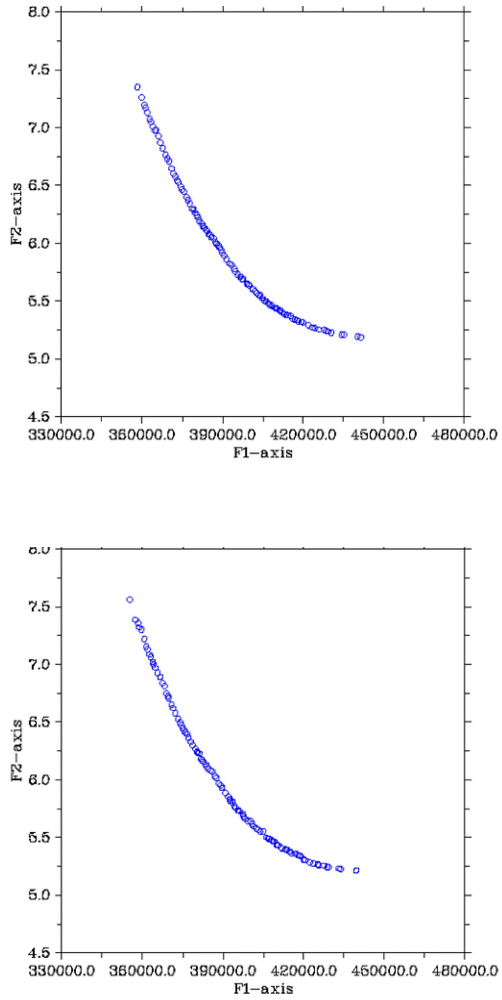
Funkcija	Vidutinis hiperplotas su sMOCell1	Vidutinis hiperplotas su sMOCell2	Vidutinis hiperplotas su sMOCell3
Shaffer	0.26796	0.32648	0.30606
Fonseca	0.62443	0.66062	0.65859
Kursawe	0.74235	0.75118	0.77616

7 pav. Vidutiniai hiperplotai, gauti testinėms funkcijoms taikant įvairius genetinius algoritmus

Shaffer funkcija: $f_1(x) = x^2$

$f_2(x) = (x-2)^2$, kitos dvi funkcijos (Fonseca ir Kursawe) pateikiamos [AD+06, 734].





8 pav. Pareto aibės, gautos taikant Shaffer funkciją sMOCell1, sMOCell2, sMOCell3 ląsteliniams genetiniams algoritmams

Geriausius rezultatus pateikė, tai yra tolygiausiai išdėstė Pareto aibę sMOCell1 algoritmas.

3.Ląstelinių genetinių algoritmų taikymas vizualizavimui

3.1.Grafinis vizualizavimas naudojant OpenGL

Magistro baigiamajame darbe buvo sukurtos programos, kurių rezultatai atvaizduojami naudojant OpenGL grafinę biblioteką. Kad būtų aiškiau, kas yra grafinis vaizdavimas su OpenGL, pateikiama šiek tiek teorijos apie grafinį vaizdavimą minėtąja priemone.

Pradėkime nuo dvimačio vaizdavimo. Daugelis grafinių objektų kompiuterinėje grafikoje sudaryti iš dvimačių dalių, kurios vaizduojamos trimatėje erdvėje. Dvimačių objektų dydis ir forma nusakomi dvimatėmis skaitinėmis reikšmėmis, susietomis su konkrečia koordinačių sistema, pavyzdžiui, Dekarto koordinačių sistema. Dvimatės dalys sudaromos iš primityvių grafinių figūrų (taškų, atkarpų, daugiakampių), kurios dar vadinamos grafiniais primityvais.

Kiekvienas grafinis primityvas aprašomas viršūnių taškų koordinatėmis bei tokiais papildomais duomenimis kaip spalva, permatomumo laipsnis ir panašiai. Iš primityvų suformuojami dvimačiai objektai, su kuriais galima atlikti įvairias transformacijas: keisti jų dydį, padėtį, formą ir taip toliau.

Grafiniai primityvai OpenGL metodais gali būti aprašomi viršūnių koordinatėmis, taikant metodą *glVertex()*. Naudojant *glVertex()* metodą, laikomasi reikalavimo, kad šis metodas būtų iškviestas tarp metodų *glBegin()* ir *glEnd()*. Šie metodai nurodo, kada pradedamas ir baigiamas grafinio primityvo aprašymas. Aprašomo primityvo tipas privalo būti nurodomas funkcijoje *glBegin()* ir tam OpenGL bibliotekoje yra dešimt konstantų. Kelios iš jų: `GL_POINTS`, `GL_LINES`, `GL_TRIANGLES`, `GL_POLYGON`.

Pats mažiausias ir nedalomas grafinis primityvas – figūros taškas (angl. pixel, picture element). Maksimalus taškų skaičius, kurį galima pavaizduoti kompiuterio monitoriuje yra susijęs su monitoriaus skiriamąja geba. Kuo didesnė monitoriaus skiriamoji geba, tuo daugiau taškų galima atvaizduoti. Kompiuterio monitoriaus taško adresas atitinka figūros taško koordinates. Kompiuterinės grafikos vaizdai ekrane sukuriami atskirai valdant kiekvieno monitoriaus taško švytėjimo intensyvumą ir spalvą.

Kitas grafinis primityvas yra atkarpos. Atkarpos aprašomos dviejų taškų, priklausančių tai atkarpai, koordinatėmis (x, y), kur x žymi horizontalų atstumą nuo koordinačių sistemos atskaitos taško, o y – vertikalų atstumą nuo koordinačių sistemos pradžios.

OpenGL bibliotekoje atkarpa gali būti aprašoma grafinio primityvo konstanta `GL_LINES`. Ši konstanta pažymi, kad pradžios ir pabaigos koordinatėmis bus aprašomos atskiros atkarpos. Atkarpos turi būti aprašomos tarp metodų *glBegin()* ir *glEnd()*.

Tokie grafiniai primityvai, kaip apskritimai ir elipsės piešiami panaudojant primityvą atkarpą.

Daugiakampis yra pagrindinis paviršiaus primityvas. Daugiakampis gali būti aprašomas kaip atkarpų, besijungiančių galų taškais ir sudarančių uždarą kontūrą, rinkinys. Jį taip pat galima aprašyti viršūnių taškais.

Daugiakampiai skiriami į dvi grupes:

1) Iškiluosius.
2) Neiškiluosius.

Iškiluoju vadinamas toks daugiakampis, kurio bet kuriuos du taškus jungiančios atkarpos visi taškai priklauso daugiakampiui.

OpenGL priemonėmis aprašant grafinius objektus galima naudoti tik iškiluosius daugiakampius. Daugiakampiai, kaip ir atkarpos, gali būti aprašomi viršūnių taškų

koordinatėmis, naudojant funkciją *glVertex()*. Daugiakampiams aprašyti taip pat naudojamos konstantos, pavyzdžiui: `GL_TRIANGLES`, `GL_QUADS`, `GL_POLYGON`.

