VILNIAUS UNIVERSITETAS

Vytautas Jančauskas

DALELIŲ SPIEČIŲ OPTIMIZAVIMO ALGORITMŲ TAIKYMO DAUGIAKRITERIAMS UŽDAVINIAMS EFEKTYVUMO TYRIMAS

Daktaro disertacijos santrauka
Fiziniai mokslai, informatika (09P)

Vilnius, 2016

Disertacija rengta 2011 — 2015 metais Vilniaus universiteto Matematikos ir informatikos institute.

Mokslinis vadovas:

prof. habil. dr. Antanas Žilinskas (Vilniaus universitetas, fiziniai mokslai, informatika - 09P).

Disertacija ginama Vilniaus universiteto Informatikos mokslo krypties taryboje:

Pirmininkas
prof. habil. dr. Gintautas Dzemyda (Vilniaus universitetas, fiziniai mokslai, informatika - 09P),
Nariai:
dr. Iskander Aliev (Kardifo Universitetas, Jungtinė Karalystė, fiziniai mokslai, informatika - 09P),
prof. habil. dr. Leonidas Sakalauskas (Vilniaus universitetas, fiziniai mokslai, informatika - 09P),
prof. habil. dr. Rimantas Šeinauskas (Kauno technologijos universitetas, technologijos mokslai, informatikos inžinerija - 07T),
prof. dr. Rimantas Vaicekauskas (Vilniaus universitetas, fiziniai mokslai, informatika - 09P).

Disertacija bus ginama viešame Vilniaus universiteto Informatikos mokslo krypties tarybos posėdyje 2016 m. birželio mėn. 8 d. 13 val. Vilniaus universiteto Matematikos ir informatikos instituto 203 auditorijoje.
Adresas: Akademijos g. 4, 08663 Vilnius, Lietuva.

Disertacijos santrauka išsiuntinėta 2016 m. gegužės mėn. 6 d.
Disertaciją galima peržiūrėti Vilniaus universiteto bibliotekoje ir VU interneto svetainėje adresu: www.vu.lt/lt/naujienos/ivykiu-kalendorius.

VILNIUS UNIVERSITY

Vytautas Jančauskas

EVALUATING THE PERFORMANCE OF MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION ALGORITHMS

Summary of Doctoral Dissertation
Physical Sciences, Informatics (09P)

Vilnius, 2016

The dissertation work was carried out at the Insitute of Mathematics and Informatics, Vilnius University from 2011 to 2015.

Scientific Supervisor:

Prof. Dr. Habil. Antanas Žilinskas (Vilnius University, Physical Sciences, Informatics - 09P).

The dissertation will be defended at the Council of the Scientific Field of Informatics of Vilnius University:

Chairman
Prof. Dr. Habil. Gintautas Dzemyda (Vilnius University, Physical Sciences, Informatics - 09P),
Members:
dr. Iskander Aliev (Cardiff University, UK, Physical Sciences, Informatics - 09P),
Prof. Dr. Habil. Leonidas Sakalauskas (Vilnius University, Physical Sciences, Informatics - 09P),
Prof. Dr. Habil. Rimantas Šeinauskas (Kaunas University of Technology, Technological Sciences, Informatics Engineering - 07T),
Prof. Dr. Rimantas Vaicekauskas (Vilnius University, Physical Sciences, Informatics - 09P).

The dissertation will be defended at the public meeting of the Council of the Scientific Field of Informatics Sciences of Vilnius University in the auditorium number 203 at the Institute of Mathematics and Informatics of Vilnius University at 13 a. m. on the 8th of June 2016.
Address: Akademijos g. 4, 08663 Vilnius, Lithuania.

The summary of the doctoral dissertation was distributed on the 6th of May, 2016.
The dissertation is available at the library of Vilnius University and online at www.vu.lt/lt/naujienos/ivykiu-kalendorius.

Turinys

1 Įvadas

Daugiakriteriai optimizavimo uždaviniai – tai tokie uždaviniai, kurių sprendimai vertinami naudojant kelis (galimai vienas kitam prieštaraujančius) kriterijus. Kitaip tariant, vienu metu optimizuojamos kelios funkcijos. Tokie yra dauguma uždavinių inžinerijoje, ekonomikoje ir kitose srityse. Pavyzdžiui, norima padidinti detalės patvarumą, bet kartu sumažinti jos pagaminimo kainą. Dažniausiai tokiais atvejais nėra vieno sprendimo. Metodai sukurti tokioms problemoms spręsti paprastai ieško sprendimų aibės, dar vadinamos Pareto aibe. Pareto aibę atitinka Pareto frontas, sudarytas iš Pareto aibę atitinkančių taškų kriterijų erdvėje. Pareto frontas yra aibė tarpusavyje nedominuojamų taškų. Tai reiškia, kad sprendimai vieno kriterijaus atžvilgiu negali būti pagerinti nepabloginant kitų kriterijų. Tradiciniai šių uždavinių sprendimo metodai aprašyti, pavyzdžiui, J. L. Cohon [2] [1], R. E. Steuer [10], J. Koski [7].

Egzistuoja daug metodų, skirtų daugiakriteriams uždaviniams spręsti. Išspręsti tokius uždavinius analitiškai yra įmanoma tik nedideliam visų daugiakriterių uždavinių poaibiui. Jei apie uždavinio matematines savybes yra žinoma nedaug (arba nežinoma nieko), tikslūs metodai negali būti taikomi. Tokiais atvejais taikomos įvairios euristikos. Dažnai naudojami gamtos įkvėpti metaeuristiniai metodai. Dažnai tokie metodai yra gaunami pritaikius optimizacijos metodus, skirtus spręsti vieno kriterijaus uždavinius daugiakriterei optimizacijai. Paprastas būdas tam pasiekti yra naudoti agreguotas funkcijas, sudarytas iš kriterijų reikšmių, susumuotų naudojant įvairius svorius. Taip daugiakriteris uždavinys paverčiamas į daug vieno kriterijaus uždavinių. Šie uždaviniai yra sprendžiami atskirai, naudojant vieno kriterijaus optimizavimo metodus, taip gaunant sprendimų aibę. Idealiu atveju vis dėlto norėtųsi, kad Pareto fronto aproksimaciją būtų galima gauti paleidus metodą vieną kartą. Šioje situacijoje iškyla kelios problemos. Pirma, reikia rasti ne vieną daugiakriterės optimizacijos sprendinį, o jų aibę. Dažniausiai tam naudojamas sprendimų archyvas, į kurį sprendimai, rasti algoritmo, yra priimami, remiantis tuo, ar jie yra dominuojami sprendimų, jau esančių archyve, ar ne. Sprendimai priimami į archyvą, jeigu jie yra nedominuojami nė vieno sprendimo, esančio archyve. Antra problema yra ta, kad reikia išlaikyti sprendimų įvairovę. Norisi, kad naudojant metodą rasti sprendimai tolygiai padengtų visą Pareto frontą. Tai skiriasi nuo vieno kriterijaus optimizacijos tuo, kad ten populiacijos individų rasti sprendimai konverguoja į vieną tašką. Tarp populiarių daugiakriterės optimizacijos metodų yra genetiniai algoritmai. Jų veikimo principas (bendrais bruožais) yra toks: iš sprendimų populiacijos išrenkami ge-

riausi individai ir jų pagrindu kuriami nauji sprendimai. Iš populiacijos išmetami prasti sprendimai ir viskas kartojama iš naujo. Tam, kad veiktų su daugiakriteriais uždaviniais, šie algoritmai yra modifikuojami atsižvelgiant į dvi mūsų anksčiau įvardytas problemas.

Dalelių spiečių optimizacija (DSO), aprašyta J. Kennedy et al. [6], yra populiacijomis pagrįstas metodas, įkvėptas tam tikrų gyvūnų (paukščių ir žuvų) socialinio elgesio. Pagal šį metodą simuliuojama paukščių (žuvų, ar kt.) spiečiaus dinamika. Plačiau apie gyvūnų elgesį kalba F. Heppner et al. [5]. Šiuo atveju dalelės yra taškai sprendimų erdvėje, kurios juda pagal pozicijos ir greičio vektorių atnaujinimo taisykles. Šios taisyklės yra sukurtos taip, kad dalelės judėtų geriausio sprendimo, kurį jos rado iki šiol, ir geriausio sprendimo, kurį jų kaimynės rado iki šiol, link. Į tai gali būti žiūrima kaip į asmeninę bei socialinę patirtį. Šios patirtys yra derinamos naudojant atsitiktinius svorius. Taip formuojamas greičio vektorius, nusakantis kurios sprendimų erdvės srities link judės dalelės. Dalelių spiečiaus optimizacija, nors ir sukurta vieno kriterijaus uždaviniams spręsti, gali būti pritaikyta daugiakriterei optimizacijai, naudojant tuos pačius metodus, kurie taikomi daugiakriteriuose genetiniuose algoritmuose. Šiuo metu dalelių spiečių algoritmų taikymas daugiakriterėje optimizacijoje yra plačiai tiriamas. Literatūroje pasiūlyta daugybė algoritmų ir jų variantų.

Galima išskirti dvi problemas, kurios trukdo suprasti DSO taikymo daugiakriteriams uždaviniams perspektyvas. Pirmiausia yra neaišku, kurie iš egzistuojančių metodų yra daugiausia žadantys ir kuriuos labiausiai apsimoka tirti toliau. Egzistuojantys metodai paprastai lyginami tik su pora kitų (anksčiau sukurtų) metodų. Be to, jų savybės tiriamos naudojant mažą skaičių testinių uždavinių. Tai lemia, kad nežinoma, kurie metodai (ar metodų grupės) yra daugiausia žadantys. Būtų įdomu sužinoti, kurios DSO kūrime taikomos idėjos veda į sėkmingų metodų sukūrimą. Kita problema kyla iš to, kad nėra vienareikšmiškai priimtų būdų palyginti du daugiakriterio uždavinio sprendimus. Palyginti du vieno kriterijaus uždavinio sprendimus yra paprasta – tas, kuris duos mažesnę (ar didesnę, priklausys nuo uždavinio) funkcijos reikšmę, ir yra geresnis. Be to, galima tikrinti, per kiek funkcijos įvertinimų algoritmai randa sprendimą nurodytu tikslumu. Tai parodo, kuris iš dviejų metodų randa sprendimą greičiau. Galima tikrinti, kiek kartų, atliekant visus bandymus, metodas (jei jis stochastinis) randa globalų optimumą ir t. t. Visi šie aspektai yra lengvai išmatuojami. Taip nėra daugiakriterės optimizacijos atveju. Šiuokart sprendimai yra aibės, o palyginti dvi aibes yra sudėtinga. Dažnai įvertinimas priklauso nuo to, ką metodo naudotojas laiko geru sprendimu.

