KAUNO TECHNOLOGIJOS UNIVERSITETAS

Julius Žilinskas

PADENGIMO METODAI JUODOSIOS DĖŽĖS GLOBALIAI OPTIMIZACIJAI IR JŲ LYGIAGRETINIMAS

Daktaro disertacijos santrauka

Technologijos mokslai, informatikos inžinerija (07T)

KAUNAS 2002

Darbas atliktas 1998-2002 metais Kauno technologijos universitete.

Doktorantūros teisė suteikta Lietuvos Respublikos Vyriausybės 1998 04 14 nutarimu Nr. 457.

Doktorantūros komitetas:

pirmininkas ir darbo vadovas
prof. habil. dr. (technologijos mokslai, informatikos inžinerija, 07T)
Rimantas ŠEINAUSKAS (Kauno technologijos universitetas);

nariai:
prof. habil. dr. (technologijos mokslai, informatikos inžinerija, 07T)
Vytautas KAMINSKAS (Vytauto Didžiojo universitetas),

doc. dr. (technologijos mokslai, informatikos inžinerija, 07T)
Egidijus KAZANAVIČIUS (Kauno technologijos universitetas),

prof. dr. (technologijos mokslai, informatikos inžinerija, 07T)
Kaj MADSEN (Danijos technikos universitetas),

prof. habil. dr. (fiziniai mokslai, informatika, 09P)
Vydūnas ŠALTENIS (Matematikos ir informatikos institutas).

Oponentai:
prof. habil. dr. (fiziniai mokslai, informatika, 09P)
Raimondas Čiegis (Vilniaus Gedimino technikos universitetas),

prof. habil. dr. (technologijos mokslai, informatikos inžinerija, 07T)
Gintautas Dzemyda (Matematikos ir informatikos institutas).

Disertacija bus ginama viešame doktorantūros komiteto posėdyje, kuris įvyks 2002 m. birželio 28 d. 11 val. Kauno technologijos universiteto Informatikos fakulteto 407a auditorijoje.
Adresas: Studentų 50, LT-3031 Kaunas, Lietuva.
tel. +370 37 300394.

Disertacijos santrauka išsiųsta 2002 m. gegužės 28 d.

Su disertacija galima susipažinti Kauno technologijos universiteto bibliotekoje.

Problemos aktualumas

Daug inžinerijos, fizikos, ekonomikos ir kitų sričių uždavinių gali būti sprendžiami kaip globalios optimizacijos uždaviniai. Matematiškai uždavinys formuluojamas taip:

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

čia $f(x)$ yra netiesinė tolydžiųjų kintamųjų funkcija f: $\Re^n \to \Re$; $D \subseteq \Re^n$ – leistinoji sritis; n – kintamųjų skaičius. Be globalaus minimumo f^*, turi būti surastas vienas arba visi globalaus minimumo taškai x^*: $f(x^*)=f^*$.

Globalios optimizacijos uždaviniai yra sudėtingi algoritmų sudėtingumo teorijos prasme. Praktiniams uždaviniams spręsti reikia atlikti daug skaičiavimų.

Visada yra didelių, šiuolaikiniais kompiuteriais neišsprendžiamų, praktinių uždavinių. Kai įprastų kompiuterių skaičiavimo pajėgumo neužtenka, gali padėti galingi lygiagretieji kompiuteriai. Algoritmai, turintys lygiagrečiąsias versijas, gali būti plačiau taikomi – jais galima išspręsti didesnius praktinius uždavinius. Dėl to naujų algoritmų lygiagrečiųjų versijų realizavimas ir įvertinimas yra viena iš tyrimo dalių.

Darbo tikslai ir uždaviniai

Šio darbo tikslai:

- Sudaryti naujus juodosios dėžės globalios optimizacijos algoritmus, pagrįstus padengimo metodais.
- Realizuoti pasiūlytuosius algoritmus ir eksperimentais įvertinti jų pajėgumą ir patikimumą.
- Realizuoti efektyvias lygiagrečiąsias pasiūlytų algoritmų versijas.
- Apibrėžti sukurtųjų algoritmų racionalaus taikymo sritis.

Suformuluotų tikslų siekiama sprendžiant tokius uždavinius:

- Išanalizuoti padengimo metodų privalumus ir trūkumus globaliojoje optimizacijoje.
- Padidinti padengimo algoritmų pajėgumą ir pritaikyti juos juodosios dėžės situacijai.
- Realizuoti nuosekliąsias pasiūlytų algoritmų versijas.
- Eksperimentiškai įvertinti realizuotuosius algoritmus.
- Išanalizuoti lygiagrečiųjų skaičiavimų modelius.
- Išanalizuoti lygiagrečiuosius skaičiavimus padengimo globalios optimizacijos algoritmų realizavimo požiūriu.
- Sukurti programinę įrangą realizuojančią lygiagrečiąsias pasiūlytųjų algoritmų versijas ir įvertinti lygiagretinimo efektyvumą.
- Pritaikyti pasiūlytus algoritmus tipiniams praktiniams uždaviniams spręsti ir įvertinti jų taikymo perspektyvas.

Mokslinis naujumas

- Padengimo globalios optimizacijos metodų modifikavimas gerinant jų pajėgumą ir pritaikant juos juodosios dėžės globaliai optimizacijai.
- Naujų algoritmų nuosekliųjų ir lygiagrečiųjų versijų realizavimas ir tyrimas.

Darbo aprobavimas

Disertacijos darbo rezultatai pristatyti ir aptarti šiose mokslinėse konferencijose ir seminaruose:

- Tarptautinėje konferencijoje „International Workshop on Global Optimization 1999", Florencijos universitetas, Firenze, 1999.
- Konferencijoje „Informacinės technologijos'2000", KTU, Kaunas, 2000.
- Tarptautinėje konferencijoje „Second International Conference Simulation, Gaming, Training, Business Process Reeingineering in Operations", RTU, Riga, 2000.
- Edinburgo universiteto optimizavimo seminare, Edinburgh, 2001.
- Tarptautinėje konferencijoje „TRACS-ACCESS-MINOS User Group Meeting," Edinburgh, 2001.
- Tarptautinėje doktorantų vasaros mokykloje „BORNHOLM'02 Nordic Summer Course on Applied Optimization and Modeling", Bornholm, 2002.

Publikacijos

Disertacijos tema paskelbti moksliniai straipsniai:

1. **O. Tingleff, A. Žilinskas, J. Žilinskas**. A Two Dimensional Optimization Algorithm Based on New Statistical Model of Multimodal Functions. *C. Carlsson et al. (eds.), Global & Multiple Criteria Optimization and Information System Quality*, Åbo Akademi Press, Finland, 103-112, 1998.
2. **J. Žilinskas**. Optimization of Lipschitzian functions by simplex based branch and bound. *Information Technology and Control No.1(14)*, Kaunas, Technologija, 45-50, 2000.
3. **K. Madsen, J. Žilinskas**. Testing Real and Interval Methods for Global Optimization. *Technical report IMM-Report-2000-05*, Department of Mathematical Modelling, DTU, Denmark, 22 pages, 2000.
4. **K. Madsen, J. Žilinskas**. Evaluating Performance of Attraction Based Subdivision Methods for Global Optimization. *Second International Conference "Simulation, Gaming, Training, Business Process Reeingineering in Operations"*, RTU, Riga, 38-42, 2000.
5. **J. Žilinskas**. Black Box Global Optimization Inspired by Interval Methods. *Information Technology and Control No.4(21)*, Kaunas, Technologija, 53-60, 2001.

Disertacijos struktūra

Disertaciją sudaro įvadas, keturi skyriai, išvados ir literatūros sąrašas. Aiškinamajame rašte – 87 puslapiai. Literatūros sąraše – 88 nuorodos.

Disertacijos turinys

Įvadas

Įvade nurodytas problemos aktualumas, suformuluoti darbo tikslai ir iškelti uždaviniai, parodytas mokslinis naujumas, pateiktas pranešimų ir publikacijų sąrašas.

1. Globali optimizacija ir lygiagretieji skaičiavimai

Pirmajame skyriuje apžvelgti globalios optimizacijos metodai. Daugiausia dėmesio skiriama padengimo metodams realizuotiems pagal šakų ir rėžių metodologiją. Taip pat apžvelgti lygiagretieji skaičiavimai, nurodytos užduočių paskirstymo ir baigties nustatymo problemos. Apžvelgta literatūra apie lygiagrečiuosius šakų ir rėžių algoritmus. Aprašyti lygiagrečiųjų algoritmų įvertinimo kriterijai.

1.1. Globali optimizaciją

Matematiškai globalios optimizacijos uždavinys formuluojamas taip:

$$f^* = \min_{x \in D} f(x);$$

čia $f(x)$ yra netiesinė tolydinių kintamųjų funkcija f: $\Re^n \to \Re$, vadinama *tikslo funkcija*; $D \subseteq \Re^n$ – *leistinoji sritis*, n – *kintamųjų skaičius*. Be globalaus minimumo f^*, turi būti surastas vienas arba visi globalaus minimumo taškai x^*: $f(x^*)=f^*$. Globalios optimizacijos uždavinyje nedaroma prielaida, kad funkcija yra unimodali, t.y. turinti tik vieną minimumą.

Kartais tikslo funkcija gali būti išreikšta analitiškai, ar netgi analitiškai išspredžiamas optimizacijos uždavinys. Tačiau praktinių uždavinių tikslo funkcijos reikšmės paprastai būna apskaičiuojamos kompiuterinėmis programomis, o tikslo funkcijos savybes nustatyti sunku. Tokiu atveju laikoma, kad tikslo funkcijos reikšmes suformuoja *juodoji dėžė* – tikslo funkcijos kintamųjų (argumentų) reikšmės patenka į juodosios dėžės įėjimus, o jas naudodama juodoji dėžė suformuoja išėjimo reikšmę, kuri yra tikslo funkcijos reikšmė esant duotoms funkcijos kintamųjų reikšmėms.

Globalios optimizacijos uždaviniai yra sudėtingi algoritmų sudėtingumo teorijos prasme. Praktiniams uždaviniams spręsti reikia atlikti daug skaičiavimų. Sprendimo trukmė labai priklauso nuo uždavinio dimensijos.

Globalios optimizacijos metodų klasifikavimas:

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

Metodai, aproksimuojantys tikslo funkciją.

