VYTAUTO DIDŽIOJO UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS INSTITUTAS

Remigijus PAULAVIČIUS

GLOBALUSIS OPTIMIZAVIMAS SU SIMPLEKSINIAIS POSRIČIAIS

Daktaro disertacija

Fiziniai mokslai (P 000)
Informatika (09 P)
Informatika, sistemų teorija (P 175)

Vilnius, 2010

Disertacija rengta 2006–2010 metais Matematikos ir informatikos institute.

Darbo vadovas

dr. (HP) Julius ŽILINSKAS (Matematikos ir informatikos institutas, fiziniai mokslai, informatika – 09 P)

Reziumė

Disertacijoje nagrinėjami simpleksiniai šakų ir rėžių algoritmai su agreguotais Lipšico rėžiais.

Pirmiausiai dviejų matmenų atvejui pasiūlomas normų atžvilgiu patikslintas įprastas Lipšico rėžis, naudojantis pirmą ir begalinę normas. Daugiamačiu atveju šis rėžis papildomas Euklidine norma. Pateikiamas Pijavskij tipo Lipšico rėžio apskaičiavimas simpleksuose sprendžiant tiesinių lygčių sistemas. Pasiūlomas naujas rėžis, pagrįstas apibrėžtinės daugiamatės sferos spinduliu. Galiausiai sudaromi agreguotieji Lipšico rėžiai iš skirtingo tipo rėžių su įvairiomis normomis. Eksperimentinių tyrimų rezultatai rodo, kad naudojant siūlomą simpleksinį šakų ir rėžių algoritmą su agreguotuoju rėžiu ir viršūnių patikrinimu beveik visiems testo uždaviniams gaunami geresni rezultatai, lyginant su geriausiu literatūroje apžvelgtu šakų ir rėžių Lipšico optimizavimo algoritmu.

Vėliau pasiūlomi lygiagretieji simpleksiniai šakų ir rėžių algoritmai skirti bendrosios (OpenMP) ir paskirstytosios (MPI) atminties kompiuteriams. Eksperimentiškai tiriama, kaip viršūnių patikrinimas įtakoja bendrosios atminties kompiuteriams skirto algoritmo lygiagretinimo efektyvumą. Taip pat tiriama, kaip geriausios surastos funkcijos reikšmės apsikeitimas tarp procesorių paveikia paskirstytosios atminties kompiuteriams skirto lygiagrečiojo algoritmo lygiagretinimo efektyvumą ir funkcijų skaičiavimų kiekį.

Darbo pabaigoje eksperimentiškai tiriama, kaip paieškos strategijos įtakoja nuosekliųjų ir lygiagrečiųjų šakų ir rėžių algoritmų rezultatus. Nuosekliojo algoritmo efektyvumas tiriamas pagal funkcijų skaičiavimo kiekio, vykdymo laiko, ištirtų posričių skaičiaus, maksimalaus kandidatų sąrašo ir santykio, parodančio kaip greitai optimizavimo procese surandamas sprendinys, kriterijus. Nustatoma, kaip kinta paskirstytosios atminties kompiuteriams skirto lygiagrečiojo algoritmo lygiagretinimo efektyvumas ir funkcijų skaičiavimų kiekis priklausomai nuo naudojamos paieškos strategijos. Eksperimentiniai tyrimai rodo, kad nėra vienos strategijos, pagal kurią būtų gaunami geriausi rezultatai pagal visus kriterijus, todėl paieškos strategiją tikslinga pasirinkti priklausomai nuo sprendžiamo optimizavimo uždavinio skaičiavimo resursų.

Abstract

The dissertation analyzes branch and bound algorithms with simplicial partitions and aggregated Lipschitz bounds.

First of all corrected trivial Lipschitz bound with the first and infinity norms is proposed for two dimensional case. In multidimensional case this bound is appended with Euclidean norm. Then calculation of Piyavskii type Lipschitz bound over simplices is proposed solving system of linear equations. The new Lipschitz bound based on the radius of the circumscribed multidimensional sphere is proposed. Finally aggregate Lipschitz bounds of a few different Lipschitz bound types with various norms are created. The results of experiments show that proposed simplicial branch and bound algorithm with aggregate bound and vertex verification is often better than the best in literature reviewed branch and bound algorithm for Lipschitz optimization.

Then parallel simplex based branch and bound algorithms with aggregate bound are proposed for shared (OpenMP) and distributed (MPI) memory computers. Efficiency of the OpenMP version with and without new vertex point verification is investigated. Efficiency and the number of function evaluations of MPI version with and without interchange of the best currently found function values is also investigated.

Finally, the influence of selection strategies to sequential and parallel branch and bound algorithms is experimentally investigated. Performance of the sequential algorithm with different selection strategies is investigated using several criteria: numbers of function evaluations, execution time, the total number of simplices, maximal size of candidate list and the ratio, showing the progress of search to locate the global solution. Parallel MPI version is evaluated using efficiency of parallelization and the number of function evaluations criteria. The results of experiments show, that none of selection strategies is the best for all criteria, therefore it is advisable to choose the selection strategy subject to optimization problem computational resources.

Žymėjimai ir santrumpos

n	funkcijos kintamųjų skaičius;
$\mathbb{R}^n$	n-matė Euklidinė erdvė;
$\mathbf{D}$	leistinoji sritis;
ε	paklaida;
x, y, z	tyrimo kintamieji (vektoriai);
$f(x)$	tikslo funkcija;
$\nabla f(x)$	tikslo funkcijos $f(x)$ gradientas;
$F(x)$	viršutinio rėžio funkcija;
f^*	globalusis optimumas;
$f(x_{opt})$	ε-globalusis optimumas;
x^*	globaliojo optimumo taškas;
x_{opt}	ε-globaliojo optimumo taškas;
$\mathbf{S}$	sprendinys (posritis, optimumo taškas);
$\mathbf{T}$	baigtinė aibė, kurios taškuose apskaičiuojamos tikslo funkcijos reikšmės;
$\mathbb{I}$	kandidatų aibė;
$\lvert\mathbb{I}\rvert$	kandidatų aibės elementų skaičius;
$\mathbf{I}$	leistinosios srities posritis;
v	posričio viršūnė;
$V(\mathbf{I})$	posričio viršūnių aibė;
$\mathbf{C}$	daugiamatis rutulys;
LB	apatinis maksimumo rėžis;
UB	viršutinis maksimumo rėžis;
R	daugiamatės apibrėžtinės sferos spindulys;
D	determinantas;
p	procesorių skaičius; normos indeksas;
s_p	spartinimas;
e_p	lygiagretinimo efektyvumas;
pe_p	pseudo efektyvumas;
$\tilde{u}$	statistinis įvertis;
l_q	realiųjų skaičių sekų $x = (x_n)$ aibė, kurioms eilutė $\sum_n \lvert x_n\rvert^q$ konverguoja;
$\lVert x\rVert_q$	n-mačio vektoriaus $l_q, (q \geq 1)$ norma;

$\|x-y\|_q$	atstumas tarp dviejų vektorių;
L_p	Lipšico konstanta įvertinta pagal l_p normą;
K	išvestinės Lipšico konstanta;
μ, μ_1, μ_2	paprastai apskaičiuojami μ tipo Lipšico rėžiai;
φ	sudėtingai apskaičiuojamas Pijavskij tipo Lipšico rėžis;
ψ	Lipšico rėžis, pagrįstas daugiamatės apibrėžtinės sferos spinduliu R;
$\mu_2^{1,2,\infty}$	μ_2 tipo Lipšico rėžis su l_1, l_2 ir l_∞ normomis;
$\varphi^1\psi^2\mu_2^{2,\infty}$	agreguotas Lipšico rėžis sudarytas iš φ, ψ ir μ_2 tipo rėžių su įvairiomis normomis;
$\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$	agreguotas Lipšico rėžis sudarytas iš φ, ψ ir μ_2 tipo rėžių su įvairiomis normomis ir viršūnių patikrinimu;
r_{ψ^2/μ_2^2}	santykis, įvertinantis rėžio ψ^2 gerumą μ_2^2 rėžio atžvilgiu;
$r(f^*)$	santykis, parodantis kaip greitai optimizavimo procese surandamas globalusis sprendinys;
$f.k.$	funkcijos skaičiavimų kiekis;
$t(s)$	optimizavimo laikas;
PS	ištirtų posričių skaičius;
MKS	maksimalus kandidatų sąrašas;

Turinys

Padėka

Nuoširdžiai dėkoju moksliniam darbo vadovui dr. Juliui Žilinskui už nuoširdžias, atsakingas bei vertingas mokslines konsultacijas, už patarimus ir pasiūlymus bei nuolatinį skatinimą tobulėti; Matematikos ir informatikos instituto direktoriui prof. habil. dr. Gintautui Dzemydai už visapusišką paramą studijuojant doktorantūroje; disertacijos recenzentams prof. habil. dr. Antanui Žilinskui bei prof. habil. dr. Leonidui Sakalauskui pateikusiems vertingų pastabų ir patarimų; MII sistemų analizės skyriaus darbuotojams bei doktorantams už bendradarbiavimą, pagalbą ir supratimą; kolegoms iš Vilniaus pedagoginio universiteto Algebros ir statistikos katedros už pagalbą ir moralinį palaikymą; Lietuvos valstybiniam mokslo ir studijų fondui už suteiktą finansinę paramą disertacijos rengimo metu; artimiesiems ir draugams už jų kantrybę, palaikymą, supratingumą ir meilę.

Įvadas

Tyrimų sritis

Globaliojo optimizavimo metodai naudojami skaičiuojamosios chemijos ir biologijos, biomedicinos, operacijų tyrimo, ekonomikos, inžinerinio projektavimo ir valdymo bei daugelio kitų inžinerijos ir taikomųjų mokslų uždavinių sprendimui. Viena iš labiausiai tyrinėjamų globaliojo optimizavimo sričių yra Lipšico optimizavimas, kuris remiasi prielaida, kad tikslo funkcija tenkina Lipšico sąlygą. Daugiamačiai Lipšico optimizavimo metodai tiek teorine, tiek ir praktine prasme yra gerokai mažiau ištirti nei vienmačiai. Šie metodai dažniausiai naudoja stačiakampius posričius ir paprastai apskaičiuojamus Lipšico rėžius, o simpleksiniams posričiams ir rėžių gerinimui skirtas mažesnis dėmesys. Todėl šio darbo tyrimų sritis yra daugiamačiai simpleksiniai Lipšico optimizavimo algoritmai, naudojantys agreguotuosius Lipšico rėžius su įvairiomis normomis.

Darbo aktualumas

Sparčiai besivystančios informacinės technologijos ir tobulėjanti kompiuterinė technika leidžia spręsti vis sudėtingesnius optimizavimo uždavinius, kurių išsprendžiamumas dar visai neseniai buvo sunkiai arba iš viso neįmanomas.

Globaliojo optimizavimo metodai, pagrįsti Lipšico rėžių apskaičiavimu, yra pakankamai plačiai išnagrinėti ir yra plačiai taikomi įvairių optimizavimo uždavinių sprendimui. Šių metodų pritaikomumą lemia tai, kad jie pateikia tą patį atsakymą kiekvieną kartą, skirtingai negu stochastiniai optimizavimo metodai. Lipšico metodai garantuoja surasto sprendinio tikslumą per baigtinį (gali būti ir ilgas) laiką, priešingai negu tiesioginės paieškos ar stochastiniai metodai. Šie metodai gali būti naudojami „juodosios dėžės“ situacijoje, skirtingai nuo intervalų metodų.

Tačiau Lipšico optimizavimo metodai turi ir svarbių trūkumų, lyginant su kitais globaliojo optimizavimo metodais. Lipšico metodai dažniausiai remiasi prielaida, kad Lipšico konstanta žinoma iš anksto, o tai retas atvejis sprendžiant praktinius uždavinius. Lipšico algoritmai yra lėtesni negu kai kurie kiti globaliojo optimizavimo algoritmai. Didėjant matmenų skaičiui n, Lipšico algoritmų skaičiavimo apimtys didėja eksponentiškai.

Dėl to geresnių rėžių, lemiančių greitesnius Lipšico algoritmus, kūrimas užima pagrindinę disertacijos dalį. Svarbią disertacijos dalį užima pasiūlytų algoritmų efektyvių lygiagrečiųjų versijų realizavimas ir įvertinimas, siekiant spręsti didesnius optimizavimo uždavinius.

Tyrimų objektas

Disertacijos tyrimų objektas yra:

- Lipšico globaliojo optimizavimo algoritmai daugelio kintamųjų erdvėse, naudojantys simpleksinius posričius;
- lygiagrečiųjų kompiuterių sistemų taikymas globaliajam optimizavimui.

Darbo tikslas ir uždaviniai

Pagrindinis šio darbo tikslas – modifikuoti esamus Lipšico globaliojo optimizavimo algoritmus daugelio kintamųjų erdvėse ir pasiūlyti naujus siekiant greičiau ir tiksliau spręsti optimizavimo uždavinius, bei panaudojant lygiagrečiųjų kompiuterių sistemas spręsti didesnius optimizavimo uždavinius. Siekiant užsibrėžto tikslo buvo sprendžiami šie uždaviniai:

1. Apžvelgti esamus globaliojo optimizavimo algoritmus ir apsibrėžti tiriamų algoritmų grupę;
2. Ištirti, kaip optimizavimo rezultatus įtakoja Lipšico rėžių sudarymui naudojamos įvairios normos ir jas atitinkančios Lipšico konstantos.

3. Naudojant simpleksinius posričius sukonstruoti:
   - patikslintus įprastus Lipšico rėžius, naudojančius įvairias normas ir jas atitinkančias Lipšico konstantas.
   - Pijavskij tipo Lipšico rėžius, daugiamačiu atveju gaunamus sprendžiant tiesinių lygčių sistemas.
   - Lipšico rėžius, pagrįstus apibrėžtinės daugiamatės sferos spinduliu.
   - agreguotuosius Lipšico rėžius sudarytus iš kelių skirtingų Lipšico rėžių tipų su įvairiomis normomis.
4. Sukurti nuosekliuosius ir lygiagrečiuosius Lipšico optimizavimo simpleksinius šakų ir rėžių algoritmus su pasiūlytais rėžiais.
5. Palyginti sukurtų algoritmų efektyvumą naudojant įvairius pasiūlytus Lipšico rėžius.
6. Gautus rezultatus palyginti su kitais gerai žinomais Lipšico optimizavimo algoritmais.
7. Ištirti sukurtų nuosekliųjų ir lygiagrečiųjų algoritmų efektyvumą naudojant įvairias paieškos strategijas: geryn, platyn, gilyn ir statistinę.
8. Palyginti sukurtų lygiagrečiųjų algoritmų, skirtų bendrosios ir paskirstytosios atminties kompiuteriams, lygiagretinimo efektyvumą.

Mokslinis darbo naujumas

Remiantis teoriniais ir eksperimentiniais tyrimais, ištirta, kaip Lipšico optimizavimo rezultatus įtakoja naudojamos normos ir jas atitinkančios Lipšico konstantos. Gautų rezultatų pagrindu pasiūlyta įprastų Lipšico rėžių skaičiavimui vietoj vienos konkrečios normos naudoti kelias bei apjungti skirtingų tipų rėžius su įvairiomis normomis. Tokiu būdu gaunami tikslesni agreguotieji Lipšico rėžiai neskaičiuojant tikslo funkcijos papildomuose taškuose.

Naudojant pirmąją normą pasiūlyta, kaip Pijavskij tipo rėžius apskaičiuoti sprendžiant tiesinių lygčių sistemas vietoj kvadratinių Euklidinės normos atveju.

Euklidinės normos atvejui, pasiūlytas naujas Lipšico rėžis, pagrįstas apibrėžtinės daugiamatės sferos spinduliu.

Pasiūlyto simpleksinio šakų ir rėžių algoritmo efektyvumas ištirtas naudojant įvairius pasiūlytus Lipšico rėžius ir palygintas su kitais gerai žinomais Lipšico algoritmais.

Pagal funkcijų skaičiavimo kiekio, optimizavimo laiko, ištirtų posričių skaičiaus bei maksimalaus kandidatų sąrašo kriterijus algoritmo efektyvumas ištirtas priklausomai nuo naudojamos paieškos strategijos: geryn, platyn, gilyn ir statistinės.

Sukurtos pasiūlyto algoritmo lygiagrečiosios realizacijos bendrosios (naudojant OpenMP standartą) ir paskirstytosios (naudojant MPI standartą) atminties kompiuteriams.

Sukurtų algoritmų pagrindu realizuota programinė įranga:

- nuoseklioji versija: http://www.gridglobopt.vgtu.lt/files/AT_Projekto_Ataskaita/BranchBoundTemplate/BBSequential/Examples/SimplicialBBSequential/
- lygiagrečioji versija, skirta bendrosios atminties kompiuteriams: http://www.gridglobopt.vgtu.lt/files/AT_Projekto_Ataskaita/BranchBoundTemplate/BBParallel/Examples/SimplicialBBOpenMP/
- lygiagrečioji versija, skirta paskirstytosios atminties kompiuteriams: http://www.gridglobopt.vgtu.lt/files/AT_Projekto_Ataskaita/BranchBoundTemplate/BBParallel/Examples/SimplicialBBParallel/

Autoriaus dalyvavimas mokslinėse programose

Autorius dalyvavo vykdant:

- Lietuvos valstybinio mokslo ir studijų fondo finansuojamą Aukštųjų technologijų programos projektą „Globalus sudėtingų sistemų optimizavimas naudojant didelio našumo skaičiavimus ir grid technologijas“ (2007-2009). http://www.gridglobopt.vgtu.lt/
- Lietuvos Respublikos švietimo ir mokslo ministerijos finansuojamą „Lygiagrečių ir paskirstytų skaičiavimų ir e-paslaugų tinklo (LitGrid)“ programą (2008 – 2009).

Autorius nuo 2008 m. birželio 10 d. iki 2008 m. rugsėjo 9 d. remiamas Europos komisijos finansuojamo HPC-Europa projekto stažavosi superskaičiavimų centre EPCC (The Edinburgh Parallel Computing Centre), Edinburge, Jungtinėje Karalystėje.

Ginamieji disertacijos teiginiai

1. Naudojant agreguotuosius Lipšico rėžius, sudarytus iš kelių skirtingo tipo rėžių su įvairiomis normomis, gaunami tikslesni rėžiai, kurių dėka pagreitinami optimizavimo algoritmai.
2. Pijavskij tipo rėžių skaičiavimas, naudojant pirmąją normą, leidžia kvadratinių lygčių sistemų sprendimą pakeisti tiesinių lygčių sistemų sprendimu.
3. Lipšico rėžiai, pagrįsti apibrėžtinės daugiamatės sferos spinduliu, dažniausiai yra geresni už įprastai naudojamus šakų ir rėžių algoritmuose.
4. Nėra vienos konkrečios paieškos strategijos, kurią naudojant būtų gaunami geriausi optimizavimo rezultatai pagal visus tirtus kriterijus.
5. Atlikus viršūnių patikrinimą stipriai sumažėja tikslo funkcijų skaičiavimų kiekis bei padidėja lygiagrečiojo bendrosios atminties kompiuteriams skirto algoritmo efektyvumas.
6. Paskirstytosios atminties kompiuteriams skirto lygiagrečiojo algoritmo funkcijų skaičiavimų kiekis su geriausios surastos funkcijos reikšmės apsikeitimu ir be jo, skiriasi nežymiai.

Darbo rezultatų aprobavimas

Pagrindiniai disertacijos rezultatai paskelbti šiuose moksliniuose leidiniuose: *Optimization Letters, Springer Optimization and Its Applications, Mathematical Modelling and Analysis, Technological and Economic Development of Economy, Information Technology and Control.* Paskelbtos 6 publikacijos referuojamos ISI Web of science, 1 kitose tarptautinėse duomenų bazėse. Publikacijų sąrašas pateiktas papildomame sąraše „Autoriaus publikacijų disertacijos tema sąrašas“.

Disertacijos rezultatai pristatyti šiuose pranešimuose tarptautinėse ir nacionalinėse mokslinėse konferencijose ir seminaruose:

- **R. Paulavičius**, J. Žilinskas. Įvairių normų ir jas atitinkančių Lipšico konstantų analizė globaliajam optimizavimui. *Lietuvos jaunųjų mokslininkų konferencija: Operacijų tyrimas ir taikymai, LOTD - 2006*, gegužės 26 d., 2006, Vilnius. http://www.mii.lt/ot-2006/
- **R. Paulavičius**, J. Žilinskas. Įvairių normų ir jas atitinkančių Lipšico konstantų junginio panaudojimas globaliajam optimizavimui dau-

giamačiu atveju. *Lietuvos jaunųjų mokslininkų konferencija: Operacijų tyrimas ir taikymai, LOTD - 2007*, gegužės 18 d., 2007, Vilnius. http://www.mii.lt/ot-2007/

- **R. Paulavičius**, J. Žilinskas. Improved Lipschitz bounds with non Euclidean norms for function values over multidimensional simplex. *12th International Conference Mathematical Modelling and Analysis*, May 30–June 2, 2007, Trakai, Lithuania.
- **R. Paulavičius**, J. Žilinskas. Improved Bounds with Non Euclidean Norms and Combination of different norms in Lipschitz Optimization. *Informatics summer school "Modern data mining technologies"*, September 9-15, Druskininkai, Lithuania. http://www.mii.lt/inmadra/iss-2007/
- **R. Paulavičius**. Parallel multidimensional Lipschitz optimization. Edinburgo universiteto optimizavimo seminare. *HPC-Europa Visitor programme*, June 23, 2008, EPCC supercomputing centre at the University of Edinburgh.
- **R. Paulavičius**, J. Žilinskas. Parallel branch and bound algorithm with combination of Lipschitz bounds over multidimensional simplices for multicore computers. *High Performance Scientific Computing "International Networking for Young Scientists"*, February 5-8, 2008, Druskininkai, Lithuania. http://inga.vgtu.lt/~inga/inys2008/
- **R. Paulavičius**, J. Žilinskas. Branch and bound with simplicial partitions and combination of Lipschitz bounds for global optimization. *International Conference EurOPT-2008 "Continuous Optimization and Knowledge-Based Technologies EurOPT-2008"*, May 20-23, 2008, Neringa, Lithuania. http://www.mii.lt/europt-2008/
- **R. Paulavičius**, J. Žilinskas. Investigation of simplicial branch and bound algorithms for multidimensional Lipschitz optimization. *Veszprém Optimization Conference: Advanced Algorithms (VOCAL 2008)*, December 15-17, 2008, Veszprém, Hungary. http://www.dcs.vein.hu/vocal2008/
- **R. Paulavičius**, J. Žilinskas. Investigation of selection strategies in branch and bound algoritm with simplicial partitions and combination of Lipschitz bounds. *14th International Conference Mathematical Modelling and Analysis*, 2009 gegužės 27-30, Daugavpils, Latvija. http://www.de.dau.lv/mma2009/

- **R. Paulavičius**, J. Žilinskas. Simplicial and Rectangular Branch and Bound with Improved Computationally Cheap Bounds. *15th International Conference Mathematical Modelling and Analysis*, May 26-29, 2010, Druskininkai, Lithuania. http://www.vgtu.lt/mma/mma2010/

Disertacijos struktūra

Disertaciją sudaro įvadas, trys skyriai ir bendrosios išvados. Papildomai disertacijoje yra pateikta: naudotų žymėjimų ir santrumpų sąrašas, eksperimentiniuose tyrimuose naudoti optimizavimo uždaviniai, sąvokų žodynas, dalykinė rodyklė bei literatūros sąrašas. Bendra disertacijos apimtis yra 134 puslapiai, kuriuose pateikta: 20 paveikslų, 15 lentelių ir 5 algoritmai. Disertacijoje remtasi 135 literatūros šaltiniais.

1

Globaliojo optimizavimo metodai

Tradiciškai globaliojo optimizavimo uždavinys formuluojamas taip: reikia rasti netiesinės n tolydžiųjų kintamųjų funkcijos, vadinamos tikslo funkcija, $f(x) : \mathbb{R}^n \to \mathbb{R}$ maksimumą (arba minimumą):

$$f^* = \max_{x \in \mathbf{D}} f(x), \quad (\text{arba} \quad f^* = \min_{x \in \mathbf{D}} f(x)) \tag{1.1}$$

n-matėje leistinojoje srityje $\mathbf{D} \subseteq \mathbb{R}^n$. Be globaliojo maksimumo (arba minimumo) f^*, turi būti surastas vienas arba visi globaliojo maksimumo (arba minimumo) taškai $x^* : f(x^*) = f^*$ arba parodyta, kad toks taškas neegzistuoja.

Disertacijoje nagrinėjami optimizavimo uždaviniai, kurių leistinoji sritis $\mathbf{D}$ yra kompaktas, o tikslo funkcijos tenkina Lipšico sąlyga (1.3) ir turi baigtinį izoliuotų ekstremumų skaičių. Kadangi Lipšico funkcijos yra tolydžiosios, todėl taško x^* egzistavimas išplaukia iš Vejerštraso teoremos [72].

Lipšico optimizavime uždavinio (1.1) sprendimas, net ir paprasčiausiu vienmačiu ($n = 1$) atveju, yra sudėtingas uždavinys. Hansen, Jaumard ir Lu parodė [50], kad per baigtinį funkcijos skaičiavimų kiekį globaliojo optimumo taškas x^* gali būti surastas tik tada, kai egzistuoja toks $\delta > 0$, kad

$$f(x) = f(x^*) - L|x - x^*|, \quad \forall x \in [x^* - \delta, x^* + \delta] \cap \mathbf{D}.$$

Vadinasi, tik labai siaura Lipšico funkcijų klasė gali būti tiksliai išsprendžiama panaudojus baigtinį funkcijos skaičiavimų kiekį. Todėl Lipšico optimizavime,

ieškoma ε-globaliojo sprendinio, t.y. surandamas toks leistinosios srities taškas x_{opt}, kad $f(x_{opt})$ skiriasi nuo tikrojo nežinomo globaliojo sprendinio $f(x^*)$ ne daugiau nei per $\varepsilon > 0$

$$|f(x_{opt}) - f(x^*)| \leq \varepsilon. \tag{1.2}$$

Visi disertacijoje apžvelgiami maksimizavimo algoritmai, ekvivalenčiai gali būti pakeisti į minimizavimo algoritmus, pasinaudojant tuo, kad:

$$\min f(x) = -\max(-f(x)).$$

Todėl disertacijoje apsiribojama maksimizavimu.

1.1. Globaliojo optimizavimo metodų apžvalga

Globaliojo optimizavimo uždavinių sprendimui pasiūlyta aibė įvairiausių metodų. Pirmąja monografija, kurioje apibendrinta globaliųjų metodų įvairovė, laikytina [18, 19]. Išsamesnėje ir naujesnėje monografijoje [121] pasiūlyta globaliojo optimizavimo metodus klasifikuoti į tokias grupes:

- Tiesioginiai metodai:
  - Atsitiktinės paieškos metodai,
  - Apibendrinti nusileidimo metodai,
  - Grupavimo metodai;
- Netiesioginiai metodai:
  - Metodai, aproksimuojantys lygio aibes,
  - Metodai, aproksimuojantys tikslo funkciją;
- Metodai, galintys užtikrinti sprendinio tikslumą:
  - Padengimo metodai.

Atsitiktinės paieškos metodai būna adaptyvūs ir neadaptyvūs. Neadaptyvūs atsitiktinės paieškos metodai naudoja globalią paieškos strategiją, visai neatsižvelgiant į anksčiau atliktų bandymų rezultatus. Paprasčiausio atsitiktinės paieškos, Monte-Karlo metodo, bandymų taškų koordinatės – nepriklausomi atsitiktiniai dydžiai, tolygiai pasiskirstę daugiamatėje leistinojoje srityje. Atsitiktinai sugeneruoti bandymų taškai gali būti naudojami, kaip pradiniai taškai lokaliosioms paieškoms. Gali būti daroma viena lokalioji paieška iš geriausio atsitiktine paieška rasto taško arba atliekami kartotiniai nusileidimai iš visų atsitiktinai sugeneruotų taškų. Šios klasės metodai, naudojantys įvairius

skirstinius, tirti daugelio tyrėjų [1, 19, 21, 103]. Šie metodai paprasti, tačiau nelabai efektyvūs. Bendru atveju, taikant atsitiktinės paieškos metodus tikimybė surasti globalų maksimumą artėja į vienetą tik tada, kai bandymų taškų skaičius artėja į begalybę. Dažniausiai tokiuose algoritmuose sustojimo sąlyga laikomas nustatytas iteracijų skaičius. Tačiau dalis autorių siūlo naudoti alternatyvias, labiau pagrįstas, sustojimo sąlygas [7, 8, 20]. Kvazi Monte-Karlo metodai naudoja deterministinės LP sekas [86–88]. Kvazi Monte-Karlo metodu surasto globaliojo sprendinio paklaida įvertinama naudojant įvairius taškų išsidėstymo tolygumo matus.

Dėl savo paprastumo ir nesudėtingos realizacijos, kitų sričių specialistai šiuos metodus dažnai naudoja įvairių praktinių problemų sprendimui. Kartais šie metodai taikomi nustatant optimizavimo uždavinio charakteristikas: globaliųjų ir lokaliųjų optimumo taškų skaičių, tikimybę, kad lokalioji paieška iš atsitiktinio leistinosios srities taško pasieks globalųjį maksimumą ir pan. Šiuos metodus paprasta lygiagretinti. Kiekvienas procesorius nepriklausomai atlieka tą patį algoritmą visoje leistinojoje srityje ar jos dalyje. Procesoriai tarpusavyje nekomunikuoja, todėl spartinimo koeficientas lygus procesorių skaičiui, o lygiagretinimo efektyvumas – 1.

Adaptyvūs atsitiktinės paieškos metodai [76, 118] atsižvelgia į anksčiau atliktus bandymus. Šios klasės metodai bandymų taškus leistinojoje srityje generuoja ne tolygiai, o tankiau daugiau žadančiose leistinosios srities vietose. Geriausios surastos funkcijos reikšmės nurodo daugiau žadančias sritis. Adaptyviųjų atsitiktinių metodų klasei priklauso genetiniai algoritmai [45, 80], *evoliucinės strategijos* [35, 112], atkaitinimo modeliavimo metodai [13, 57, 58, 68]. Adaptyviųjų paieškos metodų efektyvumas tradiciškai priklauso nuo jų parametrų reikšmių parinkimo, todėl geriausiai veikia naudojami jų autorių.

Apibendrinti nusileidimo metodai apibendrina lokaliųjų nusileidimo metodų idėjas globaliajam atvejui. Leistinųjų krypčių metodai modifikuoja atliekamų lokalių nusileidimų kryptį taip, kad ji eitų per visus lokaliųjų maksimumų taškus [29, 124]. Apibendrinti baudų metodai modifikuoja tikslo funkciją pridedant baudą taip, kad būtų išvengtų jau surastų lokaliųjų sprendinių. Tuneliavimo metodas [73] naudoja dvi fazes: lokaliąją paiešką, kurios metu surandamas lokaliojo maksimumo taškas x' ir tuneliavimą, kurio metu maksimizuojama modifikuota tikslo funkcija, t.y. surandamas toks taškas x'', kad $f(x') < f(x'')$. Modifikuota tikslo funkcija sudaroma taip, kad lokalaus maksimizavimo procedūra iš taško x' surastų tašką x''. Törn ir Žilinskas [121] pažymi, kad apibendrinti nusileidimo metodai, savo metodika yra panašūs į kartotinio nusileidimo metodus su nelokaliu optimizavimo pobūdžiu.

Grupavimo metodai [66, 67, 120] yra vieni iš efektyviausių globaliojo optimizavimo metodų. Grupavimo metodai atsitiktinai (naudojant Monte Karlo metodą) parinktus pradinius bandymų taškus apjungia į klasterius: iš pradinių taškų leidžiamasi į lokaliuosius maksimumus bei analizuojami naujai gauti apytiksliai sprendiniai. Jei taškai sudaro klasterį, spėjama, kad jie yra iš to paties lokaliojo maksimumo traukos srities. Tuomet tik iš vieno klasterio taško yra atliekama lokalioji paieška, o taškų skaičius palaipsniui mažinamas. Kiekviename iš šių žingsnių, naudojami įvairūs klasterizavimo ir lokaliosios paieškos algoritmai. Pagrindinis grupavimo metodų privalumas, kad jais randami ne tik globalieji, bet ir lokalieji maksimumai, o tai dažnai domina specialistus. Taip pat šie metodai sėkmingai pritaikyti spręsti sudėtingiems, 40 matmenų uždaviniams. Kita vertus, jei tikslo funkcija turi daug lokaliųjų maksimumų, tai grupavimo metodai veikia lėtai, nes reikia daug lokaliųjų paieškų norint rasti globalųjį maksimumą.

Netiesioginiai metodai aproksimuojantys tikslo funkciją (ar jos lygio aibes) naudoja statistinius modelius tikslo funkcijos reikšmių modeliavimui. Tikslo funkcijos reikšmės traktuojamos kaip atsitiktiniai dydžiai. Jei funkcijos reikšmės kai kuriuose taškuose yra žinomos, tai reikšmės kituose taškuose traktuojamos kaip sąlyginiai atsitiktiniai dydžiai. Naudojant šios klasės metodus, atliekami sudėtingi skaičiavimai kito bandymo taško nustatymui, todėl šie metodai naudingi „brangių“ tikslo funkcijų atveju. Monografijos [121] autoriai pažymi, kad šie metodai sėkmingai pritaikyti iki 15-os matmenų optimizavimo uždavinių sprendimui. Bajesinė globaliojo optimizavimo teorija ir iš jos išplaukiantys metodai pateikiami šios teorijos pradininko J. Mockaus monografijoje [84].

Padengimo metodai dažniausiai yra deterministiniai. Esant tam tikroms sąlygom, tokie metodai gali garantuoti, kad per baigtinį (gali būti ir labai ilgas) laiką, bus surasta globaliojo maksimumo (ar visų maksimumų) sritis (sritys). Padengimo metodais nustatomos sritys, kuriose negali būti globaliojo maksimumo ir jos pašalinamos iš tolesnės paieškos. Posričiai baigiami dalyti, kai globaliojo maksimumo taškai lokalizuojami pakankamai mažuose leistinosios srities posričiuose. Ieškant šalintinų sričių reikia įvertinti funkcijos rėžius. Visos maksimizuojamos funkcijos reikšmės leistinojoje srityje yra mažesnės už tikslo funkcijos viršutinį rėžį šioje srityje. Jei tikslo funkcijos viršutinis rėžis tam tikrame leistinosios srities posrityje yra mažesnis už jau žinomą funkcijos reikšmę, tai šioje srityje globaliojo maksimumo taško nėra, o ši sritis gali būti pašalinta iš tolesnės paieškos.

Rėžių įvertinimui, pagal kuriuos nustatomos šalintinos sritys, dažniausiai reikalinga papildoma informacija, kaip pavyzdžiui, Lipšico konstanta ir pan.

Tačiau tokio tipo informacija ne visada iš anksto žinoma sprendžiant praktinius uždavinius.

Įvairios rėžių skaičiavimo taisyklės, naudojamos šakų ir rėžių metoduose, išsamiai apžvelgtos 1.2 skyriuje.

1.2. Šakų ir rėžių metodas

Šakų ir rėžių metodas pasiūlytas Horst [52], Horst ir Tuy [55, 56] darbuose naudojamas realizuojant globaliojo optimizavimo padengimo [133] ir kombinatorinio optimizavimo metodus. Šakų ir rėžių algoritmai gali būti naudojami posričių sąrašui ir šalinimo bei dalijimo procesui valdyti. Šakų ir rėžių algoritmus sudaro inicializavimo, išrinkimo, dalijimo bei rėžių skaičiavimo taisyklės. Inicializavimo etape leistinoji sritis $\mathbf{D}$ padengiama nurodytos formos posričiais. Toliau vykdomas ciklas: iš kandidatų aibės $\mathbb{I}$ išrenkamas ir padalijamas posritis; apskaičiuojami funkcijos maksimumo apatinis LB ir viršutinis UB rėžiai naujai gautuose posričiuose; posričiai, kuriuose maksimumo taškas negali būti, pašalinami, o nepašalinti posričiai įtraukiami į kandidatų aibę $\mathbb{I}$. Algoritmu siekiama, kad kandidatų aibė greitai mažėtų ir konverguotų į sprendinį $\mathbf{S}$. Apibendrintas globaliojo optimizavimo šakų ir rėžių algoritmas yra pateiktas 1 algoritme. Padengimo ir padalijimo taisyklės priklauso nuo naudojamų posričių formos: gali būti naudojami hiperstačiakampiai, simpleksai, hiperkūgiai ar hipersferos.

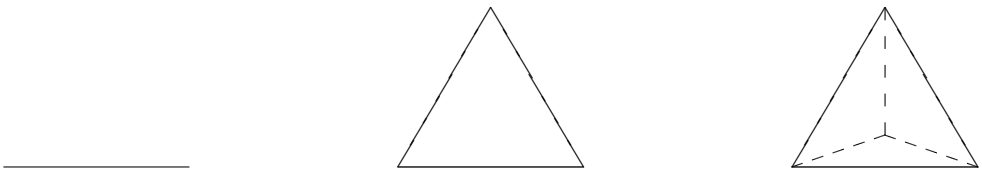
Galimos išrinkimo taisyklės:

- geriausiojo išrinkimo – išrenkamas $\mathbb{I}$ elementas su geriausiu įverčiu (su didžiausiu viršutiniu tikslo funkcijos rėžiu);
- *gilyn* – išrenkamas jauniausias $\mathbb{I}$ elementas;
- *platyn* – išrenkamas vyriausias $\mathbb{I}$ elementas.