1.1 Tikslai ir tyrimo objektai

Šio tyrimo tikslas yra pagerinti egzistuojančius metodus, skirtus vertinti neiškilos daugiakriterės optimizacijos metodų efektyvumą. Efektyvumas šiuo atveju yra skaitiškai įvertintas skirtumas tarp Pareto fronto aproksimacijos, kuri buvo apskaičiuota taikant kurį nors metodą, ir tikrojo Pareto fronto. Šie metodai naudojami vertinant daugiakriterių DSO metodų efektyvumą. Dalelių spiečių optimizavimo metodai yra populiarūs metodai sprendžiant neiškilus daugiakriterės optimizacijos uždavinius. Tai yra akivaizdu iš aktyvių tyrimų, vykdomų taikant dalelių spiečių optimizaciją daugiakriteriams uždaviniams spręsti. Vis dėlto nėra atlikta pakankamai tyrimų, kurie apžvelgia šių metodų visumą. Taigi šios srities tyrimas galėtų būti naudingas. Buvo išskirti šie potiksliai:

- Peržvelgti efektyvumo indikatorius, skirtus neiškilių daugiakriterio optimizavimo uždavinių sprendimo metodų efektyvumui įvertinti. Šie indikatoriai kuriami siekiant išmatuoti optimizavimo metodų efektyvumą tam tikrais aspektais. Paprastai matuojamas atitikimas tarp optimizavimo metodo rastos Pareto fronto aproksimacijos ir tikrojo Pareto fronto. Atitikimas gali būti matuojamas keliais aspektais. Pavyzdžiui, galima matuoti vidutinį atstumą tarp aproksimacijos taškų ir tikrojo Pareto fronto. Arba galima matuoti, kaip tolygiai aproksimacijos taškai padengia tikrąjį Pareto frontą. Jei egzistuojantys indikatoriai nėra pakankami mūsų reikmėms, turi būti pasiūlyti nauji.
- Peržvelgti literatūroje aprašytus dalelių spiečių metodus daugiakriterei optimizacijai. Išskirti dažnai naudojamas idėjas. Remiantis šia informacija pasiūlyti naują metodų klasifikavimo būdą, jeigu egzistuojantys klasifikavimo būdai nepakankami atsižvelgiant į tyrimo rezultatus.
- Daugumai literatūroje aprašytų daugiakriterių dalelių spiečių algoritmų neegzistuoja laisvai prieinamų implementacijų. Siekiant tai ištaisyti, reikia sukurti programavimo karkasą, kuris leistų įgyvendinti šiuos metodus ir ištirti jų efektyvumą. Programinis karkasas susidės iš metodų implementacijų, testinių uždavinių ir efektyvumo indikatorių. Šios dalys bus naudojamos kartu, eksperimentiškai įvertinant metodus. Tai turi būti padaryti patogiu moduliniu būdu, leidžiančiu prireikus nesunkiai pridėti naujus metodus.
- Naudojant minėtą programinį karkasą, reikia eksperimentiškai įvertinti egzistuojančius dalelių spiečių algoritmus. To tikslas yra atlikti eksperimentinę lyginamąją

analizę. Tai leis spręsti, kurie iš metodų tinka konkrečioms užduočių klasėms. Taip yra todėl, kad tiek metodų implementacijos detalės, tiek uždavinių teorinės savybės yra iš anksto žinomos. Išanalizavus eksperimentų rezultatus, galima spręsti, kokie metodai tinkamiausi uždaviniams su konkrečiomis savybėmis.

- Pasiūlyti naujus dalelių spiečių metodus, skirtus daugiakriterei optimizacijai. Tam bus panaudoti anksčiau minėtų eksperimentų rezultatai.

1.2 Metodologija

Siekiant įvertinti daugiakriterių dalelių spiečių algoritmų efektyvumą, buvo atlikti eksperimentai naudojant įvairius metodus, pasiūlytus literatūroje. Šie metodai buvo panaudoti sprendžiant 13 testinių uždavinių, kurių matematinės savybės (įskaitant Pareto frontus ir aibes) yra žinomos. Eksperimentų rezultatai buvo vertinami naudojant keletą efektyvumo indikatorių. Buvo sukurtos patogios naudoti ir įdiegti Python kalbos bibliotekos. Šios bibliotekos įgyvendina programinį karkasą, kuris leidžia kurti ir testuoti vieno ir kelių kriterijų dalelių spiečių optimizavimo algoritmus ir palaiko įvairias spiečių topologijas bei dalelių pozicijos (greičio) atnaujinimo taisykles. Programinė įranga yra modulinė ir leidžia patogiai įgyvendinti ar testuoti naujus daugiakriterius dalelių spiečių algoritmus, analizuoti eksperimentų rezultatus. Visi eksperimentai buvo atlikti superkompiuteriuose. Atlikti eksperimentus paskirstytų skaičiavimų aplinkose reikia dėl to, kad eksperimentai reikalauja daug skaičiavimo resursų. Jeigu turime n metodų, m testinių uždavinių ir norime eksperimentą atlikti k kartų (kad gautume statistiškai reikšmingų rezultatų) reikės atlikti $n \times m \times k$ bandymų. Gauti Pareto fronto aproksimaciją naudojant šiuos metodus paprastai trunka kelias minutes. Taigi eksperimentams atlikti gali prireikti savaičių ar net mėnesių, jeigu eksperimentai būtų atliekami viename kompiuteryje, o ne lygiagrečiai. Sukurta programinė įranga naudoja paskirstytų skaičiavimų sistemas eksperimentams lygiagrečiai atlikti ir rezultatams apdoroti. Komunikacijai tarp procesų buvo naudojama plačiai paplitusi MPI sistema.

1.3 Mokslinis naujumas ir rezultatai

Literatūroje yra pasiūlyta daug dalelių spiečių algoritmų daugiakriteriams uždaviniams spręsti, tačiau nėra jokios sisteminės šių metodų apžvalgos ir kokybinio vertinimo. Nauji

metodai paprastai yra lyginami su pora populiarių metodų naudojant keletą testinių uždavinių. Taigi yra poreikis ištirti šių metodų savybes ir atlikti apibendrinamąją analizę. Tai leis išrinkti efektyvius metodus ir nuspręsti, kokie metodai tinka kurioms uždavinių klasėms spręsti. Atliekant tyrimus buvo pasiekta tokių rezultatų:

- Pasiūlyti du nauji efektyvumo indikatoriai. Šie indikatoriai buvo sukurti siekiant išmatuoti, kaip tolygiai Pareto aibės aproksimacija padengia tikrąjį Pareto frontą.
- Pasiūlyti du nauji dalelių spiečių optimizacijos metodai daugiakriterei optimizacijai. Abu naudoja heterogeninius spiečius siekiant pagerinti metodų efektyvumą. Heterogeniniuose spiečiuose naudojamos dalelės su skirtingomis greičio ir pozicijos atnaujinimo taisyklėmis. Šios dalelės keičiasi informacija naudodamos vieną ir tą patį nedominuojamų taškų archyvą. Kai dalelės sukurtos taip, kad kompensuotų viena kitos trūkumus, toks spiečius leidžia pagerinti optimizavimo efektyvumą įvairiais aspektais.
- Atlikta egzistuojančių metodų apžvalga. Metodai sistematizuoti atsižvelgiant į jų naudojamus sprendimus. Taip išskiriamos kelios metodų grupės, tarpusavyje siejančios metodus, kurie naudoja tuos pačius metodus kuriant iškylančių uždavinių sprendimus.
- Pasiūlytos metodologijos palyginti skirtingiems daugiakriterio optimizavimo metodams. Metodologija susideda iš rekomendacijų eksperimentinei procedūrai, efektyvumo indikatorių ir testinių uždavinių. Buvo pasiūlyti metodai tokių eksperimentų metu gautiems duomenims analizuoti.
- Egzistuojantys dalelių spiečių metodai daugiakriterei optimizacijai buvo išsamiai eksperimentiškai ištirti ir palyginti tarpusavyje. Tyrime padengta dauguma egzistuojančių metodų, kurie buvo aprašyti mokslinėje literatūroje.
- Buvo sukurtas programinis karkasas Python programavimo kalba. Jis naudotojui leidžia panaudoti dešimtis skirtingų dalelių tipų, keletą dalelių spiečių topologijų ir dešimtis testinių daugiakriterių uždavinių. Visu tuo pasinaudojus įgalina kurti naujus metodus ir juos vertinti pagal efektyvumą.

1.4 Praktinė reikšmė

Daugiakriterė optimizacija turi daug svarbių praktinių taikymų, kadangi dauguma realiai iškylančių uždavinių ekonomikoje, inžinerijoje ir t. t. yra būtent kelių kriterijų uždaviniai. Šios disertacijos tematika buvo remtasi sprendžiant verslo proceso modeliavimo diagramų vizualizacijos uždavinius. Uždavinys buvo išspręstas sukuriant žiniatinklio paslaugą, kuri randa sprendimus diagramų piešimo uždaviniams. Paslauga naudoja daugiakriterės optimizacijos metodus ir pateikia naudotujui diagramą XML formatu šiam atsiuntus duomenis. Paslaugos išeities tekstai gali būti rasti šiuo adresu:

https://bitbucket.org/bucket_brigade/aco

Disertacijos rezultatai buvo naudojami siekiant tokių projektų tikslų:

- „Programinės įrangos skirtos verslo procesų modelių diagramų vizualizacijai kūrimas“ projektas finansuotas MITA, projekto Nr. 31V-145 (2011–2012), Nr. 31V-31 (2013).
- „Neiškiloji daugiakriterė optimizacija: metodai ir algoritmai“ finansuota iš projekto Nr. MIP-063/2012 lėšų, skirtų Lietuvos mokslo tarybos (2012–2014).

