VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Magistro baigiamasis darbas

Atskaitos taškais grįsti daugelio kriterijų optimizavimo evoliuciniai algoritmai

Reference point-based evolutionary algorithms for many-objective optimization

Atliko:
Viktoras Laukevičius (parašas)

Darbo vadovas:
doc. dr. Olga Kurasova (parašas)

Recenzentas:
prof. dr. Linas Laibinis (parašas)

Vilnius
2019

Santrauka

Daugiakriterinio optimizavimo uždaviniai susideda iš kelių prieštaringų kriterijų, išreikštų matematinėmis funkcijomis. Daugelio, keturių ir daugiau, kriterijų uždaviniai yra sudėtingai išsprendžiami, kadangi su padidėjusiu optimizavimo funkcijų skaičiumi atsiranda daugiau taškų, kuriuos reikia įvertinti aproksimuojant Pareto frontą. Šis darbas apžvelgia įvairius algoritmus, daugelio kriterijų uždaviniams spręsti, bei plačiai analizuoja NSGA-III algoritmą, nagrinėdamas sprendinių pasiskirstymo bei konvergavimo į tikrąjį Pareto frontą aspektus. Argumentuotai pristatoma modifikacija, įtraukianti elitinių narių archyvą, kuris yra naudojamas aplinkos išrinkimo žingsnyje. Panaudotos euristikos įtaka analizuojama įvairiomis metrikomis, o apversto apibendrinto atstumo rezultatai yra lyginami tiek su originalaus algoritmo, tiek su naudojamos realizacijos pateiktais rezultatais. Gautieji rezultatai patvirtina elitinių narių euristikos panaudojimo NSGA-III algoritme efektyvumą atskleisdami dar daugiau galimybių idėjos vystymui.

Raktiniai žodžiai: daugiakriterinis optimizavimas, daugelio kriterijų optimizavimas, evoliuciniai algoritmai, elitiškumas, atskaitos taškai, Pareto frontas.

Summary

Multiobjective optimization problems consist of multiple conflicting criteria using mathematical functions as formalization. Many-objective (4 or more objective) optimization problems are harder to solve due to increased size of points that have to be evaluated and construct approximated Pareto front. This work reviews multiple algorithms used for multiobjective optimization, furthermore NSGA-III algorithm is analysed thoroughly evaluating even distribution of solutions and set convergence to true Pareto front. A new procedure including elite member archive and is used in environment selection step is presented in a reasoned manner. Elitism procedure is analysed using various metrics. Moreover, inverted generational distance results are used to compare to both originally presented results and results of used implementation. Achieved results confirm effectiveness and usefulnes of presented elitism heuristic procedure in NSGA-III algorithm revealing yet more room for improvement for development of the procedure.

Keywords: multiobjective optimization, many-objective optimization, evolutionary algorithms, elitism, reference points, Pareto front.

Turinys

Įvadas

Optimizavimas yra procesas, leidžiantis pasiekti geresnių, greitesnių, stipresnių ar kitaip pranašesnių rezultatų priklausomai nuo taikomosios srities. Daugiakriterinis optimizavimas yra kelių kriterijų optimizacija. Daugiakriterinio optimizavimo uždavinius dažnai sutinkame gyvenime. Keliaujant iš namų į darbą automobiliu yra įvertinama, kiek laiko užtruks stovėjimas eismo spūstyse ir kokį atstumą teks nuvažiuoti. Turint tai omenyje kartais yra nusprendžiama pasirinkti ilgesnį kelią, kuriame nebus susidurta su eismo spūstimis, tikintis, jog išnaudojus daugiau kuro bus sutaupyta laiko. Klasikinis buitinis daugiakriterinio optimizavimo uždavinys yra kainos bei kokybės pusiausvyros sprendimas. Perkant daiktus nuo mobilaus telefono iki būsto yra trokštama kuo didesnės kokybės, tačiau norima išleisti kuo mažiau. Paminėti uždaviniai dažniausiai yra sprendžiami intuityviai ir net nesusimąstant apie tai, jog tai yra matematiškai nagrinėjami uždaviniai.

Matematiškai apibrėžtų daugiakriterinio optimizavimo uždavinių sprendimai, žinoma, vaidina svarbų vaidmenį įvariose taikymo srityse. Nuo elektros energijos sistemų planavimo [GLM12], reikalaujančio sunaudojant kuo mažiau finansinių kaštų išgauti kuo daugiau elektros energijos bei paskirstyti ją vartotojams kuo efektyviau, iki investavimo portfelių sudarymo [FEO13] didinant pelno galimybes tuo metu mažinant rizikos faktorius atsižvelgiant į vertybinių popierių kainų nepastovumą ir bendrą rinkos situaciją. Daugiakriterinis optimizavimas taikomas net ir tokiose nišinėse ir siaurose srityse kaip kriketo komandos komplektacija [FJD11], kur turint ribotą biudžetą reikia rasti kuo universalesnius žaidėjus, kuriuos renkantis reikia atsižvelgti į smūgiavimo bei ridenimo stiprumo savybes.

Šių uždavinių kompleksiškumas slypi jų kombinatorinėje kilmėje, kadangi ieškant optimalaus sprendimo reikia peržiūrėti daugybę galimų alternatyvų, o tai reiškia, jog taikant brutalios jėgos metodą reikia perrinkti visus galimus laisvų kintamųjų variantus. Tai reikalauja daug laiko kaip ir daug skaičiavimo resursų net ir palyginamai mažos kintamųjų aibės apimties sistemose.

Daugiakriterinio optimizavimo kriterijai yra savybės, kurių reikšmes norima pasiekti kuo didesnes arba kuo mažesnes. Nuo šių savybių priklauso optimizavimo funkcijų kintamieji. Gaminant produktą kriterijais galima laikyti išlaidų kiekį bei gautas pajamas, akivaizdu, jog norint pasiekti kuo didesnį pelną reikia didinti gaunamas pajamas bei mažinti išlaidas. Daugiakriteriniame optimizavime iš aibės kriterijų bent du yra prieštaringi, kas reiškia, jog negalima gauti geresnių rezultatų pagal vieną kriterijų nepabloginant rezultatų pagal kitą kriterijų.

Sprendžiant daugiakriterinio optimizavimo uždavinius itin svarbu yra teisingai interpretuoti gautus rezultatus. Vieno kriterijaus optimizacija neturi vietos interpretacijai, nes gautas rezultatas vienareikšmiškai apibrėžia optimizuojamo kriterijaus gerumą. Kitaip nei vieno kriterijaus opti-

mizacijos rezultatuose, daugiakriterinės optimizacijos rezultatas vienareikšmiškos interpretacijos neturi dėl to, jog vieno kriterijus optimizacija yra pasiekta pabloginus kito kriterijaus įvertį. Grįžtant prie pradinio, kelionės iš namų į darbą, pavyzdžio pasirenkant ilgesnį kelią galima sutaupyti laiko, tačiau tai kainuos eksploataciją.

Daugiakriterinių optimizacijų uždavinių sprendimo subjektas, interpretuojantis gautus rezultatus, yra vadinamas sprendimų priėmėju. Sprendimų priėmėjas, veikiantis sistemose, gali daryti įtaką optimizavimui keičiant kriterijų optimizavimo svarbumą. Šios sistemos yra vadinamos sprendimų paramos sistemomis ir jos leidžia peržiūrėti gautus rezultatus po optimizacijos bei sprendimų priėmėjui parinkti tenkinamą sprendimą. Taip pat šios sistemos leidžia optimizacijos metu interaktyviai keisti optimizuojamų kriterijų svarbumą.

Turint omenyje, kuriuo metu yra priimamas sprendimas dėl kriterijų svarbumo ir sprendimo priėmimo bei optimizacijos, daugiakriterinio optimizavimo metodai yra skaidomi į tris grupes [HM79]:

- Sprendimo priėmimas prieš optimalaus sprendimo paiešką: Daugiakriterinio optimizavimo tikslai yra apjungiami į vieną tikslą, kuris savyje nešasi informaciją apie sprendimų priėmėjo nurodymus;

- Optimalaus sprendimo paieška prieš sprendimo priėmimą: Optimizacija vykdoma nežinant nieko apie sprendimų priėmėjo reikalavimus ir troškimus. Galutiniai rezultatai yra gaunami ir pateikiami sprendimų priėmėjui, kuris tada pasirenka jį tenkinantį optimizacijos sprendimą;

- Sprendimo priėmimas atliekant paiešką: Atliekant optimizaciją sprendimų priėmėjas gali paieškos eigoje keisti nurodymus matydamas, jog optimizacija vyksta ne taip, kaip tikėtasi. Po kiekvieno optimizacijos žingsnio sprendimų paramos sistema sprendimų priėmėjui pateikia kompromisus, pagal kuriuos jam leidžiama interaktyviai keisti kriterijų pirmenybę ir taip vadovauti paieškai.

Matematiškai formalizuojant daugiakriterinio optimizavimo sprendimų paiešką ji išreiškiama

taip:

$$
\begin{aligned}
\text{minimizuoti/maksimizuoti} \quad & y = F(x) = [f_1(x), f_2(x), ..., f_M(x)] \\
\text{esant apribojimams} \quad & x \in S. \\
& S = \{ \\
& \quad g_j(x) \leq 0, \qquad j = 1, 2, ..., J; \\
& \quad h_k(x) = 0, \qquad k = 1, 2, ..., K; \\
& \},
\end{aligned}
$$

kur, laikant, jog sprendinys turi V kintamųjų, x yra V-matis vektorius, vadinamas sprendinių vektoriumi. Šis vektorius priklauso sprendinių kintamųjų erdvei S, apribotai nelygybių bei lygybių funkcijų g_j ir h_k, kurių yra J ir K atitinkamai. Visi sprendiniai, patenkantys į S erdvę, yra laikomi leistinais sprendiniais, o nepatenkantys - neleistinais. Sprendimo vektoriaus įvertis y pritaikius funkciją F yra vadinamas tikslo vektoriumi. Kiekviena optimizacijos funkcija f_i, vadinama tikslo funkcija, gali įgyti aibę Y_i reikšmių. Tokiu būdu visos tikslo funkcijų reikšmių aibės Y_i, kur $i = 1, 2, ..., M$, sukuria M-matės dimensijos erdvę, vadinamą tikslo erdve.

Lyginant tikslo funkcijų reikšmes, jos turi būti lyginamos kriterijaus kontekste, todėl žymima $f_t(x^{(1)}) \triangleleft f_t(x^{(2)})$, kai $x^{(1)}$ sprendinys yra geresnis už $x^{(2)}$, sprendinį kriterijaus, kurio funkcija yra f_t, atžvilgiu. Tai reiškia, jog reikšmė $x^{(1)}$ yra mažesnė už $x^{(2)}$ jei funkcija f_t yra minimizuojanti, arba didesnė, jei maksimizuojanti. Analogiškai yra žymima $f_t(x^{(1)}) \triangleright f_t(x^{(2)})$, kai $x^{(2)}$ sprendinys yra geresnis už $x^{(1)}$ sprendinį. Lygybė $f_t(x^{(1)}) = f_t(x^{(2)})$ reiškia, kad kriterijaus įverčiai yra lygūs.

Sprendimų vektoriai gali būti lyginami tarpusavyje visos daugiakriterinio optimizavimo funkcijos F kontekste. Yra teigiama, jog įmanomas sprendiniais $x^{(1)}$ yra dominuojantis prieš kitą įmanomą sprendinį $x^{(2)}$ tada ir tik tada, jei:

1. Sprendimas $x^{(1)}$ yra ne blogesnis nei $x^{(2)}$ pagal visas f_i funkcijas, kur $i = 1, 2, ..., M$.

2. Sprendimas $x^{(1)}$ yra griežtai geresnis nei $x^{(2)}$ pagal bent vieną $f_{i'}$ funkciją, kur $i' = 1, 2, ..., M$.

Savo ruožtu $x^{(2)}$, kadangi $x^{(1)}$ yra dominuojantis prieš $x^{(2)}$, yra dominuojamas sprendinio $x^{(1)}$. Matematiškai tai yra išreiškiama kaip $x^{(1)} \prec x^{(2)}$ arba $x^{(2)} \succ x^{(1)}$.

Žinant apie sprendinių kintamųjų erdvę bei sprendinių dominavimo sąryšį galima pastebėti, jog sprendinių kintamųjų erdvė gali turėti begalę sprendinių, kurie yra dominuojami sprendinio x^*. Taip pat tokių sprendinių yra daugybė. Kitaip tariant, aibės sprendinių, kurie yra dominuojantys

prieš likusią sprendinių kintamųjų erdvėje aibę, negalima optimizuoti pagal vieną tikslo funkciją nepabloginant sprendinių, pagal kitą tikslo funkciją. Šitokių sprendinių aibė yra vadinama Pareto optimalių (nedominuojamų) sprendinių aibe. Matematiškai Pareto optimalių sprendinių aibė P^* išsireiškia kaip:

$$P^* = \{\ x^* \in S \mid\ \nexists x' \in S : x' \prec x^*\ \}.$$

Daugiakriterinio optimizavimo funkcijos F reikšmės Pareto optimaliems sprendiniams iš aibės P^* sudaro aibę, vadinamą Pareto frontu. Matematiškai Pareto frontas PF^* formalizuojamas kaip:

$$PF^* = \{\ F(x^*) \mid\ x^* \in P^*\ \}.$$

Spręsti daugikriterinio optimizavimo uždavinius tradiciniais algoritmais yra sudėtinga dėl jau minėtos kombinatorinės uždavinių kilmės. Evoliuciniai algoritmai apjungia kelias euristikas, kurios pasitelkdamos natūralios evoliucijos idėjas padeda spręsti daugiakriterinio optimizavimo uždavinius. Didžiausias evoliucinių algoritmų pranašumas yra tas, jog jiems nėra būtinos žinios apie išsamias bei konkrečias taikomosios srities detales. Šių algoritmų veikimas remiasi tik žiniomis apie tinkamumo, kitaip vadinamą taikinio, funkciją. Papildomos detalės taip pat gali būti įtrauktos į evoliucinių algoritmų įgyvendinimą, kurios padeda tiek atrasti tikslesnius rezultatus, tiek atlikti paiešką greičiau. Evoliuciniai algoritmai veikia vykdydami generacijas. Kiekvienoje generacijoje atrenkami geriausi sprendiniai bei konstruojamos sekančios generacijos. Sekančių generacijų rezultatai gaunami sujungiant atrinktus geriausius rezultatui bei papildant juos naujais, sugeneruotais geriausių praeitos generacijos sprendinių pavyzdžiu atliekant išrinkimo, kryžminimo bei mutacijos operacijas.

Daugiakriterinio optimizavimo uždavinių specifikoje kiekvienos generacijos rezultatas yra aibė sprendinių, o tinkamumo funkcijos reikšmėmis yra laikomos Pareto fronto reikšmės. Kiekvienoje generacijoje bandoma perkombinuojant bei mutuojant geriausius praeitos generacijos sprendinius gauti naujus sprendinius, mažinančius atstumą nuo gautųjų sprendinių iki Pareto fronto.

Evoliucinių daugiakriterinio optimizavimo algoritmų efektyvumas įprastai matuojamas pagal du aspektus: tikslumą ir tolygų gautųjų reikšmių pasiskirstymą. Algoritmo tikslumo matas parodo kaip arti Pareto fronto yra gautos aproksimuotos funkcijos reikšmės. Algoritmo gautų rezultatų tolygaus pasiskirstymo matas parodo kaip tolygiai pasiskirsčiusios yra aproksimuotos funkcijos reikšmės. Dažniausiai naudojami matai:

- Apverstas apibendrintas atstumas (angl. *Inverted generalized distance*) yra matas, skaičiuo-

jantis vidutinį atstumą nuo aproksimuoto Pareto fronto taškų iki jiems artimiausių tikrojo Pareto fronto taškų.

- Hipertūris (angl. *Hypervolume*) yra dominuojančio regiono, gauto iš gautųjų tikslo vektorių aproksimuojant Pareto frontą bei nurodyto taško, tūris. hipertūrio reikšmė parodo tiek aproksimuoto Pareto fronto tikslumą, tiek ir reikšmių pasiskirstymo tolygumą.

- Apibendrintas atstumas (angl. *Generalization distance*) yra atstumas tarp gautųjų vektorių, aproksimuojant Pareto frontą, bei artimiausių jiems vektorių iš tikslo funkcijos (Pareto fronto). Mažesnis gautas apibendrintas atstumas reiškia tiksliau aproksimuotą Pareto frontą, taigi siekiama, jog apibendrintas atstumas būtų kuo mažesnis. Šis matas vertina tiek tolygų pasiskirtymą, tiek ir sprendinių konvergavimą į tikrąjį Pareto frontą.

- Išsibarstymas (angl. *Spread*) yra matas, parodantis kaip netolygiai išsibarsčiusios aproksimuotos Pareto fronto reikšmės. Siekiama kuo mažesnė išsibarstymo reikšmė.

Kuriant bei tobulinant daugiakriterinio optimizavimo algoritmus įprastai jų patikimumas vertinamas sprendžiant gerai žinomus, testinius, daugiakriterinio optimizavimo uždavinius. Šie uždaviniai sudaryti pasitelkiant skirtingas uždavinių charakteristikas, kompleksiškumą, nevienodumą. Daugelis iš uždavinių turi kelis lokalius Pareto frontus. Keli rinkinių pavyzdžiai yra ZDT1-ZDT4 [EKL00] bei DTLZ1-DTLZ7 [DTL+05] - keičiamo dydžio tikslo funkcijų uždaviniai.

Literatūros analizė parodė, jog atskaitos taškai bei vektoriai padeda išlaikyti populiacijos įvairovę bei yra pakankamas įrankis leidžiantis populiacijoms konverguoti į Pareto frontą. Atskaitos vektoriais grįsti algoritmai turi gerą savybę, jog vektorių panašumas gali būti lyginamas pagal kampo dydį tarp jų (3.3 poskyris). Tap pat, viena dažniausiai naudojamų euristikų įvairuose algoritmuose, susigrūdinimo matas (2.3 poskyris) pateikia gerus rezultatus ieškant per daug tankių regionų bei yra naudojamas daugelyje algoritmų. Evoliuciniai algoritmai turi daugybę būdų įgyvendinti pagrindines operacijas (1 skyrius), taigi algoritmų modifikacijos gali remtis įvairiais galimais pasirinkimais.

Šio darbo tikslas yra atrasti perspektyvias atskaitos taškais grįstų daugelio kriterijų optimizavimo evoliucinių algoritmų modifikacijas bei jas pritaikyti kuriant naująjį algoritmą, pranašumą pagrindžiant algoritmų efektyvumų matais.

Šio darbo uždaviniai yra:

- Identifikuoti efektyvias bei perspektyvias įvairių algoritmų dalis, tinkančias naująjam algoritmui;

- Pasinaudojus pagrindinėmis atskaitos vektorių paremtų algoritmų idėjomis pritaikyti modifikacijas;
- Ištirti kaip kinta algoritmų efektyvumas, remiantis tikslo funkcijos vykdymo skaičiumi, pritaikius skirtingas modifikacijas;
- Ištirti kaip kinta algoritmų efektyvumas kitais aspektais, kurie gali lemti algoritmo darbo greitį.

1. Evoliuciniai algoritmai

Kaip jau buvo minėta anksčiau, vykdant evoliucinius algoritmus yra atliekamos trijų tipų operacijos su populiacijų nariais, tai yra išrinkimas, mutacija bei kryžminimas. Nors šios operacijos yra esminis evoliucinių algoritmų veikimo pagrindas, kiti aspektai kaip pradinės populiacijos generavimas, kandidatų reprezentavimas bei elitiškumas yra neatsiejamos šių algoritmų sudedamosios dalys ir atlieka svarbų vaidmenį [Sim13]. Vykdant evoliucinius algoritmus populiacijos kartais konverguoja į vieną sprendinį, taip sudarydamos populiaciją beveik identiškų sprendinių. Sprendžiant šį uždavinį algoritmui galima nurodyti išplėsti paiešką, tačiau tokiu atveju jis gali užstrigti ir nebekonverguoti į tenkinamus sprendinius. Šiais aspektais evoliucinių algoritmų veikimo bei daugiakriterinių uždavinių optimizavimo kriterijai sutampa - sprendiniai turi būti kuo labiau tolygiai pasiskirstę per sprendinių aibę ir kuo arčiau tikslo funkcijos reikšmių.

1.1. Inicializacija

Pradedant bet kurį evoliucinį algoritmą yra sugeneruojama pradinė populiacija. Paprasčiausias būdas tai pasiekti - visiškai atsitiktinai sugeneruoti populiacijos dydžio individų skaičių. Inicializacija atrodo elementarus žingsnis, tačiau įdėjus šiek tiek daugiau darbo galima pagelbėti algoritmui nurodant tinkamesnius parametrus bei sukuriant tinkamesnę pradinę populiaciją konkretiems uždaviniams.