Darbe tiriami padengimo metodai. Esant tam tikroms prielaidoms, šiais metodais teoriškai garantuotu tikslumu sprendžiami globalios optimizacijos uždaviniai. Padengimo metodais nustatomos sritys, kuriose negali būti globalaus minimumo, ir jos pašalinamos iš tolesnės paieškos. Posričiai baigiami dalyti, kai globalaus minimumo taškai lokalizuojami pakankamai mažuose leistinosios srities posričiuose.

Ieškant šalintinų sričių reikia įvertinti funkcijos rėžius. Visos funkcijos reikšmės srityje yra didesnės už funkcijos apatinį rėžį šioje srityje, todėl jei funkcijos apatinis rėžis srityje yra didesnis už jau žinomą funkcijos reikšmę, tai šioje srityje globalaus minimumo taško nėra. Apatinis tikslo funkcijos rėžis gali būti nustatytas naudojant išgaubtus funkcijos apvalkalus. Naudojant Lipschitzo optimizaciją, laikoma, kad tikslo funkcija yra aprėžto nuožulnumo. Intervalų metoduose tikslo funkcijos rėžiai įvertinami naudojant intervalų aritmetiką. Šakų ir rėžių algoritmai gali būti naudojami posričių sąrašo ir šalinimo bei dalijimo procesui valdyti.

Lipschitzo optimizacija pagrįsta prielaida, kad tikslo funkcija yra aprėžto nuožulnumo. Funkcija f: $D\to\Re$, $D\subseteq\Re^n$ yra laikoma Lipschitzo funkcija, jei ji tenkina sąlygą
$|f(x)-f(y)|\le L\|x-y\|$, $\forall x\in D$, $\forall y\in D$;
čia $L>0$ yra konstanta, vadinama Lipschitzo konstanta; D yra glaudi; $\|\ \|$ žymi Euklido normą. Gali būti naudojamos ir kitokios normos.

Lipschitzo globali optimizacija gali būti taikoma, kai yra žinoma tikslo funkcijos Lipschitzo konstanta. Yra efektyvių vienmačių Lipschitzo optimizacijos algoritmų. Skaičiuoti tikslo funkcijos apatinį rėžį, naudojant Lipschitzo funkcijos sąlygą, vienmačiu atveju yra paprasta. Tačiau daugiamačiu atveju tikslaus apatinio tikslo funkcijos rėžio skaičiavimai yra brangūs, o kartu ir gaišlūs. Be to, Lipschitzo konstanta priklauso nuo kintamųjų mastelių, dėl to optimizacija gali būti neefektyvi, kai kintamųjų masteliai nesuderinti. Kai Lipschitzo konstantos įvertis yra per mažas, globalaus minimumo taškas gali būti neaptiktas. Tačiau, kai Lipschitzo konstantos įvertis yra per didelis, apskaičiuoti rėžiai nėra tikslūs, o optimizacija lėta. Dar vienas Lipschitzo optimizacijos trūkumas yra tas, kad neįvertinama, jog Lipschitzo konstanta gali labai skirtis įvairiose leistinosios srities dalyse. Praktinių uždavinių tikslo funkcijų, išreikštų kompiuterinėmis programomis, Lipschitzo konstanta paprastai nežinoma.

Intervaliniai globalios optimizacijos metodai yra pagrįsti intervalų aritmetika. Intervalų aritmetika operuoja intervalais $X=[\underline{x},\overline{x}]=\{x\in\Re\mid \underline{x}\le x\le\overline{x}\}$, kurių rėžiai $\underline{x}$ ir $\overline{x}$ yra apibrėžtos realiais skaičiais. Kiekviena realių skaičių aritmetikos operacija x op y atitinka intervalų aritmetikos operaciją X op Y:
$$X\ \mathrm{op}\ Y=[\min_{x\in X,\,y\in Y} x\,\mathrm{op}\,y,\ \max_{x\in X,\,y\in Y} x\,\mathrm{op}\,y]=\{x\,\mathrm{op}\,y\mid x\in X, y\in Y\}.$$
Intervalų aritmetikos operacijos X op Y rezultatas yra intervalas, kuriam priklauso visos galimos realių skaičių aritmetikos operacijos x op y rezultato reikšmes, kai $x\in X$ ir $y\in Y$.

Jei funkcija $f(x)$ yra apibrėžta srityje $X = \{x \in \Re^n \mid \underline{x}_i \le x_i \le \overline{x}_i\}$, tai jos reikšmių apatinis ir viršutinis rėžiai šioje srityje gali būti įvertinti funkcijos išraiškoje pakeičiant realių skaičių aritmetikos operacijas su realiais skaičiais x_i intervalų aritmetikos operacijomis su intervalais $[\underline{x}_i, \overline{x}_i]$. Įvertinti rėžiai gali būti naudojami nustatyti, ar srityje nėra globalaus minimumo taško. Tokios sritys gali būti pašalinamos iš tolesnės paieškos. Jei tikslo funkcija yra diferencijuojama, galima įvertinti jos išvestinių intervalus ir pašalinti sritis, kuriose tikslo funkcija yra monotoninė. Jei tikslo funkcija yra dukart tolydžiai diferencijuojama, galima įvertinti antrųjų išvestinių rėžius ir pašalinti sritis, kuriose tikslo funkcija yra įgaubta. Jei tikslo funkcija yra dukart tolydžiai diferencijuojama, specialus intervalų Newtono metodas gali būti naudojamas sritims mažiniti ir sritims, kuriose nėra stacionarių taškų, nustatyti ir pašalinti.

Intervalų metodų trūkumas yra tas, kad tikslo funkcija turi būti užrašyta matematine formule arba specialia algoritmine kalba, t.y. intervalų algoritmai negali būti taikomi juodosios dėžės tipo funkcijoms minimizuoti. Kitas intervalų metodų trūkumas yra priklausomybės problema: kiekvienas to paties kintamojo panaudojamas intervalinėje išraiškoje nepagrįstai traktuojamas kaip naujas kintamasis. Todėl gaunami pernelyg platūs funkcijos reikšmių rėžiai, sąlygojantys lėtą optimizavimą. Į tai galima atsižvelgti, kai intervaliniai metodai taikomi analitiškai apibrėžtos funkcijos rėžių skaičiavimui, bet to sunku išvengti, kai funkcija yra aprašyta algoritmu.

Padengimo globalios optimizacijos metodai gali būti realizuojami pagal šakų ir rėžių metodiką. Šakų ir rėžių algoritmus sudaro inicializavimo, išrinkimo bei dalinimo taisyklės. Inicializavimo etape leistinoji sritis padengiama nurodytos formos posričiais. Toliau vykdomas ciklas:

- iš kandidatų aibės C išrenkamas ir padalijamas posritis,
- apskaičiuojami funkcijos minimumo rėžiai naujai gautuose posričiuose,
- posričiai, kuriuose minimumo taškas negali buti, pašalinami,
- nepašalinti posričiai įtraukiami į kandidatų aibę.

Algoritmu siekiama, kad kandidatų aibė C greitai mažėtų ir konverguotų į X^*. Bendras šakų ir rėžių algoritmas pateiktas 1 pav. Algoritme $LB(X)$ ir $UB(X)$ žymi atitinkamai viršutinį ir apatinį tikslo funkcijos minimalios reikšmės posrityje X rėžį: $LB(X) \le \min_{x \in X} f(x)$, $UB(X) \ge \min_{x \in X} f(x)$.

$UB(X) \Leftarrow \infty$.
Padengiamas $D: C = \{C_j \mid j = 1, \ldots, m\}$, $D \subseteq \cup C_j$.
While $C \neq \emptyset$,
    Išrenkamas $B \in C$, $C \Leftarrow C \setminus \{B\}$.
    Padalijamas $B: B \subseteq \cup T_j$, $T = \{T_j \mid T_j \cap D \neq \emptyset\}$.
    $UB(D) \Leftarrow \min(\{UB(D)\} \cup \{UB(T_j \cap D) \mid j = 1, \ldots, p\})$.
    $C \Leftarrow \{B: C \cup T \mid LB(B) < UB(D) + \varepsilon\}$.

1 pav. Bendras šakų ir rėžių algoritmas

Padengimo, išrinkimo, padalijimo ir rėžių skaičiavimo taisyklės priklauso nuo konkretaus algoritmo. Padengimo ir padalijimo taisyklės priklauso nuo naudojamų posričių formos. Gali būti naudojami hiperstačiakampiai, simpleksai, hiperkūgiai ar hipersferos. Dažniausiai bendros globalios optimizacijos uždavinių leistinosios sritys yra hiperstačiakampiai. Visi intervalų ir dauguma Lipschitzo globalios optimizacijos šakų ir rėžių algoritmai naudoja hiperstačiakampius posričius.

Naudojama viena iš trijų išrinkimo taisyklių:

- geriausiojo išrinkimo – išrenkamas *C* elementas su geriausiu įverčiu (su mažiausiu apatiniu tikslo funkcijos rėžiu);
- gilyn – išrenkamas jauniausias *C* elementas;
- platyn – išrenkamas vyriausias *C* elementas.

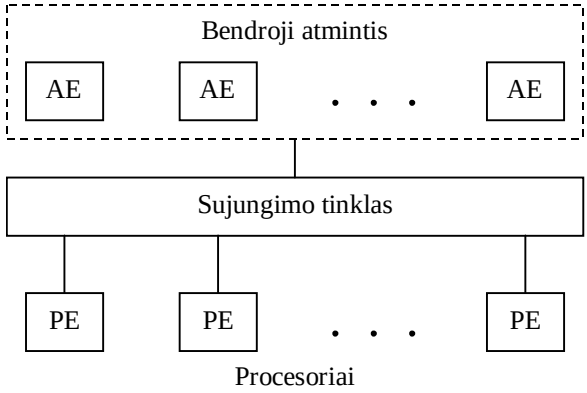
Rėžių skaičiavimo taisyklės nusako, kaip skaičiuojami tikslo funkcijos minimumo rėžiai. Geriausia rasta tikslo funkcijos reikšmė dažniausiai naudojama kaip viršutinis tikslo funkcijos minimumo rėžis. Apatinis rėžis įvertinamas naudojant išgaubtus funkcijos apvalkalus, Lipschitzo sąlygą arba intervalų aritmetiką.

