Neiškiliojo optimizavimo algoritmas su nauju bikriteriniu potencialiųjų simpleksų išrinkimu naudojant Lipšico konstantos įvertį

Albertas Gimbutas

2018 m. birželio 19 d.

Vadovas: **Prof. habil. dr. Antanas Žilinskas**

Įvadas

Tyrimų sritis

- Įvairiose inžinerinėse srityse sutinkamas globaliosios optimizacijos uždavinys

$$f(\boldsymbol{x}^*) = \min_{\boldsymbol{x} \in D} f(\boldsymbol{x}), \tag{1}$$

  čia $D = [\boldsymbol{l}, \boldsymbol{u}] = \{\boldsymbol{x} \in \mathbb{R}^d : \ l_i \leq x_i \leq u_i, \ i = 1, ..., d\}$.

- Dažnoje *juodosios dėžės* situacijoje, kai tikslo funkcijos analitinė išraiška nėra žinoma, sėkmingai taikomi Lipšico optimizacijos algoritmai, kurie daro prielaidą, kad tikslo funkcija yra Lipšico tolydi:

$$|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)| \leq L\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|, \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in D, \tag{2}$$

  čia L vadinama Lipšico konstanta, kuri dažniausiai nežinoma, o $\|\cdot\|$ dažniausiai yra Euklidinės normos atstumas.

Darbo tikslai

Siekiama pasiūlyti Lipšico optimizavimo algoritmą nenaudojantį Lipšico konstantos, siekiant padidinti efektyvumą tikslo funkcijų įvertinimų skaičiaus atžvilgiu.

Darbo tikslai

1. Pasiūlyti algoritmą, kuris apjungtų stipriąsias savybes DIRECT-tipo algoritmų ir algoritmų, naudojančių Lipšico konstantos įvertį, siekiant padidinti efektyvumą tikslo funkcijos įvertinimų skaičiaus atžvilgiu.
2. Ištirti surogatinių Lipšico rėžių griežtinimo įtaką sunkių tikslo funkcijų optimizavimo proceso efektyvumui.

Darbo uždaviniai

1. Identifikuoti DIRECT tipo simpleksinių Lipšico optimizavimo algoritmų teigiamus aspektus ir pasiūlyti algoritmo modifikaciją, išsaugančią šiuos aspektus bei naudojančią adaptyviai randamą Lipšico konstantos įvertį, kad būtų gautas geresnių savybių optimizavimo algoritmas.
2. Patikrinti pasiūlytųjų ir populiarių algoritmų efektyvumą sprendžiant dirbtinai sugeneruotas sudėtingas optimizavimo problemas.
3. Eksperimentiškai palyginti, kokį efektą turi skirtingo griežtumo surogatinių Lipšico rėžių strategijos optimizavimo algoritmų efektyvumui.

Rezultatų santrauka

Rezultatai

- Lipšico optimizacijos algoritmai taikytini, kai funkcijos reikšmių skaičiavimas brangus, dėl to svarbu didinti algoritmų efektyvumą, mažinant sprendinio paieškai sunaudojamų tikslo funkcijų įvertinimų skaičių.
  - Tuo tikslu disertacijoje pasiūlomas Lipšico globaliojo optimizavimo derinimo su lokaliąja paieška metodas LIBRE.
  - Pasiūlomos LIBRE algoritmo modifikacijos bei jos skaitiškai palyginamos.

Ginamieji teiginiai

1. Pasiūlytasis globalaus optimizavimo algoritmas LIBRE yra efektyvesnis tikslo funkcijos įvertinimų skaičiaus atžvilgiu nei kitos populiariausios alternatyvos blogiausio atvejo atžvilgiu, sprendžiant sunkius globalaus optimizavimo uždavinius.
2. Tikslesnių surogatinių Lipšico rėžių naudojimas ne visada padidina DIRECT tipo optimizavimo algoritmo efektyvumą tikslo funkcijos įvertinimų skaičiaus atžvilgiu.

Lipšico optimizacijos globaliosios paieškos efektyvumo didinimas

Globaliosios paieškos efektyvumo didinimas

- Brangūs tikslo funkcijos įvertinimai skatina didinti algoritmų efektyvumą.
- Populiarus DIRECT Lipšico optimizavimo algoritmas derina lokaliąją ir globaliąją paieškas.
- Neseniai sukurti DISIMPL ir GB-DISIMPL [1] algoritmai, kuriuose naudojamas dalinimas simpleksais. Parodyta, kad šie metodai yra efektyvesni nei DIRECT bei paieškos sritis gali būti sumažinta atsižvelgus į tikslo funkcijos simetriškumus.
- Siekiama dar labiau sumažinti tikslo funkcijos įvertinimų skaičių.

[1] Paulavičius et al [2014]

Disimpl-v algoritmas

Algorithm: DISIMPL-V algoritmas

1 Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.
2 Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.
3 Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali viršūnė $S, i = 1, ..., 2^d\}$. Nustatyti f_{min}, $m = 2^d$.
4 **while** $m < M_{max}$ **do**
5 Select a set of potentially optimal simplices for division.
6 S_j is potentially optimal if $\exists\, \widehat{L} \in [0, \infty)$ such, that:
7 $\min_{\boldsymbol{v} \in V(S_j)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_j) \ \leq\ \min_{\boldsymbol{v} \in V(S_i)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_i),\ \forall S_i \in S,$
8 $\min_{\boldsymbol{v} \in V(S_j)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_j) \ \leq\ f_{min} - \epsilon |f_{min}|,$
9 **foreach** $S_l \in P$ **do**
10 Divide S_l into two new simplices S_l^1, S_l^2, by adding a vertex $\boldsymbol{v}$ in the middle of the longest edge of S_l.
11 Update $S = S \setminus \{S_l\} \cup \{S_l^1, S_l^2\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$, set $m = m + 1$ and update f_{min}.

Disimpl-v algoritmas

Algorithm: DISIMPL-V algoritmas

1 Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.
2 Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.
3 Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali viršūnė $S, i = 1, ..., 2^d\}$. Nustatyti f_{min}, $m = 2^d$.
4 **while** $m < M_{max}$ **do**
5 Select a set of potentially optimal simplices for division.
6 S_j is potentially optimal if $\exists\, \widehat{L} \in [0, \infty)$ such, that:
7 $\min_{\boldsymbol{v} \in V(S_j)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_j) \ \leq\ \min_{\boldsymbol{v} \in V(S_i)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_i),\ \forall S_i \in S,$
8 $\min_{\boldsymbol{v} \in V(S_j)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_j) \ \leq\ f_{min} - \epsilon |f_{min}|,$
9 **foreach** $S_l \in P$ **do**
10 Divide S_l into two new simplices S_l^1, S_l^2, by adding a vertex $\boldsymbol{v}$ in the middle of the longest edge of S_l.
11 Update $S = S \setminus \{S_l\} \cup \{S_l^1, S_l^2\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$, set $m = m + 1$ and update f_{min}.

Disimpl-v algoritmas

Algorithm: DISIMPL-V algoritmas

Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.
Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.
Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali viršūnė $S, i = 1, ..., 2^d\}$. Nustatyti f_{min}, $m = 2^d$.
while $m < M_{max}$ **do**
    Select a set of potentially optimal simplices for division.
    S_j is potentially optimal if $\exists\, \widehat{L} \in [0, \infty)$ such, that:
        $\min_{\boldsymbol{v} \in V(S_j)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_j) \ \leq\ \min_{\boldsymbol{v} \in V(S_i)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_i),\ \forall S_i \in S,$
        $\min_{\boldsymbol{v} \in V(S_j)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_j) \ \leq\ f_{min} - \epsilon |f_{min}|,$
    foreach $S_l \in P$ **do**
        Divide S_l into two new simplices S_l^1, S_l^2, by adding a vertex $\boldsymbol{v}$ in the middle of the longest edge of S_l.
        Update $S = S \setminus \{S_l\} \cup \{S_l^1, S_l^2\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$, set $m = m + 1$ and update f_{min}.

Disimpl-v algoritmas

Algorithm: DISIMPL-V algoritmas

1 Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.
2 Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.
3 Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali viršūnė $S, i = 1, ..., 2^d\}$. Nustatyti f_{min}, $m = 2^d$.
4 **while** $m < M_{max}$ **do**
5 Select a set of potentially optimal simplices for division.
6 S_j is potentially optimal if $\exists\, \widehat{L} \in [0, \infty)$ such, that:
7 $\min_{\boldsymbol{v} \in V(S_j)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_j) \ \leq\ \min_{\boldsymbol{v} \in V(S_i)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_i),\ \forall S_i \in S,$
8 $\min_{\boldsymbol{v} \in V(S_j)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_j) \ \leq\ f_{min} - \epsilon|f_{min}|,$
9 **foreach** $S_l \in P$ **do**
10 Divide S_l into two new simplices S_l^1, S_l^2, by adding a vertex $\boldsymbol{v}$ in the middle of the longest edge of S_l.
11 Update $S = S \setminus \{S_l\} \cup \{S_l^1, S_l^2\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$, set $m = m + 1$ and update f_{min}.

Disimpl-v algoritmas

Algorithm: DISIMPL-V algoritmas

1 Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.
2 Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.
3 Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali viršūnė $S, i = 1, ..., 2^d\}$. Nustatyti f_{min}, $m = 2^d$.
4 **while** $m < M_{max}$ **do**
5 Select a set of potentially optimal simplices for division.
6 S_j is potentially optimal if $\exists\, \widehat{L} \in [0, \infty)$ such, that:
7 $\min_{\boldsymbol{v} \in V(S_j)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_j) \ \leq\ \min_{\boldsymbol{v} \in V(S_i)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_i),\ \forall S_i \in S,$
8 $\min_{\boldsymbol{v} \in V(S_j)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_j) \ \leq\ f_{min} - \epsilon|f_{min}|,$
9 **foreach** $S_l \in P$ **do**
10 Divide S_l into two new simplices S_l^1, S_l^2, by adding a vertex $\boldsymbol{v}$ in the middle of the longest edge of S_l.
11 Update $S = S \setminus \{S_l\} \cup \{S_l^1, S_l^2\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$, set $m = m + 1$ and update f_{min}.