Remiantis atsitiktiniu generavimu galima patobulinti procedūrą - sugeneruoti kelis kartus daugiau individų nei nustatytas populiacijos dydis ir lokaliai optimizavus išfiltruoti tik geriausius individus, kurie ir bus laikomi pradiniais duomenimis. Lokalus optimizavimas gali būti pasiektas keliais būdais, vienas iš jų - gradientinis nusileidimas (angl. *gradient descent*). Dar vienas, labiau specifinis kiekvienai taikomąjai sričiai bei daugiau rankinio darbo reikalaujantis, metodas, vadinamas tiesiogine inicializacija (angl. *direct initialization*), siūlo pasinaudoti ekspertų žiniomis generuojant pradinę populiaciją. Ekspertai gali pateikti pagrįstai priimtinus sprendinius, kuriuos galimai norėtume optimizuoti, naudojamus pradinėje populiacijoje. Taip pat informacija, tai yra galimai optimizuoti sprendiniai, iš išorinių šaltinių gali būti panaudoti kaip pradinė populiacija.

1.2. Elitiškumas

Elitiškumas (angl. *elitism*) - būdas užtikrinti, jog geriausi populiacijos individai išgyvena iš generacijos į generaciją. Įprastai evoliuciniuose algoritmuose geriausieji tėvai kryžminami bei mutuojami tam, jog būtų sukurtas palikuonis, o tėvai nebeįtraukiami į sekančią generaciją. Tai atitinka

biologinę evoliuciją, tačiau tokiu būdu gali būti prarandami elitiniai individai, jei palikuonys nebėra geresni už savo tėvus. Elitiškumui užtikrinti dažniausiai yra naudojamas išorinis archyvas (angl. *external archive*), kuriame geriausieji individai yra arba kopijuojami iš pagrindinės populiacijos arba perkeliami, taip sudarant populiacijų plėtinius. Taip pat galimos strategijos, kai prasčiausi individai populiacijoje keičiami naujais, o tėvai nėra šalinami iš populiacijos.

1.3. Populiacijos įvairovė

Evoliucijos metu populiacija yra rekombinuojama iš generacijos į generaciją. Nebandant užtikrinti populiacijos įvairovės (angl. *population diversity*) sprendinių pasiskirstymas mažėja ir su kiekviena iteracija sprendiniai tampa panašūs į vieni kitus. Padidinus mutacijos tikimybę galima to išvengti, bet turint dažną mutaciją algoritmas panašėja į atsitiktinę paiešką. Įprastas būdas yra apibrėžti faktorių, pagal kurį individai yra laikomi per daug panašiais. Panašūs individai yra arba išmetami iš populiacijos arba yra perdaromi. Per daug išplėtus populiaciją galima prieiti prie kitos problemos - per daug skirtingų individų. Dvi išvardintos geros populiacijos savybės yra panašios į biologinę evoliuciją: nepastebimas poravimasis tarp individų iš skirtingų rūšių, tačiau tos pačios giminės individai nesiporuoja. Poravimui tinkamoms grupėms (angl. *mating pool*) sudaryti dažniausiai naudojamas nišos metodas (angl. *niching*), kuris skirsto individus, tinkamus poruotis tarpusavyje, į vadinamąsias nišas.

1.4. Išrinkimas

Išrinkimas (angl. *selection*), kitaip algoritmuose dar vadinamas aplinkos išrinkimu (angl. *environmental selection*), atsakingas už tinkamų, stipriausių individų atranką. Populiariausias ir paprasčiausias yra ruletės rato (angl. *roulette-wheel*) metodas, kai pagal tinkamumo reikšmes, stipresni individai turi didesnius šansus būti išrinkti. Vienas pagrindinių kriterijų išrinkimo operatoriui - priešlaikiškai nekonverguoti į Pareto frontą ištiriant kuo daugiau galimų sprendinių ankstyvose generacijose, tačiau velėsnėse generacijose neužstrigti ir išnaudoti stipriuosius individus konvergavimui į frontą. Ruletės ratas kartais neįgyvendina šių reikalavimų taip gerai kaip norėtųsi, todėl yra alternatyvių išrinkimo būdų:

- Stochastinis visuotinis atrinkimas (angl. *stochastic universal sampling*) labai panašus į ruletės rato metodą, kadangi ratas padalintas taip pat pagal tikimybes. Tik šiuo atveju išrinkimas vykdomas ne kelis kartus sukant ratą ir turint vieną rodyklę, kuri išrenka individą, o rate sukūrus vienodai pasiskirsčiusias rodykles ir išsukus ratą vieną kartą.

- Sigma mastelio keitimas (angl. σ *scaling*) normuoja tinkamumo reikšmes priklausomai nuo standartinio visos populiacijos nuokrypio.

- Rangu paremtas (angl. *rank-based*) išrinkimas paskirsto išrinkimo tikimybes pagal rangą, o ne pagal absoliučias tinkamumų reikšmes.

- Tiesinis rangu paremtas (angl. *linear ranking*) išrinkimas yra rangu paremto metodo apibendrinimas, kai apibrėžiama tiesinė funkcija, priklausanti nuo individo rango.

- Turnyras (angl. *tournament*) suformuoja turnyrą (mažesnio dydžio rinkinį nei populiacija) iš geriausių sprendinių ir atlieka įprastą išrinkimą turnyro individų rinkinyje. Pagrindinis pranašumas, jog nereikia skaičiuoti tikslių tinkamumo reikšmių turnyrui sudaryti, reikia žinoti tik reliatyvias reikšmes.

- Tiesmukiškas, tačiau kartais naudojamas, yra lyderio išrinkimas (angl. *stud selection*), kuris kiekvienoje generacijoje išrenka geriausią individą, o jam į poras išrenkamas individas įprastu būdu.

1.5. Kryžminimas

Atlikus išrinkimą, išskirti individai tampa tėvais ir yra kryžminami vieni su kitais. Algoritmuose taip pat tai vadinama pertvarka (angl. *recombination*) arba porinimu (angl. *mating*). Dažniausias yra vieno taško (angl. *single point*) kryžminimas dvejetainėje sekoje paimant fiksuotą arba atsitiktinį tašką ir abi tėvų sekas padalijant į dvi dalis. Palikuonio dvejetainė seka sudaroma paimant pirmąją dalį iš pirmojo tėvo, o antrąją - iš antrojo. Kryžminimas gali būti įgyvendintas ir kitais, alternatyviais, būdais:

- Dalių (angl. *segmented*) kryžminimas yra panašus į vieno taško kryžminimą, tik čia imami keli taškai, taip sudarydami segmentus dvejetainėje sekoje. Kiekvienas segmentas imamas iš vieno iš tėvų atsitiktiniu būdu.

- Įprastai kryžminimas vyksta tarp dviejų tėvų, bet kartais yra naudojamas kelių tėvų (angl. *multi-parent*) metodas. Tai dažnai įgyvendinama kaip dalių kryžminimas tik daugiau nei su dviem tėvais.

- Linijinis (angl. *linear*) kryžminimas sukuria kelis palikuonis, bet išlaiko mažiau nei buvo sukurta, priklausomai nuo jų tinkamumo.

- Simuliuojamas dvejetainis (angl. *simulated binary, SBX*) kryžminimas atsižvelgia į tėvų dvejetaines išraiškas ir jų charakteristikas (pvz., binarinių reikšmių vidurkį). Gaunami vaikai bandant išlaikyti tėvų binarinių išraiškų charakteristikas.

1.6. Mutacija

Mutacija biologinėje evoliucijoje vyksta labai retai, dėl to evoliuciniuose algoritmuose mutacija yra vykdoma labai retai - dažniausiai naudojama apie 2 % tikimybė. Nepaisant to, mutacija yra svarbus algoritmo operatorius, suteikiantis galimybę atrasti galimai naujus sprendinius, nes trūkstama genetinė informacija gali būti atsitiktinai sugeneruota mutacijos procese. Įprastas būdas mutacijai atlikti yra individo, gauto po kryžminimo, dvejetainėje išraišoje kiekvieną bitą, su nedidele tikimybe, apversti. Alternatyvūs mutacijos metodai dažniausiai sprendžia ne kada, o ką mutuoti. Vienas iš būdų - užkoduoti sprendinį dvejetaine forma ir pasirinkti bito reikšmę iš sugeneruoto tolygaus (angl. *uniform*) arba Gauso pasiskirstymo, kurio vidutinė reikšmė yra ieškomos taikomosios srities centras. Taip pat vidutinė reikšmė gali būti laikoma ir nepakeisto bito reikšmė.

2. Daugiakriterinio optimizavimo algoritmai

Daugiakriterinio uždavinių optimizavimo algoritmai skirstomi grubiai į tris grupes: Pareto dominavimu grįsti, dekompozicija grįsti bei indikatoriumi grįsti algoritmai [GD04]. Pareto aibe grįsti algoritmai yra ypač efektyvūs sprendžiant mažos dimensijos uždavinius, kadangi jie vienu perbegimu gali rasti aproksimuotą aibę sprendinių, naudojantis jų populiacija grįsta savybe. Šios grupės reprezentaciniai daugiakriterinio optimizavimo algoritmai yra NSGA-II (2.3 poskyris) bei SPEA2 (2.4 poskyris). Antroji grupė algoritmų dalina komplikuotą daugiakriterinio optimizavimo uždavinį į kelis dalinius uždavinius ir sprendžia juos atskirai. Dalinių uždavinių sprendimai dalijasi informacija tarpusavyje. Šios grupės plačiausiai naudojamas algoritmas yra MOEA/D (2.5 poskyris). Trečioji grupė jungia efektyvumu arba indikatoriumi grįstus (angl. *indicator-based*) algoritmus. Labiausiai žinomi šios grupės algoritmai yra IBEA, kurio modifikacija mIBEA bus apžvelgta (2.6 poskyris).

Suprasti algoritmų, skirtų daugiakriteriniams uždaviniams bei grįstų genetinėmis operacijomis, ištakas, šiame skyriuje nagrinėjamas algoritmas VEGA (2.1 poskyris). Taip pat, ištakos, vieno labiausiai naudojamų algoritmų NSGA-II, yra nagrinėjamos tiriant pirmąją NSGA (2.2 poskyris) versiją.

2.1. VEGA

Vienas pirmųjų genetinių algoritmų skirtų daugiakriteriniams uždaviniams spręsti buvo VEGA algoritmas, padėjęs pamatus daugeliui naujų algoritmų [Sch85]. Pirmoji šio algoritmo versija modifikavo GENESIS algoritmą įvelkant įprastą išrinkimo procedūrą į ciklą ir vykdant jį nepriklausomai. Šis ciklas leido atlikti išrinkimą kiekvienam kriterijui atskirai, taip užpildant poravimo aibę. Atlikus perrinkimą pagal kiekvieną kriterijų visai gautųjų sprendinių populiacijai pritaikomos kryžminimo bei mutacijos operacijos. Ši procedūra atliekama tam, jog būtų suporuoti individai iš skirtingų populiacijų.

Kaip teigia K. Deb, VEGA algoritmas veikia gerai ne visada, kadangi kai kurios generacijos padaro didelę įtaką sprendinių konvergavimui į vieną sprendinį ar regioną sprendinių. VEGA algoritmo autorius Schaffer badė minimizuoti šį šališkumą įvedant dvi euristikas - nedominuojamų sprendinių euristiką bei partnerio išrinkimo euristiką. Nedominuojamų sprendinių euristika naudojasi turto perskirstymo schema. Ši euristika pritaiko nuobaudą atimant fiksuoto dydžio svorį iš kiekvienos sprendinio kopijos populiacijoje vykdant perrinkimą. Visa nuobaudų suma už dominuojamus narius tada yra perskirstoma nedominuojamiems individams pridedant prie jau sukauptos nuobaudos už kopijas populiacijoje. Tačiau ši euristika nepasiteisina, kai populiacijoje yra nedaug

nedominuojamų sprendinių, kadangi tai reiškia jog tinkamumo funkcijos reikšmės šiems sprendiniams yra didelės, o tai priveda prie atrankos spaudimo. Taikant partnerio išrinkimo euristiką yra naudojamasi veislių kryžminimo schema. Šia euristika buvo siekiama sukryžminti narius iš skirtingų pogrupių. Tai buvo įgyvendinama pasirenkant individą bei atsitiktinai pasirenkant kitą individą, kuris turi didžiausią Euklido atstumą našumo erdvėje nuo pirmojo pasirinkto individo. Šie nariai paskelbiami partneriais. Deja, bet ši euristika nepasiekė tikslo apriboti prastesnių narių dalyvavimo partnerių sudaryme. Euristika nepasiteisino, kadangi pirmasis narys procedūroje yra renkamas atsitiktinai bei atstumas tarp čempiono ir vidutinio populiacijos nario gali būti didelis.

Euristikai nepasiteisinus buvo iškeltos kitos dvi idėjos: nedominuojamų sprendinių rikiavimo procedūra bei tinkamumo funkcijos reikšmės dalijimas. Buvo manoma, kad šios idėjos padės išvengti šališkumo bei padės palaikyti tolygų pasiskirstymą. Šiomis idėjomis vadovaujantis sukurtas algoritmas, atnešęs laukiamų rezultatų, buvo NSGA.

2.2. NSGA

Atsižvelgdamas į VEGA algoritmo trūkumus ir stengdamasis juos eliminuoti K. Deb sukūrė NSGA (angl. *Nondominated sorting genetic algorithm*) algoritmą. Pagrindinė algoritmo idėja yra šio algoritmo vidinė procedūra vertinanti bei reitinguojanti nedominuojamus sprendinius taip, jog yra išryškinami labiausiai tinkantys sprendiniai [SD94]. Taip pat ši procedūra naudojasi nišos metodu siekiant išlaikyti dalinę labiausiai tinkamų sprendinių populiaciją. Nuo bendrinio genetinio algoritmo NSGA skiriasi tik savo išrinkimo procedūra, kadangi mutacijos bei kryžminimo operacijos yra atliekamos įprastai.

Vykdant algoritmą prieš išrinkimo žingsnį populiacijoje esantys nedominuojami sprendiniai yra identifikuojami. Išskirti nedominuojami sprendiniai laikomi pirmuoju nedominuojamu frontų. Kiekvienam šio fronto nariui yra priskiriama viena ir ta pati didelė laikinoji tinkamumo reikšmė, tam, jog vieno fronto individai tūrėtų lygias dauginimosi galimybes. Tokiu būdu visi populiacijos individai yra sugrupuojami į frontus.

Siekiant išlaikyti tolygų pasiskirstymą sugrupuoti fronto nariai, kartu su jų laikinosiomis tinkamumo reikšmėmis, vėliau yra dalijami. Dalijimas yra atliekamas vykdant išrinkimo operaciją į skaičiavimus įtraukiant sumažintas tinkamumo reikšmes. Mažinamos tinkamumo reikšmės savo ruožtu yra apskaičiuojamos dalijant originalią individo tinkamumo reikšmę iš dydžio, proporcingo populiacijos narių skaičiui aplink tiriamą individą. Tinkamumo reikšmių dalijimas leidžia keletui optimalių sprendinių gyvuoti populiacijoje. Atlikus skaičiavimus su fronto nariais ir gavus jų sumažintas tinkamumo reikšmes jie yra trumpam pamirštami, tam jog būtų suformuotas sekantis

nedominuojamų sprendinių frontas tokiu pačiu būdu. Naujai suformuotame fronte individams yra priskiriamos padalintos tinkamumo reikšmės ne didesnės nei mažiausia tinkamumo reikšmė prieš tai buvusiame fronte.

Suformavus visus frontus vyksta dauginimasis populiacijoje vadovaujantis laikinosiomis individų tinkamumo reikšmėmis. Kadangi pirmojo fronto tinkamumo reikšmės yra didžiausios tai šis frontas turės daugiausiai kopijų. Tai leidžia populiacijai (sprendiniams) sėkmingai konverguoti į nedominuojamus regionus, o tinkamumo reikšmių dalijimas leidžia palaikyti tenkinamą tolygų populiacijos pasiskirstymą.

Atliekant minėtąjį dalijimą frontuose, dalijimo funkcijos reikšmė tarp dviejų individų skaičiuojama atsižvelgiant į fenotipinį atstumą tarp šių narių bei didžiausią leidžiamą fenotipinį atstumą σ_{share}. Viršijus σ_{share} individai nepatenka į tą pačią nišą. Nišos dydis, proporcingas narių skaičiui fronte, apskaičiuojamas sudedant visų individų fronte dalijimo funkcijos reikšmes. Galiausiai padalinta tinkamumo reikšmė individui apskaičiuojama dalijant laikinąją individo tinkamumo funkcijos reikšmę iš apskaičiuoto nišos dydžio.

2.3. NSGA-II

NSGA algoritmo autorius toliau tęsdamas darbą daugiakriterinio uždavinių optimizavimo srityje tobulino sąvajį NSGA algoritmą. Senoji NSGA versija metams bėgant buvo kritikuojama dėl kelių priežasčių:

1. Nedominuojamų sprendinių rikiavimo procedūra reikalauja didelių laiko bei skaičiavimo resursų.

2. Neišnaudotos žinios apie elitiškumą. Kadangi elišikumas leidžia išlaikyti gerus sprendinius populiacijoje, šis metodas optimizuoja algoritmo vykdymo darbą.

3. Dalijimo funkcija reikalauja išorinių parametrų, o jos skaičiavimas yra kompleksiškas.

Įprastai, kaip buvo aprašyta NSGA (2.2 poskyris) algoritme, grupuojant sprendinius į nedominuojamus lygius (frontus) sprendinys populiacijoje turi būti lyginamas su kitais sprendiniais pagal kiekvieną kriterijų. Palyginimus atlikus kiekvienam nariui ir sudarius pirmąją nedominuojamą aibę bendras algoritmo sudetingumas siekia $O(MN^2)$. Sekančiam frontui sudaryti atliekama ta pati procedūra tik jau be pirmojo fronto narių. Blogiausiu atveju, kai frontus sudaro po vieną sprendinį, algoritmo sudėtingumas pakyla iki $O(MN^3)$.

Nagrinėjamas algoritmas sprendžia daug laiko reikalaujančio rikiavimo į frontus uždavinį [DPA+02]. Verta pažymėti, jog rikiavimas atliekamas aibei, kuri yra tėvų bei poravimo metu gautų

vaikų aibių sąjunga. Rikiavimo į nedominuojamus frontus operacijai atlikti yra apskaičiuojamos dvi esybės kiekvienam individui: skaičius sprendinių, kurie dominuoja prieš individą n_p bei aibė sprendinių S_p, prieš kuriuos individas yra dominuojantis. Kiekvieno pirmojo fronto ($n_p = 0$) sprendinio aibės S_p elemento q skaičius n_q mažinamas vienetu. Jei sumažinta reikšmė tampa lygi 0, tai elementas perkeliamas į laikinąją aibę Q, kurią sudarys visi antrojo fronto nariai. Procedūra iteratyviai kartojama naujai sudarytai aibei Q, kuri suformuoja trečiąjį, ketvirtąjį ir sekančius frontus. Ši procedūra kartojama kol nėra sudaromi visi frontai. Kadangi šiame algoritme nėra perskaičiuojamas dominavimas kiekvieną kartą sudarius frontą, bendras algoritmo sudėtingumas siekia $O(MN^2)$.

Nors NSGA algoritmas pakankamai gerai išlaiko sprendinių tolygų pasiskirstymą fronte naudodamas dalijimo funkciją, šis būdas turi kelis trūkumus. Dalijimo funkcija stipriai priklauso nuo nurodyto dalijimo parametro σ_{share}, o tai sukelia nepatogumų vykdant algoritmą, kadangi žinios apie populiacijos narius yra reikalingos norint parinkti tinkamą reikšmę. Naujasis algoritmas NSGA-II keičia dalijimo funkciją nauju metodu, paremtu susigrūdimo atstumu (angl. *crowding distance*). Susigrūdimo atstumų skaičiavimas susideda iš dviejų dalių - tankio vertinimo metrikos bei susigrūdimo palyginimo operatoriaus. Tankio, aplink sprendinį, nustatymui skaičiuojamas vidutinis atstumas iki artimiausių dviejų spendinių, į abi puses nuo taško vienoje dimensijoje, pagal kiekvieną kriterijų. Taip iš rastųjų taškų suformuojamas kuboidas, kurio perimetras vadinamas susigrūdimo atstumu. Kai visi populiacijos nariai turi priskirtas susigrūdimo atstumų reikšmes individai gali būti palyginti tarpusavyje. Palyginimui yra naudojamas susigrūdimo palyginimo operatorius. Turint du populiacijos narių atributus, nedominavimo rangą (fronto indeksą) bei susigrūdimo atstumą, operatorius apibrėžia dalinę tvarką. Teikiama pirmenybė sprendiniams, kurie turi mažesnį rangą arba, jei rangas sutampa, sprendiniams, kurie yra retesniuose regionuose.

