 Doi: 10.21277/jmd.v1i45.44

GLOBALIOS OPTIMIZACIJOS ALGORITMAS, NAUDOJANTIS LOKALŲ LIPŠICO KONSTANTOS ĮVERTĮ

Albertas Gimbutas
Vilniaus universiteto Matematikos ir informatikos institutas

1. Įvadas

Nagrinėjama neiškiliųjų Lipšico funkcijų globalioji minimizacija apibrėžiama kaip

$$\min_{\boldsymbol{x}\in A} f(\boldsymbol{x}),\ \ \boldsymbol{x}\in A\subset\mathbb{R}^d, \tag{1}$$

$$|f(\boldsymbol{x})-f(\boldsymbol{y})|\le L\|\boldsymbol{x}-\boldsymbol{y}\|,\ \ \boldsymbol{x},\boldsymbol{y}\in A, \tag{2}$$

čia leistinoji sritis yra hiperstačiakampis.

Šiuo darbu siekiame parodyti, kad globalios optimizacijos algoritmai, naudojantys lokalų Lipšico konstantos įvertį, yra perspektyvūs. Siūlomas deterministinis optimizavimo algoritmas, susidedantis iš dviejų lygmenų optimizavimo uždavinių. Jis artimas vienmačiam Pijavskio ir Shuberto [9, 10] ir daugiamačiam DIRECT [1, 4] optimizavimo algoritmams. Leistinoji sritis yra dalijama į posričius, kuriems apibrėžiamas tikslo funkcijos apatinis rėžis; posričiai rekursyviai dalinami pasitelkiant DIRECT algoritme pirmą kartą pristatytą iškilojo kontūro strategiją. Siūlomame algoritme naudojamas lokalus Lipšico konstantos įvertis. Tikimasi, kad jo naudojimas gali pagerinti algoritmo konvergavimo savybes, nes algoritmui suteikiama daugiau informacijos apie funkcijos formą nei kitiems aibę Lipšico konstantų naudojantiems algoritmams, tokiems kaip DIRECT [1, 4], DIRECT*l* [2], DISIMPL-V [7, 8] ir Gb-DISIMPL-V [6].

Siūlomą algoritmą sudaro vidinis ir išorinis optimizavimo lygmenys. Išoriniame lygmenyje optimizuojama tikslo funkcija $f(\cdot)$, kurios įvertinimai laikomi brangiais. Išoriniame lygmenyje leistinoji sritis skaidoma simpleksais, atsižvelgiant į vidinio lygmens optimizavimo uždavinio rezultatą. Vidiniame lygmenyje sprendžiamas duotajame simplekse mažiausios galimos funkcijos reikšmės radimo uždavinys. Šiam lygmeniui pasiūlytas efektyvus optimizavimo algoritmas, leidžiantis vidinio lygmens optimizavimo uždavinį priimtinu tikslumu išspręsti per santykinai trumpą laiką.

Antrame darbo skyriuje pristatomas pasiūlytasis dviejų lygmenų optimizavimo algoritmas. Trečiame skyriuje pateikiami eksperimentinio tyrimo rezultatai. Išvados pateikiamos ketvirtame skyriuje.

2. Pasiūlytasis algoritmas

2.1. Lokalus Lipšico konstantos įvertis

Algoritmu sprendžiamas pagalbinis optimizavimo uždavinys, siekiant įvertinti apatinį tikslo funkcijos reikšmių rėžį duotajame simplekse $\mathbb{S}$. Pagalbinis optimizavimo uždavinys sudaromas naudojant euristinį Lipšico konstantos įvertį $\acute{L}$, kurio nustatymas simpleksui $\mathbb{S}$ aprašomas šiame skyrelyje.

Pirmiausia simpleksui $\mathbb{S}$ nustatomas apatinis tikrosios Lipšico konstantos jame L rėžis:

$$\hat{L}=\max\left\{\frac{|f(\boldsymbol{v}_i)-f(\boldsymbol{v}_j)|}{\|\boldsymbol{v}_i-\boldsymbol{v}_j\|}:\boldsymbol{v}_i,\boldsymbol{v}_j\in\mathbb{S}_{vertices},\boldsymbol{v}_i\ne\boldsymbol{v}_j\right\}. \tag{3}$$

Kitaip sakant, viršūnių porai $\boldsymbol{v}_i, \boldsymbol{v}_j \in \mathbb{S}_{vertices}$ nustatomas Lipšico konstantos L apatinis rėžis:

$\hat{L}_{ij}=\frac{|f(\boldsymbol{v}_i)-f(\boldsymbol{v}_j)|}{\|\boldsymbol{v}_i-\boldsymbol{v}_j\|}\le L$. Taigi perrinkus visas poras gaunama, kad:

$$\hat{L}=\max\{\hat{L}_{ij}:i,j=1,\dots,d+1,i\ne j\},\hat{L}\le L. \tag{4}$$

Tada mažiausią funkcijos reikšmę turinčioje viršūnėje $v_1 \in \mathbb{S}$ apskaičiuojamas simplekso gradientas $D(f:\mathbb{S})$ (žr. [5, 113 p.]) ir jo euklidinė norma $G(f:\mathbb{S})$:

$$V(\mathbb{S})=(\boldsymbol{v}_2-\boldsymbol{v}_1,\boldsymbol{v}_3-\boldsymbol{v}_1,\dots,\boldsymbol{v}_{d+1}-\boldsymbol{v}_1), \tag{5}$$

$$\delta(f:\mathbb{S})=\big(f(\boldsymbol{v}_2)-f(\boldsymbol{v}_1),f(\boldsymbol{v}_3)-$$

$$-f(\boldsymbol{v}_1),\dots,f(\boldsymbol{v}_{d+1})-f(\boldsymbol{v}_1)\big)^T, \tag{6}$$

$$D(f:\mathbb{S})=V(\mathbb{S})^{-T}\delta(f:\mathbb{S}), \tag{7}$$

$$G(f:\mathbb{S})=\|D(f:\mathbb{S})\|. \tag{8}$$

Sudaromas simplekso $\mathbb{S}$ Lipšico konstantos įvertis:

$$\tilde{L}_{\mathbb{S}}=\max\{\hat{L},G(f:\mathbb{S})\}. \tag{9}$$

Toliau kiekvienam simpleksui $\mathbb{S}$ randama kaimyninių simpleksų aibė $\boldsymbol{K}(\mathbb{S})$. Kaimyniniu laikomas simpleksas, kurio ne daugiau kaip dvi viršūnės skiriasi nuo $\mathbb{S}$.

Euristiniu lokaliu Lipšico konstantos įverčiu simplekso $\mathbb{S}$ aplinkoje laikomas skaičius:

$$\acute{L} = \max\{\tilde{L}_{\mathbb{K}}: \mathbb{K} \in \boldsymbol{K}(\mathbb{S})\}. \quad (10)$$

2.2. Vidinis optimizavimo uždavinys

Pagalbinis optimizavimo uždavinys skirtas įvertinti duotojo simplekso $\mathbb{S}$ apatinį tikslo funkcijos reikšmių rėžį su bet kokiu duotuoju Lipšico konstantos viršutiniu įverčiu $\bar{L}$. Funkcijos $f(\cdot)$ reikšmės, įgyjamos simplekse $\mathbb{S}$, yra aprėžtos iš apačios šia funkcija:

$$g(\boldsymbol{x}) = max_{\boldsymbol{v}\in\mathbb{S}_{vertices}}\{f(\boldsymbol{v}) - \bar{L}\|\boldsymbol{v}-\boldsymbol{x}\|\}, \boldsymbol{x} \in \mathbb{S}. \quad (11)$$

Ši funkcija atitinka paviršių, apibrėžtą kūgiais, kurių viršūnės sutampa su simplekso viršūnėmis. Šių kūgių sankirtos minimumo taškas atitinka minimalią galimą funkcijos reikšmę simplekse $\mathbb{S}$:

$$M_{\mathbb{S}} = min_{\boldsymbol{x}\in\mathbb{S}}\, g(x). \quad (12)$$

Siekiant skaitiškai išspręsti uždavinį (12), naudotas vidinio lygmens optimizavimo algoritmas. Jo idėja yra skaidyti pradinį simpleksą smulkesniais simpleksais $\mathbb{S}^i$; kiekviename iš jų $g(\cdot)$ aprėžti iš apačios; pagal gautą rėžį pasirinkti geriausią iš jų ir jį vėl analogiškai dalinti. Norint simplekse $\mathbb{S}^i$ atlikti $g(\cdot)$ aprėžimą, naudojama formulė:

$$\check{M}_{\mathbb{S}^i} = \max\{g(l_1^i) - \bar{L}\mathbb{S}^i_{diameter}, g(l_2^i - \bar{L}\mathbb{S}^i_{diameter}\}, \quad (13)$$

čia $l_1^i, l_2^i \in \mathbb{S}^i_{vertices}$ yra simplekso $\mathbb{S}^i$ ilgiausios kraštinės viršūnės, o $\mathbb{S}^i_{diameter}$ – ilgiausios kraštinės ilgis. Šis $\check{M}_{\mathbb{S}^i}$ įvertis nėra visiškai tikslus, bet reikalauja nedaug skaičiavimo operacijų.