Pagrindiniai globalios optimizacijos algoritmų įvertinimo kriterijai yra tikslo funkcijos (kartais ir gradiento funkcijos) skaičiavimų kiekis ir sprendimo trukmė. Kai tikslo funkcijos skaičiavimas trunka ilgiau negu pagalbiniai optimizavimo skaičiavimai, t.y. kai tikslo funkcija yra „brangi”, abu kriterijai praktiškai ekvivalentūs, ir vartojamas paprastesnis tikslo funkcijos skaičiavimų kiekis. Priešingu atveju vartotinas sprendimo trukmės kriterijus. Paprastai praktinių uždavinių tikslo funkcijos yra brangios. Šiame darbe funkcijų skaičiavimų kiekio kriterijus naudojamas nuosekliesiems globalios optimizacijos algoritmams vertinti.

1.2. Lygiagretieji kompiuteriai

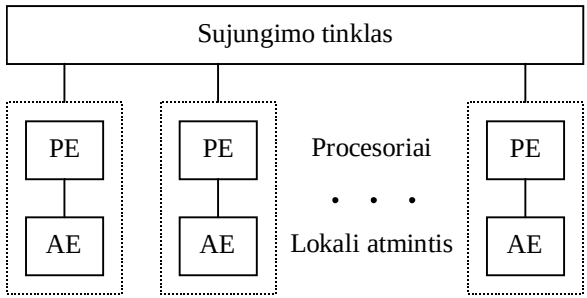
Kai įprastų nuosekliųjų kompiuterių pajėgumo praktiniam uždaviniui išspręsti neužtenka, vilčių teikia lygiagretieji kompiuteriai. Juose tą patį uždavinį tuo pačiu metu gali spręsti keli procesoriai, todėl sutrumpėja bendra sprendimo trukmė. Lygiagretieji kompiuteriai yra skirstomi į bendrosios atminties ir paskirstytosios atminties lygiagrečiuosius kompiuterius.

Bendrosios atminties lygiagretieji kompiuteriai turi vieną bendrą atminties bloką ir visi procesoriai gali tiesiogiai pasiekti visas atminties vietas (2 pav.). Procesoriai bendrauja tarpusavyje per bendrąsias duomenų struktūras, esančias bendrojoje atmintyje.



2 pav. Lygiagretusis bendrosios atminties kompiuteris

Paskirstytosios atminties lygiagrečiajame kompiuteryje kiekvienas procesorius gali tiesiogiai perskaityti ir įrašyti tik jo lokalioje atmintyje esančius duomenis (3 pav.). Procesoriai bendrauja tarpusavyje, siųsdami vienas kitam pranešimus. Duomenų persiuntimu turi pasirūpinti programuotojas.



3 pav. Lygiagretusis paskirstytosios atminties kompiuteris

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 abiejų tipų lygiagrečiaisiais kompiuteriais. Standartinė pranešimų persiuntimo biblioteka MPI (Message Passing Interface) naudojama pranešimų persiuntimui.

1.3. 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-vergai*. Procesorius šeimininkas saugo užduočių rinkinį, kuris vadinamas *darbo fondu* arba *užduočių eile*. Kai procesorius vergas neturi darbo, jis prašo užduoties. Procesorius šeimininkas išsiunčia sudėtingiausią arba svarbiausią turimą užduotį. Baigęs užduotį, procesorius vergas išiunčia rezultatą šeimininkui. Procesorius šeimininkas nustato, kada skaičiavimai yra baigti. Šios schemos trūkumas yra tas, kad procesorius šeimininkas gali nespėti aptarnauti vergų, ypač kai procesorių yra daug, o bendraujama dažnai. Tokiu atveju procesoriai vergai 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.

Lygiagretieji šakų ir rėžių algoritmai realizuojami panaudojant vieną iš trijų paminėtų užduočių paskirstymo schemų.

1.4. Lygiagrečiųjų algoritmų vertinimo koeficientai

Lygiagretieji algoritmai analizuojami naudojant lygiagretinimo koeficientus. Dažniausiai naudojamas lygiagrečiojo algoritmo *spartinimo koeficientas*:

$$s_m = \frac{t_1}{t_m};$$

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

$$e_m = \frac{s_m}{m}.$$

Paprastai $1 \le s_m \le m$ ir $0 \le e_m \le 1$. 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 $s_m<1$; peržiūrima daugiau posričių negu vieno procesoriaus atveju;
- lėtinimo anomaliją $s_{m_1} > s_{m_2}$, kai $m_1<m_2$; didėjant procesorių skaičiui didėja peržiūrimų posričių skaičius;
- greitinimo anomaliją $s_m>m$; peržiūrima mažiau posričių, negu vieno procesoriaus atveju.

Esant anomalijoms, spartinimo ir efektyvumo koeficientai netinka algoritmams vertinti. Anomalijos mažiau įtakoja *pseudoefektyvumo koeficientą*

$$pe_m = \frac{t_1 / |T_1|}{m \times t_m / |T_m|},$$

kur $|T_m|$ yra darbo kiekio matas naudojant m procesorių, ir *laiko panaudojimo skaičiavimams proporcijos koeficientą*. Geras darbo kiekio matas turi būti naudojamas, kai naudojamas pseudoefektyvumo 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.

2. Lipschitzo globali optimizacija su simpleksiniais posričiais

Šiame skyriuje pristatytas naujas Lipschitzo globalios optimizacijos algoritmas su simpleksiniais posričiais. Nuoseklioji algoritmo versija yra palyginta su kitais Lipschitzo šakų ir rėžių globalios optimizacijos algoritmais pagal jų eksperimentinio testavimo rezultatus. Aprašytas algoritmo lygiagretinimas. Lygiagrečiosios algoritmo realizacijos yra palygintos naudojant lygiagretinimo koeficientus.

Aptartas šitoks globalios optimizacijos uždavinys: reikia rasti tokį $x_{\min} \in X \subseteq \Re^n$, kad būtų $f_{\min} = f(x_{\min}) \le \min_{x \in X} f(x) + \varepsilon$; čia ε yra maža teigiama konstanta. Laikoma, kad tikslo funkcija *f*, apibrėžta leistinoje srityje $X \subseteq \Re^n$, tenkina Lipschitzo sąlygą, t.y.
$|f(x)-f(y)| \le L\|x-y\|$, $\forall x \in D$, $\forall y \in D$,
čia *L* yra Lipschitzo konstanta, $\| \ \|$ žymi Euklido normą.

Dauguma paskelbtų Lipschitzo globalios optimizacijos šakų ir rėžių algoritmų naudoja daugiamačius stačiakampius posričius. Jeigu vertinant funkcijos apatinį rėžį tokiuose posričiuose naudojamos funkcijos reikšmės viršūnėse, tai funkcijų skaičiavimų kiekis proporcingas viršūnių skaičiui – 2^n, auga eksponentiškai *n* atžvilgiu.

2.1. Simpleksiniai posričiai

Simpleksas yra iškilus *n*-matis daugiakampis, turintis *n*+1 viršūnę. Vienmatėje erdvėje simpleksas yra atkarpa, dvimatėje – trikampis, trimatėje – tetraedras. Simpleksas yra daugiasienis, *n*-matėje erdvėje turintis mažiausiai viršūnių. Jeigu vertinant funkcijos apatinį rėžį naudojamos funkcijos reikšmės viršūnėse, tai simpleksas yra tinkamiausia posričių forma. Stengiantis pagerinti Lipschitzo globalios optimizacijos algoritmų pajėgumą, pasiūlytas Lipschitzo globalios optimizacijos šakų ir rėžių algoritmas, naudojantis netaisyklingų simpleksų posričius.

Dažniausiai Lipschitzo optimizacijos uždavinių leistinoji sritis yra daugiamatis stačiakampis. Mūsų algoritmų inicializacijai reikia padengti daugiamatį stačiakimpį pradiniais simpleksiniais posričiais. Mes siūlome dvi pradinio padengimo strategijas: srities perteklinis uždengimas vienu dideliu simpleksu arba viršūnių trianguliacija – padalinimas simpleksais, kurių viršūnės taip pat yra ir daugiamačio stačiakampio viršūnės.

Simpleksiniai posričiai dalijami į mažesnius simpleksus. Pasiūlytame algoritme naudojamas simplekso padalijimas į du per ilgiausios briaunos vidurį arba į kelis per visų briaunų vidurius.

2.2. Lipschitzo funkcijos rėžiai daugiasienyje

Šakų ir rėžių globalios optimizacijos algoritmo vykdymo sparta labai priklauso nuo rėžių skaičiavimo. Jeigu įvertinamas tikslus apatinis tikslo funkcijos rėžis, neperspektyvūs posričiai pašalinami anksčiau, ir bendras tikslo funkcijos skaičiavimų kiekis sumažėja. Tačiau skaičiavimai tiksliems rėžiams įvertinti gali būti gaišlūs. Gero algoritmo rėžių įvertinimas turi būti gana paprastas, bet rėžiai pakankamai tikslūs.

Bendras funkcijos *f* minimumo viršutinio *UB*(*I*) ir apatinio *LB*(*I*) rėžio daugiasienyje *I* įvertinimo algoritmas parodytas 4 pav. Pirmiausia daugiasienyje parenkami taškai ir juose apskaičiuojamos funkcijos reikšmės. Viršutinis rėžis *UB*(*I*) yra tiesiog mažiausia iš šių funkcijos reikšmių. Apatiniam rėžiui *LB*(*I*) įvertinti, naudojantis apskaičiuotomis funkcijos reikšmėmis ir Lipshitzo sąlyga, sudaroma apatinių rėžių funkcija *F*, kurios minimumas yra apatinio rėžio įvertis.

Parenkama baigtinė taškų daugiasienyje *I* aibė *D*.
$UB(I) = \min_{x \in D} f(x)$.
Sudaroma apatinių rėžių funkcija *F*.
Apskaičiuojamas $LB(I) = \min_{x \in I} F(x)$.