Disimpl-v algoritmas

Algorithm: DISIMPL-V algoritmas

1 Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.
2 Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.
3 Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali viršūnė $S, i = 1, ..., 2^d\}$. Nustatyti f_{min}, $m = 2^d$.
4 **while** $m < M_{max}$ **do**
5 Select a set of potentially optimal simplices for division.
6 S_j is potentially optimal if $\exists\, \widehat{L} \in [0, \infty)$ such, that:
7 $\min_{\boldsymbol{v} \in V(S_j)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_j) \ \leq\ \min_{\boldsymbol{v} \in V(S_i)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_i),\ \forall S_i \in S,$
8 $\min_{\boldsymbol{v} \in V(S_j)} f(\boldsymbol{v}) - \widehat{L}\Delta(S_j) \ \leq\ f_{min} - \epsilon|f_{min}|,$
9 **foreach** $S_l \in P$ **do**
10 Divide S_l into two new simplices S_l^1, S_l^2, by adding a vertex $\boldsymbol{v}$ in the middle of the longest edge of S_l.
11 Update $S = S \setminus \{S_l\} \cup \{S_l^1, S_l^2\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$, set $m = m + 1$ and update f_{min}.

Disimpl-v pavyzdys

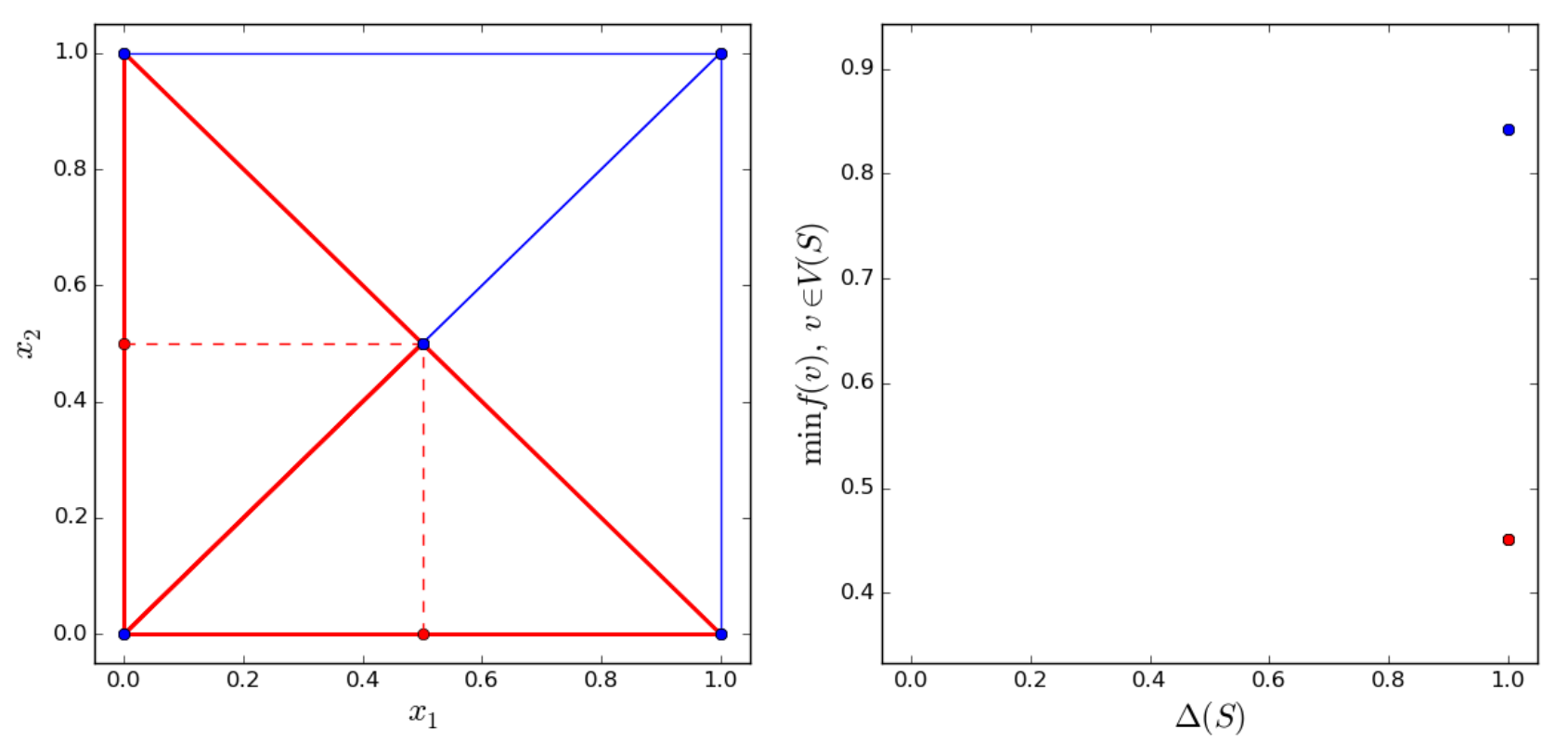


Figure 1: DISIMPL-V algoritmo 1 iteracijos pavyzdys. Kairėje pavaizduotas leistinosios srities padalinimas simpleksais. Dešinėje pavaizduoti simpleksus charakterizuojantys paramterai.

Disimpl-v pavyzdys

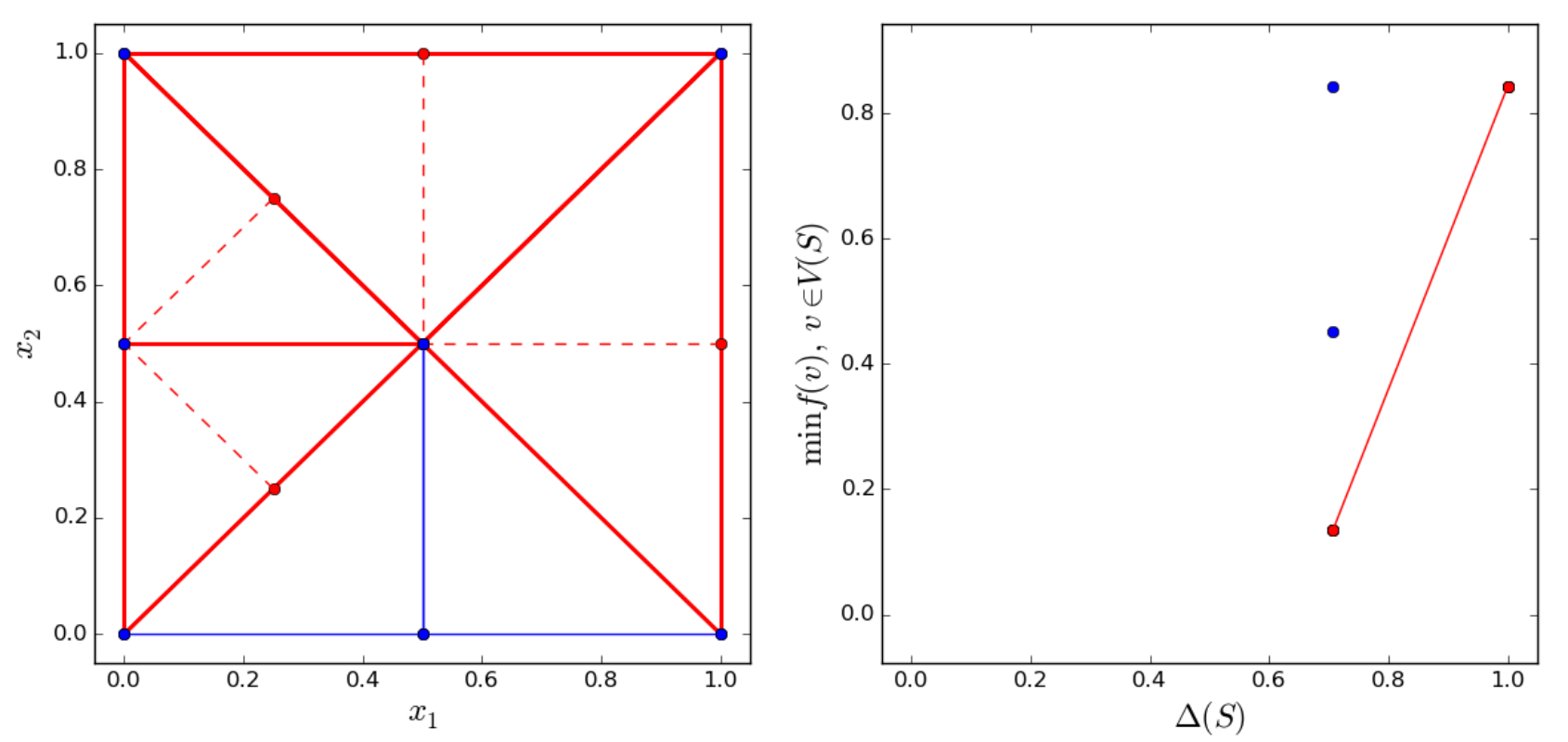


Figure 2: DISIMPL-V algoritmo 2 iteracijos pavyzdys. Kairėje pavaizduotas leistinosios srities padalinimas simpleksais. Dešinėje pavaizduoti simpleksus charakterizuojantys paramterai.

