Vilniaus universitetas
Matematikos ir informatikos institutas
L I E T U V A

INFORMATIKA (09 P)

< NAŠUMO PROGRAMINIŲ ROBOTŲ METAEURISTINIŲ MODELIŲ KŪRIMAS, TYRIMAS IR TAIKYMAS >

<Donatas Kavaliauskas>

2018 m. spalis

Mokslinė ataskaita MII-DS-09P-14-<ataskaitos nr. 1>

VU Matematikos ir informatikos institutas, Akademijos g. 4, Vilnius LT-08663

www.mii.lt

Santrauka

Tekste yra pateikiamas statistinio modeliavimo metodologijos aprašymas paremtas Monte Karlo metodu. Ši metodologija yra pritaikyta metaeuristinių algoritmų efektyvumo tyrimui. Statistinio modeliavimo metodologija panaudota tirti 3 planavimo uždavinius. Kompiuteriniuose modeliavimo uždaviniuose tirti modeliuojamo atkaitinimo algoritmas, dirbtinių skruzdžių kolonijos algoritmas bei šakų ir ribų algoritmas.

Reikšminiai žodžiai: PSPlib, metaeuristika, modeliuojamo atkaitinimo algoritmas, dirbtinių skruzdžių kolonijos algoritmas, šakų ir ribų algoritmas, Monte Karlo metodas.

Turinys

1 Įvadas

„P versus NP“ problema yra laikoma svarbiausia informatikos problema. Ji pirmąkart buvo paminėta 1956 m. žymaus vokiečių logiko Kurto Giodelio laiške vienam iš šiuolaikinių kompiuterių kūrėjų Džonui fon Neimanui, o matematiškai griežtai suformuluota amerikiečių mokslininko Stefeno Kuko 1971 m.

Vienas tokių uždavinių yra dviejų procesorių tvarkaraščio sudarymas. Šiam darbui mes turime sudėti darbus taip procesoriams , kad jų vykdymo laikas būtų trumpiausias. Optimalų sprendinį galime gauti patikrinus visus galimus variantus, tačiau kartais dėl duomenų kiekio tai padaryti, kainuoja per daug laiko arba kompiuterio atminties. Tam yra pasitelkiama įvairūs meta-euristiniai algoritmai.

2 <ataskaitos pagrindinė dalis>

Šiame darbe buvo taikoma kompiuterinio modeliavimo metodika. Ši metodika leidžia patikrinti matematinio modelio patikimumą kuriamai gamybos planavimo sistemai. Taip pat tai suteikia papildomos informacijos apie problemą.

Metaeuristinių algoritmų efektyvumo tyrimas turėtų būti atliekamas reguliariai, nes procesų gamybos planavimo uždaviniai gali kisti kiekvieną dieną dėl užsakymo kiekio, rečiau dėl žmogiškųjų ar technologinių resursų kaitos.

2.1 Metodikos

2.1.1 Monte Carlo metodika

Tikslumo tyrimas susideda iš tokių etapų:

- *Modeliuojami elementų parametrų skirstiniai $W(x_i)$;*
- *Skaičiuojamos išėjimo parametro y reikšmės, esant atsitiktinėms x_i reikšmių kombinacijoms, atitinkančioms $w(x_i)$ dėsnius, t. y. modeliuojamas kūrybinis procesas;*
- *Modeliavimo rezultatai apdorojami statistiniais metodais. [1],[2].*

Šios metodikos tikslas yra įvertinti skaitines išėjimo parametro charakteristikas (vidutinę reikšmę ir dispersiją D(y)), nustatyti išėjimo parametro skirstinį w(y) ir surasti tikimybę, kad išėjimo parametro reikšmės bus duotosiose ribose, kintant elementų parametrų reikšmėms pagal skirstinius. [1],[2].

Tiriant išėjimo parametrų tikslumą statistinių bandymų metodu, reikalingos elementų parametrų atsitiktinės reikšmės. Šių reikšmių modeliavimui naudojami atsitiktinių skaičių generatoriai. Didžiausią praktinę reikšmę turi vienodos tikimybės skaičiai intervale [0, 1]. Skaičiai su kitokiais

norimais skirstiniais w(x) gaunami, naudojant vienodos tikimybės skaičius ir sprendžiant lygtį parametro x atžvilgiu, esant įvairioms tikimybės P reikšmėms: ($0 \leq P \leq 1$).

Tokiu būdu gaunami pseudo atsitiktiniai skaičiai x_i, pasiskirstę pagal skirstinį $w(x_i)$. Naudojami ir kitokie atsitiktinių skaičių gavimo būdai pagal norimą skirstinį. Nepriklausomai nuo atsitiktinių skaičių gavimo būdo, apie jų kokybę galima spręsti iš gauto statistinio skirstinio sutapimo su norimu teoriniu skirstiniu. Apie skirstinių sutapimo laipsnį sprendžiama iš sutapimo kriterijų.

Statistinių bandymų metodo pagrindiniai privalumai yra šie:

- *Ištiriamas išėjimo parametrų tikslumas, esant bet kokiems elementų parametrų skirstiniams;*
- *Gaunami rezultatai su maža paklaida, kai bandymų skaičius N artėja prie begalybės, skaičiavimų paklaida artėja prie nulio;*
- *Sudaromos kiekybinės išėjimo parametrų charakteristikos (vidutinė reikšmė, dispersija), randamas skirstinys w(y), kuriam galime rasti teorinį skirstinį ir tikimybę, kad išėjimo parametras bus duotose ribose. [1],[2].*

2.1.2 Weibull'o skirstinys

Weibull'o skirstinys yra tolydusis tikimybių skirstinys. Šis skirstinys yra naudojamas duomenų analizei, tikimybės nustatymui. [3].

Parametras k nurodo skirstinio formą. Kuo didesnis k parametras, tuo greičiau auga skirstinys. Šis parametras yra apskaičiuojamas pagal formules:

$$\hat{k}^{-1} = \frac{\sum_{i=1}^{n} x_i^k * \ln x_i}{\sum_{i=1}^{n} x_i^k} - \frac{1}{n}\sum_{i=1}^{n} \ln x_i$$

$$k = \frac{1}{\hat{k}^{-1}}$$

Parametras λ nurodo skirstinio mastelį. Šis parametras yra apskaičiuojamas pagal formules:

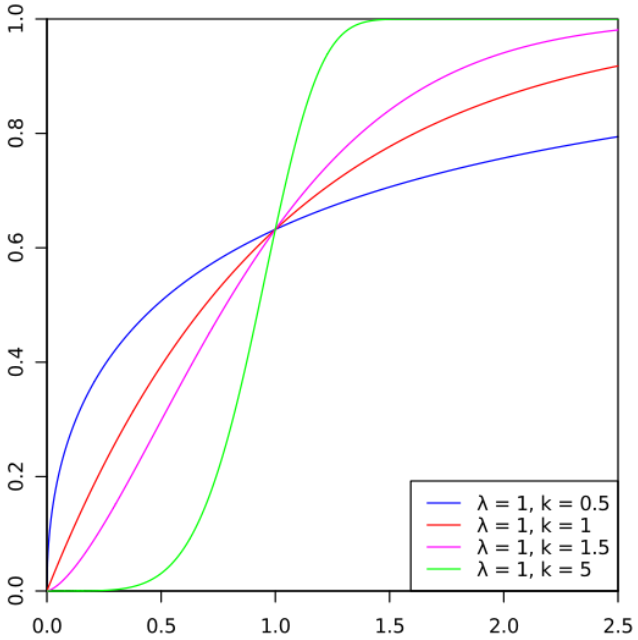
$$\hat{\lambda}^k = \frac{1}{n}\sum_{i=1}^{n} x_i^k$$

$$\lambda = \sqrt[k]{\hat{\lambda}^k}$$

Skirtinio formulė (angl. Cumulative distribution function) apskaičiuojama pagal:

$$F(x;k,\lambda) = \begin{cases} 1 - e^{-(x/\lambda)^k}, & x \geq 0, \\ 0, & x < 0; \end{cases}$$

Grafinis Weibull'o skirtinio vaizdas pavaizduotas pav. 1.