`GL_TRIANGLES` – pateiktas viršūnes jungia į trikampius. Jei trikampis nesusidaro, likusios viršūnės atmetamos.

`GL_QUADS` – pateiktas viršūnes jungia į keturkampius. Jeigu pateikiamų viršūnių nepakanka keturkampiui sudaryti, jos atmetamos. Vaizduojamas objektas turi būti iškilus.

`GL_POLYGON` – pateiktos viršūnės nuosekliai jungiamos į uždarą kontūrą.

Vaizduojamus objektus tenka spalvinti. OpenGL bibliotekoje spalvas galima aprašyti keliais formatais, vienas iš jų – RGB. RGB atveju spalva suformuojama maišant trijų spalvų reikšmes: raudonos, žalios ir mėlynos. Jeigu šias spalvas vaizduotume kaip realius skaičius, tai 0.0 reikštų mažiausią, o 1.0 didžiausią reikšmes. Šiuo atveju (1.0, 1.0, 1.0) atitiks baltą spalvą, o (0.0, 0.0, 0.0) – juodą spalvą.

OpenGL biblioteka veikia būsenų principu. Spalvos yra būsenos dalis. Tai reiškia, kad figūrų piešimui bus naudojama nustatyta piešimo spalva ir ji bus naudojama tol, kol nebus pakeista į kitą einamąją spalvą.

OpenGL biblioteka turi tam tikrą skaičių naudojimui paruoštų trimačių primityvų, tokių kaip: kubas, kūgis, sfera ir taip toliau. Taip pat yra ir sudėtingesnių figūrų: arbatinukas, lankas (angl. *torus*). Minėtuosius objektus galima piešti kaip karkasines figūras arba kaip užpildytas figūras. Šiame darbe naudotos sferos ir cilindrai piešti penkiamačiam kubui.

Trimatėje erdvėje su objektais tenka atlikti įvairias transformacijas, tokias kaip pastumti ar pasukti objektą.

OpenGL bibliotekoje taško (x, y, z) postūmio į tašką (x', y', z') transformacija trimatėje koordinačių sistemoje atliekama taikant metodą *glTranslate()*.

Trimačių objektų posūkis trimatėje erdvėje atliekamas posūkio transformacijos veiksmais. Posūkio transformacija trimatėje erdvėje sudėtingesnė negu dvimatėje, nes sukama ne taško, bet tiesės atžvilgiu. Be to, trimatį objektą ir reikiamas transformacijas trimatėje erdvėje sunkiau įsivaizduoti. Sukimas apie laisvai pasirinktą sukimo ašį trimatėje erdvėje atliekamas sukant apie vieną iš trijų koordinačių sistemos ašių. Sukant tašką apie koordinačių sistemos ašį, tą ašį atitinkanti taško koordinatė nekeičiama. Pavyzdžiui, sukant tašką apie x ašį, keisis taškų y ir z koordinatės, o taško x reikšmė nesikeis.

Atliekant kelias posūkio transformacijas, svarbu nesupainioti jų tvarkos, nes atlikus transformacijas skirtinga tvarka, skirsis ir galutiniai rezultatai.

OpenGL bibliotekoje posūkio transformacijas atlieka metodas *glRotate()*.

3.2.Stereoskopija

Taip pat verta pasidomėti ir stereoskopija, nes sukurtos programos gali būti paleidžiamos erdviniame kompiuteryje. Perspektyva dažnai painiojama su stereoskopija (3D). Tai nėra teisinga, nes perspektyvoje trečioji dimensija (gylis) yra „simuliuojamas“. Todėl perspektyvą būtų tiksliau vadinti ne 3D, o $2^{1}/_{2}$D.

Žodis „stereo“ kilęs iš graikų kalbos ir reiškia „susijęs su erdve“. Šiandien dažniausiai kalbama apie stereofoninį garsą. Iš pradžių šis terminas buvo susijęs su steroskopiniais paveikslais, kurie buvo piešti arba fotografuoti. Siekiant nepainioti su stereofoniniu garsu, vartojamas terminas 3D kalbant apie paveikslus arba filmus, kur 3D reiškia trijų dimensijų (matmenų).

Žmogus gyvena trijų matmenų aplinkoje. Neturėdami erdvės supratimo, mes negalime joje judėti. Mūsų erdvės suvokimas sukuriamas daugiausia akimis. Yra daug būdų orientuotis erdvėje: pagal perspektyvą, spalvą, kontrastą, judesį.

Lęšiukai sveiko žmogaus akyse į tinklaines projektuoja šiek tiek skirtingus vaizdus, kuriuos smegenys transformuoja į erdvinį pavidalą. Tikras stereoskopinis vaizdas priklauso nuo abiejų akių suvokimo.

Yra sveikų žmonių, turinčių abi akis, tačiau su tam tikrais defektais nuo gimimo dėl kurių neįmanoma matyti stereoskopinio vaizdo. Dar būdami kūdikiai jie išmoko orientuotis aplinkoje. Panašiai yra ir su žmonėmis, turinčiais tik vieną sveiką akį.

Įprastas paveikslas ant popieriaus arba filmas, skirti tik vienai akiai. Tokia nuotrauka nufotografuota tik su vieno lęšio kamera, todėl neperteikia tikro erdvinio vaizdo ir yra plokščia. Kameroje naudojant du lęšius, imituojamos akys ir gaunamas erdvinis vaizdas.

Kai žiūrime į stereoskopinį paveikslą, mūsų smegenyse sukuriamas trimatis vaizdas. 3D fotografija kopijuoja žmogaus žiūrėjimą į 3D objektą ar sceną, imdama porą fotografijų atskirtų atstumu, lygiu atstumui tarp žmogaus akių. Tada po vieną fotografiją pateikiama kairei ir dešinei akims ir smegenyse suformuojamas trimatis vaizdas.

Stereoskopinis vaizdas gali būti gautas taikant fotografiją, kompiuterius ar lazerius.

3D vaizdai gali būti peržiūrimi naudojantis stereoskopu (pavyzdžiui, skaidrės) arba ant kompiuterio ekrano. Taip pat yra anaglifinių paveikslų (peržiūrai reikalingi raudonai žali arba raudonai mėlyni 3D akiniai) bei skaitmeninė stereo projekcija (reikalingi arba „pasyvūs“ poliarizuoti 3D akiniai arba „aktyvūs“ skystųjų kristalų akiniai).

Nors OpenGL gali kurti trimačius vaizdus, kompiuterio ekrane matoma trimačio vaizdo 2D projekcija. Norint matyti tikrą trimatį vaizdą kompiuterio ekrane, reikalinga speciali techninė ir programinė įranga. Neturint tokios įrangos, taip pat galima kurti 3D vaizdus. Tada reikia remtis

stereofotografijos principu ir pateikti atskirą vaizdą kiekvienai akiai. Norint matyti taip sukurtą vaizdą reikia žiūrėti tam tikru būdu arba dėvėti 3D akinius (anaglifiniams paveikslams).