4 pav. Bendras funkcijos minimumo rėžių daugiasienyje įvertinimo algoritmas

Apatinių rėžių funkcija yra viršutinis kūgių, turinčių lygiagrečias simetrijos ašis, apvalkalas. Jei $n>1$, ($I \subseteq \Re^n$) ir atsižvelgiama daugiau negu į vieną tašką aibėje *D*, tokios apatinių rėžių funkcijos minimumo paieška yra gaišli. Kai atsižvelgiama tik į vieną tašką, apatinis rėžis apskaičiuojamas gana paprastai:

$$LB(I) = f(x_o) - L \max_{x \in I} \|x - x_o\| .$$

Tereikia nustatyti atstumą nuo pasirinktojo taško iki labiausiai nutolusio daugiasienio taško, kuris bus vienoje iš viršūnių:

$$LB(I) = f(x_o) - L \max \|x_v - x_o\| .$$

Tiksliausias apatinis rėžis būtų įvertinamas, jei atstumas nuo pasirinkto taško iki labiausiai nutolusio daugiasienio taško būtų minimalus:

$$x_o = \arg\min_{x_t \in I} \left(\max \|x_v - x_t\| \right).$$

Jei daugiasienis būtų simpleksas ir gaubiančiosios daugiamatės sferos centras būtų simplekso viduje, tai toks taškas x_0 būtų šios sferos centras. Jį galima rasti sprendžiant tiesinių lygčių sistemą.

Funkcijos reikšmės daugiasienio viršūnėse gali būti panaudotos funkcijos reikšmių rėžiams gretimose ir palikuoniniuose daugiasieniuose įvertinti. Tačiau jei $n>1$ ir įvertinama daugiau negu viena viršūnė, apatinių rėžių funkcijos minimumo paieška yra gaišli. Skaičiavimai supaprastėja, kai įvertinama tik viena viršūnė. Tokiu atveju apatinis rėžis įvertinamas naudojant tikslo funkcijos reikšmę šioje viršūnėje ir ilgiausios briaunos iš šios viršūnės ilgį. Jei toks rėžis yra per mažai tikslus, galima įvertinti tokius rėžius atskirai kelioms viršūnėms ir paimti didžiausią iš jų:

$$LB(I) = \max_{x_v} \left(f(x_v) - L \max_{x \in I} \|x - x_v\| \right).$$

2.3. Eksperimentų rezultatai

Pasiūlytas Lipschitzo globalios optimizacijos šakų ir rėžių algoritmas su simpleksiniais posričiais buvo realizuotas C++ programa. Padengimo, išrinkimo, padalijimo ir rėžių skaičiavimo taisyklės buvo pagrįstos eksperimentų rezultatais. Eksperimentuose buvo naudojamos dvimatės literatūroje nagrinėtos testo funkcijos. Testo funkcijų minimizavimo efektyvumas buvo vertinamas naudojant tikslo funkcijos skaičiavimų kiekį, reikalingą minimizavimo uždaviniui išspręsti.

Eksperimentai parodė, kad, vertinant tikslo funkcijos apatinį rėžį simplekse, geriau naudoti funkcijos reikšmes simplekso viršūnėse, o ne simplekso viduje. Taip yra todėl, kad funkcijos reikšmes simplekso viršūnėse galima panaudoti vertinant tikslo funkcijos rėžį gretimuose ir palikuoniniuose simpleksuose.

Vienos iš testo funkcijų minimizavimo rezultatai, gauti naudojant įvairias padengimo, padalijimo ir išrinkimo taisykles, pateikti 1 lentelėje. Rėžiai įvertinami naudojant tikslo funkcijos reikšmes simplekso viršūnėse. Kitų testo funkcijų minimizavimo rezultatai yra panašūs.

1 lentelė. Vienos iš testo funkcijų minimizavimo rezultatai, gauti naudojant įvairias padengimo, padalijimo ir išrinkimo taisykles

Pradinio padengimo taisyklė	Padalijimo taisyklė	Išrinkimo taisyklė	Tikslo funkcijos skaičiavimų kiekis
Viršūnių trianguliacija	Į du per ilgiausios kraštinės vidurį	Geriausiojo	244
		Platyn	244
		Gilyn	259
	Į keturis per visų kraštinių vidurius	Geriausiojo	371
		Platyn	384
		Gilyn	658
Padengimas vienu taisyklingu simpleksu	Į du per ilgiausios kraštinės vidurį	Geriausiojo	263
		Platyn	262
		Gilyn	427
	Į keturis per visų kraštinių vidurius	Geriausiojo	328
		Platyn	328
		Gilyn	321
Padengimas vienu stačiuoju simpleksu	Į du per ilgiausios kraštinės vidurį	Geriausiojo	257
		Platyn	260
		Gilyn	275
	Į keturis per visų kraštinių vidurius	Geriausiojo	363
		Platyn	383
		Gilyn	401

Iš eksperimento rezultatų matyti, kad geriausiojo išrinkimo strategija yra tinkamiausia, platyn strategijos rezultatai šiek tiek blogesni, gilyn strategija yra blogiausia. Geriausia dalyti simpleksą į du per ilgiausios kraštinės vidurį negu į kelis per visų kraštinių vidurius. Nėra prasmės išlaikyti simpleksų taisyklingumą. Pradinis padengimas nedaro didesnės įtakos, tačiau viršūnių trianguliacija dažniau yra geresnė negu perdengimai simpleksais.

Remiantis eksperimentų rezultatais, buvo pasirinktos tokios šakų ir rėžių algoritmo su simpleksiniais posričiais taisyklės:

- Stačiakampė leistinoji sritis padengiama simpleksais naudojant viršūnių trianguliaciją.
- Naudojama geriausiojo posričio išrinkimo strategija.
- Simpleksas padalijamas į du per ilgiausios briaunos vidurį ir viršūnes, nepriklausančias ilgiausiai briaunai.
- Tikslo funkcijos minimumo rėžiai simplekse vertinami naudojant tikslo funkcijos reikšmes viršūnėse.

Pasiūlytas algoritmas buvo eksperimentiškai palygintas su geriausiu literatūroje aprašytu Lipschitzo šakų ir rėžių algoritmu su stačiakampiais posričiais GHJ[1] ir dvimačiu Lipschitzo šakų ir rėžių algoritmu su taisyklingais simpleksiniais posričiais CZ[2].

Įvairių testo funkcijų minimizavimo rezultatai, gauti naudojant GHJ, CZ ir pasiūlytą algoritmą, pateikti 2 lentelėje.

2 lentelė. Testo funkcijų minimizavimo rezultatai, gauti naudojant skirtingus algoritmus

Testo funkcija	GHJ	CZ	Pasiūlytas
1	643	489	611
2	167	37	132
3	3531	2618	2185
3.1	3953	3245	1559
3.2	3035	2665	1225
3.3	3689	3387	1429
4	45	41	70
5	73	53	80
6	969	629	838
7	7969	6370	3117
8	301	255	244
9	13953	8759	3773
9.1	14559	9531	3890
9.2	13281	9002	3729
9.3	12295	8917	3610
10	1123	820	848
11	2677	2222	1566
12	12643	10851	4001
13	15695	10643	4084

Rezultatai rodo, kad daugumai testo funkcijų minimizuoti reikalingas tikslo funkcijų skaičiavimų pasiūlytu algoritmu kiekis yra mažiausias. Kitoms testo funkcijoms

[1] **Hansen P. and Jaumard B.** Lipshitz optimization, *R.Horst and P.Pardalos (eds.), "Handbook of Global Optimization", Kluwer Academic Publishers, Dodrecht, 404-493*. 1995.
[2] **Clausen J. and Žilinskas A.** Global Optimization by Means of Branch and Bound with Simplex Based Covering. *Technical report. IMM-REP-1998-23, DTU*. 1998.

minimizuoti reikalingas funkcijų skaičiavimų kiekis yra panašus kaip visų algoritmų. Vertinant funkcijų skaičiavimo kiekio kriterijumi, pasiūlytas šakų ir rėžių globalios optimizacijos algoritmas su simpleksiniais posričiais paprastai yra greitesnis negu kiti Lipschitzo optimizacijos šakų ir rėžių algoritmai.

2.4. Lygiagretieji algoritmai

Realizuotos trys pasiūlyto šakų ir rėžių algoritmo su simpleksiniais posričiais lygiagrečiosios versijos:

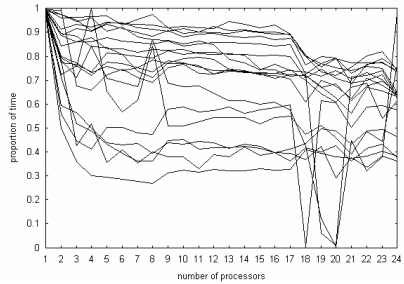
- algoritmas su pradiniu užduočių paskirstymu,
- šeimininko-vergų algoritmas su centralizuotu užduočių perskirstymu,
- algoritmas su paskirstytu dinaminiu užduočių perskirstymu.

Algoritmui su paskirstytu dinaminiu užduočių perskirstymu buvo naudojamos dvi pradinio užduočių paskirstymo strategijos: paskirstyta ir centralizuota. Paskirstyta strategija yra panaši kaip algoritmo su pradiniu užduočių paskirstymu: kiekvienas procesorius padengia dalį leistinosios srities. Centralizuota strategija yra panaši į algoritmo su centralizuotu užduočių perskirstymu: vienas procesorius padengia leistinąją sritį, kiti procesoriai yra neaktyvūs po inicializavimo.

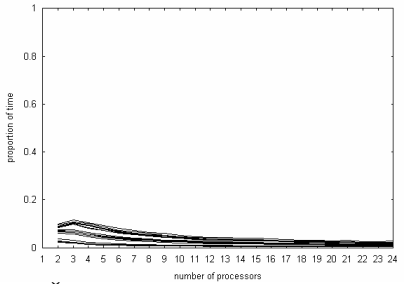
Lygiagretieji algoritmai buvo realizuoti naudojant C++ ir MPI pranešimų persiuntimo biblioteką. Algoritmai buvo testuojami Sun HPC 6500 UltraSPARC-II sistemoje Edinburgo lygiagrečiųjų skaičiavimų centre. Buvo naudojama iki 24 lygiagrečiųjų procesorių.

Lygiagretieji algoritmai buvo palyginti naudojant algoritmų lygiaretinimo koeficientus, kurie buvo įvertinti minimizuojant tas pačias testo funkcijas, kaip ir nuosekliajame variante. Buvo nustatyta, kad dėl anomalijų standartiniai lygiagretinimo koeficientai netinka objektyviam realizuotų lygiagrečiųjų algoritmų įvertinimui. Algoritmams palyginti buvo naudojamas laiko panaudojimo skaičiavimams proporcijos koeficientas. Laiko proporcijos koeficiento priklausomybės nuo procesorių skaičiaus diagramos pateiktos 5 paveiksle. Kiekviena kreivė diagramose žymi skirtingos testo funkcijos minimizavimo rezultatus.