Disimpl-v pavyzdys

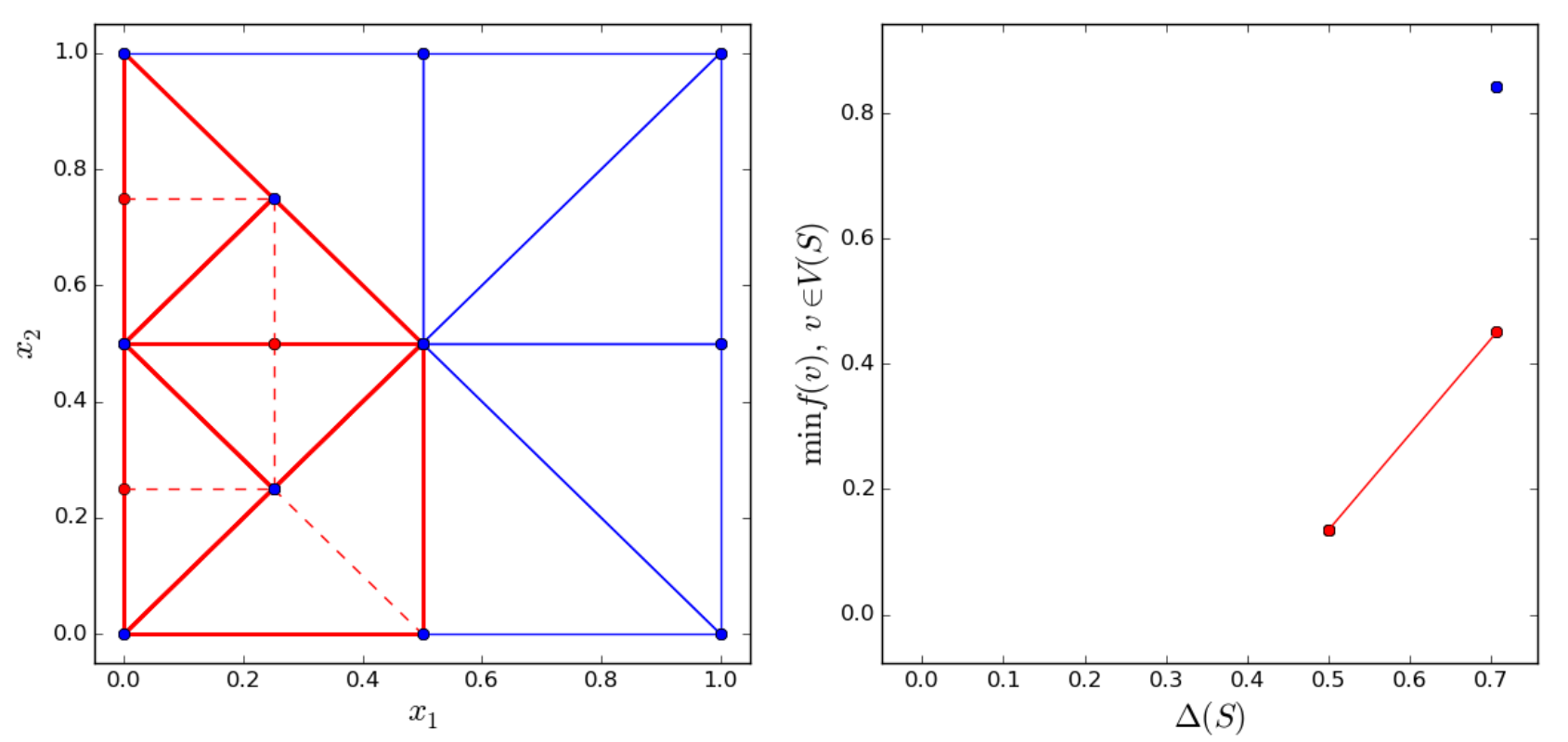


Figure 3: DISIMPL-V algoritmo 3 iteracijos pavyzdys. Kairėje pavaizduotas leistinosios srities padalinimas simpleksais. Dešinėje pavaizduoti simpleksus charakterizuojantys paramterai.

Disimpl-v pavyzdys

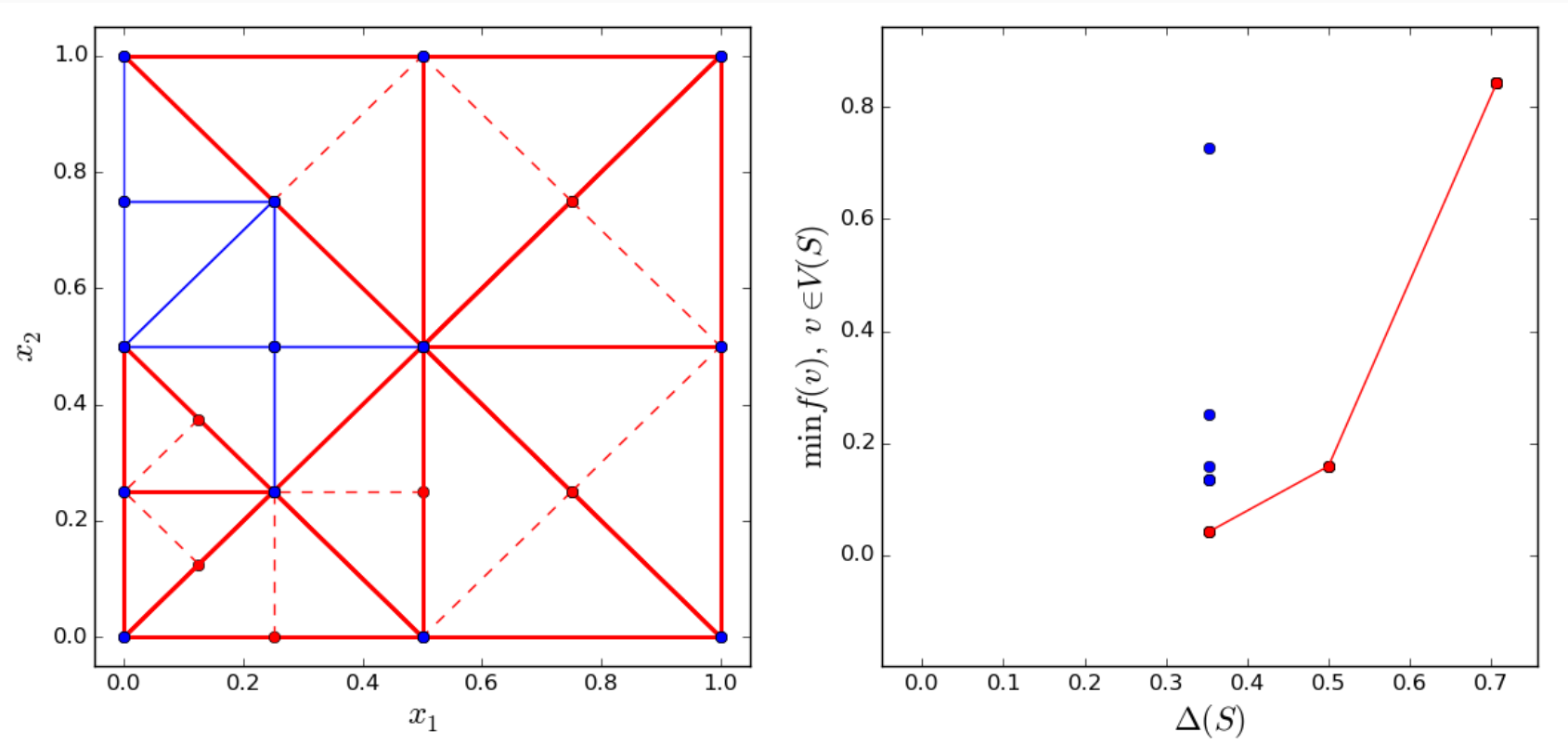


Figure 4: DISIMPL-V algoritmo 4 iteracijos pavyzdys. Kairėje pavaizduotas leistinosios srities padalinimas simpleksais. Dešinėje pavaizduoti simpleksus charakterizuojantys paramterai.

Disimpl-v pavyzdys

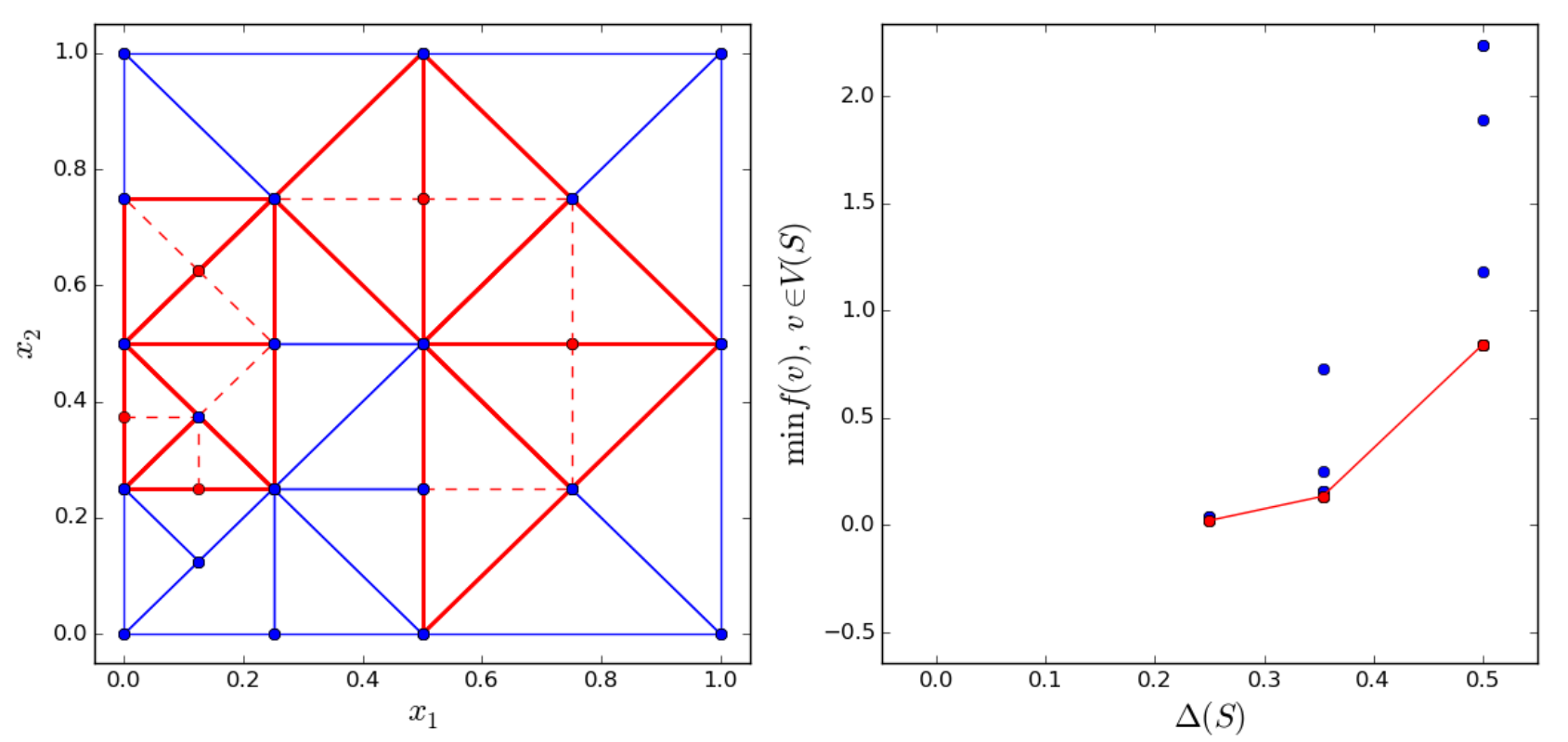


Figure 5: DISIMPL-V algoritmo 5 iteracijos pavyzdys. Kairėje pavaizduotas leistinosios srities padalinimas simpleksais. Dešinėje pavaizduoti simpleksus charakterizuojantys paramterai.