Autostereoskopija reiškia bet kokį metodą, kai 3D turinys peržiūrimas nedėvint 3D įrangos, pavyzdžiui, 3D akinių. Angliškai tai vadinama "Glasses-free 3D" arba "Glasses-less 3D". Ši nauja pažangi technologija įgalina kiekvieną akį peržiūrėti būtent jai skirtą vaizdą, kad būtų sukurtas 3D efektas.

Žinomiausios autostereoskopinės technologijos yra paralaksinis barjeras ir lęšiuko formos (abipusiai išgaubtas) lęšis. Toliau bus aptarti abu paminėti metodai.

Paralaksinis barjeras yra įrenginys, panašus į kaukę, kuris patalpintas priešais skystųjų kristalų ekraną ar šviesos šaltinį, kad nukreiptų pikselių stulpelių skleidžiamą šviesą į kiekvieną akį per jame esančius plyšius. Taigi kiekviena akis mato skirtingą pikselių aibę, skirtą tai akiai ir smegenyse susiformuoja 3D vaizdas. Pagrindinis šios technikos trūkumas yra tas, kad žiūrovas turi būti tam tikroje, iš anksto nustatytoje vietoje, kad patirtų 3D efektą. Jei judama, prarandama 3D iliuzija. Paralaksinis barjeras leidžia persijungti nuo 3D vaizdo į 2D. Tai atliekama nutraukiant įtampos tiekimą barjerui ir jį išjungiant.

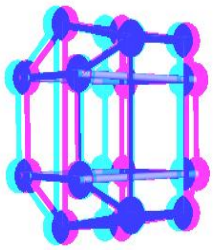
Lęšiuko formos lęšio veikimas paremtas paprastu pavyzdžiu. Kai uždengiame vieną akį ir žiūrime į cilindrinį objektą, geriau matomi galiniai cilindro taškai. Lęšiuko formos lęšiai yra pagaminti iš mažų cilindro formos plastikinių lęšių, sudėliotų ant permatomos lentelės. Šita lentelė yra patalpinta priešais skystųjų kristalų ekraną. Kai stebėtojas žiūri į ekraną, kiekviena akis mato 2D vaizdą. Šie vaizdai smegenyse yra „sujungiami" ir gaunamas 3D vaizdas.

Autostereoskopija taikoma daugelyje sričių. Žinomiausi yra paveikslai, filmai ir žaidimai. Paralaksinis barjeras taikomas ne tik filmams ir žaidimams, bet ir molekuliniam modeliavimui, oro uostų saugos sistemoms.

Lęšiuko formos lęšis gali būti taikomas perversti knygos vaizdams, kas gali suteikti spaudiniui 3D animacijos patirtį. Daug kompanijų taip pat siekia panaudoti šią technologiją reklamai, pavyzdžiui reklaminiams plakatams ir lankstinukams.

Anaglifiniai vaizdai – tai tokie vaizdai, sudaryti iš dviejų persidengiančių vaizdų, kurių vienas paprastai raudonas, o kitas – žalsvai mėlynas, žalias arba mėlynas. Taip sukuriamas 3D efektas. Į šiuos paveikslus reikia žiūrėti pro 3D akinius (raudonus-žalsvai mėlynus, raudonus-žalius ar raudonus-mėlynus).

Tarkime, kad žiūrime 3D filmą su 3D akiniais. Žiūrovas ekrane mato du vaizdus, kurių vienas yra raudonas, o kitas mėlynas (žalsvai mėlynas arba žalias). Akinių filtrai leidžia tik vienam vaizdui pasiekti kiekvieną akį, o smegenys iš šių 2D vaizdų sukuria 3D vaizdą.



9 pav. Keturmatis kubas erdvėje (žiūrėti pro raudonai žalius akinius)

3.3.Eksperimentinis tyrimas

Daugelyje žmogaus veiklos sričių, pavyzdžiui, medicinoje, technikoje ir kitur tenka analizuoti daugiamačius duomenis. Tokių duomenų apimtys nuolat didėja, nes jiems kaupti ir analizuoti pasitelkiama naujausia techninė ir programinė įranga. Teisingai interpretavus daugiamačius duomenis, galima padaryti tam tikras išvadas.

Praktikoje žinoma daugybė daugiamačių duomenų pavyzdžių. Yra įvairių naudojimui paruoštų duomenų bazių, kuriose saugomi daugiamačiai duomenys. Su šiais duomenimis galima atlikti įvairius eksperimentus. Plačiausiai žinomi keturmačiai irisų duomenys, tai yra 150- čiai irisų, kurie yra trijų skirtingų veislių, išmatuoti vainiklapių pločiai, vainiklapių ilgiai, taurėlapių pločiai ir taurėlapių ilgiai. Taip pat dažnai naudojami krūties vėžio duomenys. Šiuo atveju 683 moterims kurioms nustatytas piktybinis ir nepiktybinis navikas, išmatuoti devyni fiziologiniai parametrai. Tokiu būdu gauti devynmačiai duomenys. Minėtieji irisai ir krūties vėžio duomenys yra vieni iš daugelio daugiamačių duomenų pavyzdžių.

Norint gauti tam tikras išvadas, daugiamačiai duomenys analizuojami. Praktikoje yra nemažai daugiamačių duomenų analizės metodų:

- Klasifikavimo.
- Klasterizavimo.
- Statistinės analizės.

Naudojantis minėtaisiais metodais, galima vertinti duomenų artimumą, taip pat padaryti ir kitas išvadas. Tačiau dažnai kyla poreikis neformaliai pažinti turimus duomenis, tai yra suvokti jų vaizdą. Žinoma, tai ganėtinai sudėtinga, kai duomenys, pavyzdžiui, yra devynmačiai. Kad žmogus galėtų savo akimis pamatyti daugiamačių duomenų sandarą, taikomi įvairūs duomenų vizualizavimo metodai.

Sunku suvokti duomenis, kai jie yra daugiamačiai, tai yra juos apibūdina daug parametrų, kurie gali būti ne tik skaitiniai, bet ir tekstiniai ar dar kokie kitokie. Taip pat analizuojant

duomenis tenka susidurti su panašių objektų grupėmis – klasteriais bei labai išsiskiriančias objektais – taškais atsiskyrėliais.

Kalbant apie daugiamačius duomenis, kurios siekiama vizualizuoti, galima skirti dvi pagrindines sąvokas: objektus ir parametrus. Objektais paprastai vadiname mus supančius dalykus ir daiktus, o parametrais – objektų savybes. Tarkime, kad turime m objektų. Tam tikras visų parametrų reikšmių rinkinys apibūdina vieną analizuojamos aibės objektą $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$, $i \in \{1, \dots, m\}$, čia n yra parametrų skaičius, i yra objekto eilės numeris. Parametrų skaičius n dar vadinamas duomenų matmenų skaičiumi. Kai objektą apibūdina daugiau nei vienas parametras, duomenys yra daugiamačiai.