1.5 Ginamieji teiginiai

- Pasiūlyti daugiakriterės optimizacijos metodų efektyvumo indikatoriai gali būti naudojami Pareto fronto aproksimacijos pasiskirstymo tolydumui palei tikrąjį Pareto frontą matuoti. Šie (ir panašūs) indikatoriai reikalingi, nes egzistuojantys efektyvmo indikatoriai turi trūkumų, kurie aprašyti disertacijos 4 skyriuje. Dėl šių trūkumų indikatoriai neatspindi intuityvios tolydumo sąvokos ir bendru atveju negali būti naudojami padengimo tolydumui vertinti. Šie trūkumai yra sprendžiami pasiūlytuose indikatoriuose. Suformuluojame, ką turime omenyje, sakydami „intuityvi padengimo tolydumo“ sąvoka ir sukuriame indikatorius taip, kad būtų užtikrinama, jog šios sąvokos būtų laikomasi skaičiuojant indikatoriaus reikšmę.
- Iš eksperimentų rezultatų matyti, kad pasiūlyti heterogeniniai dalelių spiečių variantai duoda gerų rezultatų optimizuojant pasirinktas testines funkcijas ir naudojant pasirinktus kokybės indikatorius, palyginti su egzistuojančiais metodais.

- Literatūroje aprašyta daug atvejų, kai mutacijos operatorių panaudojimas pagerina vieno kriterijaus optimizavimo metodų efektyvumą (mutacija nenaudota originaliame DSO algoritme). Nėra išsamesnių tyrimų, kurie tirtų mutacijos panaudojimą daugiakriterėje optimizacijoje panaudojant DSO. Disertacijos rašymo metu buvo ištirta daug metodų, tarp jų tokių, kurie naudoja mutaciją, ir tokių, kurie jos nenaudoja. Iš rezultatų matyti, kad mutacijos panaudojimas pagerina efektyvumą optimizuojant daugiakriterius uždavinius. Visais atvejais, metodai kurie nenaudoja mutacijos duoda didesnes I_{IGD} ir I_{GD} reikšmes. Vienintelė išimtis yra dekompozicija paremti metodai.

- Metodai, naudojantys daugiakriterių uždavinių dekompoziciją, efektyviai veikia su uždaviniais, kurių Pareto frontas nėra visur tolydus (pvz., sudarytas iš diskrečių taškų). Taip gali būti dėl to, kad šie metodai nesiremia prielaidomis apie Pareto fronto tolydumą.

- Iš eksperimentinių rezultatų, gautų trečiame disertacijos skyriuje, matome, kad „Vector Evaluated MOPSO“ metodai neduoda gerų rezultatų, palyginti su kitais daugiakriteriais DSO metodais. Keli tokie metodai buvo ištirti ir nė vienas iš jų nedavė tokių rezultatų, kurie efektyvumu prilygtų kitiems daugiakriteriams DSO metodams.

1.6 Autoriaus publikacijos

Čia pateikiamas sąrašas publikacijų, savo tema susijusių su disertacija ir atspausdintų recenzuojamuose leidiniuose. Be to, pateikiamas sąrašas tarptautinėse konferencijose skaitytų pranešimų disertacijos temomis.

1.6.1 Periodiniai leidiniai

Publikacijos šios disertacijos tema buvo išspausdintos šiuose periodiniuose leidiniuose:

1. Jančauskas, Vytautas. Empirical Study of Particle Swarm Optimization Mutation Operators. Baltic Journal of Modern Computing, Vol. 2 (2014), No.4, pp. 199 – 214.

2. Jančauskas, Vytautas. Heterogeneous Multi-Objective Particle Swarm Optimiser with a Spread Particle. International Journal of Research Studies in Science, Engineering and Technology, Vol. 2, Issue 9, 2015 rugsėjis, pp. 30 – 49. ISSN 2349 – 4751.

1.6.2 Pranešimai konferencijose

Darbai šios disertacijos tema buvo pristatyti šiose tarptautinėse konferencijose:

1. Jančauskas, Vytautas; Žilinskas, Antanas. On Multi-Objective Black Box Optimization of Expensive Objectives. 25th European Conference on Operational Research (EURO XXV), 2012 liepa, Vilnius, Lietuva.
2. Jančauskas, Vytautas; Kaukas, Giedrius; Žilinskas, Antanas; Žilinskas, Julius. On Multi-Objective Optimization Aided Visualization of Graphs Related to Business Process Diagrams. Tenth International Baltic Conference on Databases and Information Systems (Baltic DB&IS), 2012 liepa, Vilnius, Lietuva.
3. Jančauskas, Vytautas. Empirical Study of Particle Swarm Optimization Mutation Operators. Veszprém Optimization Conference: Advanced Algorithms (VOCAL) 2012 gruodis, 2012, Vešpremas, Vengrija.
4. Jančauskas, Vytautas. Optimizing Neighbourhood Distances for a Variant of Fully-Informed Particle Swarm Algorithm. VI International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2013), 2013 rugsėjis, Kenterberis, Jungtinė Karalystė.
5. Jančauskas, Vytautas. Using Nature-Inspired Optimization Methods to Fit Spectra of Supernovae. 9th Wuerzburg Workshop on Supernovae and Related Topics, 2015 gruodis, Viurcburgas, Vokietija.

1.6.3 Publikacijos recenzuojamuose konferencijų leidiniuose

Darbai šios disertacijos tema buvo atspausdinti šiuose recenzuojamuose konferencijoms skirtuose leidiniuose:

1. Jančauskas, Vytautas, Aušra Mackute-Varoneckiene, Audrius Varoneckas, and Antanas Žilinskas. "On the multi-objective optimization aided drawing of connectors for graphs related to business process management." In Information and Software Technologies, pp. 87–100. Springer Berlin Heidelberg, 2012.

2. Jančauskas, Vytautas, Giedrius Kaukas, Antanas Žilinskas, and Julius Žilinskas. "On Multi-Objective Optimization Aided Visualization of Graphs Related to Business Process Diagrams." In DB&Local Proceedings, pp. 71–80. 2012.

3. Jančauskas, Vytautas. "Optimizing neighbourhood distances for a variant of fully-informed particle swarm algorithm." In Nature Inspired Cooperative Strategies for Optimization (NICSO 2013), pp. 217–229. Springer International Publishing, 2014.

2 Daugiakriterė optimizacija naudojant dalelių spiečių algoritmus

Dauguma realaus gyvenimo praktikoje iškylančių problemų susideda iš kelių kriterijų. Šiuos kriterijus reikia minimizuoti arba maksimizuoti tuo pačiu metu. Šie kriterijai dažnai yra tarpusavyje nesuderinami. Pavyzdžiui, noras sumažinti dalelės svorį gali sumažinti šios detalės tvirtumą ar padidinti kainą. Turint tokias užduotis dažnai tampa akivaizdu, kad jos neturi vieno sprendinio, tenkinančio visų kriterijų minimalumo sąlygą. Šiuo atveju sprendimas yra ne vienas vektorius, o vektorių aibė, dar vadinama Pareto aibe. Ši aibė sudaryta iš nedominuojamų taškų. Turint sprendimą iš šios aibės, nė vieno kriterijaus negalima pagerinti nepabloginant kitų kriterijų. Pareto frontas yra taškai kriterijų erdvėje, atitinkantys Pareto aibės taškus. Algoritmai kuriami taip, kad rastų šios aibės aproksimaciją. Aproksimacijos taškai turi leisti algoritmo naudotojui susidaryti tikslų vaizdą apie tai, kaip atrodo Pareto frontas. Bendru atveju metodai, leidžiantys rasti Pareto fronto analitinę išraišką, yra negalimi. Naudotojas turi iš aibės aproksimacijos išsirinkti tašką, kuris atitinka jo ar jos tuometinius prioritetus. Taškai turėtų tolygiai padengti tikrąjį Pareto frontą, kad naudotojas susidarytų tikslų jo paveikslą.

Kuriant daugiakriterės optimizacijos algoritmus, reikia atsižvelgti į daug aspektų. Iš jų galima išskirti šiuos kriterijus, kuriuos turėtų tenkinti sėkmingas metodas:

1. Maksimizuoti taškų skaičių aproksimacijoje.
2. Minimizuoti atstumą tarp aproksimacijos taškų ir Pareto fronto.
3. Maksimizuoti sprendimų pasiskirstimą palei Pareto frontą. Tai reiškia, kad aproksimacijos taškai turėtų tolygiai padengti Pareto frontą. Neturėtų būti Pareto fronto dalių, nepadengtų aproksimacijos taškais.

Į šiuos aspektus turėtų būti atsižvelgiama ir kuriant kokybės indikatorius, kurie matuoja Pareto fronto aproksimacijos „gerumą“. Kitais žodžiais tariant, sėkmingas algoritmas turėtų perteikti tikslų Pareto fronto paveikslą algoritmo naudotojui. Be to, pageidautina, kad algoritmas Pareto fronto aproksimaciją rastų per kuo trumpesnį laiką, kad nereiktų algoritmo kartoti keletą kartų. Taip yra, pavyzdžiui, su kriterijų agregavimo metodais. Tai įmanoma padaryti naudojant populiacijų algoritmus, tokius kaip DSO, kur kiekvienas populiacijos individas (dalelė) tiria atskirą Pareto fronto dalį.