Verta paminėti, jog pristatyas susigrūdimo atstumu paremtas metodas, leidžiantis individams konverguoti į tolygiai paskirstytus sprendinius turintį Pareto frontą, tapo labai sėkingas daugiakriterinio optimizavimo srityje ir yra naudojamas daugybėje naujesnių algoritmų.

2.4. SPEA2

Antroji Pareto stiprio evoliucinio algoritmo (angl. *Strength Pareto evolutionary algorithm*) versija eliminavo pirmosios versijos trūkumus. Tiek originalaus tiek ir antrosios versijos algoritmų idėja yra ta pati - individų išrinkimas priklausomas nuo Pareto dominavimo [ZLT01]. Kaip jau buvo minėta, šis algoritmas priklauso Pareto aibe grįstų algoritmų aibei.

Algoritmas SPEA2 naudoja įprastą populiaciją ir išorinį archyvą individams palaikyti. Pra-

džioje algoritmas inicializuoja, atsitiktiniu būdu, pradinę populiaciją bei sukuria tuščią archyvą. Kiekvienos generacijos pradžioje, pradedant nuo pirmosios, tinkamumo reikšmės apskaičiuojamos individams įprastoje populiacijoje bei archyve. Individui priskiriama stiprio reikšmė, lygi skaičiui sprendinių, prieš kuriuos jis yra dominuojantis. Stiprio reikšmė yra naudojama apskaičiuoti individo grynąjai tinkamumo reikšmei, kadangi individo i grynoji tinkamumo reikšmė yra lygi sumai stiprio reikšmių visų individų j, kurie $j \prec i$. Tinkamumo reikšmių priskyrimas buvo patobulintas lyginant su SPEA, kadangi pirmoje versijoje dominavimas buvo tikrinamas tik archyvo kontekste. Taip pat stiprio reikšmė priklausė tik nuo to skaičiaus individų, prieš kuriuos buvo dominuojama.

Naujai aprašytas tinkamumo priskyrimas vis dar turi vieną trūkūmą - jei didžioji dalis individų nėra dominuojantys prieš vieni kitus, tai tinkamumo reikšmės yra panašios. Individai, turintys identišką tinkamumo reikšmę, išskiriami panaudojant tankio vertinimo (angl. *density estimation*) techniką. Tankio vertinimas atliekamas adaptavus k-tojo arčiausiojo kaimyno (angl. *k-th nearest neighbor*) metodą. Kiekvienam individui populiacijoje ir archyve yra suformuojamas atskiras, atstumų iki kitų individų, surikiuotų didėjančia tvarka, rinkinys. Surikiuotame sąraše k-tasis elementas žymi atstumą, kuris bus panaudotas skaičiuojant individo tankio įvertį. Elemento indeksas k yra bendras visiems rinkiniams, kuris yra siūlomas $k = \sqrt{N + \overline{N}}$ (kur N - populiacijos dydis, o $\overline{N}$ - archyvo dydis). Individų tikrąjai tinkamumo reikšmei gauti yra sudedama grynoji tinkamumo reikšmė su tankio įverčiu.

Atliekant individų išrinkimą, algoritme vadinamą aplinkos išrinkimu, nedominuojami individai iš populiacijos bei archyvo yra perkeliami į archyvą sekančiai generacijai. Jeigu nedominuojamų individų skaičius lygus archyvo dydžiui, tai jokių papildomų žingsnių nereikia vykdyti atliekant perkėlimą. Jeigu nedominuojamų individų skaičius neviršija archyvo dydžio, tai likusios vietos yra užpildomos geriausiais (dominuojamais) sprendiniais iš populiacijos ir prieš tai buvusio archyvo. Paskutiniuoju atveju, kai naujai suformuoto archyvo dydis viršiją apibrėžtą dydį, atliekama archyvo mažinimo (angl. *truncation*) procedūra. Ši procedūra iteratyviai šalina sprendinius esančius arčiausiai kitų sprendinių iki kol naujasis archyvas tenkina dydžio apribojimą. Jeigu yra keli sprendiniai su vienodu mažiausiu atstumu, iš jų šalinamas narys pagal antrą mažiausią atstumą.

Skirtingai nei SPEA algoritme, SPEA2 versijoje, vietoj archyvo, kuris nebuvo apribotas, narių klasterizavimo technikos, dydis yra apribotas. Taip pat yra atliekamas archyvo mažinimas, kai naujai suformuoto archyvo dydis viršija apribojimą. Naujasis metodas leidžia palaikyti pastovų (vienodą) skaičių geriausių sprendinių bei neprarasti ribinių narių, kaip kartais nutikdavo SPEA algoritmo vykdymo metu.

2.5. MOEA/D

Dekompozicija paremti (angl. *decomposition based*) algoritmai dalina vieną didelį daugiakriterinio optimizavimo uždavinį į dalinius skaliarinius optimizavimo uždavinius. Kelis dalinius uždavinius algoritmai sprendžia tuo pačiu metu. Kiekvienoje generacijoje populiaciją sudaro geriausi sprendiniai kiekvienam daliniam uždaviniui. Kaimynystės ryšiai tarp uždavinių yra apibrėžiami atstumu tarp uždavinių koeficientų vektorių. Taip pat algoritmai remiasi savybe, jog kaimyninių uždavinių sprendiniai turi būti panašūs [ZL07].

Algoritmui pristatyti naudojant Čebyševo dekomponavimo techniką. Vienas daugiakriterinio optimizavimo uždavinys dalinamas į N dalinių skaliarinių optimizavimo uždavinių. Kiekvienam daliniam uždaviniui spręsti yra sukuriamas sverto vektorius, o šie vektoriai $\lambda^1, ..., \lambda^N$, kur $\lambda^j = (\lambda^j_1, ..., \lambda^j_m)$, yra tolygiai pasiskirstę. Taip pat Čebyševo dekomponavimas remiasi atskaitos tašku m-matėje erdvėje $z^* = (z^*_1, ..., z^*_m)$. Turint visą reikalingą struktūrą Čebyševo metodas sprendinio tinkamumą, pagal j-tąjį dalinį uždavinį, įvertina apskaičiuodamas

$$g^{te}(x|\lambda^j, z^*) = \max_{1 \leq i \leq m} \{\lambda^j_i | f_i(x) - z^*_i|\}.$$

Šis rodiklis yra naudojams sprendinių tinkamumo palyginimui.

Algoritmo pradžioje kiekvienam sverto vektoriui yra suformuojamas jo T kaimynų sąrašas pagal Euklido atstumus. Vektoriaus λ^i kaimynystė apibrėžiama aibe artimiausių vektorių iš λ_1, ..., λ_N. Iš to išvedama, kad i-tojo dalinio uždavinio kaimynai yra uždaviniai, kurių svorių vektoriai yra iš λ^i kaimyninių vektorių sąrašo. Eigoje kuriami palikuonys kiekvienam daliniam uždaviniui spręsti, o poravimui tėvai renkami iš kaimynių sverto vektorių sąrašo. Tokiu būdu populiacija sudaryta iš geriausių sprendinių kiekvienam daliniam uždaviniui spręsti. Taip pat tik esami sprendiniai daliniams kaimyniniams uždaviniams yra išnaudojami.

Algoritmo autoriai pabrėžė jog algoritmas yra tik karkasas, todėl bet kuri dekomponavimo technika gali būti naudojama. Dėl to buvo pademonstruota kaip algoritmas veikia dekomponavus uždavinį svertinės sumos (angl. *weighted sum*) ir ribinės sankirtos (angl. *boundary intersection*) metodais.

2.6. mIBEA

IBEA buvo vienas pirmųjų indikatoriumi paremtų evoliucinių algoritmų (angl. *indicator-based evolutionary algorithms*). Pirmieji indikatoriai, kurie buvo naudojami šiame algoritme buvo ϵ dominavimas, žymimas $IBEA_\epsilon$, bei hyper tūris, žymimas $IBEA_{HD}$. Pristatomame algoritme nau-

dojamas yra hyper tūrio indikatorius.

Pagrindinė šio algoritmo ($IBEA_{HD}$) idėja yra dvejetainio hyper tūrio indikatoriaus (angl. *binary hypervolume indicator*) panaudojimas išrinkimo operatoriuje. Nurodžius atskaitos tašką dvejetainis hyper tūrio indikatorius priskiria realių skaičių aibės reikšmę dviem aibėms atsižvelgiant į atskaitos tašką. Indikatorius aibėms A ir B apibrėžtas kaip erdvė, kurioje sprendiniai yra dominuojami B aibės narių, bet nėra dominuojami aibės A narių.

Dar viena algoritmo savybė - tikslo reikšmių mastelio didinimas bei normavimas kiekvienoje generacijoje. Šiam tikslui pasiekti reikalaujama nurodyti mastelio keitimo faktorių ρ. Kiekvienam einamosios populiacijos nariui yra apskaičiuojama tikslo reikšmė ir nustatomos visos populiacijos mažiausia bei didžiausia tikslo reikšmės. Pagal didžiausią bei mažiausią reikšmes ir mastelio keitimo faktorių yra išvedamas naujasis viršutinis rėžis. Apatinis rėžis bei naujasis viršutinis rėžis yra panaudojami tikslo reikšmių normavimui.

Normuotos tikslo reikšmės yra įtraukiamos skaičiuojant indikatoriaus reikšmes bei maksimalią absoliučia indikatoriaus reikšmę. Kiekvienam sprendiniui populiacijoje yra apskaičiuojama tinkamumo reikšmė panaudojant indikatoriaus aibes bei indikatoriaus reikšmę, maksimalų įvertį ir nurodytą indikatoriaus reikšmę didinantį faktoriu κ. Jeigu populiacijos narių skaičius viršija nurodytą maksimalų dydį μ, tai sprendiniai populiacijoje yra iteratyviai šalinami. Šalinimo iteracija randa mažiausią tinkamumo reikšmę turintį individą ir jį pašalina, o tinkamumo reikšmės kiekvienam likusiam individui populiacijoje yra perskaičiuojamos. Poravimo operacijai algoritme pasirinktas įprastas dvejetainio turnyro (angl. *binary tournament*) metodas. Išrinkus du tėvus poruojant atliekamas kryžminimas pagimdantis palikuonį, kuris yra mutuojamas ir pridedamas prie bendros populiacijos.

Buvo pastebėta, jog sprendžiant kai kuriuos uždavinius klasikiniu IBEA algoritmu paieška, po kažkiek generacijų, užstrigdavo tam tikruose regionuose ir nebekonverguodavo į Pareto frontą. Modifikuotas mIBEA algoritmas neįtraukia dominuojamų sprendinių atliekant mastelio didinimo bei normavimo žingsnį. Siūlomas būdas dominuojamiems sprendiniams rasti yra susigrūdimo atstumo metodas (2.3 poskyris), tačiau pabrėžiama, jog gali būti naudojama bet kuri kita schema. Neįtraukiant dominuojamų sprendinių tikslo reikšmių mastelio didinimui, sprendiniai, esantys toli nuo geriausių nedominuojamų sprendinių, nebedaro įtakos.

2.7. Apibendrinimas ir išvados

Šiame skyriuje buvo apžvelgti populiariausi ir stabiliausi evoliuciniai algoritmai daugiakriteriniams uždaviniams spręsti. Apžvelgtos evoliucinių algoritmų, skirtų šiems uždaviniams spręsti,

ištakos, VEGA ir SPEA2 algoritmai. Taip pat buvo suprasta apie algoritmus iš kiekvienos kategorijos: Pareto aibe grįstų, dekompozicija grįstų bei indikatoriumi vadovaujamų. Iš kiekvienos kategorijos buvo išnagrinėta po vieną algoritmą: NSGA, MOEA/D ir mIBEA atitinkamai. Didelį pasisekimą atnešęs NSGA-II algoritmas, patobulinta NSGA versija, kuriame pristatoma viena efektyviausių ir dažniausiai naudojamų, susigrūdimo atstumo euristika, buvo apžvelgtas ir suprastas. Šis skyrius suteikė daugiau žinių apie esminius žingsnius ir euristikas bei evoliucinių algoritmų, daugiakriteriniams uždaviniams spręsti, efektyvumo pagrindimus.

3. Daugelio kriterijų optimizavimo algoritmai

Daugelio kriterijų optimizavimo uždaviniai yra atskiras daugiakriterinių optimizavimo uždavinių atvejis. Šio tipo uždaviniai susideda iš keturių ar daugiau kriterijų, tai yra keturių ar daugiau optimizavimo funkcijų. Optimizuoti daugiau nei tris kriterijus tenka dažnai, kadangi įvairūs architektūriniai bei dizaino sprendiniai optimizuoja daugybę atributų. Šie uždaviniai yra sudėtingiau išsprendžiami. Sudėtingumas kyla dėl daugiau resursų reikalaujančių operacijų, kadangi su padidėjusiu optimizavimo funkcijų skaičiumi atsiranda daugiau taškų, kuriuos reikia įvertinti aproksimuojant Pareto frontą [SYY18]. Didėjant kriterijų dimensijai tampa sudėtinga išlaikyti tinkamą populiacijos įvairovę, kadangi sprendiniai yra tankiai pasiskirstę didelioje tikslo funkcijų reikšmių dimensijoje [ABV18]. Tankus pasiskirstymas didelėje dimensijoje taip pat kelia nematytų uždavinių įvairovės valdymo strategijoms, taikomoms egzistuojančiuose daugiakriterinio optimizavimo algoritmuose [MR17]. Taip pat egzistuojantys bei patikimi daugiakriterinio optimizavimo algoritmai nėra pritaikyti daugelio kriterijų optimizavimo uždaviniams spręsti, todėl ne visų algoritmų efektyvumas išlieka toks pats sprendžiant šiuos uždavinius [LLT+15].

Kadangi daugelio kriterijų optimizavimo uždaviniai nagrinėjami ne taip seniai, tai nėra patikimų kiekvienoje situacijoje, reprezentacinių, algoritmų šiems uždaviniams spręsti. Dėl šios priežasties šiame skyriuje nagrinėjami algoritmai yra plataus spektro. Darant apžvalgą yra atsižvelgta į algoritmų patikimumą, naujumą bei skirtingų įdėjų bei konceptų realizavimą.

Šiame skyriuje visų pirma aprašomas, vieno stabiliausių bei labiausiai naudojamų NSGA-II algoritmų adaptacijos daugelio kriterijų optimizavimo uždavinių sprendimui NSGA-III modifikacija EliteNSGA-III (3.1 poskyris), įtraukianti elitinių (geriausiųjų visose iteracijose) narių archyvą. Iš indikatoriais grįstų algoritmų, pritaikytų daugelio kriterijų uždaviniams spręsti nagrinėjamas taip pat dažnai naudojamas HypE (3.2 poskyris) algoritmas. Viena naujausių euristikų, atskaitos taškai bei vektoriai, naudojama daugybėje algoritmų, dėl šios priežasties išrinkti du, krypties vektoriumi orientuotas algoritmas (3.4 poskyris) bei krypties vektoriumi orientuotas bendros evoliucijos algoritmas (3.3 poskyris). Taip pat, kadangi ši sritis yra plačiai nagrinėjama ir yra bandoma atrasti euristikų, sprendžiančių uždavinius kuo efektyviau, buvo išrinkta viena naujausių evoliucijos euristikų, dirbtinė bičių kolonija, kuri apžvelgiama 3.5 poskyryje.

3.1. NSGA-III ir EliteNSGA-III

Vieno populiariausių bei patikimiausių NSGA-II (2.3 poskyris) algoritmo autorius K. Deb siekė adaptuoti algoritmą daugelio kriterijų uždavinių sprendimui. Autorius atkreipė dėmesį, jog Pareto aibe grįstos schemos nebėra efektyvios ir neleidžia individams konverguoti greitai prie tinkamų

regionų, sprendžiant minėtos grupės uždavinius. Dėl šios priežasties dėmesys buvo nukreiptas į sprendinių įvairovę didinančias schemas.

Daugiakriterinių uždavinių sprendimui adaptuotame algoritme NSGA-III, lyginant su NSGA-II, vienas pagrindinių pasikeitimų buvo atskaitos taškų euristikos įvedimas [IRM+16]. Sudarius individų aibę naujai populiacijai, aibės elementų skaičius yra patikrinamas ar neviršija leistino populiacijos dydžio. Jeigu dydis viršija nustatytą didžiausią individų skaičių populiacijoje, prasčiausius rezultatus demonstruojantys individai yra išmetami taip, jog naujosios aibės narių skaičius neviršytų leistino. Individai yra lyginami pasinaudojant susigrūdimo atstumu NSGA-II algoritme. Tuo tarpu NSGA-III algoritme individų tinkamumas lyginamas remiantis nurodytais atskaitos taškais.

Naujasis, pristatomas, algoritmas EliteNSGA-III praplėtė NSGA-III algoritmą elitiškumo euristika [IRM+16]. Buvo nusitaikyta panaikinti galimybę geriausiems sprendiniams būti prarastiems evoliucijos metu. Taip pat buvo iškeltas tikslas pagerinti populiacijos įvairovę.

Pirmąjam tikslui pasiekti įvestas elitiškumo užtikrinimas - elito archyvas. Elito archyve yra laikomi tinkamiausi individai kiekvienam atskaitos taškui, kurie yra perskaičiuojami kiekvienoje generacijoje, taip įtraukiant ir naujuosius individus. Taigi elito archyvo narių skaičius yra lygus atskaitos taškų skaičiui. Taip pat, iš elito individą gali išmesti tik naujas, tinkamesnis, priklausomai nuo atskaitos taško, individas.

Naudojantis elito archyvu populiacijos įvairovei padidinti buvo modifikuota tėvų išrinkimo procedūra. Lyginant su NSGA-III procedūra, naujoji tėvus renka ne tik iš einamos populiacijos, tačiau ir iš archyvo. Kadangi ankstyvosiose generacijose keli individai gali būti priskirti tiems patiems atskaitos taškams, išrinkus du tėvus palikuonis bus labai panašus į savo tėvus taip sumažindamas populiacijos įvairovę. Naujoji procedūra minimizuoja tikimybę pasirinkti tėvus, susietus su tais pačiais atskaitos vektoriais, kadangi archyve yra bent vienas elitinis narys reprezentuojantis kiekvieną atskaitos vektorių.

Naujai pristatytas EliteNSGA-III algoritmas, įvedęs elito archyvą bei leisdamas išrinkti tėvus iš šio archyvo, neleidžia geriausiems sprendiniams galimai būti išmestiems evoliucijos metu bei padidina populiacijos įvairovę.

3.2. HypE

Daugelio kriterijų uždavinių sprendimui galima pasinaudoti gautųjų sprendinių kokybės matu - hyper tūriu. Algoritme HypE yra pristatoma procedūra, leidžianti apskaičiuoti hyper tūrį tiksliai. Deja, šio indikatoriaus tikslaus skaičiavimo didžiausias trūkumas yra didžiulių skaičiavimo kaštų reikalaujančios procedūros, klasifikuojančios apskaičiavimo procedūrą į eksponentinį $O(N^n)$ al-

goritmo sudėtingumą. Dėl šios priežasties, algoritmas taip pat pristato greitos hyper tūrio įverčio paieškos būdą aproksimuojant reikšmes atliekant Monte Karlo simuliacijas [BZ11]. Pristačius šias dvi skaičiavimo procedūras pabrėžiama, jog svarbiausios yra ne tikslios hyper tūrio reikšmės, bet sprendinių reitingavimas pagal įverčius.

Algoritmas vykdomas įprastai evoliuciniams algoritmams - individai yra suporuojami, o populiacijos perviršis iteratyviai atmetamas. HypE algoritmo specifika yra individų tinkamumo palyginimas tiek porinant tiek atmetant dalį populiacijos. Individai yra lyginami išdalinant tūrį į nesikertančias aibes bei skaičiuojant hyper tūrius šioms aibėms. Kadangi tikslus hyper tūrio skaičiavimas trunka ilgai, tikslios hyper tūrio reikšmės skaičiavimas yra naudojamas, kai kriterijų skaičius nerviršija 4. Taigi, aproksimuojanti hyper tūrio įverčio skaičiavimo schema naudojama daugelio kriterijų uždaviniams.