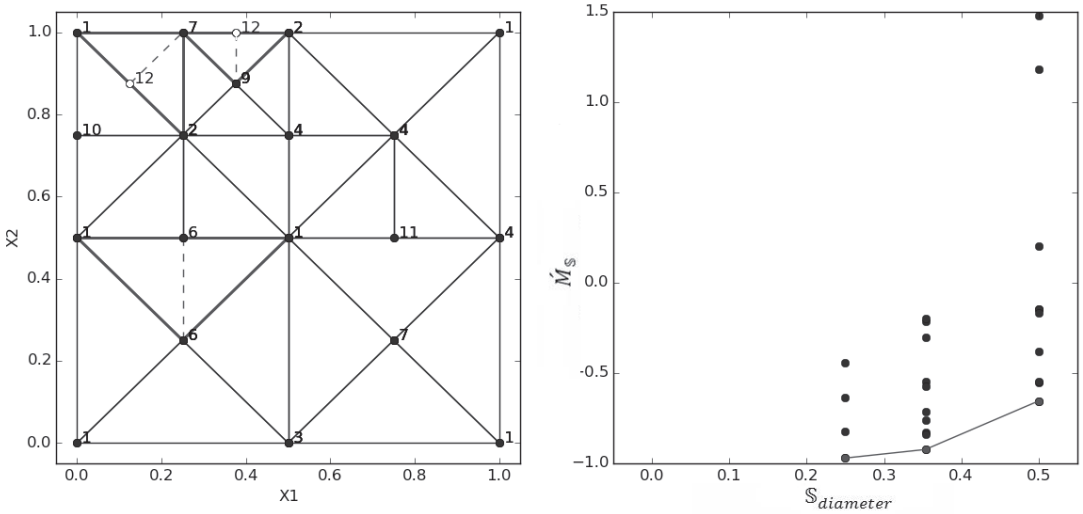
Optimizavimo uždaviniui (12) spręsti pasiūlyta procedūra VIDINIS, įvertinanti šio uždavinio savybes: a) leistinoji sritis yra simplekso formos, b) $g(\cdot)$ funkcijos įvertinimai yra santykinai pigūs.

procedure VIDINIS ($\mathbb{S}$, $\bar{L}$):

1. $\boldsymbol{S} = \{\mathbb{S}\}$, $\acute{M} = \min\{g(\boldsymbol{v}) : \boldsymbol{v} \in \mathbb{S}_{vertices}\}$.
2. $\forall\, \mathbb{S}^i \in \boldsymbol{S}$ rasti $\check{M}_{\mathbb{S}^i}$ pagal formulę (13).
3. Parinkti simpleksą $\mathbb{S}^{best}$ su mažiausia $\check{M}_{\mathbb{S}^i}$ reikšme.
4. Padalinti $\mathbb{S}^{best}$ į $\mathbb{S}^{new,1}$ ir $\mathbb{S}^{new,2}$, atliekant $g(\cdot)$ įvertinimą ilgiausios kraštinės vidurio taške $\boldsymbol{v}_{new}$.
5. Atnaujinti $\acute{M} = \min\{\acute{M}, g(\boldsymbol{v}_{new})\}$.
6. $\boldsymbol{S} = \boldsymbol{S} \setminus \mathbb{S}^{best} \cup \{\mathbb{S}^{new,1}, \mathbb{S}^{new,2}\}$.
7. Jeigu pasiektas maksimalus iteracijų skaičius, einama į 8 žingsnį; jeigu nepasiektas, einama į 2 žingsnį.
8. **return** $\acute{M}$.

Šiame darbe naudotas maksimalus iteracijų skaičius vidiniam optimizavimo uždaviniui išspręsti – 10.