Rėžių skaičiavimo taisyklės nusako, kaip funkcijos maksimumo rėžiai yra įvertinami. Šakų ir rėžių algoritmų efektyvumas priklauso nuo rėžių tikslumo [128]. Viršutinis tikslo funkcijos rėžis gali būti nustatytas naudojant išgaubtus funkcijos apvalkalus [31, 47, 61, 62]. Intervalų metoduose tikslo funkcijos rėžiai įvertinami naudojant intervalų aritmetiką [48] ar jų modifikacijas [134, 135]. Statistiniai [127] arba euristiniai įverčiai [75, 131] gali būti naudojami. Nors tokiu atveju negalima garantuoti, kad rastas globaliojo optimumo taškas, tokie algoritmai gali būti pritaikyti „juodosios dėžės“ uždaviniams. Viena paprasčiausių ir dažniausiai naudojamų prielaidų yra, kad tikslo funkcija tenkina Lipšico sąlyga [53]. Geriausi Lipšico rėžiai gaunami realizuojant Pijavskij [100] metodo idėją. Tačiau Pijavskij tipo rėžių apskaičiavimas daugiamačiu atveju

1 algoritmas Apibendrintas šakų ir rėžių algoritmas

Padengiama $\mathbf{D}$: $\mathbb{I} \leftarrow \{\mathbf{I}_j | \mathbf{D} \subseteq \bigcup \mathbf{I}_j, j = 1, \ldots, m\}$ naudojant padengimo taisyklę
$\mathbf{S} \leftarrow \emptyset$, $LB(\mathbf{D}) \leftarrow -\infty$
while $(\mathbb{I} \neq \emptyset)$ **do**
  Išrenkamas $\mathbf{I}_j \in \mathbb{I}$ naudojant **išrinkimo taisyklę**, $\mathbb{I} \leftarrow \mathbb{I} \backslash \{\mathbf{I}_j\}$
  if $(LB(\mathbf{D}) < UB(\mathbf{I}_j) - \varepsilon)$ **then**
    Padalijamas $\mathbf{I}_j$ į p posričius $\mathbf{I}_{j_k}$ naudojant **dalijimo taisyklę**
    for all $(\mathbf{I}_{j_k}, k = 1, \ldots, p)$ **do**
      Randami $LB(\mathbf{I}_{j_k} \bigcap \mathbf{D})$ ir $UB(\mathbf{I}_{j_k})$ naudojant **rėžių skaičiavimo taisyklę**
      $LB(\mathbf{D}) \leftarrow \max\{LB(\mathbf{D}), LB(\mathbf{I}_{j_k} \bigcap \mathbf{D})\}$
      if $(LB(\mathbf{D}) < UB(\mathbf{I}_{j_k}) - \varepsilon)$ **then**
        if ($\mathbf{I}_{j_k}$ gali būti sprendinys) **then**
          $\mathbf{S} \leftarrow \mathbf{I}_{j_k}$
        else
          $\mathbb{I} \leftarrow \mathbb{I} \cup \{\mathbf{I}_{j_k}\}$
        end if
      end if
    end for
  end if
end while



$n = 1$ $n = 2$ $n = 3$

1.1 pav. *Vienmatis, dvimatis ir trimatis simpleksas*

($n \geq 2$) yra sudėtingas uždavinys. Todėl dauguma šakų ir rėžių algoritmų naudoja ne tokius efektyvius, bet lengviau apskaičiuojamus rėžius.

Simpleksas yra iškiloji n-matė geometrinė figūra, turinti $n + 1$ viršūnę. Vienmatėje erdvėje simpleksas yra atkarpa, dvimatėje – trikampis, trimatėje – tetraedras (1.1 paveikslas). Simpleksas yra figūra, n-matėje erdvėje turinti mažiausiai viršūnių. Jeigu vertinant funkcijos rėžius naudojamos funkcijos reikšmės viršūnėse, tai simpleksas yra tinkamiausia posričių forma [130].

Simpleksiniai posričiai dalijami į mažesnius. Yra žinoma, kad funkcijos rėžiai iškreiptame simplekse nėra geri. Netaisyklingas trikampis (dvimatis simpleksas) gali būti padalytas į 4 panašius trikampius, status lygiašonis trikampis gali būti padalytas į 2 panašius, kaip parodyta 1.2 paveiksle. Jei naudoja-

1.2 pav. *Dvimačių simpleksų dalijimas į panašius simpleksus*

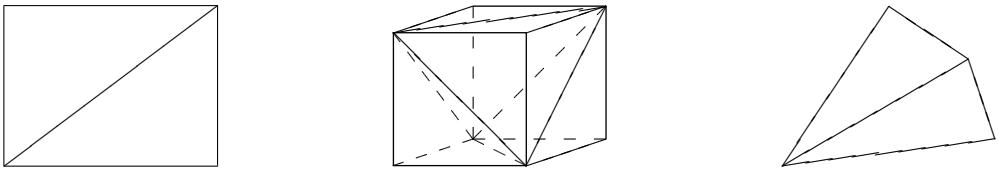
1.3 pav. *Simpleksų dalijimas į du hiperplokštuma, einančia per ilgiausios briaunos vidurį ir viršūnes, nepriklausančias ilgiausiajai briaunai*

mos kitos dalijimo taisyklės, reikia išvengti simpleksų iškraipymo. Dažniausiai naudojama iškraipymo išvengianti simpleksų dalijimo strategija yra simpleksų dalijimas į du hiperplokštuma, einančia per ilgiausios briaunos vidurį ir viršūnes, nepriklausančias ilgiausiajai briaunai. Taip užtikrinama, kad ilgiausia simplekso briauna nebūtų daugiau negu dvigubai ilgesnė už kitas. Tokio dalijimo pavyzdžiai yra pateikti 1.3 paveiksle.

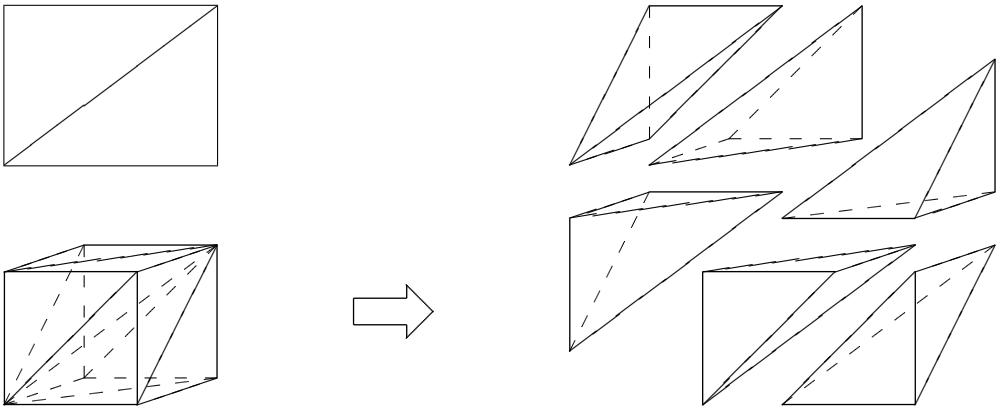
Simpleksinių posričių trūkumas – poreikis padengti leistinąją sritį simpleksais, nes dažniausiai globaliojo optimizavimo uždavinių leistinosios sritys yra hiperstačiakampiai, apibrėžti kintamųjų intervalais. Naudojant viršūnių trianguliaciją stačiakampis gali būti padalytas į du trikampius. Trimatis stačiakampis gali būti padalytas į penkis tetraedrus, kaip parodyta 1.4 paveiksle. Tačiau toks padalijimas nėra bendras – nėra tinkamas bet kokio matmenų skaičiaus erdvei. Apibendrintas (bet kokio matmenų skaičiaus erdvei) kombinatorinis hiperstačiakampio viršūnių trianguliacijos algoritmas pasiūlytas [119]. Toks viršūnių trianguliacijos būdas yra deterministinis, gaunamų simpleksų skaičius yra iš anksto žinomas – $n!$. Visi simpleksai yra vienodo hipertūrio – $1/n!$ hiperstačiakampio hipertūrio. Be to pridėjus vieną tašką hiperstačiakampio įstrižainės viduryje, galima padalyti visus simpleksus į du.

Dvimačio ir trimačio hiperstačiakampių kombinatorinės viršūnių trianguliacijos pavyzdžiai yra pateikti 1.5 paveiksle.

Kadangi simpleksų skaičius yra iš anksto žinomas ($n!$), efektyvus lygiagretusis simpleksų vardinimas yra galimas. Kėlinių pažymėjimui gali būti nau-



1.4 pav. *Leistinųjų sričių viršūnių trianguliaciją*

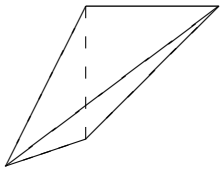


1.5 pav. *Dvimačio ir trimačio hiperstačiakampių kombinatorinė viršūnių trianguliacija*

dojami faktoriniai numeriai, kurį žinant galima nesunkiai nustatyti atitinkantį kėlinį. Toks algoritmas gali būti naudojamas padengti globaliojo optimizavimo uždavinių hiperstačiakampias leistinąsias sritis simpleksais ir paskirstyti pradinį darbą lygiagrečiuosiuose šakų ir rėžių algoritmuose su simpleksiniais posričiais.

Simpleksinių posričių privalumas – bendresnės formos tiesiniais nelygybiniais apribojimais apibrėžtos leistinosios sritys gali būti padengtos simpleksais naudojant viršūnių trianguliaciją, kas nėra galima hiperstačiakampių posričių atveju. Tokiu būdu ribojimai yra įvertinami pradiniu padengimu.

Jeigu kintamųjų x_i ir x_j reikšmių sukeitimas nekeičia tikslo funkcijos reikšmės, ji yra simetrinė hiperplokštumos $x_i = x_j$ atžvilgiu. Tokiu atveju egzistuoja ekvivalentūs leistinosios srities posričiai ir optimumo taškai. Paieškos sritis gali būti apribota tiesiniais nelygybiniais apribojimais išvengiant ekvivalenčių leistinosios srities posričių ir ieškant tik vieno iš ekvivalenčių sprendinių. Tai užtikrinama tiesiniais nelygybiniais apribojimais $x_i \leq x_j$ nusakant apkeičiamų kintamųjų reikšmių tvarką. Gauta paieškos sritis su apribojimais gali būti padengta simpleksais naudojant viršūnių trianguliaciją. Tokiu būdu išvengiant



1.6 pav. *Trimatė paieškos sritis*

simetrijų yra sumažinama paieškos sritis ir lokaliųjų optimumų bei globaliojo optimumo taškų skaičiai.

Pavyzdžiui, tegu optimizavimo uždavinys yra:

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(x_i) + 1, \mathbf{D} = [-500, 700]^n.$$

Tokia tikslo funkcija yra simetrinė hiperplokštumų $x_i = x_j$ atžvilgiu. Apribojimai $x_1 \leq x_2 \leq \ldots \leq x_n$ gali būti nustatyti simetrijoms išvengti. Gauta paieškos sritis yra simpleksas, kurio viršūnių koordinatės yra aprašytos tokios matricos eilutėse:

$$D = \left\{ \begin{array}{ccccc} -500 & -500 & \ldots & -500 & -500 \\ -500 & -500 & \ldots & -500 & 700 \\ \vdots & & \ddots & & \vdots \\ -500 & 700 & \ldots & 700 & 700 \\ 700 & 700 & \ldots & 700 & 700 \end{array} \right\}.$$

Trimatė paieškos sritis pavaizduota 1.6 paveiksle. Bendru atveju ji yra daugiamatis simpleksas. Paieškos sritis ir optimumų taškų skaičius tokiu būdu sumažinamas $n!$ kartų lyginant su originalia hiperstačiakampe optimizavimo uždavinio leistinąja sritimi.

1.3. Lipšico optimizavimas

Lipšico optimizavimo metodai yra pakankamai plačiai išnagrinėti [56, 98, 102, 117] ir taikomi įvairių optimizavimo uždavinių sprendimui [22, 24, 54, 96]. Lipšico optimizavimas sprendžia globaliojo optimizavimo uždavinius, kuriuose tikslo funkcija tenkina Lipšico sąlygą. Reali n-kintamųjų funkcija vadinama Lipšico funkcija aibėje $\mathbf{D} \subseteq \mathbb{R}^n$, jeigu egzistuoja Lipšico konstanta

$L = L(f, \mathbf{D}) \geq 0$ tokia, kad

$$|f(x_1) - f(x_2)| \leq L\|x_1 - x_2\|, \quad \forall x_1, x_2 \in \mathbf{D}, \tag{1.3}$$

čia $\|\cdot\|$ – žymime vektoriaus normą. Lipšico optimizavime dažniausiai yra naudojama Euklidinė norma $\|\cdot\|_2$, tačiau gali būti naudojamos ir kitos l_p $(1 \leq p \leq \infty)$ normos

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}, \text{ kai } 1 \leq p < \infty \text{ ir } \|x\|_\infty = \max_{i=1,\ldots,n} |x_i|.$$

Lipšico funkcijų lokaliųjų ekstremumų aibės struktūra gali būti labai sudėtinga. Ši aibė gali būti kontinumo galios, o lokaliųjų Lipšico tikslo funkcijos ekstremumų reikšmės gali užpildyti kokį nors visą realiųjų skaičių intervalą. Disertacijoje apsiribojama Lipšico tikslo funkcijomis, turinčiomis baigtinį izoliuotų ekstremumų skaičių ir apimanti kvadratinio priskyrimo uždavinio, kuris yra NP sudėtingumo, tikslo funkcijas [42].

Lipšico funkcijos yra diferencijuojamos beveik visur, t.y. visur egzistuoja funkcijos gradientas $\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right)$, išskyrus nulinio Lebego mato aibes [101]. Kiekvieną Lipšico funkciją galima aproksimuoti glodžiai diferencijuojama funkcija norimu tikslumu [6].

Lipšico optimizavimo algoritmai į klases skirstomi pagal įvairius kriterijus. Vienas iš jų - Lipšico konstantos prigimtis.

1.3.1. Lipšico konstanta

Lipšico konstanta nėra universali, nes priklauso nuo konkrečios tikslo funkcijos bei srities, kurioje ta konstanta įvertinta, todėl pažymint šias priklausomybes kartais ji žymima $L(f, \mathbf{D})$. Trumpumo vardan dažniausiai naudojamas trumpesnis žymėjimas L.

Dauguma Lipšico optimizavimo metodų remiasi prielaida, kad Lipšico konstanta žinoma iš anksto, tačiau taip yra ne visada, o praktinių uždavinių atveju – retai. Dėl to atsirado įvairių Lipšico optimizavimo algoritmų, kuriuose Lipšico konstantos įvertinimas įeina į patį algoritmą. Pagal Lipšico konstantos prigimtį, Lipšico optimizavimo algoritmus galima suklasifikuoti į keturias grupes [110]:

1. laikoma, kad Lipšico konstanta žinoma iš anksto;
2. Lipšico konstanta įvertinama globaliai (visoje leistinojoje srityje $\mathbf{D}$);

1.1 lentelė. *Lipšico optimizavimo algoritmų klasifikacija pagal Lipšico konstantos prigimtį*

Lipšico konstanta	Lipšico funkcija	Lipšico funkcija ir išvestinė
Žinoma iš anksto	[4, 25, 56, 82, 100]	[4, 5, 9, 56]
Įvertinama globaliai	[56, 71, 98, 116]	[44, 106, 107, 116]
Lokalūs konstantos įverčiai	[70, 105, 109, 126]	[106, 109, 116]
Pasirenkama iš galimų reikšmių aibės	[33, 65, 108, 109]	–

3. Lipšico konstanta įvertinama lokaliai (leistinosios srities $\mathbf{D}$ posričiuose $\mathbf{I}_i : \mathbf{D} \subseteq \bigcup \mathbf{I}_i$ naudojami lokalūs Lipšico konstantos įverčiai);
4. Lipšico konstanta pasirenkama iš galimų reikšmių aibės.

Dalis šioms klasėms priklausančių Lipšico optimizavimo algoritmų pateikti 1.1 lentelėje. Didelė šių algoritmų dalis smulkiai aprašyta 1.4 skyriuje. Pateiktoje lentelėje išskirta atskira Lipšico optimizavimo algoritmų dalis, kuri remiasi prielaida, kad tikslo funkcijos išvestinė $f'(x)$ tenkina Lipšico sąlygą

$$|f'(y_1) - f'(y_2)| \leq K\|y_1 - y_2\|, \quad \forall y_1, y_2 \in \mathbf{D}, \tag{1.4}$$

čia $0 < K < \infty$. Šios klasės metodai, naudoja ne tik pačias tikslo funkcijos reikšmes, bet ir jų išvestinių reikšmes, ko pasekoje greičiau surandami globalūs Lipšico optimizavimo uždavinio (1.2) sprendiniai. Tačiau dauguma šios klasės algoritmų remiasi prielaida, kad išvestinės konstanta K yra žinoma iš anksto.

1.3.2. Lipšico rėžiai

Lipšico optimizavime konstantos žinojimo svarba išplaukia skaičiuojant Lipšico rėžius. Tegu leistinosios srities posritis $\mathbf{I} \subseteq \mathbf{D}$ yra n-matis stačiakampis ar n-matis simpleksas, o L – Lipšico konstanta, tuomet su visais $x_1, x_2 \in \mathbf{I}$ yra teisinga nelygybė

$$f(x_1) \leq f(x_2) + L\|x_1 - x_2\|.$$

Vadinasi, fiksavę laisvai pasirinktą tašką $x_1 \in \mathbf{I}$, gauname iškiląją viršutinio rėžio funkciją $F_1(x)$

$$F_1(x) = f(x_1) + L\|x - x_1\|, \tag{1.5}$$

kuri tikslo funkciją $f(x)$ aprėžia iš viršaus visame posrityje $\mathbf{I}$. Tuomet suradę „daugiausiai žadantį“ tašką $x_2 \in \mathbf{I}$, t.y. kuriame viršutinio rėžio funkcijos

$F_1(x)$ reikšmė didžiausia

$$x_2 = \arg\max_{x\in\mathbf{I}} F_1(x),$$

gauname antrą iškiląją funkciją $f(x_2) + L\|x - x_2\|$ aprėžiančią tikslo funkciją $f(x)$ iš viršaus visame posrityje $\mathbf{I}$. Po $k = 2$ žingsnių, geriausia viršutinio rėžio funkcija, išnaudojant turimą informaciją, gaunama tokiu būdu:

$$F_2(x) = \min_{i=1,2}\{f(x_i) + L\|x - x_i\|\}.$$

Analogiškai tęsiant procesą, po k žingsnių geriausia, Pijavskij tipo, viršutinio rėžio funkcija $F_k(x)$ (žr. 1.7 pav.) yra

$$F_k(x) = \min_{i=1,\ldots,k} \{f(x_i) + L\|x - x_i\|\}, \tag{1.6}$$

o $k+1$ iteracijos tašku x_{k+1} parenkamas, tas, kuriame viršutinio rėžio funkcija $F_k(x)$ įgija maksimumą

$$x_{k+1} = \arg\max_{x\in\mathbf{I}} F_k(x). \tag{1.7}$$

Tokiu būdu surandamas Pijavskij (φ) tipo rėžis, pagrįstas (1.6) ir (1.7)

$$\varphi(\mathbf{I}) = \max_{x\in\mathbf{I}} \min_{x_i\in\mathbf{T}}\{f(x_i) + L\|x - x_i\|\}, \tag{1.8}$$

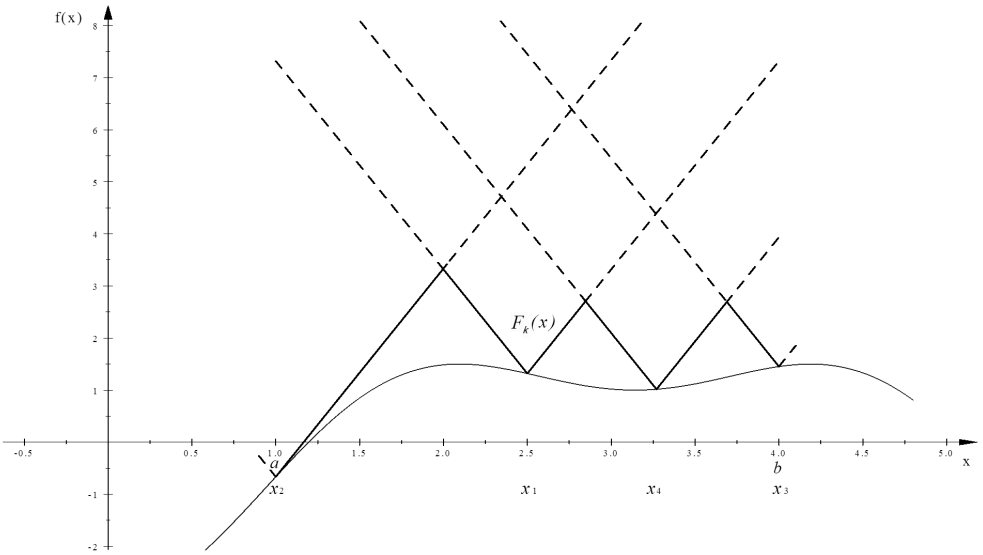
čia $\mathbf{T} \subset \mathbf{D}$ yra baigtinė aibė, kurios taškuose apskaičiuojamos maksimizuojamos tikslo funkcijos reikšmės $f(x_i), x_i \in \mathbf{T}$. Tačiau φ tipo rėžių radimas daugiamačiu atveju yra sudėtingas globaliojo optimizavimo uždavinys apimantis kvadratinių ir tiesinių lygčių sistemų sprendimą.

Tarkime, kad $\delta(\mathbf{I}) = \sup\{\|x_1 - x_2\| : x_1, x_2 \in \mathbf{I}\}$ yra posričio $\mathbf{I} \subseteq \mathbf{D}$ diametras. Jei $\mathbf{I}$ yra n-matis stačiakampis $\mathbf{I} = \{x \in \mathbb{R}^n : a \le x \le b\}$, tai diametras lygus $\|b - a\|$, jei n-matis simpleksas, tai diametras lygus ilgiausios kraštinės ilgiui. Tuomet iš (1.5) gauname viršutinį μ_1 tipo rėžį

$$\mu_1(\mathbf{I}) = \min_{x_i\in\mathbf{T}} f(x_i) + L\delta(\mathbf{I}), \tag{1.9}$$

nes $f(x) \le \mu_1(\mathbf{I})$, $\forall x \in \mathbf{I}$. Kai $\mathbf{I}$ yra n-matis stačiakampis ar n-matis simpleksas, aibė $\mathbf{T}$ gali būti aibės $\mathbf{I}$ viršūnių aibė $V(\mathbf{I})$.

Dalis Lipšico optimizavimo algoritmų viršutiniams μ_1 tipo rėžiams stačiakampyje apskaičiuoti vietoj funkcijos reikšmių viršūnėse $V(\mathbf{I})$, naudoja posričio



1.7 pav. *Pijavskij tipo viršutinio rėžio funkcija $F_k(x)$ vienmačiu atveju po $k = 4$ iteracijų*

I vidurio tašką $x_m = (a + b)/2$

$$\mu_m(\mathbf{I}) = f(x_m) + L\delta(\mathbf{I})/2. \tag{1.10}$$

Geresnį nei μ_1 tipo (1.9) viršutinį rėžį gauname

$$\mu_2(\mathbf{I}) = \min_{x_i \in \mathbf{T}} \left\{ f(x_i) + L \max_{x \in \mathbf{I}} \|x - x_i\| \right\}, \tag{1.11}$$

tačiau μ_2 apskaičiavimas yra brangesnis.

1.4. Lipšico optimizavimo algoritmai

Lipšico optimizavimas vienmačiu ($n = 1$) atveju nuodugniai ištirtas ir pasiūlyta aibė įvairių algoritmų. Vienmačių Lipšico optimizavimo algoritmų gerumo įverčiu laikomas funkcijų skaičiavimo kiekio skirtumas konkretaus algoritmo nuo teoriškai įmanomo geriausio algoritmo, besiremiančio prielaida, kad globalus maksimumo taškas yra žinomas [15, 16]. Išsami teorinė ir eksperi-

mentinė vienmačių Lipšico optimizavimo algoritmų lyginamoji analizė pateikta [49].

Disertacijoje pagrindinis dėmesys skiriamas daugiamačiui ($n \geq 2$) Lipšico optimizavimui, kuris tiek teoriškai, tiek praktiškai gerokai mažiau ištirtas nei vienmatis. Net ir dvimačiu ($n = 2$) atveju iki šiol nepasiūlytas teorinis geriausias įmanomas algoritmas. Daugiamačius Lipšico optimizavimo algoritmus galime suskirstyti į klases pagal įvairius kriterijus. Lipšico algoritmų klasifikaciją pagal Lipšico konstantos prigimtį pateikėme 1.3.1 poskyryje. P. Hansen kartu su bendraautoriais pateikia dar vieną Lipšico optimizavimo algoritmų klasifikaciją į tris klases [49].

Pirmajai klasei priklauso algoritmai, kuriuose daugiamačius uždavinius bandoma keisti vienmačiais. Pirmąją šios klasės įgyvendinimo idėją, daugiažingsnę kintamųjų skaičiaus mažinimo, pasiūlė Pijavskij darbuose [99, 100]. Antroji, panaudojant Peano kreives išnagrinėta Butz [10] ir Strongin [115] darbuose.

Antroji klasė remiasi Pijavskij metodo realizavimu daugiamačiu atveju, t.y. viršutinio rėžio skaičiavimui naudoja φ tipo rėžius (1.8). Šioje klasėje pažymėtini paties Pijavskij darbai [99, 100]. Įvairias Pijavskij metodo modifikacijos naudojant Euklidinę normą pasiūlė: Mladineo [82, 83], Jaumard, Herrmann ir Ribault [64]. Iš algoritmų, naudojančių kitas normas ar įvairias aproksimacijas pažymėtini: Mayne ir Polak [77], Wood [122, 123], Zhang, Wood ir Baritompa [125] darbai. Nors dauguma šios klasės algoritmų žvelgiant iš teorinės pusės yra išradingi, tačiau sunkiai realizuojami praktiniams uždaviniams, kai $n > 2$. Didžioji šiai klasei priklausančių algoritmų dalis gali būti realizuota kaip šakų ir rėžių algoritmai.

Trečiosios klasės algoritmai realizuoti simpleksiniu arba hiperstačiakampiu šakų ir rėžių algoritmų pagrindu bei viršutinių rėžių skaičiavimui naudojantys prastesnius, bet lengviau apskaičiuojamus μ_1, μ_m ir μ_2 tipo rėžius: Galperin [38, 40], Pinter [93–95], Meewella ir Mayne [78, 79], Gourdin Hansen ir Joumard [46]. Algoritmai vienas nuo kito skiriasi padengimo, išrinkimo, padalijimo ir rėžių skaičiavimo taisyklėmis.

Apžvelgsime kiekvienos klasės pagrindinius algoritmus plačiau.

1.4.1. Algoritmai, daugiamatį uždavinį keičiantys vienmačiu

Šiai klasei priklausantys algoritmai daugiamačius uždavinius keičia vienmačiais.

A. Daugiažingsnė kintamųjų skaičiaus mažinimo schema

Pijavskij daugiažingsnė kintamųjų skaičiaus mažinimo schema pasiūlyta darbe [100]. Jei n-matė leistinoji sritis yra hiperstačiakampis $\mathbf{D} = [a_1, b_1] \times \cdots \times [a_n, b_n]$ tai daugiamatis maksimizavimo uždavinys keičiamas eile vienmačių uždavinių

$$\max_{x \in \mathbf{D}} f(x) = \max_{x_1 \in \mathbf{D}_1} \left\{ \max_{x_2 \in \mathbf{D}_2(x_1)} \left\{ \dots \max_{x_n \in \mathbf{D}_n(x_1, \dots, x_{n-1})} f(x) \right\} \right\},$$

čia $\mathbf{D}_j(x_1, \dots, x_{j-1}), j = 1, 2, \dots, n$ yra vienmačiai intervalai $[a_j, b_j]$, o kintamieji $x_1, \dots, x_j$ laikomi konstantomis.

Tačiau norint rasti vienmačių tikslo funkcijų reikšmes reikia daug kartų spręsti maksimizavimo uždavinius, todėl viso uždavinio sprendimas pagal šią schemą nepalengvėja.

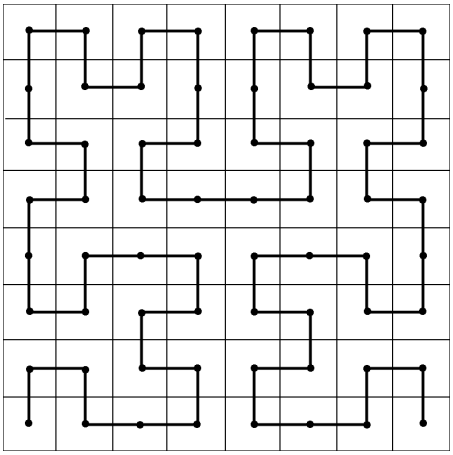
B. Peano kreivės metodas

Butz [10] ir Strongin [114] pasiūlė daugiamačių uždavinių keitimą vienmačiais panaudojant Peano kreives. N matmenų hiperkubinė leistinoji sritis, tarkime, kad tai dvimatis hiperkubas (kvadratas), padalijamas į 2^n lygių kvadratų. Kiekvieną iš naujai gautų kvadratų analogiškai dalijame vėl į 2^n naujų ir taip kartojame k kartų, kol gauname 2^{km} kvadratų, su kraštinių ilgiais $\frac{1}{2^k}$. Tuomet kiekvieno kvadrato centrą sujungiame atkarpomis taip, kad gautoji vienmatė kreivė „apeitų“ kvadrato taškus (žr. 1.8 pav.). Tokiu būdu daugiamatis uždavinys pakeičiamas vienmačiu uždaviniu ant Peano kreivės.

Lyginant daugiažingsnę kintamųjų skaičiaus mažinimo ir Peano kreivės idėjas, pastebime, kad pirmuoju atveju sprendinys surandamas panaudojant sąlyginai mažai leistinosios srities taškų, priešingai nei Peano kreivės atveju, kuris naudoja visų leistinosios srities taškų aproksimaciją. Taip pat pastebime, kad Euklidinis atstumas tarp dviejų hiperkubų centrų, kai kuriais atvejais mažesnis, nei Peano kreivės ilgis jungiantis tuos centrus, dėl ko pirmuoju atveju gaunami geresni viršutiniai rėžiai. Galiausiai naudojant Peano kreivės idėją „sudarkomos“ paprastos ir patogios optimizuoti daugiamatės funkcijos, pavyzdžiui iš lokaliojo optimizavimo uždavinio gali gautis globaliojo optimizavimo uždavinys.

1.4.2. Vieną viršutinio rėžio funkciją naudojantys algoritmai

Šios klasės algoritmai remiasi vienmačio Pijavskij algoritmo praplėtimu daugiamačiui atvejui.



1.8 pav. *Peano kreivės iliustracija dvimačiu* $(n = 2)$ *atveju*

A. Pijavskij ir Mladineo algoritmai

Daugiamačiu Euklidinės normos atveju Pijavskij tipo viršutinio rėžio funkcija $F_k(x)$ (1.6) yra mažiausias apvalkalas sudarytas iš n-mačių kūgių. Kiekvienoje iteracijoje surandamas viršutinio rėžio funkcijos globaliojo maksimumo taškas x_{k+1} (1.7), kuris pasirenkamas nauju iteracijos tašku. Tokiu būdu sudaromas nuoseklus konverguojantis iteracinis procesas, kurį nepriklausomai pasiūlė Pijavskij [99, 100] ir Shubert [111], pateiktas 2 algoritme. Algoritmas sustoja, kai skirtumas tarp φ tipo viršutinio rėžio ir surastos maksimalios reikšmės f_{opt} tampa mažesnis už ε.

Tačiau φ tipo viršutinio rėžio suradimas daugiamačiu atveju yra sudėtingas uždavinys. Pijavskij parodė [100] kaip šis taškas surandamas sprendžiant kvadratinių lygčių sistemas. Kiekvienos iš šių sistemų sprendinys yra lokalus viršutinio rėžio funkcijos maksimumo taškas, o maksimalus iš visų sprendinių - ieškomasis globalusis viršutinio rėžio funkcijos taškas, kartu ir naujasis iteracijos taškas.

Mladineo pasiūlė [82] efektyvesnį būdą, kaip šis taškas gali būti surastas sprendžiant lygčių sistemas sudarytas iš n tiesinių ir vienos kvadratinės lygties. Mladineo ir Pijavskij pasiūlyti algoritmai ieško visų n-mačių kūgių sankirtos taškų ir iš jų išrenka tašką, kuriame viršutinio rėžio reikšmė didžiausia. Strigul [113] pasiūlė algebrinį metodą, kaip išvengti dalies lygčių sistemų sprendimo, kurių sprendinys negali būti didžiausia viršutinio rėžio reikšmė.

2 algoritmas Pijavskij algoritmas

Pasirenkamas $x_1 \in \mathbf{I} \subseteq \mathbf{D}$
$\varphi \leftarrow +\infty$
$x_{opt} \leftarrow x_1$
$f_{opt} \leftarrow f(x_{opt})$
$k \leftarrow 1$
while $(\varphi - f_{opt} > \varepsilon)$ **do**
  $F_k(x) = \min\limits_{i=1,\ldots,k} \{f(x_i) + L\|x - x_i\|\}$
  $x_{k+1} \leftarrow \arg\max\limits_{x \in \mathbf{I}} F_k(x)$
  $\varphi \leftarrow F_k(x_{k+1})$
  if $(f(x_{k+1}) > f_{opt})$ **then**
    $f_{opt} \leftarrow f(x_{k+1})$
    $x_{opt} \leftarrow x_{k+1}$
  end if
  $k \leftarrow k + 1$
end while

B. Wood algoritmas

Wood pasiūlė [122] algoritmą standartinėms leistinosioms sritims $\mathbf{D}$, kurios apibrėžiamos tokiu būdu: tegu $u^1, \ldots, u^{n+1}$ yra vienetiniai vektoriai, tokie kad $u^1 + \ldots + u^{n+1} = 0$ ir $u^i \cdot u^j = -\frac{1}{n}$ su visais $i, j = 1, 2, \ldots, n+1$, kai $i \neq j$. Leistinoji sritis $\mathbf{D} \subset \mathbb{R}^n$ yra formos $c + rU$, čia $c \in \mathbb{R}^n$, $r \leq 0$ ir U yra vektorių suma $U^1 + \ldots + U^{n+1}$.

Pirmiausia leistinoji sritis $\mathbf{D}$ padengiama n-mačiu simpleksu $\mathbf{I}_0 = \mathbf{I}\left(x^0, y^0, h^0\right)$, kurio pagrindas $\left(x^0, y^0\right) \in \mathbb{R}^n \times \mathbb{R}$, o aukštis h^0 tenkina lygybes:

$$
\begin{aligned}
x^0 &= c + \frac{1}{L(n+1)} \sum_{k=1}^{n+1} \left(f_{opt} - f\left(c - ru^k\right)\right) u^k, \\
y^0 &= Lnr + \frac{1}{n+1} \sum_{k=1}^{n+1} f\left(c - ru^k\right), \\
h^0 &= \frac{1}{n+1} \sum_{k=1}^{n+1} \left(f\left(c - ru^k\right) - f_{opt}\right) + Lnr,
\end{aligned}
$$

čia $f_{opt} = \max\limits_{k=1,\ldots,n+1} f\left(c - ru^k\right)$ ir $y^0 + h^0 = f_{opt}$.

Kiekvienoje iteracijoje išrenkama n-mačių simpleksų $\mathbf{T}\left(x^j, y^j, h^j\right)$ aibė (simpleksų viršūnės priklauso tai pačiai plokštumai $x_{n+1} = f_{opt}$ iš $\mathbb{R}^{n+1}$) talpi-

nanti ieškomosios tikslo funkcijos globaliuosius maksimumo taškus. Simpleksas, kurio viršūnė (arba pagrindas) yra žemiausiame taške dalijamas į mažesnius simpleksus. Algoritmas sustoja, kai ε-optimali reikšmė yra surasta, t.y., kai $\max\limits_{j} y^j - f_{opt} \leq \varepsilon$.

1.4.3. Šakų ir rėžių algoritmai

Pastebėsime, kad antros klasės algoritmai gali būti sprendžiami kaip šakų ir rėžių algoritmai su ne visai tradicine šakojimosi procedūra.

A. Galperin kubinis algoritmas

Serijoje straipsnių [27, 38–41] Galperin pasiūlė kubinį algoritmą hiperkubinėms leistinosioms sritims $\mathbf{D} = [a, b]^n \subset \mathbb{R}^n$. Šio algoritmo šakų ir rėžių taisyklės yra:

Inicializavimo taisyklė. Pasirinktam sveikajam skaičiui $N \geq 2$, pradinis hiperkubas su diametru $\delta(\mathbf{D}) = \sqrt{n}(b-a)$ dalijamas į N^n mažesnių vienodo tūrio hiperkubų su diametrais lygiais $\delta(\mathbf{I}_j) = \sqrt{n}(b-a)/N$:

$$\mathbf{D} : \mathbb{I} \leftarrow \{\mathbf{I}_j | \mathbf{D} = \bigcup \mathbf{I}_j, j = 1, \ldots, N^n\}.$$

Apatinio rėžio (LB) taisyklė. Funkcijos reikšmės įvertinamos tik vienoje iš hiperkubo viršūnių $x^j \in V(\mathbf{I}_j)$. Maksimali iš šių įvertintų funkcijos reikšmių yra apatinis maksimumo rėžis:

$$LB(\mathbf{D}) = \max\{LB(\mathbf{D}), f(x^j)\}.$$

Hiperkubo viršūnė, kurioje įvertinama funkcijos reikšmė parenkama tokiu būdu:

$$x^j = x^0 + p_j, j = 1, \ldots, N^n,$$

čia vektorius $p_j = (p_{j_1}, \ldots, p_{j_n})$ parenkamas taip, kad x^j perbėga tik po vieną kiekvieno naujai gauto hiperkubo viršūnę. Pavyzdžiui, jei $\mathbf{D} = [0, 2]^2$ ir $N = 2$, tuomet po leistinosios srities $\mathbf{D}$ inicializavimo (padalijimo) gauname:

$$\mathbf{D} : \mathbb{I} \leftarrow \{\mathbf{I}_j | \mathbf{D} = \bigcup \mathbf{I}_j, j = 1, \ldots, 2^2\},$$

ir viršūnės, kuriose apskaičiuojamos funkcijos reikšmės yra:

$$x^1 = (0, 0), x^2 = (0, 1), x^3 = (1, 0), x^4 = (1, 1),$$

čia $x^j \in V(\mathbf{I}_j)$.
Viršutinio rėžio (UB) taisyklė. Straipsniuose [38, 39] autorius viršutinį rėžį siūlė skaičiuoti tokiu būdu:

$$UB(\mathbf{I}_j) = f(x^j) + L\delta(\mathbf{I}_j), \tag{1.12}$$

čia $x^j \in V(\mathbf{I}_j), \delta(\mathbf{I}_j) = \sqrt{n}(b-a)/N^k$, o k šakojimosi gylis. Vėlesniame straipsnyje autorius pastebi [40], kad geresnis viršutinio rėžio įvertis gaunamas

$$UB(\mathbf{I}_j) = f(x_c^j) + L\delta(\mathbf{I}_j)/2, \tag{1.13}$$

čia $x_c^j \in \mathbf{I}_j$ - hiperkubo centro taškas. Autorius taip pat pastebi, kad funkcijos reikšmė, apskaičiuota centro taške $f(x_c^j)$ pakartotinai gali būti panaudota tolesniame šakojimo procese tuo atveju jei N^n yra lyginis skaičius, kas nebuvo būtina pirminio algoritmo atveju.
Išrinkimo taisyklė. k-ajame algoritmo žingsnyje dalijimui išrenkami visi sąraše esantys hiperkubai $\mathbf{I}_j \in \mathbb{I}$.
Dalijimo taisyklė. Sąraše esantys hiperkubai $\mathbf{I}_j \in \mathbb{I}$ dalijami pagal tą pačią taisyklę, kaip ir inicializavimo etape, t.y. į N^n mažesnių vienodo tūrio hiperkubų.