Pav. 1. Weibull'o skirstinio grafinis vaizdavimas [3].

Atliekame skaičiavimus Weibull'o tikėtinumo funkcijai gauti:

$$\sum_{i=1}^{n} \log\left(\frac{\lambda}{k}\right) - (k-1) * \log\left(\frac{x_i}{\lambda}\right) + \left(\frac{x_i}{\lambda}\right)^k$$

$$\log\left(\frac{x_i}{\lambda}\right) = \log x - \log\lambda$$

$$\sum_{i=1}^{n} \log\left(\frac{\lambda}{k}\right) - (k-1) * (\log x_i - \log\lambda) + \left(\frac{x_i}{\lambda}\right)^k$$

Po tam tikrų pertvarkymų turime galutinę logaritminės tikėtinumo funkcijos išraišką:

$$\Lambda = \log\left(\frac{\lambda}{k}\right) * n + n * (k-1) * \log\lambda - (k-1) * \sum_{i+1}^{n} \log x_i + \frac{\sum_{i=1}^{n} x_i^k}{\lambda^k}$$

2.1.3 Didžiausio tikėtinumo įverčiai

Matematikoje didžiausio tikėtinumo įverčiai (angl. maximum likelihood estimation trump. MLE*)* yra tikėtinumo funkcijos optimizavimo rezultatas, kad ji turi bent vieną fiksuotą tašką (taškas x, kuriam F(x)=x). Parametrų įverčiai parenkami taip, jog padidintų tikimybę, kad modeliu apibūdintas procesas sukūrė duomenis, kurie labiausiai tikėtini. Fiksuotam duomenų rinkiniui taikant statistinį modelį, MLE išrenka tokias modelio parametrų rinkinių reikšmes, kurios maksimizuoja tikėtinumo funkciją. Metaeuristinių algoritmų parametrų t skirstinio parametrams α įvertinti didžiausio tikėtinumo metodu atliekame šiuos veiksmus:

1. *Sudarome tikėtinumo funkciją;*
2. Apskaičiuojame tikėtinumo funkcijos dalines išvestines pagal visus parametrus;
3. Prilyginame rastas išvestines nuliui ir išsprendžiame gautą lygčių sistemą nagrinėjamų parametrų atžvilgiu;

4. Gautas $\hat{\alpha}$ laikomas ieškomaisiais įverčiais. [4],[5].

Taigi, sudarome funkciją g(x), kurioje palyginame Weibull'o skirstinį su Pareto skirstiniu.

$$g(x) = \left(\frac{\sum_{i=1}^{n} x_i}{n}\right)^2 / \frac{\sum_{i=1}^{n} (x_i)^2}{n},$$

čia n yra skirstinio dydis ir jis privalo tenkinti sąlygą n>0. X yra skirstinio elementas.

Jeigu gautas santykis yra daugiau negu 0.5, skirstinys bus artimesnis Pareto skirstiniui. Jei šis santykis yra mažesnis negu 0,5, tai bus Weibull'o skirstinys.

Atlikti skaičiavimai parodė, kad skirstinys yra panašesnis į Weibull'o skirstinį, nes gauta reikšmė yra mažesnė negu 0,5.

2.2 Metaeuristiniai algoritmai

2.2.1 Modeliuojamo atkaitinimo algoritmas

Modeliuojamojo atkaitinimo algoritmas atkaitinimo (angl. Simulated annealing trump. SA)formaliai kombinatorinio optimizavimo uždaviniams spręsti buvo pristatytas 1983 metais Kirkpatrik ir kitų autorių darbuose. Šio modelio pradininkai buvo Metropolis, Rosenbluth ir kiti (1953). [8] Šis algoritmas remiasi energetinių procesų, vykstančių sistemose, sudarytuose iš didelio skaičiaus dalelių, imitavimu. Pagrindinė algoritmo idėja – imituoti kūno atkaitinimą (grūdinimą), tai nurodo ir algoritmo pavadinimas . Modeliuojant sistemos atkaitinimą, iš pradžių sistemai suteikiama aukšta temperatūra, kuri palaipsniui mažinama, kol sistema pereina į stacionarią būseną (plienas užsigrūdina)]. Optimizavimo uždaviniui spręsti, sistemos perėjimas iš vieno energijos lygio (E1) į kitą (E2) priimamas su tikimybe :

$$P_E = \begin{cases} 1, & \Delta E < 0, \\ e^{-\frac{\Delta}{E}/C_B T_R}, & \Delta E \geq 0; \end{cases}$$

Čia T_e yra kontrolės parametras atitinkantis temperatūrą, , $\Delta E = E_2 - E_1$, o C_B – Bolcmano konstanta, kurią optimizavimo metu autoriai dažnai prilygina vienetui. [7].

Nuo temperatūros pasirinkimo priklauso atsakymo tikslumas. Jei pasirinksime per didelę temperatūrą, galime „prašokti" (nerasti) optimalaus sprendinio. Kita vertus pasirinkus mažą temperatūrą, tai algoritmas veiks vienoje vietoje ir ras tik lokalų optimumą. Taigi temperatūros pasirinkimas yra svarbus faktorius šiam algoritmui.