3.3. Krypties vektoriumi orientuotas bendros evoliucijos algoritmas

Tradiciniai evoliuciniai algoritmai veikia naudodamiesi Darvino natūralios atrankos teorija, tačiau neatsižvelgia į bendradarbiavimo tarp skirtingų rūšių galimybę. Siekiant išlaikyti populiacijos įvairovę kai kurie algoritmai veikia vystydami kelias populiacijas kur populiacijos vystosi nepriklausomai ir apsikeičia informacija tarpusavyje tik karts nuo karto. Bendrąją evoliuciją (angl. *co-evolution*) galima suprasti kaip Darvino teorijos išpildymą, kur evoliucijos proceso metu įvykę pokyčiai vienoje rūšyje padaro įtaką kitai rūšiai. Bendros evoliucijos algoritmas pabrėžia bendravimą bei dalijimąsį resursais tarp skirtingų populiacijų [JHS+13]. Pagrindinė šio algoritmo idėja - individo tinkamumo funkcijos reikšmė pirklauso ne tik nuo pačio individo bet ir nuo kitų populiacijų.

Krypties vektorius jau nuo seno yra naudojamas analitinėje geometrijoje, o šiame algoritme šie vektoriai padeda apibrėžti konvergavimą į Pareto frontą. Sprendžiant m kriterijų uždavinius yra apibrėžiami m krypties vektoriai $\lambda = (\lambda_1, \lambda_2, ..., \lambda_m)$ bei nustatomi m atskaitos taškai $R = (r_1, r_2, ..., r_m)$. Dydis θ_i yra kampas tarp sprendinio vektoriaus bei i-tosios tikslo funkcijos ašies, reliatyviai skaičiuojamas nuo atskaitos taško R.

Pradedant vykdyti algoritmą m kriterijų uždaviniui spręsti yra atliekamas krypties vektorių generacinis operatorius. Generacinis operatorius inicializuoja m krypties vektorius $\lambda^{(m)}$, kurie yra tolygiai pasiskirstę hyper sferos ketvirtadalyje. Jei uždavinys yra dviejų kriterijų, tai krypties vektoriai yra inicializuojami apskritimo ketvirtadalyje. Pradiniai krypties vektoriai apibrėžia regionus, aplink kuriuos yra susitelkę populiacijos nariai, kur vektorius λ_i yra laikomas i-tosios dalinės populiacijos centru. Kiekvienas individas yra priskiriamas prie dalinės populiacijos pagal inividualios

klasifikacijos operatorių, kuris nusako, pagal pradinius krypties vektorius bei individo krypties vektorių, kuriai dalinei populiacijai jis priklauso. Turint grupes yra skaičiuojamos tinkamumo reikšmės kiekvienai dalinei populiacijai panaudojant autorių siūlomą nuobaudos pagal ribos susikirtimą (angl. *penalty-based boundary intersection*) funkciją. Ši funkcija priskiria mažesnias tinkamumo reikšmes priklausomai nuo to, kaip toli sprendinys yra nuo pradinio krypties vektoriaus.

Bendros evoliucijos algoritmas pabrėžia individų varžymąsi dalinėje populiacijoje, todėl išrinkimui yra naudojamas varžymosi operatorius. Atsidūrus situacijoje, kai individai nėra tolygiai pasiskirstę dalinėse populiacijose, individai, esantys retesniuose regionuose, yra išlaikomi, o individai, esantys tankesniuose regionuose, yra eliminuojami pagal šį operatorių. Taip pat šis algoritmas pabrėžia dalinių populiacijų bendradarbiavimą. Šiame algoritme dalinių populiacijų kaimynais yra laikomos dalinės populiacijos, kurių regionai yra greta. Jei dvi dalinės populiacijos yra greta viena kitos tai šių populiacijų krypties vektoriai nukreipti panašia linkme, o tai reiškia, jog taškai, konverguojantys krypties vektoriais, susiduria su panašiais uždaviniais, todėl šios dalinės populiacijos kuria glaudų bendradarbiavimą. Dviem dalinėms pupuliacijoms, nesančioms kaimynėmis, įmanoma bendradarbiauti, jei jos turi bendrų tarpinių populiacijų. Komunikuojant individams iš skirtingų regionų (dalinių populiacijų) atliekamas individų kryžminimas. Kryžminimo operatoriui įgyvendinti yra pasitelktas modifikuotas simuliuojamas dvejetainis kryžminimas (1.5 poskyris), kuris autorių yra pavadintas tiesios linijos kryžminimu (angl. *straight line crossover*). Šis kryžminimas randa geriausius sprendinius tiesėje, jungiančioje du kaimyninių dalinių populiacijų individus. Atlikus išrinkimą bei kryžminimą visoje populiacijoje yra atliekamas euristinė išorinio archyvo paieška (angl. *heuristic search of the external archive*). Prasčiausi sprendiniai iš archyvo yra išmetami pagal NSGA-II pristatytą (2.3 poskyris) susigrūdimo atstumą. Šis archyvas išlaiko geriausius sprendinius iš generacijos į generaciją užtikrindamas sprendinių elitiškumą.

3.4. Atskaitos vektoriumi orientuotas algoritmas

Atskaitos vektoriumi orientuoto evoliucinio algoritmo (angl. *reference vector guided evolutionary algorithm*) charakterizuojantys bruožai yra regionų apibrėžimas panaudojant centro vektorių bei spindulį. Taip pat šis algoritmas pristato naują, kampu tarp vektorių orientuotą išrinkimo operatorių [CJO+16].

Algoritmo pradžioje yra sugeneruojama pradinė populiacija atsitiktiniu būdu ir pradinis, tolygiai pasiskirsčiusių vienetinių atskaitos vektorių, rinkinys pagal vektorių generacijos operatorių. Operatorius, generuojantis vektorius naudoja kanoninį simplekso grotelių (angl. *canonical simplex-lattice*) dizaino metodą, kuris iš pradžių sugeneruoja tolygiai pasiskirsčiusius taškus vienetinėje

hyper plokštumoje. Turint taškus atlikus transformaciją gaunami vienetiniai vektoriai hyper sferoje. Verta paminėti, jog atskaitos vektoriai gali būti nurodyti išreikštinai ir generacijos žingsnis gali būti praleistas. Tokiu būdu algoritmas tampa sprendimų prioritetais grįstas.

Individų mutacijai bei kryžminimui, atvedant naujus individus į populiaciją, algoritmas naudoja vienus populiariausių metodų - simuliuojamo dvejetainio kryžminimo (1.5 poskyris) bei polinominės mutacijos (angl. *polynomial mutation*) operatorius. Siekiant išlaikyti elitiškumą taikomas metodas, prisatytas NSGA-II algoritme (2.3 poskyris), kur palikuonių populiacija yra sujungiama su tėvų populiacija, tad išrinkimo operatorius taikomas bendrąjai populiacijai.

Vykdant išrinkimą tikslo funkcijos reikšmių erdvė yra suskirstoma į dalines erdves pagal atskaitos vektorius, o išrinkimas, naujai pristatytu operatoriumi, vykdomas nepriklausomai kiekvienoje dalinėje erdvėje. Naujai pristatytas operatorius susideda iš trijų žingsnių.

Pirmasis žingsnis - tikslo funkcijos reikšmės perkėlimas (angl. *objective value translation*). Algoritme naudojami atskaitos vektorių pradiniai taškai yra koordinačių sistemos pradžios taškas, todėl kiekvieno individo tikslo funkcijos reikšmės taškas yra pastūmiamas koordinačių pradžios taško linkme.

Antrasis žingsnis - populiacijos dalijimas (angl. *population partition*). Kaip ir prieš tai apžvelgtame bendros evoliucijos algoritme (3.3 poskyris), taip ir šiame algoritme populiacija yra dalijama į dalines populiacijas lyginant sprendinių vektorius su atskaitos vektoriais atsižvelgiant į kampus tarp jų.

Abu algoritmai taip pat atlieka trečiąjį nagrinėjamo algoritmo žingsnį - priskiria tinkamumo reikšmes lygindami vektorius. Atskaitos vektoriumi orientuotas algoritmas, skirtingai nuo bendros evoliucijos algoritmo, atlieka vektorių lyginimą panaudojant kampo tarp vektorių dydį bei priskiria nuobaudą pagal kampo dydį (angl. *angle-penalized distance*). Kampo dydžio nuobaudos metodas šiuo atveju yra palankesnis, kadangi skaičiuojant panašumą tarp sprendinio ir taško atskaitos vektoriuje atstumas Euklido erdvėje gali būti didelis, tačiau kampas tarp vektorių išlieka konstantinis. Taip pat kampai gali būti lengviau normuoti (pvz. į intervalą [0, 1]).

Paskutinis viso algoritmo žingsnis yra atskaitos vektoriaus adaptacija. Ruošiant atskaitos vektorius sekančiai generacijai jie yra normuojami (adaptuojami) prisitaikant prie tinkamumo funkcijos reikšmių rėžių. Algoritmo autoriai teigia, jog tinkamumo reikšmių normavimas nėra tinkamas šiame algoritme dėl to, kad normuotos reikšmės iškraipo tikrąsias reikšmes, o tai reiškia, jog tai gali padaryti įtaką sekančioms generacijoms. Taip pat reikšmių rėžiai skiriasi kiekvienoje generacijoje, tad tai veda prie skaičiavimo kompleksiškumo. Normuoti vektoriai leidžia gauti tolygiai pasiskirsčiusius sprendinius net ir nenormavus sprendinių tinkamumo reikšmių.

3.5. Dirbtinė bičių kolonija

Šiame poskyryje pristatomas dirbtinės bičių kolonijos (angl. *artificial bee colony*) algoritmas skiriasi nuo prieš tai pristatytų algoritmų tuo, jog glaudžiai naudoja vienos konkrečios biologinės rūšies elgseną bei hierarchiją. Algoritmas priklauso spiečių intelektu grįstų (angl. *swarm intelligence-based*) algoritmų kategorijai ir yra įkvėptas bičių spiečiaus maitinimosi elgsena [LLY+17].

Šiame algoritme bendradarbiauja trijų tipų bitės - įdarbintos (angl. *employed*), stebinčios (angl. *onlooker*) bei žvalgai (angl. *scout*). Įdarbintos bitės tiria maisto šaltinius (angl. *food sources*), sprendinius, daugiakriterinio optimizavimo kontekste. Informacija apie šaltinius yra perduodama bitėms stebėtojoms, kurios keičia savo pozicija priklausomai nuo šaltinio tinkamumo, gauto iš įdarbintų bičių. Taip pat stebinčios bitės stengiasi pagerinti maisto šaltinius. Besižvalgančios bitės peržvelgia naudojamus maisto šaltinius, kurie nebuvo pagerinti kurį laika, ir perkuria juos.

Pradedant algoritmą yra inicializuojami $FoodNumber$ maisto šaltiniai, kur maisto šaltinio įvertis, atsitikiniu būdu, patenka į rėžį [lb_d .. ub_d]. Reikšmės lb_d ir ub_d yra apatinis ir viršutinis, atitinkamai, galimų, maisto šaltinio d-tosios dimensijos, reikšmių rėžiai. Taip pat kiekvienam maisto šaltiniui algoritmo pradžioje priskiriama $trial$ reikšmė 0, žyminti kiek kartų šaltinis buvo bandytas pagerinti. Bandymų skaičiui $trial$ perkopus iš anksto nustatytą ribą $limit$ maisto šaltinis yra apleidžiamas, o įdarbintoji bitė, atsakinga už šį šaltinį tampa žvalgu ir pradeda naujų maisto šaltinių paiešką.

Kiekvienas maisto šaltinis turi priskirtą įdarbintą bitę, tad populiacija turi $FoodNumber$ įdarbintų bičių. Įdarbinta bitė, tirianti maisto šaltinio regioną pasirenka atsitiktinį kaimyninį maisto šaltinį. Evoliucinio algoritmo kontekste yra vykdoma mutacija. Algoritmas siūlo naudoti sąlyginę mutaciją: jei atsitiktinai sugeneruotas realus skaičius yra mažesnis nei iš anksto nustatytas slenkstis Pm, tai paprasta dviejų šaltinių mutacija yra atliekama. Priešingu atveju atliekama diferencialinė evoliucijos mutacija (angl. *differential evolution mutation*), kuri mutuoja tris atsitiktinai pasirinktus maisto šaltinius, nesutampančius su tiriamu šaltiniu. Jeigu naujai gautas maisto šaltinis yra labiau tinkamas, tai jis keičia tiriamąjį šaltinį. Priešingu atveju tiriamojo šaltinio bandymų pagerinti skaičius $trial$ padidinamas vienetu. Gavus naująją maisto šaltinio poziciją jos tinkamumo reikšmė, kuri yra lygi $I_{\epsilon+}$ indikatoriaus reikšmei, yra apskaičiuojama ir priskiriama.

Kai įdarbintos bitės baigia darbą jos grįžta į avilį ir pasidalina informacija su bitėmis stebėtojomis. Stebėtojos pasirenka geriausius šaltinius pagal tinkamumo reikšmes ir bando dar naudingiau juos išnaudoti. Stebėtojos peržiūri visus šaltinius, kurie yra išrikiuoti mažėjančia, pagal tinkamumą, tvarka ir apskaičiuoja šaltinių galios, nepriklausomos nuo mastelio, pasiskirstymą (angl.

scale-invariance power law distribution). Atrinkus šaltinius, kurių distribucijos reikšmės viršija nustatytą slenkstį, jiems priskiriamos tinkamumo reikšmės bei vykdoma mutacija. Minėtąsias operacijas stebėtojos vykdo tokiu pačiu principu kaip ir įdarbintos bitės.

Kai stebėtojos baigia darbą, bitės-žvalgai pradeda savo darbą. Šių bičių užduotis yra paprasta, peržiūrėti visus šaltinius ir nustatyti ar nebuvo viršyta optimizavimo bandymų riba. Kiekvieną šaltinį, kurį buvo bandyta optimizuoti per daug kartų, žvalgai pašalina ir sukuria, atsitiktinai kaip ir inicializavimo metu, naują maisto šaltinį.

Kiekvienos generacijos pabaigoje yra peržiūrimi maisto šaltiniai ir geriausieji dedami į išorinį archyvą. Išorinis archyvas implementuotas populiariausiu būdu, aprašytu NSGA-II algoritme (2.3 poskyris).

3.6. Apibendrinimas ir išvados

Šiame skyriuje buvo apžvelgti įvairaus spektro evoliuciniai algoritmai daugelio kriterijų uždaviniams spręsti. Labiausiai buvo koncentruojamasi į atskaitos taškų euristiką, kadangi ši euristika leidžia optimizuoti pagal abu daugiakriterinių optimizavimo uždavinių sprendimo aspektus - tolygų sprendinių pasiskirtstymą bei konvergavimą į Pareto frontą. Buvo apžvelgti trys atskaitos taškais grįsti algoritmai, tam, jog būtų suprasta euristika ir jos efektyvumas sprendžiant daugelio kriterijų optimizavimo uždavinius. Šiame skyriuje buvo išnagrinėtas nedidelis NSGA-III algoritmo pakeitimas, EliteNSGA-III, kuris keičia palikuonių populiacijos konstrukcijos žingsnį, tačiau didžioji dalis algoritmo, aplinkos išrinkimas, nėra keičiamas. NSGA-III algoritmas puikiai išnaudoja atstumus iki atskaitos vektorių, jog būtų optimizuotas aproksimuotas Pareto frontas. Žvelgiant į NSGA-III pasisekimą ir tai, jog jis yra paremtas atskaitos taškų euristika, buvo nuspręsta, jog šis algoritmas yra labiausiai tinkamas tyrimams.

4. Analizė ir eksperimentai

Atliekant eksperimentus bei ieškant patobulinimo taškų nebuvo norėta kurti algoritmo nuo pat pradžių, todėl buvo pasirinkta vieno algoritmo bazė ir atliekami eksperimentai, siekiant atrasti vietas, kurias galima būtų patobulinti. Algoritmas buvo renkamas iš atskaitos vektoriais grįstų algoritmų šeimos, kadangi tai yra pakankamai nauja sritis bei atskaitos vektoriai atrodo puikus įrankis išlaikyti sprendinių įvairovę bei konvergavimą į Pareto frontą. Iš minėtosios šeimos algoritmų buvo pasirinktas NSGA-III [DJ14] algoritmas, kadangi šiuo metu jis yra laikomas vienu stabiliausiu algoritmų daugelio kriterijų optimizavimo uždaviniams spręsti, paremtų atskaitos vektoriais. Šio algoritmo populiarumą patvirtina ir tai, jog tai yra dažniausiai cituojamas darbas, o naujai pristatomų algoritmų autoriai įprastai renkasi rezultatus lyginti būtent su NSGA-III algoritmo rezultatais. Taip pat šis algoritmas yra palankus ekperimentams, kadangi turi daug įvairių žingsnių, atveriančių galimybes modifikacijoms: nuo pasiekimus skaliarizuojančios funkcijos iki individų asociacijos su atskaitos vektoriais.

Deja, tačiau oficiali NSGA-III algoritmo implementacija nėra pateikta, todėl eksperimentams atlikti buvo naudojama rasta realizacija [TC16], kuri šiame darbe bus žymima kaip NSGA-III$^{\beta}$. Algoritmo realizacija suprogramuota C++ programavimo kalba.

4.1. NSGA-III algoritmo analizė

Naujajai, kombinuotai tėvų ir vaikų $R_t = P_t \cup Q_t$ (dydis $2N$), populiacijai sudaryti naudojami įprasti genetiniai mutacijos ir kryžminimo operatoriai. Kaip ir NSGA-II, taip ir NSGA-III algoritme naudojami polinominės mutacijos bei simuliuojamo dvejetainio kryžminimo operatoriai. Pastarasis operatorius turi dvi svarbias savybes, leidžiančias greičiau sprendiniams konverguoti į Pareto frontą:

- gautųjų vaikų $\{q_i, q_j\} \in Q_t$, sukryžminus du tėvus $\{p_k, p_l\} \in P_t$, vidurkis yra lygus tėvų vidurkiui $\frac{(q_i+q_j)}{2} = \frac{(p_k+p_l)}{2}$;
- tikimybė, sukryžminus tėvus, sukurti vaikus arčiau tėvų yra eksponentiškai didėjanti.

Parametrai minėtiesiems operatoriams pažymėti 1 lentelėje. Pasiskirstymo indeksas parodo kaip kaip plačiai simuliuojama dvejetainė išraiška (dvejetainių išraiškų pozicijomis) bus mutuojama arba kryžminama.

Aproksimuojamo Pareto fronto reikšmių tolygiam pasiskirstymui užtikrinti naudojami atskaitos taškai. Sisteminis metodas, pristatytas [DD98], atideda taškus normuotoje hiperplokštumoje

Parametrai	NSGA-III
Simuliuojamo dvejetainio kryžminimo tikimybė (p_c)	0.9
Polinominė mutacija	$1/n$
Kryžminimo pasiskirstymo indeksas (η_c)	30
Mutacijos pasiskirstymo indeksas (η_m)	20

1 lentelė. Kryžminimo ir mutacijos parametrų lentelė, kur n - kintamųjų skaičius

($M-1$ dimensijos plokštumoje). Pasirinkus kiekvieną ašį, M-matėje erdvėje, suskaidyti į p dalių, minėtasis metodas sugeneruoja H atskaitos taškų:

$$H = C^p_{M+p-1}, \tag{4.1}$$

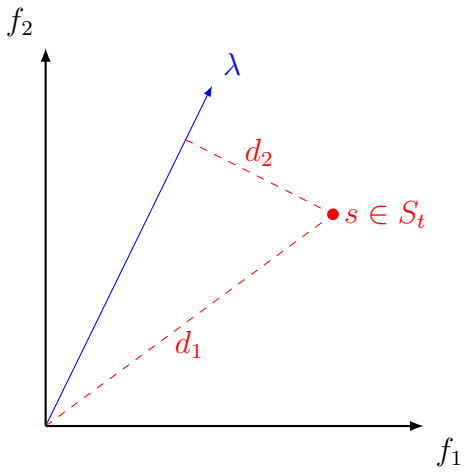
kur p - taškų skaičius vienoje ašyje (sluoksnio dydis).

Didėjant kriterijų skaičiui M atskaitos taškų skaičius stipriai didėja (pvz., kai $M = 10$, o $p = 5$ gautas atskaitos taškų skaičius yra $H = 2002$). Didelis atskaitos taškų skaičius stipriai padidina skaičiavimo sudėtingumą, todėl NSGA-III algoritmo pristatyme buvo pasiūlytas būdas padalinti atskaitos taškų generavimą į du sluoksnius, vidinį ir išorinį. Kiekvienam sluoksniui atskirai yra sugeneruojami atskaitos taškai ir sujungiami $H = C^3_{10+3-1} + C^2_{10+2-1}$. Atskaitos taškų generavimui parametrai pavaizduoti 2 lentelėje.