Disimpl-v pavyzdys

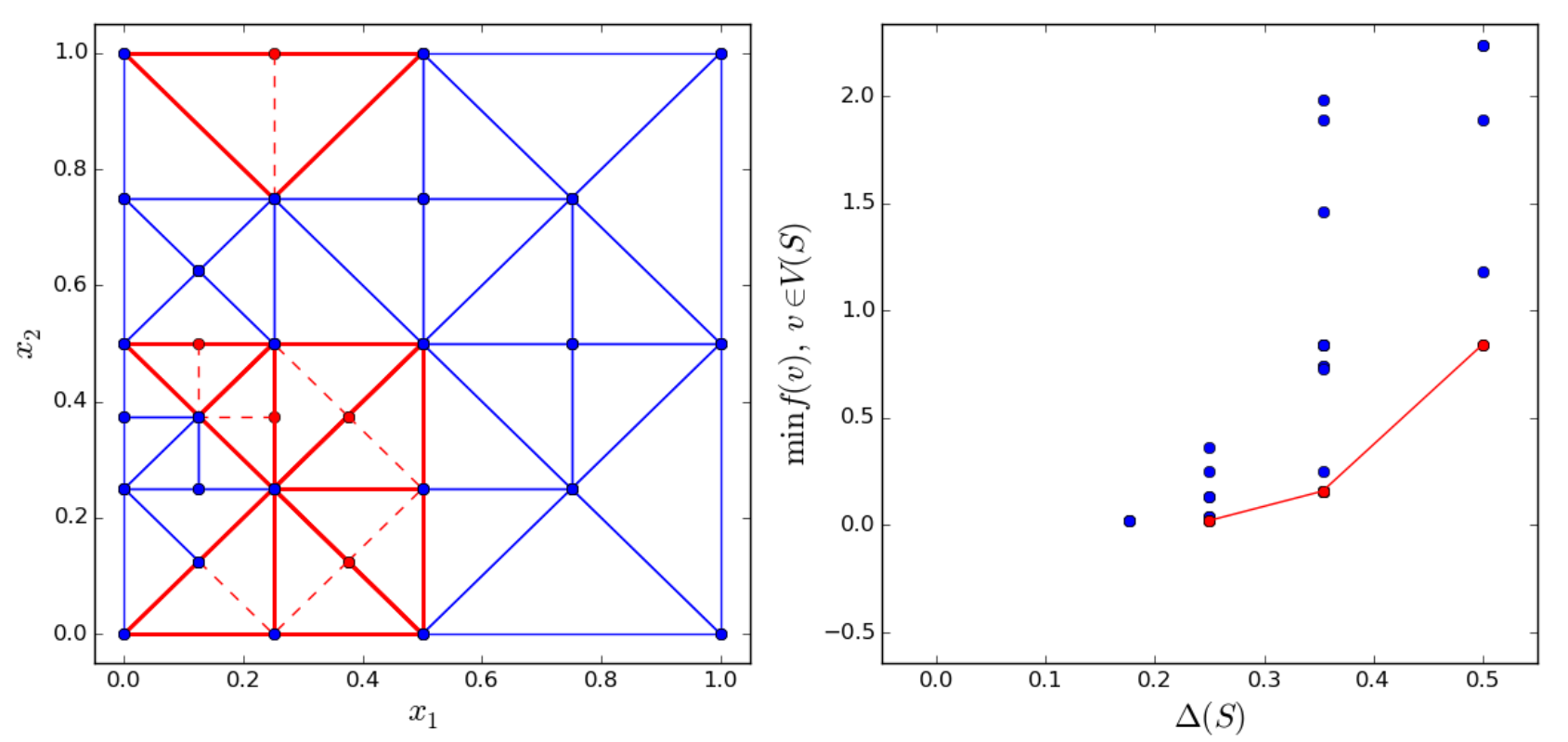


Figure 6: DISIMPL-V algoritmo 6 iteracijos pavyzdys. Kairėje pavaizduotas leistinosios srities padalinimas simpleksais. Dešinėje pavaizduoti simpleksus charakterizuojantys paramterai.

Disimpl-v pavyzdys

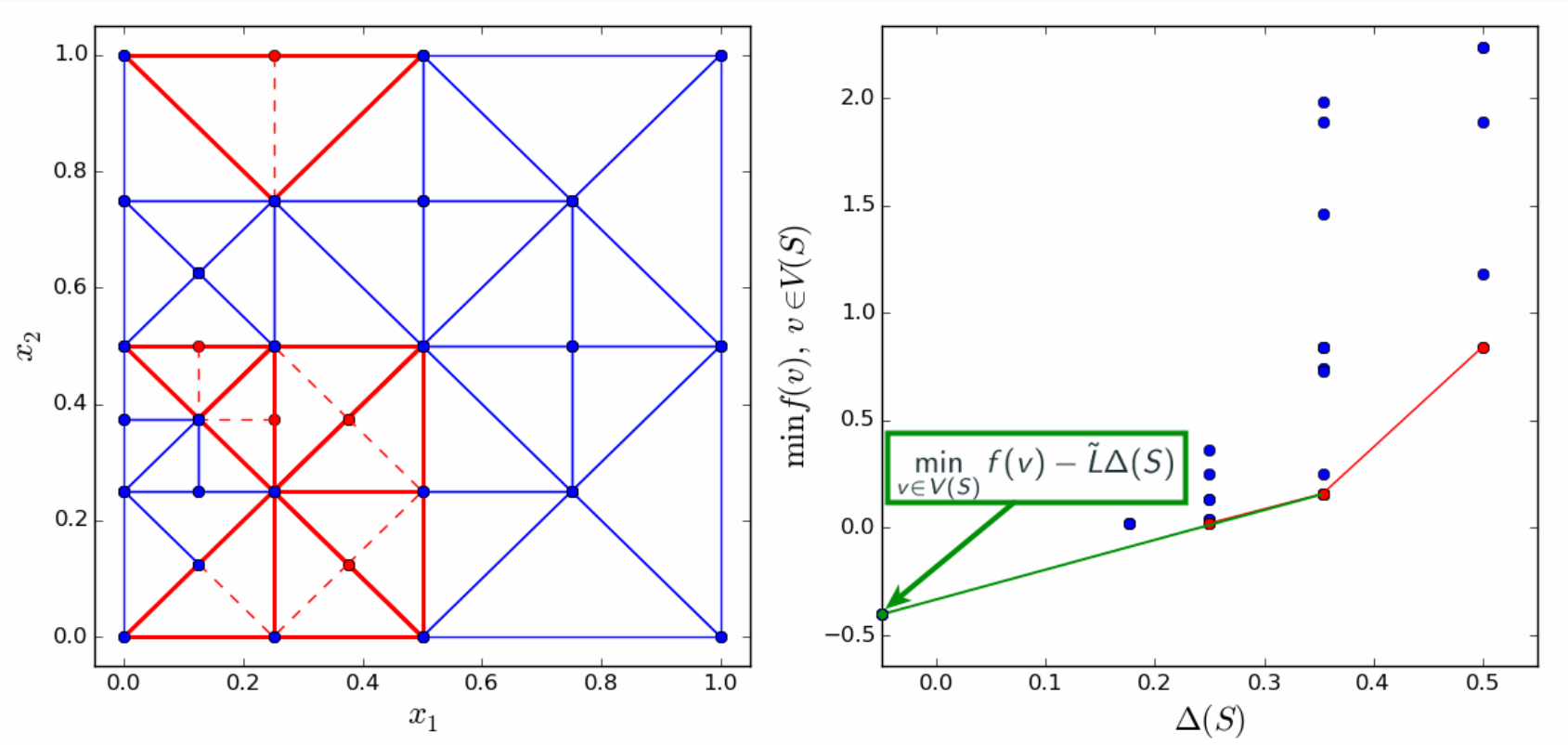


Figure 7: DISIMPL-V algorithm 6 iteration example. Left image shows feasible region partitioned into simplices. Right image shows convex-hull used to select potentially optimal simplices.

Pasiūlytasis Libre algoritmas

Kiekvienos LIBRE algoritmo iteracijos metu yra randamas Lipšico konstantos įvertis $\widetilde{L}$, kuris gaunamas

$$\widetilde{L} = \max\{\ |f(\boldsymbol{v}_i) - f(\boldsymbol{v}_j)|/||\boldsymbol{v}_i - \boldsymbol{v}_j||:\ i \neq j\ \}. \tag{3}$$

Dalinimui parenkami simpleksai, kurie turi geriausią kompromisą tarp kriterijų $\Delta(S_i)$ ir

$$G(S_i, \widetilde{L}_k) = \min_{\boldsymbol{v}_j \in V(S_i)} f(\boldsymbol{v}_j) - \widetilde{L}_k \Delta(S_i)\alpha, \tag{4}$$

t.y. dalinimui parenkami $\min_{S_i \in S}(G(S_i), -\Delta(S_i))$ sprendiniai.

Algorithm: LIBRE – pasiūlytasis globaliojo optimizavimo algoritmas

1 Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.

2 Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \left\{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\right\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.

3 Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali viršūnė $S, i = 1, ..., 2^d\}$. Nustatyti $m = 2^d, \widetilde{L} = 0$.

4 **while** *sustojimo sąlyga nėra patenkinta* **and** $m < M_{max}$ **do**

5 **foreach** $S_l \in S$ **do**

6 Rasti $\widetilde{L}_l = \max\left\{\frac{|f(\boldsymbol{v}_i)-f(\boldsymbol{v}_j)|}{||\boldsymbol{v}_i-\boldsymbol{v}_j||} : \boldsymbol{v}_i, \boldsymbol{v}_j \in V(S_l), \boldsymbol{v}_i \neq \boldsymbol{v}_j\right\}$.