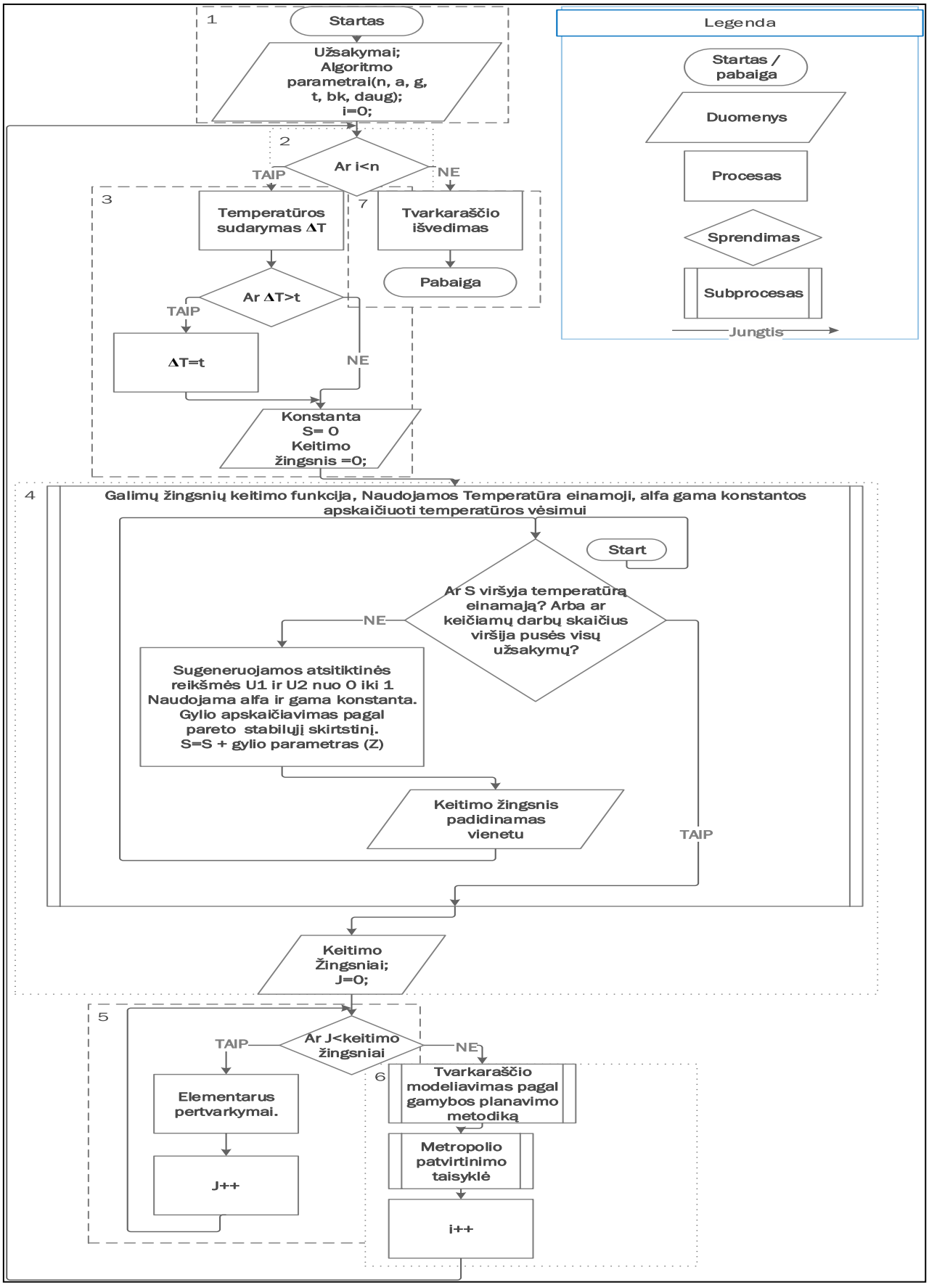
Atkaitinimo modeliavimo algoritmas praktiškai taikomas sprendžiant sudėtingas optimizavimo problemas įvairiose gyvenimiškose srityse. Vienas dažniausių atkaitinimo modeliavimo algoritmo taikymų – tvarkaraščių ir grafikų sudarymo uždavinių sprendime. Tokie uždaviniai reikalauja efektyvaus turimų resursų pozicionavimo laike ir erdvėje, kitaip tariant siekia optimizuoti tikslo funkciją. [6].

Tvarkaraščių sudarymui pritaikytas modeliuojamo algoritmas. Jo schema yra pateikiama (Pav. 2).

1-oje schemos dalyje yra nuskaitomi užsakymai su technologiniais medžių sąryšiais ir inicializuojami algoritmo parametrai kaip:

- *Iteracijų skaičiaus parametras n, kuris nusako, kada bus sustabdytas algoritmas;*
- *Pradinė temperatūra T_0, kuris naudojamas apskaičiuoti elementariųjų veiksmų kiekį;*
- *Daugiklio parametras, naudojamas apskaičiuoti elementariųjų veiksmų kiekį;*
- *Parametras γ naudojamas apskaičiuoti elementariųjų veiksmų kiekį;*
- *Bk parametras naudojamas apskaičiuoti elementariųjų veiksmų kiekį;*
- *Parametras α naudojamas skaičiuojant gylio parametrą.*

Schemos 2-ojoje dalyje yra tikrinamos ar yra pasiekta algoritmo stabdymo sąlyga.



Pav. 2 Modeliuojamo atkaitinimo algoritmo schema.

Algoritmo 3-ioje dalyje yra apskaičiuojama konstantinė temperatūra pagal formulę:

$$\Delta T = \frac{T_0 * daugiklis}{i^{\gamma}},$$

kur T_0 yra pradinė temperatūra, i yra esamos iteracijos numeris, daugiklis (angl. efficient) iš anksto nustatyta reikšmė ir γ iš anksto nustatyta reikšmė. Jei ΔT yra daugiau už T_0, tada ΔT yra priskiriama T_0, reikšmė. Kitu atveju yra paliekama apskaičiuota ΔT reikšmė.

4-oje schemos dalyje yra apskaičiuojamas iteracijoje leistinas elementariųjų keitimų skaičius. Šiam tikslui gauti yra skaičiuojamas gylio parametras, kuris yra gaunamas pagal Pareto stabilųjį reikšmes: Kol $S \leq (\Delta T * Bk)$ tada:

$$Z := \left(\frac{\sin((\alpha - 1)\cdot U1\cdot \pi)}{-\ln(U2)}\right)^{\frac{1-a}{a}} \cdot \frac{\cdot \sin(\alpha \cdot U1 \cdot \pi)}{\left(\cos(\alpha \cdot \frac{\pi}{2}) \cdot \sin(U1 \cdot \pi)\right)^{\frac{1}{\alpha}}};$$

$$S = S + Z;$$

Kur U1 ir U2 yra atsitiktinai sugeneruojamos reikšmės nuo 0 iki 1.

5-ajame schemos dalyje vykdomi elementariųjų pertvarkymo procesai, kuriuose atliekamos atsitiktiniu būdu sukeitimo (angl. swap) ir perkėlimo (angl. shift) operacijos. Perkėlimo (angl. Shift) metodas - tai vieno užsakymo perkėlimas užsakymų eilėje į kitą poziciją, taip sudarant naują užsakymų eilės sąrašą. O sukeitimo operacija leidžia du gamybos užsakymus sukeisti vietos tarpusavyje. Tokiomis operacijomis atlikti elementarieji pertvarkymai.