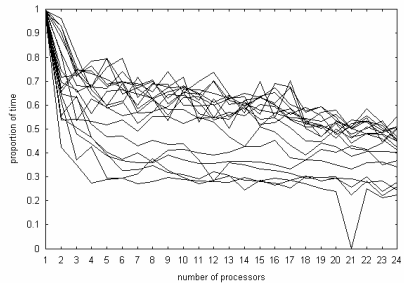
Vidutinė laiko panaudojimo skaičiavimams proporcija algoritmui su pradiniu užduočių paskirstymu, naudojant 15 procesorių, yra apie 0,6. Tai reiškia, kad šis lygiagretusis algoritmas, naudodamas 15 procesorių, atliks apie $15\times0,6=9$ kartus daugiau skaičiavimų per tą patį laiką negu naudodamas 1 procesorių. Algoritmas su paskirstytu dinaminiu užduočių perskirstymu ir paskirstytu inicializavimu, naudodamas 15 procesorių, vidutiniškai atliks apie $15\times0,5=7,5$ karto daugiau skaičiavimų per tą patį laiką negu naudodamas 1 procesorių. Algoritmas su paskirstytu dinaminiu užduočių perskirstymu ir centralizuotu inicializavimu, naudodamas 15 procesorių, vidutiniškai atliks apie $15\times0,3=4,5$ karto daugiau skaičiavimų per tą patį laiką negu naudodamas 1 procesorių.



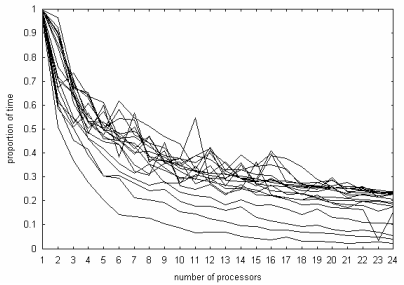
Algoritmas su pradiniu užduočių paskirstymu



Šeimininko-vergų algoritmas su centralizuotu užduočių perskirstymu



Algoritmas su paskirstytu dinaminiu užduočių perskirstymu ir paskirstytu inicializavimu



Algoritmas su paskirstytu dinaminiu užduočių perskirstymu ir centralizuotu inicializavimu

5 pav. Laiko panaudojimas skaičiavimams skirtingiems lygiagretiesiems algoritmams

Rezultatai rodo, kad šeimininko-vergų paradigma pasiūlytam algoritmui netinka. Taip yra todėl, kad tikslo funkcijos apatinio rėžio skaičiavimas yra paprastas ir, palyginti su užduoties siuntimu ir rezultatų grąžinimu, užtrunka trumpai.

Algoritmų su paskirstyta inicializacija laiko panaudojimas skaičiavimams yra geresnis negu algoritmų su centralizuota inicializacija. Taip yra todėl, kad po centralizuotos inicializacijos tik vienas procesorius turi užduotis, o kiti jų tik laukia. Be to, esant paskirstytai inicializacijai, paieškos medis yra platesnis.

Algoritmo su pradiniu užduočių paskirstymu laiko panaudojimas yra geresnis negu algoritmų su paskirstytu dinaminiu užduočių perskirstymu. Taip yra todėl, kad užduočių skaičiavimo laikas yra mažas palyginti su procesorių bendravimo valdymo laiku. Bendravimas užima papildomo laiko, tačiau paieškos medis yra siauras ir neužtenka užduočių pasidalinimui.

3. Juodosios dėžės globali optimizacija inspiruota intervalų metodų

Šiame skyriuje pristatytas naujas juodosios dėžės globalios optimizacijos metodas, inspiruotas intervalų metodų. Nuosekliojo algoritmo eksperimentų rezultatai lyginami su intervalų globalios optimizacijos algoritmo rezultatais. Aptariamas lygiagrečiosios algoritmo versijos realizavimas.

Teoriškai, nurodyto tikslumo globalios optimizacijos uždavinio sprendinys gali būti rastas, jei funkcijos reikšmių kitimo greitis yra aprėžtas, pavyzdžiui, jei funkcija yra Lipschitzo tolydinė su žinoma Lipschitzo konstanta. Tačiau juodosios dėžės situacijoje tokie metodai praranda savo teorinius privalumus, o jų sėkmingi praktiniai taikymai yra mažų dimensijų. Intervalų analizės metodai garantuoja konvergavimą į globalaus minimumo taškų aibę ir yra efektyvūs didelei klasei uždavinių. Tačiau intervalų metodai nepritaikomi juodosios dėžės situacijoje, pavyzdžiui, kai uždavinys aprašytas programos kodu, kuriame nenumatyta intervalų aritmetika.

Pasiūlytas juodosios dėžės globalios optimizacijos metodas yra inspiruotas intervalų metodų, tačiau intervalų aritmetikos procedūros yra pakeistos euristinėmis. Metodas gali būti interpretuojamas kaip lokalių paieškų valdymo strategija globalaus minimumo paieškai. Metodas negarantuoja, kad rastasis sprendinys yra nurodyto tikslumo. Todėl vartotojas turi nurodyti maksimalų sprendimo laiką, o ne norimą tikslumą, kaip to reikalauja dauguma tradicinių metodų.

3.1. Metodas

Algoritmas naudoja tikslo funkcijos f reikšmes eilėje posričio B taškų. Informacija apie funkcijos išvestinę f' nustatoma naudojant baigtinius skirtumus. Naudojamas pakankamai efektyvus lokalios paieškos algoritmas. Metodas koordinuoja lokalias paieškas. Sferos $ball_z=\{x\in B|\ \|x-z\|\le\varepsilon_{cluster}\}$ aplink rastus lokalaus minimumo taškus z posrityje B eliminuojamos iš posričio, $\varepsilon_{cluster}$ yra vartotojo nurodytas parametras. Taip užtikrinama, kad lokali paieška nepriartėtų prie jau žinomo lokalaus minimumo taško.

Aprašytas algoritmas yra panašus į bendrą šakų ir rėžių algoritmą. Kadangi šiuo atveju sprendinys gali būti nerastas, tai algoritmas yra įterptas į skaičiavimų atnaujinimo ciklą. Be kandidatų aibės C, naudojama *išorinė* kandidatų aibė G. Kai baigiamas vidinis ciklas, kandidatų aibė C yra tuščia, o aibės G elementai padengia visą leistinąją sritį D, išskyrus sferas aplink rastus lokalių minimumų taškus. Kandidatų aibė C regeneruojama papildant ją padalytais aibės G elementų posričiais. Posričio padalijimas priklauso nuo surinktos informacijos apie lokalių minimumų taškus dalijamame posrityje, kaip tas aprašyta toliau.

Naujojo algoritmo struktūra pateikta 6 paveiksle. Kintamasis *SolSet* turi konverguoti į globalaus minimumo taškų aibę X^*. Mažas teigiamas skaičius μ naudojamas atskirti X^* nuo kitų lokalių minimumų taškų. Teoriškai μ galėtų būti lygus 0, tačiau praktiškai dėl skaičiavimo paklaidų μ turi būti teigiamas.

```
Inicializuojami: C, f̄, G
while nepasiektas laiko maksimumas do
   while C ≠ ∅ do
      Išrenkamas geriausias B∈C
      Generuojamas 2n+1 atskaitos taškas posrityje B
      reduce-or-subdivide(B)→result, f̄, garbage
      C∪{result}→C
      G∪{garbage}→G
   end
   {p∈C | f(p) ≤ f̄ + μ} ∪ SolSet → SolSet
   while G ≠ ∅ do
      Išrenkamas B∈G
      subdivide(B)→result
      C∪{result}→C
   end
end
```

6 pav. Juodosios dėžės globalios optimizacijos algoritmas, inspiruotas intervalų metodų

Reduce-or-subdivide procedūra yra pateikta 7 paveiksle. Monotoniškumo testas yra pagrįstas informacija apie gradientus posričio taškuose. Newtono testo metu atliekamos lokalios paieškos iš atskaitos taškų.

```
if monotone then
   Monotone(B)→result
   B→garbage
else if (case 1) or (case 2) then
      Newton(B)→result
      B\{ball_z | z∈B}→garbage
   else if (case 3) and ( LB(B) > f̄ ) then
         ∅→result
         B\{ball_z | z∈B}→garbage
      else
         subdivide(B)→B¹,B²
         {B¹,B²}\{ball_z | z∈B}→result
         ∅→garbage
update f̄
```

7 pav. Reduce-or-subdivide procedūra

Pagal lokalių paieškų rezultatą skiriami keturi atvejai:

1. Visos lokalios paieškos išeina už B ribų. Laikoma, kad posrityje B nėra lokalių minimumų taškų. $B \cap \partial(D) \rightarrow result$, čia $\partial(D)$ žymi D kraštą.
2. Visos lokalios paieškos konverguoja į tą patį tašką $z \in B$. $\{z\} \rightarrow result$.
3. Keli lokalių minimumų taškai rasti posrityje B. Apatinis rėžis $LB(B)$ įvertinamas naudojant maksimalią gradiento normą *maxgrad*, kuri surasta vykdant lokalias paieškas posrityje B. Tarkim, kad $f(x_{\min})$ yra mažiausia žinoma funkcijos reikšmė posrityje B, tada $LB(B)=\min\{f(x_{\min})-maxgrad \times \|x-x_{\min}\| \mid x \in B\}$. Jei $LB(B) > \overline{f}$, $\varnothing \rightarrow result$, kitu atveju posritis yra padalinimas.
4. Nė vienas iš trijų prieš tai aprašytų atvejų. Posritis padalijamas.

Posritis visada dalijamas į du koordinačių ašims ortogonalia hiperplokštuma. Padalijimas priklauso nuo to, kiek lokalių minimumų taškų rasta posrityje B:

- Lokalių minimumų taškų posrityje nerasta. Posritis padalijamas į dvi lygias dalis hiperplokštuma, einančia per B centrą ir statmena ilgiausioms posričio kraštinėms.
- Rastas vienas lokalaus minimumo taškas. Randama koordinatė, kurioje atstumas nuo lokalaus minimumo taško iki posričio krašto yra didžiausias. Posritis dalijamas mažinant šį atstumą pusiau hiperplokštuma, statmena šios koordinatės ašiai.
- Rastas daugiau negu vienas lokalaus minimumo taškas. Randami du lokalių minimumų taškai su mažiausiomis funkcijos reikšmėmis. Randama koordinatė, kurioje atstumas tarp šių dviejų taškų yra didžiausias. Posritis dalijamas mažinant šį atstumą pusiau hiperplokštuma, statmena šios koordinatės ašiai.