Kuriant DSO algoritmus daugiakriterei optimizacijai, jų kūrėjas turės atsakyti mažiausiai į šiuos tris klausimus:

1. Kaip parinkti geriausią asmeninį sprendimą $\boldsymbol{p}_i$ dalelei su indeksu i ir kaip parinkti geriausią kaimynų sprendimą $\boldsymbol{g}_i$? Šių dviejų vektorių reikia, jeigu norima pritaikyti DSO greičio vektoriaus atnaujinimo formulę iš standartinio DSO algoritmo.
2. Kaip saugoti nedominuojamus sprendinius, kurie rasti einamosios algoritmo iteracijos metu? Šiuo atveju reikia atsižvelgti į skaičiavimų efektyvumą ir sprendimų sklaidos užtikrinimą. Naudojami įvairūs nedominuojamų taškų archyvų algoritmai.
3. Kaip užtikrinti tolygią sprendimų sklaidą? Kaip užtikrinti, kad nėra tokių Pareto fronto vietų, kurių algoritmas neaptiko?

Geriausias asmeninis sprendimas $\boldsymbol{p}_i$ dažniausiai atnaujinamas panašiai kaip vieno kriterijaus DSO atveju. Tiesiog vietoje paprasto aritmetinio palyginimo naudojama dominavimo sąvoka. Jeigu naujas sprendimas dominuoja seną $\boldsymbol{p}_i$, tai $\boldsymbol{p}_i$ pakeičiamas naujuoju sprendimu. Tai nėra vienintelelė galimybė. Galimos ir kitos $\boldsymbol{p}_i$ atnaujinimo taisyklės, atsižvelgiant į metodą. Parenkant $\boldsymbol{g}_i$, dažniausiai stengiamasi parinkti tokį sprendimą, kuris atitiktų menkai ištirtas Pareto fronto vietas. Tai gali būti atlikta naudojant „crowding distance“, „niching“, „hypervolume contribution“ ar kitus panašius metodus. Visi

jie siekia išrinkti sprendimą, kuris yra atokiausiai nuo kitų sprendimų kriterijų erdvėje. Nedominuojamų taškų archyvų algoritmai yra tie patys, kurie naudojami ir taikant kitus daugiakriterius metodus. Dauguma iš jų yra „paveldėti“ iš daugiakriterių evoliucinių algoritmų. Trečioji problema sprendžiama, parenkant imant kaip geriausią kaimynų sprendimą parenkant tokį, kuris atitinka menkai ištirtas Pareto fronto sritis.

Margarita Reyes-Sierra et al. [9] pateikia DSO algoritmų skirtų daugiakriterei optimizacijai apžvalgą.

Disertacijoje aprašomi šiuo metu literatūroje minimi metodai, taip pat ir senesni metodai. Į disertaciją bandoma įtraukti visus literatūroje sutinkamus DSO variantus daugiakriterei optimizacijai. Dauguma aptartų algoritmų, sutinkamų literatūroje ir aptartų disertacijoje, telpa į šabloną, pateiktą 1 algoritme. Pagrindinius skirtumus sudaro tai, kaip parenkami geriausias asmeninis ir kaimynų sprendimai – $\boldsymbol{p}_i$ ir $\boldsymbol{g}_i$. Algoritmai disertacijoje aprašyti pateikiant skirtumus tarp šio šablono ir kiekvieno konkretaus metodo.

Algorithm 1 Apibendrintas dalelių spiečiaus algoritmas daugiakriterei optimizacijai

1: **for** $i \leftarrow 1, n$ **do**
2: inicializuoti $\boldsymbol{x}_i$ ir $\boldsymbol{v}_i$
3: $\boldsymbol{p}_i \leftarrow \boldsymbol{x}_i$
4: **end for**
5: **while** netenkinamos sustojimo sąlygos **do**
6: atnaujinti nedominuojamų taškų archyvą $\mathbf{A}$
7: $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \boldsymbol{v}_i$
8: atnaujinti $\boldsymbol{p}_i$ ir $\boldsymbol{g}_i$
9: $\boldsymbol{v}_i \leftarrow w\boldsymbol{v}_i + \rho_1 \boldsymbol{U}_{(0,1)}(\boldsymbol{p}_i - \boldsymbol{x}_i) + \rho_2 \boldsymbol{U}_{(0,1)}(\boldsymbol{g}_i - \boldsymbol{x}_i)$
10: **end while**

Trumpai aptarsime, kas vyksta 1 algoritme. Visų pirma kiekvienai dalelei su indeksu i iš n dalelių inicializuojame $\boldsymbol{x}_i$ ir $\boldsymbol{v}_i$. Paprastai $\boldsymbol{x}_i$ inicializuojamas atsitiktinėmis reikšmėmis iš leidžiamų reikšmių intervalo. Vektorius $\boldsymbol{v}_i$ gali būti inicializuojamas nuliais arba atsitiktinėmis reikšmėmis. Tai atliekama cikle aprašytame nuo 1 eilutės iki 4 eilutės. Tada seka pagrindinis algoritmo ciklas. Pirmiausia atnaujinamas nedominuojamų taškų archyvas $\mathbf{A}$. Tai atliekama prieš atnaujinant pozicijos ir greičio vektorius, kad pirminės pozicijos būtų įrašytos į archyvą. Vėliau atnaujinami dalelių pozicijos vektoriai prie jų pridedant atitinkamus greičio vektorius. Dar vėliau atnaujinamas asmeninis geriausias sprendimas $\boldsymbol{p}_i$ ir kaimynų geriausias sprendimas $\boldsymbol{g}_i$. Tai kaip atliekamas šis geriausių vektorių atnaujinimas dažniausiai yra pagrindinis skirtumas tarp dviejų DSO metodų daugiakriterei optimizacijai. Paprastai $\boldsymbol{p}_i$ atnaujinamas naudojant dominavimo sąvoką.

Tikrinama, ar einamoji dalelės pozicija $\boldsymbol{x}_i$ dominuoja geriausią asmeninį sprendimą $\boldsymbol{p}_i$. Geriausias asmeninis sprendimas gali būti atnaujinamas, jeigu naujas sprendimas stipriai arba silpnai dominuoja senąjį sprendimą. Geriausias kaimynų sprendimas paprastai yra parenkamas iš nedominuojamų taškų archyvo. Parinkimas paprastai vykdomas taip, kad iš archyvo parinkti sprendimai atitiktų Pareto fronto sritis, kurios yra menkai ištirtos. Tai galima padaryti įvairiais būdais. Paprastai vienu ar kitu būdu vertinama, kiek kitų sprendimų yra netoli konkretaus sprendimo. Po to, kai parenkami $\boldsymbol{p}_i$ ir $\boldsymbol{g}_i$, naudojama standartinė greičio vektoriaus atnaujinimo procedūra. Tai kartojama tol, kol netenkinami kokie nors sustojimo kriterijai. Paprastai naudojamas fiksuotas iteracijų skaičius – pagrindinis algoritmo ciklas kartojimas naudotojo nurodytą skaičių kartų.

3 Eksperimentinė procedūra

Kiekvienas algoritmas ar algoritmo konfigūracija buvo tiriama naudojant tą pačią eksperimentinę procedūrą. Sakydami „algoritmo konfigūracija“ galvoje turime optimizacijos metodą su jo parametrais: topologijos aprašymu, c_1 ir c_2 reikšmėmis, mutacijos dažniu ir kt.

Kiekvienam testiniam uždaviniui 30 kartų kartojama tokia procedūra:

1. Algoritmui leidžiama veikti 300 000 funkcijos įvertinimų. Vienas funkcijos įvertinimas susideda iš visų uždavinį sudarančių kriterijų įvertinimo.
2. Surenkami algoritmo rasti taškai sudarantys Pareto fronto aproksimaciją. Šie taškai išsaugomi kietajame diske.
3. Apskaičiuojami indikatorių I_{GD}, I_{IGD}, I_{SP} ir I_N reikšmių vidurkiai ir išsaugojami lentelėse.

Prieš tai aprašyta procedūra kartojama visiems algoritmams aprašytiems disertacijoje. Galiausiai turime 30 bandymų kiekvienai algoritmo ir testinio uždavinio kombinacijai. Rezultatai yra analizuojami tam skirtame disertacijos skyriuje. Čia išsamiai neaptarinėsime testinių uždavinių. Uždavinių pavadinimai ir pagrindinės savybės yra pateikiami

Pavadinimas	Kriterijų sk.	Taškų sk.	Geometrija
UF01	2	1000	Iškila
UF02	2	1000	Iškila
UF03	2	1000	Iškila
UF04	2	1000	Neiškila
UF05	2	21	Diskreti
UF06	2	1000	Neiškila
UF07	2	1000	Neiškila
UF08	3	10000	Neiškila
UF09	3	10000	Neiškila
UF10	3	10000	Neiškila
UF11	5	5000	Neiškila
UF12	5	5000	Neiškila
UF13	5	5000	Iškila

1 lentelė: Testinių uždavinių savybių santrauka

toliau esančioje lentelėje. Čia taškų sk. yra skaičius taškų tikrojo Pareto fronto diskretizacijoje, su kuria yra lyginama algoritmo gauta aproksimacija. Uždaviniai yra paimti iš CEC 2009 daugiakriterės optimizacijos metodų konkurso surinktų Q. Zhang et al. [11].

Testinių uždavinių savybių santrauka yra pateikiama 1 lentelėje. Matome, kad pateikiama daug skirtingų uždavinių, tarp jų 2, 3 ir penkių kriterijų. Uždaviniai yra iškilūs, neiškilūs ir diskretūs (turima omenyje jų Pareto frontai). Dauguma uždavinių yra neiškilūs, todėl jiems netinka tik iškiliems uždaviniams taikomi metodai, pavyzdžiui, svorinio agregavimo metodai. Vienas iš šio uždavinių rinkinio trūkumų yra tas, kad trūksta taškų penkių kriterijų uždavinių Pareto frontų diskretizacijose.