Schemos 6-ame žingsnyje yra tikrinamas skaičiavimo rezultatas naudojant metropolio patvirtinimo taisyklė, kurioje yra panaudotas Metropolis‘o-Hastings‘o (MH) kriterijaus algoritmas. Algoritmas leidžia modeliuojamo atkaitino algoritmui sumažinti tikimybę užstrigti lokaliame minimume. Pagal priimtiną santykį (galimybę) yra leidžiama modeliuojamo atkaitinimo algoritmui pereiti į blogesnį rezultatą teikiantį tvarkaraštį nepamirštant geriausiojo. Šis algoritmas sudarytas sudarytas iš šių žingsnių:

1. Pasirenkama pradinė reikšmė θ.

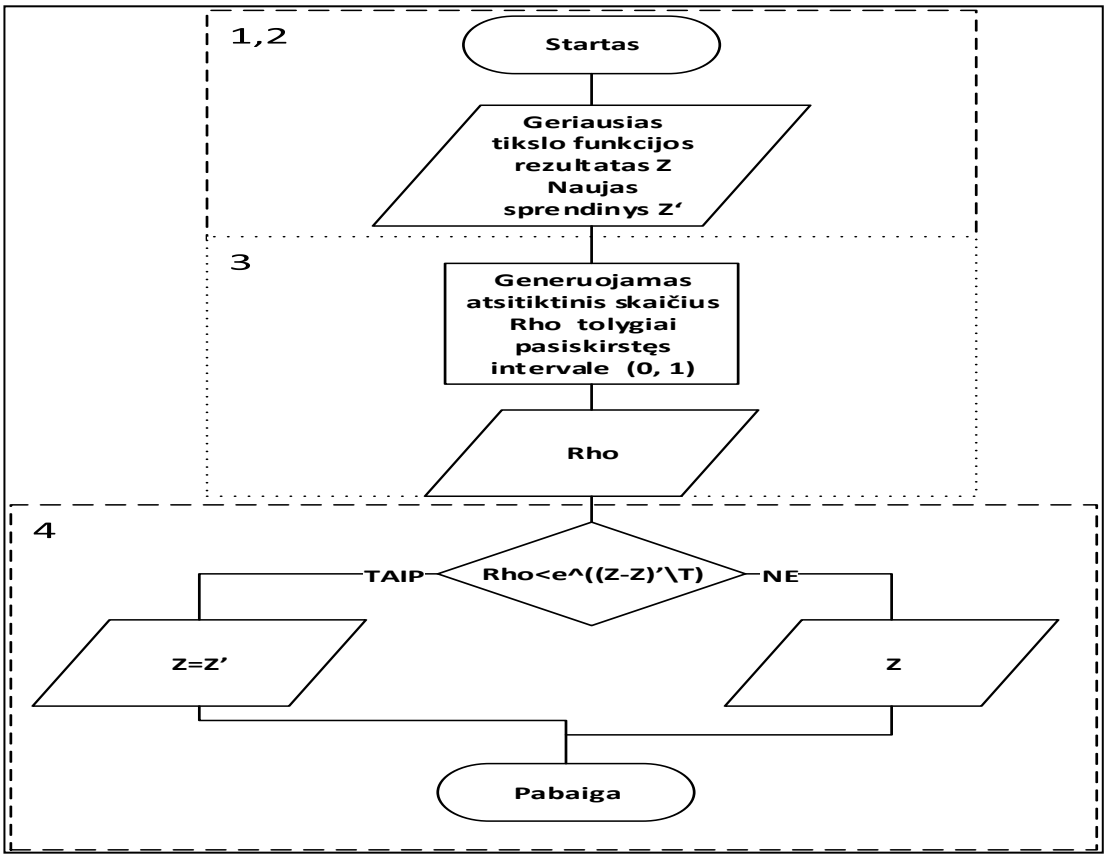
2. Iteracijoje t pasiūloma pereiti prie reikšmės θ* su tikimybe Jt(θ*| θ(t-1)).

3. Apskaičiuojama priimtinas santykis (galimybė):

$$r = \frac{p\left(\theta^0 \middle| y\right)/J_t(\theta^* | \theta^{(t-1)})}{p\left(\theta^{(t-1)} \middle| y\right)/J_t(\theta^{(t-1)} | \theta^*)}.$$

4. Priimame θ*kaip θ(t) su galimybe min(r, 1). Jei θ* nėra priimta, tada θ(t)= θ(t-1). [9].

Sukurto metropolio patvirtinimo taisyklės schema, kuri naudojama šeštojoje modeliuojamo atkaitinimo algoritmo dalyje yra pateikiama (Pav. 3).



Pav. 3 Metropolio patvirtinimo kriterijus.

7-toje schemos dalyje yra išvedamas geriausias rastas tvarkaraštis vykdymo metu.

2.2.2 Šakų ir ribų algoritmas

Šakų ir ribų algoritmas (angl. „Branch & Bound“ sutr. BAB) yra tikslusis algoritmas skirtas įvairiems optimizavimo uždaviniams spręsti. Pirmą kartą šis metodas pasiūlytas 1960 metais A.H. Land ir A. G. Doig.[12]. Šio algoritmo idėja yra skaldyk ir valdyk. Tai yra imamas abstrakti sprendinių aibė ir yra tikslinama kol gauname optimalų sprendinį. Pavyzdžiui, turime 4 užsakymus, kurios reikia sudėti eilės tvarką taip, kad užsakymus įvykdytų per trumpiausią laiką. Tvarkaraštį atitinka keturi užsakymus sudėti seka.

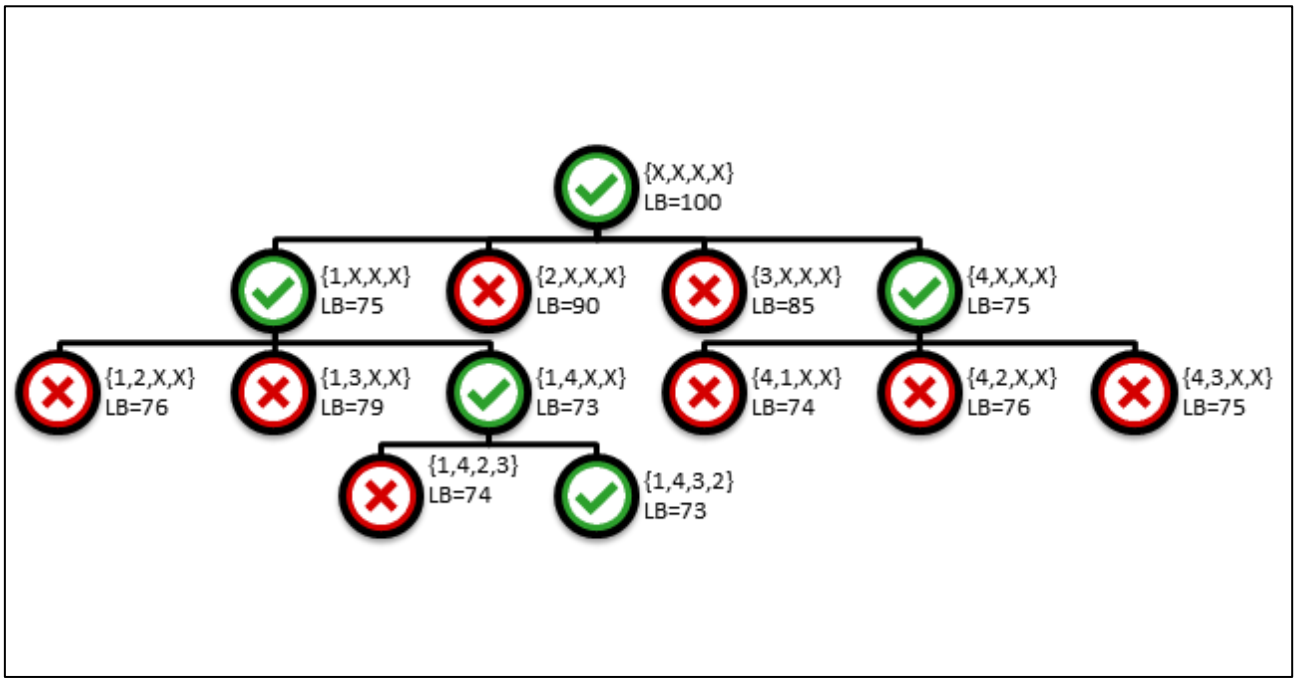
Inicijavimas: tvarkaraščio sugeneravimas be užduočių eiliškumo ir tai nurodome (*,*,*,*). Čia „*“ darbo sekoje rodo, kad nė vienas darbas nebuvo priskirtas į poziciją.