Kriterijų sk. (M)	Sluoksnių dydžiai (p)		Atskaitos taškų sk. (H)	NSGA-III populiacija (N)
	Išorinis	Vidinis		
5	6		210	212
8	3	2	156	156
10	3	2	275	276
15	2	1	135	136

2 lentelė. Populiacijos dydžio, atskaitos vektorių skaičiaus ir kriterijų skaičiaus priklausomybės

Atskaitos vektoriais paremtuose algoritmuose įprastai skaičiuojamas atstumas iki atskaitos vektoriaus. Taip pat neretai skaičiuojamas atstumas iki koordinačių pradžios taško, tačiau šis atstumas gali būti skaičiuojamas tiek nuo individo, tiek ir nuo susikirtimo su atskaitos vektoriumi taško. Šiame darbe atstumas iki koordinačių pradžios taško skaičiuojamas nuo sprendinio taško, o abu paminėti atstumai vaizduojami 1 paveiksle. Nors ir originaliai pristatytame NSGA-III algoritme d_1 atstumas nėra naudojamas, tačiau šiame darbe vėliau pristatomoje procedūroje šis atstumas atlieka svarbų vaidmenį.



1 pav. Individui priskirti atstumų atributai

Atstumai d_1 ir d_2 apskaičiuojami taip:

$$d_1(x) = \sqrt{\sum_{i \in M} f_i(x)^2}, \tag{4.2}$$

$$d_2(x, w_i) = \left\| f_i(x) - w_i \frac{w_i \cdot f_i(x)}{w_i \cdot w_i} \right\|, \tag{4.3}$$

kur $a \cdot b$ yra skaliarinė vektorių sandauga ($a^T b$). Taip pat verta atkreipti dėmesį, jog formulėse funkcijų reikšmės naudojamos nenormuotos. Akivaizdu, jog mažesnė d_1 reikšmė identifikuoja greitesnį konvergavimą į Pareto frontą, o mažesnė d_2 reikšmė identifikuoja tolygesnį sprendinių pasiskirstymą.

Viena svarbiausių operacijų atliekant individų išrinkimą yra optimizavimo funkcijų reikšmių normavimas, tam, jog individų palyginimas būtų kuo objektyvesnis bei tikslesnis. Iš visos einamosios iteracijos t populiacijos S_t randama mažiausia kriterijaus (pagal kiekvieną ašį) reikšmė z_i^{min} ($i \in 1...M$), o iš visų mažiausių kriterijų reikšmių sudaromas idealusis taškas $\overline{z} = (z_1^{min}, z_2^{min}, ..., z_M^{min})$. Šis taškas normavime naudojamas kaip apatinis rėžis. Viršutiniam rėžiui rasti iš visų populiacijos narių sukonstruojama M dimensijos hiperplokštuma, kuriai formuojant apskaičiuojami tolimiausi taškai pagal kiekvieną ašį. Tolimiausi taškai yra tokie, kurie maksimizuoja pasiekimus skaliarizuojančią funkciją (angl. *achievement scalarizing function*):

$$ASF(x, w) = \max_{i=1}^{M} \frac{f_i(x) - z_i^{min}}{max(w_i, 10^{-6})}, \tag{4.4}$$

kur w yra ašies krypties vektorius. Skaičiuojant ASF funkciją $w_i = 0$ reikšmės pakeičiamos į

mažą reikšmę $w = 10^{-6}$.

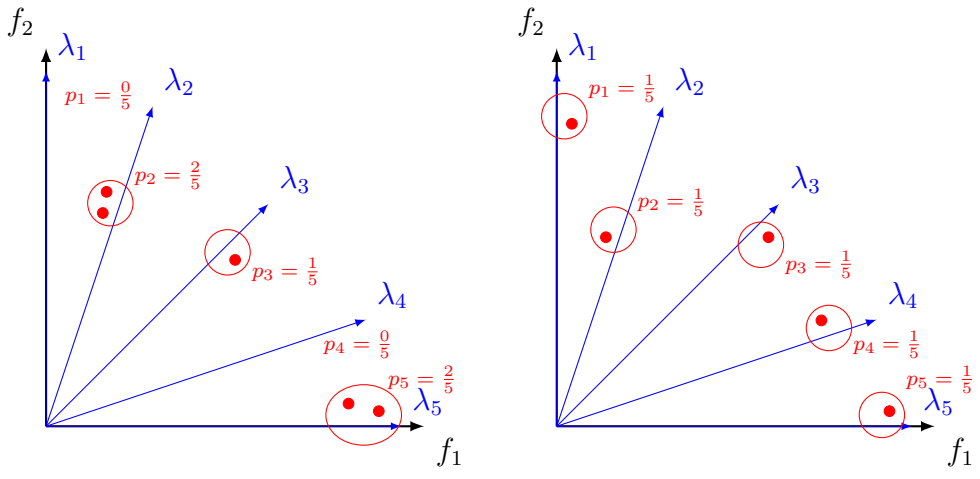
Naudojantis Gauso metodu, tiesinių lygčių sistemos sprendiniui rasti, apskaičiuojamos reikšmės a_i, kiekvienoje ašyje, kuriose hiperplokštuma susikerta su ašimis bei suformuojamas viršutinio rėžio taškas $\overline{a} = (a_1, a_2, ..., a_M)$. Turint tiek viršutinį, tiek apatinį rėžius, optimizavimo funkcijų reikšmės normuojamos naudojantis formule:

$$f_i^n(x) = \frac{f_i(x) - z_i^{min}}{a_i - z_i^{min}}. \tag{4.5}$$

Tokiu būdu normuotos reikšmės turi savybę visų reikšmių, kuriose normuota hiperplokštuma susikerta su ašimis, sumai būti lygiai 1 ($\sum_{i=1}^{M} f_i^n = 1$).

4.2. Individų pasiskirstymo analizė

Pirminė originalaus NSGA-III algoritmo analizė buvo pradėta nuo individų įvairovės išlaikymo tyrimo. Kadangi NSGA-III algoritmas priklauso atskaitos vektoriais paremtų algoritmų klasei, o atskaitos vektorių euristika yra ypač orientuota į populiacijos narių tolygų pasiskirstymo gerinimą, individų pasiskirstymas gali būti skaičiuojamas pagal tai, kiek narių yra priskirta kiekvienam atskaitos vektoriui. Tam, jog būtų galima įvertinti kaip gerai pasiskirstę yra individai, buvo pasinaudota Šanono entropijos įverčiu, kuriuo yra grindžiami entropija paremti algoritmai [ZL18] ir [ZDZ+18].



(a) (b)

2 pav. Atskaitos vektoriams priskirti individai ir tikimybių skirstinys entropijos skaičiavimui. Maksimali entropijos reikšmė gaunama (b) dalyje.

Apibrėžus atsitiktinį dydį $X_k \in Z$ kaip atsitiktinai pasirinktą atskaitos tašką k, kuris yra gaunamas renkantis bet kurį su juo susietą sprendinį, įvykio tikimybe yra $p_k = \frac{N_k}{|Z|}$ (N_k - atskaitos

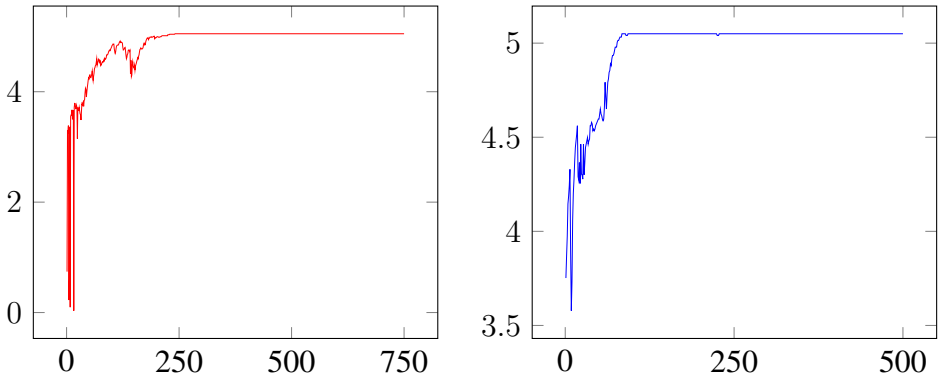
vektoriui k priskirtų sprendinių skaičius, o $|Z|$ - atskaitos vektorių skaičius). Taigi visi atskaitos taškai sudaro atsitiktinio dydžio tikimybinį skirstinį (2 paveikslas), kuriuo naudojantis apskaičiuojama entropija:

$$H(Z) = -\sum_{i=1}^{|Z|} p_i \ln(p_i) \tag{4.6}$$

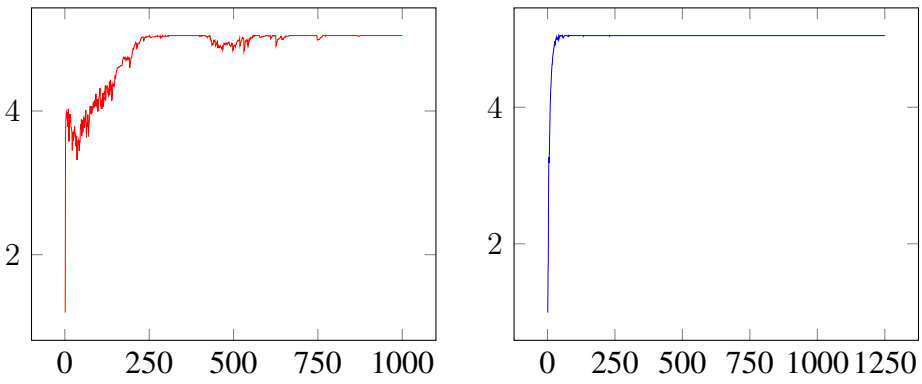
Maksimali entropija pasiekiama, kai lygios tikimybės egzituoja kiekvienam atsitiktiniam įvykiui ($p_1 = p_2 = ... = p_N$), o apskaičiuojama taip:

$$H_{max} = -\ln\left(\frac{1}{N}\right) \tag{4.7}$$

Reikia paminėti tai, jog norint pasiekti maksimalią entropiją sprendžiant daugiakriterinius optimizavimo uždavinius populiacijos dydis N turi sutapti su atskaitos taškų skaičiumi H (2 lentelė). Nekeičiant NSGA-III algoritmo maksimalią entropija galima pasiekti tik sprendžiant $M = 8$ kriterijų uždavinius.



(a) DTLZ1 (b) DTLZ2



(c) DTLZ3 (d) DTLZ4

3 pav. Entropijos įverčio pokytis kiekvienos iteracijos metu algoritmo NSGA-III$^{\beta}$, kur M = 8

Analizuojant NSGA-III$^{\beta}$ algoritmo sprendinių pasiskirstymą buvo sprendžiami DTLZ1-DTLZ4

testiniai uždaviniai su $M = 8$ kriterijais, skaičiuojant entropiją kiekvienoje iteracijoje. Gautieji rezultatai pavaizduoti 3 paveiksle. Pagal 4.7 formulę, maksimali galima entropija yra ≈ 5.0499. Algoritmas, sprendžiant visus keturis uždavinius, pasiekia pirmą kartą maksimalią entropiją per pirmas 30 % iteracijų, greičiausiai, per 45 iteracijas (3.6 %), sprendžiant DTLZ4 uždavinius. Vėliausiai pirmoji iteracija, pasiekianti maksimalią entropiją, yra pasiekiama sprendžiant DTLZ1 uždavinį, kur iteracijos indeksas yra 241 (32.1 %), tačiau tai vienintelis uždavinys, kur iteracija, kurioje pasiekiama maksimali entropija, ir iteracija, nuo kurios entropija lieka maksimali ir nebekinta, sutampa.

Anksčiausiai iteracija, kurioje entropija yra maksimali ir nuo kurios nebekinta, pasiekiama, ypač ankstyvoje 231 (18.4 %) iteracijoje, sprendžiant DTLZ4 uždavinį, o vėliausiai, 873 (87.3 %), sprendžiant DTLZ3 uždavinį. Taip pat DTLZ3 turi labiausiai svyruojantį entropijos įvertį maždaug vidurinėje visos evoliucijos dalyje, tačiau svyravimas yra tik apie 5 % (nuo ≈ 4.837 iki maksimalios entropijos).

Turint bendrą entropijos pokyčio vaizdą sprendžiant visus keturis uždavinius buvo nuspręsta, jog individų pasiskirstymas yra pakankamas ir NSGA-III algoritmas jį išlaiko puikiai, bent jau žvelgiant iš atskaitos vektorių ir sprendinių, priskirtų jiems.

4.3. Individų išrinkimo analizė

Patvirtinus tai, jog NSGA-III algoritmas gerai išlaiko individų tolygų pasiskirstymą, buvo analizuojamas antrasis daugiakriterinio optimizavimo sprendinių gerumo aspektas - individų konvergavimas į Pareto frontą.

Šiame ir tolimesniuose skyreliuose yra išsamiai pristatomas NSGA-III algoritmas: originalus algoritmas yra pristatomas nuo aplinkos išrinkimo aprašo (**1 algoritmas**), kuriame yra praleisti sprendinių apdorojimo, aprašyto tekste, žingsniai. Paskutinis aplinkos išrinkimo žingsnis, nišinimo procedūra, yra pristatomas **2 algoritmo** apraše, kuriame (mėlynu fonu) pažymėta šiame darbe pasiūlytos modifikacijos, elitinimo procedūros (**4 algoritmas**), inicijavimo vieta. Taip pat trumpai **3 algoritme** pristatoma, kokiu būdu yra sukuriamas pirminis elitinių narių archyvas.

NSGA-III algoritmo aplinkos išrinkimo, iteracijoje t, apibrėžimas pradedamas nuo bendros populiacijos R_t individų sugrupavimo ir surikiavimo į nedominuojamus frontus $\{F_1, F_2, ...\}$. Konstruojant naująją populiaciją S_t, pradedant $S_t = \emptyset$, nedominuojami frontai, pradedant F_1, yra imami vienas po kito ir prijungiami prie S_t individų. Nedominuojami frontai prijunginėjami iki tol, kol $|S_t|$ netampa lygus N arba kol pirmą kartą neperžengia šio slenksčio. Jeigu pridėjus praskutinį (pridėtą) frontą (pažymėkime jį F_l) $|S_t| = N$, tai sekanti iteracija vykdoma laikant jog tėvinės

populiacijos įvestis yra $P_{t+1} = S_t$. Jeigu $|S_t| > N$, tai iš populiacija sekančiai iteracijai užpildoma frontų nariais, neįskaitant paskutiniojo $P_{t+1} = \bigcup_{i=1}^{l-1} F_i$, o likusi dalis, $K = N - |P_{t+1}|$, užpildoma atliekant individų išrinkimą iš fronto F_l.

Pirmasis žingsnis normuoja išrinktų individų aibės S_t reikšmes vadovaujantis 4.1 poskyryje aprašytais, pasiekimų skaliarizacijos, hiperplokštumos susikirtimo su ašimis taškų paieškos bei normuotų reikšmių apskaičiavimo žingsniais. Normavus sprendinių reikšmes vykdomas individų asociavimas su atskaitos taškais. Asociavimo rezultatas yra dvi funkcijos, pirmoji - individui priskirto atskaitos taško radimo $\pi(s)$, kuri randa artimiausią atskaitos tašką. Antroji - priskirto artimiausio taško atstumo funkcija $d(s)$ (pagal 1 paveikslą tai yra d_2 atstumas, kur s yra normuotų reikšmių individas).

Kiekvienam atskaitos taškui $j \in Z^r$, naudojantis individui priskirto atskaitos taško funkcija $\pi(s)$, skaičiuojamas priskirtų populiacijos narių skaičius ρ_j. Paskutinė, nišinimo, procedūra išrenka reikiamą kiekį K individų iš fronto F_l (**2 algoritmas** be pažymėtos eilutės).

1 algoritmas: NSGA-III išrinkimo procedūros du paskutinieji žingsniai (generacija t)

Įvestis:
atskaitos taškų aibė Z^r,
nedominuojami frontai $(F_1, ..., F_l)$,
potencialių individų aibė $S_t = \bigcup_{i=1}^{l} F_i$,
artimiausio vektoriaus nuo taško funkcija π,
atstumo nuo atskaitos vektoriaus iki taško funkcija d,
jau įtrauktų individų aibė $P_{t+1} = \bigcup_{i=1}^{l-1} F_i$
Išvestis: nauja populiacija P_{t+1}

1 Apskaičiuojamas nišos (klasterio) dydis kiekvienam atskaitos taškui $j \in Z^r : \rho_j = \sum_{s \in S_t / F_l}((\pi(s) = j)?1 : 0)$
2 Renkami K nariai vienas po kito iš F_l ir konstruojama individų populiacija $P_{t+1} = \textit{Nišinimas}(K, \rho_j, \pi, d, Z^r, F_l, P_{t+1})$ (**2 algoritmas**)

Išrinkimas randa atskaitos taškus J_{min}, su kuriais susietas mažiausias individų skaičius, o iš išrinktų taškų grupės išrenkamas atsitiktinis j^* (svarbu, kai yra keli taškai su mažiausiu skaičiumi priskirtų individų). Iš paskutiniojo pridėto fronto išrenkami visi taškai I_{j^*}, susieti su pasirinktu atskaitos tašku j^*. Jeigu nėra nei vieno taško, priskirto atskaitos taškui, tuomet pasirinktas atskaitos taškas išmetamas iš atskaitos taškų aibės Z^r ir elitinimo procedūra vykdoma iš naujo. Svarbu pažymėti tai, jog išmetimas galioja tik šios procedūros ribose ir nedaro įtakos sekančioms evoliucijos iteracijoms. Jeigu išrinktų taškų aibė I_{j^*} nėra tuščia, tai tikrinama kiek yra taškų ρ_{j^*}, priskirtų pasirinktam atskaitos taškui. Jeigu dar nei vienas taškas nėra priskirtas j^*, tai išrenkamas taškas, artimiausias, naudojantis $d(s)$ funkcija, atskaitos vektoriui (d_2 atstumas). Kitu atveju, išrenkamas taškas iš I_{j^*} aibės atsitiktiniu būdu. Papildžius sekančios iteracijos populiaciją, skaičius priskirtų

individų ρ_{j^*} atskaitos taškui padidinamas vienetu bei ką tik pridėtas individas (pažymėkime jį s) išmetamas iš fronto F_l. Procedūra kartojama kol nėra pasirenkami K nariai.

2 algoritmas: NSGA-III nišinimo procedūra

Įvestis:
K, ρ_j, π, d, Z^r, F_l, P_{t+1} apibrėžti NSGA-III išrinkimo algoritme (1)
atskaitos taškui priskirto elitinio nario funkcija $E(j)$
Išvestis: nauja populiacija P_{t+1}

$k = 1$
while $k \leq K$ **do**
  $J_{min} = \{j : \text{argmin}_{j \in Z^r} \rho_j\}$
  $j^* = \text{random}(J_{min})$
  $I_{j^*} = \{s : \pi(s) = j^*, s \in F_l\}$
  if $I_{j^*} \neq \emptyset$ **then**
    if $\rho_{j^*} = 0$ **then**
      $P_{t+1} = P_{t+1} \cup \left(s : \text{argmin}_{s \in I_{j^*}} d(s)\right)$
    else
      $P_{t+1} = P_{t+1} \cup \text{random}(I_{j^*})$
    end
    Elitinimas(E, j^*, d, P_{t+1}) (4 algoritmas)
    $\rho_{j^*} = \rho_{j^*} + 1$
    $F_l = F_l / s$
    $k = k + 1$
  end
  $Z^r = Z^r / \{j^*\}$
end

4.4. Elitinių individų išlaikymas

NSGA-III algoritmo analizės metu buvo pastebėta, jog evoliucijos metu, atliekant kryžminimą ir mutaciją, galimai būdavo prarandami geriausi individai, kadangi jie būdavo mutuojami bei kryžminami. Šitokia ypatybė sulėtina individų konvergavimą į Pareto frontą. Geriausių individų išlaikymo euristika vadinama elitiškumu (1.2 poskyris), kuri yra ne nauja ir yra naudojama kituose algoritmuose, pvz. SPEA (2.4 poskyris) arba EliteNSGA-III (3.1 poskyris). Tačiau ši euristika dažniausiai, kaip ir minėtuose algoritmuose, yra realizuojama taip, jog geriausieji individai yra naudojami tik mutacijos ir kryžminimo metu, bet ne populiacijos išrinkimo metu. Minėtasis žingsnis realizuojamas įvedant atsitiktinumo faktorių, kuris mutuojant ir kryžminant kai kada vietoje populiacijos narių išrenka individą iš elitinių narių archyvo.