3.2. Eksperimentinis testavimas

Pasiūlyto algoritmo įvertinimo kriterijai yra tikslo funkcijos skaičiavimų kiekis (Nrf) ir gradiento funkcijos skaičiavimų kiekis (Nrg). Jei gradientas yra aprašytas analitine funkcija, jo skaičiavimas gali kainuoti nuo vieno iki n kartu brangiau už tikslo funkcijos skaičiavimą. Kai gradientas įvertinamas naudojant baigtinius skirtumus, jo skaičiavimas kainuoja apie $n+1$ kartą brangiau už tikslo funkcijos skaičiavimą.

Intervalų metodai yra artimiausi pasiūlytam metodui. Dėl to pasiūlyto algoritmo optimizavimo rezultatai yra lyginami su intervalų metodo realizacijos. Naudota intervalų metodo realizacija naudoja ir intervalų, ir realiųjų skaičių aritmetiką. Realios funkcijos skaičiavimų kiekis (Nrf) ir intervalų funkcijos skaičiavimų kiekis (Nif) yra šio algoritmo įvertinimo kriterijai. Šio metodo autoriai teigia, kad intervalų funkcijos skaičiavimas užtrunka vidutiniškai dvigubai ilgiau už realios funkcijos skaičiavimą.

Kai algoritmų sustojimo salygos yra vienodos, algoritmai lyginami pagal viso sprendimo metu įvykdytų funkcijų skaičiavimų kiekius. Kadangi mūsų atveju lyginamų algoritmų sustojimo sąlygos yra nevienodos, algoritmai lyginami pagal funkcijos skaičiavimų kiekius, reikalingus visiems globalaus minimumo taškams surasti.

Eksperimentuose buvo naudojamos skirtingos dimensijos testo funkcijos su skirtingais globalaus ir lokalių minimumų taškų kiekiais. Be testo funkcijų, buvo spręsti du praktiniai uždaviniai: Cola ir Bone Growth. Algoritmų įvertinimų palyginimas yra pateiktas 3 lentelėje. Palyginimui naudojami funkcijų skaičiavimų kiekiai visiems

globalaus minimumo taškams surasti: tikslo funkcijos skaičiavimų kiekis (Nrf) ir gradiento funkcijos skaičiavimų kiekis (Nrg) juodosios dėžės globalios optimizacijos algoritmui bei realios funkcijos skaičiavimų kiekis (Nrf) ir intervalų funkcijos skaičiavimų kiekis (Nif) intervalų globalios optimizacijos algoritmui

3 lentelė. Juodosios dėžės ir intervalų algoritmų įvertinimų palyginimas

Funkcijos	Juodosios dėžės		Intervalų	
	Nrf	Nrg	Nrf	Nif
Rosenbrock	27	23	104	19
McCormic	11	10	96	13
Box and Betts	6	6	103	33
Paviani	14	11	524	221
Generalized Rosenbrock	359	346	14260	1921
Goldstein and Price	75	51	243	226
Shekel 5	66	37	98	41
Shekel 7	24	13	101	41
Shekel 10	22	12	101	41
Levy4	30502	23798	111	41
Levy5	2786	2147	127	61
Levy6	5292	4154	235	85
Levy7	35	30	235	113
Griewank	124	81	186	221
Cola	8995	7226	fails	
Bone Growth	529	414	fails	
Six Hump Camel back	49	45	342	5411
Branin	666	527	803	687
Shubert	6825	4412	1550	4776
Hansen	7655	5037	892	1686

Pasiūlytas juodosios dėžės globalios optimizacijos algoritmas ilgai minimizuoja testo funkcijas su labai dideliu panašaus gylio lokalių minimumų kiekiu, pvz. kad Levy, Shubert ir Hansen funkcijos. Kiti uždaviniai, nors ir su daugeliu lokalių minimumų, pvz. Griewank, Cola ir Bone Growth, buvo išpręsti pakankamai greitai.

Intervalų algoritmamui nepavyko rasti globalaus minimumo praktiniuose Cola ir Bone Growth uždaviniuose. Skaičiavimai nutrūko dėl atminties persipildymo. Cola uždavinio sprendimas sustojo sugeneravus 131072 posričius ir atlikus 393213 intervalų funkcijos skaičiavimus. Bone Growth uždavinio rezultatai yra panašūs. Funkcijų skaičiavimų kiekiai uždaviniuose su analitiškai apibrėžtomis funkcijomis yra palyginus maži – intervalų metodas juos sprendžia sparčiai.

Eksperimentų rezultatai rodo, kad pasiūlytas juodosios dėžės globalios optimizacijos algoritmas yra patikimas – jis išsprendė visus bandytus uždavinius. Kartais jis yra spartesnis už intervalų algoritmą, kai kurioms funkcijoms – lėtesnis. Tačiau pasiūlytojo

metodo taikymo sritis yra žymiai platesnė. Nors šis metodas negarantuoja, kad bus nurodytu tikslumu surastas globalus minimumas, tačiau jis yra geras intervalų algoritmų pakaitalas, kai šie negali būti taikomi, pavyzdžiui, juodosios dėžės situacijoje.

3.3. Lygiagretusis algoritmas

Paprasčiausia pasiūlytąjį metodą lygiagretinti naudojant pradinį užduočių paskirstymą. Leistinoji uždavinio sritis pradžioje padalijama į posričius, kurie paskirstomi procesoriams. Toliau procesoriai nepriklausomai vykdo 6 paveikslo algoritmą ir neperskirsto užduočių. Procesoriai tik pasikeičia tarpusavyje geriausia surasta tikslo funkcijos reikšme. Baigtis yra tokia pat kaip ir pagal nuoseklioji varianto sustojimo sąlygą – laiko limitas. Išorinio ciklo egzistavimas 6 paveikslo algoritme užtikrina, kad procesoriai visada turės darbo. Tačiau sunku nustatyti, ar procesoriai atlieka skaičiavimus su vienodai perspektyviais posričiais.

Mobilus lygiagretusis algoritmas buvo realizuotas C++ naudojant MPI biblioteką. Mobilumo problemos dėl nemobilių C++ transliatorių ir MPI realizacijų buvo išspręstos. Algoritmas yra patikrintas visuose Edinburgo lygiagrečiųjų skaičiavimų centro lygiagrečiuosiuose kompiuteriuose:

- Sun HPC 6500 UltraSPARC-II lygiagrečioji sistema su Sun MPI,
- Beowulf tipo asmeninių kompiuterių klasteris su MPICH 1.2.0,
- Cray T3E,

ir IBM RS/6000 SP Vilniaus Gedimino technikos universitete.

4. Juodosios dėžės globalios optimizacijos algoritmo taikymas praktiniams uždaviniams spręsti

Šiame skyriuje yra aprašyti pasiūlyto juodosios dėžės globalios optimizacijos algoritmo taikymai. Algoritmas yra taikomas trims tipiniams praktiniams globalios optimizacijos uždaviniams: daugiamačiam vertinimui, žmogaus žandikaulio augimo modelio sudarymui ir daugelio kūnų energijos minimizavimui.

Pasiūlyto juodosios dėžės globalios optimizacijos algoritmo taikymas priklauso nuo šių jo savybių:

- Tai daugiamatės globalios optimizacijos algoritmas.
- Leistinoji sritis apibrėžta kintamųjų intervalais. Algoritmas taikomas globalios optimizacijos uždaviniams be ribojimų.
- Algoritmas gali būti taikomas juodosios dėžės situacijoje. Nesvarbu, kaip tikslo funkcija yra išreikšta.
- Tikslo funkcijos gradientai yra skaičiuojami vartotojo sudaryta paprograme arba įvertinami naudojant baigtinius skirtumus.

4.1. Daugiamatis vertinimas

Eksperimentiniai mokslai surenka daug duomenų, kurie vėliau anlizuojami įvairiais būdais. Dažnai matematiniai metodai norimoms žinioms išskirti iš duomenų turi būti derinami su tyrėjo patirtimi ir intuicija. Tačiau žmonių euristiniai sugebėjimai yra skirti

ir išvystyti dvimatei arba trimatei erdvei. Daugiamačių duomenų atvaizdavimas dvimatėje arba trimatėje erdvėje labai pagelbėja euristinei analizei ir užtikrina teisingesnes išvadas. Daugiamatis vertinimas dažnai yra naudojamas daugiamačių duomenų struktūros analizei atvaizduojant juos dvimatėje arba trimatėje erdvėje.

Daugiamatis vertinimas sprendžia, kaip N artumo duomenimis apibrėžtų objektų gali būti patikimai atvaizduoti taškais mažo mato Euklido erdvėje. Objektų artumas yra apibrėžiamas jų porų skirtingumu. i-ojo ir j-ojo objektų skirtingumas yra apibrėžtas realiu skaičiumi δ_{ij}. Objektai turi būti atvaizduoti taškais m-matėje erdvėje. Atstumas tarp taškų $\mathbf{x}_i$ ir $\mathbf{x}_j$ paprastai matuojamas Euklido atstumu $d_{ij}=\|\mathbf{x}_i-\mathbf{x}_j\|$. Atvaizdavimo kokybė yra matuojama *įtempimo* funkcija, kuri lygina objektų skirtingumą ir atstumą tarp juos atvaizduojančių taškų. Daugiamatis vertinimas gali būti sprendžiamas minimizuojant šią funkciją: turi būti rastos tokios N taškų koordinatės m-matėje erdvėje, kad įtempimo funkcija būtų minimali. Dažnai naudojama įtempimo funkcija

$$f(X)=\sum_i\sum_j w_{ij}(d_{ij}-\delta_{ij})^2 ,$$

čia w_{ij} yra neneigiami svoriai.

Įtempimo funkciją minimizuoti yra sudėtinga, nes:

- praktiniai uždaviniai paprastai yra daugiamačiai, optimizacijos kintamųjų skaičius yra lygus $N\times m$;
- įtempimo funkcija yra ne visur diferencijuojama;
- yra daug lokalių minimumų taškų;
- įtempimo funkcija yra invariantinė perkėlimui, sukimui ir atspindžiams.