Žingsnis 1: Norėdami sukonstruoti tvarkarašti, pradedant nuo pirmos pozicijos, mes perkeliame iš mazgo (*,*,*,*) Į vieną iš keturių galimų mazgų (1,*,*,*), (2,*,*,*), (3,*,*,*), (4,*,*,*).

Žingsnis 2: Apskaičiuojama šakų apatinius rėžius. Priskiriame naujas užduotis tik tose šakose, kur yra mažiausias apatinis rėžis. Šaka (1,*,*,*) suteikia tris galimybes (1,2,*,*), (1,3,*,*) ir (1,4,*,*) ir taip toliau.

Žingsnis 3: kartojamas žingsnis 2, kol lieka dvi nepriskirtos užduotys. Likus dviem užduotims yra iškart fiksuojami su paskutine užduotimi. Šaka (1,4,*,*) davė geriausią apatinį rėžį tai, kad nereikėtų atlikti papildomo žingsnio yra iškart statomi abu paskutiniai darbai. Tai yra (1,4,2,3) ir (1,4,3,2). [10],[11]

Pavyzdys pateikiamas pav. 4.



Pav. 4 Šakų ir ribų algoritmo pavyzdys.

šakų ir ribų (angl. Branch and bound sutr. BAB) algoritmas. Algoritmo veikimo schema yra pateikta (Pav. 5).

Pirmoje schemos dalyje yra gaunami užsakymai. Pats algoritmas neturi savo parametrų. Taip pat yra suformuojamas tuščias vektorius į kurį rikiuosime užsakymus.

Antroje schemos dalyje tikrinama ar rikiuotas masyvas turi duomenų jei neturi keliaujama į patikrinimą ar yra sistemoje užsakymų, kurie turi iš anksto nustatytą vietą užsakymų sąraše.

Schemos 3-ioje dalyje yra skaičiuojamas kiekvieno užsakymo kritinis laikas.

4-oje schemos dalyje yra tikrinama kiek užsakymų yra suplanuotą. Jeigu liko du paskutiniai užsakymai, tada planuojami du variantai iškarto sumažinant algoritmo iteracijų skaičių.

Schemos 5-oje dalyje atliekamas tvarkaraščio sudarymas pagal planavimo logiką paskutinį kartą. Gauti du tvarkaraščiai palyginami tarpusavyje ir geriausias sudarytas tvarkaraštis atiduodamas.

6-toje schemos dalyje yra atliekamas nesuplanuotų užsakymo mažinimo operacija, prie kiekvienos neatmesto sprendinio šakos pridedamas po naują užsakymą. Tai yra daromas medžio šakojimas (vieno medžio lygio praplėtimas).

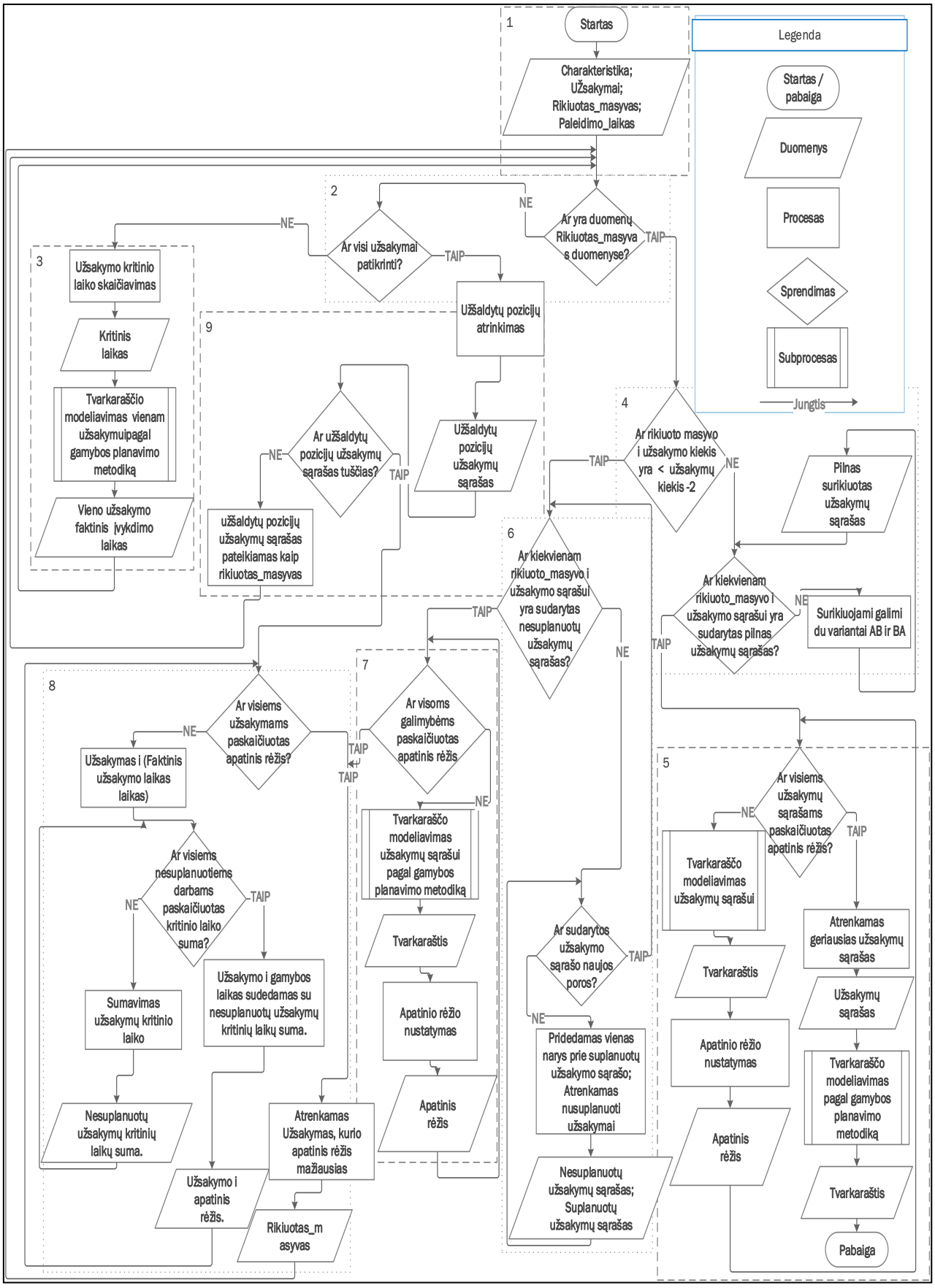
Septintoje schemos dalyje yra apskaičiuojama kiekvienos šakos apatinis rėžis reikšmė pagal formulę:

$$LB = \underbrace{\left(\sum_{i=1}^{k} T_i\right)}_{Suplanuotos\ užduotys} + \underbrace{\left(\sum_{i=k+1}^{n} T_i\right)}_{Nesuplanuotos\ užduotys} ,(9).[13].$$

kur k yra suplanuotų užduočių skaičius ir k priklauso n. N yra visų užduočių skaičius, o T yra laikas.