Taigi kubinis algoritmas [38] kaip ir modifikuotasis [40] yra lengvai realizuojami, tačiau turi nemažai trūkumų. Pirmiausia, leistinoji sritis $\mathbf{D}$ turi būti hiperkubas. Tais atvejais kai leistinoji sritis ne hiperkubas ją reikia padengti hiperkubu. Jei leistinoji sritis - hiperstačiakampis, algoritmas nesunkiai gali būti pritaikytas, bet jis bus mažiau efektyvus, nei tokio paties tūrio hiperkubinėms sritims, dėl gaunamos ilgesnės įstrižainės. Algoritme pasiūlytas neefektyvus išrinkimo strategijos platyn variantas, nes kiekvienoje iteracijoje dalinant visus sąraše esančius hiperstačiakampius, sąrašo dydis greitai auga, ypač didėjant dimensijai n ar skaičiui N. Efektyviau realizavus išrinkimą, dalies hiperstačiakampių iš viso nereikėtų dalinti, jei k-ojoje iteracijoje išrenkamas dalijimui būtų ne visas sąrašas, o po vieną sąrašo elementą, kuriuose yra galimybė gauti gerą $LB(\mathbf{D})$ įvertį ir todėl atmesti dalį k-osios iteracijos likusio sąrašo elementų jų nedalinant.

B. Meewella ir Mayne algoritmas

Meewella ir Mayne [78] pasiūlytas šakų ir rėžių algoritmas hiperstačiakampėms sritims viršutinį rėžį apskaičiuoja sprendžiant tiesinio programavimo uždavinį.
Apatinio rėžio (LB) taisyklė. Funkcijos reikšmės įvertinamos visose hiperstačiakampių viršūnėse ir apatinis rėžis lygus maksimaliai funkcijos reikšmei,

1.2 lentelė. *Leistinosios srities viršūnių numeracija dvimačiu* ($n = 2$) *atveju*

m	1	2	3	4
$c(m)$	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
y_m	(a_1^j, a_2^j)	(a_1^j, b_2^j)	(b_1^j, a_2^j)	(b_1^j, b_2^j)

gautai vienoje iš hiperstačiakampio viršūnių $x^j \in V(\mathbf{I}_j)$:

$$LB(\mathbf{D}) = \max\{LB(\mathbf{D}), f(x^j)\}.$$

Viršutinio rėžio (UB) taisyklė. Išrinkto hiperstačiakampio $\mathbf{I}_j = [a_1^j, b_1^j] \times \cdots \times [a_n^j, b_n^j]$ viršūnės $y_m : m = 1, \ldots, 2^n$ sunumeruojamos tokiu būdu: tarkime, kad vektorius $c(m) = (c^1(m), \ldots, c^n(m))$ susideda iš skaičiaus $m-1$ dvejetainės išraiškos:

$$m - 1 = \sum_{i=0}^{n-1} 2^i c^{n-i}(m).$$

Tuomet viršūnės $y_m = (y_m^1, \ldots, y_m^n)$ koordinatės yra:

$$y_m^i = a_i^j[1 - c^i(m)] + b_i^j c^i(m), \quad i = 1, \ldots, n.$$

Pavyzdžiui, dimensijos $n = 2$ atveju $\mathbf{I}_j = [a_1^j, b_1^j] \times [a_2^j, b_2^j]$, $m = 1, \ldots, 2^2 = 4$ ir pagal aprašytą taisyklę gaunami vektoriai $c(m)$ bei jas atitinkančios viršūnės y_m pateiktos 1.2 lentelėje.

Apibrėžkime vektorius

$$g_m = L \sum_{i=1}^{n} u_i(m) e_i, \quad m = 1, \ldots, 2^n,$$

čia

$$u_i(m) = 2c^i(m) - 1$$

ir $e_i, i = 1, \ldots, n$ yra vienetiniai vektoriai. Tuomet 2^n tiesinių funkcijų:

$$h_m(x) = f(y_m) + (y_m - x)g_m$$

apriboja funkciją $f(x)$ iš viršaus posrityje $\mathbf{I}_j$. Todėl funkcijos $f(x)$ viršutinis rėžis posrityje $\mathbf{I}_j$ yra

$$UB(\mathbf{I}_j) = \max_{x \in \mathbf{I}_j} \min_{m=1,\ldots,2^n} h_m(x),$$

kuris surandamas sprendžiant tiesinio programavimo uždavinį $n+1$ kintamiesiems

$$\max z,$$

kai

$$z \leq f(y_m) + (y_m - x)g_m, \quad m = 1, \ldots, 2^n, \quad x \in \mathbf{I}_j.$$

Išrinkimo taisyklė. Išrenkamas hiperstačiakampis $\mathbf{I}_j \in \mathbb{I}$, kurio viršutinio rėžio įvertis $UB(\mathbf{I}_j)$ didžiausias.
Dalijimo taisyklė. Sąraše esantys hiperkubai $\mathbf{I}_j \in \mathbb{I}$ dalijami į hiperkubus gaunamus iš sankirtos visų hiperplokštumų, lygiagrečių koordinačių ašims ir einančių per tašką $x^* \in \mathbf{I}_j$, kur x^* optimalus aprašyto viršutinio rėžio skaičiavime tiesinio programavimo uždavinio sprendinys. Tokiu būdu po dalijimo gaunamų naujų hiperstačiakampių skaičius varijuoja nuo 2 iki 2^n.

Taigi Meewela ir Mayne algoritmas viršutinio rėžio skaičiavimui naudoja funkcijos reikšmes, gautas leistinosios srities visose viršūnėse, todėl gaunami viršutiniai rėžiai yra neblogesni, nei Galperin kubinio algoritmo atveju [38], kuris viršutinio rėžio skaičiavimui naudoja funkcijos reikšmės įvertį tik vienoje iš leistinosios srities viršūnių. Tačiau Meewela ir Mayne siūlomas viršutinio rėžių apskaičiavimas sprendžiant tiesinio programavimo uždavinius reikalauja daug laiko sąnaudų. Taip pat pasiūlyta dalijimo taisyklė yra neefektyvi, nes vienu metu gali gautis didelis skaičius naujų hiperstačiakampių posričių, kurių dalis galėtų būti iš karto atmesta, jeigu tie posričiai būtų gaunami ne iš karto, o nuosekliai.

C. Gourdin, Hansen ir Jaumard algoritmas

Gourdin, Hansen ir Jaumard siūlomas algoritmas [46] pasižymi nesudėtingomis rėžių, išrinkimo bei dalijimo taisyklėmis, kurios tuo pačiu yra ir efektyviausios iš visų apžvelgtų.
Apatinio rėžio (LB) taisyklė. Tikslo funkcijos reikšmės apskaičiuojamos posričių $\mathbf{I}_j$ centro taškuose $x_c^j = (\frac{a_1^j + b_1^j}{2}, \ldots, \frac{a_n^j + b_n^j}{2}) \in \mathbf{I}_j$. Maksimali iš šių reikšmių yra tikslo funkcijos $f(x)$ apatinis rėžis:

$$LB(\mathbf{D}) = \max\{LB(\mathbf{D}), f(x_c^j)\}.$$

Viršutinio rėžio (UB) taisyklė. Tegu x_c^j yra posričio $\mathbf{I}_j = [a_1^j, b_1^j] \times \cdots \times [a_n^j, b_n^j]$ centras. Tuomet viršutinis rėžis yra:

$$UB(\mathbf{I}_j) = f(x_j^c) + \frac{L}{2}[\sum_{i=1}^{n}(b_i^j - a_i^j)^2]^{\frac{1}{2}}. \tag{1.14}$$

Išrinkimo taisyklė. Išrenkamas hiperstačiakampis $\mathbf{I}_j \in \mathbb{I}$, kurio viršutinio rėžio įvertis $UB(\mathbf{I}_j)$ didžiausias.

Dalijimo taisyklė. Pažymėkime $b_t^j - a_t^j = \max\limits_{i=1,\ldots,n}(b_i^j - a_i^j)$ srities $\mathbf{I}_j$ ilgiausios briaunos ilgį. Tuomet hiperstačiakampis $\mathbf{I}_j$ dalijamas į p lygių hiperstačiakampių einančių per taškus, gautus briauną $b_t^j - a_t^j$ padalinus į p lygių dalių. Autoriai pastebi, kad efektyviausias dalijimas gaunamas su $p = 3$, nes tuomet gali būti pakartotinai panaudotos hiperstačiakampių centruose apskaičiuotos funkcijos reikšmės.

Toks algoritmas paprastas realizuoti, o funkcijų skaičiavimų kiekis, po k skirtingų sričių rėžių apskaičiavimų yra tik $\frac{2}{3}k$, nes pakartotinai panaudojama dalis anksčiau apskaičiuotų funkcijos reikšmių.

Pastebėsime, kad panašų į Gourdin, Hansen ir Jaumard algoritmą, 1988 m. pasiūlė Evtushenko ir Rat'kin [26], tik jų siūlomame algoritme viršutinių rėžių skaičiavimui vietoj Euklidinės naudojama begalinė norma, o apatiniai rėžiai srityse pagerinami lokaliojo optimizavimo metodais.

1.4.4. Lipšico optimizavimo algoritmų apibendrinimas

Disertacijoje pagrindinis dėmesys skiriamas Lipšico optimizavimui, todėl išsamiau apžvelgta tik šios klasės algoritmai. Išsamios pastarojo dešimtmečio globaliojo optimizavimo metodų apžvalgos pateiktos [30, 32].

Pirmos klasės Lipšico algoritmai eksperimentiškai nebuvo realizuoti, dėl aptartų akivaizdžių jų trūkumų, lyginant su kitų dviejų klasių algoritmais. Antros klasės algoritmai remiasi Pijavskij idėjos realizavimu, t.y. visoje srityje naudoja bendrą viršutinio rėžio funkciją. Šios klasės algoritmai labiau tinkami spręsti tuomet, kai reikalaujamas didelis tikslumas arba kai tikslo funkcijos reikšmių apskaičiavimas yra brangus. Tačiau kai erdvės dimensija $n > 2$ arba kai tikslo funkcijos Lipšico konstanta yra didelė, šios klasės algoritmai nėra efektyvūs. Trečios klasės, šakų ir rėžių algoritmai, naudoja kiekvienam posričiui atskiras viršutinio rėžio funkcijas. Su šios klasės algoritmais beveik visuomet galima išspręsti reikiamu tikslumu $n = 3$ dimensijos Lipšico optimizavimo uždavinius. Aktualūs praktiniai uždaviniai trečios klasės algoritmais išspręsti $n = 5$ dimensijos atveju [51, 93, 122]. Pažymėtina, kad $n = 2$ atveju trečiosios klasės

algoritmais surasti sprendinį reikalingas didesnis funkcijų skaičiavimų kiekis, todėl jie yra mažiau efektyvūs brangių tikslo funkcijų atveju.

1.5. Lygiagretieji skaičiavimai

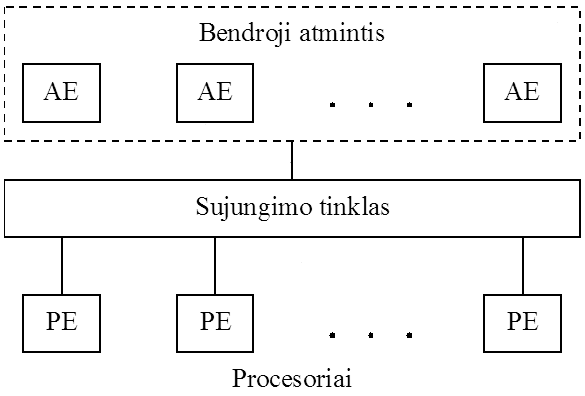
Lygiagretieji skaičiavimai šiuo metu yra viena iš labiausiai vystomų mokslo ir technologijų sričių. Lygiagretiesiems skaičiavimams skirtus klasterius turi nemaža dalis Lietuvos universitetų bei mokslo institutų. Sukurtas ir vystomas „Lietuvos lygiagrečių ir paskirstytų skaičiavimų ir e-paslaugų tinklas (LitGrid)“, turintis virš 100 naudotojų, jungiantis apie 490 procesorių bei suteikiantis naudotojams skaičiavimų ir e-paslaugų galimybes. Lygiagretiesiems skaičiavimams skirtų knygų anglų kalba – labai daug, tačiau lietuvių kalba literatūros vis dar trūksta, o iš esančios paminėtinas R. Čiegio vadovėlis [11]. Lygiagrečiųjų skaičiavimų srityje Lietuvoje apginama vis daugiau disertacijų [2, 59, 60, 92, 104, 132].

1.5.1. Lygiagretieji kompiuteriai

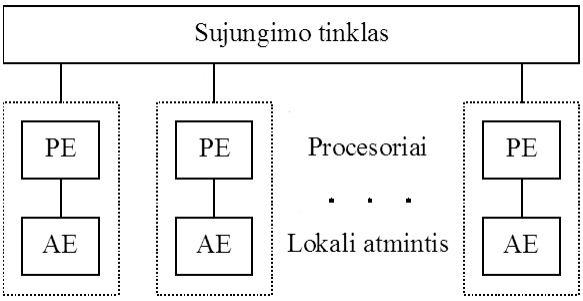
Kai įprastų nuosekliųjų kompiuterių pajėgumo uždaviniui išspręsti neužtenka, vilčių teikia lygiagretieji kompiuteriai. Lygiagrečiosiose sistemose tą patį uždavinį tuo pačiu metu gali spręsti keli procesoriai, todėl sutrumpėja bendra sprendimo trukmė. Lygiagretieji kompiuteriai yra skirstomi į bendrosios atminties ir paskirstytosios atminties lygiagrečiuosius kompiuterius. Bendrosios atminties lygiagretieji kompiuteriai turi vieną bendrą atminties bloką ir visi procesoriai gali tiesiogiai pasiekti visas atminties vietas (žr. 1.9 pav.). Procesoriai bendrauja tarpusavyje per bendrąsias duomenų struktūras, esančias bendrojoje atmintyje.

Paskirstytosios atminties lygiagrečiajame kompiuteryje kiekvienas procesorius gali tiesiogiai perskaityti ir įrašyti tik jo lokaliojoje atmintyje esančius duomenis (1.10 pav.). Procesoriai bendrauja tarpusavyje siųsdami vienas kitam pranešimus.

Pageidautina, kad ta pati programa galėtų būti vykdoma abiejų tipų lygiagrečiaisiais kompiuteriais. Bendrosios atminties kompiuteryje emuliuoti paskirstytosios atminties kompiuterį, panaudojant bendrąją atmintį persiunčiamiems pranešimams, nesunku. Kur kas sunkiau paskirstytosios atminties kompiuteryje emuliuoti bendrosios atminties kompiuterį. Dėl šios priežasties universalus lygiagrečiųjų skaičiavimų modelis buvo realizuotas pranešimais bendraujančiomis programomis pagal skaičiavimų paskirstytosios atminties kompiuteriuose modelį. Todėl universalaus modelio skaičiavimai gali būti vykdomi



1.9 pav. *Lygiagretusis bendrosios atminties kompiuteris*



1.10 pav. *Lygiagretusis paskirstytosios atminties kompiuteris*

abiejų tipų lygiagrečiaisiais kompiuteriais. Standartinė pranešimų persiuntimo biblioteka MPI [36] naudojama pranešimų persiuntimui.

1.5.2. Užduočių paskirstymas ir baigties nustatymas

Algoritmo plėtojimas, įgalinantis jo dalis vykdyti lygiagrečiai, vadinamas lygiagretinimu. Lygiagretinimo tikslas – sutrumpinti uždavinio sprendimo trukmę. Idealiu atveju procesoriai vykdytų skaičiavimus nepertraukiamai ir baigtų darbą tuo pačiu laiko momentu. Siekiama užduotis paskirstyti taip, kad procesorių darbo apimtis būtų vienoda. Nustatymas, kad skaičiavimai yra užbaigti, vadinamas baigties nustatymu.

Pradiniu (statiniu) užduočių paskirstymo atveju užduotys yra paskirstomos prieš skaičiavimus ir vėliau neperskirstomos. Toks paskirstymas dar vadinamas tvarkaraščio sudarymu. Jei pradinių užduočių skaičius yra lygus procesorių

skaičiui, kiekvienas procesorius gauna po užduotį. Jei užduočių yra daugiau, naudojami tvarkaraščių sudarymo metodai. Kadangi užduotys neperskirstomos tarp procesorių, nustatyti baigtį nesudėtinga. Šios schemos trūkumas yra tas, kad procesoriai gali užbaigti skaičiavimus skirtingu metu, o bendra sprendimo trukmė apsprendžiama ilgiausiai skaičiavusio procesoriaus darbo trukme.

Dinaminiais užduočių paskirstymo algoritmais siekiama tolygiai perskirstyti užduotis skaičiuojant. Centralizuoto dinaminio užduočių perskirstymo atveju vienas procesorius, dažniausiai vadinamas šeimininku, paskirsto užduotis. Tai atitinka paradigmą šeimininkas-darbininkai. Procesorius šeimininkas saugo užduočių rinkinį, kuris vadinamas darbo fondu arba užduočių eile. Kai procesorius darbininkas neturi darbo, jis prašo užduoties. Procesorius šeimininkas išsiunčia sudėtingiausią arba svarbiausią turimą užduotį. Baigęs užduotį, procesorius darbininkas išsiunčia rezultatą šeimininkui. Procesorius šeimininkas nustato, kada skaičiavimai yra baigti. Šios schemos trūkumas yra tas, kad procesorius šeimininkas gali nespėti aptarnauti darbininkų, ypač kai procesorių yra daug, o bendraujama dažnai. Tokiu atveju procesoriai darbininkai turi laukti, kol gaus darbo.

Paskirstyto dinaminio užduočių perskirstymo atveju procesoriai apsikeičia užduotimis tarpusavyje. Siekiama užduotis perskirstyti kaip galima tolygiau. Procesoriai ne tik atlieka skaičiavimus, bet ir apsikeičia užduotimis. Užduočių perskirstymas užima laiko. Jei jis trunka ilgiau, negu sutaupoma uždavinio sprendimo laiko, dinaminio užduočių perskirstymo taikyti neverta. Be to, perskirstymui kartais trūksta užduočių.

Gali būti sunku nustatyti, kada baigiami paskirstyti dinaminiai skaičiavimai. Bendros paskirstytų skaičiavimų baigties sąlygos yra šios:

- Visi procesoriai baigė savo užduotis. Tai lokali baigties sąlyga.
- Nėra persiunčiamų užduočių. Persiunčiama užduotis gali aktyvinti skaičiavimus.

Algoritmams su paskirstytu dinaminiu užduočių perskirstymu naudojamas bendras baigties nustatymo metodas su užklausų ir patvirtinimų pranešimais.

Disertacijoje pateikiami lygiagretieji šakų ir rėžių algoritmai realizuoti naudojant statinį užduočių paskirstymą.

1.5.3. Lygiagrečiųjų algoritmų vertinimo kriterijai

Lygiagretieji algoritmai analizuojami naudojant lygiagretinimo koeficientus [37]. Dažniausiai naudojamas lygiagrečiojo algoritmo spartinimo koeficientas:

$$s_p = \frac{t_1}{t_p},$$

čia t_p yra algoritmo sprendimo, naudojant p procesorių, trukmė. Spartinimo koeficientas, padalytas iš procesorių skaičiaus, yra vadinamas algoritmo efektyvumo koeficientu:

$$e_p = \frac{s_p}{p}.$$

Paprastai $1 \leq s_p \leq p$ ir $0 \leq e_p \leq 1$. Tačiau esant skirtingam procesorių skaičiui lygiagrečiojo šakų ir rėžių algoritmo vykdymas gali skirtis posričių peržiūros tvarkos ir skaičiaus prasme. Pernelyg dideli skirtumai sukelia anomalijas:

- nuostolingą anomaliją $s_p < 1$; peržiūrima daugiau posričių negu vieno procesoriaus atveju;
- lėtinimo anomaliją $s_{p_1} > s_{p_2}$, kai $p_1 < p_2$; didėjant procesorių skaičiui didėja peržiūrimų posričių skaičius;
- greitinimo anomaliją $s_p > p$; peržiūrima mažiau posričių, negu vieno procesoriaus atveju.

Esant anomalijoms, spartinimo ir efektyvumo koeficientai netinka algoritmams vertinti. Anomalijos mažiau įtakoja pseudo efektyvumo koeficientą:

$$pe_p = \frac{t_1 / |T_1|}{p \times t_p / |T_p|},$$

čia T_p yra darbo kiekio matas naudojant p procesorių ir laiko panaudojimo skaičiavimams proporcijos koeficientą. Geras darbo kiekio matas turi būti naudojamas, kai naudojamas pseudo efektyvumo koeficientas.

Laiko panaudojimo skaičiavimams proporcijos koeficientas dar vadinamas procesorių panaudojimo koeficientu. Tai vidutinė procesorių laiko, panaudoto skaičiavimams, proporcija; kitas laikas panaudojamas bendravimui tarp procesorių arba laukimui. Šis koeficientas visada yra tarp 0 ir 1.

1.5.4. Lygiagrečiųjų šakų ir rėžių algoritmų klasifikacija

Literatūroje išskiriami įvairūs lygiagrečiųjų šakų ir rėžių algoritmų klasifikavimo būdai: pagal tai, kuri algoritmo dalis vykdoma lygiagrečiai, ar pro-

cesoriai apsikeičia duomenimis, dirba sinchronizuotai ar asinchroniškai, kiek neištirtų užduočių sąrašų kuriama ir pan. Viena dažniausiai naudojamų lygiagrečiųjų šakų ir rėžių algoritmų klasifikacijų pasiūlyta [43]. Autoriai išskiria tris pagrindinius būdus kurti lygiagrečiuosius šakų ir rėžių algoritmus:

- Algoritmai, realizuoti 1 tipo lygiagretumo būdu, operacijas su kiekviena užduotimi atlieka atskirai. Pavyzdžiui, tiriamos funkcijos reikšmės ar rėžių skaičiavimai vykdomi lygiagrečiai. Tokio tipo lygiagretumas neturi įtakos bendrai šakų ir rėžių algoritmo struktūrai ir priklauso tik nuo duoto uždavinio.
- Algoritmuose, realizuotuose 2 tipo lygiagretumo būdu, šakų ir rėžių paieškos medis konstruojamas lygiagrečiai, t.y. lygiagrečiai atliekami veiksmai su keliomis užduotimis. Tokio tipo lygiagretumas gali įtakoti algoritmo vykdymą ir užduočių tyrimo tvarką. Šio tipo algoritmai, papildomai klasifikuojami [43], atsižvelgiant į realizavimo techniką:
  - sinchronizuoti,
  - asinchroniniai,

  bei kiek neištirtų užduočių sąrašų kuriama:
  - viena neištirtų užduočių eilė,
  - kelios neištirtų užduočių eilės.

  Sinchronizuoti algoritmai realizuoti fazėmis. Fazių metu procesoriai dirba nepriklausomai ir apsikeičia duomenis tik tarp fazių. Asinchroniniuose algoritmuose procesoriai gali apsikeisti duomenimis bet kuriuo metu. Algoritmai naudojantys vieną neištirtų užduočių eilę, naudojasi viena bendra atmintimi. Algoritmai naudojantys kelias eiles, neištirtas užduotis saugo keliose atminties vietose.
- 3 tipo lygiagretumo algoritmuose keli šakų ir rėžių paieškos medžiai yra kuriami vienu metu. Šiuos medžius charakterizuoja skirtingos operacijos (rėžio, šakos, atmetimo, pasirinkimo iš eilės). Rezultatai, gauti konstruojant vieną medį gali būti panaudoti konstruojant kitą.

1.6. Lipšico algoritmų testavimas

Programinės įrangos, skirtos globaliojo optimizavimo uždavinių sprendimui, apžvalga pateikta [97]. Išsamiausi globaliojo optimizavimo metodų efektyvumo tyrimų rezultatai pateikti [85]. Spręsta virš 1000 iš literatūros parinktų

globaliojo optimizavimo testinių uždavinių, kurių didžiausią dalį sudaro uždaviniai iš [34].

Disertacijoje eksperimentiniuose tyrimuose naudojami tie patys testiniai uždaviniai, kurie naudoti lyginant Lipšico optimizavimo algoritmus [49]. Šis testinių uždavinių rinkinys papildytas tipiniais didesnio matmenų skaičiaus ($n \geq 4$) testiniais uždaviniais iš [63, 74]. Dviejų ir trijų matmenų tikslo funkcijos pavadintos analogiškai šaltiniuose [49, 74] pateiktiems vardams (funkcijų numeriams). Didesnio matmenų skaičiaus ($n \geq 4$) funkcijoms suteikiami vardai iš [63, 74] literatūros šaltinių. Eksperimentiniuose tyrimuose naudojamų testinių uždavinių eilės numeriai (Nr.), vardai (Funkcijos vardas), dimensija (n), bei optimizavimo tikslumas (ε) pateikti 1.3 lentelėje. Dvimačiai ir trimačiai optimizavimo uždaviniai sprendžiami tuo pačiu tikslumu, kaip ir kitų autorių darbuose, su kuriais yra lyginami rezultatai. Kita papildoma informacija: leistinoji sritis $\mathbf{D}$, funkcijų analitinės išraiškos, dalinės išvestinės $f'_{x_i}, i = 1, \ldots, n$, globalieji optimumo taškai x^* bei funkcijos reikšmės juose $f(x^*)$ pateikti priede „A. Testiniai optimizavimo uždaviniai“. Aptariant eksperimentiniuose tyrimuose gautus rezultatus, funkcijų vardo, optimizavimo paklaidos, matmens bei leistinosios srities stulpeliai neįtraukiami į eksperimentinių rezultatų lenteles, nes visuose tyrimuose jie nekinta.

Nuoseklieji algoritmai vertinami naudojant funkcijų skaičiavimo kiekio, optimizavimo laiko, ištirtų posričių skaičiaus bei maksimalaus kandidatų sąrašo kriterijus. Pagrindinis disertacijoje naudojamas algoritmų efektyvumo kriterijus yra tikslo funkcijų skaičiavimų kiekis. Lygiagrečiosios algoritmų versijos vertinamos naudojant tradicinius kriterijus: spartinimo koeficientą bei lygiagretinimo efektyvumą.

1.7. Pirmojo skyriaus apibendrinimas

1. Globaliojo optimizavimo metodai, pagrįsti Lipšico rėžių apskaičiavimu, yra gerai žinomi ir plačiai taikomi įvairių optimizavimo uždavinių sprendimui. Tačiau daugiamačiai Lipšico optimizavimo metodai tiek teorine, tiek ir praktine prasme yra gerokai mažiau ištirti nei vienmačiai.
2. Daugiamačiai Lipšico optimizavimo algoritmai dažniausiai naudoja Pijavskij arba μ tipo rėžius. Tačiau Pijavskij rėžius naudojantys algoritmai nėra efektyvūs kai erdvės dimensija $n > 2$ arba kai tikslo funkcijos Lipšico konstanta yra didelė, o naudojantys μ tipo rėžius mažiau tinkami, kai tikslo funkcijos reikšmių apskaičiavimas yra brangus.

3. Lipšico optimizavime dažniausiai yra naudojama Euklidinė norma. Tačiau nėra ištirta, kaip kinta optimizavimo rezultatai naudojant kitas normas.
4. Dažniausiai daugiamačiai Lipšico metodai naudoja stačiakampius posričius, o simpleksiniams posričiams skirtas mažesnis dėmesys.
5. Šakų ir rėžių algoritmai dažniausiai naudoja paieškos geryn strategiją, tačiau nėra pakankamai ištirta ir palyginta, kaip kinta optimizavimo rezultatai naudojant kitas paieškos strategijos.

1.3 lentelė. *Testiniai optimizavimo uždaviniai*

Nr.	Funkcijos vardas	n	ε
1.	1. [49]	2	0,355
2.	2. [49]	2	0,0446
3.	3. [49]	2	11,9
4.	4. [49]	2	0,0141
5.	5. [49]	2	0,1
6.	6. [49]	2	44,9
7.	7. [49]	2	542,0
8.	8. [49]	2	3,66
9.	9. [49]	2	62900
10.	10. [49]	2	0,691
11.	11. [49]	2	0,335
12.	12. [49]	2	0,804
13.	13. [49]	2	6,92
14.	20. [49]	3	2,12
15.	21. [49]	3	0,369
16.	23. [49]	3	8,333
17.	24. [49]	3	0,672
18.	25. [49]	3	0,0506
19.	26. [49]	3	4,51
20.	Rosenbrock [74]	3	2500
21.	Levy 15 [63]	4	L_2
22.	Rosonbrock [74]	4	L_2
23.	Shekel 5 [63]	4	L_2
24.	Shekel 7 [63]	4	L_2
25.	Shekel 10 [63]	4	L_2
26.	Schwefel 1.2 [63]	4	L_2
27.	Powell [63]	4	L_2
28.	Levy 9 [63]	4	L_2
29.	Levy 16 [63]	5	$1,5L_2$
30.	Rosenbrock [74]	5	$1,5L_2$
31.	Levy 10 [63]	5	$1,5L_2$
32.	Levy 17 [63]	6	$4L_2$
33.	Rosenbrock [74]	6	$4L_2$

2

Lipšico optimizavimas su simpleksiniais posričiais

Lipšico optimizavime dažniausiai naudojama Euklidinė norma. Vienmačiu atveju visos l_p normos yra lygios, bet daugiamačiu atveju - ne. Todėl Lipšico rėžiai priklauso nuo naudojamų normų ir neeuklidinių normų naudojimas gali būti prasmingas. Tačiau tai nėra pakankamai ištirta.

Koks ryšys sieja normas ir Lipšico konstantas? Kokias normas tikslingiausia būtų naudoti simpleksiniuose posričiuose? Kaip paprasčiau tam tikro tipo rėžius apskaičiuoti? Kokio tipo rėžius pasirinkti? O gal tikslingiau naudoti naujo tipo rėžius? Kituose poskyriuose pateiksime teorinius atsakymus į šiuos klausimus.

2.1. Normų ir jas atitinkančių Lipšico konstantų ryšys

Skirtingų normų panaudojimui būtina apibrėžti sąryšį tarp naudojamos normos ir Lipšico konstantos.

2.1 teiginys (A1). *Tegu* $\mathbf{D} \subset \mathbb{R}^n$ *yra iškiloji uždaroji aibė,* $f(x) : \mathbf{D} \to \mathbb{R}$ *tolygiai diferencijuojama funkcija. Tuomet* $f(x)$ *yra Lipšico funkcija ir* $\forall x_1, x_2 \in \mathbf{D}$ *teisinga nelygybė*

$$|f(x_1) - f(x_2)| \leq L_p \|x_1 - x_2\|_q, \tag{2.1}$$

čia $L_p = \max\{\|\nabla f(x)\|_p : x \in \mathbf{D}\}$ *yra Lipšico konstanta ir* $1/p + 1/q = 1$, $1 \le p, q \le \infty$.

Įrodymas. *Bet kuriems* $x_1, x_2 \in \mathbf{D}$ *egzistuoja* $z = x_2 + \lambda(x_1 - x_2) \in \mathbf{D}, 0 < \lambda < 1$, *toks kad*

$$|f(x_2) - f(x_1)| = |\nabla f(z) \cdot (x_2 - x_1)|,$$

(iš vidurinių reikšmių teoremos)

$$\le \sum_{i=1}^{n} \left| \frac{\partial f}{\partial z_i}(x_{2_i} - x_{1_i}) \right| \le \left(\sum_{i=1}^{n} \left| \frac{\partial f}{\partial z_i} \right|^p \right)^{1/p} \left(\sum_{i=1}^{n} |x_{2_i} - x_{1_i}|^q \right)^{1/q},$$

čia $1/p + 1/q = 1$, $1 \le p, q \le \infty$ *(iš Hölderio nelygybės).*
Kadangi

$$\|\nabla f(z)\|_p = \left(\sum_{i=1}^{n} \left| \frac{\partial f}{\partial z_i} \right|^p \right)^{1/p}, \quad \|x_1 - x_2\|_q = \left(\sum_{i=1}^{n} |x_{1_i} - x_{2_i}|^q \right)^{1/q},$$

todėl

$$|f(x_2) - f(x_1)| \le \|\nabla f(z)\|_p \|x_1 - x_2\|_q \le \max\{\|\nabla f(z)\|_p : z \in \mathbf{D}\} \|x_1 - x_2\|_q.$$

Pažymėję $L_p = \max\{\|\nabla f(z)\|_p : z \in \mathbf{D}\}$ *gauname (2.1) nelygybę.* □

Iš čia seka, kad naudojant l_q normą, Lipšico konstanta yra lygi tikslo funkcijos gradiento l_p normos supremumui.

2.2. Lipšico konstantų apskaičiavimas

1.3.1 poskyryje pateikta Lipšico algoritmų klasifikacija pagal Lipšico konstantos prigimtį. Disertacijoje siūlomus Lipšico optimizavimo algoritmus galima priskirti tiek pirmos, tiek antros grupės algoritmams. Siūlomi algoritmai remiasi prielaida, kad Lipšico konstanta žinoma iš anksto, tačiau kai ji nėra žinoma, tuomet prieš algoritmą surandamas globalusis konstantos įvertis.

Lipšico konstantų L_p įvertis surandamas panaudojus tinklelio paieškos strategiją. Tuo tikslu leistinoji sritis padengiama 1000^n leistinosios srities taškais $n = 2, 3$ atveju (kaip siūloma [49]) ir 100^n taškais $n \ge 4$ atveju bei surandamas taškas (vektorius) $z \in \mathbf{D}$, kuriame tikslo funkcijos gradiento l_p norma didžiausia. Apskaičiuoti Lipšico konstantų įverčiai $L_p : p = 1, 2, \infty$ bei optimalūs

2.1 lentelė. *Lipšico konstantų įverčiai dviejų ir trijų ($n = 2, 3$) matmenų tikslo funkcijoms naudojant l_∞, l_2, l_1 normas*

	Lipšico konstanta				Optimalus vektorius		
Nr.	L_1	L_2	L_2[49]	L_∞	z_1	z_2	z_3
1	50,27	50,27	50,2665	50,27	1,000	1,000	–
2	7,98	6,32	6,3183	6,00	1,000	0,380	–
3	141,34	112,44	112,44	107,09	0,000	0,000	–
4	2,00	2,00	2	2,00	1,000	1,000	–
5	2,00	1,42	$\sqrt{2}$	1,00	0,000	0,000	–
6	1284,80	1059,59	1059,59		4,000	4,000	–
				1034,00	-2,000	4,000	–
7	14708,00	12781,70	12781,7	12608,00	-3,000	-1,500	–
8	122,00	86,32	86,3134	63,00	-2,500	4,500	–
9	2639040	2225890	2225890	2177520	-2,000	2,000	–
10	24,00	17,03	17,034	13,04	4,000	-3,000	–
11	64,14	47,55	47,426	42,16	1,000	2,000	–
12	80,40	56,85	56,852	40,40	1,000	1,000	–
13		993,72	988,82	993,72	0,010	0,458	–
	995,29				0,010	0,020	–
14	600,00	346,41	244,95	200,00	1,000	1,000	0,000
15		18,33	42,626	18,32	0,091	0,556	0,990
	21,36				0,202	0,444	0,970
16		988,79	971,59	988,79	0,010	0,130	0,250
	991,05				0,010	0,020	0,210
17	33,52	19,39	19,39		4,000	3,528	4,000
				14,80	1,177	4,000	4,000
18	4,77	2,92	2,919	2,38	1,000	1,000	1,000
19	224,00	130,00	130,0	80,00	2,000	-2,000	2,000
20	33616,00	22267,90	–	16808,00	-3,000	-3,000	-3,000

vektoriai $z = (z_1, \ldots, z_n)$ kuriuose surastos Lipšico konstantos, pateikti 2.1 ir 2.2 lentelėse.

Euklidinę normą l_2 atitinkančios Lipšico konstantos L_2 gautieji įverčiai dviejų ir trijų matmenų tikslo funkcijoms palyginti su autorių [49] pateikiamais. Daugumai tikslo funkcijų apskaičiuoti L_2 įverčiai sutampa arba skiriasi nežymiai lyginant su [49]. Tik 14 ir 15 tikslo funkcijoms disertacijoje pateikiami įverčiai skiriasi iš esmės, tačiau tikėtina, kad disertacijoje gautieji Lipšico konstantų įverčiai šioms funkcijoms yra tikslesni. Pastebime, kad 14-tos funkcijos visų dalinių išvestinių reikšmės absoliutiniu didumu taške $(1, 1, 0) \in \mathbf{D} = [0, 1]^3$

2.2 lentelė. *Lipšico konstantų įverčiai daugiamatėms* ($n \geq 4$) *tikslo funkcijoms naudojant* l_∞, l_2, l_1 *normas*

	Lipšico konstanta			Optimalus vektorius					
Nr.	L_1	L_2	L_∞	z_1	z_2	z_3	z_4	z_5	z_6
21	1370,2	1196,4	1195,5	-10,00	-10,00	-10,00	-9,58	–	–
22	108720	60402	36026	4,00	-4,00	-4,00	-4,00	–	–
23	204,1	102,4	56,1	10,00	0,00	0,00	0,00	–	–
24	300,1	151,5	86,1	0,00	0,00	10,00	0,00	–	–
25	408,2	204,5	110,8	0,00	0,00	10,00	0,00	–	–
26	600,0	313,7	200,0	10,00	10,00	10,00	10,00	–	–
27	92216	48252		-4,00	-4,00	5,00	5,00	–	–
			29270	5,00	5,00	-4,00	-4,00	–	–
28	26,1	14,4	8,3	-10,00	-10,00	-10,00	-10,00	–	–
29		370,7	369,7	-5,00	-5,00	-5,00	-5,00	-5,00	–
	476,7			-5,00	-5,00	-5,00	-4,92	-4,58	–
30	264385	129425	66032	5,00	-5,00	-5,00	-5,00	-5,00	–
31	34,4	16,6	8,3	-10,00	-10,00	-10,00	-10,00	-10,00	–
32		370,9	369,7	-5,00	-5,00	-5,00	-5,00	-5,00	-4,59
	488,7			-5,00	-5,00	-5,00	-5,00	-4,92	-4,58
33	546547	240918	109238	6,00	-6,00	-6,00	-6,00	-6,00	-6,00

lygios 200, todėl iš (2.1) gauname

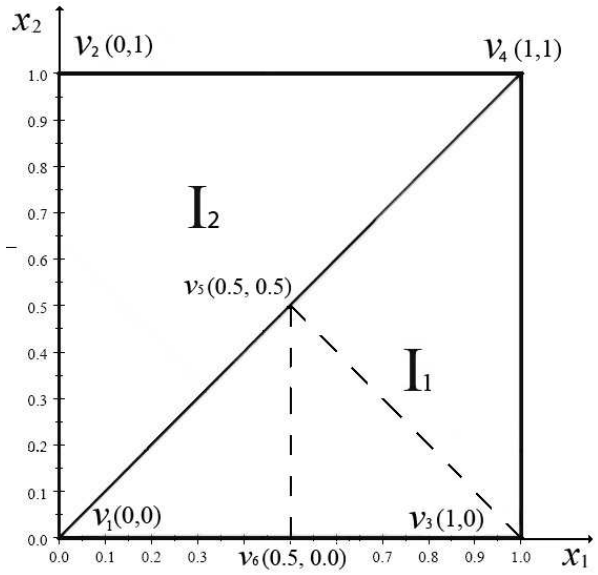
$$L_2 = \|\nabla f(1,0;1,0;0,0)\|_2 = \sqrt{200^2 + 200^2 + 200^2} \approx 346,41.$$

2.3. μ tipo rėžių tikslinimas parenkant l_p normas

Apibrėžus ryšį tarp normų ir jas atitinkančių Lipšico konstantų atlikime teorinius įvairių normų ir jas atitinkančių Lipšico konstantų efektyvumo tyrimus simpleksiniuose posričiuose. Paprastumo tikslu analizuojamos (vienetinės) hiperkubinės leistinosios sritys, o simpleksiniai posričiai gaunami atlikus leistinosios srities viršūnių trianguliaciją, aprašytą 1.2 poskyryje.