2.3. Posričių išrinkimas remiantis iškilojo kontūro sudarymu



1 pav. Perspektyvių simpleksų parinkimas nustatant iškiląjį kontūrą.
Kairėje pavaizduotas leistinosios srities suskaidymas simpleksais, ties taškais nurodyti numeriai iteracijų, kurių metu jie buvo sugeneruoti. Dešinėje kiekvienas simpleksas vaizduojamas kaip taškas koordinačių sistemoje, kurios viena ašis yra simplekso diametras, o kita – $\acute{M}_{\mathbb{S}}$ reikšmė. Vaizduojama GKLS antros funkcijų klasės 1-os funkcijos 12-ta iteracija

Kiekvienam simpleksui turint apskaičiuotą $\acute{M}_{\mathbb{S}}$ reikšmę, visus simpleksus galima išdėstyti plokštumoje, kaip parodyta 1 paveiksle, siekiant kitu žingsniu atrinkti simpleksus algoritmo dalinimui. Ši idėja buvo išpopuliarinta DIRECT algoritmu, tačiau naudojant kitas ašių prasmes. Mūsų variante abscisių ašyje atidėtas simplekso diametras suprantamas kaip ilgiausia simplekso kraštinė. Ordinačių ašyje atidėtas $\acute{M}_{\mathbb{S}}$ įvertis. Gautiesiems taškams randamas iškilasis kontūras palei apatinį dešinįjį kraštą 1 paveiksle dešinėje pavaizduotas kaip laužtė. Simpleksai, priklausantys šiai laužtei, laikomi perspektyviais, nes jie turi geriausią $\acute{M}_{\mathbb{S}}$ įvertį savo dydžio kategorijoje. Jeigu tas pats taškas ant iškilojo kontūro žymi kelis simpleksus, tada jie visi laikomi perspektyviais.

2.4. Siūlomas algoritmas

Optimizavimas pasiūlytame algoritme prasideda leistinąją sritį A padalinant į nesusiliejančių simpleksų aibę, panaudojus standartinį kombinatorinių viršūnių metodą. Šis metodas hiperkubą padalina į $n!$ simpleksų [6, 12 p.]. Pateikiame hiperkubo kombinatorinių viršūnių trianguliavimo metodo pseudokodą, kuriame v_{ij} žymi i-tosios simplekso viršūnės j-tąją koordinatę, o A_{ja}, A_{jb} – leistinosios srities j-tojo kintamojo apatinį ir viršutinį rėžius:

procedure TRIANGULIUOK (A):
1. Inicializuojama pradinė simpleksų aibė $\boldsymbol{S} = \emptyset$.
2. **for** t = vienas iš visų galimų {1,...,d} kėlinių **do**
3. **for** $j = 1,...,$d **do**
4. $v_{1j} = A_{ja}$
5. **end for**
6. **for** $i = 1,...,d$ **do**
7. **for** $j = 1,...,d$ **do** S
8. $v_{(i+1)j} = v_{ij}$ l’
9. **end for**
10. $v_{(i+1)t_i} = A_{t_i b}$
11. **end for**
12. $\boldsymbol{S} = \boldsymbol{S} \cup \{v\}$
13. **end for**
14. **return** $\boldsymbol{S}$

Padengus leistinąją sritį simpleksais, atliekami tikslo funkcijos įvertinimai simpleksų viršūnėse, išsisaugant funkcijų reikšmes ir nedarant pakartotinių funkcijos įvertinimų tuose pačiuose taškuose. Tolesni pasiūlytojo algoritmo žingsniai pateikiami pseudokode:

procedure PASIŪLYTASISALGORITMAS ($f(\cdot)$):
1. Kombinatoriškai suskaidyti leistinąją sritį A į simpleksų aibę $\boldsymbol{S}$ = TRIANGULIUOK(A). Atlikti funkcijų įvertinimus simpleksų viršūnėse, išsaugant taškus sekoje $\{\boldsymbol{x}_i\}$, o funkcijos reikšmes – sekoje $\{y_i\}$.
2. $\forall \mathbb{S} \in \boldsymbol{S}$ rasti:
   2.1. $\hat{L}_{\mathbb{S}}$ pagal formulę (3).
   2.2. Kaimyninių simpleksų aibę $\boldsymbol{K}(\mathbb{S})$.
   2.3. $\acute{L}_{\mathbb{S}}$ pagal formulę (5).
   2.4. $\acute{M}_{\mathbb{S}}$ = VIDINIS($\mathbb{S}$, $\acute{L}_{\mathbb{S}}$).
3. Atrinkti potencialius simpleksus, sudarant iškiląjį kontūrą, kaip aprašyta 2.3 skyrelyje.
4. Padalinti potencialius simpleksus pusiau, atliekant matavimą ilgiausios kraštinės viduryje. Prie $\boldsymbol{S}$ pridėti naujus ir pašalinti padalintus simpleksus. Naujais matavimais papildyti sekas $\{\boldsymbol{x}_i\}$, $\{y_i\}$.
5. Patikrinti sustojimo sąlygą. Jeigu ji nepatenkinama, grįžtama į 2 žingsnį.
6. **return** $\{\boldsymbol{x}_i\}$, $\{y_i\}$.