Bendru duomenų analizės atveju, parametrai turi tam tikras skaitines reikšmes. Todėl analizuojamų duomenų aibė yra matrica

$$X = \{X_1, X_2, \dots, X_m\} = \{x_{ij}, i = 1, \dots, m, j = 1, \dots, n\},$$

kurios i – oji eilutė yra vektorius $X_i \in R_n$, čia $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$, $i \in \{1, \dots, m\}$, m – analizuojamų objektų skaičius.

Kartais negalime skaitiškai įvertinti parametrų, tačiau galime nustatyti artimumus tarp objektų porų – panašumus ar skirtingumus. Tai dažnai taikoma psichologiniuose tyrimuose. Tada naudojama artimumų (panašumų ar skirtingumų) matrica:

$$\Delta = \{\delta_{ij}, i, j = 1, \dots, m\},$$

čia δ_{ij} yra objektų X_i ir X_j artimumo įvertis.

Augant duomenų apimčiai, tampa vis sunkiau suvokti objektų visumos ypatybes. Tokių duomenų analizė yra sudėtingas uždavinys. Kadangi analizuojant galima padaryti įvairias išvadas, tam naudojami duomenų analizės metodai, tai yra tokie kaip klasifikavimo, klasterizavimo, vizualizavimo ir kiti. Kad žmogui duomenis būtų lengviau suvokti, dažnai renkamasi duomenų vizualizavimas. Vizualizavimas reiškia grafinį informacijos pateikimą. Vizualus duomenų pateikimas leidžia geriau suvokti duomenų sandarą bei savybes. Be to, žmogus vizualią informaciją daug greičiau suvokia.

Paprasčiausia duomenis pateiktus įvairiose lentelėse atvaizduoti histogramomis ir kitokiais grafikais. Sudėtingesni duomenų vizualizavimo metodai leidžia įvertinti daugiamačių duomenų struktūrą: panašių objektų grupes (klasterius), labai išsiskiriančius objektus (taškus atsiskyrėlius).

Pastaruosius 40 metų ypač intensyviai plėtojami duomenų vizualizavimo metodai. Šis darbas vyksta dviem pagrindinėmis kryptimis. Tai yra tiesioginio vizualizavimo metodai bei

projekcijos metodai. Tiesioginio vizualizavimo metodo atveju kiekvienas objekto parametras pateikiamas tam tikra grafine forma. Projekcijos metodais analizuojama duomenų aibė $X = \{X_1, X_2, \dots, X_m\}$ atvaizduojama iš n – matės erdvės R^n į mažesnio matmenų skaičiaus erdvę R^d $(d < n)$, kur galima stebėti duomenų aibės transformacijos

$$Y = \{Y_1, Y_2, \dots, Y_m\} = \{y_{ij}, i = 1, \dots, m, \quad j = 1, \dots, d\}$$

taškų išsidėstymą. Žinoma, taikant projekcijos metodus atsiranda duomenų iškraipymų ir paklaidų.

Skiriami du esminiai duomenų vizualizavimo tikslai:

- Tiriamoji analizė – duomenys pirmiausia vizualizuojami, o tada iškeliamos hipotezės.
- Patvirtinančioji analizė – duomenis vizualizavus patvirtinamos jau egzistuojančios hipotezės.

Tiesioginio vizualizavimo metodai skirstomi į geometrinius, simbolinius, hierarchinio vizualizavimo.

Geometriniai metodai: taškinių grafikų, taškinių grafikų matricos, linijinių grafikų, perstatymų matricos, apžiūros grafikų, Andrews kreivių, lygiagrečiųjų koordinačių, spindulinio vizualizavimo.

Simboliniai metodai: Černovo veidų, žvaigždžių, brūkšnelinės figūros, spalvotos piktogramos.

Hierarchinio vizualizavimo metodai: matmenų įterpimo, grotelių, fraktalų, hierarchinių lygiagrečiųjų koordinačių.

Projekcijos metodai skirstomi į tiesinės projekcijos ir netiesinės projekcijos metodus:

Tiesinės projekcijos metodai:

- Pagrindinių komponenčių analizės.
- Tiesinės diskriminantinės analizės.
- Projekcijos paieškos.

Netiesinės projekcijos metodai:

- Daugiamačių skalių.
- ISOMAP.
- Lokaliai tiesinio vaizdavimo.
- Pagrindinės kreivės.
- Trianguliacijos.

Daugiamačius duomenis taip pat galima vizualizuoti taikant dirbtinius neuroninius tinklus. Jais vaizduojanos įvairios netiesinės projekcijos.

Dabar plačiau aptarkime kai kuriuos tiesioginio vizualizavimo metodus.

Taškinių grafikų metodu atvaizduojami dvimačiai ar trimačiai taškai dvimatėje plokštumoje ar trimatėje erdvėje.

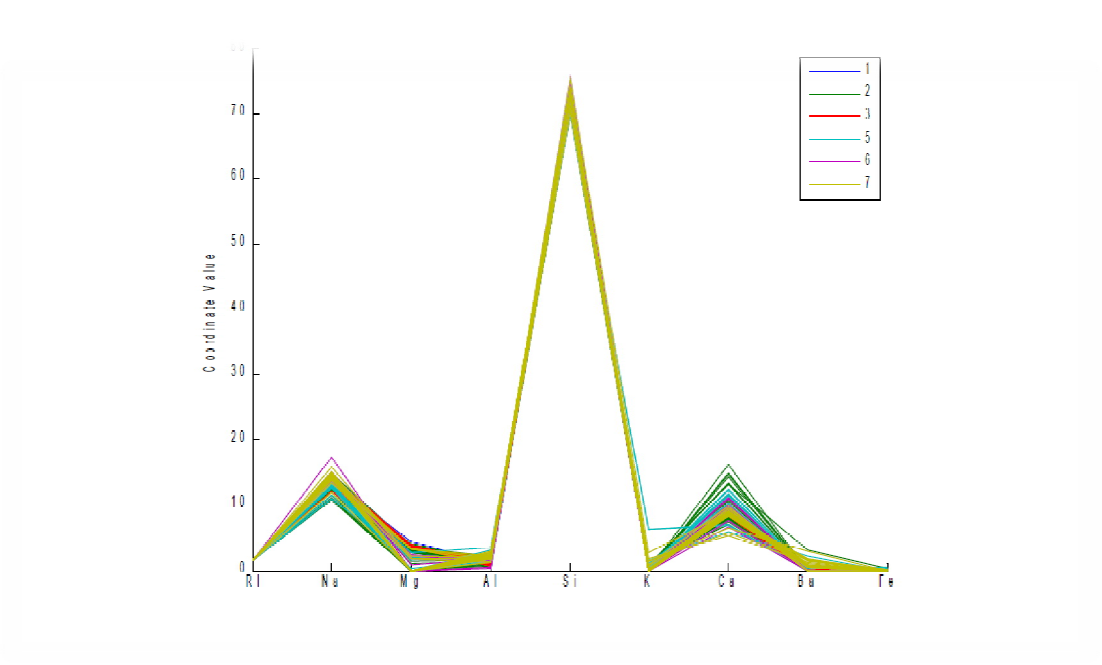
Taškinių grafikų matricą galima naudoti atvaizduoti daugiau nei trijų matmenų duomenims. Tokioje matricoje atvaizduojamos visos galimos daugiamačius duomenis apibūdinančių parametrų poros. Matricos įstrižainėje gali būti viena iš grafinių statistikinių parametro išraiškų.