2.3.1. Dvimatis atvejis

Dvimačiu atveju leistinoji sritis $\mathbf{D} = [0,1]^2$ yra (vienetinis) kvadratas, kuris panaudojant viršūnių trianguliaciją padalijamas į 2 dvimačius simpleksus $\mathbf{I}_1 = [v_1, v_3, v_4], \mathbf{I}_2 = [v_1, v_2, v_4]$ (žr. 2.1 pav.). Laužtiniuose skliaustuose pažymima,



***2.1* pav.** *Kvadrato dalijimas į simpleksus*

kokios viršūnės sudaro konkretų simpleksą. Šakų ir rėžių pagrindu realizuoti Lipšico optimizavimo algoritmai dažniausiai viršutinius rėžius skaičiuoja pagal vieną iš μ tipo formulių: μ_1 (1.9) arba μ_2 (1.11). Simpleksiniuose posričiuose naudojant μ_2 tipo rėžius, gaunami tikslesni rėžiai, o funkcijų skaičiavimo kiekis skaičiuojant μ_2 tipo rėžį simplekse toks pat, kaip ir μ_1 tipo atveju. Dėl to koncentruosimės į μ_2 tipo rėžius.

Laisvai pasirinkime vieną iš simpleksų, tarkime $\mathbf{I}_1$. Apskaičiuokime kam lygus μ_2 tipo viršutinis rėžis:

$$\mu_2(\mathbf{I}_1) = \min_{v_i \in V(\mathbf{I}_1)} \left\{ f(v_i) + L \max_{x \in V(\mathbf{I}_1)} \|x - v_i\| \right\}. \tag{2.2}$$

Skaičiuojant μ_2 tipo rėžį viršūnėje v_1 (arba v_4) maksimalus atstumas

$$\max_{x \in V(\mathbf{I}_1)} \|x - v_1\|$$

yra tarp viršūnių v_1 ir v_4, kuris priklausomai nuo normos yra lygus:

$$\begin{aligned}\|v_1 - v_4\|_1 &= (|0-1| + |0-1|) = 2,\\ \|v_1 - v_4\|_q &= (|0-1|^q + |0-1|^q)^{\frac{1}{q}} = 2^{\frac{1}{q}}, \quad 1 < q < \infty,\\ \|v_1 - v_4\|_\infty &= \max\{|0-1|, |0-1|\} = 1.\end{aligned}$$

Pasinaudoję (2.1) gauname, kad μ_2 tipo rėžio apskaičiavimui reikalinga sandauga

$$L_p \max_{x \in V(\mathbf{I}_1)} \|x - v_i\|_q$$

priklausomai nuo normos ir ją atitinkančios Lipšico konstantos lygi:

$$\begin{aligned}L_\infty\|v_1 - v_4\|_1 &= \max\left\{|f'_{x_1}(z_1)|, |f'_{x_2}(z_2)|\right\} \cdot 2,\\ L_p\|v_1 - v_4\|_q &= \left(|f'_{x_1}(z_1)|^p + |f'_{x_2}(z_2)|^p\right)^{\frac{1}{p}} \cdot 2^{\frac{1}{q}}, \quad \frac{1}{p} + \frac{1}{q} = 1, 1 < p, q < \infty,\\ L_1\|v_1 - v_4\|_\infty &= \left(|f'_{x_1}(z_1)| + |f'_{x_2}(z_2)|\right) \cdot 1,\end{aligned}$$

čia $z = (z_1, z_2) \in \mathbf{D}$. Iš 2.1 ir 2.2 lentelių pastebime, kad leistinosios srities taškai z, kuriuose surastos maksimalios Lipšico konstantos naudojant skirtingas l_p normas, ne visada sutampa. Paprastumo tikslu tariame, kad jos sutampa. Pasinaudoję lema

2.2 Lema ([90]). *Jei $p > 0$, tai su visais $a, b \geq 0$ teisinga nelygybė*

$$(a+b)^p \leq \max\{1, 2^{p-1}\}(a^p + b^p),$$

gauname

$$\left(|f'_{x_1}(z_1)| + |f'_{x_2}(z_2)|\right) \cdot 1 \leq 2^{\frac{p-1}{p}} (|f'_{x_1}(z_1)|^p + |f'_{x_2}(z_2)|^p)^{\frac{1}{p}}. \tag{2.3}$$

Išreiškę $q = \frac{p}{p-1}$ bei įstatę į (2.3) gauname

$$\begin{aligned}\left(|f'_{x_1}(z_1)| + |f'_{x_2}(z_2)|\right) \cdot 1 &\leq (|f'_{x_1}(z_1)|^p + |f'_{x_2}(z_2)|^p)^{\frac{1}{p}} \cdot 2^{\frac{1}{q}}\\ &\leq \max\left\{|f'_{x_1}(z_1)|, |f'_{x_2}(z_2)|\right\} \cdot 2,\end{aligned}$$

t.y.

$$L_1\|v_1 - v_4\|_\infty \leq L_p\|v_1 - v_4\|_q \leq L_\infty\|v_1 - v_4\|_1. \tag{2.4}$$

Iš (2.4) seka, kad įvertinant μ_2 tipo viršutinį rėžį simplekso $\mathbf{I}_1$ viršūnėse v_1 ir v_4 optimalu naudoti l_∞ normą ir ją atitinkančią Lipšico konstantą L_1, t.y.

$$\mu_2^{\infty}(v_i) = f(v_i) + L_1\|v_1 - v_4\|_\infty, \tag{2.5}$$

čia $i = 1$ arba $i = 4$. Viršutinis indeksas parodo pagal kurią normą, o ateityje kelias normas yra apskaičiuojamas viršutinis rėžis.

Apskaičiuojant rėžį paskutinėje simplekso $\mathbf{I}_1$ viršūnėje v_3, pastebime, kad atstumas nuo jos iki viršūnės v_1 (arba v_4) nepriklausomai nuo normos yra 1:

$$\begin{aligned}
\|v_3 - v_1\|_1 &= (|1-0| + |0-0|) = 1,\\
\|v_3 - v_1\|_q &= (|1-0|^q + |0-0|^q)^{\frac{1}{q}} = 1, \quad 1 < q < \infty,\\
\|v_3 - v_1\|_\infty &= \max\{|1-0|, |0-0|\} = 1.
\end{aligned}$$

Todėl skaičiuojant rėžį viršūnėje v_3 optimalu naudoti l_1 normą ir ją atitinkančią L_∞ konstantą

$$\mu_2^{1}(v_3) = f(v_3) + L_\infty\|v_3 - v_1\|_1, \tag{2.6}$$

nes $L_\infty \le L_p, 1 \le p < \infty$.

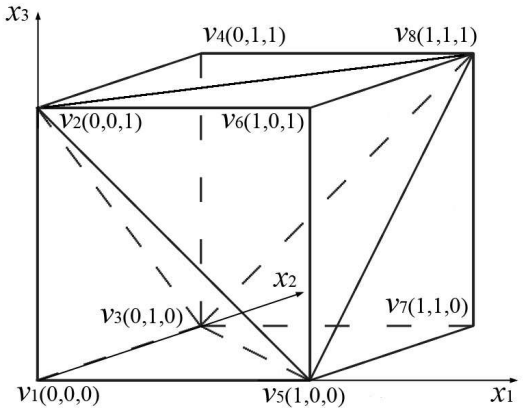
Taigi normų atžvilgiu geriausi rėžiai simplekso $\mathbf{I}_1$ viršūnėse gaunami naudojant (2.6) pasiūlytą l_1 normą ir ją atitinkančią Lipšico konstantas L_∞ arba (2.5) pasiūlytą l_∞ normą ir ją atitinkančią Lipšico konstantą L_1.

Nesunku pastebėti, kad likusiam simpleksui $\mathbf{I}_2$, kaip ir bet kuriam vėliau gautam simpleksui (lygiašoniui stačiajam trikampiui $[v_3, v_4, v_5], [v_3, v_5, v_6], \ldots$ (žr. 1.3 pav.)) dalijant į du hiperplokštuma, einančia per ilgiausios briaunos vidurį ir viršūnes, nepriklausančias ilgiausiajai briaunai, geriausi μ_2 tipo rėžiai gaunami naudojant (2.5) arba (2.6) pasiūlytas normas ir jas atitinkančias Lipšico konstantas. Todėl siekiant geresnių μ_2 tipo rėžių dvimačiuose simpleksiniuose posričiuose $\mathbf{I}$, tikslinga šio tipo rėžius skaičiuoti ne pagal vieną konkrečią normą, o apjungti l_1 ir l_∞ normas ir jas atitinkančias Lipšico konstantas L_∞ ir L_1:

$$\mu_2^{1,\infty}(\mathbf{I}) = \min_{v_i \in V(\mathbf{I})} \left\{ f(v_i) + \min\left\{ L_\infty \max_{x \in V(\mathbf{I})} \|x - v_i\|_1, L_1 \max_{x \in V(\mathbf{I})} \|x - v_i\|_\infty \right\} \right\}. \tag{2.7}$$

Pastebime, kad ir skaičiuojant μ_1 tipo rėžius pagal l_1 ir l_∞ normas ir jas atitinkančias Lipšico konstantas

$$\mu_1^{1,\infty}(\mathbf{I}) = \min_{v_i \in V(\mathbf{I})} f(v_i) + \min\{L_\infty \delta(\mathbf{I}_1)_1, L_1 \delta(\mathbf{I}_1)_\infty\}, \tag{2.8}$$



2.2 pav. *Kubinės leistinosios srities padalijimas į* 5 *tetraedrus*

gaunami tikslesni ir μ_1 tipo rėžiai, negu skaičiuojant pagal vieną konkrečią normą.

Tuomet, kai dvimatė leistinoji sritis yra ne kvadratas, o stačiakampis, po leistinosios srities padengimo simpleksais, gaunami ne lygiašoniai statieji simpleksai, todėl nelygybė (2.4) yra ne visuomet teisinga, o pasiūlytasis rėžis $\mu_2^{1,\infty}$ (2.7) yra ne visada tikslesnis, negu skaičiuojant pagal vieną konkrečią normą. Tačiau atlikti eksperimentiniai tyrimai parodė (žr. 3.1 poskyrį), kad pasiūlytas rėžis $\mu_2^{1,\infty}$ stipriai pagerina optimizavimo rezultatus ir stačiakampio tipo leistinųjų sričių atveju.

2.3.2. Trimatis atvejis

Pratęskime tyrimą, kai leistinoji sritis $\mathbf{D} = [0,1]^3$ yra (vienetinis) kubas. Trimačiu atveju pasiūlėme du skirtingus leistinosios srities padalijimus į simpleksinius posričius (žr. 1.4 pav.) ir (žr. 1.5 pav.). Pradėkime nuo atvejo, kai kubas padalijamas į 5 tetraedrus (žr. 2.2 pav.).

A. Naudojant padalijimą į 5 tetraedrus

Atlikus padalijimą gaunami keturi vienodo, o vidurinis $[v_2, v_3, v_5, v_8]$ - dvigubai didesnio tūrio simpleksai. Laisvai pasirinkę vieną iš 4, tarkime simpleksą, kurio viršūnės $[v_1, v_2, v_3, v_5]$ apskaičiuokime μ_2 tipo rėžius, gaunamus kiekvienoje iš šio simplekso viršūnių, kaip tai darėme dviejų matmenų atveju.

Skaičiuojant viršutinį rėžį viršūnėje v_1, maksimalus atstumas

$$\max_{x\in\{v_2,v_3,v_5\}} \|x - v_1\|$$

sutampa su atstumu iki bet kurios kitos viršūnės (pasirinkime v_2) ir nepriklausomai nuo normos lygus 1:

$$\begin{aligned}
\|v_1 - v_2\|_1 &= (|0-0| + |0-0| + |0-1|) = 1,\\
\|v_1 - v_2\|_q &= (|0-0|^q + |0-0|^q + |0-1|^q)^{\frac{1}{q}} = 1, \quad 1 \le q < \infty,\\
\|v_1 - v_2\|_\infty &= \max\{|0-0|, |0-0|, |0-1|\} = 1.
\end{aligned}$$

Todėl skaičiuojant viršutinį rėžį viršūnėje v_1, optimalu naudoti l_1 normą ir ją atitinkančią Lipšico konstantą L_∞:

$$\mu_2^1(v_1) = f(v_1) + L_\infty \|v_1 - v_2\|_1. \tag{2.9}$$

Skaičiuojant rėžius likusiose simplekso viršūnėse v_2, v_3, v_5 maksimalus atstumas, nepriklausomai nuo viršūnės (pasirinkime viršūnę v_2)

$$\max_{x\in\{v_1,v_3,v_5\}} \|x - v_2\|$$

yra tarp viršūnių v_2 ir v_3 (arba v_2 ir v_5), kuris priklausomai nuo normos lygus:

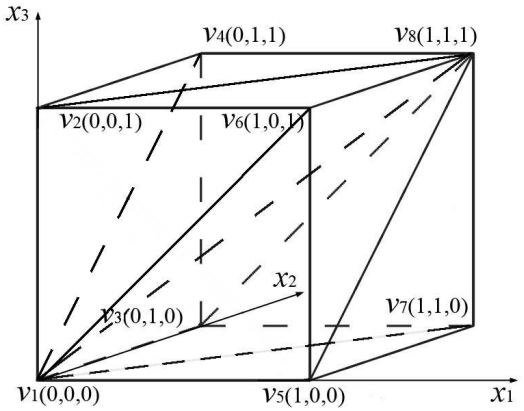
$$\begin{aligned}
\|v_2 - v_3\|_1 &= (|0-0| + |0-1| + |1-0|) = 2,\\
\|v_2 - v_3\|_q &= (|0-0|^q + |0-1|^q + |1-0|^q)^{\frac{1}{q}} = 2^{\frac{1}{q}}, \quad 1 \le q < \infty,\\
\|v_2 - v_3\|_\infty &= \max\{|0-1|, |0-0|, |1-0|\} = 1.
\end{aligned}$$

Todėl galimos tokios, Lipšico konstantos ir ją atitinkančios normos, sandaugos:

$$\begin{aligned}
L_\infty\|v_2 - v_3\|_1 &= \max\left\{|f'_{x_1}(z_1)|, |f'_{x_2}(z_2)|, |f'_{x_3}(z_3)|\right\} \cdot 2,\\
L_p\|v_2 - v_3\|_q &= \left(|f'_{x_1}(z_1)|^p + |f'_{x_2}(z_2)|^p + |f'_{x_3}(z_3)|^p\right)^{\frac{1}{p}} \cdot 2^{\frac{1}{q}}, 1 < p, q < \infty,\\
L_1\|v_2 - v_3\|_\infty &= \left(|f'_{x_1}(z_1)| + |f'_{x_2}(z_2)| + |f'_{x_3}(z_3)|\right) \cdot 1.
\end{aligned}$$

Trijų matmenų atveju (2.4) nelygybių analogas

$$\begin{aligned}
\left(|f'_{x_1}(z_1)| + |f'_{x_2}(z_2)| + |f'_{x_3}(z_3)|\right) &\le \left(|f'_{x_1}(z_1)|^p + |f'_{x_2}(z_2)|^p + |f'_{x_3}(z_3)|^p\right)^{\frac{1}{p}} \cdot 2^{\frac{1}{q}}\\
&\le \max\left\{|f'_{x_1}(z_1)|, |f'_{x_2}(z_2)|, |f'_{x_3}(z_3)|\right\} \cdot 2
\end{aligned} \tag{2.10}$$



2.3 pav. *Kubinės leistinosios srities padalijimas į* $3! = 6$ *simpleksus*

teisingas tik tada, kai viena iš dalinių išvestinių reikšmių $|f'_{x_i}(z_i)|, i = 1, \ldots, 3$ lygi arba artima 0. Todėl trimačiu atveju rėžis $\mu_2^{1,\infty}$ yra ne visada tikslesnis, negu skaičiuojant μ_2 pagal vieną konkrečią normą.

Beliko ištirti vidurinį simpleksą su viršūnėmis $[v_2, v_3, v_5, v_8]$. Pastebime, kad atstumai iš bet kurios šio simplekso viršūnės iki kitų yra vienodi ir sutampa su ką tik nagrinėto simplekso $[v_1, v_2, v_3, v_5]$ atstumu tarp viršūnių v_2 ir v_3. Iš to seka, kad ir šiam simpleksui pasiūlytas rėžis $\mu_2^{1,\infty}$ yra ne visada tikslesnis.

B. Naudojant apibendrintą padalijimą

Pratęskime tyrimą simpleksams, gautiems po kombinatorinio padalijimo (žr. 2.3 pav.). Pastebime, kad naudojant šį padalijimą, viršūnės v_1 ir v_8 priklauso visiems gautiems simpleksams, o visi gautieji simpleksai yra vienodo tūrio. Laisvai pasirinkę vieną iš simpleksų, pvz. su viršūnėmis $[v_1, v_2, v_4, v_8]$ apskaičiuokime μ_2 tipo rėžius kiekvienoje viršūnėje.

Maksimalus atstumas iš viršūnės v_1 yra iki viršūnės v_8, kuris priklausomai nuo normos yra:

$$\begin{aligned}
\|v_1 - v_8\|_1 &= (|0-1| + |0-1| + |0-1|) = 3, \\
\|v_1 - v_8\|_q &= (|0-1|^q + |0-1|^q + |0-1|^q)^{\frac{1}{q}} = 3^{\frac{1}{q}}, \quad 1 \le q < \infty, \\
\|v_1 - v_8\|_\infty &= \max\{|0-1|, |0-1|, |0-1|\} = 1.
\end{aligned}$$

Todėl galimos tokios, Lipšico konstantos ir ją atitinkančios normos, sandaugos:

$$\begin{aligned} L_\infty\|v_1-v_8\|_1 &= \max\left\{|f'_{x_1}(z_1)|,|f'_{x_2}(z_2)|,|f'_{x_3}(z_3)|\right\}\cdot 3,\\ L_p\|v_1-v_8\|_q &= \left(|f'_{x_1}(z_1)|^p+|f'_{x_2}(z_2)|^p+|f'_{x_3}(z_3)|^p\right)^{\frac{1}{p}}\cdot 3^{\frac{1}{q}},\quad 1<p,q<\infty,\\ L_1\|v_1-v_8\|_\infty &= \left(|f'_{x_1}(z_1)|+|f'_{x_2}(z_2)|+|f'_{x_3}(z_3)|\right)\cdot 1. \end{aligned}$$

Šiuo atveju nelygybės

$$\begin{aligned} \left(|f'_{x_1}(z_1)|+|f'_{x_2}(z_2)|+|f'_{x_3}(z_3)|\right) &\le \left(|f'_{x_1}(z_1)|^p+|f'_{x_2}(z_2)|^p+|f'_{x_3}(z_3)|^p\right)^{\frac{1}{p}}\cdot 3^{\frac{1}{q}}\\ &\le \max\left\{|f'_{x_1}(z_1)|,|f'_{x_2}(z_2)|,|f'_{x_3}(z_3)|\right\}\cdot 3, \end{aligned}$$

t.y.

$$L_1\|v_1-v_8\|_\infty \le L_p\|v_1-v_8\|_q \le L_\infty\|v_1-v_8\|_1 \tag{2.11}$$

yra visada teisingos, todėl skaičiuojant μ_2 tipo rėžį viršūnėje v_1 (arba v_8) optimalu naudoti l_∞ normą ir ją atitinkančią Lipšico konstantą:

$$f(v_1)+L_1\|v_1-v_8\|_\infty. \tag{2.12}$$

Taigi naudojant kombinatorinį leistinosios srities padalijimą, visuose gaunamuose simpleksuose yra dvi viršūnės, kuriose iš anksto žinoma, kad μ_2 tipo viršutinio rėžio skaičiavimui optimalu naudoti begalinę normą ir ją atitinkančią pirmąją Lipšico konstantą.

Apskaičiuokime viršutinius rėžius likusiose viršūnėse v_2 ir v_4. Maksimalus atstumas iš viršūnės v_2 yra iki v_8, o iš viršūnės v_4 iki v_1. Abiems atvejais maksimalūs atstumai yra lygūs ir priklausomai nuo normos gauname:

$$\begin{aligned} \|v_2-v_8\|_1 &= (|0-1|+|0-1|+|1-1|)=2,\\ \|v_2-v_8\|_q &= (|0-1|^q+|0-1|^q+|1-1|^q)^{\frac{1}{q}}=2^{\frac{1}{q}},\quad 1\le q<\infty,\\ \|v_2-v_8\|_\infty &= \max\{|0-1|,|0-1|,|1-1|\}=1. \end{aligned}$$

Gauta situacija yra analogiška išnagrinėtai (2.10), todėl ir šio padalijimo atveju rėžis $\mu_2^{1,\infty}$ trimačiuose simpleksuose yra ne visada tikslesnis, negu skaičiuojant μ_2 pagal vieną konkrečią normą.

Dalijant pradinius simpleksus toliau, situacija išlieka panaši, kaip ir su pirminiais simpleksais: ne visose viršūnėse pasiūlytas rėžis $\mu_2^{1,\infty}$ yra tikslesnis nei μ_2 tipo rėžis skaičiuojant pagal kažkurią konkrečią normą, todėl tikrasis pasiūlyto rėžio gerumas didesnės dimensijos atveju parodomas atlikus eksperimentinius tyrimus (žr. 3.1 poskyrį).

2.4. Pijavskij tipo rėžis pirmosios normos atveju

Pijavskij (φ) tipo (1.8) rėžių radimo skaičiavimo apimtys didėja eksponentiškai didėjant erdvės $\mathbf{D}$ dimensijai. Todėl šakų ir rėžių pagrindu realizuoti algoritmai viršutinio rėžio skaičiavimui dažniausiai naudoja lengviau apskaičiuojamus μ tipo rėžius. Tačiau „brangių“ tikslo funkcijų atveju Pijavskij rėžių naudojimas gali būti tikslingas. Todėl šiame poskyryje pasiūlomas metodas, kuriuo Pijavskij tipo Lipšico rėžiai simplekse gaunami sprendžiant tiesinių lygčių sistemas, vietoj kvadratinių lygčių sistemų Pijavskij algoritmo atveju [100] arba kvadratinių ir tiesinių lygčių sistemų Mladineo algoritmo [82] atveju. Pirmiausia įrodykime pagalbinį teiginį.

2.3 teiginys. *Tegu* $F_{v_1}(x) = f(v_1) + L_\infty \|x - v_1\|_1$ *ir* $F_{v_2}(x) = f(v_2) + L_\infty \|x - v_2\|_1$ *yra* n*-matės piramidės ir* $f(v_1) \geq f(v_2)$*. Tuomet šių* n*-mačių piramidžių sankirta yra* $n-1$ *matmens daugdaroje, apibrėžtoje lygybe*

$$\sum_{i=1}^{n} d(v_{1i}, v_{2i}) - \frac{f(v_2) - f(v_1)}{L_\infty} = 0, \tag{2.13}$$

čia

$$d\left(v_{1i}, v_{2i}\right) = \begin{cases} |2x_i - v_{1i} - v_{2i}| & \text{kai } v_{1i} \leq x_i \leq v_{2i} \\ 0 & \text{kai } v_{1i} = v_{2i} \\ |v_{1i} - v_{2i}| & \text{kai } x_i \notin [v_{1i}, v_{2i}] \end{cases}$$

ir sankirtos taškai x *yra arčiau viršūnės* v_1 *nei* v_2*, t.y.*

$$||x - v_1||_1 \leq ||x - v_2||_1. \tag{2.14}$$

Įrodymas. *Iš lygybės* $F_{v_1}(x) = F_{v_2}(x)$ *gauname*

$$\sum_{i=1}^{n} \left(|x_i - v_{1i}| - |x_i - v_{2i}|\right) = \frac{f\left(v_2\right) - f\left(v_1\right)}{L_\infty}. \tag{2.15}$$

Kiekvienam iš skirtumų $|x_i - v_{1_i}| - |x_i - v_{2_i}|$, $i = 1, \ldots, n$ *yra teisinga viena iš šių lygybių:*

$$\begin{cases} 2x_i - v_{1i} - v_{2i} & \text{kai } v_{1i} \leq x_i \leq v_{2i},\ v_{1i} < v_{2i} \\ -2x_i + v_{1i} + v_{2i} & \text{kai } v_{1i} \leq x_i \leq v_{2i},\ v_{1i} > v_{2i} \\ 0 & \text{kai } v_{1i} = v_{2i} \\ v_{1i} - v_{2i} & \text{kai } x_i > v_{1i}, v_{2i} \\ v_{2i} - v_{1i} & \text{kai } x_i > v_{1i}, v_{2i} \end{cases}$$

Todėl iš (2.15) gauname (2.13).
Kadangi $f(v_1) > f(v_2)$, *todėl iš lygybės* $F_{v_1}(x) = F_{v_2}(x)$ *gauname*

$$||x - v_1||_1 - ||x - v_2||_1 = \frac{f(v_2) - f(v_1)}{L_\infty} \leq 0,$$

vadinasi (2.14) yra teisinga

$$||x - v_1||_1 \leq ||x - v_2||_1.$$

□

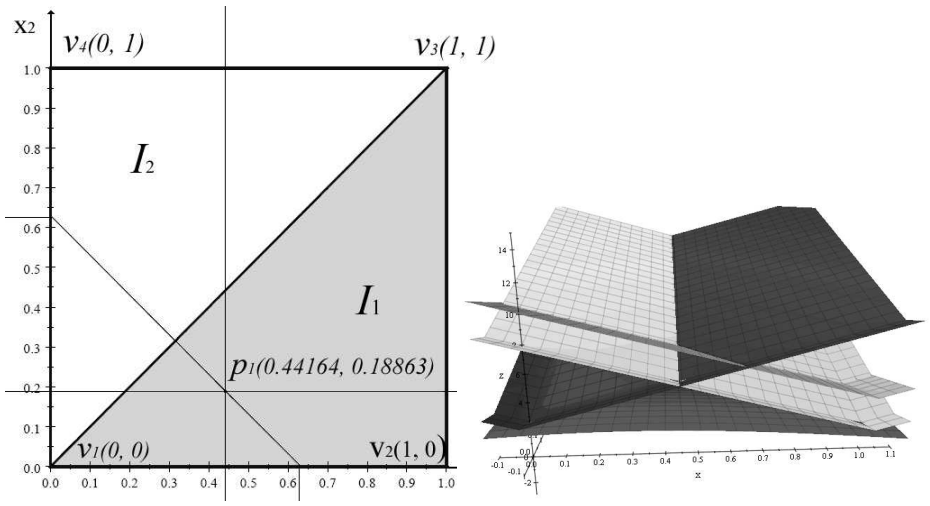
2.4 teiginys. *Pijavski tipo viršutinio rėžio su pirmąja norma* φ^1 *radimui užtenka išspręsti* n *tiesinių lygybių sistemą.*

Įrodymas. *Leistinosios srities viršūnes* v_i *sunumeruokime taip, kad* $f(v_1) \geq f(v_2) \geq \ldots \geq f(v_{n+1})$. *Piramidžių* $F_{v_1}(x) = f(v_1) + L_\infty \|x - v_1\|_1$ *ir* $F_{v_i}(x) = f(v_i) + L_\infty \|x - v_i\|_1$, $i = 2, \ldots, n+1$ *sankirta yra* $n-1$*-daugdara, apibrėžta (2.13). Atsižvelgę į (2.14) galime nagrinėti tik dalį daugdaros, apibrėžtos tiesinėmis lygtimis ir apribojimais. Tokiu būdu sudarome sistemą iš* n *tiesinių lygčių, nusakančių sankirtą* $F_{v_1}(x) = F_{v_i}(x)$. *Jeigu šios sistemos sprendinys tenkina apribojimus (žr. 1 pavyzdį), tuomet jis yra ieškomasis viršutinio rėžio taškas, o ieškomasis Pijavskij* (φ) *tipo rėžis su pirmąja norma* φ^1 *yra lygus viršutinio rėžio funkcijos reikšmei šiame taške. Tačiau jeigu sistemos sprendinys netenkina apribojimų, tuomet viršutinio rėžio maksimumas yra lygus minimaliai funkcijos reikšmei sankirtos taškuose, kaip parodyta 2 pavyzdyje.* □

C. Pavyzdžiai

2.1 pavyzdys

Tarkime, kad tikslo funkcija $f(x_1, x_2) = \sin(2x_1 + 1) + 2\sin(3x_2 + 2)$, *o leistinoji sritis* $[0,1] \times [0,1]$ *padengta simpleksais* $\mathbf{I}_1(v_1, v_2, v_3)$ *ir* $\mathbf{I}_2(v_1, v_3, v_4)$ *(žr. 2.4 pav. a)). Nagrinėkime simpleksą* $\mathbf{I}_1(v_1, v_2, v_3)$.
Kadangi $f(v_1) = f(0,0) = 2,6601 > f(v_2) = f(1,0) = 1,9597 > f(v_3) = f(1,1) = -1,7767$, *sankirtos taškas yra arčiau viršūnės* v_1 *ir yra pakankama surasti piramidžių sankirtą* $F_{v_1} = F_{v_2}$ *ir* $F_{v_1} = F_{v_3}$.



***2.4* pav.** *Pijavskij tipo rėžių su pirmąja norma 1 pavyzdys: a) sankirtos tiesių projekcijos; b) viršutinio rėžio funkcijų vizualizacija*

Piramidžių sankirta $F_{v_1} = F_{v_2}$ *yra:*

$$\begin{aligned} |x_1 - 0| - |x_1 - 1| + |x_2 - 0| - |x_2 - 0| &= \frac{f(1,0) - f(0,0)}{6}, \\ \textit{kadangi } 0 &\leq x_1 \leq 1, \\ x_1 + x_1 - 1 &= \frac{f(1,0) - f(0,0)}{6}, \\ x_1 &= 0,44164. \end{aligned} \tag{2.16}$$

Analogiškai piramidžių sankirta $F_{v_1} = F_{v_3}$ *yra:*

$$|x_1 - 0| - |x_1 - 1| + |x_2 - 0| - |x_2 - 1| = \frac{f(1,1) - f(0,0)}{6},$$

$$\begin{cases} x_1 + x_2 = \frac{f(1,1) - f(0,0)}{12} = 0,63027, & \textit{kai } 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1, \\ x_2 = 0,63027, & \textit{kai } x_1 \leq 0, \\ x_1 = 0,63027, & \textit{kai } x_2 \leq 0. \end{cases} \tag{2.17}$$

Iš (2.14), (2.16), (2.17) $\Rightarrow$

$$\begin{cases} x_1 = 0,44164 \\ x_1 + x_2 = 0,63027 \\ 0 \le x_1 \le 0,5 \\ 0 \le x_2 \le 1,0 \end{cases} \Rightarrow \text{sankirtos taškas } p_1 = (0,44164; 0,18863)\,.$$

Todėl viršutinis rėžis $\varphi^1(\mathbf{I}_1)$ *(žr. 2.4 pav.) yra*

$$\begin{aligned} \varphi^1(\mathbf{I}_1) &= F_{v_i}(p_1) = F_{v_1}(p_1) = f(v_1) + L_\infty \|p_1 - v_1\|_1 \\ &= f(0,0) + 6\,(|0,44164 - 0| + |0,18863 - 0|) = 6,441. \end{aligned} \quad (2.18)$$

Parodykime, kad rėžis φ^1 *iš tiesų yra geresnis nei lengviau apskaičiuojamas* μ_2 *tipo rėžis su pirmąja norma* μ_2^1*:*

$$\begin{aligned} \mu_2^1(v_1) &= f(0,0) + 6 \max_{x\in\mathbf{I}} \|x - v_1\|_1 = f(0,0) + 6\cdot 2 = 14,66, \\ \mu_2^1(v_2) &= f(1,0) + 6 \max_{x\in\mathbf{I}} \|x - v_2\|_1 = f(1,0) + 6\cdot 1 = 7,9597, \\ \mu_2^1(v_3) &= f(1,1) + 6 \max_{x\in\mathbf{I}} \|x - v_3\|_1 = f(1,1) + 6\cdot 2 = 10,223, \\ \mu_2^1(\mathbf{I}_1) &= \min\left\{\mu_2^1(v_1), \mu_2^1(v_2), \mu_2^1(v_3)\right\} = 7,9597. \end{aligned} \quad (2.19)$$

Iš (2.18) ir (2.19) $\Rightarrow \varphi^1(\mathbf{I}_1) < \mu_2^1(\mathbf{I}_1)$.

2.2 pavyzdys

Padaliję simpleksus $\mathbf{I}_1$ *ir* $\mathbf{I}_2$ *gauname keturis naujus* $\mathbf{I}_3, \ldots, \mathbf{I}_6$ *(žr. 2.5 pav.). Pasirinkime simpleksą* $\mathbf{I}_3 = [v_1, v_2, v_5]$*. Kadangi*

$$f(v_1) = 2,6601 > f(v_2) = 1,9597 > f(v_5) = 0.20773,$$

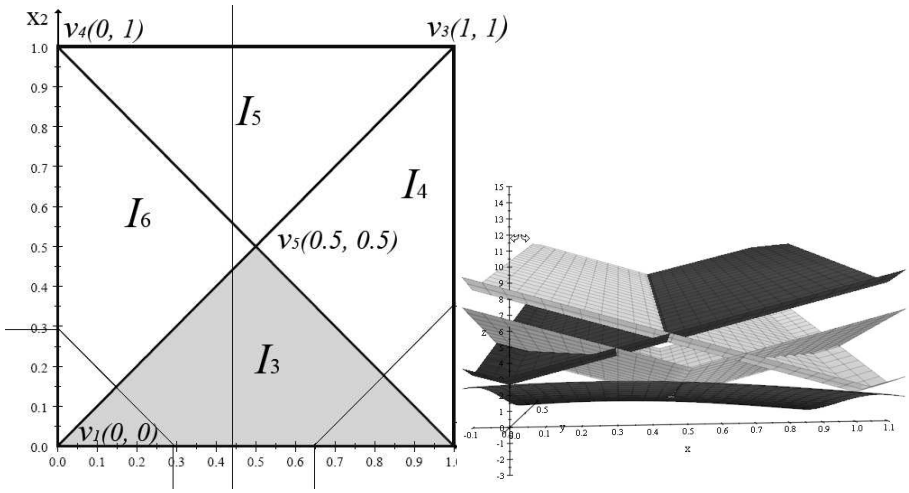
sankirtos taškas yra arčiausiai viršūnės v_1*, todėl pakanka surasti piramidžių* $F_{v_1} = F_{v_2}$ *ir* $F_{v_1} = F_{v_5}$ *sankirtas.*
Sankirta piramidžių $F_{v_1} = F_{v_2}$ *surasta 1 pavyzdyje*

$$x_1 = 0,44164. \quad (2.20)$$

Sankirta piramidžių $F_{v_1} = F_{v_5}$*:*

$$|x_1 - 0| - |x_1 - 0.5| + |x_2 - 0| - |x_2 - 0.5| = \frac{f(0.5, 0.5) - f(0,0)}{6},$$

$$\begin{cases} x_1 + x_2 = \frac{f(0.5,0.5)-f(0,0)}{12} + \frac{1}{2} = 0,29564, & kai\ 0 \le x_1 \le 0.5, 0 \le x_2 \le 0.5, \\ x_2 = 0,29564, & kai\ x_1 \le 0, \\ x_1 = 0,29564, & kai\ x_2 \le 0. \end{cases} \tag{2.21}$$



***2.5* pav.** *Pijavskij tipo rėžių su pirmąja norma 2 pavyzdys: a) Sankirtos tiesių projekcijos; b) Viršutinio rėžio funkcijų vizualizacija*

Iš (2.14),(2.20), (2.21) $\Rightarrow$

$$\begin{cases} x_1 = 0.44164 \\ x_1 + x_2 = 0.29564 \\ 0 \le x_1 \le 0.5 \\ 0 \le x_2 \le 0.5 \end{cases} \Rightarrow \varnothing \ (\check{z}r.\ 2.5\ pav.)$$

Todėl viršutinis rėžis įgyjamas viename iš piramidžių atkarpos taškų (žr. 2.5 pav.), t.y.

$$\varphi^1(\mathbf{I}_3) = F_{v_1}(p_1),$$

čia p_1 *yra bet kuris atkarpos* $x_1 + x_2 = 0,29564$ *taškas* $0 \le x_1 \le 0.5, 0 \le x_2 \le 0.5$. *Tarkime* $p_1 = (0,29564; 0)$, *tada*

$$\begin{aligned} \varphi^1(\mathbf{I}_3) &= F_{v_1}(p_1) = f(v_1) + L_\infty \|p_1 - v_1\|_1 \\ &= f(0,0) + 6(|0,29564 - 0| + |0 - 0|) = 4.4339. \end{aligned} \tag{2.22}$$

Analogiškai parodykime, kad φ tipo rėžis su pirmąja norma φ^1 iš tiesų yra geresnis nei rėžis μ_2^1:

$$\begin{aligned}
\mu_2^1(v_1) &= f(0,0) + 6\max_{x\in\mathbf{I}} \|x - v_1\|_1 = f(0,0) + 6\cdot 1 = 8,66,\\
\mu_2^1(v_2) &= f(1,0) + 6\max_{x\in\mathbf{I}} \|x - v_2\|_1 = f(1,0) + 6\cdot 1 = 7,9597,\\
\mu_2^1(v_5) &= f(0.5,0.5) + 6\max_{x\in\mathbf{I}} \|x - v_5\|_1 = f(0.5,0.5) + 6\cdot 1 = 6,20773,\\
\mu_2^1(\mathbf{I}_3) &= \min\left\{\mu_2^1(v_1), \mu_2^1(v_2), \mu_2^1(v_5)\right\} = 6,20773. \qquad (2.23)
\end{aligned}$$

Iš (2.22) ir (2.23) $\Rightarrow \varphi^1(\mathbf{I}_3) < \mu_2^1(\mathbf{I}_3)$.

2.5. Lipšico rėžis, pagrįstas daugiamatės apibrėžtinės sferos spinduliu

Šiame poskyryje pasiūlytas naujas rėžis, kuris gali būti naudojamas kaip alternatyva μ ir φ tipo rėžiams.

2.5 teiginys. *Tegu $\mathbf{I} \subset \mathbb{R}^n$ yra n-matis simpleksas, $R(\mathbf{I})$ – apie simpleksą $\mathbf{I}$ apibrėžtinės n-matės sferos spindulys, $V(\mathbf{I})$ simplekso viršūnių aibė. Tuomet*

$$\psi^2(\mathbf{I}) = \max_{v\in V(\mathbf{I})} f(v) + LR(\mathbf{I}), \qquad (2.24)$$

yra Lipšico viršutinis rėžis.