7 Atnaujinti $\widetilde{L} = \max\left\{\widetilde{L}\right\} \cup \left\{\widetilde{L}_l : S_l \in S\right\}$.

8 **foreach** $S_l \in S$ **do**

9 find $G(S_l, \widetilde{L}) = \min_{\boldsymbol{v}_i \in V(S_l)} f(\boldsymbol{v}_i) - \widetilde{L}\Delta(S_l)\alpha$

10 Parinkti aibę simpleksų dalinimui: $P = \min\limits_{S_i \in S}(G(S_i), -\Delta(S_i))$

11 **foreach** $S_l \in P$ **do**

12 Padalinti S_l į du naujus simpleksus S_l^1, S_l^2, pridedant viršūnę $\boldsymbol{v}$ ilgiausios S_l briaunos vidurio taške.

13 Atnaujinti $S = S \setminus \left\{S_l\right\} \cup \left\{S_l^1, S_l^2\right\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$ and set $m = m + 1$.

Algorithm: LIBRE – pasiūlytasis globaliojo optimizavimo algoritmas

1 Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.

2 Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \left\{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\right\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.

3 Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali viršūnė $S, i = 1, ..., 2^d\}$. Nustatyti $m = 2^d, \widetilde{L} = 0$.

4 **while** *sustojimo sąlyga nėra patenkinta* **and** $m < M_{max}$ **do**

5 **foreach** $S_l \in S$ **do**

6 Rasti $\widetilde{L}_l = \max\left\{\frac{|f(\boldsymbol{v}_i)-f(\boldsymbol{v}_j)|}{||\boldsymbol{v}_i-\boldsymbol{v}_j||} : \boldsymbol{v}_i, \boldsymbol{v}_j \in V(S_l), \boldsymbol{v}_i \neq \boldsymbol{v}_j\right\}$.

7 Atnaujinti $\widetilde{L} = \max\left\{\widetilde{L}\right\} \cup \left\{\widetilde{L}_l : S_l \in S\right\}$.

8 **foreach** $S_l \in S$ **do**

9 find $G(S_l, \widetilde{L}) = \min_{\boldsymbol{v}_i \in V(S_l)} f(\boldsymbol{v}_i) - \widetilde{L}\Delta(S_l)\alpha$

10 Parinkti aibę simpleksų dalinimui: $P = \min\limits_{S_i \in S}(G(S_i), -\Delta(S_i))$

11 **foreach** $S_l \in P$ **do**

12 Padalinti S_l į du naujus simpleksus S_l^1, S_l^2, pridedant viršūnę $\boldsymbol{v}$ ilgiausios S_l briaunos vidurio taške.

13 Atnaujinti $S = S \setminus \left\{S_l\right\} \cup \left\{S_l^1, S_l^2\right\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$ and set $m = m + 1$.

Algorithm: LIBRE – pasiūlytasis globaliojo optimizavimo algoritmas

1 Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.

2 Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \left\{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\right\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.

3 Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali viršūnė $S, i = 1, ..., 2^d\}$. Nustatyti $m = 2^d, \widetilde{L} = 0$.

4 **while** *sustojimo sąlyga nėra patenkinta* **and** $m < M_{max}$ **do**

5 **foreach** $S_l \in S$ **do**

6 Rasti $\widetilde{L}_l = \max\left\{\frac{|f(\boldsymbol{v}_i)-f(\boldsymbol{v}_j)|}{||\boldsymbol{v}_i-\boldsymbol{v}_j||} : \boldsymbol{v}_i, \boldsymbol{v}_j \in V(S_l), \boldsymbol{v}_i \neq \boldsymbol{v}_j\right\}$.

7 Atnaujinti $\widetilde{L} = \max\left\{\widetilde{L}\right\} \cup \left\{\widetilde{L}_l : S_l \in S\right\}$.

8 **foreach** $S_l \in S$ **do**

9 find $G(S_l, \widetilde{L}) = \min_{\boldsymbol{v}_i \in V(S_l)} f(\boldsymbol{v}_i) - \widetilde{L}\Delta(S_l)\alpha$

10 Parinkti aibę simpleksų dalinimui: $P = \min\limits_{S_i \in S}(G(S_i), -\Delta(S_i))$

11 **foreach** $S_l \in P$ **do**

12 Padalinti S_l į du naujus simpleksus S_l^1, S_l^2, pridedant viršūnę $\boldsymbol{v}$ ilgiausios S_l briaunos vidurio taške.

13 Atnaujinti $S = S \setminus \left\{S_l\right\} \cup \left\{S_l^1, S_l^2\right\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$ and set $m = m + 1$.

Algorithm: LIBRE – pasiūlytasis globaliojo optimizavimo algoritmas

Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.

Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \left\{ S_i : \overline{D} = \cup S_i,\ i = 1, ..., d! \right\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.

Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali viršūnė $S, i = 1, ..., 2^d\}$. Nustatyti $m = 2^d$, $\widetilde{L} = 0$.

while *sustojimo sąlyga nėra patenkinta* **and** $m < M_{max}$ **do**

  foreach $S_l \in S$ **do**

    Rasti $\widetilde{L}_l = \max \left\{ \frac{|f(\boldsymbol{v}_i) - f(\boldsymbol{v}_j)|}{||\boldsymbol{v}_i - \boldsymbol{v}_j||} : \boldsymbol{v}_i, \boldsymbol{v}_j \in V(S_l), \boldsymbol{v}_i \neq \boldsymbol{v}_j \right\}$.

  Atnaujinti $\widetilde{L} = \max \left\{ \widetilde{L} \right\} \cup \left\{ \widetilde{L}_l : S_l \in S \right\}$.

  foreach $S_l \in S$ **do**

    find $G(S_l, \widetilde{L}) = \min_{\boldsymbol{v}_i \in V(S_l)} f(\boldsymbol{v}_i) - \widetilde{L}\Delta(S_l)\alpha$

  Parinkti aibę simpleksų dalinimui: $P = \min\limits_{S_i \in S} (G(S_i), -\Delta(S_i))$

  foreach $S_l \in P$ **do**

    Padalinti S_l į du naujus simpleksus S_l^1, S_l^2, pridedant viršūnę $\boldsymbol{v}$ ilgiausios S_l briaunos vidurio taške.

    Atnaujinti $S = S \setminus \left\{ S_l \right\} \cup \left\{ S_l^1, S_l^2 \right\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$ and set $m = m + 1$.

Algorithm: LIBRE – pasiūlytasis globaliojo optimizavimo algoritmas

1 Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.

2 Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \left\{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\right\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.

3 Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali viršūnė $S, i = 1, ..., 2^d\}$. Nustatyti $m = 2^d, \widetilde{L} = 0$.

4 **while** *sustojimo sąlyga nėra patenkinta* **and** $m < M_{max}$ **do**

5 **foreach** $S_l \in S$ **do**

6 Rasti $\widetilde{L}_l = \max\left\{\frac{|f(\boldsymbol{v}_i)-f(\boldsymbol{v}_j)|}{||\boldsymbol{v}_i-\boldsymbol{v}_j||} : \boldsymbol{v}_i, \boldsymbol{v}_j \in V(S_l), \boldsymbol{v}_i \neq \boldsymbol{v}_j\right\}$.

7 Atnaujinti $\widetilde{L} = \max\left\{\widetilde{L}\right\} \cup \left\{\widetilde{L}_l : S_l \in S\right\}$.

8 **foreach** $S_l \in S$ **do**

9 find $G(S_l, \widetilde{L}) = \min_{\boldsymbol{v}_i \in V(S_l)} f(\boldsymbol{v}_i) - \widetilde{L}\Delta(S_l)\alpha$

10 Parinkti aibę simpleksų dalinimui: $P = \min\limits_{S_i \in S}(G(S_i), -\Delta(S_i))$

11 **foreach** $S_l \in P$ **do**

12 Padalinti S_l į du naujus simpleksus S_l^1, S_l^2, pridedant viršūnę $\boldsymbol{v}$ ilgiausios S_l briaunos vidurio taške.

13 Atnaujinti $S = S \setminus \left\{S_l\right\} \cup \left\{S_l^1, S_l^2\right\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$ and set $m = m + 1$.

Algorithm: LIBRE – pasiūlytasis globaliojo optimizavimo algoritmas

1 Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.

2 Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \left\{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\right\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.

3 Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali viršūnė $S, i = 1, ..., 2^d\}$. Nustatyti $m = 2^d, \widetilde{L} = 0$.

4 **while** *sustojimo sąlyga nėra patenkinta* **and** $m < M_{max}$ **do**

5 **foreach** $S_l \in S$ **do**

6 Rasti $\widetilde{L}_l = \max\left\{\frac{|f(\boldsymbol{v}_i) - f(\boldsymbol{v}_j)|}{||\boldsymbol{v}_i - \boldsymbol{v}_j||} : \boldsymbol{v}_i, \boldsymbol{v}_j \in V(S_l), \boldsymbol{v}_i \neq \boldsymbol{v}_j\right\}$.

7 Atnaujinti $\widetilde{L} = \max\left\{\widetilde{L}\right\} \cup \left\{\widetilde{L}_l : S_l \in S\right\}$.

8 **foreach** $S_l \in S$ **do**

9 find $G(S_l, \widetilde{L}) = \min_{\boldsymbol{v}_i \in V(S_l)} f(\boldsymbol{v}_i) - \widetilde{L}\Delta(S_l)\alpha$

10 Parinkti aibę simpleksų dalinimui: $P = \min\limits_{S_i \in S}(G(S_i), -\Delta(S_i))$

11 **foreach** $S_l \in P$ **do**

12 Padalinti S_l į du naujus simpleksus S_l^1, S_l^2, pridedant viršūnę $\boldsymbol{v}$ ilgiausios S_l briaunos vidurio taške.

13 Atnaujinti $S = S \setminus \left\{S_l\right\} \cup \left\{S_l^1, S_l^2\right\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$ and set $m = m + 1$.

Algorithm: LIBRE – pasiūlytasis globaliojo optimizavimo algoritmas

1 Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.

2 Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \left\{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\right\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.

3 Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali viršūnė $S, i = 1, ..., 2^d\}$. Nustatyti $m = 2^d, \widetilde{L} = 0$.