Elitinių narių archyvas atskaitos taškais grįstame algoritme gali būti išnaudotas labai pagrįstai, kadangi atsiranda aibė elementų (atskaitos taškų), su kuria galima sieti elitinius narius. Taip pat, 1 paveiksle pateikti atstumai iki atskaitos vektoriaus bei iki koordinačių pradžios taško puikiai

tinka bendram, konvergavimo ir įvairovės, įverčio sudarymui. Turint omenyje atskaitos vektoriais paremtų algoritmų paminėtas ypatybes ir pranašumus bei evoliucinių algoritmų prigimtį, naujasis algoritmo išrinkimo žingsnis turėjo pasižymėti šiomis sąvybėmis:

- Kiekvienam krypties vektoriui yra priskirtas tik vienas elitinis individas (arba nei vieno, jei tai yra algoritmo vykdymo pradžia);
- Lyginant natūralios atrankos individą su elitiniu yra lyginami tiek d_1, tiek d_2 atstumai ir yra įtraukiami į įvertį;
- Natūralios atrankos individams suteikiamas pranašumas prieš elitinius narius, kai yra bandoma priskirti naują elitinį narį krypties vektoriui.

Ieškant algoritmo patobulinimo buvo įvariai naudojami d_1 bei d_2 atstumai: jų palyginimai dvejetainiu būdu buvo vertinami nepriklausomai, taip pat buvo bandoma įvertinti proporcingai, priskiriant svarbumo koeficientą vienam arba kitam atstumui. Minėtieji atstumai taip pat buvo bandyti vertinti priklausomai nuo evoliucijos etapo. Lyginant buvo bandoma įvesti viršenybės koeficientus tiek natūralios atrankos nariams tiek elitiniams nariams. Visi minėtieji metodai pateikė įvairių rezultatų: dažnai pagal vieną konkretų uždavinį bei vieną konkretų kriterijų skaičių rezultatai pagerėdavo, tačiau pablogėdavo pagal kitus, todėl reikėjo atrasti stabiliausią parametrų rinkinį bei atstumų panaudojimą.

Po daugybės bandymų ir parametrų derinimo buvo prieita prie individų išrinkimo žingsnio, pasitelkiančio elitinių narių archyvą, pristatyto **4 algoritmo** aprašyme. Originalaus NSGA-III algoritmo nišinimo žingsnis papildomas elitinių narių archyvu bei funkcija, elitiniam nariui pagal atskaitos tašką gauti $E(j)$. Tam, jog vykdymo eigoje nenutiktų taip, jog atskaitos taškas neturėtų priskirto elitinio naro, uždavinio sprendimo pradžioje, prieš pirmąją iteraciją, yra sukuriamas atsitiktinių eilitinių narių archyvas pagal **3 algoritme** pateiktą aprašymą. Atsitiktiniai, be galo toli nutolę nuo koordinačių pradžios taško, nariai turi būti priskirti kiekvienam atskaitos taškui. Savaime suprantama, jog tokie nariai taip pat turės neapibrėžtai didelį atstumą iki atskaitos vektorių, sukonstruotų nuo koordinačių pradžios taško iki atskaitos taškų.

3 algoritmas: Pradinio elitinių individų archyvo sukūrimas

Įvestis: atskaitos taškų aibė Z^r
Išvestis: sukurta elitinių narių archyvo funkcija $E(j \in Z^r)$

```
for ∀j ∈ Z^r do
    E(j) = m_atsitiktinis : |m_atsitiktinis| = +∞
end
```

Taip pat minėtasis žingsnis papildomas elitinimo procedūra, kuri yra vykdoma **2 algoritme** pažymėtu momentu.

4 algoritmas: NSGA-III elitinimo procedūra

Įvestis:
$E(j)$, j^*, d, P_{t+1} apibrėžti NSGA-III nišinimo algoritme (2)
Išvestis: nauja populiacija P_{t+1}

$m^* = P_{t+1}$[**paskutinis**]
$m = Nenormuoti(m^*)$
$e = E(j^*)$
if $\neg(\|m\| < \|e\|$ ***or*** $d(m) < d(e))$ **then**
  $P_{t+1} = (P_{t+1}/m) \cup e$
end
$c_v =$ random$(\{1.1, 1.3\})$
if $\|m\| < \|e\| \cdot c_v$ ***and*** $d(m) < d(e) \cdot c_v$ **then**
  $E(j^*) = m$
end

Elitinimo procedūros idėja pakeisti ką tik įdėtą individą, gautą natūrialia atranka, elitiniu, jei jis galimai suteiks daugiau naudos ateityje. Prisiminus nišinimo procedūrą (**2 algoritmas**), paskutinis P_{t+1} populiacijos individas m yra artimiausias atskaitos vektoriui j^* narys, o atskaitos vektorius priklauso nišai, iš kurios yra paimta mažiausiai individų. Ypač svarbu paminėti tai, jog palyginimas su elitiniais individais yra vykdomas imant nenormuotas kriterijų reikšmes (pažymint individą m), kadangi išsaugoti elitiniai individai buvo kitose evoliucijos stadijose, kuriose individai buvo normuojami galimai kitoje skalėje. Išsaugoti elitiniai individai išsaugant buvo nenormuoti. Individų palyginimui imamas atskaitos taškui priskirtas elitinis narys $e = E(j^*)$.

Potencialiai keičiant individus įvedamas griežtas reikalavimas elitiniam nariui, kuris turi būti geresnis nei individas, kilęs iš natūralios evoliucijos tiek pagal d_1, tiek pagal d_2. Algoritme atstumas d_1 yra žymimas kaip vektoriaus ilgis $\|\cdot\|$, o d_2 yra funkcijos $d(\cdot)$ reikšmė. Dažnai tam, jog galima būtų santykinai palyginti d_1 ir d_2 atstumus vienu metu yra įvedama šių atstumų svertinės sumos išraiška $d_1 + \theta d_2$. Kadangi buvo norėta lyginti tiek d_1, tiek d_2 atstumus nepriklausomai, o ne jų svertinę skaliarinę išraišką, elitinimo procedūroje atstumai lyginami atskirai, o jų palyginimų įverčiai sujungiami binariniais operatoriais. Atnaujinant elitinį narį, priskirtą pasirinktam atskaitos taškui j^*, natūralios evoliucijos nariui suteikiamas pranašumas. Elitinis narys yra atnaujinamas jeigu natūralios atrankos individas yra geresnis nei elitinis individas pagal abu, d_1 ir d_2, atstumus su viršenybės koeficientu c_v, kuris atsitiktinai gali būti arba 1.1, arba 1.3. Šis koeficientas buvo išvestas empiriniu būdu atlikus ekperimentus.

4.5. Eksperimentai ir rezultatai

Atliekant eksperimentus ir vertinant pagerinimus bei naujų žingsnių įvedimus buvo sprendžiami pavyzdiniai (testiniai) uždaviniai, leidžiantys įvertinti algoritmų efektyvumą. Uždaviniai buvo pasirinkti iš DTLZ rinkinio (DTLZ1 - DTLZ4) [DTL+05] ir sprendžiami su 5, 8, 10 ir 15 kriterijų. Kintamųjų skaičius kiekviename uždavinyje yra lygus (M + k - 1), kur M yra kriterijų skaičius, k = 5 DTLZ1 uždaviniui, o DTLZ2 - DTLZ4 uždaviniams k = 10. DTLZ1 uždavinio Pareto fronto kriterijų reikšmės išsidėsčiusios tiesiškai $f_i \in [0, 0.5]$, kai tuo tarpu DTLZ2 - DTLZ4 Pareto frontas yra įgaubtas, o kriterijų reikšmės išsidėsčiusios $f_i \in [0, 1]$.

Uždavinys	Pareto fronto charakteristikos
DTLZ1	Tiesinis, daugiamodalinis
DTLZ2	Įgaubtas
DTLZ3	Įgaubtas, daugiamodalinis
DTLZ4	Įgaubtas, šališkas

3 lentelė. DTLZ rinkinio DTLZ1-DTLZ4 uždavinių Pareto fronto charakteristikos

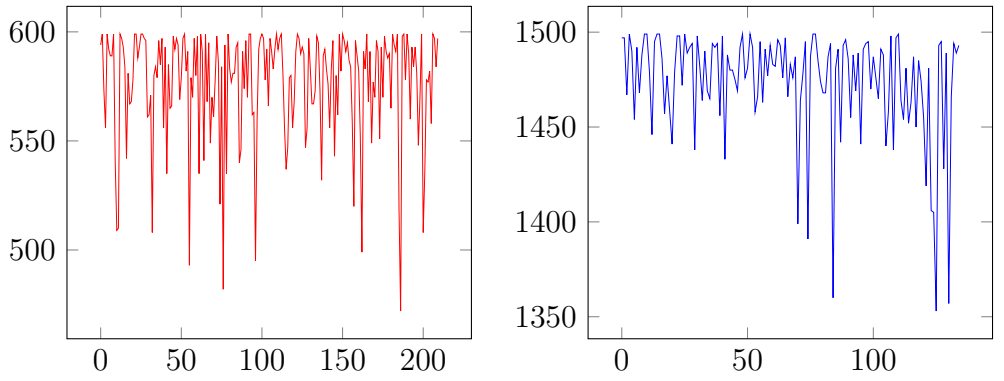
Kadangi tiek konvergavimas į tikrąjį Pareto frontą, tiek sprendinių tolygus pasiskirstymas yra vienodai svarbūs, tai gautiems sprendiniams įvertinti naudojamas apversto apibendrinto atstumo matas, kuris skaičiuojamas taip:

$$IGD(P^*, P) = \frac{\sum_{v \in P^*} d(v, P)}{|P^*|}, \tag{4.8}$$

kur P^* yra tolygiai pasiskirstę tikrojo Pareto fronto taškai, P yra gautieji, aproksimuoto Pareto fronto taškai, o $d(v, P)$ yra minimalus Euklidinis atstumas tarp taško v ir taškų aibės P.

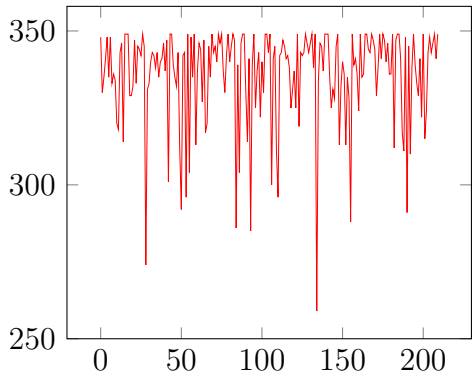
Įvedus elitinių narių archyvą yra svarbu suprasti kiek individų iš šio archyvo yra panaudojami evoliucijos metu, kaip panaudojami lyginant su natūralios atrankos individais bei kaip dažnai yra atnaujinamas elitinių narių sąrašas. Iteracijų indeksai, kai kiekvienam atskaitos taškui paskutinį kartą buvo pasirinktas individas, kilęs iš natūralios atrankos, pavaizduoti 4 paveiksle, o panaudotų ir atnaujintų elitinių narių skaičiai kiekvienoje evoliucinėje iteracijoje pavaizduoti 5 paveiksle.

Sprendžiant, tiek 5, tiek 15 kriterijų, visus keturis uždavinius DTLZ1-DTLZ4, pirmąsias 90 % natūralios atrankos individai yra naudojami, o paskutinių iteracijų, kai paskutinį kartą pasirenkamas individas iš natūralios atrankos individų, pasiskirstymas visiems atskaitos taškams, paskutinėse 10 % iteracijų, yra patenkinamai tolygus. Stebėtinai nei vienas atskaitos vektorius neturi ankstyvose iteracijose pasiektų elitinių narių, kurių nenurungia natūralios atrankos individai.

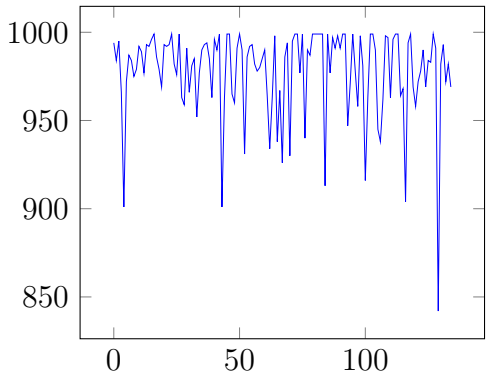


(a) DTLZ1

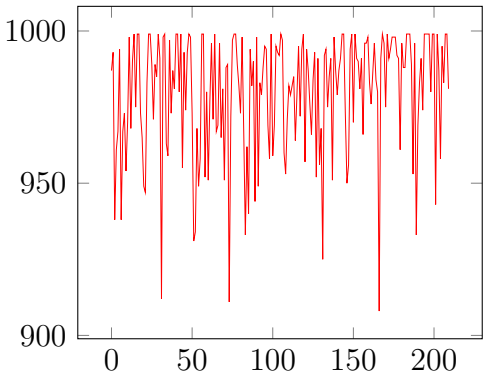
(b) DTLZ1



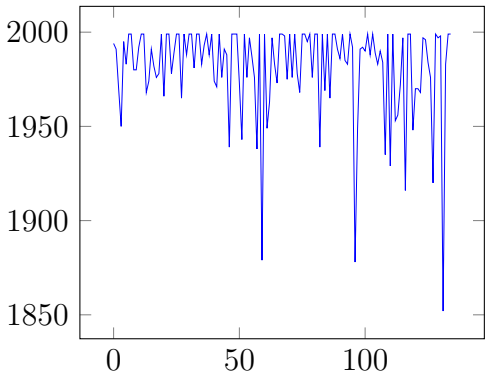
(c) DTLZ2



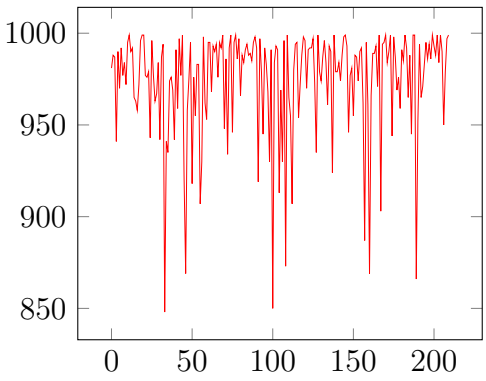
(d) DTLZ2



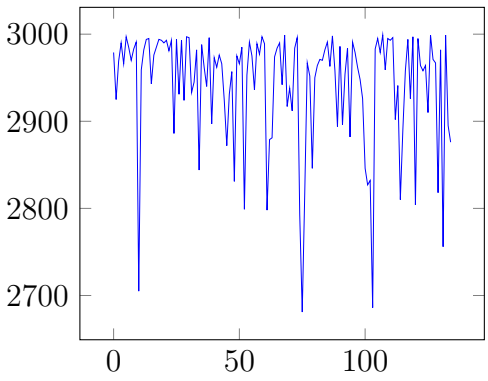
(e) DTLZ3



(f) DTLZ3



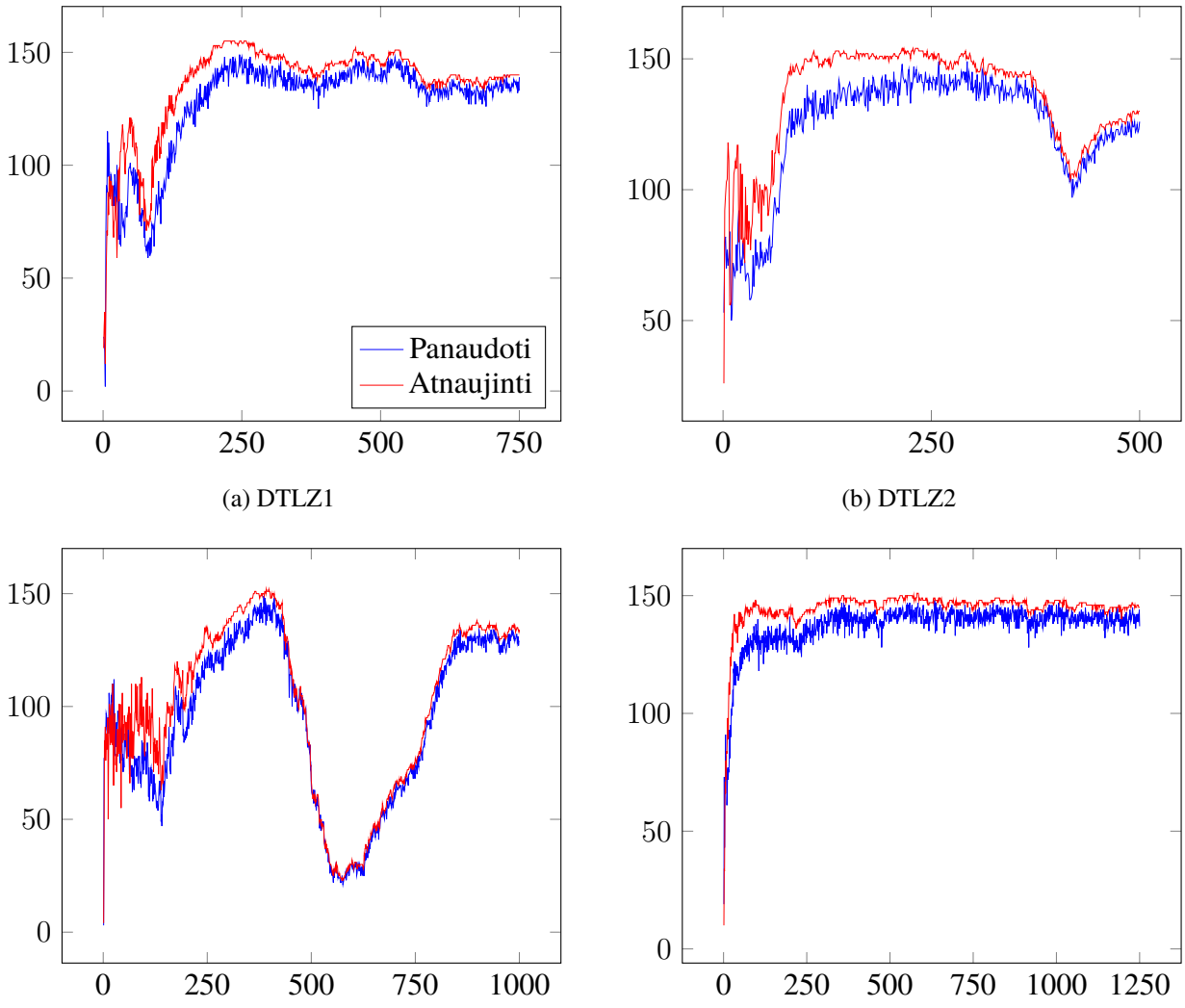
(g) DTLZ4



(h) DTLZ4

4 pav. Iteracijos indeksas, kai paskutinį kartą susietas su atskaitos tašku sprendinys yra išrenkamas natūraliu atrankos metodu, o ne iš elitinių narių. Vaizdai a, c, e, g kai M = 5, b, d, f, h - M = 15. Abscisėje vaizduojami atskaitos taškų indeksai, ordinatėje - iteracijų indeksai.

Žvelgiant į panaudotų bei atnaujintų narių skaičius visos evoliucijos metu, DTLZ1 bei DTLZ4 uždavinių sprendimų demonstruojami skaičių pokyčiai evoliucijos metu yra stabilūs. Narių skaičius pakyla į paskutiniąją virš 85 % zoną ir svyruoja joje. Tuo tarpu, sprendžiant DLTZ2 uždavinį, elitinių narių skaičius pabaigoje nukrenta, tačiau DTLZ3 turi dar ryškesnį nukritimą, kai tiek panaudotų tiek atnaujintų elitinių narių skaičius yra žemas. Žemas abiejų požymių skaičius parodo tai, jog didžioji dalis natūralios evoliucijos narių nesugebėjo nurungti elitinių narių, o žemas atnaujinimų skaičius rodo ir tai, kad nesugebėjo nurungti net jiems suteikus viršenybę (koeficientą c_v). Šitokia ypatybė yra netikėta, tačiau vėliau pristatyti aproksimuoto Pareto fronto įverčių rezultatai parodo, jog minėtoji ypatybė netrukdo pasiekti gerų rezultatų. Vis dėlto, pašalinus šią ypatybę rezultatai galėtų būtų pasiekiami dar geresni.



(a) DTLZ1

(b) DTLZ2

(c) DTLZ3

(d) DTLZ4

5 pav. Panaudotų ir atnaujintų elitinių individų skaičius kiekvienoje iteracijoje. Kiekvienos problemos kriterijų skaičius M = 8. Abscisėje vaizduojami iteracijų indeksai, ordinatėje - elitinių narių skaičius.

Išnagrinėjus elitinių narių naudojimą bei atnaujinimą rezultatai tenkina lūkesčius - padeda po-

puliacijai konverguoti greičiau į Pareto frontą naudojant elitinius narius, tačiau elitiniai nariai neužgožia natūralia evoliucija kilusių individų.