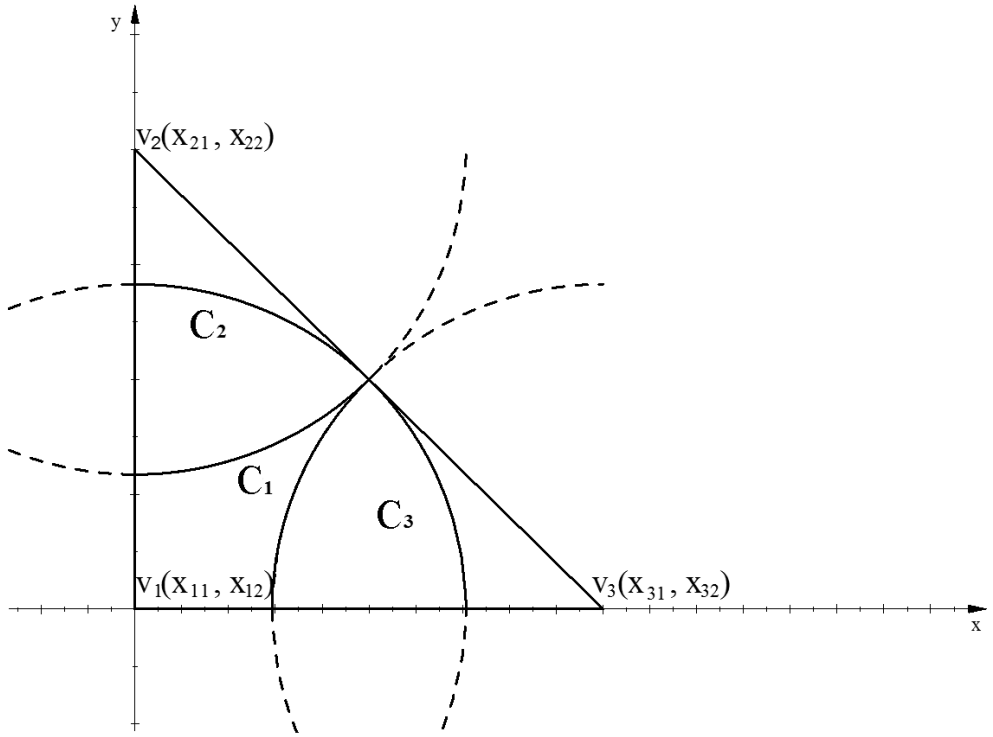
Įrodymas. *N-matis simpleksas $\mathbf{I}$ gali būti padengtas n-mačiais rutuliais $\mathbf{C}_i, i = 1,\ldots,n+1$, kurių spindulys lygus $R(\mathbf{I})$, o centrai sutampa su simplekso viršūnėmis. Dvimatis tokio padengimo pavyzdys pavaizduotas 2.6 paveiksle. Tuomet $\forall i = 1,\ldots,n+1$ teisinga nelygybė*

$$f(x) \le f(v_i)_{v_i\in V(\mathbf{I})\cap\mathbf{C}_i} + LR(\mathbf{I}), \quad \forall x \in \mathbf{C}_i \cap \mathbf{I}. \qquad (2.25)$$

Todėl iš (2.25) $\Rightarrow$

$$f(x) \le \max_{v\in V(\mathbf{I})} f(v) + LR(\mathbf{I}) = \psi^2(\mathbf{I}), \quad \forall x \in \mathbf{I}. \qquad (2.26)$$

□



2.6 pav. *Dvimačio simplekso padengimas trimis 2-mačiais rutuliais (skrituliais)* $\mathbf{C}_i, i = 1, \ldots, n+1$ *su spinduliu* $R(\mathbf{I})$

Pasiūlyto ψ tipo rėžio skaičiavimo sunkumas daugiausiai priklauso nuo apibrėžtinės daugiamatės sferos spindulio R radimo. Išveskime apie n–matį simpleksą apibrėžtos daugiamatės sferos spindulio R formulę. Pirmiausia išvesime $n = 2$ atvejui, o tuomet apibendrinsime bet kokiam.

Pasinaudokime iš analizinės geometrijos žinomu faktu [91], kad dvimatės sferos (apskritimo), einančios per tris taškus (mūsų atveju per visas simplekso (trikampio) viršūnes) $v_1(x_{11}, x_{12})$, $v_2(x_{21}, x_{22})$, $v_3(x_{31}, x_{32})$, lygtis išreikšta determinantine forma yra

$$\begin{vmatrix} x_1^2 + x_2^2 & x_1 & x_2 & 1 \\ x_{11}^2 + x_{12}^2 & x_{11} & x_{12} & 1 \\ x_{21}^2 + x_{22}^2 & x_{21} & x_{22} & 1 \\ x_{31}^2 + x_{32}^2 & x_{31} & x_{32} & 1 \end{vmatrix} = 0. \tag{2.27}$$

Išskleidę pagal pirmąją eilutę, gauname

$$a\left(x_1^2+x_2^2\right)+D_{x_1}x_1+D_{x_2}x_2+c=0, \tag{2.28}$$

čia

$$\begin{aligned}
a &= \begin{vmatrix} x_{11} & x_{12} & 1 \\ x_{21} & x_{22} & 1 \\ x_{31} & x_{32} & 1 \end{vmatrix}, \\
D_{x_1} &= -\begin{vmatrix} x_{11}^2+x_{12}^2 & x_{12} & 1 \\ x_{21}^2+x_{22}^2 & x_{22} & 1 \\ x_{31}^2+x_{32}^2 & x_{32} & 1 \end{vmatrix}, \\
D_{x_2} &= \begin{vmatrix} x_{11}^2+x_{12}^2 & x_{11} & 1 \\ x_{21}^2+x_{22}^2 & x_{21} & 1 \\ x_{31}^2+x_{32}^2 & x_{31} & 1 \end{vmatrix}, \\
c &= -\begin{vmatrix} x_{11}^2+x_{12}^2 & x_{11} & x_{12} \\ x_{21}^2+x_{22}^2 & x_{21} & x_{22} \\ x_{31}^2+x_{32}^2 & x_{31} & x_{32} \end{vmatrix}.
\end{aligned}$$

Perrašę lygybę (2.28) į formą

$$a\left(x_1+\frac{D_{x_1}}{2a}\right)^2+a\left(x_2+\frac{D_{x_2}}{2a}\right)^2-\frac{D_{x_1}^2}{4a}-\frac{D_{x_2}^2}{4a}+c=0,$$

gauname

$$(x_1-c_1)^2+(x_2-c_2)^2=R^2$$

formos dvimatės sferos (apskritimo) lygtį

$$\left(x_1+\frac{D_{x_1}}{2a}\right)^2+\left(x_2+\frac{D_{x_2}}{2a}\right)^2=\frac{D_{x_1}^2+D_{x_2}^2-4ac}{4a^2}$$

su centru

$$(c_1,c_2)=\left(-\frac{D_{x_1}}{2a},-\frac{D_{x_2}}{2a}\right)$$

ir spinduliu

$$R=\sqrt{\frac{D_{x_1}^2+D_{x_2}^2-4ac}{4a^2}}.$$

Analogiškai (2.27) n-matės apibrėžtinės sferos, einančios per n-mačio simplekso viršūnes $v_1(x_{11},\ldots,x_{1n}),\ldots,v_{n+1}(x_{(n+1)1},\ldots,x_{(n+1)n})$, lygtis išreikšta

determinantine forma yra

$$\begin{vmatrix} \sum\limits_{i=1}^{n} x_i^2 & x_1 & x_2 & \dots & x_n & 1 \\ \sum\limits_{i=1}^{n} x_{1i}^2 & x_{11} & x_{12} & \dots & x_{1n} & 1 \\ \sum\limits_{i=1}^{n} x_{2i}^2 & x_{21} & x_{22} & \dots & x_{2n} & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum\limits_{i=1}^{n} x_{(n+1)i}^2 & x_{(n+1)1} & x_{(n+1)2} & \dots & x_{(n+1)n} & 1 \end{vmatrix} = 0 \qquad (2.29)$$

Išskleidę pagal pirmąją eilutę, gauname

$$a\left(\sum_{i=1}^{n} x_i^2\right) + \left(\sum_{i=1}^{n} D_{x_i} x_i\right) + c = 0, \qquad (2.30)$$

čia

$$a = \begin{vmatrix} x_{11} & x_{12} & \dots & x_{1n} & 1 \\ x_{21} & x_{22} & \dots & x_{2n} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{(n+1)1} & x_{(n+1)2} & \dots & x_{(n+1)n} & 1 \end{vmatrix},$$

$$D_{x_1} = - \begin{vmatrix} \sum\limits_{i=1}^{n} x_{1i}^2 & x_{12} & \dots & x_{1n} & 1 \\ \sum\limits_{i=1}^{n} x_{2i}^2 & x_{22} & \dots & x_{2n} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum\limits_{i=1}^{n} x_{(n+1)i}^2 & x_{(n+1)2} & \dots & x_{(n+1)n} & 1 \end{vmatrix},$$

$$\vdots$$

$$D_{x_n} = (-1)^n \begin{vmatrix} \sum\limits_{i=1}^{n} x_{1i}^2 & x_{11} & \dots & x_{1n-1} & 1 \\ \sum\limits_{i=1}^{n} x_{1i}^2 & x_{21} & \dots & x_{2n-1} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum\limits_{i=1}^{n} x_{(n+1)i}^2 & x_{(n+1)1} & \dots & x_{(n+1)n-1} & 1 \end{vmatrix},$$

$$c = (-1)^{n+1} \begin{vmatrix} \sum\limits_{i=1}^{n} x_{1i}^2 & x_{11} & \dots & x_{1n} \\ \sum\limits_{i=1}^{n} x_{1i}^2 & x_{21} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sum\limits_{i=1}^{n} x_{(n+1)i}^2 & x_{(n+1)1} & \dots & x_{(n+1)n} \end{vmatrix}.$$

Perrašę lygybę (2.30) į formą

$$a\left(x_1 + \frac{D_{x_1}}{2a}\right)^2 + \ldots + a\left(x_n + \frac{D_{x_n}}{2a}\right)^2 - \frac{D_{x_1}^2}{4a} - \ldots - \frac{D_{x_n}^2}{4a} + c = 0$$

gauname n-matės sferos lygtį

$$\left(x_1 + \frac{D_{x_1}}{2a}\right)^2 + \ldots + \left(x_2 + \frac{D_{x_2}}{2a}\right)^2 = \frac{D_{x_1}^2 + D_{x_2}^2 - 4ac}{4a^2}$$

su centru

$$(c_1, c_2, \ldots, c_n) = \left(-\frac{D_{x_1}}{2a}, \ldots, -\frac{D_{x_n}}{2a}\right)$$

ir spinduliu

$$R = \sqrt{\frac{D_{x_1}^2 + D_{x_2}^2 + \ldots + D_{x_n}^2 - 4ac}{4a^2}}.$$

2.6. Nuoseklusis simpleksinis šakų ir rėžių algoritmas su Lipšico rėžiais

Šakų ir rėžių algoritmuose padengimo, išrinkimo, padalijimo ir rėžių skaičiavimo taisyklės priklauso nuo konkretaus algoritmo. Bendras n-matis simpleksinis šakų ir rėžių Lipšico optimizavimo algoritmas pasiūlytas [130]. Disertacijoje siūloma algoritmo modifikacija, naudojanti μ_2, φ, ψ tipo Lipšico su įvairiomis normomis bei agreguotus rėžius, sudarytus iš skirtingo tipo rėžių su įvairiomis normomis.

Pagrindinis simpleksinių posričių trūkumas – poreikis padengti leistiną-ją sritį simpleksais, todėl inicializavimo etape leistinoji sritis padalijama į $n!$

vienodo tūrio simpleksų, panaudojant apibendrintą kombinatorinį hiperstačiakampio viršūnių trianguliacijos algoritmą (žr. 1.2 skyrių).

Elemento išrinkimo/įterpimo strategijos įtakoja šakų ir rėžių algoritmų efektyvumą, todėl siūlomame algoritme gali būti naudojamos įvairios (geryn, gilyn, platyn, statistinė) strategijos.

Posričiai dalijami atsižvelgiant į eksperimentų rezultatus [130]. Juose atliktas tyrimas parodė, kad naudingiausias yra iškraipymo išvengiantis simpleksų dalijimas į du hiperplokštuma, einančia per ilgiausios briaunos vidurį ir viršūnes, nepriklausančias ilgiausiai briaunai (žr. 1.3 pav.).

Tikslo funkcijos maksimumo apatinis rėžis (LB) yra maksimali funkcijos reikšmė simplekso viršūnėse. Viršutinis rėžis apskaičiuojamas naudojant įvairius pasiūlytus μ, φ, ψ tipo bei agreguotus rėžius su įvairiomis normomis. Pilnas nuoseklusis simpleksinis šakų ir rėžių algoritmas su tokiais rėžiais pavaizduotas 3 algoritme.

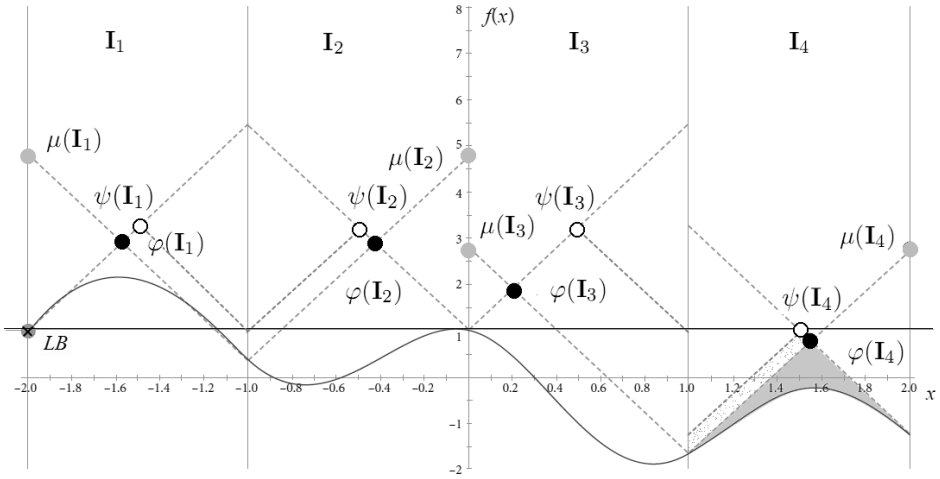
3 algoritmas Simpleksinis šakų ir rėžių algoritmas su Lipšico rėžiais

1: Leistinoji sritis $\mathbf{D}$ padengiama n-mačiais simpleksais $\mathbb{I} \leftarrow \{\mathbf{I}_j | \mathbf{D} = \cup \mathbf{I}_j, j = 1, \ldots, n!\}$ naudojant viršūnių trianguliaciją
2: $\mathbf{S} \leftarrow \emptyset$, $LB(\mathbf{D}) \leftarrow -\infty$
3: **while** ($\mathbb{I} \neq \emptyset$) **do**
4: Pasirenkamas $\mathbf{I}_j \in \mathbb{I}$ naudojant vieną iš (geryn, gilyn, platyn, statistinė) išrinkimo strategijų, $\mathbb{I} \leftarrow \mathbb{I} \backslash \{\mathbf{I}_j\}$
5: $LB(\mathbf{D}) \leftarrow \max\{LB(\mathbf{D}), \max\limits_{v \in V(\mathbf{I}_j)} f(v)\}$
6: $\mathbf{S} \leftarrow \arg\max\{f(\mathbf{S}), \max\limits_{v \in V(\mathbf{I}_j)} f(v)\}$
7: Apskaičiuojamas $UB(\mathbf{I}_j)$ pagal μ, φ, ψ tipo sudarytą rėžį su įvairiomis normomis.
8: **if** $(UB(\mathbf{I}_j) > LB(\mathbf{D}) + \varepsilon)$ **then**
9: Padalijamas $\mathbf{I}_j$ į $\mathbf{I}_{j_1}$, $\mathbf{I}_{j_2}$ hiperplokštuma, einančia per ilgiausios briaunos vidurį ir viršūnes, nepriklausančias ilgiausiajai briaunai
10: $\mathbb{I} \leftarrow \mathbb{I} \cup \{\mathbf{I}_{j_1}, \mathbf{I}_{j_2}\}$
11: **end if**
12: **end while**

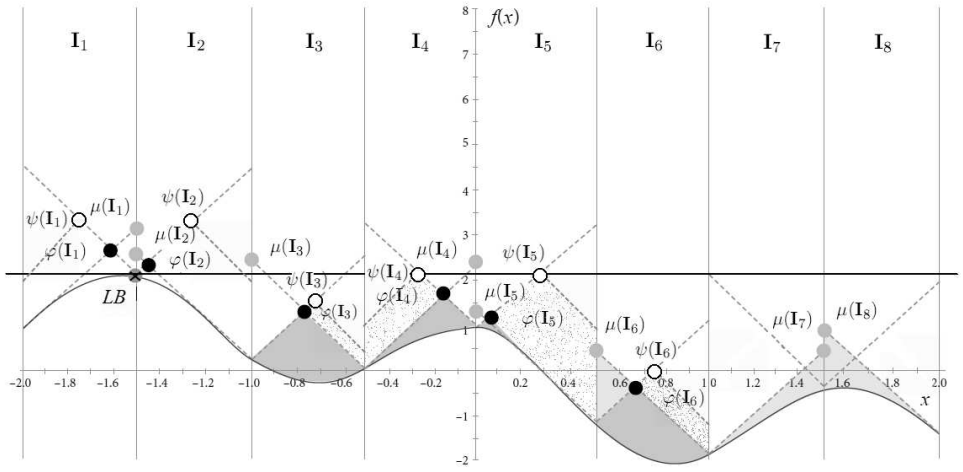
Pavaizduokime šakų ir rėžių procesą paprasčiausiu, vienmačiu atveju.

2.3 pavyzdys

Tegu $\mathbf{D} = [-2, 2], f(x)-$ *Lipšico funkcija. Vienmačiu atveju pirminis padalijimas yra trivialus: visa leistinoji sritis yra pirminis simpleksas. Pirmiausia jis dalijimas į du per vidurio tašką (pirmasis algoritmo žingsnis), kurie toliau*



2.7 pav. *Simpleksinio šakų ir rėžių algoritmo antrasis žingsnis*



2.8 pav. *Simpleksinio šakų ir rėžių algoritmo trečiasis žingsnis*

dalijami per vidurio tašką į du naujus (antrasis algoritmo žingsnis). Šiame algoritmo etape gaunami keturi simpleksai $\mathbf{I}_i, i = 1, \ldots, 4$ *(žr. 2.7 pav.). Kiekvienam iš simpleksų apskaičiuojami viršutiniai rėžiai pagal skirtingas taisykles. Vienmačiu atveju visos normos lygios, todėl užtenka įvertinti rėžius* φ, ψ *ir* μ, *neakcentuojant pagal kurias normas jie skaičiuoti. Naudojant* μ *tipo rėžius,*

visiems simpleksams $\mathbf{I}_i : \mu(\mathbf{I}_i) > LB, i = 1, \ldots, 4$, *tačiau naudojant* φ *ir* ψ *tipo rėžius gauname* $\varphi(\mathbf{I}_4) < LB$ *ir* $\psi(\mathbf{I}_4) < LB$. *Todėl* φ *ir* ψ *tipo rėžių atveju simpleksas* $\mathbf{I}_4$ *yra pašalinamas ir nedalyvauja tolesniame šakų ir rėžių algoritme. Nepašalinti simpleksai toliau yra dalijami (žr. 2.8 pav.). Gauname 6 simpleksus* φ *ir* ψ *rėžių atveju ir 8 simpleksus* μ *rėžio atveju. Naujai gautiems simpleksams gauname,* $\mu(\mathbf{I}_i) < LB$ *kai* $i = 4, 5, 6, 7$, $\varphi(\mathbf{I}_i) < LB$ *kai* $j = 3, 4, 5, 6$ *ir* $\psi(\mathbf{I}_i) < LB$ *kai* $j = 3, 4, 5, 6$. *Po šio žingsnio* φ *ir* ψ *rėžių atveju lieka du simpleksai* $\{\mathbf{I}_1, \mathbf{I}_2\}$, *o* μ *rėžio atveju - keturi* $\{\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3, \mathbf{I}_4\}$. *Todėl naudojant* φ *ir* ψ *tipo rėžius, galime greičiau aptikti ir pašalinti iš tolesnės paieškos sritis, kuriose negali būti globaliojo maksimumo.*

2.7. Lygiagretieji simpleksiniai šakų ir rėžių algoritmai su Lipšico rėžiais

Lipšico optimizavimo algoritmai reikalauja daug skaičiavimų resursų, todėl nuosekliųjų algoritmų lygiagrečiosios versijos yra svarbi ir šiuo metu aktyviai vystoma sritis [12, 17, 28, 81, 89].

Tuo tikslu disertacijoje realizuota dvi lygiagrečiosios pasiūlytojo nuosekliojo algoritmo (žr. 3 algoritmą) versijos: bendrosios atminties kompiuteriams (OpenMP versija) ir paskirstytosios atminties kompiuteriams (MPI versija).

2.7.1. Bendrosios atminties kompiuteriams skirtas lygiagretusis algoritmas

Lygiagrečioji algoritmo versija, skirta bendrosios atminties kompiuteriams, realizuota OpenMP 2.0 standarto pagrindu. Joje realizuotos tos pačios padengimo, išrinkimo, padalijimo ir rėžių skaičiavimo taisyklės, kaip ir nuosekliojoje algoritmo versijoje.

Šioje lygiagrečioje versijoje realizuota galimybė po simplekso padalijimo atlikti patikrinimą, ar naujai gautoje viršūnėje nebuvo apskaičiuota funkcijos reikšmė anksčiau. T.y. prieš apskaičiuojant funkcijos reikšmę naujai gautoje simplekso viršūnėje, atliekamas patikrinimas visų iki tol surastų viršūnių aibėje, ar ta viršūnė nebuvo rasta ankstesniame šakų ir rėžių algoritmo žingsnyje. Jeigu funkcijos reikšmė tame taške buvo apskaičiuota anksčiau, tai ta reikšmė paimama iš apskaičiuotų viršūnėse reikšmių aibės. Ši modifikacija leidžia išvengti pasikartojančių funkcijos reikšmių skaičiavimų toje pačioje viršūnėje, todėl yra prasminga atlikti „brangių“ funkcijų atveju.

Lygiagretusis (OpenMP) simpleksinis šakų ir rėžių algoritmas pavaizduotas 4 algoritme. Duomenų lygiagretinimas naudojamas. C/C++ programavimo kalboje OpenMP direktyvos nurodomos pradedant #pragma žodžiu. Direktyva „for“ nurodo, kad ciklo iteracijos atliekamos lygiagrečiai skirtinguose procesuose. Direktyva „schedule (static)“ nurodo, kad ciklo iteracijos yra po lygiai (jeigu įmanoma) išdalinamos ir statiškai paskirstomos tarp skirtingų procesų. Direktyva „critical“ nurodo, kuri algoritmo dalis turi būti vykdoma tik vieno proceso tam tikru laiko momentu.

4 algoritmas Lygiagretusis (OpenMP) simpleksinis šakų ir rėžių algoritmas su Lipšico rėžiais

1: Leistinoji sritis $\mathbf{D}$ padengiama n-mačiais simpleksais $\mathbb{I} \leftarrow \{\mathbf{I}_j | \mathbf{D} = \cup \mathbf{I}_j, j = 1, \ldots, n!\}$ naudojant viršūnių trianguliaciją
2: $\mathbf{S} \leftarrow \emptyset$, $LB(\mathbf{D}) \leftarrow -\infty$
3: **while** $(\mathbb{I} \neq \emptyset)$ **do**
4: $k = |\mathbb{I}|$
5: **#pragma omp parallel private** (UB)
6: **#pragma omp for schedule (static)**
7: **for** $(j = 0;\ j <= k;\ j{+}{+})$ **do**
8: Pasirenkamas $\mathbf{I}_j \in \mathbb{I}$ naudojant vieną iš (geryn, gilyn, platyn, statistinė) išrinkimo strategijų, $\mathbb{I} \leftarrow \mathbb{I} \backslash \{\mathbf{I}_j\}$
9: **#pragma omp critical** $(LB(\mathbf{D}))$
10: $LB(\mathbf{D}) \leftarrow \max\{LB(\mathbf{D}), \max\limits_{v \in V(\mathbf{I}_j)} f(v)\}$
11: **#pragma omp critical** $(\mathbf{S})$
12: $\mathbf{S} \leftarrow \arg\max\{f(\mathbf{S}), \max\limits_{v \in V(\mathbf{I}_j)} f(v)\}$
13: Apskaičiuojamas $UB(\mathbf{I}_j)$ pagal μ, φ, ψ tipo sudarytą rėžį su įvairiomis normomis.
14: **if** $(UB(\mathbf{I}_j) > LB(\mathbf{D}) + \varepsilon)$ **then**
15: Padalijamas $\mathbf{I}_j$ į $\mathbf{I}_{j_1}$, $\mathbf{I}_{j_2}$ hiperplokštuma, einančia per ilgiausios briauunos vidurį ir viršūnes, nepriklausančias ilgiausiajai briaunai
16: Patikrinama ar gauta po dalijimo viršūnė yra „nauja“
17: **#pragma omp critical** $(\mathbb{I})$
18: $\mathbb{I} \leftarrow \mathbb{I} \cup \{\mathbf{I}_{j_1}, \mathbf{I}_{j_2}\}$
19: **end if**
20: **end for**
21: **end while**

2.7.2. Paskirstytosios atminties kompiuteriams skirtas lygiagretusis algoritmas

Lygiagrečioji algoritmo versija, skirta paskirstytosios atminties kompiuteriams, realizuota naudojantis MPI 1.2 standarto pagrindu sukurtu šakų ir rėžių algoritmo šablonu [2, 3]. Pradinis (statinis) užduočių paskirstymas yra naudojamas. Pradinės užduotys iš pradžių atliekamos tik procesoriaus šeimininko, kol pradinis užduočių skaičius pasidaro didesnis už procesorių skaičių. Tuomet užduotys atsitiktinai lygiomis dalimis (jei įmanoma) padalijamos tarp procesorių darbininkų. Vėliau užduotys neperskirstomos tarp procesorių, o kiekvienas procesorius atlieka tą patį algoritmą, kuris pavaizduotas 5 algoritme.

Šios schemos trūkumas yra tas, kad procesoriai gali užbaigti skaičiavimus skirtingu metu, o bendra sprendimo trukmė apsprendžiama ilgiausiai skaičiavusio procesoriaus darbo trukme.

MPI algoritmo versijoje realizuotos tos pačios padengimo, išrinkimo, padalijimo ir rėžių skaičiavimo taisyklės, kaip ir nuoseklioje algoritmo versijoje:

5 algoritmas Lygiagretusis (MPI) simpleksinis šakų ir rėžių algoritmas su Lipšico rėžiais

1: Leistinoji sritis $\mathbf{D}$ padengiama n-mačiais simpleksais $\mathbb{I} \leftarrow \{\mathbf{I}_j | \mathbf{D} = \cup \mathbf{I}_j, j = 1, \ldots, n!\}$ naudojant viršūnių trianguliaciją
2: Padalijama $\mathbb{I}$ po lygiai (jei įmanoma) tarp p procesorių $\mathbb{I} = \cup \mathbb{I}^r, |\mathbb{I}^r| \approx |\mathbb{I}|/p$
3: $\mathbf{S}^r \leftarrow \emptyset$, $LB(\mathbb{I}^r) \leftarrow -\infty$.
4: **while** $(\mathbb{I}^r \neq \emptyset)$ **do**
5: Pasirenkamas $\mathbf{I}_j^r \in \mathbb{I}^r$ naudojant vieną iš (geryn, gilyn, platyn, statistinė) išrinkimo strategijų, $\mathbb{I}^r \leftarrow \mathbb{I}^r \backslash \{\mathbf{I}_j^r\}$
6: $LB(\mathbb{I}^r) \leftarrow \max\{LB(\mathbb{I}^r), \max\limits_{v \in V(\mathbf{I}_j^r)} f(v)\}$
7: Persiunčiama geriausia surasta funkcijos reikšmė $LB(\mathbb{I}^r)$ tarp procesorių
8: $\mathbf{S}^r \leftarrow \arg\max\{f(\mathbf{S}^r), \max\limits_{v \in V(\mathbf{I}_j^r)} f(v)\}$
9: Apskaičiuojamas $UB(\mathbf{I}_j^r)$ pagal μ, φ, ψ tipo sudarytą rėžį su įvairiomis normomis.
10: **if** $(UB(\mathbf{I}_j^r) > LB(\mathbb{I}^r) + \varepsilon)$ **then**
11: Padalijamas $\mathbf{I}_j^r$ į $\mathbf{I}_{j_1}^r$, $\mathbf{I}_{j_2}^r$ hiperplokštuma, einančia per ilgiausios briaunos vidurį ir viršūnes, nepriklausančias ilgiausiajai briaunai
12: $\mathbb{I}^r \leftarrow \mathbb{I}^r \cup \{\mathbf{I}_{j_1}^r, \mathbf{I}_{j_1}^r\}$
13: **end if**
14: **end while**
15: Surenkami rezultatai $\mathbf{S}^r$

- Kiekvienas procesorius gauna po lygią (jeigu įmanoma) dalį neišspręstų posričių. Simboliškai tai pavaizduota $\mathbb{I} = \cup\mathbb{I}^r, |\mathbb{I}^r| \approx |\mathbb{I}|/p$, čia p – procesorių skaičius, r – procesoriaus numeris.
- Geriausia tikslo funkcijos reikšmė $LB(\mathbb{I}^r)$ yra lokali, todėl baziniu atveju procesoriai ja nesikeičia. Modifikuotoje algoritmo versijoje - ja apsikeičia, kai tik kažkuris procesorius randa geresnę už jau žinomą.
- Po užbaigimo visų procesorių rezultatai surenkami ir geriausia rasta reikšmė yra ieškomo globaliojo maksimumo aproksimacija.

2.8. Antrojo skyriaus apibendrinimas

1. Pasiūlyta Lipšico šakų ir rėžių algoritmuose viršutinių μ_2 tipo rėžių skaičiavimui vietoj vienos konkrečios normos apjungti kelias normas, kurių dėka gaunami tikslesni Lipšico rėžiai nenaudojant papildomų tikslo funkcijos skaičiavimų.
2. Dvimačiu atveju parodyta, kad geriausiai normų atžvilgiu μ_2 tipo rėžius patiksliname naudojant pasiūlytą rėžį $\mu_2^{1,\infty}$ su pirmąja ir begaline normomis.
3. Daugiamačiu atveju ($n \geq 3$) parodyta, kad pasiūlytas rėžis $\mu_2^{1,\infty}$ yra ne visuomet tikslesnis, negu skaičiuojant μ_2 pagal vieną konkrečią normą.
4. Pasiūlyta, kaip naudojant pirmą normą galima Pijavskij (φ) tipo rėžius apskaičiuoti sprendžiant tiesines lygčių sistemas vietoj kvadratinių arba kvadratinių ir tiesinių lygčių Euklidinės normos atveju.
5. Pasiūlytas naujas ψ tipo rėžis, pagrįstas apibrėžtinės daugiamatės sferos spinduliu. Šis rėžis gali būti naudojamas kaip alternatyva μ_2 ir φ tipo rėžiams.
6. Sukurtas nuoseklusis simpleksinis šakų ir rėžių algoritmas su pasiūlytais rėžiais, bei dvi lygiagrečios šio algoritmo versijos skirtos bendrosios ir paskirstytosios atminties kompiuteriams.

3

Lipšico optimizavimo algoritmų su simpleksiniais posričiais eksperimentinis tyrimas

Lipšico optimizavime rėžių skaičiavimui dažniausiai naudojama Euklidinė norma, tačiau kitų normų naudojimas gali pagerinti optimizavimo rezultatus. 2 skyriuje gauti teoriniai rezultatai rodo, kad kitų normų naudojimas patikslina viršutinius rėžius, o Pijavskij tipo rėžio atveju – palengvina jų apskaičiavimą. Tačiau ar reikšmingi gautieji teoriniai rezultatai sprendžiant globaliojo optimizavimo uždavinius? Ar pasiūlytas ψ tipo rėžis yra tinkamas naudoti, kaip alternatyva μ_2 ir φ tipo rėžiams?

Šiame skyriuje atliksime eksperimentinius pasiūlytų rėžių $\mu_2^{1,\infty}$, φ^1, ψ^2 tyrimus dvimačiu [A1] ir daugiamačiu [A2, A3] leistinųjų sričių atvejais. Pagal funkcijų skaičiavimo kiekio kriterijų palyginsime pasiūlyto nuoseklaus simpleksinio šakų ir rėžių algoritmo efektyvumą priklausomai nuo naudojamų Lipšico rėžių [A3, A7, A4] bei gautus rezultatus palyginsime su kitais gerai žinomais Lipšico optimizavimo algoritmų rezultatais [A4]. Ištirsime kurias paieškos strategijas ir kada tikslinga naudoti nuosekliuosiuose bei lygiagrečiuosiuose šakų ir rėžių algoritmuose [A6] bei palyginsime lygiagrečiųjų algoritmų, skirtų bendrosios ir paskirstytosios atminties kompiuteriams efektyvumą [A8, A5].

3.1. μ_2 tipo rėžių su įvairiomis normomis efektyvumo tyrimas

Naudojant 1.6 poskyryje aprašytus Lipšico testo uždavinius, atliktas pasiūlyto nuosekliojo simpleksinio šakų ir rėžių algoritmo (žr. 3 algoritmą) eksperimentinis tyrimas patvirtino 2.3.1 poskyryje gautus teorinius rezultatus. Dvimačiu ($n = 2$) atveju (žr. 3.1 lentelę) nėra vienos konkrečios normos ir ją atitinkančios Lipšico konstantos, su kuria gautume geriausius rezultatus visiems testo uždaviniams. Tačiau naudojant pasiūlytą patikslintą rėžį $\mu_2^{1,\infty}$ (2.7) visiems testo uždaviniams gauname geriausius rezultatus. Naudojant rėžį $\mu_2^{1,\infty}$ dvimačiu ($n = 2$) atveju funkcijų skaičiavimo kiekis vidutiniškai yra apie 21% mažesnis, nei μ_2 tipo viršutinį rėžį skaičiuojant su Euklidine norma μ_2^2.

Kai leistinoji sritis yra nekubinė arba erdvės dimensija ($n \geq 3$), 2.3.2 poskyryje parodyta, kad naudojant simpleksinius posričius ne visuomet tiksliausius rėžius gauname naudojant rėžį $\mu_2^{1,\infty}$. Tas atsispindi gautuosiuose rezultatuose. Daliai testo uždavinių geresni rezultatai gaunami naudojant kitas normas. Eksperimentiniu būdu pastebėta, kad iš kitų normų, dažniausiai geriausius rezultatus duoda Euklidinė norma. Todėl pasiūlyta viršutinio rėžio skaičiavimui daugiamačiu atveju rėžį $\mu_2^{1,\infty}$ papildyti Euklidine norma

$$\mu_2^{1,2,\infty}(\mathbf{I}) = \min_{v \in V(\mathbf{I})} \{f(v) + K(V(\mathbf{I}))\}, \tag{3.1}$$

čia

$$K(V(\mathbf{I})) = \min \left\{ L_\infty \max_{x \in V(\mathbf{I})} \|x - v\|_1, L_2 \max_{x \in V(\mathbf{I})} \|x - v\|_2, L_1 \max_{x \in V(\mathbf{I})} \|x - v\|_\infty \right\}.$$

Dvimačiu atveju Euklidinės normos papildymas yra naudingas tik nekubinėms sritims. Iš trylikos dvimačių testo uždavinių, vienintelio dešimtojo uždavinio leistinoji sritis yra stačiakampė. Pastebime, kad jai vienintelei funkcijos skaičiavimų kiekis naudojant rėžį $\mu_2^{1,2,\infty}$ gavosi mažesnis, nei su rėžiu $\mu_2^{1,\infty}$. Didesnės dimensijos atveju ($n \geq 3$) rėžis $\mu_2^{1,2,\infty}$ beveik visiems testo uždaviniams pagerina rezultatus lyginant su rėžiu $\mu_2^{1,\infty}$. Vidutiniškai funkcijų skaičiavimų kiekis yra apie 5% mažesnis nei su $\mu_2^{1,\infty}$ ir apie 52% nei su viena Euklidine norma μ_2^2.

Naudojant kai kuriuos rėžius dalis optimizavimo uždavinių nebuvo iš viso išspręsta su pasirinktu maksimaliu tikslo funkcijų skaičiavimų kiekiu – 10000000.

3.1 lentelė. *Tikslo funkcijų skaičiavimo kiekis optimizuojant simpleksiniu šakų ir rėžių algoritmu su μ_2^1, μ_2^2, μ_2^∞ rėžiais bei su patikslintais $\mu_2^{1,\infty}$ ir $\mu_2^{1,2,\infty}$ rėžiais*

Nr.	μ_2^1	μ_2^2	μ_2^∞	$\mu_2^{1,\infty}$	$\mu_2^{1,2,\infty}$
1.	2019	1356	970	970	970
2.	321	299	246	216	216
3.	10700	8935	10825	7539	7539
4.	40	52	72	8	8
5.	61	126	153	61	61
6.	2864	2046	1953	1751	1751
7.	34814	23100	24098	20200	20200
8.	528	729	1042	521	521
9.	73333	42844	46233	40314	40314
10.	2421	3055	4397	2369	2297
11.	6783	6986	9426	5654	5654
12.	29369	37756	58690	29357	29357
13.	44908	37774	22262	22241	22241
vid.	**16012**	**12697**	**13874**	**10092**	**10087**
14.	4157824	6043338	5693624	4157824	4157740
15.	51730	14140	3864	3864	3864
16.	>10000000	>10000000	3145728	3145728	3145728
17.	1663261	2702422	3721012	1596633	1596633
18.	47080	19632	38888	38792	18720
19.	25292	50812	60088	24748	24748
20.	904186	713398	777482	777478	668510
vid.	**>2407053**	**>2791963**	**1920098**	**1392152**	**1373706**
21.	>10000000	>10000000	6496801	6496791	6264147
22.	256557	225298	721331	239217	175107
23.	556488	1048292	5347547	556488	556412
24.	1031808	1066128	5353683	1031808	1031808
25.	573359	1066128	5359481	573281	573025
26.	2036200	1565127	5801659	1852688	1253836
27.	572414	496904	3818039	549730	438141
28.	408540	489831	1933931	384746	331977
vid.	**>1929421**	**>1994714**	**4354059**	**1460594**	**1328057**
29.	>10000000	7914387	1917124	1917124	1917092
30.	6936368	8284881	>10000000	6932546	6239743
31.	6490797	8535473	>10000000	6482327	6099520
vid.	**>7809055**	**8244914**	**>7305708**	**5110666**	**4752118**
32.	>10000000	6269636	823320	823320	821892
33.	1926497	7419819	>10000000	1926497	1919569
vid.	**>5963249**	**6844728**	**>5411660**	**1374909**	**1370731**

3.2. Pijavskij tipo rėžio su pirmąja norma (φ^1) efektyvumo tyrimas

Pijavskij (φ) tipo rėžiai pasižymi tuo, kad žinant funkcijos reikšmes tam tikruose leistinosios srities taškuose bei Lipšico konstantą L pagal šiuos rėžius gaunami tiksliausi įmanomi Lipšico rėžiai. Tačiau bendru atveju šių rėžių radimas yra sudėtingas optimizavimo uždavinys. Naudojant Euklidinę normą šie rėžiai apskaičiuojami sprendžiant kvadratinių lygčių [100] arba kvadratinių ir tiesinių lygčių [82] sistemas. Tačiau naudojant pirmąją normą 2.4 skyriuje pasiūlyta, kaip galima šio tipo rėžius apskaičiuoti sprendžiant tik tiesinių lygčių sistemas.