8-toji schemos dalyje yra apskaičiuojamas kiekvienam užsakymui suplanuotas laikas, jeigu tą užsakymą vykdysime patį pirmą, užsakymo procesai išdėliojami pagal tuo metu esamus žmogiškuosius ir techninius išteklių laisvumus. Tuo pačiu skaičiuojamas kiekvieno užsakymo nesuplanuotas užsakymų kritinis laikas pagal kiekvieno užsakymo įvykdymo laiką, jeigu yra planuojamas tik vienas užsakymas.

Schemos 9-toji dalyje yra tikrinama ar yra užsakymų, kurie turi iš anksto nustatyta vietą užsakymų sąraše. Jei užsakymų su iš anksto suplanuota užsakymo eilės tvarka yra, tada jie yra sudedami į užsakymų sąrašą pagal nurodyta tvarką.

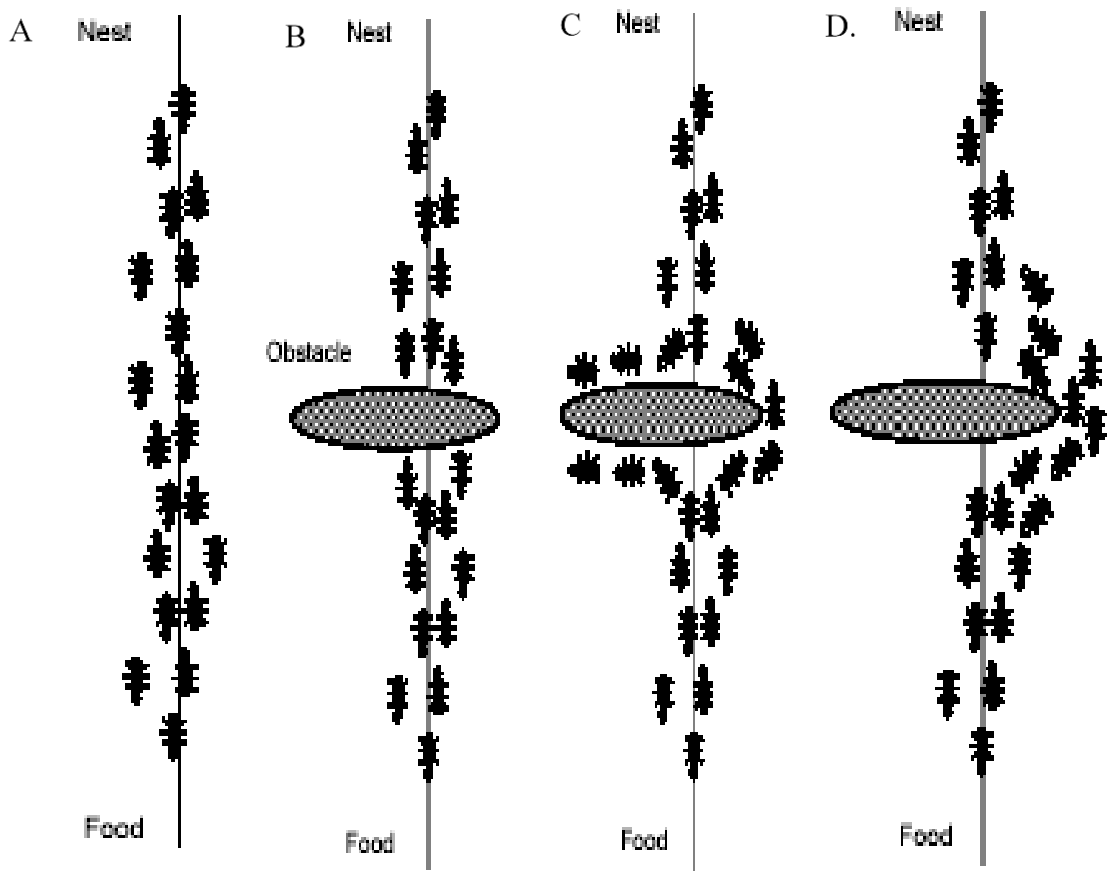


Pav. 5 Šakų ir ribų algoritmo schema.

2.2.3 Dirbtinių skruzdžių kolonijos algoritmas

Šešto dešimtmečio pabaigoje, septinto pradžioje imta domėtis vabzdžių kolektyviniu elgesiu. Pagal realų skruzdžių elgsena buvo sukurtas 1991 metais skruzdžių sistemos algoritmas. Šiuo metu skruzdžių kolonijų algoritmai turi daugiau negu 20 variacijų. [14].

Geriausiai skruzdžių elgesį parodo pavyzdys su kliūties įveikimu, kai kelių ilgiai yra skirtingi. Šis pavyzdys pateiktas pav. 6. Paveikslėlio A dalyje matome skruzdžių judėjimą nuo savo lizdo į maisto šaltinį be kliūčių. Tuomet padedame skruzdžių kelyje kliūti. Tai yra pavaizduota paveikslėlio B dalyje. Kai atsiranda skruzdžių kelyje kliūtis, skruzdės iš pradžių eis į kairę arba į dešinę praktiškai atsitiktiniu principu. Tai yra pateikta paveikslėlio D dalyje. Tačiau tos skruzdės, kurios pasirenka trumpesnį kelią, kelionėje užtruks trumpiau ir šiuo maršrutu per tą patį laiką pereis daugiau skruzdžių, o tai reiškia, kad feromonų šiame maršrute bus palikta daugiau. Taigi šis kelias taps patrauklesnis ir kitoms skruzdėms. Tai yra pavaizduota paveikslėlio D dalyje.



Pav. 6 Skruzdžių kolonijos algoritmo veikimo principas.

Kronologija apie skruzdžių kolonijos optimizavimo algoritmus:

- 1959, Piere-Paul Grasse išrado stigmatizmo teoriją paaiškinti lizdų pastato terminus terminituose [15].