Sekančioje analizės stadijoje buvo nagrinėjama kaip greitai yra pasiekiama geriausia kriterijaus reikšmė ir kaip anksti ar vėlai pasiekta reikšmė daro įtaką galutiniam rezultatui. Geriausia kriterijaus reikšmė yra apibrėžiama kaip:

$$f_i^* = \max_{x \in P_{1..maxGen}} f_i(x), \qquad i \in 1, ..., M, \tag{4.9}$$

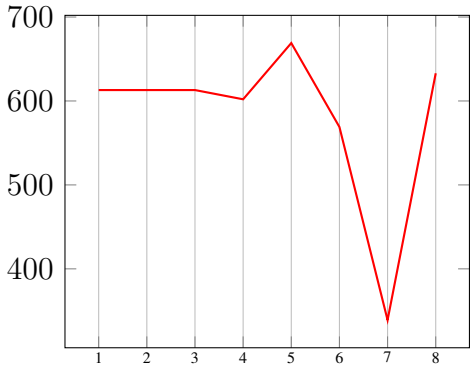
kur $P_{1..maxGen} = P_1 \cup P_2 \cup ... \cup P_{maxGen}$ yra visų iteracijų populiacijų sąjunga. Šiai analizei atlikti buvo pasirinkta M=8 ir M=15, o rezultatai pavaizduoti 6 paveiksle. Reikia pažymėti tai, jog gautieji iteracijų indeksai neidentifikuoja, jog toje iteracijoje buvo gauta uždavinio rezultato populiacijos geriausia kriterijaus reikšmė. Paveikslas parodo, kurioje iteracijoje buvo gauta geriausia kriterijaus reikšmė, kas reiškia, jog jeigu geriausia reikšmė buvo gauta ankstyvoje evoliucijos stadijoje, tai yra didesnė tikimybė nukrypti nuo tos reikšmės vykdant evoliuciją ir atliekant kryžminimo ir mutacijos operacijas.

Žvelgiant į 6 paveiksle pateiktus rezultatus įdomiausiai atrodo DTLZ4 uždavinio iteracijų, kuriose buvo gautos geriausios kriterijų reikšmės, indeksai. Tiek kai M=8, tiek kai M=15 yra po 3 kriterijus, kurių geriausios reikšmės buvo pasiektos ypač ankstyvose iteracijose. Taip pat yra dar keli kriterijai, kurių geriausios reikšmės buvo pasiektos per pirmas 20 % iteracijų.

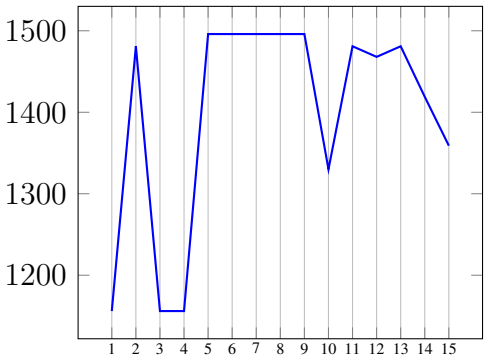
Atliekant eksperimentus ir lyginant algoritmus kiekvienas unikalus uždavinys buvo sprendžiamas 20 kartų (tiek pat, kiek ir originaliame NSGA-III algoritmo pristatyme bei realizacijoje NSGA-III$^{\beta}$) su skirtingomis atsitiktinėmis populiacijomis. Po atliktos paskutinės iteracijos, gautoji populiacija laikoma rezultatu ir yra skaičiuojamas apverstas apibendrintas atstumas. Iš gautųjų įverčių yra išsaugomos minimali, mediana ir maksimali reikšmės. Taip pat naujojo algoritmo apversto apibendrinto atstumo įverčiai yra lyginami su originalaus NSGA-III algoritmo pateiktais rezultatais bei algoritmo realizacijos rezultatais. Gautieji rezultatai yra vaizduojami 4 lentelėje NSGA-III* stulpelyje, kurioje jie yra lyginami su originaliu algoritmu, pažymėtu kaip NSGA-III, bei naudojamos realizacijos baze, kuri yra pažymėta kaip NSGA-III$^{\beta}$. Tam, jog būtų lengviau suprasti įverčių skirtumus yra įvestas paskutinis, rezultatų palyginimo, stulpelis. Stulpelyje yra pažymėti palyginimų rezultatai: pirmasis bei antrasis ženklai žymi palyginimų rezultatus lyginant su originaliu bei realizacijos algoritmais, atitinkamai. Visą apversto apibendrinto atstumo palyginamų įverčių aibę sudaro 48 rezultatai: 4 uždaviniai, kur kiekvienas sprendžiamas su keturiais skirtingais kriterijų dydžiais ir lyginami pagal mažiausią, medianą bei didžiausią reikšmes. Įverčių skirtumai yra vertinami atsižvelgiant į įprastą 5 % reikšmingo skirtumo koeficientą.

Pagal pagrindinį algoritmų efektyvumo įvertinimo kriterijų, apverstą apibendrintą atstumą, naujoji elitinimo procedūra dažnu atveju veikia efektyviau bei pasiekia geresnį aproksimuotą Pareto frontą. Lyginant su originalaus NSGA-III algoritmo pateiktais įverčiais net 69 % (33) buvo gauti geresni, o tik 27 % buvo prastesni. Rezultatus lyginant su naudojama algoritmo implementacija NSGA-III$^{\beta}$ procedūros efektyvumas ypač jaučiamas, kadangi net 73 % (35) įverčių gauti geresni, o prastesni tik 12 % (6).

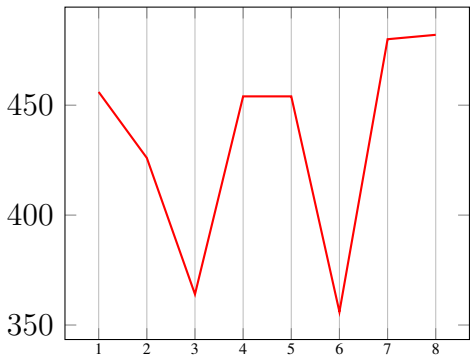
Žvelgiant į geriausiai patobulintus atvejus, lyginant su originalaus algoritmo įverčiais, tai net 4 kartus pavyko pasiekti virš 90 % pageriminą. Tokį žymų pageriminą pavyko pasiekti lyginant DTLZ4 M=8, M=10, M=15 blogiausius įverčius bei DTLZ4 M=15 medianas. Lyginant su naudojamos implementacijos baze, didžiausias pagerinimas buvo pasiektas sprendžiant M=10 kriterijų DTLZ3 uždavinį, kurio blogiausias įvertis pagerėjo 69.9 %.



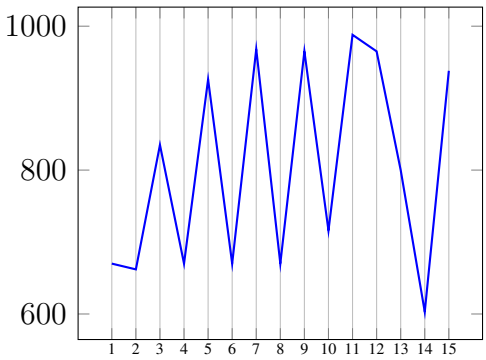
(a) DTLZ1



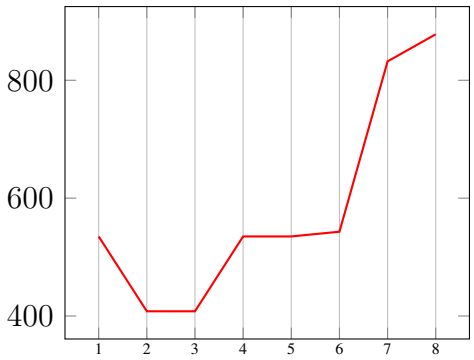
(b) DTLZ1



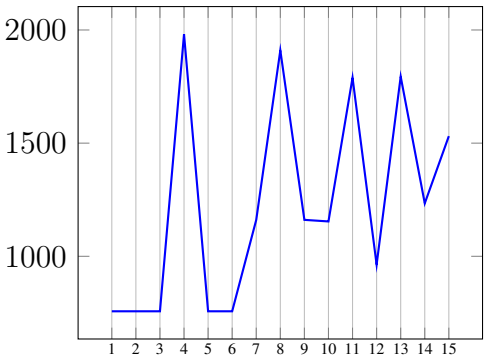
(c) DTLZ2



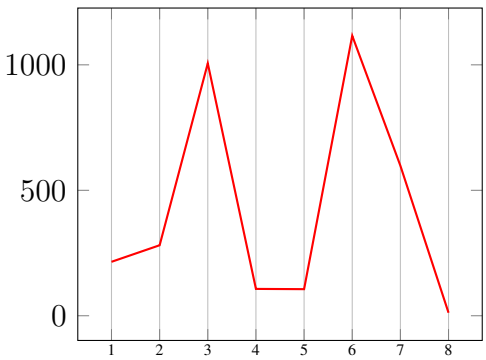
(d) DTLZ2



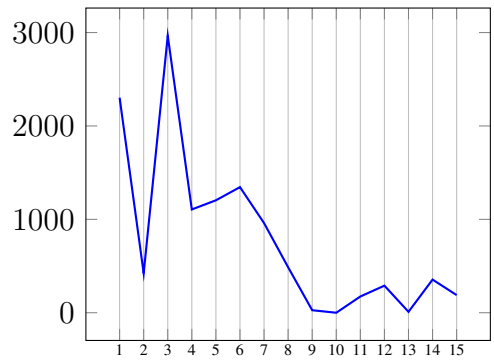
(e) DTLZ3



(f) DTLZ3



(g) DTLZ4



(h) DTLZ4

6 pav. Iteracija, kurioje buvo gautas geriausia kriterijaus reikšmė. Vaizdai a, c, e, g kai M = 8, b, d, f, h - M = 15. Abscisėje vaizduojami kriterijų indeksai, ordinatėje - iteracijų indeksai.

Uždavinys	M	Max Gen	NSGA-III	NSGA-III$^{\beta}$	NSGA-III*	Rez.
DTLZ1	5	600	5.17E-04	7.88E-04	4.36E-04	+ +
			9.80E-04	2.01E-03	1.12E-03	– +
			1.98E-03	1.42E-02	1.18E-02	– +
	8	750	2.04E-03	3.45E-03	2.52E-03	– +
			3.98E-03	5.28E-03	4.38E-03	– +
			8.72E-03	4.29E-02	2.07E-02	– +
	10	1000	2.22E-03	2.26E-03	2.13E-03	≈ +
			3.46E-03	4.24E-03	2.78E-03	+ +
			6.87E-03	2.75E-02	8.72E-03	– +
	15	1500	2.65E-03	2.82E-03	2.22E-03	+ +
			5.06E-03	6.71E-03	3.20E-03	+ +
			1.12E-02	1.10E-02	8.00E-03	+ +
DTLZ2	5	350	4.25E-03	4.80E-03	3.95E-03	+ +
			4.98E-03	6.16E-03	4.28E-03	+ +
			5.86E-03	7.61E-03	5.44E-03	+ +
	8	500	1.37E-02	1.54E-02	9.88E-03	+ +
			1.57E-02	1.74E-02	1.22E-02	+ +
			1.81E-02	2.00E-02	1.48E-02	+ +
	10	750	1.35E-02	1.38E-02	9.88E-03	+ +
			1.53E-02	1.73E-02	1.14E-02	+ +
			1.70E-02	2.00E-02	1.29E-02	+ +
	15	1000	1.36E-02	1.72E-02	1.01E-02	+ +
			1.73E-02	1.97E-02	1.25E-02	+ +
			2.11E-02	2.30E-02	3.23E-02	– –
DTLZ3	5	1000	3.09E-03	4.22E-03	1.74E-03	+ +
			5.96E-03	1.12E-02	4.51E-03	+ +
			1.20E-02	6.73E-02	1.87E-01	– –
	8	1000	1.24E-02	1.56E-02	1.21E-02	≈ +
			2.38E-02	3.01E-02	2.24E-02	+ +
			9.65E-02	1.39E-01	1.28E-01	– +
	10	1500	8.85E-03	1.06E-02	7.45E-03	+ +
			1.19E-02	1.53E-02	1.10E-02	+ +
			2.08E-02	1.33E-01	4.00E-02	– +
	15	2000	1.40E-02	1.30E-02	1.30E-02	+ ≈
			2.15E-02	2.84E-02	2.75E-02	– ≈
			4.20E-02	5.80E-02	3.62E-01	– –
DTLZ4	5	1000	9.85E-04	6.23E-04	3.59E-04	+ +
			1.26E-03	1.55E-03	7.14E-04	+ +
			1.72E-03	6.04E-03	4.81E-03	– +
	8	1250	5.08E-03	3.41E-03	3.26E-03	+ ≈
			7.05E-03	4.35E-03	4.23E-03	+ ≈
			6.05E-01	5.50E-03	9.23E-03	+ –
	10	1500	5.69E-03	3.63E-03	3.77E-03	+ ≈
			6.34E-03	4.60E-03	4.79E-03	+ ≈
			1.08E-01	5.30E-03	7.69E-03	+ –
	15	3000	7.11E-03	5.60E-03	5.50E-03	+ ≈
			3.43E-01	8.27E-03	1.05E-02	+ –
			1.07E+00	2.76E-02	2.19E-02	+ +

4 lentelė. Invertuoto apibendrindo atstumo įverčiai.

Naujoji elitinimo procedūra yra ypač efektyvi sprendžiant DTLZ2 uždavinį pagal visus nagrinėtus kriterijų skaičius (5, 8, 10, 15), net pagal 11 iš 12 palyginamų įverčių procedūra pasiekia geresnius rezultatus tiek už originalų algoritmą tiek, ir už realizaciją. Lyginant su realizacijos baze NSGA-III$^{\beta}$, elitinimo procedūra visus (12 iš 12), sprendžiamą DTLZ1 uždavinį, palyginamus įverčius. O pagal 22 įverčius iš 36, kai naudojama implementacijos bazė veikia prasčiau už originaliai pateiktus apversto apibendrinto atstumo įverčius, naujoji procedūra pagerina originalaus algoritmo rezultatą. Iš mažiausiai pokyčių patyrusių uždavinių sprendinių, DTLZ4 rezultatai atrodo mažiausiai pakitę, kadangi tik pagal 3 iš 12 palyginamų įverčių naujoji procedūra atrodo geriau.

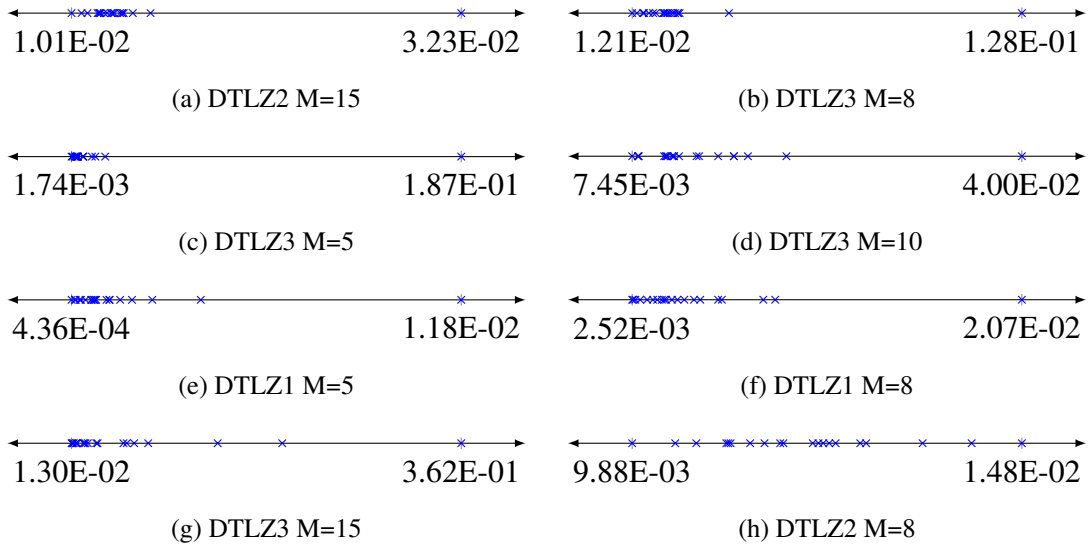
Naujai pristatyta procedūra šešiais atvejais pasiekia mažesnius minimalius apverstus apibendrintus atstumus nei originalus algoritmas, tačiau taip pat ir didesnius maksimalius atstumus. Toks rezultatų pasiskirstymas parodo, jog tais atvejais elitinimo procedūra nors ir pagerina rezultatus, tačiau taip pat ir įveda didesnį nestabilumą.

Analizuojant prasčiausius rezultatus pastebima, jog vienu atveju, sprendžiant 5 kriterijų DTLZ3 uždavinį blogiausias IGD įvertis buvo gautas virš 90 % (93.5 %) prastesnis nei originalaus algoritmo įvertis. Kadangi realizacijos bazės įvertis šiuo atveju buvo geresnis nei originalaus NSGA-III algoritmo įvertis tai elitinimo procedūra sukėlė prastą pasiekimą, kuris yra išskirtis žvelgiant į apversto apibendrinto atstimo įverčių išsibarstymą DTLZ3 M=5 atvejui (7 paveikslas). Pristatyta elitinimo procedūra trimis atvejais, blogiausiais DLTZ2 M=15, DTLZ3 M=5 ir M=15 įverčiais, pasiekia ypač prastus rezultatus, kadangi gaunami atstumai yra blogesni tiek už originalaus algoritmo pateiktus rezultatus, tiek ir už algoritmo realizaciją NSGA-III$^{\beta}$.

Svarbu ne tik tai, kaip gerai pasiskirsto trys pagrindinės atversto apibendrinto atstumo reikšmės, tačiau ir kaip per visus bandymus išsibarsto įverčiai. Įverčių išsibarstymai pavaizduoti 7 bei 8 paveiksluose, o jų išsibarstymo dydis įvertinamas standartiniu nuokrypiu, kurio rezultatai bei atverstų apibendrintų atstumų vidurkiai pavaizduoti 5 lentelėje. Kadangi standarinio nuokrypio reikšmė priklauso nuo reikšmių mastelio, tai sąlyginai įvertinti sprendinių apversto atstumo įverčių pasiskirstymą įverčiai buvo normuoti, o standartinio nuokrypio reikšmė apskaičiuota naudojant normuotus įverčius.

Kaip galima pastebėti, mažas standarinio nuokrypio įvertis įprastai reiškia, jog yra vienas apversto apibendrinto atstumo įvertis, nutolęs nuo visos grupės (išskirtis). Iš visų rezultatų išsibarstymų pasiekiančių geriausią standartinį nuokrypį didžioji dalis turi aiškiai matomą regioną taškų, kuriame išsibarsto absoliuti dauguma rezultatų ir yra tik vienas ar keli nutolę gerokai toliau. Tai ypač atsispindi sprendžiant M=5 kriterijų DLTZ3 uždavinį, kur visi išskyrus vieną sprendiniai išsibarstę nuo 1.74×10^{-3} iki 1.77×10^{-2}, o vienos išskirties rezultatas yra lygus 1.87×10^{-1}. Geriausiai pasiskirstę IGD įverčiai atlikus 20 bandymų, ypač nedideliame, tarp 9.88×10^{-3} ir 1.48×10^{-2},

reikšmių rėžyje, yra 8 kriterijų DTLZ2 uždavinio sprendiniai.



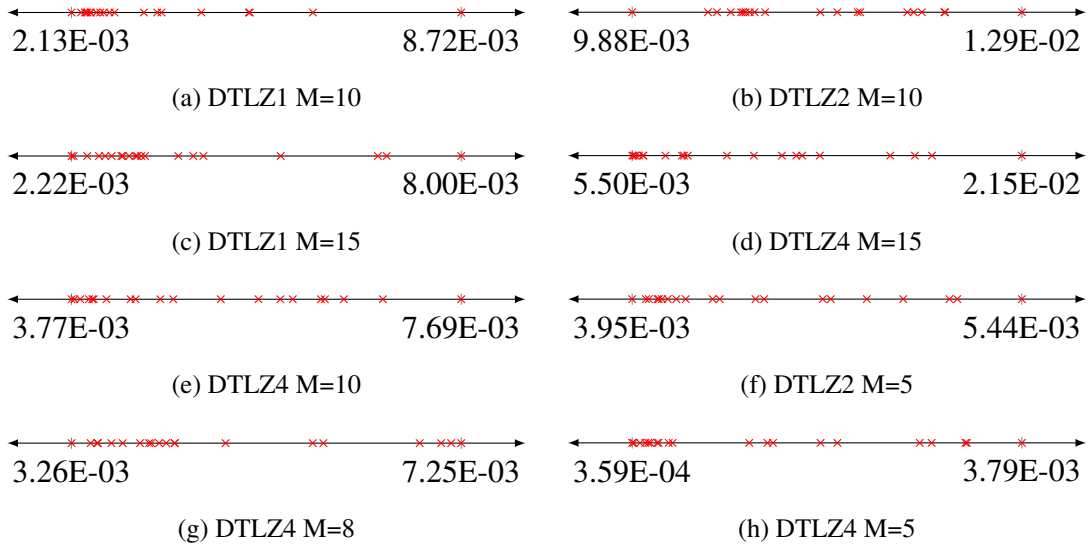
(a) DTLZ2 M=15 (b) DTLZ3 M=8

(c) DTLZ3 M=5 (d) DTLZ3 M=10

(e) DTLZ1 M=5 (f) DTLZ1 M=8

(g) DTLZ3 M=15 (h) DTLZ2 M=8

7 pav. 8 atvejai, kuriuose santykinis standartinis nuokrypis yra mažiausias. Vaizduojami 20 bandymų apversto apibendrinto atstumo įverčiai.