Atliekant eksperimentinius tyrimus, pirmiausia lygintas tikslo funkcijų skaičiavimų kiekis viršutinį rėžį skaičiuojant pagal φ ir μ_2 tipo rėžius su pirmąja norma φ^1 ir μ_2^1. Iš 3.2 lentelėje pateiktų rezultatų matyti, kad visiems testo uždaviniams geresni rezultatai gaunami skaičiuojant viršutinį rėžį pagal φ^1 negu pagal μ_2^1. Naudojant rėžį φ^1 tikslo funkcijos skaičiavimų kiekis vidutiniškai apie 18% mažesnis negu naudojant μ_2^1 rėžį.

Atlikti eksperimentiniai tyrimai (žr. 3.1 lentelę) patvirtino, kad nėra vienos konkrečios normos ir ją atitinkančios konstantos, su kuria būtų gauti geriausi rezultatai visiems testo uždaviniams. Todėl ir Pijavskij tipo rėžis pirmai normai φ^1 yra ne visoms funkcijoms efektyvus. Iš 3.2 rezultatų lentelės matyti, kad toms funkcijoms, kurioms pirma norma yra neefektyvi, pasiūlytas rėžis φ^1 duoda blogesnius rezultatus lyginant su rėžiu $\mu_2^{1,2,\infty}$. Todėl siekiant sumažinti funkcijų skaičiavimų kiekį, neskaičiuojant naujų funkcijos reikšmių, tikslinga būtų rėžį μ_2^1 pakeisti Pijavskij tipo rėžių φ^1. Tokiu būdu gaunamas naujas agreguotasis rėžis

$$\varphi^1\mu_2^{2,\infty}(\mathbf{I}) = \min\left\{\varphi^1(\mathbf{I}), \mu_2^{2,\infty}(\mathbf{I})\right\} \tag{3.2}$$

$$= \min\left\{\max_{x\in\mathbf{I}}\left(\min_{v\in V(\mathbf{I})}\left\{f(v) + L_\infty \left\|x - v\right\|_1\right\}\right), \min_{v\in V(\mathbf{I})}\left\{f(v) + K'(V(\mathbf{I}))\right\}\right\},$$

čia

$$K'(V(\mathbf{I})) = \min\left\{L_1 \max_{x\in\mathbf{I}} \left\|x - v\right\|_\infty, L_2 \max_{x\in\mathbf{I}} \left\|x - v\right\|_2\right\}.$$

Naudojant šį agreguotąjį rėžį vidutiniškai gauname apie 9% mažesnį funkcijų skaičiavimų kiekį nei rėžio $\mu_2^{1,2,\infty}$ atveju ir apie 25% mažesnį lyginant su rėžiu φ^1.

3.2 lentelė. *Tikslo funkcijų skaičiavimo kiekis optimizuojant simpleksiniu šakų ir rėžių algoritmu su μ_2^1 ir φ^1 rėžiais, bei su patikslintu $\mu_2^{1,2,\infty}$ ir agreguotu $\varphi^1\mu_2^{2,\infty}$ rėžiais*

Nr.	μ_2^1	φ^1	$\mu_2^{1,2,\infty}$	$\varphi^1\mu_2^{2,\infty}$
1.	2019	1668	970	967
2.	321	215	216	188
3.	10700	8264	7539	6653
4.	40	28	8	8
5.	61	39	61	39
6.	2864	2465	1751	1723
7.	34814	26792	20200	19171
8.	528	381	521	380
9.	73333	54630	40314	38860
10.	2421	1866	2297	1807
11.	6783	5416	5654	4789
12.	29369	21160	29357	21153
13.	44908	33917	22241	22094
vid.	**16012**	**12065**	**10087**	**9064**
14.	4157824	3423524	4157740	3423480
15.	51730	41212	3864	3863
16.	>10000000	>10000000	3145728	3145728
17.	1663261	1410165	1596633	1363004
18.	47080	34464	18720	16884
19.	25292	20876	24748	20487
20.	904186	733234	668510	547622
vid.	**>2407053**	**>2237639**	**1373706**	**1217295**
21.	>10000000	>10000000	6264147	6262233
22.	256557	240901	175107	167217
23.	556488	536356	556412	536280
24.	1031808	988493	1031808	988493
25.	573359	544034	573025	543712
26.	2036200	1867961	1253836	1176018
27.	572414	546240	438141	420417
28.	408540	383084	331977	314023
vid.	**>1929421**	**>1888384**	**1328057**	**1301049**
29.	>10000000	>10000000	1917092	1916941
30.	6936368	6730455	6239743	6064924
31.	6490797	5641951	6099520	5940304
vid.	**>7809055**	**>7682866**	**4752118**	**4640723**
32.	>10000000	>10000000	821892	821892
33.	1926497	1875911	1919569	1868983
vid.	**>5963249**	**>5937956**	**1370731**	**1345438**

3.3. ψ tipo rėžio su Euklidine norma (ψ^2) efektyvumo tyrimas

Pasiūlyto rėžio ψ^2 (2.24) eksperimentinis tyrimas pateiktas 3.3 lentelėje. Pirmiausia palyginti rezultatai, gauti naudojant ψ^2 ir μ_2^2 tipo rėžius. Daugumai testinių uždavinių funkcijų skaičiavimų kiekis naudojant ψ^2 yra mažesnis nei su μ_2^2. Vidutiniškai dvimačiams ir trimačiams ($n = 2, 3$) uždaviniams funkcijų skaičiavimų kiekis yra apie 47%, o didesnio matmens ($n \geq 4$) testo uždaviniams funkcijų skaičiavimų kiekis apie 26% mažesnis negu naudojant rėžį μ_2^2.

Santykis r_{ψ^2/μ_2^2} yra lygus viršūnių skaičiui, kuriose rėžis ψ^2 yra geresnis nei rėžis μ_2^2, padalintas iš viso funkcijų skaičiavimo kiekio. Natūralu, kad kuo šis santykis artimesnis vienetui, tuo dažniau tikslesnius viršutinius rėžius gauname skaičiuojant pagal ψ^2 taisyklę lyginant su μ_2^2. Iš santykio r_{ψ^2/μ_2^2} gautų reikšmių pastebime, kad ne visose viršūnėse rėžis ψ^2 yra geresnis už μ_2^2, ypač didėjant erdvės dimensijai. Pirmos normos atveju Pijavskij rėžis φ^1 visuomet geresnis už μ_2^1, todėl rėžyje $\mu_2^{1,2,\infty}$ pakeitę rėžį μ_2^1 geresniu rėžiu φ^1, gavome tikslesnį agreguotąjį rėžį $\varphi^1\mu_2^{2,\infty}$ (3.2). Šiuo atveju nebūtų tikslinga rėžį μ_2^2 pakeisti rėžiu ψ^2, nes ne visuomet pastarasis yra geresnis. Todėl agreguotąjį rėžį $\varphi^1\mu_2^{2,\infty}$ papildę rėžiu ψ^2, gauname tikslesnį agreguotąjį rėžį

$$\varphi^1\psi^2\mu_2^{2,\infty}(\mathbf{I}) = \min\left\{\varphi^1\mu_2^{2,\infty}(\mathbf{I}), \psi^2(\mathbf{I})\right\}, \tag{3.3}$$

kuris vidutiniškai apie 20% sumažina funkcijų skaičiavimų kiekį lyginant su agreguotuoju rėžiu $\varphi^1\mu_2^{2,\infty}$.

Siekiant išvengti pasikartojančių funkcijos reikšmių skaičiavimų toje pačioje viršūnėje pasiūlytame 4 algoritme realizuota galimybė atlikti patikrinimą ar gauta po dalijimo viršūnė yra „nauja“, t.y. prieš apskaičiuojant funkcijos reikšmę naujai gautoje simplekso viršūnėje, atliekamas patikrinimas visų iki tol surastų viršūnių aibėje, ar ta viršūnė nebuvo rasta ankstesniame šakų ir rėžių algoritmo žingsnyje. Jeigu funkcijos reikšmė tame taške buvo apskaičiuota anksčiau, tai reikšmė paimama iš apskaičiuotų viršūnėse reikšmių aibės. Atlikus viršūnių patikrinimą su agreguotuoju rėžiu $\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$ funkcijų skaičiavimų kiekis vidutiniškai sumažėja nuo 1,91 karto dvimačiu iki 62,73 kartų šešiamačiu atvejais.

3.3 lentelė. *Tikslo funkcijų skaičiavimo kiekis optimizuojant simpleksiniu šakų ir rėžių algoritmu su μ_2^2 ir ψ^2 rėžiais, agreguotaisiais $\varphi^1\mu_2^{2,\infty}$ ir $\varphi^1\psi^2\mu_2^{2,\infty}$ rėžiais bei su agreguotuoju rėžiu ir viršūnių patikrinimu $\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$*

Nr.	μ_2^2	ψ^2	$\varphi^1\mu_2^{2,\infty}$	$\varphi^1\psi^2\mu_2^{2,\infty}$	$\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$	r_{ψ^2/μ_2^2}
1.	1356	856	967	716	412	0,93
2.	299	286	188	185	122	0,39
3.	8935	5048	6653	4843	2551	1,00
4.	52	128	8	8	5	0,02
5.	126	155	39	39	36	0,06
6.	2046	1284	1723	1241	691	0,99
7.	23100	12547	19171	12195	6240	0,99
8.	729	479	380	341	212	0,96
9.	42844	22038	38860	21724	11049	1,00
10.	3055	1734	1807	1495	830	0,98
11.	6986	3915	4789	3598	1931	0,00
12.	37756	19211	21153	16938	8926	1,00
13.	37774	20366	22094	18737	9855	1,00
vid.	**12697**	**6773**	**9064**	**6312**	**3297**	**0,72**
14.	6043338	2642392	3423480	2355490	495711	0,99
15.	14140	11928	3863	3784	1030	0,64
16.	>10000000	6325269	3145728	3145728	536761	1,00
17.	2702422	1131286	1363004	1020782	229049	0,95
18.	19632	14368	16884	12032	3091	0,75
19.	50812	20776	20487	17105	4684	0,91
20.	713398	281998	547622	262311	55605	0,98
vid.	**>2791963**	**1489717**	**1217295**	**973890**	**189419**	**0,89**
21.	>10000000	7359930	6262233	5349848	540697	0,64
22.	225298	182447	167217	120144	13707	0,59
23.	1048292	635716	536280	450084	44421	0,67
24.	1066128	635832	988493	563835	68502	0,67
25.	1066128	635922	543712	465955	44991	0,67
26.	1565127	965474	1176018	749518	52078	0,65
27.	496904	426493	420417	333568	5769	0,64
28.	489831	918077	314023	271614	37981	0,24
vid.	**>1994714**	**1469986**	**1301049**	**1038071**	**101018**	**0,60**
29.	7914387	5826460	1916941	1633849	84406	0,57
30.	8284881	8079412	6064924	4590448	162989	0,52
31.	8535473	11291062	5940304	5192437	256963	0,47
vid.	**8244914**	**8398978**	**4640723**	**3805578**	**168119**	**0,52**
32.	6269636	1623674	821892	524940	9840	0,57
33.	7419819	6818423	1868983	1685793	25398	0,50
vid.	**6844728**	**4221049**	**1345438**	**1105367**	**17619**	**0,53**

3.4. Palyginimas su kitais Lipšico optimizavimo algoritmais

Gautieji rezultatai naudojant pasiūlytą simpleksinį šakų ir rėžių algoritmą su agreguotuoju rėžiu ir viršūnių patikrinimu $\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$ palyginti su kitais gerais žinomais Lipšico optimizavimo algoritmais, apžvelgtais 1.4 poskyryje bei eksperimentiškai palygintais [49]. Išskirtinos dvi pagrindinės Lipšico algoritmų klasės:

1. Algoritmai, naudojantys vieną viršutinio rėžio funkciją (t.y. įvairūs Pijavskij algoritmo [99, 100] variantai):
   - Mladineo (Mla) [82],
   - Jaumard, Herrmann ir Ribault (JHR) [64],
   - Wood (Wood) [122, 123].
2. Šakų ir rėžių algoritmai:
   - Pasiūlytas simpleksinis šakų ir rėžių algoritmas su rėžiu $\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$,
   - Galperin (Gal85, Gal88) [38, 40],
   - Pinter (Pint) [94],
   - Meewella and Mayne (MM) [78],
   - Gourdin, Hansen and Joumard (GHJ) [46].

Gauti rezultatai pateikti 3.4 lentelėje. Algoritmus realizavę autoriai [49] pastebi, kad visi algoritmai realizuotomis priemonėmis išsprendė optimizavimo uždavinį tik tuomet, kai reikalaujamas tikslumas (ε) nėra labai mažas. Taip pat jie pastebi, kad kai kuriais algoritmais dalis optimizavimo uždavinių nebuvo iš viso išspręsti per fiksuotą laiko tarpą arba pritrūkus turimos atminties.

Lyginant nagrinėjamų Lipšico optimizavimo algoritmų rezultatus mažiausias funkcijų skaičiavimo kiekis gaunamas sprendžiant Mladineo (Mla86) ir Jaumard, Herrmann, Ribault (JHR) algoritmais. Tačiau šie algoritmai priklauso pirmąjai algoritmų klasei, kurių sprendimas reikalauja gerokai daugiau laiko negu šakų ir rėžių klasės algoritmai, ypač didėjant erdvės dimensijai. Dėl to su šios klasės algoritmais $n = 3$ atveju dažnai nebuvo surastas sprendinys per fiksuotą maksimalų laiką, kai tuo tarpu beveik visi nagrinėti šakų ir rėžių algoritmai beveik visada surasdavo sprendinį per mažesnį, nei fiksuotas maksimalus laiko tarpas. Todėl pirmosios klasės algoritmų naudojimas pasiteisina tik brangių tikslo funkcijų ir mažo matmenų skaičiaus atveju.

3.4 lentelė. *Pasiūlyto simpleksinio šakų ir rėžių algoritmo su agreguotuoju rėžiu ir viršūnių patikrinimu $\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$ palyginimas (pagal tikslo funkcijų skaičiavimų kiekį) su kitais Lipšico optimizavimo algoritmais*

Nr.	Mla	JHR	Wood	$\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$	Gal85	Gal88	Pint	MM	GHJ
1.	320	323	5528	**412**	3553	1713	3807	1749	643
2.	80	80	2861	**122**	1036	577	1762	744	167
3.	2066	2066	70955	**2551**	24214	16089	28417	10839	3531
4.	6	6	157	**5**	106	73	1527	94	45
5.	41	41	209	**36**	430	217	907	424	73
6.	548	548	14740	**691**	7729	2929	7772	2684	969
7.	-	5088	183759	**6240**	43123	34705	62917	22799	7969
8.	177	177	1403	**212**	2113	1289	2272	964	301
9.	-	8838	309763	**11049**	57814	49873	88932	53549	13953
10.	673	673	18613	**830**	8508	5628	9022	3814	1123
11.	1613	1613	53348	**1931**	18235	12737	20312	9224	2677
12.	-	8414	470200	**8926**	63088	56177	105572	45389	12643
13.	-	9617	-	**9855**	65536	59049	109227	35949	15695
14.	>460	>41700	-	495711	5383113	3886897	-	-	**215061**
15.	>290	9363	-	**1030**	635909	347075	-	-	24249
16.	>290	>12000	-	**536761**	15620627	-	-	-	1297205
17.	>280	>14400	-	**229049**	12481708	-	-	-	268279
18.	>690	1309	-	**3091**	46411	23765	-	-	3219
19.	446	445	-	**4684**	35463	18669	-	-	7177

Pasiūlytas simpleksinis šakų ir rėžių algoritmas beveik visiems testo uždaviniams duoda geresnius rezultatus, lyginant su geriausiu apžvelgtu šakų ir rėžių Lipšico optimizavimo algoritmu (GHJ).

3.5. Lygiagrečiųjų algoritmų efektyvumo tyrimas

Lygiagrečiosios pasiūlyto simpleksinio šakų ir rėžių algoritmo versijos bendrosios atminties kompiuteriams (žr. 4 algoritmą) ir paskirstytosios atminties kompiuteriams (žr. 5 algoritmą) pateiktos 3.5 poskyryje. Šiame poskyryje pateikti pasiūlytų lygiagrečiųjų algoritmų efektyvumo tyrimai.

Lygiagrečiojo OpenMP algoritmo eksperimentiniai skaičiavimai atlikti Edinburgo lygiagrečiųjų skaičiavimų centro (EPCC) bendrosios atminties skaičiavimo mašinoje Ness (http://www.epcc.ed.ac.uk/facilities/ness/). Šioje skaičiavimo mašinoje realizuota galimybė tirti tiek OpenMP, tiek ir MPI sistemoms skirtus algoritmus. Lygiagrečiąją mašiną Ness sudaro du mazgai X4600 SMP, kurių kiekvienas turi po 16 procesorių-branduolių (2.6 GHz AMD Opte-

ron (AMD64e)) ir 2GB operatyviosios atminties. Iki 16 procesorių-branduolių naudota eksperimentiniuose tyrimuose.

Lygiagrečiojo MPI algoritmo eksperimentiniai skaičiavimai atlikti Vilniaus Gedimino technikos universiteto klasteryje VILKAS (http://vilkas.vgtu.lt/). Klasterį VILKAS sudaro 14 mazgų su Intel Core 2 Quad Q6600 (2.4GHz, 4 branduoliai) ir 9 mazgai su Intel Core i7-860 (2.8GHz, 4 branduoliai, 8 sijos) procesoriais, sujungtų į Gigabit Ethernet lokalųjį tinklą. Iki 16 branduolių Intel Core i7-860 tipo naudota eksperimentiniuose tyrimuose. Pasiūlyti lygiagretieji algoritmai analizuoti naudojant lygiagrečiojo algoritmo spartinimo s_p ir efektyvumo e_p koeficientus.

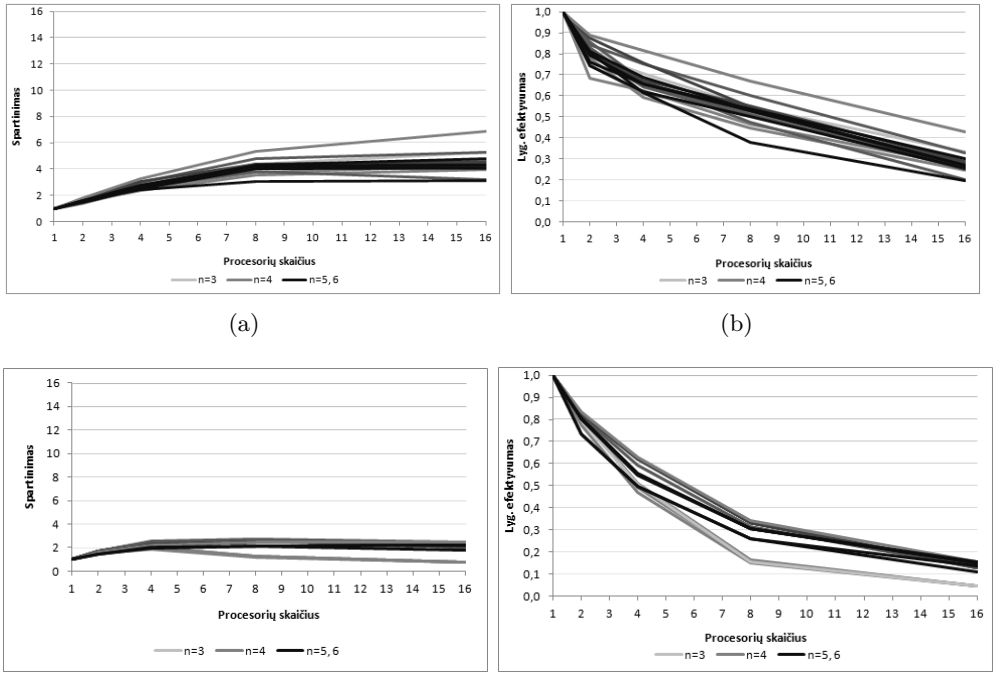
Kai optimizavimo uždavinys yra lengvas, tokio uždavinio sprendimas naudojant lygiagrečiąsias algoritmo versijas yra neefektyvus. Todėl testo uždaviniai, kuriems globaliojo optimumo taško paieška vienu procesoriumi užtruko trumpiau negu 1 s., nenaudoti lygiagrečiųjų algoritmų efektyvumo tyrimuose.

Bendrosios ir paskirstytosios atminties kompiuteriams skirtų algoritmų spartinimai ir lygiagretinimo efektyvumai pateikti 3.1 ir 3.2 paveiksluose. Juose matyti, kad MPI algoritmo lygiagretinimo efektyvumas yra žymiai geresnis, nei OpenMP. Viena iš galimų to priežasčių, kad MPI versija „išeikvoja“ mažiau laiko komunikavimui.

Viršūnių patikrinimas stipriai sumažina funkcijų skaičiavimų kiekį (žr. 3.3 lentelė), bet tuo pačiu pailgina optimizavimo laiką. Naudojant MPI versiją, viršūnių patikrinimo realizacija yra sudėtingesnė nei OpenMP, todėl ši galimybė buvo realizuota tik pastarojoje versijoje. Pastebime, kad OpenMP versijos lygiagretinimo efektyvumas su viršūnių patikrinimu yra geresnis nei be jo.

Tikslo funkcijų skaičiavimų kiekis naudojant abi MPI versijas: su geriausios surastos funkcijos reikšmės apsikeitimu (MPIa) ir be jo (MPI), su skirtingu procesorių skaičiumi yra panašus (žr. 3.5 lentelę). Daliai testo uždavinių, funkcijų skaičiavimų kiekis naudojant MPI su viršūnių apsikeitimu yra apie 5% mažesnis nei be jo, bet tuo pačiu programos vykdymo laikas yra vidutiniškai apie 10% didesnis (žr. 3.6 lentelę). Todėl naudojant MPI su geriausios funkcijos reikšmės apsikeitimu lygiagretinimo efektyvumas yra mažesnis, nei be jo.

Tiek OpenMP tiek ir MPI algoritmo efektyvumas gerėja, kai tikslo funkcijos dimensija didėja, t.y. funkcija sudėtingėja. Lygiagretinimo efektyvumas sudėtingesnėms tikslo funkcijoms mažėja lėčiau, didėjant procesorių skaičiui, nei paprastesnių funkcijų atveju.



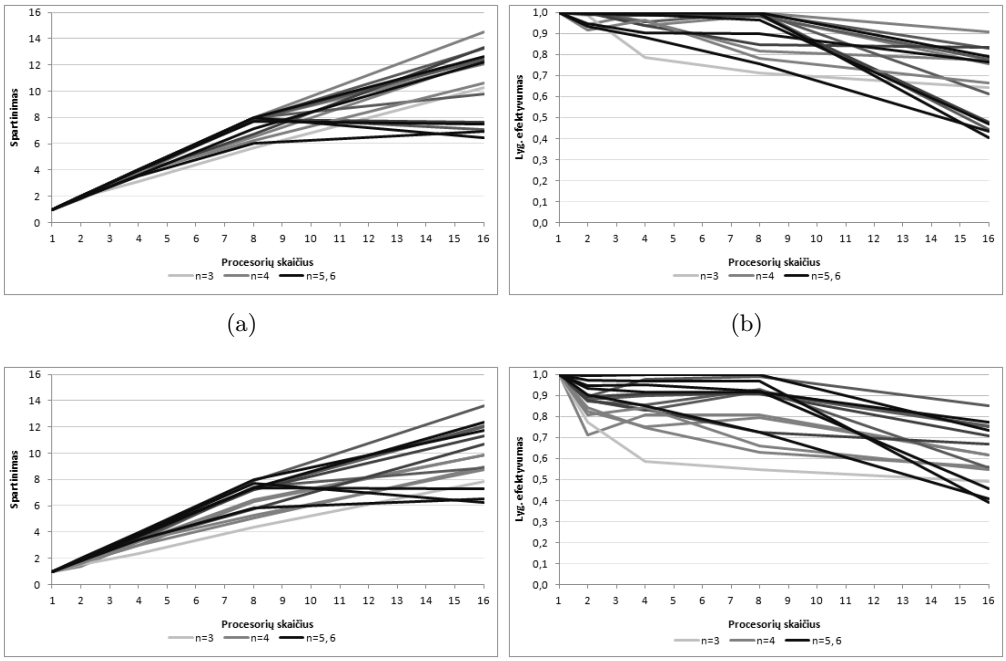
3.1 pav. *Lygiagrečiojo simpleksinio šakų ir rėžių Lipšico optimizavimo algoritmo su OpenMP spartinimas ir lygiagretinimo efektyvumas (su viršūnių patikrinimu – (a), (b); be viršūnių patikrinimo – (c), (d))*

3.6. Paieškos strategijų tyrimas šakų ir rėžių algoritmuose

Paieškos strategijos įtakoja šakų ir rėžių algoritmų rezultatus, todėl tikslinga ištirti tai. Nors eksperimentai buvo atlikti tik su pasiūlytu šakų ir rėžių Lipšico optimizavimo algoritmu, tačiau panašūs rezultatai tikėtini ir kituose šakų ir rėžių algoritmuose.

Pagrindinės paieškos (kandidato (elemento) išrinkimo) strategijos yra:

- geryn – išrenkamas kandidatas, kurio viršutinio rėžio reikšmė yra didžiausia. Kandidatų sąrašas gali būti realizuotas panaudojant krūvos arba prioritetinės eilės duomenų struktūras.



(a) (b)

(c) (d)

3.2 pav. *Lygiagrečiojo simpleksinio šakų ir rėžių Lipšico optimizavimo algoritmo su MPI spartinimas ir lygiagretinimo efektyvumas (be geriausios reikšmės apsikeitimo – (a), (b); su geriausios reikšmės apsikeitimu – (c), (d))*

- gilyn – išrenkamas jauniausias kandidatų aibės $\mathbb{I}$ elementas. FILO pagrindu realizuotas kandidatų sąrašas panaudojant dėklo duomenų struktūrą.
- platyn – išrenkamas seniausias aibės $\mathbb{I}$ elementas. FIFO principu realizuotas kandidatų sąrašas panaudojant eilės duomenų struktūrą.
- pagerinta – paremta euristiniais [14, 69], tikimybiniais [23] ar statistiniais [129] kriterijais. Kandidatų sąrašas gali būti realizuotas panaudojant krūvos arba prioritetinės eilės duomenų struktūras. Statistinio

3.5 lentelė. *Tikslo funkcijų skaičiavimų kiekis optimizuojant lygiagrečiuoju simpleksiniu šakų ir rėžių algoritmu su MPI ir agreguotuoju rėžiu $\varphi^1\mu_2^{2,\infty}$ (Vilko klasteryje)*

Nr.	MPI				MPIa			
	2 p.	**4 p.**	**8 p.**	**16 p.**	**2 p.**	**4 p.**	**8 p.**	**16 p.**
14.	3423480	3423480	3423480	3423480	3423480	3423480	3423480	3423480
16.	3145728	3145728	3145728	3145728	3145728	3145728	3145728	3145728
17.	1363119	1363700	1427885	1425216	1363025	1363002	1363010	1363019
20.	547622	547622	547981	548079	547622	547622	547622	547622
21.	6262233	6262233	6262233	6264548	6262211	6262191	6262224	6262191
22.	167217	167217	167217	167217	167217	167217	167217	167217
23.	536280	536282	536300	541431	536148	536030	536068	535241
24.	988493	988493	988493	988493	988493	988493	988493	988493
25.	543712	543716	543717	628371	543451	542736	542984	539629
26.	1176018	1176020	1177106	1178412	1176134	1176102	1176018	1176006
28.	314025	314195	316543	318247	313579	313702	313478	313189
29.	1916941	1916941	1916941	1916941	1916941	1916941	1916941	1916941
30.	6064924	6064924	6064924	6064924	6064924	6064924	6064924	6064924
31.	5940304	5940304	5940304	5940304	5940034	5939938	5939645	5938389
32.	1488939	1488939	1488939	1488939	1488939	1488939	1488939	1488939
33.	11948747	11948747	11948747	11948747	11948747	11948747	11948747	11948747

išrinkimo strategijos atveju išrenkamas kandidatas, kurio įvertis $\tilde{u}(\mathbf{I})$

$$\tilde{u}(\mathbf{I}) = -\frac{\left(f^* - \frac{1}{n+1}\sum\limits_{v\in V(I)} f(v)\right)^2 - \left(\max\limits_{v\in V(I)} f(v) - \frac{1}{n+1}\sum\limits_{v\in V(I)} f(v)\right)^2}{\min\limits_{v\in V(\mathbf{I})}\left\|v - \frac{1}{n+1}\sum\limits_{v\in V(\mathbf{I})} v\right\|_2}$$

yra didžiausias.

Nuosekliojo simpleksinio šakų ir rėžių algoritmo rezultatai ištirti naudojant funkcijų skaičiavimo kiekio ($f.k.$), vykdymo laiko ($t(s)$), ištirtų posričių (simpleksų) skaičiaus (PS), maksimalaus kandidatų (simpleksų) sąrašo (MKS) dydžio bei santykio $r(f^*)$ kriterijus:

$$r(f^*) = \frac{f.k.(f^*)}{f.k.},$$

3.6 lentelė. *Programos vykdymo laikas (s) optimizuojant lygiagrečiuoju simpleksiniu šakų ir rėžių algoritmu su MPI ir agreguotuoju rėžiu $\varphi^1\mu_2^{2,\infty}$ (Vilko klasteryje)*

Nr.	MPI				MPIa			
	2 p.	**4 p.**	**8 p.**	**16 p.**	**2 p.**	**4 p.**	**8 p.**	**16 p.**
14.	5,04	2,63	1,54	0,82	5,77	3,27	1,93	1,09
16.	4,92	2,12	1,06	0,63	6,52	2,86	1,43	0,94
17.	1,87	1,00	0,47	0,31	2,39	1,31	0,62	0,40
20.	0,67	0,42	0,23	0,12	0,86	0,56	0,30	0,17
21.	20,37	10,32	4,85	3,28	23,68	12,52	5,65	3,43
22.	0,44	0,23	0,11	0,09	0,51	0,26	0,12	0,10
23.	1,49	0,75	0,36	0,24	1,78	0,87	0,43	0,26
24.	2,96	1,48	0,70	0,45	4,04	1,80	0,83	0,53
25.	1,55	0,76	0,37	0,40	1,78	0,88	0,44	0,28
26.	2,52	1,35	0,75	0,38	2,90	1,53	0,88	0,48
28.	0,99	0,47	0,29	0,17	1,13	0,54	0,34	0,21
29.	26,18	13,06	6,66	8,16	27,02	13,55	6,79	8,44
30.	80,69	40,57	20,86	21,36	84,67	42,14	21,80	21,87
31.	87,29	46,13	26,99	23,40	90,24	47,91	28,04	25,02
32.	126,00	63,17	31,58	20,00	125,25	63,24	31,62	21,52
33.	1015,47	531,86	267,25	157,45	1040,17	529,17	263,96	157,17

čia $(f.k.(f^*))$ yra funkcijos skaičiavimų kiekis iki tol, kol buvo surasta globaliojo maksimumo f^* aproksimacija ε tikslumu. Šis santykis visuomet yra tarp nulio ir vieneto bei parodo, kaip greitai optimizavimo procese yra surandamas sprendinys.

Lygiagrečiojo simpleksinio šakų ir rėžių algoritmo efektyvumas, naudojant įvairias paieškos strategijas, buvo matuojamas vertinant spartinimo koeficiento ir lygiagretinimo efektyvumo kriterijus.

3.6.1. Paieškos strategijų įtaka nuosekliesiems šakų ir rėžių algoritmams

Nuosekliosios algoritmo versijos (žr. 3 algoritmą) funkcijų skaičiavimo kiekis $(f.k.)$ bei programos vykdymo laikas $(t(s))$ naudojant įvairias paieškos strategijas pavaizduoti 3.7 lentelėje. Lentelės eilutėse, pažymėtose „vid.“ pateikti vidurkiniai funkcijų skaičiavimo kiekiai bei programos vykdymo laikai skirtingos dimensijos testo uždaviniams. Iš jų matyti, kad $n = 2, 3$ dimensijos testo uždaviniams pagal funkcijų skaičiavimo kiekio kriterijų mažiausiai efektyvi pa-

3.7 lentelė. *Tikslo funkcijų skaičiavimo kiekis ir optimizavimo laikas optimizuojant simpleksiniu šakų ir rėžių algoritmu su agreguotuoju rėžiu* $\varphi^1\psi^2\mu_2^{2,\infty}$ *ir įvairiomis paieškos strategijomis*

	geryn		gilyn		platyn		statistinė	
Nr.	$f.k.$	$t(s)$	$f.k.$	$t(s)$	$f.k.$	$t(s)$	$f.k.$	$t(s)$
1	716	0,002	4513	0,013	720	0,002	707	0,002
2	185	0,001	199	0,000	185	0,001	184	0,001
3	4843	0,014	5312	0,015	4844	0,014	4843	0,015
4	8	0,000	8	0,000	8	0,000	8	0,000
5	39	0,000	39	0,000	53	0,001	39	0,000
6	1241	0,003	5921	0,016	1241	0,003	1241	0,003
7	12195	0,037	12296	0,034	12195	0,034	12202	0,041
8	341	0,001	462	0,001	342	0,001	340	0,001
9	21724	0,068	21760	0,062	21724	0,058	21724	0,066
10	1495	0,005	1606	0,005	1511	0,004	1493	0,005
11	3598	0,010	5195	0,015	3601	0,009	3979	0,012
12	16938	0,052	17001	0,046	16939	0,046	16937	0,053
13	18737	0,062	18781	0,056	18747	0,055	18766	0,066
vid.	**6312**	**0,020**	**7161**	**0,020**	**6316**	**0,018**	**6343**	**0,020**
14	2355490	19,73	2355490	16,90	2355490	17,11	2355490	19,02
15	3784	0,03	6442	0,05	4193	0,03	3797	0,03
16	3145728	27,53	3145728	23,10	3145728	23,15	3145728	27,22
17	1020782	8,58	1059124	7,79	1022743	7,62	1020772	8,50
18	12032	0,09	15025	0,11	12321	0,09	12160	0,10
19	17105	0,14	108820	0,78	17105	0,13	17105	0,13
20	262311	2,09	262340	1,89	262351	1,89	262308	2,07
vid.	**973890**	**8,31**	**993281**	**7,23**	**974276**	**7,15**	**973909**	**8,15**
21	5349848	190,17	5350803	185,14	5349944	185,65	5349782	191,10
22	120144	4,13	120144	4,12	120144	4,00	120144	4,15
23	450084	15,73	443780	15,17	443219	15,02	443588	15,74
24	563835	20,02	563836	19,34	563835	19,20	563835	19,96
25	465955	15,81	445906	15,23	443433	15,27	445309	15,91
26	749518	26,27	750735	25,43	750092	25,35	749851	26,26
27	333568	11,79	333471	11,30	333512	11,45	333470	11,63
28	271614	9,38	275474	9,46	276474	9,47	269485	9,35
vid.	**1038071**	**36,66**	**1035519**	**35,65**	**1035082**	**35,68**	**1034433**	**36,76**
29	1633849	307,36	1633969	316,83	1633837	310,06	1633837	305,74
30	4590448	864,75	4590448	849,56	4590448	856,56	4590448	879,39
31	5192437	976,94	5202842	976,77	5222690	970,65	5189886	960,17
vid.	**3805578**	**716,35**	**3809086**	**714,39**	**3815658**	**712,42**	**3804724**	**715,10**
32	524940	805,94	524940	794,75	524940	779,13	524940	829,04
33	1685793	2451,97	1685793	2462,63	1685793	2464,22	1685793	2578,57
vid.	**1105367**	**1324,75**	**1105367**	**1323,92**	**1105367**	**1318,59**	**1105367**	**1374,24**

3.8 lentelė. *Vidutiniai santykiai $r(f^*)$ optimizuojant simpleksiniu šakų ir rėžių algoritmu su agreguotuoju rėžiu $\varphi^1\psi^2\mu_2^{2,\infty}$ ir įvairiomis paieškos strategijomis*

n	geryn	gilyn	platyn	statistinė
2	0,47	0,47	0,63	0,20
3	0,27	0,26	0,52	0,09
4	0,18	0,14	0,32	0,05
5-6	0,19	0,03	0,21	0,00

ieškos gilyn strategija. Didesnės dimensijos testo uždaviniams $n \geq 4$ vidutinis funkcijų skaičiavimo kiekis naudojant visas tiriamas paieškos strategijas yra labai panašus ir gautieji skirtumai yra nežymūs. Mažiausi optimizavimo laikai gaunami su paieškos gilyn ir platyn strategijomis. Pagrindinė to priežastis yra tai, kad šių strategijų realizavimui naudojamos eilės ir dėklo duomenų struktūros, kurioms naujo elemento (simplekso) įterpimo ir ištrynimo operacijos laikas nepriklauso nuo esamų elementų skaičiaus sąraše. Naudojant paieškos geryn ir statistinę strategijas, kurios realizuotos krūvos duomenų struktūros pagrindu, yra sudaromas prioritetinis sąrašas, todėl naujo elemento įterpimas reikalauja daugiau laiko sąnaudų, ypač kai kandidatų aibės elementų skaičius yra didelis.

Gauti vidutiniai santykiniai dydžiai $r(f^*)$ skirtingos dimensijos testo uždaviniams naudojant įvairias paieškos strategijas pavaizduoti 3.8 lentelėje.

Beveik visiems testo uždaviniams mažiausias dydis $r(f^*)$ gaunamas naudojant statistinę išrinkimo strategiją ir jis yra vidutiniškai daugiau nei du kartus mažesnis, negu naudojant kitas strategijas. Lyginant kitas paieškos strategijas pastebime, kad dvimačiams ir trimačiams ($n = 2, 3$) testo uždaviniams gaunami panašūs santykiai taikant paieškos geryn ir gilyn strategijas, bet $n \geq 4$ testo uždaviniams geresni rezultatai (mažesnis santykis) gaunami naudojant paieškos gilyn strategiją. Blogiausi vidutiniai dydžiai $r(f^*)$ gaunami naudojant paieškos platyn strategiją.