4 Pasiūlyti kokybės indikatoriai

Sprendimų sklaidos matavimo tikslas yra nustatyti, kaip gerai aproksimacijos taškai padengia tikrąjį Pareto frontą. Pagrindinė mintis čia yra užtikrinti, kad aproksimacija algoritmo naudotojui perteikia tikslų Pareto fronto paveikslą. Pirma, reikia aužtikrinti, kad aproksimacijoje netrūktų Pareto fronto dalių. Kitaip tariant, nėra taip, kad dalis Pareto fronto padengta išsamiai, o kitoje dalyje taškų nėra iš viso. Antra, reikia užtikrinti, kad taškai padengtų frontą tolygiai, kad atstumai tarp artimiausių taškų fronto aproksimacijoje būtų vienodi. Tam reikia daryti prielaidą, kad Pareto frontas yra daugiausia

tolydus. Egzistuojantys indikatoriai šios funkcijos neatlieka. Trumpai apžvelgsime egzistuojančius indikatorius, kurie gali būti panaudoti Pareto fronto sklaidos tolygumui matuoti.

Visų pirma yra I_{IGD} indikatorius. Šio indikatoriaus reikšmė bus didesnė, jeigu aproksimacijoje trūksta didelių Pareto fronto regionų. Taip yra dėl to, kad šis indikatorius matuoja vidutinį atstumą tarp tikrojo Pareto fronto diskretizacijos visų taškų ir jiems artimiausių taškų aproksimacijoje. Jeigu aproksimacijoje yra didelių trūkių – regionų nepadengtų taškais, šie atstumai padidės net tuo atveju, jeigu likę aproksimacijos taškai yra arti Pareto fronto. Šio indikatoriaus trūkumas matuojant tolygią taškų sklaidą yra tas, kad indikatoriaus reikšmei įtakos turi ir tai, kaip arti tikrojo Pareto fronto yra aproksimacijos taškai. Dėl to nėra visiškai aišku, kokią įtaką indikatoriaus reikšmei turi vienas ir kitas aspektai. Jeigu indikatoriaus reikšmė yra maža, neaišku, ar tai yra dėl to, kad aproksimacijos taškai arti tikrojo Pareto fronto, ar dėl to, kad nėra trūkių. Dėl šių priežasčių šis indikatorius netinka sprendimų sklaidai matuoti.

Kitas plačiai naudojamas kokybės indikatorius sprendimų sklaidai matuoti yra Schott'o indikatorius I_{SP}. Jis matuoja atstumo tarp kiekvieno aproksimacijos taško ir jam artimiausio kito aproksimacijos taško standartinį nuokrypį. Matuojamas euklidinis atstumas tarp taškų. Iš pirmo žvilgsnio šis metodas atrodo neblogai. Jeigu tarp taškų ir jiems artimiausių kitų taškų yra panašus atstumas, reiškia standartinis nuokrypis bus nedidelis. Taip bus, jeigu taškai yra tolygiai pasklidę palei Pareto frontą. Jeigu taškai bus pasiskirstę netolygiai, tada standartinis nuokrypis bus didesnis. Tačiau šis indikatorius atsižvelgia tik į vieną tolygios sklaidos aspektą. Šis metodas neveiks teisingai, jeigu aproksimacijoje yra trūkių. Jeigu dideli tikrojo Pareto fronto regionai yra nepadengti aproksimacijos, tačiau likę aproksimacijos taškai yra pasiskirstę tolygiai, šis indikatorius rodys, kad aproksimacija gera.

Dėl šių indikatorių trūkumų reikalingi nauji indikatoriai. Šie indikatoriai turės atsižvelgti į intuityvią sklaidos tolygumo sąvoką. Disertacijoje siūlomi du tokie indikatoriai. Abu sukurti atsižvelgti į anksčiau išdėstytus egzistuojančių indikatorių trūkumus.

4.1 Siūlomas indikatorius I_{UNI_1}

Visų pirma aproksimacijos A taškai pakeičiami jų projekcijomis diskretizuotame Pareto fronte PF. Projekcija susideda iš PF taškų kurie yra artimiausi (matuojant Euklido

atstumu) kiekvienam A taškui. Metodas naudotas gauti projekcijai yra pateikiamas (1) lygtyje. Čia, ir visur kitur, nebent pasakyta kitaip, apibrėžiame $\min_{x \in X} f(x)$ kaip x reikšmę, duodančią mažiausią $f(x)$ reikšmę. Taigi (1) lygtyje projekcijos taškas $\boldsymbol{p}_i$ yra taškas diskretizuotame Pareto fronte PF iki kurio atstumas anuo proksimacijos taško $\boldsymbol{a}_i$ yra mažiausias. Čia ir kitur $\| \dots \|$ reiškia euklidinę normą. Kodėl reikia skaičiuoti projekciją, yra aptariama disertacijoje.

$$\boldsymbol{p}_i = \min_{\boldsymbol{r} \in PF} \|\boldsymbol{r} - \boldsymbol{a}_i\| \tag{1}$$

Po to skaičiuojami taškai PF, kuriems p_i yra artimiausias taškas projekcijoje. Kaip tai daroma yra parodoma (2) lygtyje. Gautas skaičius c_i yra lygus taškų skaičiui PF, kuriems p_i yra artimiausias projekcijos taškas.

$$c_i = \left| \left\{ \boldsymbol{r} \mid \boldsymbol{r} \in PF, \boldsymbol{p}_i = \min_{\boldsymbol{a} \in A} \|\boldsymbol{r} - \boldsymbol{a}\| \right\} \right| \tag{2}$$

Galiausiai suskaičiuojama indikatoriaus reikšmė. Skaičiavimas pateikiamas lygtyje (3). Rezultatas yra tiesiog standartinis reikšmių $c_1, \dots, c_{|A|}$ nuokrypis. Čia $\bar{c}$ yra reikšmių $c_1, \dots, c_{|A}$ aritmetinis vidurkis.

$$I_{UNI1}(A) = \sqrt{\frac{1}{|A|} \sum_{i=1}^{|A|} (c_i - \bar{c})^2} \tag{3}$$

Šis indikatorius, autoriaus nuomone, gana tiksliai atspindi intuityvią Pareto fronto tolygaus padengimo sąvoką. Jis lengvai ir greitai apskaičiuojamas. Jo panaudojimas kartu su kitais indikatoriais gali būti naudingas vertinant daugiakriterio optimizavimo algoritmus. Pavyzdžiui, galima naudoti I_{GD} vertinti vidutiniams atstumui iki tikrosios Pareto aibės ir šį indikatoriui vertinti sklaidai.

4.2 Siūlomas indikatorius I_{UNI_2}

Tam tikrais, disertacijoje aptartais atvejais kokybės indikatorius I_{UNI_2} veikia netiksliai. Šie atvejai sprendžiami pritaikius tam tikras toliau aprašytas modifikacijas. Tų modifi-

kacijų kaina yra ta, kad apskaičiuoti indikatoriaus reikšmę tampa brangiau kompiuterio resursų atžvilgiu.

1. Apskaičiuoti aproksimacijos projekciją į Pareto fronto diskretizaciją. Projekcijos skaičiavimas paaiškintas (1) formulėje.
2. Sugrupuoti Pareto fronto diskretizacijos taškus pagal tai, kuris projekcijos taškas jiems artimiausias (naudojamas Euklido atstumas).
3. Projekcijos taškai sujungiami lankais, jeigu jie yra gretimuose regionuose. Regionai yra gretimi, jeigu jų diskretizacijos taškai yra šalia. Kaip tai padaryti, aptarta disertacijoje.
4. Apskaičiuojamas lankų ilgių standartinis nuokrypis. Ši skaitinė reikšmė yra imama kaip kokybės indikatoriaus I_{UNI^2} reikšmė.

5 Pasiūlyti algoritmai

Disertacijoje pasiūlyti du DSO algoritmai, skirti daugiakriterei optimizacijai. Abu algoritmai yra paremti heterogeninių spiečių principu – viename spiečiuje naudojamos skirtingų tipų dalelės.

5.1 Heterogeninis spiečius HMOPSO-I

Pasiūlyto metodo esmė – naudoti kelių skirtingų tipų daleles viename spiečiuje. Šios dalelės parinktos taip, kad būtų atsižvelgta į skirtingas daugiakriterėje optimizacijoje iškylančias problemas. Šios dalelės keisdamosi informacija naudoja tą patį nedominuojamų taškų archyvą. Algoritmai, skirti nedominuojamų archyvų priežiūrai, pateikti disertacijoje. Optimizacijos procedūra kartojama fiksuotą iteracijų skaičių. Metodo veikimas aprašytas 2 algoritme. Vektorių $\boldsymbol{p}_i$ ir $\vec{g}_i$ atnaujinimo taisyklės priklauso nuo konkretaus dalelės tipo. Parametras $w = 0.4$ abiem dalelių tipams, parametrai $\rho_1 = \rho_2 = 1$. Šios reikšmės sutampa su reikšmėmis, naudojamomis kitų DSO algoritmų daugiakriterei optimizacijai. Spiečiuje yra tam tikras procentas abiejų dalelių aprašytų šiame poskyryje.

Tas procentas yra algoritmo parametras. Pozicijos vektoriai $\boldsymbol{x}_i$ yra inicializuojami atsitiktiniais skaičiais iš tinkamų intervalų, atsižvelgian į užduotį. Greičio vektoriai $\boldsymbol{g}_i$ yra inicializuojami nuliais.