(a) DTLZ1 M=10 (b) DTLZ2 M=10

(c) DTLZ1 M=15 (d) DTLZ4 M=15

(e) DTLZ4 M=10 (f) DTLZ2 M=5

(g) DTLZ4 M=8 (h) DTLZ4 M=5

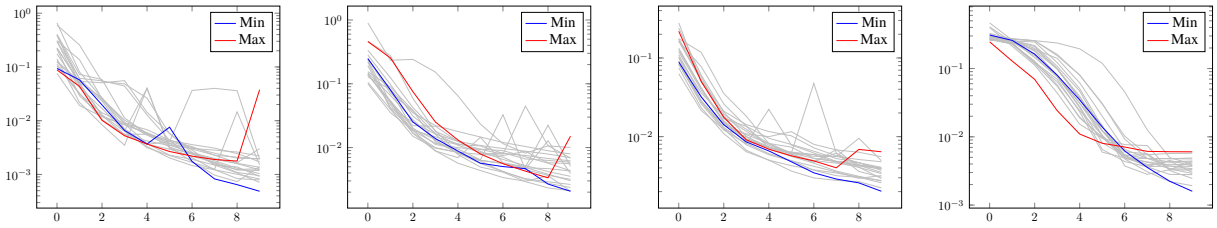
8 pav. 8 atvejai, kuriuose santykinis standartinis nuokrypis yra didžiausias. Vaizduojami 20 bandymų apversto apibendrinto atstumo įverčiai.

Žiūrint į didžiausią dimensiją turinčias, 15 kriterijų, užduotis DLTZ1 ir DTLZ4, jų sprendiniai pasiskirstė tolygiai, o DTLZ2, nors ir turi vieną išskirtį, tačiau atotrūkis nėra didelis. DTLZ3 užduotis tiek ir 15-os kriterijų tiek ir visais kitais atvejais sunkiausiai atranda visus sprendinius, pasiskirsčiusius nedideliame atstume. Visų DTLZ3 uždavinių sprendiniai sugeneruoja bent vieną sprendinį, kurio apversto apibendrinto atstumo įvertis yra gerokai nutolęs nuo grupės. Priešingai nei DTLZ3, DTLZ4 uždavinių, pagal visus kriterijus, sprendiniai išsibarstę tolygiai dažniausiai net ir mažame atstume.

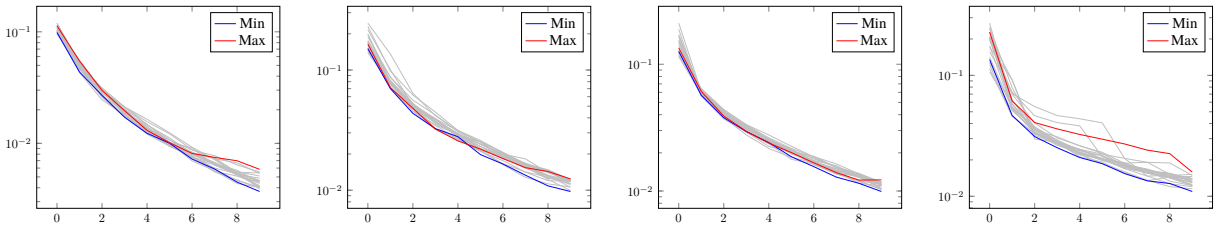
Uždavinys	**M**	μ	σ	**Uždavinys**	**M**	μ	σ
DTLZ1	5	1.88E-03	5.43E-04	DTLZ3	5	1.50E-02	8.86E-03
DTLZ1	8	5.50E-03	8.83E-04	DTLZ3	8	2.71E-02	5.37E-03
DTLZ1	10	3.51E-03	3.65E-04	DTLZ3	10	1.33E-02	1.56E-03
DTLZ1	15	3.80E-03	3.60E-04	DTLZ3	15	6.29E-02	1.86E-02
DTLZ2	5	4.45E-03	1.02E-04	DTLZ4	5	1.51E-03	2.56E-04
DTLZ2	8	1.20E-02	2.68E-04	DTLZ4	8	4.75E-03	2.97E-04
DTLZ2	10	1.13E-02	1.74E-04	DTLZ4	10	5.11E-03	2.63E-04
DTLZ2	15	1.33E-02	1.00E-03	DTLZ4	15	1.03E-02	1.05E-03

5 lentelė. Vidurkis ir standartinio nuokrypis sprendžiant 5,8,10,15 kriterijų DTLZ1-DTLZ4 problemas.

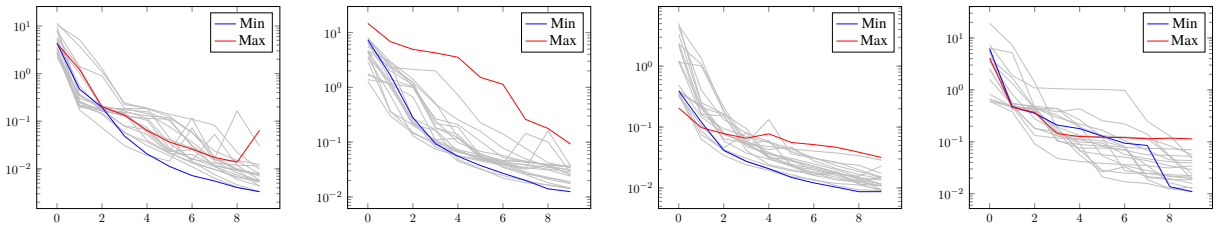
Norint suprasti kaip keičiasi apversto apibendrinto atstumo įvertis evoliucijos metu, sprendžiant kiekvieną uždavinį, kas 10 % buvo fiksuojamas apversto apibendrinto įvertis. Gautieji rezultatai yra pavaizduoti 9 paveiksle. Kaip galima pastebėti, net 10 iš 16 atvejų geriausių ir blogiausių įverčių kreivės susikerta, o dažnu atveju net ne kartą, evoliucijos tekmėje. Didžiausią dėmesį patraukia DTLZ1 M=5 uždavinio kreivės, kadangi blogiausio įverčio kreivė indikuoja staigų suprastėjimą, o geriausio įverčio kreivė sėkmingai leidžiasi žemyn. Labiausiai nemonotoniškas įverčių kreivių mažėjimas yra matomas sprendžiant visus DTLZ4 uždavinius, o ypač 15 kriterijų atveju. Stabiliausiai įverčių kreivės monotoniškai juda sprendžiant DTLZ2 uždavinius.



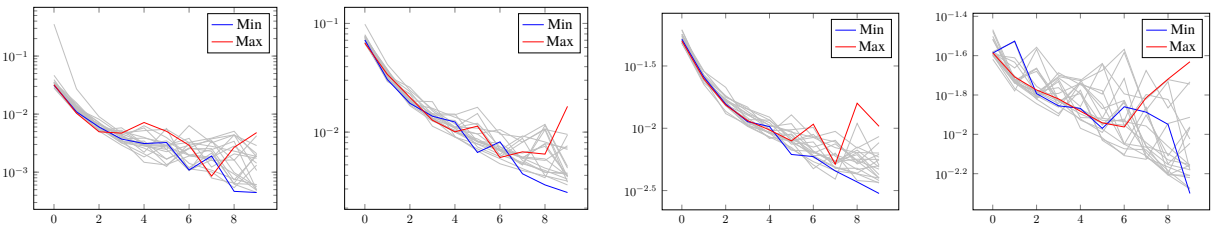
(a) DTLZ1 M=5 (b) DTLZ1 M=8 (c) DTLZ1 M=10 (d) DTLZ1 M=15



(e) DTLZ2 M=5 (f) DTLZ2 M=8 (g) DTLZ2 M=10 (h) DTLZ2 M=15



(i) DTLZ3 M=5 (j) DTLZ3 M=8 (k) DTLZ3 M=10 (l) DTLZ3 M=15



(m) DTLZ4 M=5 (n) DTLZ4 M=8 (o) DTLZ4 M=10 (p) DTLZ4 M=15

9 pav. Kiekvieno uždavinio 20 bandymų apversto apibrendinto atstumo įverčiai visos evoliucijos metu išskaidžius etapais.

4.6. Apibendrinimas ir išvados

Šiame skyriuje buvo išnagrinėtas NSGA-III algoritmas bei buvo ieškoma patobulinimo galimybių. Visų pirma buvo išnagrinėtas originalus NSGA-III algoritmas atkreipiant dėmesį į kiekvieną svarbią operaciją. Pirmąjai tyrimų zonai buvo pasirinkta įvertinti, kaip gerai algoritmas išlaiko individų tolygų pasiskirstymą kriterijų reikšmių erdvėje tam pasitelkiant Šanono entropijos įvertį. Po algoritmo analizės buvo pastebėta, jog algoritmas gerai išlaiko sprendinių tolygumą, tačiau galimai prarasdavo geriausius individus evoliucijos metu, o tai trukdydavo greičiau konverguoti į Pareto frontą. Šiam trūkumui pašalinti buvo pasinaudota atskaitos vektorių bei eilitinių narių, priskirtų jiems, euristika. Naujai pristatytas elitinimo procedūra aplinkos išrinkimo metu įtraukia elitinius narius į populiaciją jei jie galimai suteikia daugiau naudos.

Naujai pristatytos procedūros veiksmingumas išanalizuotas pagal tai, kaip anksti yra randama geriausia kriterijaus reikšmė, kurioje evoliucijos vietoje paskutinį kartą yra panaudojamas natūralios atrankos individas. Taip pat buvo nagrinėjama kaip kinta panaudotų bei atnaujintų elitinio archyvo narių skaičius kiekvienoje evoliucinėje iteracijoje.

Pagal pagrindinį algoritmų efektyvumo įvertinimo kriterijų, apverstą apibendrintą atstumą, naujoji elitinimo procedūra veikia efektyviai. Lyginant su originaliu NSGA-III algoritmo pateiktais įverčiais iš 48 palyginamų įverčių (DTLZ1-DTLZ4 $\times$ M=5,8,10,15 $\times$ {geriausia, mediana, blogiausia} reikšmės) net 69 % (33) buvo gauti geresni, o tik 27 % buvo prastesni. Rezultatus lyginant su naudojama algoritmo implementacija NSGA-III$^{\beta}$ elitinimo procedūros efektyvumas ypač jaučiamas, kadangi net 73 % (35) įverčių gauti geresni, o prastesni tik 12 % (6).

Pagal atliktų ekperimentų ir gautų kiekvieno sprendimo apverstų apibendrintų atstumų įverčius matoma, jog dažnai pasitaiko, jog uždavinio sprendinių sugeneruoti atstumai daugeliu atveju pasiekia gerus rezultatus, tačiau atsiranda vienas sprendinys, ženkliai nutolęs nuo grupės. Ypatingai gerai yra sprendžiamas 8 kriterijų DTLZ2 uždavinys, kurio IGD įverčiai pasiskirsto mažiausiame rėžyje. DTLZ4 uždavinių sprendiniai sugeneruoja labiausiai tolygiai pasiskirsčiusius rezultatus, tačiau rėžai kartais yra didelis, kai tuo tarpu DTLZ3 dažniausiai generuoja didžiąją daugumą sprendinių nedideliame rėžyje, tačiau atsiranda vienas ar keli sprendiniai reikšmingai nutolę nuo grupės.

Po atliktos analizės ir eksperimentų buvo įžvelgta dar daugiau vietų, kurios tūrėtų būti atrastos ir suprastos bei būdai, kaip naujojosios procedūros efektyvumas gali būti verifikuotas. Galimi tolimesni tyrimai:

- Empiriškai išvesti koeficientai **4 algoritme** 7 eilutėje kol kas neturi analitinio pagrindimo ir reikėtų detaliau ištirti kodėl natūralia atranka išrinkti individai leidžia pasiekti geresnių rezultatų;

- Rezultatams ir efektyvumui verifikuoti reikia įtraukti dar vieną, kitų algoritmų pristatymuose dažnai naudojamą, rinkinį, WFG [HBW+05], kuris savyje turi įvairių charakteristikų uždavinių, pvz. nutrūkstamo ir/arba išgaubto Pareto fronto uždavinius;

- Vizualizuoti, netrivialiai perteikiamus daugelio kriterijų, didelių dimensijų, sprendinius ir atlikti vizualaus duomenų perteikimo analizę.

Rezultatai pristatyti mokslinėje konferencijoje „Lietuvos magistrantų informatikos ir IT tyrimai (2019)“, 2019 m. gegužės 14 d.

Publikuoti straipsnyje: Laukevičius, Viktoras. Atskaitos taškais grįsti daugelio kriterijų optimizavimo evoliuciniai algoritmai. Lietuvos magistrantų informatikos ir IT tyrimai (2019): 24–30, https://doi.org/10.15388/LMITT.2019.

Literatūra

[ABV18] R. F. Alexandre, C. H. N. R. Barbosa ir J. A. Vasconcelos. LONSA: A labeling-oriented non-dominated sorting algorithm for evolutionary many-objective optimization. *Swarm and Evolutionary Computation*, 38:275–286, 2018. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2017.08.003. URL: http://www.sciencedirect.com/science/article/pii/S2210650217306806.

[Amo12] Kaveh Amouzgar. *Multi-Objective Optimization using Genetic Algorithms*. PhD dissertation, School of Engineering in Jönköping, 2012.

[BZ11] J. Bader ir E. Zitzler. Hype: An Algorithm for Fast Hypervolume-based Many-objective Optimization. *Evol. Comput.*, 19(1):45–76, 2011-03. ISSN: 1063-6560. DOI: 10.1162/EVCO_a_00009. URL: http://dx.doi.org/10.1162/EVCO_a_00009.

[CJO+16] R. Cheng, Y. Jin, M. Olhofer ir B. Sendhoff. A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization. 20, 2016-01.

[DD98] I. Das ir J. E. Dennis. Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems. *SIAM J. on Optimization*, 8(3):631–657, 1998-03. ISSN: 1052-6234. DOI: 10.1137/S1052623496307510. URL: http://dx.doi.org/10.1137/S1052623496307510.

[DJ14] K. Deb ir H. Jain. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014. DOI: 10.1109/TEVC.2013.2281535.

[DPA+02] K. Deb, A. Pratap, S. Agarwal ir T. A. M. T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. 6:182–197, 2002-05.

[DTL+05] K. Deb, L. Thiele, M. Laumanns ir E. Zitzler. *Scalable Test Problems for Evolutionary Multiobjective Optimization*. *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*. Springer London, London, 2005, p. 105–145. ISBN: 978-1-84628-137-2. DOI: 10.1007/1-84628-137-7_6. URL: https://doi.org/10.1007/1-84628-137-7_6.

[EKL00] E. Zitzler and K. Deb and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evol. Comput.*:173–195, 2000. ISSN: 1063-6560. DOI: 10.1162/106365600568202. URL: http://dx.doi.org/10.1162/106365600568202.

[FEO13] M. Fayek, H. M. El-Boghdadi ir S. M. Omran. Multi-objective Optimization of Technical Stock Market Indicators using GAs. 68:41–48, 2013-04.

[FJD11] A. Faez, A. Jindal ir K. Deb. Cricket Team Selection Using Evolutionary Multi-objective Optimization:71–78, 2011. DOI: 10.1007/978-3-642-27242-4_9. URL: https://doi.org/10.1007/978-3-642-27242-4_9.

[FLK+17] E. Filatovas, A. Lančinskas, O. Kurasova ir J. Žilinskas. A preference-based multi-objective evolutionary algorithm R-NSGA-II with stochastic local search. *Central European Journal of Operations Research*, 25(4):859–878, 2017-12.

[GD04] A. Ghosh ir S. Dehuri. Evolutionary Algorithms for Multi-Criterion Optimization: A Survey. 2, 2004-01.

[GLM12] M. O. W. Grond, N. H. Luong ir J. Morren. Multi-objective optimization techniques and applications in electric power systems. *2012 47th International Universities Power Engineering Conference (UPEC)*, p. 1–6, 2012-09.

[HBW+05] S. Huband, L. Barone, L. While ir P. Hingston. A Scalable Multi-objective Test Problem Toolkit. Carlos A. Coello Coello, Arturo Hernández Aguirre ir Eckart Zitzler, redaktoriai, *Evolutionary Multi-Criterion Optimization*, p. 280–295. Springer Berlin Heidelberg, 2005. ISBN: 978-3-540-31880-4.

[HM79] C. Hwang ir A. S. M. Masud. *Multiple Objective Decision Making—Methods and Applications: A State-of-the-Art Survey*, tom. 164. 1979-01. ISBN: 978-3-642-45511-7.

[IRM+16] A. Ibrahim, S. Rahnamayan, M. V. Martin ir K. Deb. EliteNSGA-III: An improved evolutionary many-objective optimization algorithm. *2016 IEEE Congress on Evolutionary Computation (CEC)*:973–982, 2016.

[JHS+13] L. C. Jiao, W. Handing, R. H. Shang ir F. Liu. A co-evolutionary multi-objective optimization algorithm based on direction vectors. *Information Sciences*, 228:90–112, 2013. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2012.12.013. URL: http://www.sciencedirect.com/science/article/pii/S0020025512007967.

[LLY+17] J. Luo, Q. Liu, Y. Yang, X. Li, M. Chen ir W. Cao. An artificial bee colony algorithm for multi-objective optimisation. *Applied Soft Computing*, 50:235–251, 2017. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2016.11.014. URL: http://www.sciencedirect.com/science/article/pii/S156849461630583X.

[LLT+15] B. Li, J. Li, K. Tang ir X. Yao. Many-Objective Evolutionary Algorithms: A Survey. *ACM Comput. Surv.*, 48(1):13:1–13:35, 2015-09. ISSN: 0360-0300. DOI: 10.1145/2792984. URL: http://doi.acm.org/10.1145/2792984.

[MR17] S. Mane ir M. R. N. Rao. Many-Objective Optimization: Problems and Evolutionary Algorithms - A Short Review. *International Journal of Applied Engineering Research*, 12, 2017-11.

[Sch85] J. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms:93–100, 1985-01.

[SD94] N. Srinivas ir K. Deb. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evol. Comput.*, 2(3):221–248, 1994-09. ISSN: 1063-6560. DOI: 10.1162/evco.1994.2.3.221. URL: http://dx.doi.org/10.1162/evco.1994.2.3.221.

[SYY18] T. Sun, G. G. Yen ir Z. Yi. IGD Indicator-based Evolutionary Algorithm for Many-objective Optimization Problems. *CoRR*, abs/1802.08792, 2018.

[Sim13] Dan Simon. *Evolutionary Optimization Algorithms*. Wiley, Hoboken, United States, 2013.

[Str02] Felix Streichert. Introduction to Evolutionary Algorithms, 2002.

[TC16] T. Chiang. nsga3cpp: A C++ implementation of NSGA-III, 2016. URL: http://web.ntnu.edu.tw/~tcchiang/publications/nsga3cpp/nsga3cpp-validation.htm.

[ZDZ+18] C. Zhou, G. Dai, C. Zhang, X. Li ir K. Ma. Entropy based evolutionary algorithm with adaptive reference points for many-objective optimization problems. *Information Sciences*, 465:232–247, 2018. DOI: https://doi.org/10.1016/j.ins.2018.07.012. URL: http://www.sciencedirect.com/science/article/pii/S0020025518305255.

[Zit99] Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD dissertation, Swiss Federal Institute of Technology Zurich, 1999.

[ZL07] Q. Zhang ir H. Li. MOEA/D: A multi-objective evolutionary algorithm based on decomposition. 6, 2007-01.

[ZL18] M. Zhang ir H. Li. A reference direction and entropy based evolutionary algorithm for many-objective optimization. *Applied Soft Computing*, 70:108–130, 2018. DOI: https://doi.org/10.1016/j.asoc.2018.05.011. URL: http://www.sciencedirect.com/science/article/pii/S1568494618302746.

[ZLT01] E. Zitzler, M. Laumanns ir L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. 103, 2001-07.