Ištirtų posričių (PS) ir maksimalus kandidatų (simpleksų) skaičius (MKS) naudojat įvairias paieškos strategijas pateikti 3.9 lentelėje. Pastebime, kad $n = 2, 3$ dimensijos testo uždaviniams didžiausias posričių skaičius gaunamas naudojant paieškos gilyn strategiją. Didesnės dimensijos ($n \geq 4$) uždaviniams naudojant visas analizuojamas paieškos strategijas dydis PS yra labai panašus. Gerokai labiau varijuoja naudojant paieškos strategijas maksimalus kandidatų sąrašas. Geriausi rezultatai (mažiausias MKS) gaunamas naudojant paieškos gilyn strategiją ir yra iki 7000 kartų mažesnis, nei naudojant kitas paieškos strategijas. Naudojant paieškos statistinę strategiją MKS yra iki 5 kartų ma-

3.9 lentelė. *Ištirtų posričių skaičius (PS) bei maksimalus kandidatų skaičius (MKS) optimizuojant simpleksiniu šakų ir rėžių algoritmu su agreguotuoju rėžiu $\varphi^1\psi^2\mu_2^{2,\infty}$ ir įvairiomis paieškos strategijomis*

	geryn		gilyn		platyn		statistinė	
Nr.	PS	MKS	PS	MKS	PS	MKS	PS	MKS
1	1412	216	9024	14	1430	179	1412	161
2	366	53	396	12	368	30	366	29
3	9684	1843	10622	12	9686	1147	9684	303
4	14	3	14	3	14	4	14	4
5	74	10	74	5	74	16	76	6
6	2480	382	11840	14	2480	283	2480	204
7	24388	5489	24590	13	24388	3603	24402	580
8	678	105	922	11	680	63	678	33
9	43446	10444	43518	14	43446	8508	43446	636
10	2984	483	3210	13	2994	341	2984	93
11	7192	1371	10388	14	7194	1037	7956	210
12	33872	5698	34000	13	33874	7258	33872	1268
13	37466	8559	37560	15	37466	8192	37530	2638
vid.	**12620**	**2666**	**14320**	**12**	**12623**	**2359**	**12685**	**474**
14	4710974	901316	4710974	24	4710974	689262	4710974	123184
15	7460	761	12878	20	8046	554	7588	102
16	6291450	1543482	6291450	24	6291450	1572864	6291450	368469
17	2041520	378083	2118242	24	2042846	320538	2041538	154576
18	23954	4660	30044	23	24602	3033	24314	2653
19	34204	6321	217634	24	34204	5534	34204	4780
20	524610	111467	524674	21	524652	76043	524610	11060
vid.	**1947739**	**420870**	**1986557**	**23**	**1948111**	**381118**	**1947811**	**94975**
21	10699540	1016395	10701582	39	10699674	1656157	10699540	358063
22	240264	28168	240264	35	240264	27865	240264	5170
23	884808	147320	887536	37	885914	159881	887152	132836
24	1127646	164100	1127648	37	1127646	162612	1127646	144028
25	882342	162790	891788	37	885484	157913	890594	130912
26	1498916	216403	1501446	38	1499432	171915	1499678	16513
27	666916	100877	666918	37	666948	98238	666916	8669
28	538946	57926	550924	39	547820	48453	538946	9126
vid.	**2067422**	**236747**	**2071013**	**37**	**2069148**	**310379**	**2068842**	**100665**
29	3267554	496255	3267818	132	3267554	398547	3267554	107605
30	9180776	998386	9180776	133	9180776	977293	9180776	159233
31	10379652	1049544	10405564	138	10413428	784132	10379652	136655
vid.	**7609327**	**848062**	**7618053**	**134**	**7620586**	**719991**	**7609327**	**134498**
32	1049160	126600	1049160	728	1049160	126720	1049160	69684
33	3370866	249612	3370866	729	3370866	360473	3370866	142415
vid.	**4009784**	**188106**	**4012693**	**729**	**4013537**	**243597**	**4009784**	**106050**

žesnis, nei su paieškos strategija geryn. Didžiausias maksimalus kandidatų sąrašas gaunamas naudojant paieškos platyn strategiją.

3.6.2. Paieškos strategijų įtaka lygiagretiesiems šakų ir rėžių algoritmams

Lygiagrečioji algoritmo versija su MPI naudojant įvairias paieškos strategijas vertinta naudojant tradicinius kriterijus: spartinimo koeficientą bei lygiagretinimo efektyvumą. Vidutiniai spartinimo koeficientai bei vidutiniai efektyvumai pateikti 3.10 lentelėje. Geriausias vidutinis lygiagretinimo efektyvumas su skirtingu procesorių skaičiumi (p) gaunamas naudojant paieškos platyn strategiją. Taikant paieškos geryn ir statistinę strategijas gaunami panašūs lygiagretinimo efektyvumai. Blogiausias efektyvumas gaunamas kuomet paieškos gilyn strategija yra naudojama. Optimizuojant dimensijos $n \geq 5$ uždavinius gaunami vidutiniai efektyvumai su visomis paieškos strategijomis skiriasi nežymiai. Galiausiai nepriklausomai pasirinktai strategijai, efektyvumo koeficientai sudėtingesniems uždaviniams yra geresni, lyginant su paprastesniais.

3.7. Trečiojo skyriaus apibendrinimas

1. Pasiūlyta didesnės dimensijos ($n \geq 3$) atveju rėžį $\mu_2^{1,\infty}$ papildyti Euklidine norma – $\mu_2^{1,2,\infty}$. Naudojant pasiūlytą simpleksinį šakų ir rėžių algoritmą su šiuo rėžiu tikslo funkcijų skaičiavimų kiekis skirtingos dimensijos testo uždaviniams vidutiniškai nuo 21% iki 52% mažesnis lyginant su rezultatais gautais naudojant įprastą rėžį μ_2^2.
2. Ištirta, kad naudojant φ^1 rėžį tikslo funkcijų skaičiavimų kiekis vidutiniškai apie 18% mažesnis nei naudojant rėžį μ_2^1.
3. Pasiūlytas agreguotasis rėžis $\varphi^1\mu_2^{2,\infty}$ su kuriuo vidutiniškai gauname apie 9% mažesnį tikslo funkcijų skaičiavimų kiekį nei rėžio $\mu_2^{1,2,\infty}$ atveju ir apie 25% mažesnį lyginant su rėžiu φ^1.
4. Ištirta, kad rėžis ψ^2 dvimačiu ir trimačiu ($n = 2, 3$) atvejais tikslo funkcijų skaičiavimų kiekį sumažina apie 47%, o ($n \geq 4$) atveju – apie 26% lyginant su μ_2^2 rėžiu.
5. Pasiūlyta agreguotąjį rėžį $\varphi^1\mu_2^{2,\infty}$ papildyti ψ^2 rėžiu. Naudojant patikslintą agreguotąjį rėžį $\varphi^1\psi^2\mu_2^{2,\infty}$ funkcijų skaičiavimų kiekis vidutiniškai apie 20% sumažėja lyginant su agreguotuoju $\varphi^1\mu_2^{2,\infty}$ rėžiu.

3.10 lentelė. *Šakų ir rėžių Lipšico optimizavimo lygiagrečiojo algoritmo su MPI vidutinis spartinimas ir lygiagretinimo efektyvumas su įvairiomis paieškos strategijomis*

	2 p.		4 p.		8 p.		16 p.	
n	$\overline{s_p}$	$\overline{e_p}$	$\overline{s_p}$	$\overline{e_p}$	$\overline{s_p}$	$\overline{e_p}$	$\overline{s_p}$	$\overline{e_p}$
				geryn				
3	1,97	0,98	3,13	0,78	4,45	0,56	8,18	0,51
4	1,95	0,98	3,73	0,93	7,28	0,91	11,58	0,72
5	1,90	0,95	3,67	0,92	6,70	0,84	12,68	0,79
6	1,82	0,91	3,60	0,90	6,88	0,86	13,36	0,83
vid.	**1,91**	**0,96**	**3,53**	**0,88**	**6,33**	**0,79**	**11,45**	**0,72**
				gilyn				
3	1,87	0,94	2,98	0,74	4,57	0,57	4,59	0,29
4	1,91	0,96	3,74	0,94	7,02	0,88	10,34	0,65
5	1,81	0,90	3,53	0,88	6,75	0,84	12,40	0,77
6	1,76	0,88	3,52	0,88	6,60	0,83	13,13	0,82
vid.	**1,84**	**0,92**	**3,44**	**0,86**	**6,23**	**0,78**	**10,11**	**0,63**
				platyn				
3	1,91	0,95	3,53	0,88	6,40	0,80	10,10	0,63
4	1,93	0,97	3,79	0,95	7,24	0,91	10,35	0,65
5	1,81	0,91	3,61	0,90	6,83	0,85	13,25	0,83
6	1,77	0,89	3,53	0,88	6,63	0,83	13,22	0,83
vid.	**1,86**	**0,93**	**3,61**	**0,90**	**6,78**	**0,85**	**11,73**	**0,73**
				statistinė				
3	1,93	0,96	3,18	0,80	4,61	0,58	8,07	0,50
4	1,96	0,98	3,79	0,95	7,48	0,94	11,23	0,70
5	1,92	0,96	3,64	0,91	6,85	0,86	13,13	0,82
6	1,79	0,90	3,58	0,90	6,86	0,86	13,44	0,84
vid.	**1,90**	**0,95**	**3,55**	**0,89**	**6,45**	**0,81**	**11,47**	**0,72**

6. Nustatyta, kad atlikus viršūnių patikrinimą, tikslo funkcijų skaičiavimų kiekis vidutiniškai sumažėja nuo $1,91$ karto dvimačiu iki $62,73$ kartų šešiamačiu atvejais.
7. Gauta, kad pasiūlytasis simpleksinis šakų ir rėžių algoritmas naudojantis agreguotąjį rėžį su viršūnių patikrinimu $\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$, beveik visiems

testo uždaviniams duoda geresnius rezultatus, lyginant su geriausiu apžvelgtu šakų ir rėžių Lipšico optimizavimo algoritmu (GHJ).

8. Pastebėta, kad nepriklausomai nuo lygiagrečiojo algoritmo realizacijos, lygiagretinimo efektyvumas gerėja, o didėjant procesorių skaičiui mažėja lėčiau, kai optimizavimo uždaviniai yra sudėtingesni.

9. Nustatyta, kad naudojant lygiagrečiąją MPI versiją su geriausios surastos funkcijos reikšmės apsikeitimu ir be jo funkcijos skaičiavimo kiekis skyriasi nežymiai.

10. Nustatyta, kad pagal tikslo funkcijų skaičiavimo kiekio kriterijų mažiausiai efektyvi paieškos gilyn strategija, tačiau su ja ir paieškos platyn strategijomis gaunami mažiausi optimizavimo laikai.

11. Ištirta, kad mažiausias maksimalus kandidatų sąrašas gaunamas naudojant paieškos gilyn strategiją ir yra iki 7000 kartų mažesnis, nei naudojant kitas paieškos strategijas. Naudojant statistinę paieškos strategiją MKS yra iki 5 kartų mažesnis, nei su paieškos strategija geryn. Didžiausias maksimalus kandidatų skaičius gaunamas naudojant platyn strategiją.

12. Didžiausias posričių skaičius gaunamas naudojant paieškos gilyn strategiją, nors kai $n \geq 4$ gaunami PS yra labai panašūs su visomis paieškos strategijomis.

13. Nustatyta, kad optimizavimo procese greičiausiai sprendinys surandamas naudojant statistinę paieškos strategiją, nes gaunamas mažiausias santykis $r(f^*)$. Blogiausias santykis $r(f^*)$ gaunamas naudojant paieškos platyn strategiją.

14. Ištirta, kad geriausias lygiagretinimo efektyvumas gaunamas naudojant paieškos platyn strategiją. Taikant geryn ir statistinę paieškos strategijas gaunami panašūs lygiagretinimo efektyvumai. Blogiausias efektyvumas gaunamas kai paieškos gilyn strategija yra naudojama.

Bendrosios išvados

Išsprendus darbe suformuluotus uždavinius, gauti šie rezultatai:

1. Sukonstruoti tikslesni μ_2 tipo Lipšico rėžiai, naudojantys įvairias normas. Pasiūlyti nauji ψ^2 tipo Lipšico rėžiai, pagrįsti daugiamatės apibrėžtinės sferos spinduliu. Pasiūlytas metodas, kuriuo Pijavskij tipo Lipšico rėžiai gaunami sprendžiant tiesinių lygčių sistemas. Sudaryti agreguotieji rėžiai, naudojantys pasiūlytus rėžius su įvairiomis normomis. Skaičiuojamojo eksperimento rezultatai su įvairiais testiniais uždaviniais parodė, kad pasiūlyti rėžiai yra tikslesni už tradicinius Lipšico rėžius, o juos naudojant stipriai sumažėja tikslo funkcijų skaičiavimų kiekis.

2. Sudarytas nuoseklusis simpleksinis šakų ir rėžių algoritmas naudojantis įvairius pasiūlytus rėžius. Jo pagrindu realizuota programinė įranga Lipšico optimizavimo uždaviniams spręsti. Atlikti eksperimentiniai tyrimai parodė pasiūlytojo simpleksinio šakų ir rėžių algoritmo pranašumą lyginant jį su geriausiu apžvelgtu šakų ir rėžių Lipšico optimizavimo algoritmu (GHJ).

3. Sudaryti lygiagretieji simpleksiniai šakų ir rėžių algoritmai bei programinė įranga skirta bendrosios ir paskirstytosios atminties kompiuteriams. Paskirstytosios atminties kompiuteriams skirto lygiagrečiojo algoritmo efektyvumas yra žymiai geresnis, nei bendrosios atminties

kompiuteriams, tačiau pastaroji algoritmo versija svarbi dėl nesudėtingos viršūnių patikrinimo realizacijos, kurios dėka stipriai sumažinamas funkcijų skaičiavimų kiekis bei padidinamas algoritmo lygiagretinimo efektyvumas.

4. Ištirta, kad lygiagrečiojo paskirstytosios atminties kompiuteriams skirto algoritmo funkcijų skaičiavimų kiekis su geriausios surastos funkcijos reikšmės apsikeitimu ir be jo, skyriasi nežymiai. Nepriklausomai nuo lygiagrečiojo algoritmo realizacijos, lygiagretinimo efektyvumas yra didesnis, o didėjant procesorių skaičiui mažėja lėčiau, kai optimizavimo uždaviniai yra sudėtingesni.

5. Pagal funkcijų skaičiavimo kiekio, optimizavimo laiko, ištirtų posričių skaičiaus bei maksimalaus kandidatų sąrašo kriterijus atlikti šakų ir rėžių algoritmų efektyvumo tyrimai parodė, kad nėra vienos konkrečios paieškos strategijos, pagal kurią būtų gaunami geriausi optimizavimo rezultatai pagal visus kriterijus, todėl paieškos strategiją tikslinga pasirinkti priklausomai nuo sprendžiamo optimizavimo uždavinio skaičiavimo resursų.

Literatūros sąrašas

[1] Anderssen, R. S.; Bloomfield, P. Properties of the random search in global optimization. Journal of Optimization Theory and Applications, 16, No. 5-6, 1975, p. 383–398.

[2] Baravykaitė, M. *Lygiagrečiųjų algoritmų šablonų tyrimas ir kūrimas* PhD thesis, VGTU, 2006.

[3] Baravykaitė, M.; Čiegis, R.; Žilinskas, J. Template realization of generalized branch and bound algorithm. Mathematical Modelling and Analysis, 10, No. 3, 2005, p. 217–236.

[4] Baritompa, W. Customizing methods for global optimization - A geometric viewpoint. Journal of Global Optimization, 3, No. 2, 1993, p. 193–212.

[5] Baritompa, W. Accelerations for a Variety of Global Optimization Methods. Journal of Global Optimization, 4, 1994, p. 37–45.

[6] Bartkutė, V. *Pozicinių statistikų taikymas optimalumo analizei ekstremumo paieškos algoritmuose* PhD thesis, MII, 2007.

[7] Bartkutė, M.; Sakalauskas, L. Statistical Inferences for Termination of Markov Type Random Search Algorithms. JOTA, 141, No. 3, 2009, p. 475–493.

[8] Boender, C. G. E.; Kan, A. H. G. R. Bayesian stopping rules for multistart global optimization methods. Mathematical Programming, 37, 1987, p. 59–80.

[9] Breiman, L.; Cutler, A. A deterministic algorithm for global optimization. Mathematical Programming, 1, No. 3, 1993, p. 179–199.

[10] Butz, A. R. Space Filling Curves and Mathematical Programming. Information and Control, 12, 1968, p. 319–330.

[11] Čiegis, R. Lygiagretieji algoritmai. Vadovėlis aukštųjų mokyklų studentams,studijuojantiems matematiką ir informatiką. Technika, 2002.

[12] Čiegis, R.; Henty, D.; Kågström, B.; Žilinskas, J.; redaktoriai Parallel Scientific Computing and Optimization., Vol. 27 of *Springer Optimization and Its Applications* Springer, 2009.

[13] Collins, N. E.; Eglese, R. W.; Golden, B. L. Simulated annealing - an annotated bibliography. American Journal of Mathematics and Management Sciences, 8, No. 3-4, 1988, p. 209–307.

[14] Csendes, T. Generalized subinterval selection criteria for interval global optimization. Numerical Algorithms, 37, No. 1–4, 2004, p. 93–100.

[15] Danilin, Y. M. Estimation of the Efficiency of an Absolute-Minimum-Finding Algorithm. USSR Computational Mathematics and Mathematical Physics, 11, 1971, p. 261–267.

[16] Danilin, Y. M.; Piyavskii, S. A. On an Algorithm for Finding an Absolute Minimum. Theory of Optimal Solutions, , 1967, p. 25–37.

[17] D'Apuzzo, M.; Marino, M.; Migdalas, A.; Pardalos, P. M.; Toraldo, G. Parallel computing in global optimization. // Kontoghiorghes, E. J.; redaktorius, *Handbook of Parallel Computing and Statistics*, p. 225–258. Chapman & Hall / CRC, 2006.

[18] Dixon, L. C. W.; Szego, G. P. Towards Global Optimization., Vol. 1 North Holland, Amsterdam, 1975.

[19] Dixon, L. C. W.; Szego, G. P. Towards Global Optimization., Vol. 2 North Holland, Amsterdam, 1978.

[20] Dorea, C. C. Y. Stopping rules for a random optimization method. SIAM Journal of Control and Optimization, 28, No. 4, 1990, p. 841–850.

[21] Dorea, C. Y. Limiting distribution for random optimization methods. SIAM Journal of Control and Optimization, 24, No. 1, 1986, p. 76–82.

[22] Drezner, Z. A solution to the Weber Location Problem on the Sphere. Journal of Operational Research Society, 36, 1985, p. 333–334.

[23] Dür, M.; Stix, V. Probabilistic subproblem selection in branch-and-bound algorithms. Journal of Computational and Applied Mathematics, 182, No. 1, 2005, p. 67–80.

[24] Erlenkotter, D. A Dual-based Procedure for Uncapacitated Facility Location. Operation Research, , 1978, p. 378–386.

[25] Evtushenko, Y. G. Numerical Optimization Techniques. Springer-Verlad, Berlin, 1985.

[26] Evtushenko, Y. G.; Rat'kin. The Method of Half-Division for Global Optimization of a Function of Many Variables. Technique of Cybernetics, 1, 1988, p. 75–83.

[27] Ferrari, A.; Galperin, E. A. Numerical Experiments with One-Dimensional Adaptive Cubic Algorithm. Computers and Mathematics with Applications, 25, No. 10/11, 1993, p. 47–56.

[28] Ferreira, A.; Pardalos, P. M.; redaktoriai Solving Combinatorial Optimization Problems in Parallel: Methods and Techniques., Vol. 1054 of *Lecture Notes in Computer Science* Springer, 1996.

[29] Fiodorova, I. Search for the global optimum of multiextremal problems. Optimal Decision Theory, 4, 1978, p. 93–100.

[30] Floudas, C.; Akrotirianakis, I.; Caratzoulas, S.; Meyer, C.; Kallrath, J. Global optimization in the 21st century: Advances and challenges. Computers & Chemical Engineering, 29, No. 6, 2005, p. 1185 – 1202. ISSN 0098-1354. *Selected Papers Presented at the 14th European Symposium on Computer Aided Process Engineering.*

[31] Floudas, C. A. Deterministic Global Optimization: Theory, Methods and Applications., Vol. 37 of *Nonconvex Optimization and its Applications* Kluwer Academic Publishers, 2000.

[32] Floudas, C. A.; Gounaris, C. E. A review of recent advances in global optimization. Journal of Global Optimization, 45, No. 1, 2008, p. 3–38.

[33] Floudas, C. A.; Pardalos, P. M. Encyclopedia of Optimization., Vol. 1-6 KAP, 2001.

[34] Floudas, C. A.; Pardalos, P. M.; Adjiman, C. S.; Esposito, W. R.; Gumus, Z. H.; Harding, S. T.; Klepeis, J. L.; Meyer, C. A.; Schweiger, C. A.

Handbook of Test Problems in Local and Global Optimization., Vol. 33 of *Nonconvex Optimization and its Applications* Kluwer Academic Publishers, 1999.

[35] Fogel1994, D. R. An introduction to simulated evolutionary optimization. IEEE Transactions on Neural Networks, 5, No. 1, 1994, p. 3–14.

[36] Forum, M. P. I. MPI: A Message-Passing Interface standard (version 1.1). Technical report, 1995.

[37] Foster, I. Designing and building parallel programs. Addison-Wesley publishing Company, 1994.

[38] Galperin, E. A. The Cubic algorithm. Journal of Mathematical Analysis and Applications, 112, No. 2, 1985, p. 635–640.

[39] Galperin, E. A. Two Alternatives for the Cubic Algorithm. Journal of Mathematical Analysis and Applications, 126, 1987, p. 229–237.

[40] Galperin, E. A. Precision, Complexity, and Computational Schemes of the Cubic Algorithm. Journal of Optimization Theory and Applications, 57, 1988, p. 223–238.

[41] Galperin, E. A. The Fast Cubic Algorithm. Computers and Mathematics with Applications, 25, No. 10/11, 1993, p. 147–160.

[42] Garey, M.; Johnson, D. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York, 1979.

[43] Gendron, B.; Crainic, T. G. Parallel branch-and-bound algorithms: survey and synthesis. Operations Research, 42, No. 6, 1994, p. 1042–1066.

[44] Gergel, V. P. A global optimization algorithm for multivariate function with Lipschitzian first derivatives. Journal of Global Optimization, 10, No. 3, 1997, p. 257–281.

[45] Goldberg, B. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.

[46] Gourdin, E.; Hansen, P.; Jaumard, B. Global Optimization of Multivariate Lipschitz Functions: Survey and Computational Comparison. Les Cahiers du GERAD, (May 1994).

[47] Grotzinger, S. Supports and Concave Envelopes. Mathematical Programming, 31, 1985, p. 339–347.

[48] Hansen, E.; Walster., G. W. Global Optimization Using Interval Analy-

sis. Marcel Dekker, New York, 2 edition, 2003.

[49] Hansen, P.; Jaumard, B. Lipshitz optimization. // Horst, R.; Pardalos, P. M.; redaktoriai, *Handbook of Global Optimization*, Vol. 1, p. 407–493. Kluwer Academic Publishers, 1995.

[50] Hansen, P.; Jaumard, B.; Lu, H. An Analytical Approach to Global Optimization. Mathematical Programming, 52, 1991, p. 227–254.

[51] Hendrix, E. M.; Pinter, J. An application of Lipschitzian global optimization to product design. Journal of Global Optimization, 2, 1991, p. 389–402.

[52] Horst, R. A General Class of Branch-and-Bound Methods in Global Optimization with Some New Approaches for Concave Minimization. Journal of Optimization Theory and Applications, 51, 1986, p. 271–291.

[53] Horst, R.; Pardalos, P. M.; Thoai, N. V. Introduction to Global Optimization. Nonconvex optimization and its application. Kluwer Academic Publishers, 1995.

[54] Horst, R.; Thoai, N. V. Branch-and-bound Methods for Solving Systems of Lipschitzian Equations and Inequalities. Journal of Optimization Theory and Applications, 58, 1988, p. 139–146.

[55] Horst, R.; Tuy, H. On the Convergence of Global Methods in Multiextremal Optimization. Journal of Optimization Theory and Applications, 54, 1987, p. 253–271.

[56] Horst, R.; Tuy, H. Global Optimization: Deterministics Approaches. Springer, Berlin, 1996.

[57] Ingber, L. Very fast simulated re-annealing. Mathematical and Computer Modelling, 12, No. 8, 1989, p. 967–973.

[58] Ingber, L.; Rosen, B. Genetic algorithms and very fast simulated reannealing - a comparison. Mathematical and Computer Modelling, 16, No. 11, 1992, p. 87–100.

[59] Ivanikovas, S. *Lygiagrečių skaičiavimų taikymo daugiamačiams duomenims vizualizuoti problemos* PhD thesis, MII, 2010.

[60] Jakušev, A. *Diferencialinių lygčių ir jų sistemų skaitinio sprendimo algoritmų lygiagretinimo technologijos kūrimas, analizė ir taikymai* PhD thesis, VGTU, 2007.

[61] Jansson, C. Construction of convex lower and concave upper bound

functions. Technical report, TU Hamburg-Harburg, 1998.

[62] Jansson, C. Convex-concave extensions. BIT Numerical Mathematics, 40, 2000, p. 291–313.

[63] Jansson, C.; Knuppel, O. A Global Minimization Method: The Multi-Dimensional Case. Technical report, TU Hamburg-Harburg, 1992.

[64] Jaumard, B.; ant H. Ribault, T. H. An On-line Cone Intersection Algorithm for Global Optimization of Multivariate Lipschitz Functions. Cahiers du GERAD, 95, No. 7, 1995.

[65] Jones, D. R.; Perttunen, C. D.; Stuckman, B. Lipschitzian Optimization Without the Lipschitz Constant. Journal of Optimization Theory and Application, 79, No. 1, 1993, p. 157–181.

[66] Kan, A. H. G. R.; Timmer, G. T. Stochastic global optimization methods - part I: Clustering methods. Mathematical Programming, 39, 1987, p. 27–56.

[67] Kan, A. H. G. R.; Timmer, G. T. Stochastic global optimization methods - part II: Multi level methods. Mathematical Programming, 39, 1987, p. 57–78.

[68] Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. Optimization by simulated annealing. Science, 220, 1983, p. 671–680.

[69] Kreinovich, V.; Csendes, T. Theoretical justification of a heuristic subbox selection criterion for interval global optimization. Central European Journal of Operations Research, 9, No. 3, 2001, p. 255–265.

[70] Kvasov, D. E.; Pizzuti, C.; Sergeyev, Y. D. Local tuning and partition strategies for diagonal GO methods. Numerische Mathematik, 94, No. 1, 2003, p. 93–106.

[71] Kvasov, D. E.; Sergeyev, Y. D. Multidimensional global optimization algorithm based on adaptive diagonal curves. Computational Mathematics and Mathematical Physics, 43, No. 1, 2003, p. 40–56.

[72] Lebedev, L. P.; Vorovich, L. L.; Gladwell, G. Functional analysis. Kluwer Academic Publishers, Dordrecht, Boston, London, 2002.

[73] Levy, A. L.; Gomez, S. The Tunneling method applied to global optimization. // P.T. Boggs, R. B.; Schnabel, R.; redaktoriai, *Numerical Optimization*, p. 213–244. SIAM, 1985.

[74] Madsen, K.; Žilinskas, J. Testing Branch-and-Bound Methods for Global

Optimization. Technical Report IMM-REP-2000-05, Technical University of Denmark, 2000.

[75] Madsen, K.; Žilinskas, J. Parallel branch-and bound attraction based methods for global optimization. // Dzemyda, G.; Šaltenis, V.; Žilinskas, A.; redaktoriai, *Stochastic and Global Optimization*, chapter Nonconvex Optimization and its Applications p. 175–187. Kluwer Academic Publishers, 2002.

[76] Masri, S. F.; Bekey, G. A.; Safford, F. B. A global optimization algorithm using adaptive random search. Applied Mathematics and Computation, 7, No. 4, 1980, p. 353–375.

[77] Mayne, D. Q.; Polak, E. Outer approximation algorithm for nondifferentiable optimization problems. Journal of Optimization Theory and Applications, 42, No. 1, 1984, p. 19–30.

[78] Meewella, C. C.; Mayne, D. Q. An Algorithm for Global Optimization of Lipschitz Continuous Functions. Journal of Optimization Theory and Applications, 57, No. 2, 1988, p. 307–323.

[79] Meewella, C. C.; Mayne, D. Q. An Efficient Domain Partitioning Algorithms for Global Optimization of Rational and Lipschitz Continuous Functions. Journal of Optimization Theory and Applications, 61, No. 2, 1989, p. 247–270.

[80] Michalewicz, Z. Genetic algorithms + data structures = evolution programs. Springer-Verlag, 3 edition, 1992.

[81] Migdalas, A.; Pardalos, P. M.; Storøy, S. Parallel Computing in Optimization., Vol. 7 of *Applied Optimization* Kluwer Academic Publishers, 1997.

[82] Mladineo, R. H. An Algorithm for Finding the Global Maximum of a Multimodal, Multivariate Function. Mathematical Programming, 34, No. 2, 1986, p. 188–200.

[83] Mladineo, R. H. Convergence rates of a global optimization algorithm. Mathematical Programming, 54, No. 1-3, 1992, p. 223–232.

[84] Mockus, J. Bayesian Approach to Global Optimization. Kluwer, 1989.

[85] Neumaier, A.; Shcherbina, O.; Huyer, W.; Vinko, T. A comparison of complete global optimization solvers. Mathematical Programming, 103, No. 2, 2005, p. 335–356.

[86] Niederreiter. Quasi-Monte Carlo methods for global optimization. // Grossmann, W.; Pflug, G.; Vincze, I.; Wertz, W.; redaktoriai, *Proceedings of the 4th Pannonian Symposium on Mathematical Statistics*, p. 251–267. Bad Tatzmannsdorf, Austria, 1983.

[87] Niederreiter, H. Random Number Generation and quasi-Monte Carlo Methods., Vol. 1 of *Applied Mathematics* SIAM, 1992.

[88] Niederreiter, H.; Peart, P. Localization of search in quasi-Monte Carlo methods for global optimization. SIAM Journal of Scientific and Statistical Computing, 7, No. 2, 1986, p. 660–664.

[89] Pardalos, P. M.; redaktorius Parallel Processing of Discrete Problems., Vol. 106 of *IMA Volumes in Mathematics and its Applications* Springer, 1999.

[90] Paulauskas, V.; Račkauskas, A. Funkcinė analizė. Erdvės., Vol. 1 Vaistų žinios, 2007.

[91] Pedoe, D. Circles: A Mathematical View. Washington, DC: Math. Assoc. Amer., 1995.

[92] Petkus, T. *Kompiuterių tinklo panaudojimas interaktyviame optimizavime* PhD thesis, VPU, 2001.

[93] Pinter, J. Extended univariate algorithms for n-dimensional global optimization. Computing, 36, No. 1, 1986, p. 91–103.

[94] Pinter, J. Globally Convergent Methods for n-dimensional Multiextremal Optimization. Optimization, 17, 1986, p. 187–202.

[95] Pinter, J. Branch-and-bound algorithms for solving global optimization problems with Lipschitzian structure. Optimization, 19, No. 1, 1988, p. 101–110.

[96] Pinter, J. Lipschitzian Global Optimization: Some Prospective Applications. // Floudas, C.; Pardalos, P.; redaktoriai, *Recent Advances in Global Optimization*, p. 399–432. Princeton University Press, 1992.

[97] Pinter, J. Continuous global optimization software: A brief review. Optika, 52, 1996, p. 1–8.

[98] Pinter, J. D. Global Optimization in Action (Continuous and Lipschitz Optimization: Algorithms, Implementations and Application., Vol. 6 of *Nonconvex optimization and its application* Kluwer Academic Publishers, 1996.

[99] Piyavskii, S. A. An algorithm for finding the Absolute Minimum of a Function. Theory of Optimal Solutions, 2, 1967, p. 13–24. *in Russian.*

[100] Piyavskii, S. A. An algorithm for finding the absolute extremum of a function. Zh. Vychisl. Mat. mat. Fiz, 12, No. 4, 1972, p. 888–896.

[101] Rademacher, H. Uber partielle und totale Differenzierbarkeit I. Mathematische Annalen, 79, 1919, p. 340–359.

[102] Rastrigin, L. A. Systems of Extremal Control. Nauka, Moscow, 1974.

[103] Rinnoy, A. H. G.; Timmer, G. T. A stochastic approach to global optimization. // Byrd, R. H.; Schnabel, R. B.; redaktoriai, *Numerical Optimization*, p. 631–662. SIAM, 1984.

[104] Šablinskas, R. *Lygiagrečiųjų algoritmų tyrimas paskirstytos atmintieslygiagretiems kompiuteriams.* PhD thesis, VDU, 1999.

[105] Sergeyev, Y. D. An information global optimization algorithm with local tunning. SIAM Journal on Optimization, 5, No. 4, 1995, p. 858–870.

[106] Sergeyev, Y. D. Global one-dimensional optimization using smooth auxiliary functions. Mathematical Programming, 81, No. 1, 1998, p. 127–146.

[107] Sergeyev, Y. D. Multidimensional global optimization using the first derivatives. Computational Mathematics and Mathematical Physics, 39, No. 5, 1999, p. 711–720.

[108] Sergeyev, Y. D.; Kvasov, D. E. Global search based on efficient diagonal partitions and a set of Lipschitz constants. SIAM Journal on Optimization, 16, No. 3, 2006, p. 910–937.

[109] Sergeyev, Y. D.; Kvasov, D. E. Diagonal Global Optimization Methods. FizMatLit, Moscow, 2008. *In Russian.*

[110] Sergeyev, Y. D.; Kvasov, D. E. Lipschitz global optimization and estimates of the Lipschitz constant. // Chaoqun, M.; Lean, Y.; Dabin, Z.; Zhongbao, Z.; redaktoriai, *Global Optimization: Theory, Methods and Applications, I*, p. 518–521. Global Link Publ.: Hong Kong, 2009.

[111] Shubert, B. O. A Sequential Method Seeking the Global Maximum of a Function. SIAM Journal on Numerical Analysis, 9, 1972, p. 379–388.

[112] Storn, R.; Price, K. Diferential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11, 1997, p. 341–359.

[113] Strigul, O. I. Search for a Global Extremum in a Certain Subclass of Functions with the Lipschitz Condition. Cybernetics, 11, 1986, p. 812–819.

[114] Strongin, R. G. Numerical Methods in Multiextremal Problems. Nauka, Moscow, 1978.

[115] Strongin, R. G. Algorithms for Multi-Extremal Mathematical Programming Problems Employing the Set of Joint Space-Filling Curves. Journal of Global Optimization, 2, 1992, p. 357–378.

[116] Strongin, R. G.; Sergeyev, Y. D. Global Optimization with Non-Convex Constraints: Sequential and Parallel Algorithms. KAP, Dordrecht, 2000.

[117] Sukharev, A. Optimal Search of Extremum. Moscow University Press, Moscow, 1975.

[118] Tang, Z. B. Adaptive partitioned random search to global optimization. EEE Transactions on Automatic Control, 39, No. 11, 1994, p. 2235–2244.

[119] Todt, M. J. The computation of Fixed Points and Applications., Vol. 24 of *Lecture Notes in Economics and Mathematical Systems* 1976.

[120] Törn, A. Global optimization as a combination of global and local search. // *Proceedings of Computer Simulation Versus Analytical Solutions for Business and Economic Models*, Gothenburg, 1973. p. 191–206.

[121] Törn, A.; Žilinskas, A. Global Optimization., Vol. 350 of *Lecture Notes in Computer Science* Springer-Verlag, Berlin, 1989.

[122] Wood, G. R. Multidimensional Bisection Applied to Global Optimisation. Computers & Mathematics with Applications, 21, No. 6-7, 1991, p. 161–172.

[123] Wood, G. R. The Bisection Method in Higher Dimensions. Mathematical Programming, 55, 1992, p. 319–337.

[124] Yang, C. M.; Beck, J. L. Generalized Trajectory Methods for Finding Multiple Extrema and Roots of Functions. Journal of Optimization Theory and Applications, 97, No. 1, 1998, p. 211–227.

[125] Zhang, B. P.; Wood, G.; Baritompa, W. Multidimensional Bisection: The Performance and the Context. Journal of Global Optimization, 3, No. 3, 1993, p. 337–358.

[126] Zhigljavsky, A.; Žilinskas, A. Stochastic Global Optimization. Springer, New York, 2008.

[127] Žilinskas, A.; Žilinskas, J. Global optimization based on a statistical model and simplicial partitioning. Computers & Mathematics with Applications, 44, No. 7, 2002, p. 957–967.

[128] Žilinskas, A.; Žilinskas, J. On efficiency of tightening bounds in interval global optimization. Lecture Notes in Computer Science, 3732, 2006, p. 197–205.

[129] Žilinskas, A.; Žilinskas, J. P-algorithm Based on a Simplicial Statistical Model of Multimodal Functions. TOP, įteiktas, 2009.

[130] Žilinskas, J. Optimization of Lipschitzian functions by simplex-based branch and bound. Information Technology and Control, 14, No. 1, 2000, p. 45–50.

[131] Žilinskas, J. Black box global optimization inspired by interval methods. Information Technology and Control, 21, No. 4, 2001, p. 53–60.

[132] Žilinskas, J. *Black Box Global Optimization: Covering methods and their parallelization* PhD thesis, KTU, 2002.

[133] Žilinskas, J. Branch and bound with simplicial partitions for global optimization. Mathematical Modelling and Analysis, , No. 1, 2008, p. 145–159.

[134] Žilinskas, J.; Bogle, I. A survey of methods for the estimation ranges of functions using interval arithmetic. // Törn, A.; Žilinskas, J.; redaktoriai, *Models and Algorithms for Global Optimization*, p. 97–108. Springer, 2007.

[135] Žilinskas, J.; Bogle, I. Global optimization: Interval analysis and balanced interval arithmetic. // *Encyclopedia of Optimization, 2nd edn.*, p. 1346–1350. Springer, 2009.

Autoriaus publikacijų disertacijos tema sąrašas

Straipsniai recenzuojamuose periodiniuose mokslo leidiniuose

[A1] Paulavičius, R.; Žilinskas, J. Analysis of different norms and corresponding Lipschitz constants for global optimization. *Technological and Economic Development of Economy*, ISSN 1392-8619, 12(4), 2006, p. 301–306. [Business Source Complete, Iconda].
http://www.tede.vgtu.lt/upload/ukis_zurn/07_2006_nr4.pdf

[A2] Paulavičius, R.; Žilinskas, J. Analysis of different norms and corresponding Lipschitz constants for global optimization in multidimensio case. *Information Technology and Control*, ISSN 1392-124X, 36(4), 2007, p. 383–387. [ISI Web of Science, Inspec].
http://itc.ktu.lt/itc364/Paulav364.pdf

[A3] Paulavičius, R.; Žilinskas, J. Improved Lipschitz bounds with the first norm for function values over multidimensional simplex. *Mathematical Modelling and Analysis*, ISSN 1392-6292, 13(4), 2008, p. 553–563. [ISI Web of Science, Academic Search Complete, Inspec, Zentralblatt MATH]. doi:10.3846/1392-6292.2008.13.553-563.
http://inga.vgtu.lt/~art/k_m13_fileslist.php?key_m=1317

[A4] Paulavičius, R.; Žilinskas, J. Global optimization using the branch-and-bound algorithm with a combination of Lipschitz bounds over simplices. *Technological and Economic Development of Economy*, ISSN 1392-

8619, 15(2), 2009, p. 310–325. [ISI Web of Science]. doi:10.3846/1392-8619.2009.15.310-325.
http://www.tede.vgtu.lt/en/lt/3/NR/PUB/15688

[A5] Paulavičius, R.; Žilinskas, J. Parallel branch and bound algorithm with combination of Lipschitz bounds over multidimensional simplices for multicore computers. In: R. Čiegis, D. Henty, B. Kågström, J. Žilinskas (Eds.), *Parallel Scientific Computing and Optimization. Vol. 27 of Springer Optimization and Its Applications*, Springer, ISSN 1931-6828, 2009, p. 93–102. [ISI Web of Science (Conference Proceedings Citation Index), SpringerLink, Inspec, Zentralblatt MATH]. doi:10.1007/978-0-387-09707-7_8.
http://www.springerlink.com/content/q135471381615624/

[A6] Paulavičius, R.; Žilinskas, J.; Grothey, A. Investigation of selection strategies in branch and bound algorithm with simplicial partitions and combination of Lipschitz bounds. *Optimization Letters*, ISSN 1862-4472, 4(2), 2010, p. 173–183. [ISI Web of Science, SpringerLink]. doi:10.1007/s11590-009-0156-3.
http://www.springerlink.com/content/9714284487524027/

Straipsniai kituose mokslo leidiniuose

[A7] Paulavičius, R.; Žilinskas, J. Branch and bound with simplicial partitions and combination of Lipschitz bounds for global optimization. In: L. Sakalauskas, G.W. Weber, E.K. Zavadskas (Eds.), *The 20th International Conference EURO Mini Conference Continuous Optimization and Knowledge-Based Technologies (EurOPT-2008), May 20-23, 2008, Neringa, Lithuania*, ISBN 978-9955-28-283-9, 2008, p. 54–58. [ISI Web of Science (Conference Proceedings Citation Index)].
http://www.vgtu.lt/upload/leid_konf/paula_54-58.pdf

[A8] Paulavičius, R. Parallel multidimensional Lipschitz optimization. *Science and Supercomputing in Europe. HPC-Europe Report.* 2008, p. 257–261.