3. Eksperimentinis tyrimas

Pasiūlytasis algoritmas buvo realizuotas C++ kalba. Algoritmas realizuotas vengiant perteklinių skaičiavimų, pavyzdžiui, kiekvienoje algoritmo iteracijoje atnaujinami įverčiai tik tų simpleksų, kurių bent vieno kaimyno įvertis keitėsi ankstesnėje iteracijoje. Eksperimentų metodika analogiška metodikai, naudotai [6–8] straipsniuose.

Eksperimentai atlikti naudojant 4 skirtingų klasių testines funkcijas iš GKLS funkcijų generatoriaus [3]. Funkcijos gaunamos daugiamatę kvadratinę funkciją (paraboloidą) sistemiškai deformuojant polinomais. Kiekvienoje klasėje yra po 100 testinių funkcijų, taigi iš viso eksperimentuota su 400 testinių funkcijų. GKLS funkcijų klases apibrėžia šie parametrai:
1. d – uždavinio dimensija,
2. m – lokalių minimumų skaičius,
3. f^* – globalaus minimumo reikšmė,
4. ρ – globalaus minimumo atstumas iki paraboloido viršūnės,
5. r – globalaus minimumo traukos regiono spindulys.

Klasių numeracija ir naudoti parametrai pateikti 1 lentelėje.

1 lentelė. ***GKLS testinių funkcijų klasių ir tikslumo parametrai***

Klasė	d	m	f^*	ρ	r	Δ
1	2	10	-1	0,90	0,2	10^{-4}
2	2	10	-1	0,90	0,1	10^{-4}
3	3	10	-1	0,66	0,2	10^{-6}
4	3	10	-1	0,90	0,2	10^{-6}

Pasiūlytasis algoritmas buvo eksperimentiškai palygintas su šiais algoritmais: DIRECT, DIRECT*l*, DISIMPL-V, Gb-DISIMPL-V. Atliekant eksperimentus su testinėmis funkcijomis, naudotas sustojimo kriterijus, įvertinantis atstumą nuo artimiausio įvertinimo iki iš anksto žinomo globalaus minimumo taško. Sustojama, atlikus matavimą taške $\boldsymbol{x}_j \in A$, tenkinantį sąlygą:

$$\left|x_j(i) - x^*(i)\right| \le \sqrt[d]{\Delta}\big(b(i) - a(i)\big), 1 \le i \le d\,, \quad (14)$$

čia $\boldsymbol{x}^* \in A$ yra žinomas globalaus minimumo taškas, Δ – tikslumo parametras (žr. 1 lentelę). Jeigu algoritmas nepatenkina sustojimo sąlygos per 1 000 000 funkcijos įvertinimų, jis vis tiek sustabdomas.

Pasiūlytojo algoritmo rezultatai su GKLS funkcijų klasėmis pateikti 2 lentelėje. Iš jos galima matyti, kad vidutinė vykdymo trukmė su visomis klasėmis yra ne ilgesnė nei 14 sekundžių, nors pasiūlytajame algoritme ir sprendžiamas antrojo lygio optimizavimo uždavinys.