- 1983 Deneubourg ir jo kolegos tyrė kolektyvinį skruzdžių elgesį.[16]
- 1989 Goss, Aron, Deneubourg ir Pasteels pristatytas darbas „ kolektyvinis elgesys Argentinos skruzdžių“ (angl. collective behavior of Argentine ants**)**, pateikė idėja apie skruzdžių kolonijos optimizavimo algoritmą. [17]
- 1989 metais įgyvendintas maisto elgesio modelis pagal Ebling ir jo kolegas. [18]
- 1991 metais M. Doringo pasiūlė skruzdžių sistemos algoritmą savo doktorantūros disertacijoje (kurios buvo publikuotos 1992). Techninė ataskaita ištraukta iš ataskaitos ir parašyta kartu su bendraautoriais V. Maniezzo ir A. Colorni buvo publikuota 1996 metais. [19]
- 1996 metais publikuotas straipsnis apie skruzdžių sistemą. 20]
- 1996 metais Hoos ir Stützle išrado max-min skruzdžių sistemą. [21]
- 1997 metais Dorigo ir Gambardella paviešino skruzdžių kolonijos sistemos algoritmą. [22]
- 1998 metais Doringo pradeda pirmąją konferencija skirta ACO algorimtams. [23]
- 1998 metais Stützle pasiūlo pradinius lygiagrečius įgyvendinimus. [24]
- 2000 metais atsiranda pirmosios programos sprendžiančios planavimo, planavimo sekos ir apribojimų tenkinimui pagal skruzdžių algoritmus.
- 2000 metais Gutjahr pateikia pirmus įrodymus dėl skruzdžių kolonijos algoritmo konvergavimo.[25]
- 2001 metais skruzdžių kolonijų algoritmus pradėjo naudoti industrijoje kompanijos kaip Eurobios ir AntOptima.
- 2001 metais Iredi ir jo kolegos publikavo pirmą daugiakriterinį algoritmą. [26]
- 2002 metais pasirodo pirmosios programos pritaikytos tvarkaraščiams ir Bayeso tinkluose.
- 2002 metai Bianchi ir jos kolegos pasiūlė algoritmo atmainą spręsti stochastines problemas. [27]
- 2004 metais Zlochin ir Dorigo įrodo, kad kai kurie skruzdžių kolonijos algoritmai yra lygia vertus stochastiniam gradiento nusileidimo, kryžminimo entropijos metodams ir algoritmams skirtiems įvertinti pasiskirstymą.[28]

- 2012 metais Prabhakar ir jo kolegos paviešino tyrimus susijusius su atskirų skruzdžių komunikacija nesinaudojant feromonais. Ši komunikacija atspindi kompiuterinių tinklų organizavimo principus. Komunikacijos modelis buvo palygintas su perdavimo valdymo protokolu. [29]
- 2017 metais sėkmingas daugiakriterinių sprendimų priėmimo metodas PRONETHEE integravimas į ACO algoritmą (HUMANT algoritmas).[30]

2.2.3.1 Panašumai ir skirtumai su tikromis skruzdėmis.

Dauguma minčių apie skruzdžių kolonijos optimizavimą kyla iš tikrų skruzdžių. Ypač tokios:

1) bendradarbiaujančių individų kolonija,

2) (dirbtinis) feromonas, kuris skirtas vietiniam susisiekimui,

3) vietinių žingsnių seka, surandant trumpiausius kelius,

4) stochastinė sprendimo politika, naudojant vietinę informaciją.

Bendradarbiaujančių individų kolonija. Kaip ir tikros skruzdžių kolonijos, skruzdžių algoritmai taip pat yra sudaryti iš gyventojų, ar kolonijos, ir asinchroniško[1]objekto, pasauliniu mastu bendradarbiaujančio, kad surastų gerą uždavinio sprendinį. Nors kiekvienos dirbtinės skruzdės sunkumas yra toks, kad ji gali sugalvoti įmanomą sprendinį (kadangi tikra skruzdė gali surasti kaip nors kelią tarp lizdo ir maisto), aukštos kokybės sprendiniai yra rezultatas bendradarbiavimo tarp visos kolonijos individų. Skruzdės bendradarbiauja informacijos pagalba, jos tuo pačiu metu skaito ir rašo, sprendžiant teritorijų problemas, kurias jos aplanko.

Feromono pėdsakas ir kvapas. Dirbtinės skruzdės pakeičia tam tikrus savo aplinkos aspektus, kadangi tikros skruzdės tai irgi daro. Tuo metu, kai tikros skruzdės palieka pasaulinę dalį, jos eina prie cheminės medžiagos feromono. Dirbtinės skruzdės keičia nedidelį skaitmeninės informacijos kiekį, išsaugotos problemos dalyje. Ši informacija atsižvelgia į skruzdės einamąją "istoriją" ir gali būti skaitoma ir rašoma bet kokios gaunančios prieigą skruzdės. Pagal analogiją, mes vadiname šią skaitmeninę informaciją dirbtiniu feromono keliu. Skruzdžių kolonijos optimizavimo algoritmuose vietiniai feromono keliai yra vieninteliai susisiekimo kanalai tarp skruzdžių. Ši susisiekimo pagal kvapus forma vaidina pagrindinį vaidmenį kolektyvinių žinių panaudojime. Jos svarbiausias tikslas yra pakeisti būdą, kurio pagalba, aplinka (problemos peizažas) yra skruzdžių suvokta kaip visos senos skruzdžių kolonijos istorijos funkcija. Paprastai skruzdžių kolonijos

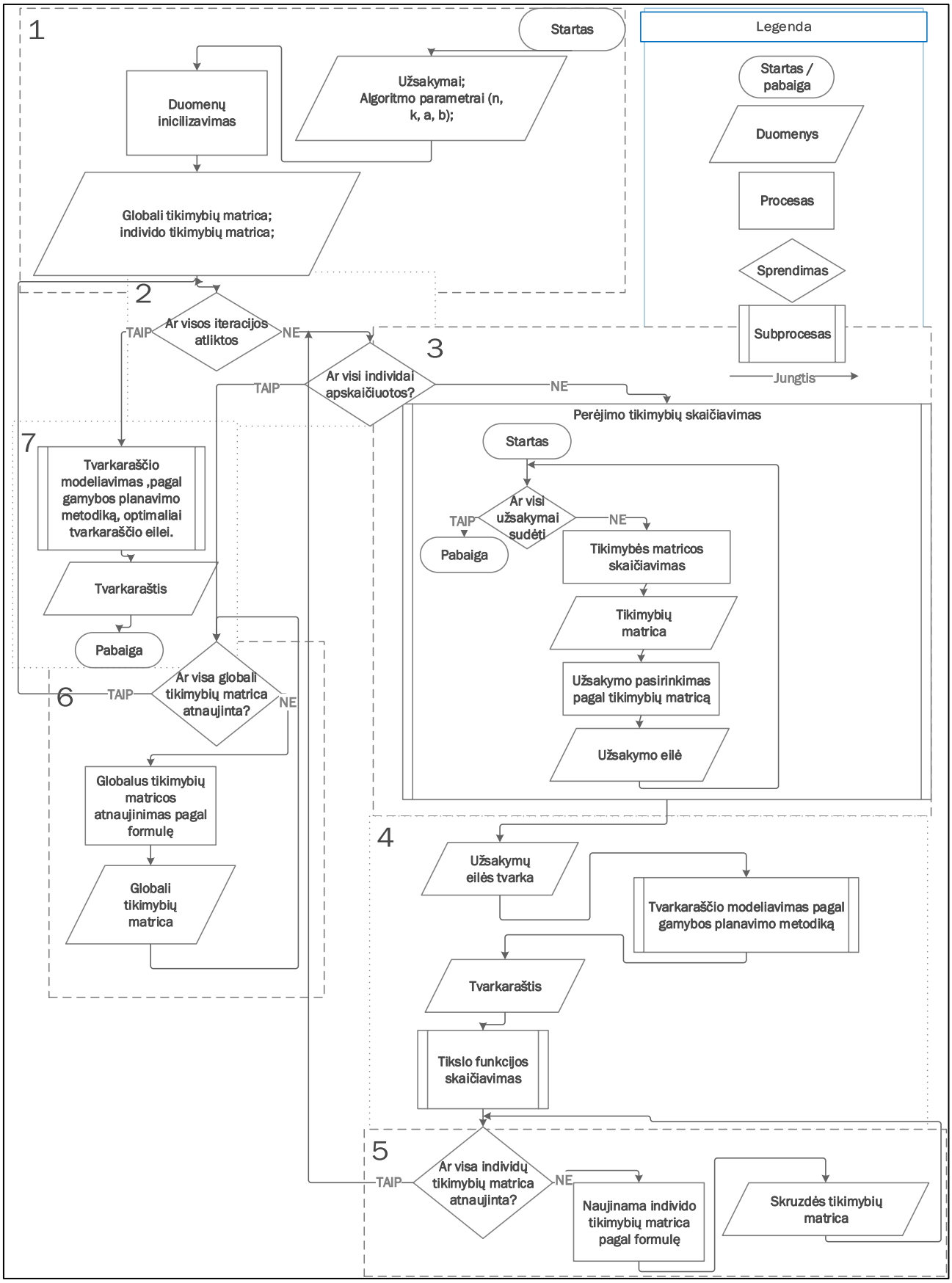
[1] asinchronija [⇑ a… + gr. synchronos – vienalaikis]: nesutapimas laiko atžvilgiu, nevienalaikiškumas.