A

Testiniai optimizavimo uždaviniai

1. $n = 2, D = [0,1]^2,\ f(x) = 4x_1 x_2 \sin(4\pi x_2)$

$$f^{'}(x_1) = 4x_2 \sin 4\pi x_2$$
$$f^{'}(x_2) = 4x_1 \sin 4\pi x_2 + 16\pi x_1 x_2 \cos 4\pi x_2$$
$$f^* = 2,51997258,\ x^* = (1,00000000; 0,63492204)$$

2. $n = 2, D = [0,1]^2, f(x) = \sin(2x_1 + 1) + 2\sin(3x_2 + 2)$

$$f^{'}(x_1) = 2\cos(2x_1 + 1)$$
$$f^{'}(x_2) = 6\cos(3x_2 + 2)$$
$$f^* = 2,81859485,\ x^* = (0,28539815; 0,00000000)$$

3. $n = 2, D = [-5,10] \times [0,15], f(x) = -(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 - 10(1 - \frac{1}{8\pi})\cos x_1 - 10$

$$f'(x_1) = -2\left(\frac{5}{\pi} - \frac{2.55}{\pi^2}x_1\right)\left(-\frac{1.275}{\pi^2}x_1^2 + \frac{5}{\pi}x_1 + x_2 - 6\right) - 10(\sin x_1)(\frac{1}{8\pi} - 1)$$
$$f'(x_2) = \frac{2.55}{\pi^2}x_1^2 - \frac{10}{\pi}x_1 - 2x_2 + 12$$
$$f^* = -0.39,\ x^* = (3.14158580, 2.25003810)$$

4. $n = 2, D = [-1,1]^2, f(x) = -\max\left(\sqrt{3}x_1 + x_2, -2x_2, x_2 - \sqrt{3}x_1\right)$

$$f^* = 0.00000000,\ x^* = (0.00000000, 0.00000000)$$

5. $n = 2, D = [0, 10]^2, f(x) = e^{-x_1^2} \sin x_1 - |x_2|$

$$f'(x_1) = e^{-x_1^2} \cos x_1 - 2x_1 e^{-x_1^2} \sin x_1$$
$$f'(x_2) = -signum(x_2)$$
$$f^* = 0.39665295,\ x^* = (0.65327118, 0.00000000)$$

6. $n = 2, D = [-2, 4]^2, f(x) = -2x_1^2 + 1.05x_1^4 - x_2^2 + x_1 x_2 - \frac{1}{6}x_2^2$

$$f'(x_1) = 4.\,2x_1^3 - 4x_1 + x_2$$
$$f'(x_2) = x_1 - \frac{7}{3}x_2$$
$$f^* = 239.696629,\ x^* = (4.00000000, -1.11976261)$$

7. $n = 2, D = [-3, 3] \times [-1.5, 4.5], f(x) = -100\left(x_2 - x_1^2\right)^2 - (x_1 - 1)^2$

$$f'(x_1) = 400x_1\left(x_2 - x_1^2\right) - 2x_1 + 2$$
$$f'(x_2) = 200x_1^2 - 200x_2$$
$$f^* = 0.0000000,\ x^* = (0.99954275, 0.99885688)$$

8. $n = 2, D = [-2.5, 3.5] \times [-1.5, 4.5], f(x) = -(x_1 - 2x_2 - 7)^2 - (2x_1 + x_2 - 5)^2$

$$f'(x_1) = 34 - 10x_1$$
$$f'(x_2) = -10x_2 - 18$$
$$f^* = -0.450000,\ x^* = (3.40000000, -1.49999999)$$

9. $n = 2, D = [-2, 2]^2,\ f(x) = -(1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) \times (30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$

$$\begin{aligned} f'(x_1) = {} & \\ & - 1152x_1^7 + 2016x_1^6x_2 + 5376x_1^6 + 3888x_1^5x_2^2 - 8064x_1^5x_2 - 5712x_1^5 \\ & - 6120x_1^4x_2^3 - 12\,960x_1^4x_2^2 + 840x_1^4x_2 - 6720x_1^4 - 5220x_1^3x_2^4 \\ & + 16\,320x_1^3x_2^3 + 21\,480x_1^3x_2^2 + 30\,720x_1^3x_2 + 9816x_1^3 + 5508x_1^2x_2^5 \\ & + 10\,440x_1^2x_2^4 - 3720x_1^2x_2^3 - 29\,520x_1^2x_2^2 - 17\,352x_1^2x_2 + 3216x_1^2 \\ & + 2916x_1x_2^6 - 7344x_1x_2^5 - 17\,460x_1x_2^4 - 10\,080x_1x_2^3 - 15\,552x_1x_2^2 \\ & - 14\,688x_1x_2 - 2520x_1 - 972x_2^7 - 1944x_2^6 + 1188x_2^5 + 11\,880x_2^4 \\ & + 23\,616x_2^3 + 19\,296x_2^2 + 4680x_2 - 720 \end{aligned}$$

$$
\begin{aligned}
f'(x_2) = \\
&288x_1^7 + 1296x_1^6x_2 - 1344x_1^6 - 3672x_1^5x_2^2 - 5184x_1^5x_2 + 168x_1^5 \\
&- 5220x_1^4x_2^3 + 12\,240x_1^4x_2^2 + 10\,740x_1^4x_2 + 7680x_1^4 + 9180x_1^3x_2^4 \\
&+ 13\,920x_1^3x_2^3 - 3720x_1^3x_2^2 - 19\,680x_1^3x_2 - 5784x_1^3 + 8748x_1^2x_2^5 \\
&- 18\,360x_1^2x_2^4 - 34\,920x_1^2x_2^3 - 15\,120x_1^2x_2^2 - 15\,552x_1^2x_2 - 7344x_1^2 \\
&- 6804x_1x_2^6 - 11\,664x_1x_2^5 + 5940x_1x_2^4 + 47\,520x_1x_2^3 + 70\,848x_1x_2^2 \\
&+ 38\,592x_1x_2 + 4680x_1 - 5832x_2^7 + 4536x_2^6 + 26\,568x_2^5 - 9720x_2^4 \\
&- 57\,384x_2^3 - 36\,864x_2^2 - 6120x_2 - 720 \\
f^* = -3.,\ x^* &= (0.00000000, -1.00045725)
\end{aligned}
$$

10. $n = 2, D = [-1.5, 4] \times [-3, 3], f(x) = -\sin(x_1 + x_2) - (x_1 - x_2)^2 + 1.5x_1 - 2.5x_2 - 1$

$$
\begin{aligned}
f'(x_1) &= 2x_2 - 2x_1 - \cos(x_1 + x_2) + 1.5 \\
f'(x_2) &= 2x_1 - 2x_2 - \cos(x_1 + x_2) - 2.5 \\
f^* &= 1.91322295,\ x^* = (-0.54719386, -1.54720091)
\end{aligned}
$$

11. $n = 2, D = [1, 2]^2, f(x) = -(x_1 - 2)^2 - (x_2 - 1)^2 - \frac{0.04}{-\frac{x_1^2}{4} - x_2^2 + 1} - \frac{(x_1 - 2x_2 + 1)^2}{0.2}$

$$
\begin{aligned}
f'(x_1) &= 20x_2 - 12x_1 - 0.02\frac{x_1}{\left(\frac{1}{4}x_1^2 + x_2^2 - 1\right)^2} - 6 \\
f'(x_2) &= 20x_1 - 42x_2 - 0.08\frac{x_2}{\left(\frac{1}{4}x_1^2 + x_2^2 - 1\right)^2} + 22 \\
f^* &= -0.16904267,\ x^* = (1.79541003, 1.37786415)
\end{aligned}
$$

12. $n = 2, D = [1, 3]^2, f(x) = -0.1\left(12 + x_1^2 + \frac{\left(1 + x_2^2\right)}{x_1^2} + \frac{x_1^2x_2^2 + 100}{x_1^4x_2^4}\right)$

$$
\begin{aligned}
f'(x_1) &= \frac{0.2}{x_1^3}\left(x_2^2 + 1\right) - \frac{0.2}{x_1^3x_2^2} + \frac{0.4}{x_1^5x_2^4}\left(x_1^2x_2^2 + 100\right) - 0.1 \\
f'(x_2) &= \frac{0.4}{x_1^4x_2^5}\left(x_1^2x_2^2 + 100\right) - \frac{0.2}{x_1^2x_2^3} - \frac{0.2}{x_1^2}x_2 \\
f^* &= -1.74,\ x^* = (1.74333181, 2.02987349)
\end{aligned}
$$

13. $n = 2, D = [0.01, 1]^2, f(x) = -\frac{1}{2}\left(x_1^2 + x_2^2\right) + \cos(10\ln(2x_1))\cos(10\ln(3x_2)) - 1$

$$
\begin{aligned}
f'(x_1) &= -x_1 - \frac{10}{x_1}\cos(10\ln 3x_2)\sin(10\ln 2x_1) \\
f'(x_2) &= -x_2 - \frac{10}{x_2}\cos(10\ln 2x_1)\sin(10\ln 3x_2) \\
f^* &= 0.0001,\ x^* = (0.01152703, 0.01440453)
\end{aligned}
$$

14. $n = 3, D = [0,1]^3, f(x) = -100\left(x_3 - \left(\frac{x_1+x_2}{2}\right)^2\right)^2 - (1-x_1)^2 - (1-x_2)^2$

$$f'(x_1) = 200\left(x_3 - \left(\frac{1}{2}x_1 + \frac{1}{2}x_2\right)^2\right)\left(\frac{1}{2}x_1 + \frac{1}{2}x_2\right) - 2x_1 + 2$$

$$f'(x_2) = 200\left(x_3 - \left(\frac{1}{2}x_1 + \frac{1}{2}x_2\right)^2\right)\left(\frac{1}{2}x_1 + \frac{1}{2}x_2\right) - 2x_2 + 2$$

$$f'(x_3) = 200\left(\frac{1}{2}x_1 + \frac{1}{2}x_2\right)^2 - 200x_3$$

$$f^* = 0,\ x^* = (0.997942, 0.997942, 0.997942)$$

15. $n = 3, D = [0,1]^3, f(x) = \sum_{i=1}^{4} c_i \exp\left(-\sum_{i=1}^{3} \alpha_{ij}(x_j - p_{ij})^2\right)$

$$\begin{aligned} f'(x_1) = {} & \\ & -\alpha_{11}c_1 \exp\left(-\alpha_{11}(x_1-p_{11})^2 - \alpha_{12}(x_2-p_{12})^2 - \alpha_{13}(x_3-p_{13})^2\right) \\ & (2x_1 - 2p_{11}) - \alpha_{21}c_2 \\ & \exp\left(-\alpha_{21}(x_1-p_{21})^2 - \alpha_{22}(x_2-p_{22})^2 - \alpha_{23}(x_3-p_{23})^2\right) \\ & (2x_1 - 2p_{21}) - \alpha_{31}c_3 \\ & \exp\left(-\alpha_{31}(x_1-p_{31})^2 - \alpha_{32}(x_2-p_{32})^2 - \alpha_{33}(x_3-p_{33})^2\right) \\ & (2x_1 - 2p_{31}) - \alpha_{41}c_4 \\ & \exp\left(-\alpha_{41}(x_1-p_{41})^2 - \alpha_{42}(x_2-p_{42})^2 - \alpha_{43}(x_3-p_{43})^2\right) \\ & (2x_1 - 2p_{41}) \end{aligned}$$

$$\begin{aligned} f'(x_2) = {} & \\ & -\alpha_{12}c_1 \exp\left(-\alpha_{11}(x_1-p_{11})^2 - \alpha_{12}(x_2-p_{12})^2 - \alpha_{13}(x_3-p_{13})^2\right) \\ & \times (2x_2 - 2p_{12}) - \alpha_{22}c_2 \\ & \times \exp\left(-\alpha_{21}(x_1-p_{21})^2 - \alpha_{22}(x_2-p_{22})^2 - \alpha_{23}(x_3-p_{23})^2\right) \\ & \times (2x_2 - 2p_{22}) - \alpha_{32}c_3 \\ & \times \exp\left(-\alpha_{31}(x_1-p_{31})^2 - \alpha_{32}(x_2-p_{32})^2 - \alpha_{33}(x_3-p_{33})^2\right) \\ & \times (2x_2 - 2p_{32}) - \alpha_{42}c_4 \\ & \times \exp\left(-\alpha_{41}(x_1-p_{41})^2 - \alpha_{42}(x_2-p_{42})^2 - \alpha_{43}(x_3-p_{43})^2\right) \\ & \times (2x_2 - 2p_{42}) \end{aligned}$$

$$
\begin{aligned}
f'(x_3) = & \\
& -\alpha_{13}c_1 \exp\left(-\alpha_{11}(x_1-p_{11})^2 - \alpha_{12}(x_2-p_{12})^2 - \alpha_{13}(x_3-p_{13})^2\right) \\
& \times (2x_3 - 2p_{13}) - \alpha_{23}c_2 \\
& \times \exp\left(-\alpha_{21}(x_1-p_{21})^2 - \alpha_{22}(x_2-p_{22})^2 - \alpha_{23}(x_3-p_{23})^2\right) \\
& \times (2x_3 - 2p_{23}) - \alpha_{33}c_3 \\
& \times \exp\left(-\alpha_{31}(x_1-p_{31})^2 - \alpha_{32}(x_2-p_{32})^2 - \alpha_{33}(x_3-p_{33})^2\right) \\
& \times (2x_3 - 2p_{33}) - \alpha_{43}c_4 \\
& \times \exp\left(-\alpha_{41}(x_1-p_{41})^2 - \alpha_{42}(x_2-p_{42})^2 - \alpha_{43}(x_3-p_{43})^2\right) \\
& \times (2x_3 - 2p_{43}) \\
f^* = 3.8627814,\ & x^* = (0.114540, 0.555784, 0.852538)
\end{aligned}
$$

Čia

i	c_i	α_{i1}	α_{i2}	α_{i3}	p_{i1}	p_{i2}	p_{i3}
1	1.0	3.0	10	30	0.3689	0.1170	0.2673
2	1.2	0.1	10	35	0.4699	0.4387	0.7470
3	3.0	3.0	10	30	0.1091	0.8732	0.5547
4	3.2	1.0	10	35	0.0382	0.5743	0.8828

16. $n = 3, D = [0.01, 1]^3, f(x) = -\frac{1}{3}(x_1^2 + x_2^2 + x_3^2) + \cos(10\ln(2x_1))\cos(10\ln(3x_2)) \times \cos(10\ln(4x_3)) - 1$

$$
\begin{aligned}
f'(x_1) &= -\frac{2}{3}x_1 - \frac{10}{x_1}\cos(10\ln 3x_2)\cos(10\ln 4x_3)\sin(10\ln 2x_1) \\
f'(x_2) &= -\frac{2}{3}x_2 - \frac{10}{x_2}\cos(10\ln 2x_1)\cos(10\ln 4x_3)\sin(10\ln 3x_2) \\
f'(x_3) &= -\frac{2}{3}x_3 - \frac{10}{x_3}\cos(10\ln 2x_1)\cos(10\ln 3x_2)\sin(10\ln 4x_3) \\
f^* = 0,\ & x^* = (0.040555, 0.069074, 0.052778)
\end{aligned}
$$

17. $n = 3, D = [0, 4]^3, f(x) = \sin x_1 \sin x_1x_2 \sin x_1x_2x_3$

$$
\begin{aligned}
f'(x_1) &= \sin x_1x_2 \cos x_1 \sin x_1x_2x_3 + x_2 \cos x_1x_2 \sin x_1 \sin x_1x_2x_3 \\
&\quad + x_2x_3 \sin x_1x_2 \sin x_1 \cos x_1x_2x_3 \\
f'(x_2) &= x_1 \cos x_1x_2 \sin x_1 \sin x_1x_2x_3 + x_1x_3 \sin x_1x_2 \sin x_1 \cos x_1x_2x_3 \\
f'(x_3) &= x_1x_2 \sin x_1x_2 \sin x_1 \cos x_1x_2x_3 \\
f^* = 0.9,\ & x^* = (1.572016, 2.998628, 2.998628)
\end{aligned}
$$

18. $n = 3, D = [-1, 1]^3, f(x) = \left(x_1^2 - 2x_2^2 + x_3^2\right) \sin x_1 \sin x_2 \sin x_3$

$$f'(x_1) = (\cos x_1 \sin x_2 \sin x_3)\left(x_1^2 - 2x_2^2 + x_3^2\right) + 2x_1 \sin x_1 \sin x_2 \sin x_3$$
$$f'(x_2) = (\cos x_2 \sin x_1 \sin x_3)\left(x_1^2 - 2x_2^2 + x_3^2\right) - 4x_2 \sin x_1 \sin x_2 \sin x_3$$
$$f'(x_3) = (\cos x_3 \sin x_1 \sin x_2)\left(x_1^2 - 2x_2^2 + x_3^2\right) + 2x_3 \sin x_1 \sin x_2 \sin x_3$$
$$f^* = 0.51637406,\ x^* = (-1.000000, -0.555968, -1.000000)$$

19. $n = 3, D = [-2, 2]^3, f(x) = (x_1 - 1)(x_1 + 2)(x_2 + 1)(x_2 - 2)x_3^2$

$$f'(x_1) = x_3^2(x_1 - 1)(x_2 + 1)(x_2 - 2) + x_3^2(x_1 + 2)(x_2 + 1)(x_2 - 2)$$
$$f'(x_2) = x_3^2(x_1 - 1)(x_1 + 2)(x_2 + 1) + x_3^2(x_1 - 1)(x_1 + 2)(x_2 - 2)$$
$$f'(x_3) = 2x_3(x_1 - 1)(x_1 + 2)(x_2 + 1)(x_2 - 2)$$
$$f^* = -35.99999997,\ x^* = (-0.500000, -2.000000, -2.000000)$$

20. $n = 3, D = [-3, 3]^3, f(x) = -\sum_{i=1}^{n-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right)$

$$f'(x_1) = 400\left(x_2 - x_1^2\right)x_1 - 2(x_1 - 1)$$
$$f'(x_i) = 400\left(x_{i+1} - x_i^2\right)x_i - 2(x_{i+1} - 1) - 200\left(x_i - x_{i-1}^2\right)$$
$$f'(x_n) = -200\left(x_n - x_{n-1}^2\right)$$
$$f^* = 0,\ x^* = (1, 1, 1)$$

21. $n = 4, D = [-10, 10]^4, f(x) = -\sin^2 3\pi x_1 - \sum_{i=1}^{n-1}(x_i - 1)^2\left(1 + \sin^2 3\pi x_{i+1}\right) - (x_n - 1)^2 \times \left(1 + \sin^2 2\pi x_n\right)$

$$f'(x_1) = -6\sin(3\pi x_1)\cos(3\pi x_1)\pi - 2(x_1 - 1)\left(1 + \sin^2(3\pi x_2)\right)$$
$$f'(x_i) = -6(x_{i-1} - 1)^2 \sin(3\pi x_i)\cos(3\pi x_i)\pi - 2(x_i - 1) \times \left(1 + \sin^2(3\pi x_{i+1})\right)$$
$$f'(x_n) = -6(x_{n-1} - 1)^2 \sin(3\pi x_n)\cos(3\pi x_n)\pi - 1 - \sin^2(2\pi x_n) - 4(x_n - 1)\sin(2\pi x_n)\cos(2\pi x_n)$$
$$f^* = 0,\ x^* = (1, 1, 1, 1)$$

22. $n = 4, D = [-4, 4]^4, f(x) = -\sum_{i=1}^{n-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right)$

$$f'(x_1) = 400\left(x_2 - x_1^2\right)x_1 - 2(x_1 - 1)$$
$$f'(x_i) = 400\left(x_{i+1} - x_i^2\right)x_i - 2(x_{i+1} - 1) - 200\left(x_i - x_{i-1}^2\right)$$
$$f'(x_n) = -200\left(x_n - x_{n-1}^2\right)$$
$$f^* = 0,\ x^* = (1, 1, 1, 1)$$

23. $n = 4, D = [0, 10]^4, f(x) = -\sum_{i=1}^{5} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$

$$f'(x_j) = -\sum_{i=1}^{5} \frac{2(x_j - a_{i,j})}{\left((x-a_i)(x-a_i)^T + c_i\right)^2}$$

$$f^* = 10.1532,\ x^* = (4.00004, 4.00013, 4.00004, 4.00013)$$

Čia

i	a_i	c_i
1	(4.0, 4.0, 4.0, 4.0)	0.1
2	(1.0, 1.0, 1.0, 1.0)	0.2
3	(8.0, 8.0, 8.0, 8.0)	0.2
4	(6.0, 6.0, 6.0, 6.0)	0.4
5	(3.0, 7.0, 3.0, 7.0)	0.4

24. $n = 4, D = [0, 10]^4, f(x) = -\sum_{i=1}^{7} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$

$$f'(x_j) = -\sum_{i=1}^{7} \frac{2(x_j - a_{i,j})}{\left((x-a_i)(x-a_i)^T + c_i\right)^2}$$

$$f^* = 10.4029,\ x^* = (4.00057, 4.00069, 3.99949, 3.99961)$$

Čia

i	a_i	c_i
1	(4.0, 4.0, 4.0, 4.0)	0.1
2	(1.0, 1.0, 1.0, 1.0)	0.2
3	(8.0, 8.0, 8.0, 8.0)	0.2
4	(6.0, 6.0, 6.0, 6.0)	0.4
5	(3.0, 7.0, 3.0, 7.0)	0.4
6	(2.0, 9.0, 2.0, 9.0)	0.6
7	(5.0, 5.0, 3.0, 3.0)	0.6

25. $n = 4, D = [0, 10]^4, f(x) = -\sum_{i=1}^{10} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$

$$f'(x_j) = -\sum_{i=1}^{10} \frac{2(x_j - a_{i,j})}{\left((x-a_i)(x-a_i)^T + c_i\right)^2}$$

$$f^* = 10.5364,\ x^* = (4.00075, 4.00059, 3.99966, 3.99951)$$

Čia

i	a_i	c_i
1	$(4.0, 4.0, 4.0, 4.0)$	0.1
2	$(1.0, 1.0, 1.0, 1.0)$	0.2
3	$(8.0, 8.0, 8.0, 8.0)$	0.2
4	$(6.0, 6.0, 6.0, 6.0)$	0.4
5	$(3.0, 7.0, 3.0, 7.0)$	0.4
6	$(2.0, 9.0, 2.0, 9.0)$	0.6
7	$(5.0, 5.0, 3.0, 3.0)$	0.6
8	$(8.0, 1.0, 8.0, 1.0)$	0.7
9	$(6.0, 2.0, 6.0, 2.0)$	0.5
10	$(7.0, 3.6, 7.0, 3.6)$	0.5

26. $n = 4, D = [-5, 10]^4, f(x) = -\sum_{i=1}^{4} \left(\sum_{j=1}^{i} x_j \right)^2$

$$
\begin{aligned}
f'(x_1) &= -8x_1 - 6x_2 - 4x_3 - 2x_4 \\
f'(x_2) &= -6x_1 - 6x_2 - 4x_3 - 2x_4 \\
f'(x_3) &= -4x_1 - 4x_2 - 4x_3 - 2x_4 \\
f'(x_4) &= -2x_1 - 2x_2 - 2x_3 - 2x_4 \\
f^* &= 0,\ x^* = (1, 1, 1, 1)
\end{aligned}
$$

27. $n = 4, D = [-4, 5]^4, f(x) = -(x_1 + 10x_2)^2 - 5(x_3 - x_4)^2 - (x_2 - 2x_3)^4 - 10(x_1 - x_4)^4$

$$
\begin{aligned}
f'(x_1) &= -2x_1 - 20x_2 - 40(x_1 - x_4)^3 \\
f'(x_2) &= -20x_1 - 200x_2 - 4(x_2 - 2x_3)^3 \\
f'(x_3) &= 10x_4 - 10x_3 + 8(x_2 - 2x_3)^3 \\
f'(x_4) &= 10x_3 - 10x_4 + 40(x_1 - x_4)^3 \\
f^* &= 0,\ x^* = (0, 0, 0, 0)
\end{aligned}
$$

28. $n = 4, D = [-10, 10]^4, f(x) = -\sin^2 3\pi y_1 - \sum_{i=1}^{n-1} (y_i - 1)^2 \left(1 + 10 \sin^2 \pi y_{i+1}\right) - (y_n - 1)^2$

$$
\begin{aligned}
f'(x_1) &= -\left(10 \sin^2 \pi \left(\frac{1}{4}x_2 + \frac{3}{4}\right) + 1\right)(2x_1 - 2) \\
&\quad - \frac{3}{2}\pi \cos 3\pi \left(\frac{1}{4}x_1 + \frac{3}{4}\right) \sin 3\pi \left(\frac{1}{4}x_1 + \frac{3}{4}\right) \\
f'(x_2) &= -\left(10 \sin^2 \pi \left(\frac{1}{4}x_3 + \frac{3}{4}\right) + 1\right)\left(\frac{1}{8}x_2 - \frac{1}{8}\right) \\
&\quad - 5\pi \left(\cos \pi \left(\frac{1}{4}x_2 + \frac{3}{4}\right) \sin \pi \left(\frac{1}{4}x_2 + \frac{3}{4}\right)\right)(x_1 - 1)^2 \\
f'(x_3) &= -\left(10 \sin^2 \pi \left(\frac{1}{4}x_4 + \frac{3}{4}\right) + 1\right)\left(\frac{1}{8}x_3 - \frac{1}{8}\right) \\
&\quad - 5\pi \left(\cos \pi \left(\frac{1}{4}x_3 + \frac{3}{4}\right) \sin \pi \left(\frac{1}{4}x_3 + \frac{3}{4}\right)\right)\left(\frac{1}{4}x_2 - \frac{1}{4}\right)^2 \\
f'(x_4) &= \frac{1}{8} - 5\pi \left(\cos \pi \left(\frac{1}{4}x_4 + \frac{3}{4}\right) \sin \pi \left(\frac{1}{4}x_4 + \frac{3}{4}\right)\right)\left(\frac{1}{4}x_3 - \frac{1}{4}\right)^2 \\
&\quad - \frac{1}{8}x_4 \\
f^* &= 0,\ x^* = (1, 1, 1, 1),\ y_i = 1 + (x_i - 1)/4
\end{aligned}
$$

29. $n = 5, D = [-5, 5]^5, f(x) = -\sin^2 3\pi x_1 - \sum_{i=1}^{n-1} (x_i - 1)^2 \left(1 + \sin^2 3\pi x_{i+1}\right)$

$$
\begin{aligned}
&\quad - (x_n - 1)^2 \left(1 + \sin^2 2\pi x_n\right) \\
f'(x_1) &= -6 \sin(3\pi x_1) \cos(3\pi x_1)\, \pi - 2(x_1 - 1)\left(1 + \sin^2(3\pi x_2)\right) \\
f'(x_i) &= -6(x_{i-1} - 1)^2 \sin(3\pi x_i) \cos(3\pi x_i)\, \pi - 2(x_i - 1) \\
&\quad \times \left(1 + \sin^2(3\pi x_{i+1})\right) \\
f'(x_n) &= -6(x_{n-1} - 1)^2 \sin(3\pi x_n) \cos(3\pi x_n)\, \pi - 1 \\
&\quad - \sin^2(2\pi x_n) - 4(x_n - 1) \sin(2\pi x_n) \cos(2\pi x_n) \\
f^* &= 0,\ x^* = (1, 1, 1, 1, 1)
\end{aligned}
$$

30. $n = 5, D = [-5, 5]^6, f(x) = -\sum_{i=1}^{n-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right)$

$$
\begin{aligned}
f'(x_1) &= 400\left(x_2 - x_1^2\right) x_1 - 2(x_1 - 1) \\
f'(x_i) &= 400\left(x_{i+1} - x_i^2\right) x_i - 2(x_{i+1} - 1) - 200\left(x_i - x_{i-1}^2\right) \\
f'(x_n) &= -200\left(x_n - x_{n-1}^2\right) \\
f^* &= 0,\ x^* = (1, 1, 1, 1, 1)
\end{aligned}
$$

31. $n = 5, D = [-10, 10]^5, f(x) = -\sin^2 3\pi y_1 - \sum_{i=1}^{n-1} (y_i - 1)^2 \left(1 + 10 \sin^2 \pi y_{i+1}\right) - (y_n - 1)^2$

$$
\begin{aligned}
f'(x_1) &= -\left(10 \sin^2 \pi \left(\frac{1}{4} x_2 + \frac{3}{4}\right) + 1\right) (2x_1 - 2) \\
&\quad - \frac{3}{2} \pi \cos 3\pi \left(\frac{1}{4} x_1 + \frac{3}{4}\right) \sin 3\pi \left(\frac{1}{4} x_1 + \frac{3}{4}\right) \\
f'(x_2) &= -\left(10 \sin^2 \pi \left(\frac{1}{4} x_3 + \frac{3}{4}\right) + 1\right) \left(\frac{1}{8} x_2 - \frac{1}{8}\right) \\
&\quad - 5\pi \left(\cos \pi \left(\frac{1}{4} x_2 + \frac{3}{4}\right) \sin \pi \left(\frac{1}{4} x_2 + \frac{3}{4}\right)\right) (x_1 - 1)^2 \\
f'(x_i) &= -\left(10 \sin^2 \pi \left(\frac{1}{4} x_{i+1} + \frac{3}{4}\right) + 1\right) \left(\frac{1}{8} x_i - \frac{1}{8}\right) \\
&\quad - 5\pi \left(\cos \pi \left(\frac{1}{4} x_i + \frac{3}{4}\right) \sin \pi \left(\frac{1}{4} x_i + \frac{3}{4}\right)\right) \left(\frac{1}{4} x_{i-1} - \frac{1}{4}\right) \\
f'(x_n) &= \frac{1}{8} - 5\pi \left(\cos \pi \left(\frac{1}{4} x_n + \frac{3}{4}\right) \sin \pi \left(\frac{1}{4} x_n + \frac{3}{4}\right)\right) \left(\frac{1}{4} x_{n-1} - \frac{1}{4}\right)^2 \\
&\quad - \frac{1}{8} x_n \\
f^* &= 0, \; x^* = (1, 1, 1, 1, 1), \; y_i = 1 + (x_i - 1)/4
\end{aligned}
$$

32. $n = 6, D = [-5, 5]^6, f(x) = -\sin^2 3\pi x_1 - \sum_{i=1}^{n-1} (x_i - 1)^2 \left(1 + \sin^2 3\pi x_{i+1}\right)$

$$
\begin{aligned}
&\quad - (x_n - 1)^2 \left(1 + \sin^2 2\pi x_n\right) \\
f'(x_1) &= -6 \sin(3\pi x_1) \cos(3\pi x_1) \pi - 2(x_1 - 1) \left(1 + \sin^2(3\pi x_2)\right) \\
f'(x_i) &= -6 (x_{i-1} - 1)^2 \sin(3\pi x_i) \cos(3\pi x_i) \pi - 2(x_i - 1) \\
&\quad \times \left(1 + \sin^2(3\pi x_{i+1})\right) \\
f'(x_n) &= -6 (x_{n-1} - 1)^2 \sin(3\pi x_n) \cos(3\pi x_n) \pi - 1 \\
&\quad - \sin^2(2\pi x_n) - 4(x_n - 1) \sin(2\pi x_n) \cos(2\pi x_n) \\
f^* &= 0, \; x^* = (1, 1, 1, 1, 1, 1)
\end{aligned}
$$

33. $n = 6, D = [-6, 6]^6, f(x) = -\sum_{i=1}^{n-1} \left(100 \left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right)$

$$
\begin{aligned}
f'(x_1) &= 400 \left(x_2 - x_1^2\right) x_1 - 2(x_1 - 1) \\
f'(x_i) &= 400 \left(x_{i+1} - x_i^2\right) x_i - 2(x_{i+1} - 1) - 200 \left(x_i - x_{i-1}^2\right) \\
f'(x_n) &= -200 \left(x_n - x_{n-1}^2\right) \\
f^* &= 0, \; x^* = (1, 1, 1, 1, 1, 1)
\end{aligned}
$$

Sąvokų žodynas

A

adaptyvioji atsitiktinė paieška – adaptive random search
agreguotasis rėžis – aggregate bound
algoritmas – algorithm
apatinis rėžis – lower bound
apibendrintas nusileidimo metodas – generalized descent method
apibrėžtinė sfera – circumscribed sphere
atkaitinimo modeliavimo metodas – simulated annealing
atsitiktinis dydis – random variable
atsitiktinės paieškos metodas – random search method
atstumo funkcija – distance function

B

baigties nustatymas – termination detection
Bajeso optimizavimas – Bayesian optimization
baudų metodas – penalty method
bendrosios atminties kompiuteris – shared memory computer

D

dalijimo taisyklė – branching rule
dalinė išvestinė – partial derivative
determinantinė forma – determinant form

dinaminis užduočių paskirstymas – dynamic load balancing
dėklas – stack
duomenų struktūra (tipas) – data type

E

ekstremumas – extremum
evoliucinės strategijos – evolution strategy

F

fazė – phase
funkcijos skaičiavimų kiekis – number of function evaluations

G

genetinis algoritmas – genetic algorithm
globalusis maksimumas – global maximum
globalusis minimumas – global minimum
globalusis optimizavimas – global optimization
gradientas – gradient
grupavimo metodas – clustering method

I

iškilas apvalkalas – convex envelope
iškiloji aibė – convex set
išrinkimo taisyklė – selection rule
iteracijų skaičius – number of iterations
iteracinis procesas – iterative process
išvestinė – derivative

J

juodoji dėžė – black box

K

kartotinių nusileidimų metodas – multi start method
klasteris – cluster
kompaktinė aibė (kompaktas) – compact set (compactum)
krūva – heap
kvadratinė lygtis – quadratic equation
kvadratinio priskyrimo uždavinys – quadratic assignment problem
Kvazi Monte-Karlo metodas – Quasi-Monte Carlo method

L

Lebego matas – Lebesgue measure

leistinųjų krypčių metodas – trajectory method
leistinoji sritis – feasible region
Lipšico funkcija – Lipschitz function
Lipšico konstanta – Lipschitz constant
Lipšico optimizavimas – Lipschitz optimization
Lipšico rėžis – Lipschitz bound
Lipšico sąlyga – Lipschitz condition
lokalioji paieška – local search
LP seka – low-discrepancy point set
lygiagretieji algoritmai – parallel algorithms
lygiagretieji kompiuteriai – parallel computers
lygiagretinimo efektymas – efficiency
lygio aibė – level set

M

maksimalus kandidatų sąrašas – maximal size of candidate list
maksimizavimas – maximization
maksimumas – maximum
minimizavimas – minimization
minimumas – minimum

N

neadaptyvi atsitiktinė paieška – nonadaptive random search
netiesinė funkcija – nonlinear function
netiesioginis metodas – indirect method
norma – norm
NP uždavinys – nondeterministic polynomial time problem

O

optimizavimo uždavinys – optimization problem
optimumas – optimum

P

padengimo metodas – covering method
pagerinta paieškos strategija – improved selection strategy
paieška tinkleliu – grid search
paieškos geryn strategija – best first selection strategy
paieškos gilyn strategija – depth first selection strategy
paieškos platyn strategija – breadth first selection strategy

paskirstytosios atminties kompiuteris – distributed memory computer
Peano kreivė – space-filling curve
pirmas įeina, paskutinis išeina – first in, last out (FILO)
pirmas įeina, pirmas išeina – first in, first out (FIFO)
posričių skaičius – number of partitions
prioritetinė eilė – priority queue
procesorių panaudojimo koeficientas – processor utilization
pseudo efektyvumo koeficientas – pseudo efficiency

Š

šablonas – template
šakų ir rėžių metodas – branch and bound method
šeimininkas-darbininkai – master-slave

S

simpleksas – simplex
simpleksiniai posričiai – simplicial partitions
skirstinys – distribution
spartinimo koeficientas – speedup
stačiakampiai posričiai – rectangular partitions
standartinė pranešimų persiuntimo biblioteka – Message Passing Interface (MPI)
statinis užduočių paskirstymas – static load balancing
sustojimo sąlyga – stopping condition

T

tiesinė lygtis – linear equation
tiesinis programavimas – linear programming
tiesioginės paieškos metodas – direct search method
tikslo funkcija – objective function
tolydžioji funkcija – continuous function
tolydusis kintamasis – continuous variable
tolygumo matas – uniformity measure
tolygusis skirstinys – uniform distribution
tuneliavimas – tunneling

U

uždaroji aibė – closed set

V

vidurinių reikšmių teorema – mean value theorem
viršūnių aibė – vertex set
viršūnių patikrinimas – vertex verification
viršūnių trianguliacija – vertex triangulation
viršutinio rėžio funkcija – upper bounding function
viršutinis rėžis – upper bound

Dalykinė rodyklė

Remigijus PAULAVIČIUS

GLOBALUSIS OPTIMIZAVIMAS SU SIMPLEKSINIAIS POSRIČIAIS
Daktaro disertacija
Fiziniai mokslai (P 000)
Informatika (09 P)
Informatika, sistemų teorija (P 175)

Remigijus PAULAVIČIUS

GLOBAL OPTIMIZATION WITH SIMPLICIAL PARTITIONS
Doctoral Dissertation
Physical Sciences (P 000)
Informatics (09 P)
Informatics, Systems Theory (P 175)

2010 08 17, 8,5 sp. l. Tiražas 20 egz.
Parengė spaudai ir išleido Matematikos ir informatikos institutas
Akademijos g. 4, LT-08663 Vilnius.
Interneto svetainė: http://www.mii.lt
Spausdino „Kauno technologijos universiteto spaustuvė“
Studentų g. 54, LT-51424 Kaunas