2 lentelė. ***Pasiūlytojo algoritmo rezultatai su GKLS funkcijų klasėmis***

Klasė	d	Funkcijų sk.	Sudėtingumas	Iškviet. vidurkis	Iškviet. mediana	Daugiausia iškviet.	Vidutinė trukmė
1	2	100	Lengva	103,68	91	277	0:00:00,1
2	2	100	Sunki	384,02	298,5	1714	0:00:00,6
3	3	100	Lengva	963,21	866,5	2162	0:00:11,9
4	3	100	Sunki	1079,56	1029,5	2859	0:00:13,9

3–6 lentelėse pasiūlytasis algoritmas lyginamas su kitais populiariais algoritmais. Lentelėse paryškinti geresnio iš lyginamų algoritmų rezultatai. Iš 3, 4 ir 5 lentelių matyti, kad pasiūlytojo algoritmo rezultatai yra geresni beveik visais atvejais. Su DIRECT, DIRECT*l* algoritmais gauta geresnė tik trečios klasės iškvietimų mediana. Tai galima paaiškinti tuo, kad naudojant paprastesnius algoritmus galima išspręsti lengvus uždavinius su mažesniu funkcijų įvertinimų skaičiumi nei naudojant sudėtingus algoritmus. Bet sprendžiant sudėtingas problemas paprastesnis algoritmas gali sunaudoti kur kas didesnį funkcijų įvertinimų skaičių nei sudėtingesnis algoritmas. Taigi pasiūlytasis algoritmas yra tinkamesnis sprendžiant sudėtingas problemas.

3 lentelė. ***DIRECT ir pasiūlytojo algoritmo rezultatų palyginimas***

Klasė	Algoritmas	Iškvietimų vidurkis	Iškvietimų mediana	Daugiausia iškvietimų
1	DIRECT	198,89	111	1159
	Pasiūlytasis	**103,68**	**91**	**277**
2	DIRECT	1063,78	1062	3201
	Pasiūlytasis	**384,02**	**298,5**	**1714**
3	DIRECT	1117,70	**386**	12507
	Pasiūlytasis	**963,21**	866,5	**2162**
4	DIRECT	> 42322,65	1749	> 1000000 (4)
	Pasiūlytasis	**1079,56**	**1029,5**	**2859**

4 lentelė. ***DIRECTl ir pasiūlytojo algoritmo rezultatų palyginimas***

Klasė	Algoritmas	Iškvietimų vidurkis	Iškvietimų mediana	Daugiausia iškvietimų
1	*DIRECTl*	292,79	152	2318
	Pasiūlytasis	**103,68**	**91**	**277**
2	*DIRECTl*	1267,07	1328	3414
	Pasiūlytasis	**384,02**	**298,5**	**1714**
3	*DIRECTl*	1785,73	**591**	13309
	Pasiūlytasis	**963,21**	866,5	**2162**
4	*DIRECTl*	4858,93	1967	29233
	Pasiūlytasis	**1079,56**	**1029,5**	**2859**

5 lentelė. ***DISIMPL-V ir pasiūlytojo algoritmo rezultatų palyginimas***

Klasė	Algoritmas	Iškvietimų vidurkis	Iškvietimų mediana	Daugiausia iškvietimų
1	DISIMPL-V	192,93	151	773
	Pasiūlytasis	**103,68**	**91**	**277**
2	DISIMPL-V	1003,56	1021	2683
	Pasiūlytasis	**384,02**	**298,5**	**1714**
3	DISIMPL-V	1061,83	787	4740
	Pasiūlytasis	**963,21**	**866,5**	**2162**
4	DISIMPL-V	2598,91	2594	7354
	Pasiūlytasis	**1079,56**	**1029,5**	**2859**