Algorithm 2 Heterogeninis DSO metodas HMOPSO-I

for $i \leftarrow 1, n$ **do**
    priskirti dalelę su indeksu i "Sigma" arba "Spread" grupei
    inicializuoti $\boldsymbol{x}_i$ atsitiktinėmis reikšmėmis iš reikšmių intervalo
    inicializuoti $\boldsymbol{v}_i$ nuliais
    $\boldsymbol{p}_i \leftarrow \boldsymbol{x}_i$
end for
while netenkinama sustojimo sąlyga **do**
    atnaujinti nedominuojamų taškų archyvą $\mathbf{A}$
    $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \boldsymbol{v}_i$
    if dalelė i yra "sigma" tipo **then**
        atnaujinti $\boldsymbol{p}_i$ jeigu $\boldsymbol{x}_i \prec \boldsymbol{p}_i$
        atnaujinti $\boldsymbol{g}_i$ parenkant sprendimą iš $\mathbf{A}$ kurio σ reikšmė yra arčiausiai $\boldsymbol{f}(\boldsymbol{x}_i)$ σ reikšmės matuojant Euklido atstumu
    else if dalelė su indeksu i yra "spread" tipo **then**
        atnaujinti $\boldsymbol{p}_i$ jeigu netiesa, kad $\boldsymbol{p}_i \prec \boldsymbol{x}_i$
        atnaujinti $\boldsymbol{g}_i$ parenkant sprendimą iš $\mathbf{A}$ kurio "crowding distance" yra didžiausias
    end if
    $\boldsymbol{v}_i \leftarrow w\boldsymbol{v}_i + \boldsymbol{U}_{(0,\rho_1)}(\boldsymbol{p}_i - \boldsymbol{x}_i) + \boldsymbol{U}_{(0,\rho_2)}(\boldsymbol{g}_i - \boldsymbol{x}_i)$
end while

5.1.1 Dalelė "Sigma"

Ši $\boldsymbol{g}_i$ parinkimo strategija buvo pasiūlyta S. Mostaghim et al. [8]. Šios strategijos veikimo principas yra priskirti tam tikrus reikšmių vektorius visiems sprendimams, esantiems nedominuojamų taškų archyve. Kodėl skaičiuojami būtent tokie vektoriai, yra paaiškinama nurodytame straipsnyje. Šios reikšmės skaičiuojamos naudojant formulę, nurodytą lygtyje (4) dviejų kriterijų atveju ir lygtyje (5) trijų kriterijų atveju. Šis metodas gali būti apibendrintas ir daugiau kriterijų atveju.

$$\sigma = \frac{f_1^2 - f_2^2}{f_1^2 + f_2^2} \tag{4}$$

$$\boldsymbol{\sigma} = \begin{pmatrix} f_1^2 - f_2^2 \\ f_2^2 - f_3^2 \\ f_3^2 - f_1^2 \end{pmatrix} / \left(f_1^2 + f_2^2 + f_3^2\right) \tag{5}$$

Tam, kad šis metodas būtų panaudotas parenkant $\boldsymbol{g}_i$, reikia pirma suskaičiuoti $\boldsymbol{\sigma}_i$ dalelei i, ir tada surasti sprendimą archyve, kurio σ reikšmė būtų artimiausia $\boldsymbol{sigma}_i$ matuojant euklidinį atstumą. Šis sprendimas ir bus $\boldsymbol{g}_i$. Autoriai teigia, kad šis būdas nukreipia daleles tiesiai Pareto fronto link. Asmeninis geriausias sprendimas $\boldsymbol{p}_i$ yra atnaujinamas tada, kai naujas sprendimas dominuoja senąjį $\boldsymbol{p}_i$. Autoriai naudoja mutacijos operatorių, kurio pavidalas yra $x_{i,j} = x_{i,j} + U_{(0,1)}x_{i,j}$, kur $U_{(0,1)}$ yra atsitiktinis skaičius iš tolydaus skirstinio iš intervalo $(0, 1)$, o $x_{i,j}$ yra j-oji i-osios dalelės pozicijos vektoriaus koordinatė. Mutacijos operatorius taikomas su tikimybe 0.05.

5.1.2 Dalelė "Spread"

Ši dalelė sukurta tam, kad užtikrintų, jog Pareto fronto aproksimacijos taškai yra tolygiai pasiskirstę palei tikrąjį Pareto frontą. Daroma prielaida, kad Pareto frontas yra tolydus. Tuo tikslu atlikti du svarbūs pakeitimai atnaujinant dalelės greičio vektorių. Pirma, parenkant $\boldsymbol{g}_i$ sprendimas iš archyvo yra parenkamas taip, kad jo „crowding distance" reikšmė būtų didžiausia. „Crowding distance" matas yra aptartas disertacijoje ir dažnai sutinkamas literatūroje. Antra, atnaujinant $\boldsymbol{p}_i$ naujas sprendimas pakeičia senąjį, jeigu naujas sprendimas nėra dominuojamas senojo. Iš eksperimentų rezultatų matyti, kad tai yra svarbu norint užtikrinti taškų pasiskirstymą palei tikrąjį Pareto frontą. Visi kiti parametrai yra tokie patys kaip ir dalelės „Sigma" atveju.

5.2 Heterogeninis spiečius HMOPSO-II

Pareto fronto aproksimacijas vertiname remdamiesi dviem pagrindinėmis savybėmis: kaip arti tikrojo Pareto fronto yra aproksimacijos taškai ir kaip tolygiai taškai padengia Pareto frontą. Norima, kad aproksimacijoje matytųsi kuo tikslesnis tikrojo Pareto fronto pavidalas. Paprastai algoritmų kūrėjai atsižvelgia į abu aspektus. Tačiau šie aspektai praktikoje dažnai vienas kitam prieštarauja. Norint priartėti prie Pareto fronto, reikia vykdyti lokalią paiešką. Norint padengti Pareto frontą tolygiai, reikia vykdyti globa-

lią paiešką. Kai kurie metodai bando priartėti kuo arčiau Pareto fronto, tačiau dėl to įstringa lokaliuose minimumuose. Kiti bando tolygiai padengti Pareto frontą aproksimacijos taškais, tačiau tokiu atveju aukojamas lokalios paieškos išsamumas. Sėkmingas algoritmas turi suderinti šiuos du optimizacijos aspektus. Pasiūlytas metodas naudoja kelis dalelių tipus, kurių kiekvienas yra sukurtas atsižvelgti į tam tikrą optimizacijos aspektą. Dalelės keičiasi informacija per bendrą nedominuojamų taškų archyvą. Toks spiečius vadinamas heterogeniniu. Dalelių tipai keičiami dinamiškai algoritmo veikimo metu, jeigu dalelė nepagerina sprendimo per tam tikrą skaičių algoritmo iteracijų. Taip spiečius prisitaiko prie uždavinio, naudodamas dalelių tipus tinkamiausius tam uždaviniui. Taip pat pasiūlomas dalelės tipas užtikrinantis tolygų Pareto fronto padengimą aproksimacijos taškais. Spiečius palyginamas su kitais literatūroje aprašytais DSO variantais naudojant kelis kokybės indikatorius. Toliau aprašomi naudojami dalelių tipai ir pats algoritmas.

5.2.1 Dalelė “Sigma”

Ši dalelė yra tokia pati kaip ir dalelė „Sigma“ anksčiau aprašytame metode.

5.2.2 Dalelė “Closest”

Ši dalelė veikia panašiai kaip dalelė „Sigma“, vienintelis skirtumas yra tas, kad $\boldsymbol{g}_i$ yra parenkamas iš archyvo taip, kad būtų artimiausias dalelės i pozicijai $\boldsymbol{x}_i$ matuojant euklidinį atstumą. Visi kiti parametrai ir logika yra ta pati kaip ir dalelės „Sigma“.

5.2.3 Dalelė “Spread”

Ši dalelė yra tokia pati kaip ir dalelė „Spread“ anksčiau aprašytame metode.

5.2.4 Heterogeninis spiečius

Pasiūlytame metode dalelės tipas yra parenkamas dinamiškai. Pradžioje dalelių tipai yra priskiriami atsitiktinai su vienodomis tikimybėmis. Taip dalelė įgyja vieną iš trijų tipų. Optimizavimo metu, jeigu dalelės $\boldsymbol{p}_i$ nesikeičia, nurodytą skaičių iteracijų dalelės

tipas yra pakeičiamas atsitiktinai, ir jos pozicija bei kiti atributai yra grąžinami į pradines reikšmes. Visos dalelės naudoja tą patį nedominuojamų taškų archyvą, kuris yra vienas iš būdų dalelėms keistis duomenimis. Panašūs dalelių spiečiai tik vieno kriterijaus optimizacijai buvo aprašyti Andries P. Engelbrecht [4] ir Marco Antonio Montes de Oca et al. [3]. Algoritmas pseudokodu pateiktas 3 algoritme.

Algorithm 3 Heterogeninis DSO metodas HMOPSO-II

for $i \leftarrow 1, n$ **do**
    priskirti dalelę su indeksu i vienai iš grupių "sigma", "closest" arba "spread" atsitiktinai
    inicializuoti $\boldsymbol{x}_i$ atsitiktinėmis reikšmėmis iš leidžiamų reikšmių intervalo
    inicializuoti $\boldsymbol{v}_i$ nuliais
    $\boldsymbol{p}_i \leftarrow \boldsymbol{x}_i$
end for
while netenkinamos susitojimo sąlygos **do**
    atnaujinti nedominuojamų taškų archyvą $\mathbf{A}$
    $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \boldsymbol{v}_i$
    if jei dalelė i yra "sigma" tipo **then**
        atnaujinti $\boldsymbol{p}_i$ jeigu $\boldsymbol{x}_i \prec \boldsymbol{p}_i$
        atnaujinti $\boldsymbol{g}_i$ parenkant sprendimą iš $\mathbf{A}$ kurio σ reikšmė yra arčiausiai $\boldsymbol{f}(\boldsymbol{x}_i)$ σ reikšmės matuojant Euklidinį atstumą
    else if dalelė i yra "closest" tipo **then**
        atnaujinti $\boldsymbol{p}_i$ jeigu $\boldsymbol{x}_i \prec \boldsymbol{p}_i$
        atnaujinti $\boldsymbol{g}_i$ parenkant sprendimą iš $\mathbf{A}$ kuris yra arčiausiai $\boldsymbol{x}_i$ matuojant Euklidiniu atstumu
    else if dalelė i yra "spread" tipo **then**
        atnaujinti $\boldsymbol{p}_i$ jeigu $\boldsymbol{p}_i \prec \boldsymbol{x}_i$
        atnaujinti $\boldsymbol{g}_i$ parenkant sprendimą iš $\mathbf{A}$ kurio crowding distance didžiausias
    end if
    if dalelės i $\boldsymbol{p}_i$ nebuvo atnaujintas nurodytą skaičių iteracijų **then**
        priskirti dalelę i vienam iš "sigma", "closest" arba "spread" tipų atsitiktinai
    end if
    $\boldsymbol{v}_i \leftarrow w\boldsymbol{v}_i + \boldsymbol{U}_{(0,\rho_1)}(\boldsymbol{p}_i - \boldsymbol{x}_i) + \boldsymbol{U}_{(0,\rho_2)}(\boldsymbol{g}_i - \boldsymbol{x}_i)$
end while