Linijiniai grafikai naudojami vieno kintamojo funkcijoms atvaizduoti. Jais dažniausiai vizualizuojami dvimačiai duomenys. Linijiniai grafikai gaunami jungiant linija taškus, kuriuos atitinka koordinatės x ir y.

Apžiūros grafike kiekviena skaitinė parametro reikšmė vaizduojama juostele, kurios ilgis atitinka to parametro reikšmę. Perstatymų matrica yra pasukama devyniasdešimt laipsnių kampu, o kiekvieną parametrą atitinkanti juostelė perstumiama per pusę ilgio.

Daugiamačius vektorius galima pavaizduoti ir Andrews kreivėmis (sinusoidžių sumomis), kurių koeficientai yra analizuojamų vektorių komponentės.

Taikant lygiagrečiųjų koordinačių metodą koordinačių sistemos ašys yra viena kitai lygiagrečios ir išdėstytos vienodu atstumu. Dekarto koordinačių dvimatis taškas lygiagrečiųjų koordinačių sistemoje yra tiesė, o tiesė – taškas.



10 pav. Duomenys apie įvairiose stikluose esančių metalų oksidų kiekį, atvaizduoti lygiagrečiųjų koordinačių metodu

Hierarchinių lygiagrečiųjų koordinačių metodas yra lygiagrečiųjų koordinačių metodo modifikacija. Duomenys šiuo metodu vaizduojami taip:

- Iš pradžių analizuojami duomenys suskirstomi į klasterius taikant kokį nors klasterizavimo metodą.
- Duomenys atvaizduojami lygiagrečiose koordinatėse; skirtingus klasterius atitinka skirtingos spalvos.

Taikant Černovo veidų metodą objektai atvaizduojami stilizuotais veidais. Objektą atitinkantys parametrai yra tam tikra veido dalis, turinti savo dydį, padėtį ar formą.

Žvaigždžių metodu objektai vaizduojami stilizuotomis žvaigždėmis. Imamas taškas ir iš jo nubrėžiama tiek spindulių, kiek objektas turi parametrų. Kampai tarp spindulių turi būti vienodi. Spindulio ilgį lemia jį atitinkančio parametro reikšmė. Išoriniai spindulių galai sujungiami linijomis.

Projekcijos vizualizavimo metodų plačiau nenagrinėsime, nes magistro darbe išsamiai aptariamas vienas iš projekcijos metodų – daugiamatės skalės.

Dabar aptarsime pasirinktą daugiamačių duomenų vizualizavimo metodą, šiuo atveju daugiamates skales – DS. Daugiamačių skalių metodas plačiai naudojamas daugiamačių duomenų analizei įvairiose šakose, ypač ekonomikoje, socialiniuose moksluose ir kitur. Gausu šio metodo realizacijų, kurios skiriasi naudojamais vizualizavimo kokybės kriterijais, optimizavimo algoritmais ar prielaidomis apie duomenis.

Naudojantis DS, ieškoma daugiamačių duomenų projekcijų mažesnio skaičiaus matmenų erdvėje (dažniausiai R^2 arba R^3), siekiant išlaikyti analizuojamos aibės objektų artimumus – panašumus arba skirtingumus. Gautuose vaizduose panašūs objektai išdėstomi arčiau vieni kitų, o skirtingi – toliau vieni nuo kitų [DKŽ08, 51].

Pradiniai daugiamačių skalių metodo duomenys yra kvadratinė simetrinė matrica, kurios elementai nusako artumą tarp analizuojamų objektų. Tai gali būti arba panašumų, arba skirtingumų matrica. Paprasčiausiu atveju tai yra Euklido atstumų tarp objektų matrica. Tačiau bendruoju atveju, tai nebūtinai turi būti atstumai griežtai matematine prasme.

Vienas daugiamačių skalių metodo tikslų yra rasti optimalų daugiamačius objektus atitinkančių taškų (vektorių) vaizdą mažo skaičiaus matmenų erdvėje.

Tarkime, kiekvieną n – matį vektorių $X_i \in R^n$, $i \in \{1, \dots, m\}$, atitinka mažesnio skaičiaus matmenų vektorius $Y_i \in R^d$, $d < n$. Atstumą tarp vektorių X_i ir X_j pažymėkime $d(X_i, X_j)$, o atstumą tarp vektorių Y_i ir $Y_j - d(Y_i,\ Y_j)$, $i, j = 1, \dots, m$.

Naudojantis DS algoritmu, bandoma atstumus $d(Y_i, Y_j)$ priartinti prie atstumų $d(X_i, X_j)$. Jei naudojama kvadratinė paklaidos funkcija, tai minimizuojama tikslo funkcija E_{DS} gali būti užrašyta taip:

$$E_{DS} = \sum_{i<j} w_{ij} \left(d(X_i, X_j) - d(Y_i, Y_j) \right)^2.$$

Paklaidos funkcija E_{DS} dar vadinama $Stress$ funkcija. Čia $w_{ij} > 0$, $d(X_i, X_j) \neq 0$, t. y. tarp vektorių $X_1, X_2, \dots, X_m$ nėra sutampančių [DKŽ08, 51-52].

Toliau paanalizuosime evoliucinio algoritmo realizaciją taikant daugiamačių skalių metodą. Kaip jau žinome, evoliucinės paieškos idėja – saugoti geriausius tikslo funkcijos reikšmės prasme sprendinius, kuriuos sukryžminus galima sukurti geresnius sprendinius. Vienas iš esminių evoliucinės paieškos elementų – sprendinių kodavimas chromosomomis. Bendruosiuose tolydaus minimizavimo genetiniuose algoritmuose sprendiniai gali būti koduojami minimizavimo kintamųjų reikšmių vektoriumi. Pradinės chromosomos yra sugeneruojamos atsitiktinai.

Vėliau gali būti naudojamos kelios genetinės operacijos. Mutacijos metu atsitiktinai su tam tikra tikimybe pakeičiamas atsitiktinis genas. Nagrinėjamuoju atveju – tai kintamojo reikšmė. Rekombinacijos metu iš dviejų populiacijos individų chromosomų sudaroma palikuonio chromosoma, paprastai viena dalis genų paimama iš vienos, o kita dalis – iš kitos tėvinės chromosomos. Prisitaikymas prie aplinkos gali būti modeliuojamas lokaliuoju minimizavimu. Kartais algoritmai, derinantys evoliucinę ir lokaliąją paiešką, vadinami memetiniais. Palikuonio gerumas yra apibrėžiamas tikslo funkcijos reikšme. Jei palikuonis yra geresnis už blogiausią populiacijos individą, jis pastarąjį pakeičia. Minimizavimas tęsiamas generuojant naujus palikuonis ir stabdomas po iš anksto numatyto iteracijų skaičiaus arba nustatyto laiko.