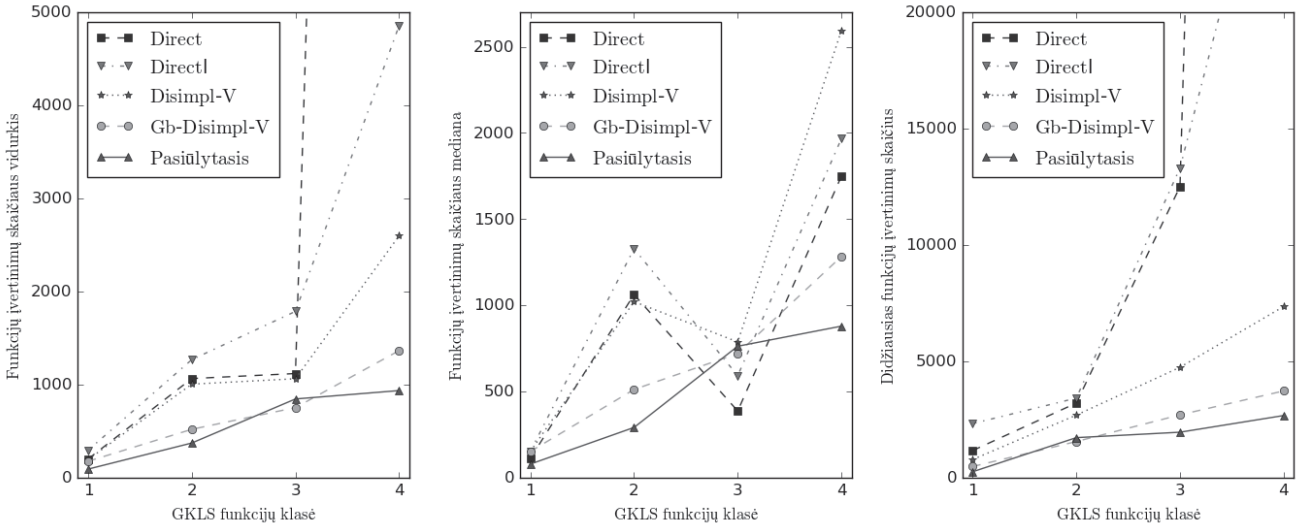
Iš lentelės (žr. 6 lentelę) matyti, kad naudojant Gb-DISIMPL-V algoritmą trečios klasės problemas galima vidutiniškai išspręsti su mažiau funkcijų įvertinimų, nors su pasiūlytuoju algoritmu ir gaunamas geresnis blogiausias atvejis. Tai dar kartą patvirtina, kad pasiūlytasis algoritmas yra tinkamesnis sprendžiant sudėtingas problemas.

Matyti, kad naudojant Gb-DISIMPL-V algoritmą gaunamas geresnis blogiausias atvejis su antros klasės sudėtingomis problemomis, todėl negalima teigti, kad pasiūlytasis algoritmas yra vienareikšmiškai tinkamesnis sprendžiant sudėtingas problemas nei Gb-DISIMPL-V.

6 lentelė. ***Gb-DISIMPL-V ir pasiūlytojo algoritmo rezultatų palyginimas***

Klasė	Algoritmas	Iškvietimų vidurkis	Iškvietimų mediana	Daugiausia iškvietimų
1	Gb-DISIMPL-V	174,25	151	472
	Pasiūlytasis	**103,68**	**91**	**277**
2	Gb-DISIMPL-V	518,11	511	**1547**
	Pasiūlytasis	**384,02**	**298,5**	1714
3	Gb-DISIMPL-V	**751,25**	**720**	2694
	Pasiūlytasis	963,21	866,5	**2162**
4	Gb-DISIMPL-V	1364,40	1283	3723
	Pasiūlytasis	**1079,56**	**1029,5**	**2859**

Paveiksle (žr. 2 pav.) grafiškai vizualizuoti 3–6 lentelėse pateikti duomenys. Iš šių grafikų galima matyti, kad Gb-DISIMPL-V ir pasiūlytojo algoritmo rezultatai yra santykinai panašūs ir tiek funkcijos įvertinimų vidurkių, tiek didžiausio iškvietimų skaičiaus atvejais rezultatai yra ženkliai geresni už kitų algoritmų.



2 pav. Vidutinis funkcijų įvertinimų skaičius, įvertinimų skaičiaus mediana ir didžiausias įvertinimų skaičius, gautas su įvairiais algoritmais optimizuojant funkcijas, sugeneruotas su GKLS testinių funkcijų generatoriumi

4. Išvados

Eksperimentiškai parodyta, kad:

1. Pasiūlytasis algoritmas yra perspektyvus sprendžiant dviejų ir trijų dimensijų optimizavimo problemas.
2. Naudojant pasiūlytąjį algoritmą sudėtingas dviejų ir trijų dimensijų problemas galima išspręsti su mažesniu funkcijų įvertinimų skaičiumi nei naudojant DIRECT, DIRECT*l* ar DISIMPL-V algoritmus.
3. Daugeliu atvejų sprendžiant sudėtingas optimizavimo problemas su pasiūlytuoju algoritmu buvo gauti geresni rezultatai nei su Gb-DISIMPL-V algoritmu.

Taigi parodyta, kad lokalų Lipšico konstantos įvertį naudojantys algoritmai yra perspektyvūs ir turėtų būti tyrinėjami plačiau.

Viena iš ateities tyrimų krypčių galėtų būti pasiūlytojo algoritmo veikimo su didesnės dimensijos problemomis tyrimas. Kita tyrimų kryptis – ieškoti tikslesnio minimalių galimų funkcijos reikšmių simplekse įverčio. Be to, reikėtų ištirti kaimynų parinkimo ir lokalios Lipšico konstantos parinkimo strategijų įtaką algoritmo kokybei.