6 Išvados

Šiame darbe buvo atlikta egiztuojančių DSO metodų, skirtų daugiakriterei optimizacijai, eksperimentinė analizė. Buvo pasiūlyti du nauji DSO metodai daugiakriterei optimizaci-

jai ir eksperimentiškai palyginti su egzistuojančiais. Literatūroje aprašyti metodai buvo suklasifikuoti naudojant naują sistemą, kuri remiasi klasifikavimu pagal algoritmuose panaudotus sprendimus ir idėjas. Dėl egzistuojančių kokybės indikatorių trūkumų, vertinant Pareto fronto aproksimacijos tolygumą padengiant tikrąjį Pareto frontą, buvo pasiūlyti du nauji kokybės indikatoriai. Buvo sukurtas programinės įrangos karkasas, skirtas naujų DSO algoritmų daugiakriterei optimizacijai kurti. Iš atliktų tyrimų galima daryti tokias išvadas:

- Pasiūlyti daugiakriterės optimizacijos kokybės indikatoriai gali būti naudojami matuoti, kaip tolygiai algoritmų rastos aproksimacijos padengia tikrąjį Pareto frontą. Šie (ar panašūs) indikatoriai reikalingi, nes egzistuojantys indikatoriai netiksliai atspindi intuityvią padengimo tolygumo sąvoką, kuri aptariama ketvirtame skyriuje. Egzistuojančių indikatorių trūkumai aprašyti disertacijoje. Šios problemos yra sprendžiamos pasiūlytuose indikatoriuose.
- Pageidautina, kad DSO metodai daugiakriterei optimizacijai (kaip ir visi optimizacijos metodai) gerai veiktų su daug skirtingų uždavinių klasių, be poreikio pritaikyti algoritmo parametrų konkrečiai uždavinių klasei. Iš atliktų eksperimentų matyti, kad taikant egzistuojančius metodus naudojamos taisyklės gerai veikia su tam tikromis uždavinių klasėmis. Naudojant daleles su skirtingomis geriausių dalelių parinkimo taisyklėmis (vadinamuosius heterogeninius spiečius), galima sukurti metodus, kurie gerai veikia su keliomis uždavinių klasėmis. Tokiais atvejais spiečius naudoja daleles, kurios geriau tinka vienai ar kitai uždavinių klasei. Iš eksperimentinių tyrimų matyti, kad pasiūlyti heterogeniniai spiečiai gerai veikia su didele dalimi testinių uždavinių, palyginti su egzistuojančiais metodais.
- Literatūroje aprašyta daug atvejų, kai mutacijos operatorių panaudojimas pagerina optimizavimo efektyvumą optimizuojant vieno kriterijaus funkcijas. Nėra daug tyrimų, kuriuose būtų tiriamas mutacijos operatorių panaudojimas, kai daugiakriterei optimizacijai naudojama DSO. Dėl didelio skaičiaus disertacijoje ištirtų metodų (tiek naudojančių mutaciją, tiek ne) ir iš eksperimentų su jais rezultatų galima daryti išvadas, kad mutacijos panaudojimas gerokai padidina metodų efektyvumą.
- Uždavinių dekompozicija paremti metodai gerai veikia optimizuojant uždavinius su netolygiais Pareto frontais. Taip gali būti dėl to, kad šie metodai nedaro prielaidų, kad Pareto frontas yra tolydus. Dauguma kitų metodų remiasi šiomis prielaidomis.

- „Vector Evaluated“ DSO metodai buvo anksti pasiūlyti kaip būdas pritaikyti vieno kriterijaus DSO metodus spręsti uždaviniams su keliais kriterijais. „Vector Evaluated“ metodai naudoja kelias populiacijas. Kiekvienoje atskiroje populiacijoje individai optimizuoja vieną iš kriterijų. Populiacijos gali keistis informacija apie gerus sprendimus. Paprasčiausiu atveju viena populiacija ima kitos populiacijos geriausią sprendimą kaip vektorių $\boldsymbol{g}$ (geriausią kaimynų sprendimą). Kadangi populiacijos optimizuoja skirtingus kriterijus, tai leidžia populiacijoms optimizuoti kelis kriterijus vienu metu. Iš trečiame skyriuje aptartų eksperimentų rezultatų matyti, kad „Vector Evaluated“ DSO metodai veikia prastai, palyginti su kitais DSO metodais. Iš kelių tirtų „Vector Evaluated“ metodų nė vienas net nepriartėja prie geresnių metodų savo efektyvumu

Kiti rašant disertaciją pasiekti rezultatai

- Buvo sukurta programinė įranga, leidžianti naudoti skirtingus DSO metodus ir matuoti jų efektyvumą su daug testinių uždavinių ir kokybės indikatorių. Šis programinis karkasas yra sukurtas atsižvelgiant į poreikius vykdyti eksperimentus superkompiuteriuose. Karkasą sudarančios bibliotekos yra viešai prieinamos, turi atvirą kodą. Bibliotekos modulinis dizainas leidžia naudotojui greitai ir patogiai kurti naujus dalelių tipus ir spiečiaus topologijas. Naudojant su bibliotekomis pateikiamus testinius uždavinius ir kokybės indikatorius, naudotojas gali atlikti empirinius tyrimus.
- Atlikta egzistuojančių DSO metodų daugiakriterei optimizacijai sisteminė apžvalga ir klasifikacija. Skirtingai nei ankstesnės klasifikacijos schemos, kurios remiasi klasifikavimu bendrais bruožais, pasiūlyta sistema atsižvelgia į kelis svarbius faktorius: pagrindines metodo idėjas, kaip užtikrinamas sprendinių tolygus pasiskirstymas ir kokie naudojami mutacijos operatoriai. Šie trys kriterijai naudojami klasifikuojant literatūroje aprašytus metodus.

7 Trumpai apie autorių

Vytautas Jančauskas 2009 m. Vilnius universitete baigė informatikos bakalaurą, o 2011 m. – informatikos magistrą. Informatikos magistrą baigė geriausiai kurse, už akademinius pasiekimus jam suteiktas Magna cum laude diplomas. Nuo 2011 m. iki 2015 m. – Vilnius universiteto Matematikos ir informatikos instituto doktorantas. Nuo 2009 m. dirbo Vilniaus universiteto Matematikos ir informatikos fakultete laborantu, nuo 2011 m. – asistentu, o nuo 2015 m. – lektoriumi. Nuo 2015 m. rengė ir skaitė kompiuterių tinklų kursą informatikos, bioinformatikos ir programų sistemų bakalauro studentams. Vedė kompiuterių tinklų, C programavimo, kompiuterių architektūros, operacinių sistemų pratybas informatikos, bioninformatikos ir programų sistemų bakalauro studentams.

Literatūra

[1] Jared L Cohon. Multicriteria programming: Brief review and application. *Design optimization*, pages 163–191, 1985.

[2] Jared L Cohon. *Multiobjective programming and planning*. Courier Corporation, 2013.

[3] Marco Antonio Montes de Oca, Jorge Peña, Thomas Stutzle, Carlo Pinciroli, and Marco Dorigo. Heterogeneous particle swarm optimizers. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 698–705. IEEE, 2009.

[4] Andries P Engelbrecht. Heterogeneous particle swarm optimization. In *Swarm Intelligence*, pages 191–202. Springer, 2010.

[5] F. Heppner and U. Grenander. A stochastic nonlinear model for coordinated bird flocks. In E. Krasner, editor, *The ubiquity of chaos*, pages 233–238. AAAS Publications, 1990.

[6] Russell C. Eberhart James Kennedy. Particle swarm optimization. *IEEE International Conference on Neural Networks, Proceedings*, 4:1942 – 1948, 1995.

[7] Juhani Koski. Multicriterion optimization in structural design. Technical report, DTIC Document, 1981.

[8] Sanaz Mostaghim and Jürgen Teich. Strategies for finding good local guides in multi-objective particle swarm optimization (mopso). In *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, pages 26–33. IEEE, 2003.

[9] Margarita Reyes-Sierra and CA Coello Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International journal of computational intelligence research*, 2(3):287–308, 2006.

[10] Ralph E Steuer. *Multiple criteria optimization: theory, computation, and applications*. Wiley, 1986.

[11] Qingfu Zhang, Aimin Zhou, Shizheng Zhao, Ponnuthurai Nagaratnam Suganthan, Wudong Liu, and Santosh Tiwari. Multiobjective optimization test instances for the cec 2009 special session and competition. *University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report*, 2008.

Summary

Multi-objective optimization problems are the problems, where several (possibly contradictory) objectives have to be optimized at the same time. This means a single solution will be evaluated with regards to several different criteria. Most problems in engineering, economics and other areas of human activity are of this nature. For example, one might want to increase the durability of a manufactured part, but also to decrease it's price. It is easy to see how those two things may be in contradiction to one another.Most of the time, no single solution will be sufficient for the problems like these. Instead, methods that attempt to solve these problems will look for a Pareto frontier. A Pareto frontier is a set of points in the objective space, where no point dominates any other point. The concept of dominance will be discussed later. For the time being, it is sufficient to say that solutions corresponding to points in such a set cannot be improved in terms of one objective without sacrificing some other objective.