4 **while** *sustojimo sąlyga nėra patenkinta* **and** $m < M_{max}$ **do**

5 **foreach** $S_l \in S$ **do**

6 Rasti $\widetilde{L}_l = \max\left\{\frac{|f(\boldsymbol{v}_i)-f(\boldsymbol{v}_j)|}{||\boldsymbol{v}_i-\boldsymbol{v}_j||} : \boldsymbol{v}_i, \boldsymbol{v}_j \in V(S_l), \boldsymbol{v}_i \neq \boldsymbol{v}_j\right\}$.

7 Atnaujinti $\widetilde{L} = \max\left\{\widetilde{L}\right\} \cup \left\{\widetilde{L}_l : S_l \in S\right\}$.

8 **foreach** $S_l \in S$ **do**

9 find $G(S_l, \widetilde{L}) = \min_{\boldsymbol{v}_i \in V(S_l)} f(\boldsymbol{v}_i) - \widetilde{L}\Delta(S_l)\alpha$

10 Parinkti aibę simpleksų dalinimui: $P = \min\limits_{S_i \in S}(G(S_i), -\Delta(S_i))$

11 **foreach** $S_l \in P$ **do**

12 Padalinti S_l į du naujus simpleksus S_l^1, S_l^2, pridedant viršūnę $\boldsymbol{v}$ ilgiausios S_l briaunos vidurio taške.

13 Atnaujinti $S = S \setminus \left\{S_l\right\} \cup \left\{S_l^1, S_l^2\right\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$ and set $m = m + 1$.

Algorithm: LIBRE – pasiūlytasis globaliojo optimizavimo algoritmas

1 Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.

2 Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \left\{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\right\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.

3 Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali viršūnė $S, i = 1, ..., 2^d\}$. Nustatyti $m = 2^d, \widetilde{L} = 0$.

4 **while** *sustojimo sąlyga nėra patenkinta* **and** $m < M_{max}$ **do**

5 **foreach** $S_l \in S$ **do**

6 Rasti $\widetilde{L}_l = \max\left\{\frac{|f(\boldsymbol{v}_i)-f(\boldsymbol{v}_j)|}{||\boldsymbol{v}_i-\boldsymbol{v}_j||} : \boldsymbol{v}_i, \boldsymbol{v}_j \in V(S_l), \boldsymbol{v}_i \neq \boldsymbol{v}_j\right\}$.

7 Atnaujinti $\widetilde{L} = \max\left\{\widetilde{L}\right\} \cup \left\{\widetilde{L}_l : S_l \in S\right\}$.

8 **foreach** $S_l \in S$ **do**

9 find $G(S_l, \widetilde{L}) = \min_{\boldsymbol{v}_i \in V(S_l)} f(\boldsymbol{v}_i) - \widetilde{L}\Delta(S_l)\alpha$

10 Parinkti aibę simpleksų dalinimui: $P = \min\limits_{S_i \in S}(G(S_i), -\Delta(S_i))$

11 **foreach** $S_l \in P$ **do**

12 Padalinti S_l į du naujus simpleksus S_l^1, S_l^2, pridedant viršūnę $\boldsymbol{v}$ ilgiausios S_l briaunos vidurio taške.

13 Atnaujinti $S = S \setminus \left\{S_l\right\} \cup \left\{S_l^1, S_l^2\right\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$ and set $m = m + 1$.

Algorithm: LIBRE – pasiūlytasis globaliojo optimizavimo algoritmas

1 Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.

2 Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \left\{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\right\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.

3 Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali viršūnė $S, i = 1, ..., 2^d\}$. Nustatyti $m = 2^d, \widetilde{L} = 0$.

4 **while** *sustojimo sąlyga nėra patenkinta* **and** $m < M_{max}$ **do**

5 **foreach** $S_l \in S$ **do**

6 Rasti $\widetilde{L}_l = \max\left\{\frac{|f(\boldsymbol{v}_i)-f(\boldsymbol{v}_j)|}{||\boldsymbol{v}_i-\boldsymbol{v}_j||} : \boldsymbol{v}_i, \boldsymbol{v}_j \in V(S_l), \boldsymbol{v}_i \neq \boldsymbol{v}_j\right\}$.

7 Atnaujinti $\widetilde{L} = \max\left\{\widetilde{L}\right\} \cup \left\{\widetilde{L}_l : S_l \in S\right\}$.

8 **foreach** $S_l \in S$ **do**

9 find $G(S_l, \widetilde{L}) = \min_{\boldsymbol{v}_i \in V(S_l)} f(\boldsymbol{v}_i) - \widetilde{L}\Delta(S_l)\alpha$

10 Parinkti aibę simpleksų dalinimui: $P = \min\limits_{S_i \in S}(G(S_i), -\Delta(S_i))$

11 **foreach** $S_l \in P$ **do**

12 Padalinti S_l į du naujus simpleksus S_l^1, S_l^2, pridedant viršūnę $\boldsymbol{v}$ ilgiausios S_l briaunos vidurio taške.

13 Atnaujinti $S = S \setminus \left\{S_l\right\} \cup \left\{S_l^1, S_l^2\right\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$ and set $m = m + 1$.

Eksperimentinis palyginimas

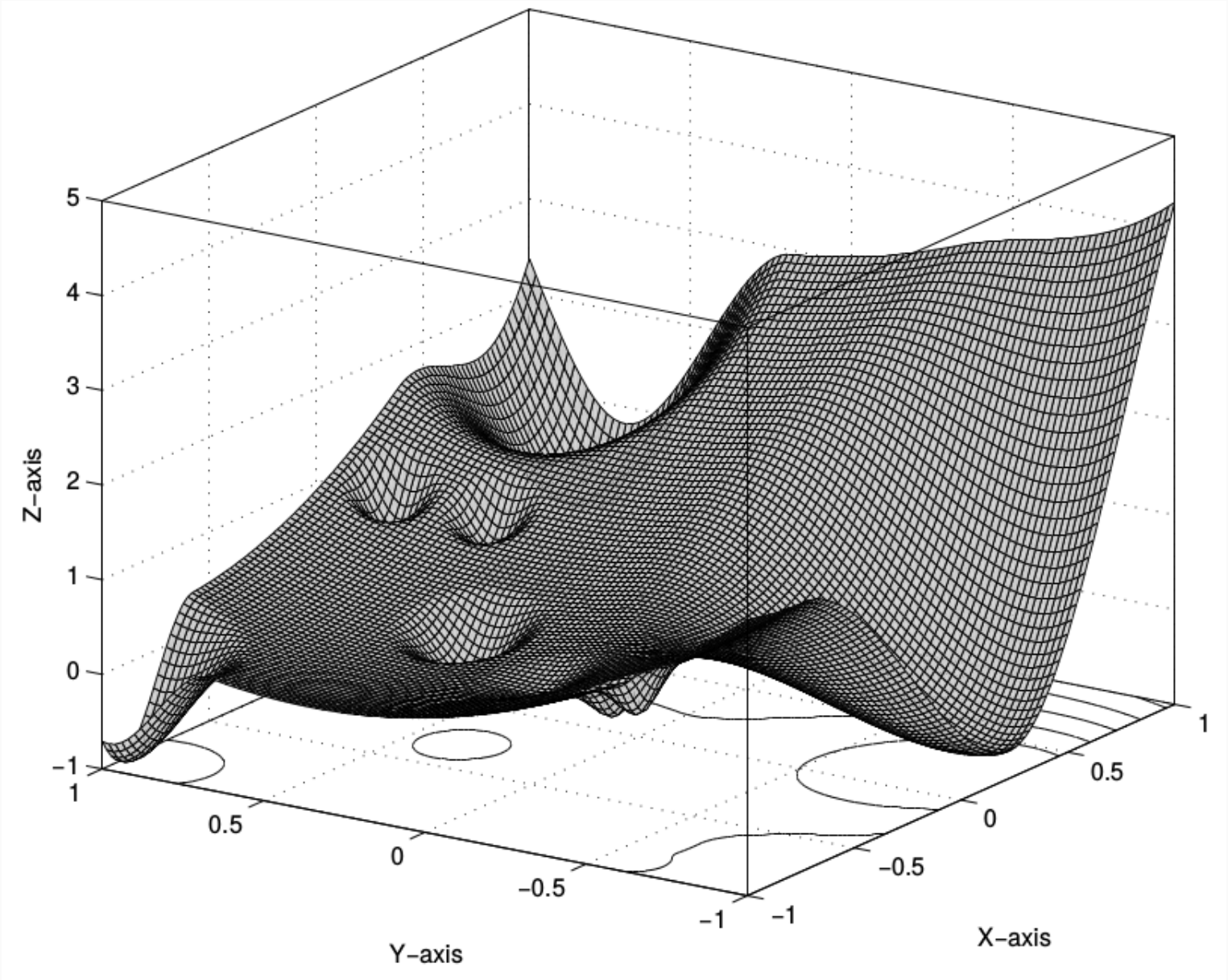
- DISIMPL-V ir pasiūlytasis algoritmai lyginti tarpusavyje bei su kitais populiariais algoritmais.
- Eksperimentinė metodika tokia pati kaip ir kituose darbuose[2]
- Naudotas GKLS testinių funkcijų generatorius [3]
- 8 skirtingo sudėtingumo funkcijų klasės po 100 funkcijų.
- Sustojama, kai randamas taškas $\boldsymbol{x} = (x_i, ..., x_d)$ pakankamai arti iš anksto žinomam globaliajam minimumo taškui $\boldsymbol{x}^* = (x_1^*, ..., x_d^*)$.

$$|x_i - x_i^*| \leq \sqrt[d]{\delta}(u_i - l_i), \ \ i = 1, ..., d, \tag{5}$$

  čia $D = \prod_{i=1}^{d}[a_i, b_i]$ - apibrėžimo sritis, δ - tikslumo parametras.
- Maksimalus funkcijos įvertinimų skaičius $M_{max} = 10^6$.

[2] Sergeyev and Kvasov 2006, Paulavičius et al [2014].
[3] Gaviano et al. [2003].