Daugiamačių skalių atveju tikslinga genais laikyti objektus vaizduojančių taškų koordinates. Tokiu atveju kryžminimo metu dalis taškų paimama iš vieno, o kita dalis – iš kito populiacijos individo, nesukryžminant skirtingų taškų. Tokiu atveju dviejų atsitiktinių esamos populiacijos individų su chromosomomis $\ddot{Y}$ ir $\ddot{Y}$ palikuonio chromosoma randama remiantis formule

$$Y_i = \begin{cases} \ddot{Y}_i, jei\ i \leq \beta, i \geq \gamma, \\ \ddot{Y}_i, jei\ \beta < i < \gamma, \end{cases}$$

čia $i = 1, \dots, m$, β ir γ - du atsitiktiniai skaičiai iš aibės $\{1, \dots, m\}$.

Evoliucinis algoritmas gali būti realizuojamas naudojantis toliau pateiktu algoritmu. Pradinių atsitiktinių sprendinių skaičius n_{init}, populiacijos dydis n_p, sustojimo sąlyga pagal laiką t_c yra algoritmo parametrai.

1. Atsitiktinai sugeneruoti n_{init} tolygiai pasiskirsčiusių md ilgio vektorių ir atsiminti n_p geriausių įtempimo funkcijos reikšmės prasme ir taip suformuoti populiaciją.

2. Pagerinti populiaciją atliekant lokalią paiešką iš populiacijos individų vektorių, atnaujinti vektorių elementų reikšmes gautais minimumų taškais.

3. ***Kol*** nepasiektas laikas t_c:

   a) Atsitiktinai parinkti du esamos populiacijos individus, pavadinkime jų chromosomas $\ddot{Y}$ ir $\ddot{Y}$.

   b) Sugeneruoti palikuonį atliekant kryžminimą ir lokalią paiešką iš gauto vektoriaus Y, atnaujinti vektoriaus elementų reikšmes gautu minimumo tašku.

   c) ***Jei*** palikuonis geresnis įtempimo funkcijos reikšmės prasme už blogiausią populiacijos individą,

   d) ***tai*** jis pastarąjį pakeičia, tai yra blogiausią populiacijos individą simbolizuojančio vektoriaus elementų reikšmės atnaujinamos palikuonio vektoriaus elementų reikšmėmis [DKŽ08, 75-76].

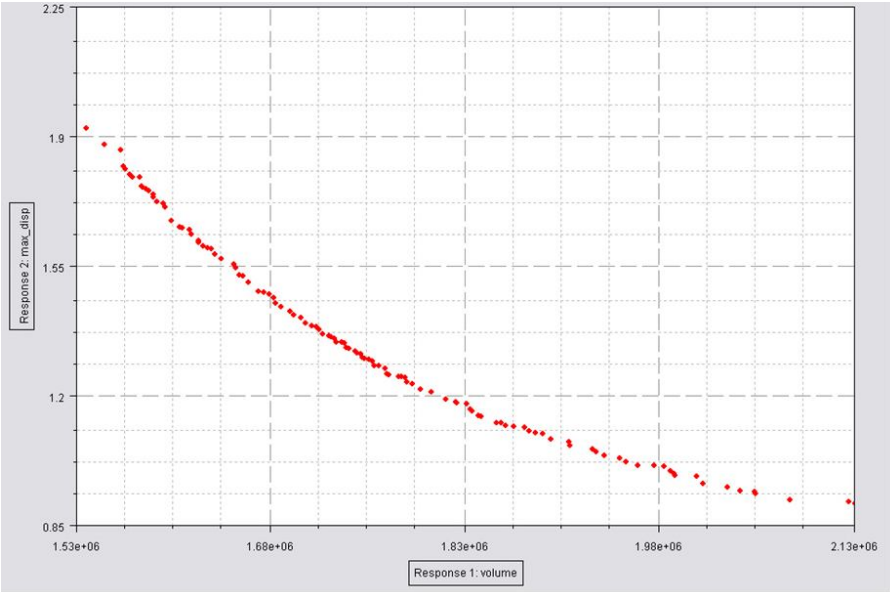
Kaip jau minėta, duomenims vizualizuoti pasirinktas daugiamačių skalių metodas (DS). Trumpai prisiminkime šį metodą. Taikant DS, ieškoma daugiamačių duomenų projekcijų mažesnio skaičiaus matmenų erdvėje išlaikant analizuojamos aibės objektų skirtingumus. Tarkime, kad skirtingumai žymimi δ_{ij}, o atstumai tarp analizuojamos aibės objektų – $d(X_i, X_j)$. Naudojantis DS algoritmu, stengiamasi skirtingumus δ_{ij} priartinti prie atstumų $d(X_i, X_j)$. Kadangi mes sprendžiame dviejų kriterijų minimizavimo uždavinį, tai minimizuojama kvadratinė tikslo funkcija (kvadratinė paklaida)

$$S_1 = \sum_{i<j} w_{ij} \, (d(X_i, X_j) - \delta_{ij})^2$$

ir absoliutinė paklaida

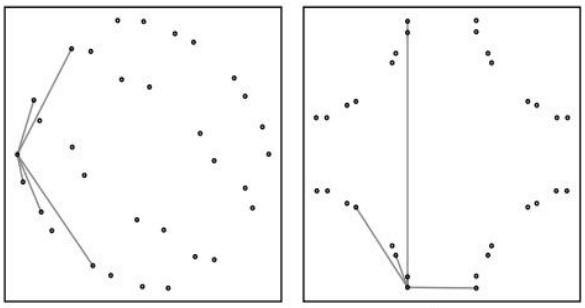
$$S_2 = \sum_{i<j} w_{ij} \, |d(X_i, X_j) - \delta_{ij}|$$

Minimizuojamas funkcijas žymėkime S_1 ir S_2, w_{ij} reiškia svorius. Taigi turime dvi funkcijas: S_1, kuri atitinka kvadratinę paklaidą ir S_2, atitinkančią absoliutinę paklaidą. Jei spręstume minimizavimo uždavinį su kvadratine ir absoliutine paklaidas atitinkančiais kriterijais, gautume Pareto aibę. Minimizavimo atveju Pareto aibė atrodo taip:



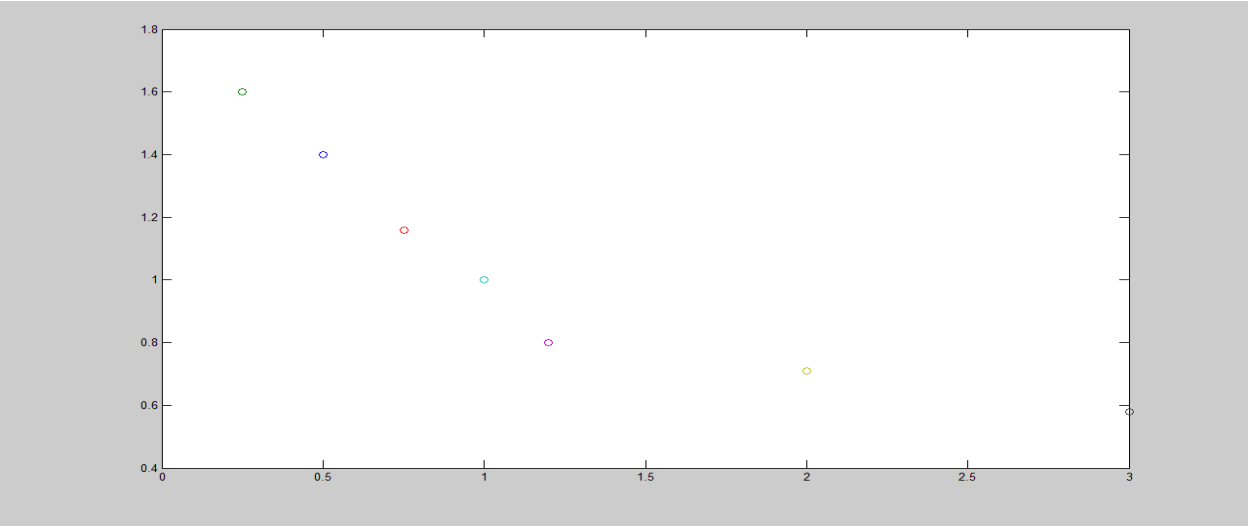
11 pav. Pareto aibė, gauta taikant MOGA algoritmą (minimizavimo uždavinys) [Del10]

Tarkime, kad abscisių ašis atitinka kvadratinės paklaidos kriterijų, o ordinačių – absoliutinės. Tada kiekvienas Pareto aibės taškas atitiks skirtingą DS metodu gautą vaizdą, šiuo atveju – penkiamatį kubą. Kuo Pareto aibės taškai arčiau kvadratinio kriterijaus, tuo gauti vaizdai labiau atitiks kvadratinės paklaidos vaizdus. Taip pat yra ir su absoliutine paklaida. Vaizdai, reiškiantys taškus iš Pareto aibės vidurio bus tarpiniai. Jie bus kvadratinio ir absoliutinių vaizdų kombinacijos. Žemiau esančiame paveiksle pateikiami penkiamačiai kubai gauti pagal S_1 ir S_2 kriterijus.

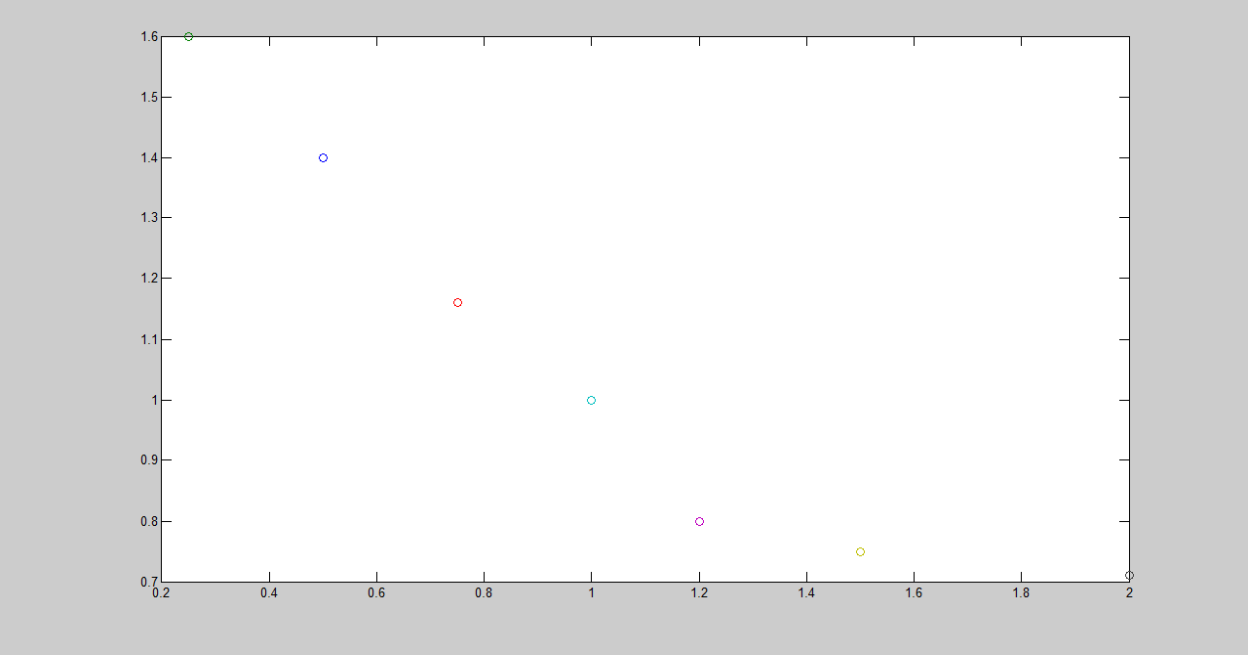


12 pav. Kubai gauti pagal kvadratinės paklaidos kriterijų (kairėje) ir absoliutinės paklaidos kriterijų (dešinėje) [DKŽ08, 97]

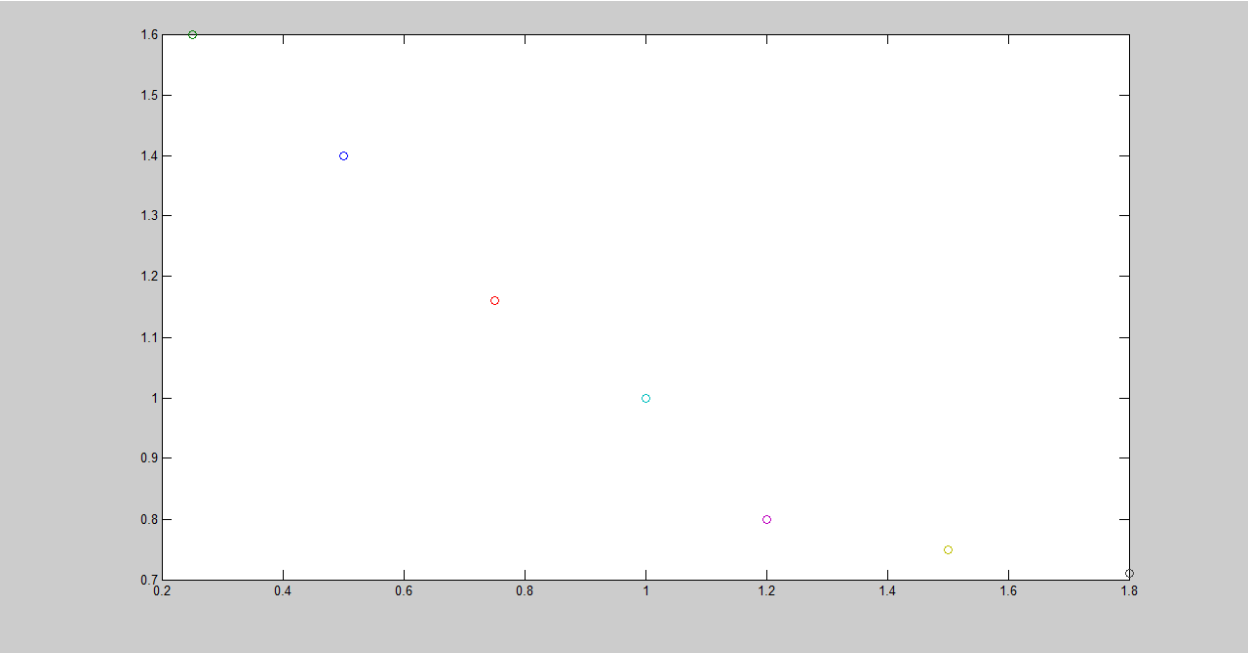
Toliau pateikiamos Pareto aibės, gautos sMOCell1, sMOCell2, sMOCell3 algoritmais.