There are a lot of methods for solving or attempting to solve the multi-objective problems. It is possible to solve such problems analytically only for a very small subset of them. If nothing or little is known about a problem or if it is non-convex, exact methods for finding solutions generally do not exist. In such cases, various heuristic methods are used. Very often these are population based meta-heuristic methods. It is often the case, that methods created for single objective optimization are adapted to work with multi-objective problems. A simple way of doing so is to aggregate objectives to a single objective using weights. That way, a problem with several objectives is converted into several single objective problems by varying those weights. This will then require running the method in question on each one of these single objective problems and storing the results. It would be ideal that one could obtain an approximation of the complete Pareto frontier in a single run of the algorithm. The nature of the multi-objective problems compared to single objective ones raises several complications when using this approach. First of all, a set of solutions, but not a single solution will have to be found. This is often solved by using a non-dominated point archive that maintains such sets using dominance criteria for accepting or rejecting a solution. Solutions are accepted into the archive if they are non-dominated by any solution already in the archive. Second of all, diversity of solutions has to be maintained throughout. We want the solutions found by the method to cover the whole of the actual Pareto frontier. This is contrary to single objective optimisation where the population will converge to a point. Heuristic algorithms will have to be changed so, that population converges to a set instead. Popular

meta-heuristic methods used with multi-objective problems are genetic algorithms that work by selecting best individuals from a population and creating new solutions based on them. To work with multi-objective problems they are modified to take the above mentioned into account.

Particle Swarm Optimisation (PSO) is a population based method inspired by social behaviour of swarming animals like birds or fish. Namely, swarming (in birds and others) or schooling (in fish) behaviour is taken as an inspiration. Particles are points in the solution space that move according to the movement and velocity update rules. These rules are designed so, that particles move towards the best solution they found so far and also towards the best solution their neighbours have found. This can be viewed as particles using personal and social experience and combining them (with random weighting) to come up with new solutions. Particle Swarm Optimisation was adapted to solve multi-objective optimisation problems in ways similar to the ones used in Genetic Algorithms. There is active ongoing research into Multi-Objective Particle Swarm Optimisation (MOPSO) algorithms. Many algorithms and their variations have been proposed in literature.

There are two important issues to consider when it comes to our understanding of using PSO for multi-objective optimisation. First of all, it is unclear which of the existing approaches is the most promising or deserves further investigation. The existing methods are often compared only to one or two other approaches. Furthermore, they are often compared only using a couple of test problems. As such, there is no way of telling which method (or group of methods) is the most promising. In particular, it would be interesting to see which choices in optimisation method design lead to successful methods. The second issue has to do with comparing two multi-objective optimisation algorithms. There are no agreed procedures to compare two solutions of a multi-objective problem. In the case of the single objective optimisation it is simple: we can compare the fitness function values obtained by the two algorithms, we can also compare the time in function evaluations (since they are usually the time limiting factor) or algorithm iterations it takes for the algorithm to find the answer to a given precision. We can also compare the percentage of times the method finds the global minimum versus the percentage of times it does not. Each of these aspects is straightforward to measure and interpret. It is not yet the case for the multi-objective optimisation. Here solutions are sets and comparing of two sets is difficult. It usually depends on what the user of the method perceives to be a good solution.

Aim and Object of the Study

- Overview performance indicators meant to evaluate performance of non-convex multi-objective optimization methods. These indicators are designed to measure aspects of the optimizer performance. The fit between the Pareto frontier of a problem and an approximation of that Pareto frontier produced by the optimizer is usually measured. The fit can be measured in several aspects, for example, the average distance of the points in the approximation to the Pareto frontier can be calculated. Or how uniformly the points cover the Pareto frontier. If the existing performance indicators seem inadequate to the task, new ones are to be proposed.

- Review MOPSO methods described in literature. Examine what design choices and underlying ideas are commonly used. Based on this information propose a new classification scheme if existing ones are not adequate to account for the findings of the review.

- There do not exist readily available implementations of many of the MOPSO methods that are described in the literature. To aid in this, develop a software framework that allows to implement and evaluate these methods. This will consist of implementations of the methods themselves, test problems and performance indicators. These can then be used together to evaluate the methods experimentally. This has to be done in a convenient and modular way, so as new methods can be implemented easily when needed.

- Using the aforementioned software framework, experimentally evaluate existing MOPSO methods. The purpose of this is to do an experimental comparative analysis. This will allow us to judge which methods are most suited for which kinds of problems. That's because the properties of the problems and the design choices behind the algorithms are known. Analyze the results to determine the types of MOPSO methods that are the most promising for certain types of multi-objective problems.

- Propose new MOPSO methods using the information obtained by studying and evaluating existing methods and what works and does not work in them. The data for this subtask will be obtained through the work done during the previous subtasks.

Scientific Novelty and Results

- Two new performance indicators were introduced. These are designed to measure how uniformly does a Pareto set approximation cover the real Pareto frontier.
- Two new MOPSO algorithms were proposed. Both use heterogeneous swarm approach in order to improve MOPSO performance. In heterogeneous swarms particles with different velocity and position update rules and other characteristics share the same nondominated point archive to exchange the information. If the particles are designed to accent different aspects of MOPSO performance ,they tend to compensate each others shortcomming and improve performance in terms of all aspects concerned.
- An overview of existing MOPSO algorithms was done. The methods were systematized and classified using several different schemes. Citations trees were constructed.
- Methodologies for comparing different multi-objective optimisation algorithms were proposed. Methodologies consist of recommendations for the experimental procedure, performance indicators and test problems. Methods for analysing the collected performance data were proposed.
- Existing MOPSO algorithms are extensively tested and evaluated relatively to each other. Most of the existing MOPSO algorithms are covered in the study as far as their descriptions are published in scientific literature.
- A large framework was developed for the Python programming language. It allows the user to use tens of different particle types, several different swarm topologies and has tens of test problems to benchmark new algorithms.

Statements to be Defended

- Proposed multi-objective optimisation performance indicators can be used to measure the uniformity of the solution spread in Pareto front approximations. These (or similar) indicators are necessary, because existing performance indicators suffer from the problems that are explained in chapter 4. Because of those problems, existing indicators that are described in chapter 2, do not accurately capture the

intuitive notion of the uniform coverage. The problems are solved by the proposed indicators by taking into careful consideration what it means to cover Pareto frontier uniformaly with a discrete approximation of that frontier.

- It can be seen from the experimental evaluations that proposed heterogeneous multi-objective optimization algorithms perform well over a wide selection of performance indicators and test problems compared to the existing methods.

- It has been known for a long time that the use of mutation operators can improve the performance of single objective PSOs (mutation is not used in original PSO). However, comparative studies that contrast the methods that use mutation operators and those that do not,have not been performed in the field of multi-objective PSO. Because of the large number of MOPSO methods that were surveyed in this research both with and without mutation, conclusions about it's use can be made. It can be seen from the data that mutation can significantly improve the performance of MOPSO methods. With all test problem, the methods that don't use mutation end up with the highest values of I_{IGD} and I_{GD} indicators by orders of magnitude. The only exception are the decomposition based methods, that do comparatively well.

- Decomposition based approaches work well on problems with discontinuous Pareto frontiers. This may be due to the fact that they do not make assumptions about the Pareto frontier being continuous the way most MOPSO approaches do.

- From the experimental results that are gathered in chapter 3 it can be seen that vector evaluated MOPSO methods do not work well compared to other MOPSO approaches. Several vector evaluated approaches have been tested and none of them come close in terms of results to other approaches.

Publications of the Author

1. Jančauskas, Vytautas. Empirical Study of Particle Swarm Optimization Mutation Operators. Baltic Journal of Modern Computing, Vol. 2 (2014), No.4, pp. 199 – 214.

2. Jančauskas, Vytautas. Heterogeneous Multi-Objective Particle Swarm Optimiser with a Spread Particle. International Journal of Research Studies in Science, Engi-

neering and Technology, Vol. 2, Issue 9, September 2015, pp. 30 – 49. ISSN 2349 – 4751.

3. Jančauskas, Vytautas, Aušra Mackute-Varoneckiene, Audrius Varoneckas, and Antanas Žilinskas. "On the multi-objective optimization aided drawing of connectors for graphs related to business process management." In Information and Software Technologies, pp. 87–100. Springer Berlin Heidelberg, 2012.

4. Jančauskas, Vytautas, Giedrius Kaukas, Antanas Žilinskas, and Julius Žilinskas. "On Multi-Objective Optimization Aided Visualization of Graphs Related to Business Process Diagrams." In DB&Local Proceedings, pp. 71–80. 2012.

5. Jančauskas, Vytautas. "Optimizing neighbourhood distances for a variant of fully-informed particle swarm algorithm." In Nature Inspired Cooperative Strategies for Optimization (NICSO 2013), pp. 217–229. Springer International Publishing, 2014.

About the Author

Vytautas Jančauskas got his bachelors degree in Computer Science from Vilnius University in 2009. In 2011 he got his masters degree in Computer Science from Vilnius University. He graduated magna cum laude – an honor given for exceptional academic achievements. From 2011 to 2015 he was a PhD student at the Institute of Mathematics and Informatics, Vilnius University. From 2009 he worked a lab assistant at the Faculty of Mathematics and Informatics, Vilnius University. From 2015 he worked as a lecturer at the same university. From 2015 he prepared and read the Computer Networks course to computer science, bioinformatics and software engineering students. He also suppervised tutorial sessions in computer networks, C programming, computer architecture and operating systems to computer science, bioinformatics and software engineering students.

Vytautas Jančauskas

DALELIŲ SPIEČIŲ OPTIMIZAVIMO ALGORITMŲ TAIKYMO DAUGIAKRITERIAMS UŽDAVINIAMS EFEKTYVUMO TYRIMAS

Daktaro disertacija

Fiziniai mokslai (P000)
Informatika (09P)
Redaktorė Vita Markevičienė

Vytautas Jančauskas

EVALUATING THE PERFORMANCE OF MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION ALGORITHMS

Doctoral Dissertation

Physical Sciences (P000)
Informatics (09P)
Editor Vilija Ambrasienė