GKLS testinės funkcijos pavyzdys



GKLS testinių funkcijų klasių parametrai

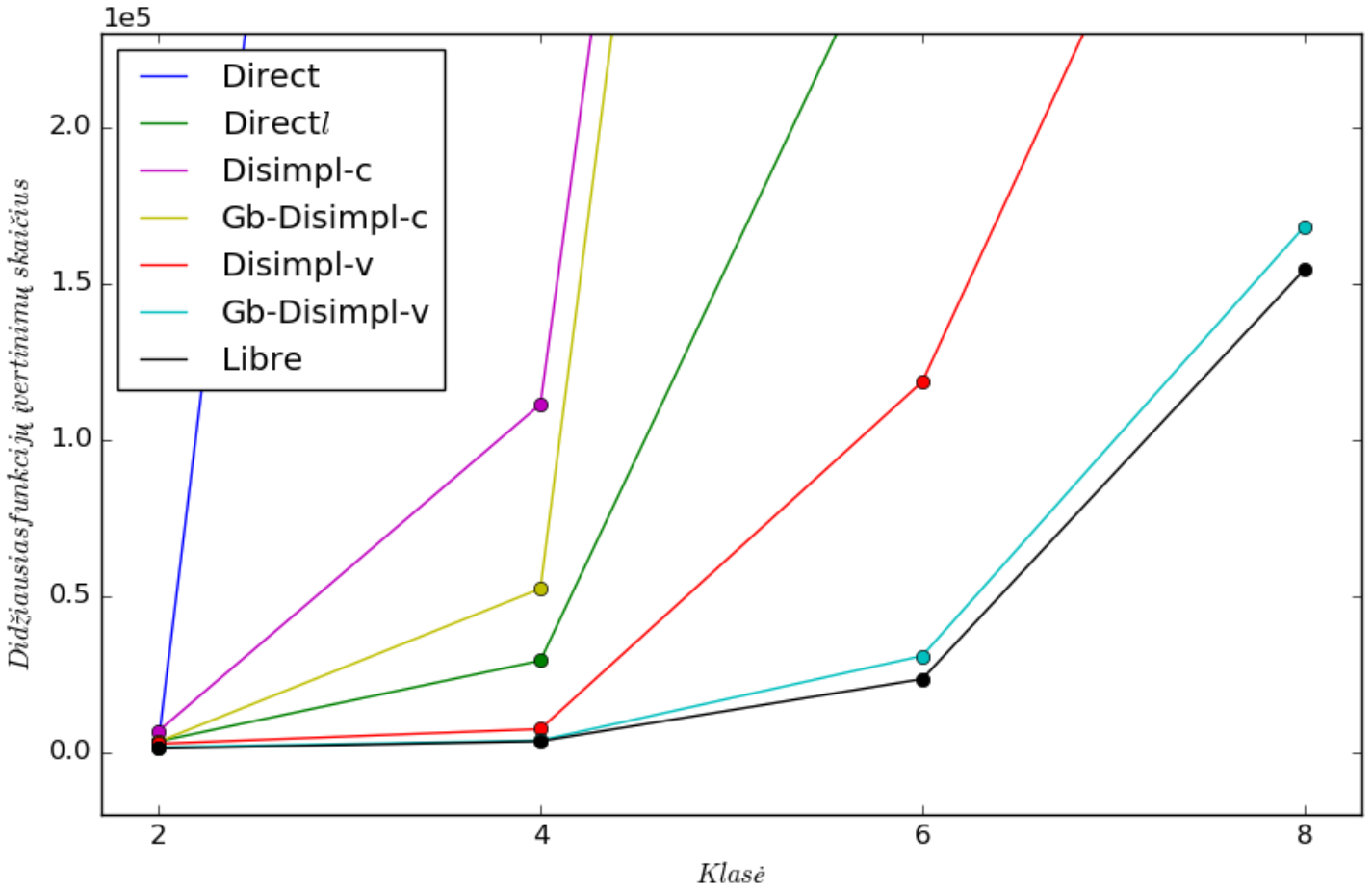
Klasė	Sudėtingumas	d	Minimumų	f^*	ρ	r	δ
1	Paprasta	2	10	-1	0.90	0.2	10^{-4}
2	Sunki	2	10	-1	0.90	0.1	10^{-4}
3	Paprasta	3	10	-1	0.66	0.2	10^{-6}
4	Sunki	3	10	-1	0.90	0.2	10^{-6}
5	Paprasta	4	10	-1	0.66	0.2	10^{-6}
6	Sunki	4	10	-1	0.90	0.2	10^{-6}
7	Paprasta	5	10	-1	0.66	0.3	10^{-7}
8	Sunki	5	10	-1	0.66	0.2	10^{-7}

1. ρ - atstumas tarp globalaus minimumo taško ir foninio paraboloido minimumo taško.
2. r - globalaus minimumo taško traukos srities spindulys.
3. „Sunkios“ klasės turi santykinai didelę ρ ir santykinai mažą r reikšmę.

Rezultatai

Klasė	Sudėtingumas	Algoritmas	Vidurkis	Mediana	Daugiausia
1	Paprasta	Disimpl-V	192.93	151.0	773
		Gb-Disimpl-V	174.25	151.0	472
		Libre	**151.97**	**146.5**	**371**
2	Sudėtinga	Disimpl-V	1003.56	1021.0	2683
		Gb-Disimpl-V	518.11	**511.0**	1547
		Libre	**431.55**	515.0	**1117**
3	Paprasta	Disimpl-V	1061.83	787.0	4740
		Gb-Disimpl-V	**751.25**	**720.0**	2694
		Libre	1009.72	959	**2113**
4	Sudėtinga	Disimpl-V	2598.91	2594.0	7354
		Gb-Disimpl-V	**1364.40**	**1283.0**	3723
		Libre	1449.18	1390	**3484**
5	Paprasta	Disimpl-V	10618.00	7334.0	58764
		Gb-Disimpl-V	**4579.24**	4615.0	**13825**
		Libre	5340.43	**4575**	16968
6	Sudėtinga	Disimpl-V	33985.20	29807.0	118482
		Gb-Disimpl-V	10700.20	10033.0	30759
		Libre	**8965.73**	**8436**	**23348**
7	Paprasta	Disimpl-V	11200.4	7252	48590
		Gb-Disimpl-V	**5997.37**	**4524**	**23126**
		Libre	17305.2	13343	65622
8	Sudėtinga	Disimpl-V	64751	42680	382593
		Gb-Disimpl-V	**28946**	**22416**	168067
		Libre	44000.4	36306	**154277**

Rezultatai


1e5
Direct
Directl
Disimpl-c
Gb-Disimpl-c
Disimpl-v
Gb-Disimpl-v
Libre
2.0
1.5
1.0
0.5
0.0
Didžiausias funkcijų įvertinimų skaičius
2
4
6
8
Klasė

Ginamieji teiginiai

1. Pasiūlytasis globalaus optimizavimo algoritmas LIBRE yra efektyvesnis tikslo funkcijos įvertinimų skaičiaus atžvilgiu nei kitos populiariausios alternatyvos blogiausio atvejo atžvilgiu, sprendžiant sunkius globalaus optimizavimo uždavinius.

Pasiūlytojo algoritmo modifikacijos

Hipotezė

- Griežtesnių surogatinių Lipšico rėžių naudojimas turėtų didinti Lipšico optimizavimo algoritmo efektyvumą tikslo funkcijų įvertinimų skaičiaus atžvilgiu, nes būtų naudojamas tikslesnis funkcijos modelis.

Lyginti surogatinių Lipšico apatinių rėžių įverčiai

Pirmasis LIBRE algoritmo kriterijus simpleksų parinkimui:

$$G(S_i, \widetilde{L}) = \min_{\boldsymbol{x} \in S_i} g(\boldsymbol{x}, S_i, \widetilde{L}). \tag{6}$$

Palyginti surogatinių Lipšico apatinių rėžių įverčiai

$$g(\boldsymbol{x}, S_i, \widetilde{L}) = f(\boldsymbol{v}_{min}(S_i)) - \widetilde{L}||\boldsymbol{x} - \boldsymbol{v}_{min}(S_i)||, \tag{7}$$

$$g(\boldsymbol{x}, S_i, \widetilde{L}) = f(\boldsymbol{v}_{max}(S_i)) - \widetilde{L}||\boldsymbol{x} - \boldsymbol{v}_{max}(S_i)||, \tag{8}$$

$$g(\boldsymbol{x}, S_i, \widetilde{L}) = \max_{\boldsymbol{v} \in V(S_i)} \{f(\boldsymbol{v}) - \widetilde{L}||\boldsymbol{v} - \boldsymbol{x}||\}, \tag{9}$$

Aproksimavimas simplekso briaunų Lipšico apatiniais rėžiais:

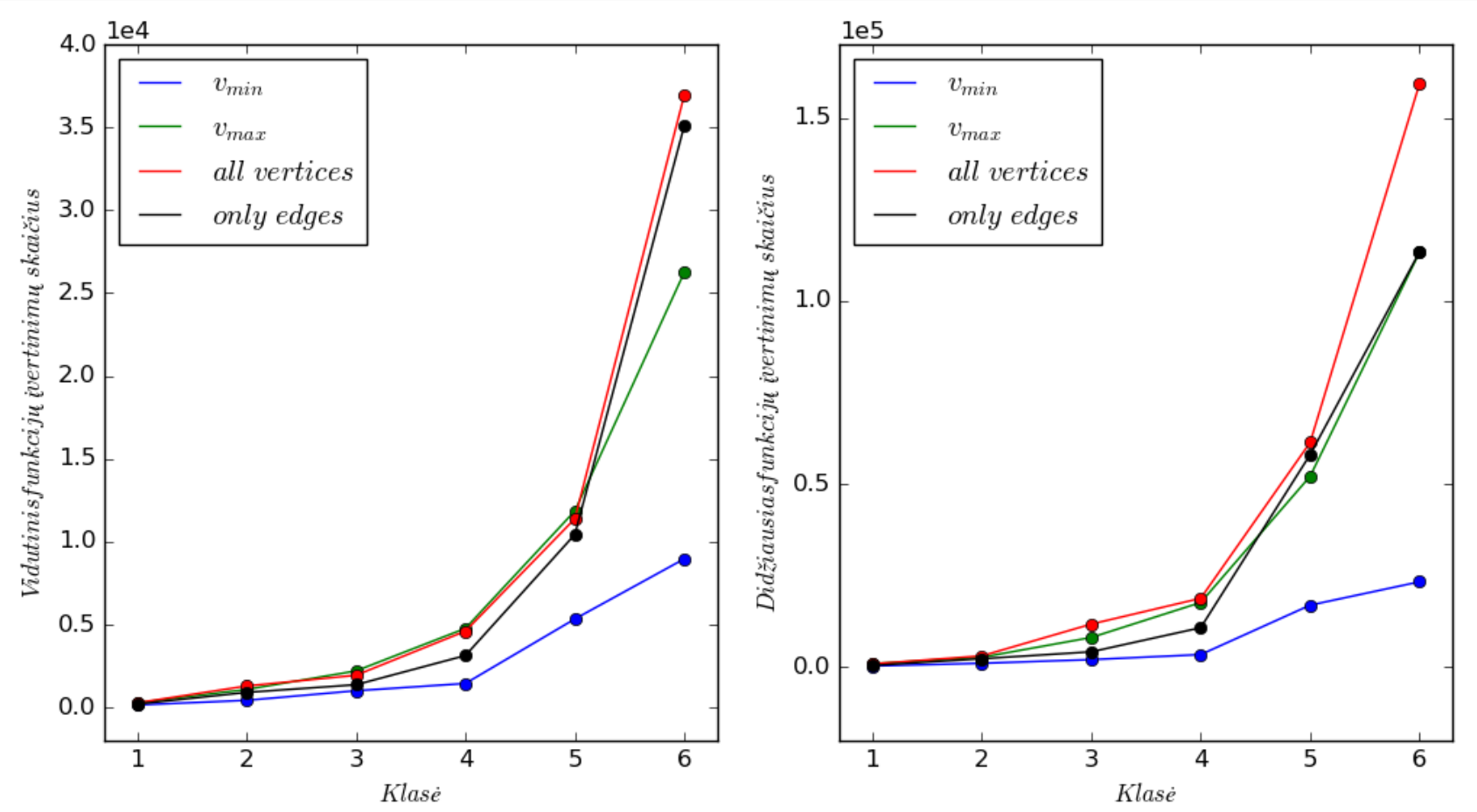
$$G^{edge}(\boldsymbol{v}_i, \boldsymbol{v}_j, \widehat{L}) = \frac{f(\boldsymbol{v}_i) + f(\boldsymbol{v}_j) - \widehat{L}||\boldsymbol{v}_i - \boldsymbol{v}_j||}{2}, \tag{10}$$

$$X^{edge}(\boldsymbol{v}_i, \boldsymbol{v}_j, \widehat{L}) = \boldsymbol{v}_i t + \boldsymbol{v}_j(1-t), \quad t = 0.5 + \frac{f(\boldsymbol{v}_j) - f(\boldsymbol{v}_i)}{2\widehat{L}||\boldsymbol{v}_j - \boldsymbol{v}_i||}. \tag{11}$$

Lyginti surogatinių Lipšico apatinių rėžių įverčiai

Klasė	Strategija	Vidurkis	Mediana	Daugiausia	Vid. sekundžių įvertinimui
1	1	**151.92**	**145**	**371**	**0.000039**
	2	283.92	247	1002	0.000063
	3	280.74	222	952	0.000349
	4	204.58	162	512	0.000156
2	1	**431.53**	**397**	**1117**	**0.000071**
	2	1085.13	947	2757	0.00024
	3	1297.6	1196	3133	0.000512
	4	911.78	764	2361	0.000284
3	1	**1009.82**	**957**	**2113**	**0.000323**
	2	2205.68	1886	8156	0.001023
	3	1942.33	1324	11803	0.004319
	4	1371.71	989	4238	0.001714
4	1	**1448.94**	**1386**	**3484**	**0.000517**
	2	4771.85	4157	17699	0.002511
	3	4605.42	3699	18843	0.007346
	4	3133.99	2632	10821	0.004411
5	1	**5339.45**	**4572**	**16968**	**0.012432**
	2	11855.8	9490	52130	0.061572
	3	11385	8784	61363	0.155959
	4	10435.2	7701	58035	0.173710
6	1	**8965.54**	**8422**	**23348**	**0.035610**
	2	26246.3	16769	113266	0.156995
	3	36930.1	28451	159262	0.582037
	4	35082.5	28524	113353	0.522561

Rezultatai


1e4
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
v_{min}
v_{max}
all vertices
only edges
Vidutinis funkcijų įvertinimų skaičius
1
2
3
4
5
6
Klasė
1e5
1.5
1.0
0.5
0.0
v_{min}
v_{max}
all vertices
only edges
Didžiausias funkcijų įvertinimų skaičius
1
2
3
4
5
6
Klasė

Ginamieji teiginiai

2. Tikslesnių surogatinių Lipšico rėžių naudojimas ne visada padidina DIRECT tipo optimizavimo algoritmo efektyvumą tikslo funkcijos įvertinimų skaičiaus atžvilgiu.

Išvados

Išvados I

- Eksperimentiškai parodyta, kad pasiūlytasis LIBRE algoritmas veikia geriausiai iš nagrinėtų alternatyvų blogiausio atvejo atžvilgiu, kai sprendžiamos sunkios optimizavimo problemos.
- LIBRE algoritmo blogiausio atvejo rezultatai nebuvo geriausi tik dviejų paprastų funkcijų klasių atveju iš 8.
- LIBRE algoritmo rezultatai buvo geresni nei originalaus DISIMPL-V algoritmo 7 iš 8 klasių tiek blogiausiu, tiek vidutiniu atveju.

Išvados II

- Eksperimentiškai nustatyta, kad efektyviausia surogatinių Lipšico rėžių sudarymo strategija pasiūlytajam algoritmui yra naudoti tik vieną viršūnę, kurioje yra mažiausia funkcijos reikšmė.
- Surogatinių Lipšico rėžių sudarymo strategija, kurioje naudojama viršūnė su mažiausia funkcijos reikšme, yra mažiausiai griežta iš visų nagrinėtų, todėl griežtesnių surogatinių Lipšico rėžių naudojimas ne būtinai padidina DIRECT-tipo optimizavimo algoritmo efektyvumą.

Doktoranto publikacijos disertacijos tema I

Publikacijos periodiniuose recenzuojamuose leidiniuose:

- Gimbutas, A. (2016). Globalios optimizacijos algoritmas, naudojantis lokalų Lipšico konstantos įvertį. *Jaunųjų mokslininkų darbai*, Vol. 1, No. 45, pp. 47-53. doi:10.21277/jmd.v1i45.44
- Gimbutas, A., and Žilinskas, A. (2017). An algorithm of simplicial Lipschitz optimization with the bi-criteria selection of simplices for the bi-section. *Journal of Global Optimization*, pp. 1-13. doi:10.1007/s10898-017-0550-9

Doktoranto publikacijos disertacijos tema II

Publikacijos recenzuojamose konferencijų leidiniuose:

- Gimbutas, A. and Žilinskas, A. (2016). On global optimization using an estimate of Lipschitz constant and simplicial partition. In: *Numerical Computations: Theory and Algorithms*. Calabria: AIP Publishing, Vol. 1776, No. 1, p.060012. doi:10.1063/1.4965346
- Gimbutas, A. and Žilinskas, A. (2018). Generalization of Lipschitzian global optimization algorithms to the multi-objective case. In: *International Workshop on Optimization and Learning: Challenges and Applications*. Alicante, pp.36-37.

Doktoranto pranešimai disertacijos tema

Pranešimai konferencijose:

- Gimbutas, A. (2014). One-step worst-case optimal bivariate algorithm for bi-objective optimization. *Data Analysis Methods for Software Systems*. Druskininkai.
- Gimbutas, A. (2015). Daugiadimensinis globalios optimizacijos algoritmas naudojantis adaptyviąją Lipšico konstantą. In: *Computer Days*. Panevėžys.
- Gimbutas, A. (2015). DAKIS - algoritmų įvertinimo ir palyginimo įrankis. In: *Operacijų tyrimas ir taikymai*. Panevėžys.
- Gimbutas, A. (2015). Multicriteria Lipschitz Optimization Algorithm Using Local Lipschitz Constant Estimate. *Data Analysis Methods for Software Systems*. Druskininkai, p. 20.
- Gimbutas, A. (2016). Daugiakriterė globali optimizacija naudojant adaptyvią Lipšico konstantą. In: *Operacijų tyrimas ir taikymai*. Kaunas.
- Gimbutas, A. and Žilinskas (2016). Remarks on a Multi-Criteria Simplicial Optimization with an Estimate of Lipschitz constant. *Data Analysis Methods for Software Systems*. Druskininkai, p. 22.
- Gimbutas, A. (2017). Daugiaobjektinis Lipšico simpleksinis optimizavimas su Lipšico konstantos įvertinimu. In: *Computer Days*. Kaunas, p. 25.

Literatūra

- Marco Gaviano, Dmitri E Kvasov, Daniela Lera, and Yaroslav D Sergeyev. "Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization". In: *ACM Trans- actions on Mathematical Software (TOMS)* 29.4 (2003), pages 469–480.
- Donald R Jones, Cary D Perttunen, and Bruce E Stuckman. "Lipschitzian optimization without the Lipschitz constant". In: *Journal of Optimization Theory and Applications* 79.1 (1993), pages 157–181.
- Remigijus Paulavičius, Yaroslav D Sergeyev, Dmitri E Kvasov, and Julius Žilinskas. "Globally-biased Disimpl algorithm for expensive global optimization". In: *Journal of Global Optimization* 59.2-3 (2014), pages 545–567.
- Remigijus Paulavičius and Julius Žilinskas. *Simplicial global optimization*. Springer, 2014.
- SA Pijavskij. "An algorithm for finding the global extremum of function". In: *Optimal Decisions* 2 (1967), pages 13–24.
- Antanas Žilinskas. "A one-step worst-case optimal algorithm for bi-objective univariate optimization". In: *Optimization Letters* 8.7 (2014), pages 1945– 1960.
- Antanas Žilinskas and Julius Žilinskas. "Adaptation of a one-step worst- case optimal univariate algorithm of bi-objective Lipschitz optimization to multidimensional problems". In: *Communications in Nonlinear Science and Numerical Simulation* 21.1-3 (2015), pages 89–98.

Ačiū už dėmesį