Literatūra

1. Björkman M., Holmström K., 1999, Global optimization using the DIRECT algorithm in matlab. *Advanced Modeling and Optimization*. Vol. 1. No. 2. P. 17–37.
2. Gablonsky J. M., Kelley C. T., 2001, A locally-biased form of the DIRECT algorithm. *Journal of Global Optimization*. Vol. 21. No. 1. P. 27–37.
3. Gaviano M., Kvasov D. E., Lera D., Sergeyev Y. D., 2003, Algorithm 829: software for generation of classes of test functions with known local and global minima for global optimization. *ACM Trans. Math. Softw*. Vol. 29. No. 4. P. 469–480.
4. Jones D. R., Perttunen C. D., Stuckman B. E., 1993, Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*. Vol. 79. No. 1. P. 157–181.
5. Kelley C. T., 1999, *Iterative methods for optimization*. Raleigh: SIAM.
6. Paulavičius R., Sergeyev Y. D., Kvasov D. E., Žilinskas J., 2014, Globally-biased DISIMPL algorithm for expensive global optimization. *Journal of Global Optimization*. Vol. 59. No. 2–3. P. 545–567.
7. Paulavičius R., Žilinskas J., 2014, *Simplicial Global Optimization*. London: Springer.
8. Paulavičius R., Žilinskas J., 2014, Simplicial Lipschitz optimization without the Lipschitz constant. *Journal of Global Optimization*. Vol. 1. No. 59. P. 23–40.
9. Pijavskij S. A., 1967, An algorithm for finding the global extremum of function. *Optimal Decisions*. Vol. 2. P. 13–24.
10. Shubert B. O., 1972, A sequential method seeking the global maximum of a function. *SIAM Journal on Numerical Analysis*. Vol. 9. No. 3. P. 379–388.

Summary

GLOBAL OPTIMIZATION ALGORITHM USING LOCAL LIPSCHITZ CONSTANT ESTIMATE

A. Gimbutas

In this paper, we propose a new deterministic global optimization algorithm for black-box Lipschitz functions with an unknown Lipschitz constant. At the beginning of the proposed algorithm the feasible region is divided into simplices. At each iteration of algorithm the local Lipschitz constant estimate is found and the lowest possible function value over the simplex is estimated for each simplex; the most promising simplices are selected and divided. An inner optimization problem is solved to find the lowest possible function value estimate for each simplex. A sub-algorithm is proposed to solve the inner optimization problem. Experiments were performed with two- and three- dimensional optimization problems using 400 test functions generated with the GKLS generator. The results showed that complex problems can be solved with less function evaluations using the proposed algorithm than using other most popular alternatives.

Keywords: Lipschitz optimization, global optimization, deterministic optimization.

Santrauka

GLOBALIOS OPTIMIZACIJOS ALGORITMAS, NAUDOJANTIS LOKALŲ LIPŠICO KONSTANTOS ĮVERTĮ

A. Gimbutas

Šiame darbe pasiūlytas naujas deterministinis globalios optimizacijos algoritmas, skirtas juodos dėžės funkcijoms, kurioms galioja Lipšico sąlyga, bet Lipšico konstanta nežinoma. Algoritmo pradinėje stadijoje leistinoji sritis yra padalinama simpleksais. Kiekvienoje algoritmo iteracijoje visiems simpleksams randami Lipšico konstantos įverčiai ir galimos mažiausios funkcijos reikšmės simplekse įverčiai; perspektyviausi simpleksai yra atrenkami ir padalinami. Galimai mažiausiai funkcijos reikšmei simplekse rasti sprendžiamas vidinis optimizavimo uždavinys, norint kurį išspręsti buvo pasiūlytas vidinis algoritmas. Eksperimentai atlikti su dviejų ir trijų dimensijų optimizavimo uždaviniais, panaudojant 400 testinių funkcijų, sugeneruotų su GKLS funkcijų generatoriumi. Rezultatai parodė, kad sudėtingi uždaviniai su pasiūlytuoju algoritmu išsprendžiami su mažesniu funkcijos įvertinimų skaičiumi negu su kitais alternatyviais algoritmais.

Prasminiai žodžiai: Lipšico optimizavimas, globalioji optimizacija, deterministinis optimizavimas.

Įteikta 2016-02-09
Priimta 2016-05-23