13 pav. Pareto aibė, gauta algoritmu sMOCell3.



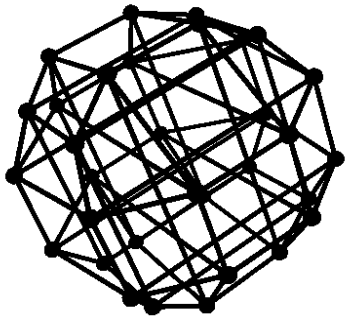
13 pav. Pareto aibė, gauta algoritmu sMOCell2.



14 pav. Pareto aibė, gauta algoritmu sMOCell1.

Pastebėkime, kad tolygiausiai taškus išdėsto sMOCell1 algoritmas. Taip pat su sMOCell1 algoritmu gaunamas mažiausias hiperplotas. Kadangi visi minėtieji algoritmai dirba apie šešias minutes, tai efektyviausiu algoritmu galime laikyti tą algoritmą, kuris tolygiausiai išdėsto taškus Pareto aibėje, šiuo atveju tai yra sMOCell1.

Šio magistro darbo užduotis yra atvaizduoti penkiamatį kubą taikant ląstelinius genetinius algoritmus, kai minimizuojamos tikslo funkcijos atitinka kvadratinę ir absoliutinę paklaidas. Žemiau pateikiamas penkiamačio kubo vaizdas, gautas ląsteliniu genetiniu algoritmu sMOCell1. Taip pat reikia paminėti, kad visi trys eksperimentiškai tirti algoritmai realizuoti Java programavimo kalba.



15 pav. Dvimatis penkiamačio kubo vaizdas, gautas taikant ląstelinį genetinį algoritmą sMOCell1

REZULTATAI IR IŠVADOS

1. Sukurtos trijų ląstelinių genetinių algoritmų programos, tai yra: sMOCell1, sMOCell2, sMOCell3, visi trys minėtieji algoritmai realizuoti Java programavimo kalba.
2. Testuojant algoritmus sMOCell1, sMOCell2, sMOCell3, parodyta, kad efektyviausias algoritmas yra sMOCell1.
3. Taikant minėtuosius algoritmus vizualizavimo uždaviniui, efektyviausias daugiamačių duomenų vizualizavimo uždaviniui buvo sMOCell1 algoritmas.
4. Sprendžiant daugiamačių duomenų vizualizavimo uždavinį, OpenGL grafikos biblioteka pritaikyta ir pasirodė tinkama duomenų atvaizdavimui, o atvaizduotiems duomenims peržiūrėti pasitelkta viena iš stereoskopijos priemonių – erdvinis ekranas.

ŠALTINIAI

[AD+02] S. Agarwal, K. Deb, T. Meyarivan, A. Pratap. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evoliutionary Computation, Vol. 6, No.2, 2002, p. 182 -197.

[AD08] E.Alba, B.Dorronsoro. Cellular Genetic Algorithms. Springer, 2008, 248 pages.

[AD+06] E. Alba, B. Dorronsoro, J. Durillo, F. Luna, A. Nebro. MOCell: A Cellular Genetic Algorithm for Multiobjective Optimization. International Journal of Intelligent Systems, 2006, pp. 726 -746.

[Bra11] 15 Real-World Uses of Genetic Algorithms.
[žiūrėta 2011-11-04]. Prieiga per internetą:
<http://brainz.org/15-real-world-applicationsgeneticalgorithms/ >

[Bro11] J.Brownlee. Clever Algorithms: Nature-Inspired Programming Recipes. LuLu, 2011, 436 pages.

[Čyr08] V.Čyras. Dirbtinis intelektas (paskaitų konspektas). Vilnius, 2008, 141 p.

[Del10] F. Delcroix. Multi-objective Optimization – Design of Control Arm Using HyperWorks.
[žiūrėta 2012-02-20]. Prieiga per internetą:
<http://insider.altairhyperworks.com/articles/multi-objective-optimization-design>

[DKŽ08] G.Dzemyda, O.Kurasova, J.Žilinskas. Daugiamačių duomenų vizualizavimo metodai.Mokslo aidai, Vilnius, 2008, 206 psl.

[DŠT07] G. Dzemyda, V. Šaltenis, V. Tiešis. Optimizavimo metodai. Mokslo aidai, Vilnius, 2007, 214 p.

[GL97] F. Glover, M. Laguna. Tabu Search. Kluver, Boston, 1997.

[Gpt11] The GP Tutorial.
[žiūrėta 2011-11-04]. Prieiga per internetą:
<http://www.geneticprogramming.com/Tutorial/ >

[KO96] I. Osman, J. Kelly. Meta-Heuristics: Theory and Applications. Kluver, Boston, 1996.

[Koc08] A.Kovacs. Solving the Vehicle Routing Problem with Genetic Algorithm and
Simulated Annealing. Sweden, 2008, 108 pages.

[LL09] G. Liutkus, A. Lenkevičius. Kompiuterinė grafika su OpenGL.Technologija, Kaunas, 2009, 246 p.

[LT+01] M. Laumamanns, L. Thiele, E. Zitzler. SPEA2: Improving the Strength Pareto Evoliutionary Algorithm. Swiss Federal Institute of Technology, 2001, 21 pages.

[VZV09]A.Varoneckiene, A.Zilinskas, A.Varoneckas. Multidimensional Scaling: Multi-Objective Optimization Approach.International Conference on Computer Systems and Technologies, 2009, 6 pages.

[Zel11] Zell. Evolutionary algorithms.
[žiūrėta 2011-11-20]. Prieiga per internetą:
<http://www.ra.cs.unituebingen.de/software/JCell/tutorial/ch03s05.html>