Paskutinis sunkumas turi būti būtinai išspręstas, jei įtempimo funkcija yra minimizuojama. Tai galima padaryti fiksuojant kai kurias koordinates:
$x_{i,i..m}=0$, kai $i=1,\ldots,m$
ir nustatant kai kuriuos būti neneigiamais:
$x_{i,j-1}\geq 0$, kai $i=2,\ldots,m+1$.
Nefiksuotos koordinatės yra uždavinio argumentai:
$x_{2,1}=u_1,\ldots,x_{p,m}=u_n$.
Uždavinio dimensija:

$$n=\frac{m(2N-m-1)}{2} .$$

Dažnai naudojamas daugiamačio vertinimo algoritmų testavimo uždavinys „Cola” yra pagrįstas eksperimentinio kelių gaiviųjų gėrimų bandymo rezultatais. 38 studentai bandė dešimt gėrimų. Kiekviena gėrimų pora buvo įvertinta jų skirtingumu 9 balų skalėje (1 – labai panašūs, 9 – visiškai skirtingi). Susumuoti skirtingumo įvertinimai yra uždavinio duomenys. Susumuoti ir padalyti iš 100 skirtingumo įvertinimai yra pateikti 4 lentelėje.

4 lentelė. 10 gaiviųjų gėrimų skirtingumai

	Pepsi	Coke	Classic Coke	Diet Pepsi	Diet Slice	Diet 7-Up	Dr. Pepper	Slice	7-Up	Tab
Pepsi	…									
Coke	1,27	…								
Classic Coke	1,69	1,43	…							
Diet Pepsi	2,04	2,35	2,43	…						
Diet Slice	3,09	3,18	3,26	2,85	…					
Diet 7-Up	3,20	3,22	3,27	2,88	1,55	…				
Dr, Pepper	2,86	2,56	2,58	2,59	3,12	3,06	…			
Slice	3,17	3,18	3,18	3,12	1,31	1,64	3,00	…		
7-Up	3,21	3,18	3,18	3,17	1,70	1,36	2,95	1,32	…	
Tab	2,38	2,31	2,42	1,94	2,85	2,81	2,56	2,91	2,97	…

Šio daugiamačio vertinimo uždavinio tikslas – rasti 10 gėrimus vaizduojančių objektų išsidėstymą dvimatėje erdvėje, kuris padėtų interpretuoti duomenis. Yra daug lokalių minimumų taškų, o interpretuojant skirtingus lokalių minimumų taškus atitinkančias konfigūracijas gaunami skirtingi rezultatai. Dėl to svarbu rasti globalų minimumą ir jį atitinkančią konfigūraciją.

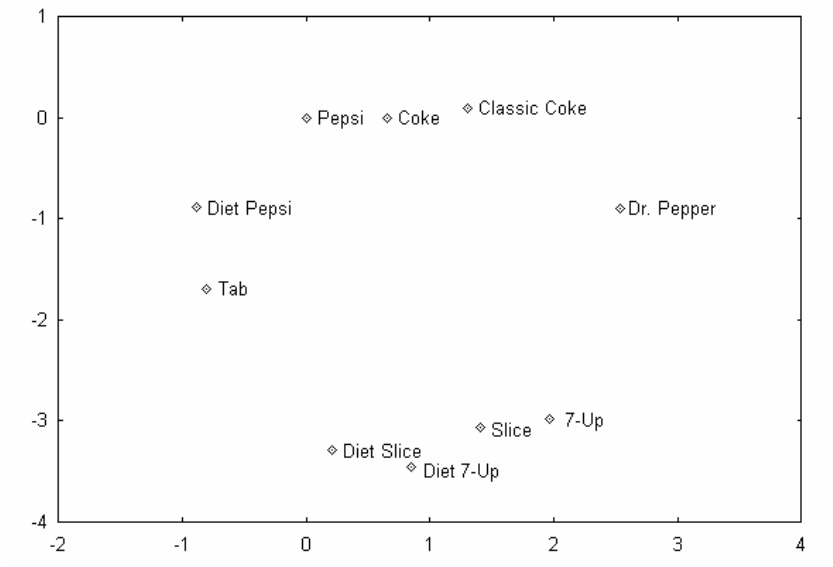
Šio globalios optimizacijos uždavinio objektų skaičius N=10. Reikia rasti konfigūraciją m=2 erdvėje. Uždavinio kintamųjų skaičius n=17. Liestinoji sritis $D=([0,4]^2,[-4,4]^{n-2})$. Tikslo funkcija

$$f(X)=\sum_{j<i}\left(\sqrt{\sum_{k=1}^{m}\left(x_{i,k}-x_{j,k}\right)^2}-\delta_{i,j}\right)^2 .$$

Geriausia žinoma tikslo funkcijos reikšmė yra f^*=11,7464. Juodosios dėžės globalios optimizacijos algoritmas rado šį minimumą po 25242 tikslo funkcijos skaičiavimų. Rastas globalaus minimumo taškas U^*=(0,6519, 1,3019, 0,0992, -0,8838, -0,8796, 0,2047, -3,2841, 0,8512, -3,4625, 2,5325, -0,8952, 1,4099, -3,0737, 1,9626, -2,9787, -0,8078, -1,6898). Šį tašką atitinkanti konfigūracija parodyta 8 paveiksle.

Šiame sprendime matyti, kad kokakolos tipo gėrimai išsidėstę vienoje grupėje. Slice ir 7-Up tipo gėrimai sudaro kitą grupę. Dr. Pepper gėrimas nepriklauso jokiai grupei, vadinasi, gerokai skiriasi nuo visų. Be to, galima įžvelgti, kad dietiniai gėrimai yra vienoje konfigūracijos pusėje, tarp jų yra ir Tab gėrimas.

Intervalų metodo realizacija, naudojama palyginimui, nesugebėjo rasti globalaus uždavinio minimumo ir sustojo perpildžius atmintį.



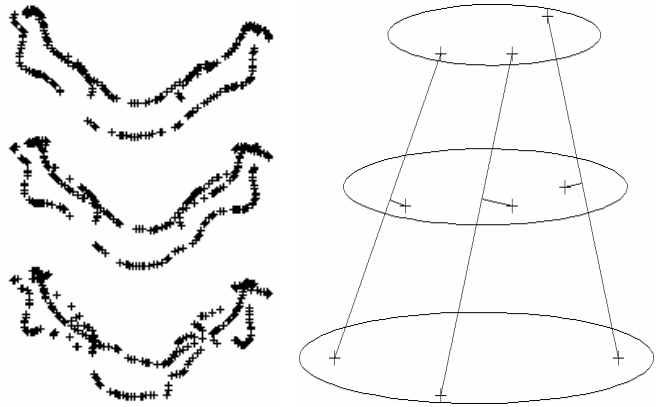
8 pav. „Cola" uždavinio sprendimas

4.2. Žmogaus žandikaulio augimo modelis

Šis uždavinys susijęs su žmogaus apatinio žandikaulio augimo tiesinio modelio nustatymu. To paties paciento, kai jis yra 9 mėnesių, 21 mėnesio ir 7 metų amžiaus, trijų žandikaulių 271 ekvivalenčios morfologijos taško koordinatės yra uždavinio duomenys. Uždavinio tikslas – nustatyti tokį trijų žandikaulių išsidėstymą trimatėje erdvėje, kad atstumų tarp taškų iš vidurinio žandikaulio ir linijų, jungiančių atitinkamus taškus iš pirmojo ir trečiojo žandikaulio, suma būtų minimali (9 pav.). Vidurinis žandikaulis yra fiksuotas, o kiti du turi po 6 laisvės laipsnius: 3 sukimosi kampus ir 3 perkėlimo kryptis. Uždavinio dimensija n=12. Leistinoji sritis D=([-π, π]^6, [-120,120]^6).

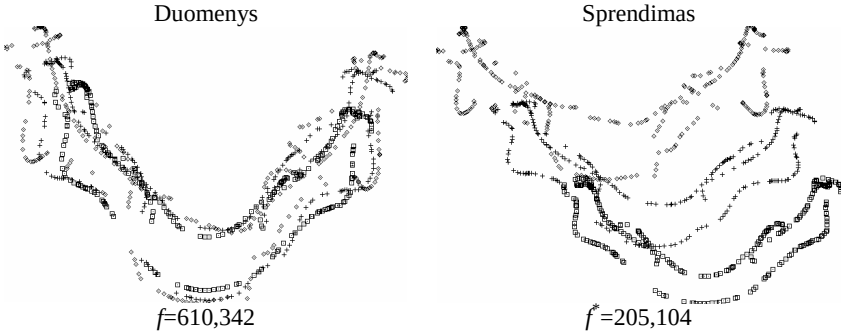
Geriausia žinoma šio uždavinio tikslo funkcijos reikšmė yra f^*=205,104. Naudojant juodosios dėžės globalios optimizacijos algoritmą, šis minimumas rastas po 13101 tikslo funkcijos skaičiavimo. Rastas globalaus minimumo taškas yra X^*=(-0,125288, -0,048084, 0,0683822, -5,28448, 13,5913, 47,4381, 0,100655, 0,00663149, 0,0861344, 15,1711, -19,2757, -44,7489). Atitinkamas sprendimas pavaizduotas 10 paveiksle. Tikslo funkcijos reikšmė yra tris kartus mažesnė, negu esant pradiniams duomenims.

Intervalų metodu, naudojamu palyginimui, nesugebėta rasti globalaus uždavinio minimumo ir skaičiavimai nutrūko perpildžius atmintį.



$$f(X) = \sum_{i=1}^{271} h_i$$

9 pav. Žandikaulio augimo modelio uždavinys



10 pav. Žandikaulio augimo modelio uždavinio sprendimas

4.3. Daugelio kūnų uždavinys

Yra daug fizikos, chemijos ir medžiagotyros uždavinių, kuriuos sprendžiant reikia nustatyti bendrą atomų sistemos energiją, kuri priklauso nuo atomų padėties erdvėje. Struktūrinės ir energinės sudėtingų sistemų savybės gali būti nustatytos naudojant empirinius tarpatominius potencialus. Bendra sistemos energija E išskirstoma į atomų energijas E_i ir į ryšių energijas V_{ij}:

$$E = \sum_i E_i = \frac{1}{2} \sum_{i \neq j} V_{ij} \ ;$$

čia indeksai i ir j žymi kiekvieną sistemos atomą.

$V_{ij} = f_C(r_{ij})[a_{ij} f_R(r_{ij}) + b_{ij} f_A(r_{ij})]$,

kur r_{ij} yra atstumas tarp i-ojo ir j-ojo atomų.

$f_R(r) = A\exp(-\lambda_1 r)$, $f_A(r) = -B\exp(-\lambda_2 r)$,

$$f_C(r)=\begin{cases}1, r \le R-D \\ \frac{1}{2}-\frac{1}{2}\sin\left[\frac{\pi(r-R)}{2D}\right], R-D<r<R+D \\ 0, r \ge R+D\end{cases},$$

$b_{ij}=\left(1+\beta^n \xi_{ij}^n\right)^{-\frac{1}{2n}}$, $\xi_{ij}=\sum_{k\neq i,j} f_C(r_{ik}) g(\theta_{ijk}) \exp[\lambda_3^3 (r_{ij}-r_{ik})^3]$,

$$g(\theta)=1+\frac{c^2}{d^2}-\frac{c^2}{d^2+(h-\cos\theta)^2},$$

čia θ_{ijk} yra kampas tarp ryšių *ij* ir *ik*.

$a_{ij}=\left(1+\alpha^n \eta_{ij}^n\right)^{-\frac{1}{2n}}$, $\eta_{ij}=\sum_{k\neq i,j} f_C(r_{ik}) \exp[\lambda_3^3 (r_{ij}-r_{ik})^3]$.

A, *B*, λ_1, λ_2, α, β, *n*, *c*, *d*, *h*, λ_3, *R*, *D* yra modelio parametrai. Arseno As ir dviejų silicio Si(B) ir Si(C) potencialų parametrai pateikti 5 lentelėje.

5 lentelė. Potencialų parametrai

	As	Si(B)	Si(C)
A	10,45561332	3,2647e+3	1,8308e+3
B	14,41961332	9,5373e+1	4,7118e+2
λ_1	6,739581257	8,7963	6,7339
λ_2	4,886847795	3,6001	4,7036
α	0,0	0,0	0,0
β	0,00748809	0,33675	1,0999e-6
n	0,60879133	22,956	0,78734
c	5,2731318	4,8381	1,0039e+5
d	0,75102662	2,0417	16,216
h	0,15292354	0,0000	-0,59826
λ_3	0,0	0,0	0,0
R	1,2381	1,1048	1,0496
D	0,0503062	0,073654	0,055240

Uždavinio tikslas – nustatyti minimalią bendrą energiją atitinkančią atomų sistemą. Tikslo funkcijos kintamieji yra atomų pozicijos. Yra daug lokalių minimumų taškų. Juodosios dėžės globalios optimizacijos algoritmo (BB) ir aštuonių stochastinių algoritmų (CSR2, CRS3, CRS4, CRS5, ABSA, SA, TMSL, MSL) Si(B) uždavinio su 3, 4, 5 ir 6 atomais minimizavimo rezultatai pateikti 6 lentelėje. Algoritmai lyginami naudojant geriausią rastą tikslo funkcijos *f* reikšmę ir tikslo funkcijos skaičiavimų kiekį FE. Kai globalus minimumas yra randamas, jo reikšmė pateikiama laužtiniuose skliaustuose.

6 lentelė. 8 stochastinių ir juodosios dėžės globalios optimizacijos algoritmų Si(B) uždavinio minimizavimo rezultatai

		CRS2	CRS3	CRS4	CRS5	ABSA	SA	TMSL	MSL	BB
3	*f*	[-7,87]	[-7,87]	[-7,87]	[-7,87]	[-7,87]	[-7,87]	[-7,87]	[-7,87]	[-7,87]
	FE	1123	1094	755	808	13992	13978	212	210	64
4	*f*	[-15,71]	[-15,71]	[-15,71]	[-15,71]	[-15,71]	[-15,71]	[-15,71]	[-15,71]	[-15,71]
	FE	4304	3724	2418	2967	27988	46458	2520	5312	186
5	*f*	[-20,40]	-20,31	[-20,40]	-15,70	-20,31	-20,31	-18,97	[-20,40]	[-20,40]
	FE	11111	18348	3971	4321	36417	112951	3594	12954	327
6	*f*	-24,51	-24,51	[-26,52]	-24,51	-25,98	-23,12	-24,44	-24,51	[-26,52]
	FE	37677	64067	44980	42134	151109	206166	4139	18666	256460

Visais algoritmais rasti uždavinio su 3 ir 4 atomais globalūs minimumai ir tikslo funkcijos skaičiavimų kiekiai buvo naudojami lyginant algoritmus. Juodosios dėžės globalios optimizacijos algoritmas yra geriausias, nes juo globalus minimumas randamas po mažiausio tikslo funkcijos skaičiavimų kiekio.

Ne visais algoritmais rasti uždavinio su 5 ir 6 atomais globalūs minimumai ir patikimumas buvo naudojamas lyginant algoritmus. Tik trim stochastiniais algoritmais išspręstas uždavinys su 5 atomais ir tik vienu iš jų išspręstas uždavinys su 6 atomais. Juodosios dėžės globalios optimizacijos algoritmas yra vienas iš patikimiausių, kadangi juo išspręstas uždavinys ir su 5, ir su 6 atomais. Algoritmas yra ir patikimas, ir greitas.

Išvados

Lipschitzo globalios optimizacijos su simpleksiniais posričiais tyrimo išvados:

- Vertinant tikslo funkcijos reikšmių apatinį rėžį simplekse, geriau naudoti funkcijos reikšmes simplekso viršūnėse, o ne simplekso viduje, nes tie patys taškai yra ir gretimų simpleksų viršūnės.
- Geriau dalyti simpleksą į du per ilgiausios kraštinės vidurį negu į kelis per visų kraštinių vidurius. Nėra prasmės išlaikyti simpleksų taisyklingumą. Pradinis padengimas nedaro didesnės įtakos, tačiau viršūnių trianguliacija dažniau geresnė negu perteklenis uždengimas simpleksais.
- Vertinant funkcijų skaičiavimo kiekio kriterijumi, pasiūlytas šakų ir rėžių globalios optimizacijos algoritmas su simpleksiniais posričiais bendru atveju yra greitesnis negu kiti šakų ir rėžių Lipschitzo optimizacijos algoritmai.
- Paradigma šeimininkas-vergai pasiūlytam metodui netinka, nes tikslo funkcijos apatinio rėžio skaičiavimas yra paprastas ir trumpas lyginant su užduoties ir rezultatų siuntimu.
- Algoritmų su paskirstyta inicializacija laiko panaudojimas skaičiavimams yra geresnis negu algoritmų su centralizuota inicializacija. Taip yra todėl, kad po centralizuotos inicializacijos tik vienas procesorius turi užduotis, o kiti jų tik laukia. Be to, paieškos medis yra platesnis, kai naudojama paskirstyta inicializacija.
- Algoritmo su pradiniu užduočių paskirstymu laiko išnaudojimas yra geresnis negu algoritmų su dinaminiu užduočių perskirstymu, nes užduočių skaičiavimas trunka trumpiau negu procesorių bendravimo valdymas. Bendravimas užima papildomo laiko, tačiau paieškos medis yra siauras ir neužtenka užduočių pasidalijimui.

Juodosios dėžės globalios optimizacijos inspiruotos intervalų metodų tyrimo išvados:

- Pasiūlytas juodosios dėžės globalios optimizacijos metodas yra patikimas. Juo rasti visų bandytų uždavinių globalūs minimumai. Metodas gali būti naudojamas vietoj intervalinio, kai pastarasis negali būti pritaikytas, pavyzdžiui, juodosios dėžės atveju.
- Pasiūlytas metodas yra pritaikytas keliems praktiniams uždaviniams spręsti. Vertinant pagal funkcijų skaičiavimo kiekio kriterijų, pasiūlytas metodas nenusileidžia metodams, kurie buvo originaliai naudojami šiems uždaviniams spręsti.
- Pasiūlytas juodosios dėžės globalios optimizacijos metodas yra efektyviai lygiagretinamas. Kadangi šakų ir rėžių algoritmas yra įterptas į skaičiavimus atnaujinantį ciklą, procesoriai visada turi ką skaičiuoti netgi kai naudojamas pradinis užduočių paskirstymas.

KAUNAS UNIVERSITY OF TECHNOLOGY

Julius Žilinskas

Black Box Global Optimization: Covering Methods and their Parallelization

Doctoral dissertation
Summary

Many problems in engineering, physics, economics and other subjects are reduced to global minimization of multimodal functions. The global optimization problems are difficult in the sense of the algorithmic complexity theory. Therefore global optimization algorithms are computationally intensive. If the computing power of usual computers is not sufficient to solve a practical problem, then high performance parallel computers may be helpful.

In chapter 1, the global optimization methods are reviewed. The main attention is paid to the covering global optimization methods and the branch-and-bound technique to solve global optimization problems. Parallel computing is discussed with respect to application in global optimization. The load balancing and termination detection of parallel algorithms is analyzed in detail. The literature on the parallel branch-and-bound technique is reviewed. Criteria of efficiency of parallel algorithms are discussed.

In chapter 2, a new Lipschitz global optimization algorithm with simplicial partitioning is presented. The numerical results of experiments with the sequential algorithm are discussed and compared with results of other Lipschitz global optimization algorithms. The parallelization of the algorithm is described. Several potential parallel implementation versions are assessed with respect to the criteria of parallelization efficiency.

In chapter 3, a new black box global optimization method inspired by interval arithmetic is presented. Sequential algorithm is implemented and investigated. The results of experimental testing are compared with the results of interval global optimization algorithm. A parallel version of the algorithm is implemented.

In chapter 4, the results of applications of the presented black box global optimization method is presented. The method is applied to the following problems: multidimensional scaling, growth model of the human mandible and many body problems. The results of optimization are discussed.

The dissertation is completed with the conclusions and with the summary of the results.

UDK 519.853.4+004.272.2] (043)

SL 344.2002 05 27. 1 apsk. Leid. 1.Tiražas 70 egz.
Užsakymas 209.
Leidykla „Technologija”, K.Donelaičio g. 73, LT-3006 Kaunas
Spausdino KTU spaustuvė, Studentų g. 54, LT-3031 Kaunas