optimizavimo algoritmuose išgaravimo mechanizmas, panašus į tikrą feromono išgaravimą, keičia feromono informaciją kas tam tikrą laiką. Feromono išgaravimas leidžia skruzdžių kolonijai lėtai užmiršti savo seną "istoriją", kad ji galėtų nukreipti savo paiešką į naujas kryptis.

Trumpiausio kelio paieška ir vietiniai žingsniai. Dirbtinės ir tikros skruzdės turi bendrą užduotį: surasti trumpą kelią, jungiantį lizdą su paskirta maršruto vieta. Tikros skruzdės nešokinėja, jos tik eina per gretimos vietovės dalis, ir taip daro dirbtinės skruzdės, judėdamos žingsnis po žingsnio per “gretimos vietovės dalis”, apeinant problemas.

Stochastinė ir trumparegė dalies perėjimo elgsena. Dirbtinės skruzdės, kaip ir tikros skruzdės, daro sprendimus, taikant tikimybės sprendimo elgseną, kad pajudėtų per gretimas vietovės dalis. Kaip ir tikroms skruzdėms, dirbtinių skruzdžių elgsena naudoja vietinę informaciją, ir ji nenaudoja "galvojimo į ateitį" metodo, kad numatytų būsimas vietovės dalis. Todėl, taikomoji elgsena yra visiškai vietinė, erdvėje ir laike. Elgsena yra abiejų apriorinė informacinė funkcija, kuri padeda sprendžiant problemos specifikaciją (ekvivalentiškas tikrų skruzdžių vietovės struktūrai).

Iš skruzdžių kolonijos klasės metaeuristinių metodų, buvo pasirinktas realizuoti dažniausiai praktikoje sutinkamas skruzdžių kolonijos algoritmas (angl. Ant colony algorithm sutr. ACA). Sukurto algoritmo schema pateikiama (pav. 7).



Pav. 7 Dirbtinių skruzdžių kolonijos algoritmo schema.

Schemos dalyje nr. 1 yra atliekamas duomenų inicializavimas. Nuskaitomi užsakymai su technologiniais medžių sąryšiais. Pradiniai algoritmo parametrai parenkami darbe sukurta metodika, kuri leidžia atsižvelgti į anksčiau išspręstų uždavinių rezultatus. Uždavinių pradinių parametrų metodika yra pateikiame skyriuje 5.3 (Pradinių metaeuristinių algoritmų parametrų rinkinių parinkimo metodologija). Metaeuristinis algoritmas konfigūruojamas pagal pasirinktus parametrus, o jo skaičiavimų kokybė yra vertinama pagal tokį charakteristikų rinkinį:

- *Iteracijų atliktų iki algoritmo sustabdymo, skaičius n.*
- *individų (dirbtinių skruzdžių) skaičius k, kuris reprezentuoja populiacijos dydį, tai yra pasakomas kiek vienoje iteracijoje bus atliekama nepriklausomų sprendinių variantų.*
- *parametras α, kuris nurodo euristinių atstumo (skruzdžių įveikto kelio ilgis) įtaką sprendimui. Nustačius α=0, bus pasinaudojama tik tikimybine informacija.*
- *Parametras β, kuris nurodo tikimybės (feromonų kiekio) įtaką sprendimui. Nustačius jį 0, bus naudojama tik atstumu informaciją.*

Taip pat šioje dalyje yra sukonstruojama kiekvienam individui sprendinių matricą G, kur stulpelis nurodo individas, o eilutė nurodo užsakymą. Taip pat suteikiama kiekvienam G matricos komponentui tam tikra tikimybė τ. Kiekvienai iš *k* individų atsitiktinai priskiriama starto pozicija, kuri įtraukiama į tabu sąrašą (draudžiamų perėjimų aibė).

2-oje schemos dalyje yra tikrinamos ar yra pasiekta algoritmo stabdymo sąlyga ir ar yra atlikti visi atskirieji sprendimai.

3-ioje schemos dalyje yra skaičiuojama tikimybė (feromonų kiekis) pagal G matricoje nurodyta briauną (*i,j*). Kiekvienas individas taikys pasirinktą perėjimų tikimybės formulę (žr., (1)) kol galiausiai surandamas tinkamas sprendinys. Kai kiekvieno individo tabu sąrašas tampa užpildytas, jo nueitas atstumas įvertinamas pagal funkciją $\Phi_{\Psi}(t)$, ir geriausias rezultatas išsaugomas. Feromonų kiekis pagal briauną (*i,j*) perskaičiuojamas remiantis formule (1) .

Kiekvieno individo sprendinys parenkamas pagal tikimybe $\tau_{i,j}$, priskiriamą matricos G briaunai (*i,j*), tačiau sprendimui priimti taip pat naudojamas ir euristinis atstumas $\eta_{i,j}$ atitinkantis briauną *(i,j),* čia perėjimo tikimybė iš *i* į *j* elementą *k*-tajai skruzdei laiko momentu *t* apibrėžiama :

$$P^{k}{}_{i,j} = \begin{cases} \dfrac{[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}]^{\beta}}{\sum_{k \in leidžiami_k}[\tau_{i,k}(t)]^{\alpha}[\eta_{i,k}]^{\beta}} \; jei (i,j) \notin tabu_k \\ 0 \end{cases}, (1).$$

kur $tabu_k$ yra draudžiamų perėjimų aibė, o $k = \{C - tabu_k\}$. Parametrai α ir β yra nustatomi eksperto, įvertinant tikimybes ir euristinis atstumų įtaka sprendimui. Nustačius $\beta = 0$, bus naudojama tik atstumo informacija ir atvirkščiai.

4-oje schemos dalyje, sudaryta užsakymų eilės tvarka yra perduodama į gamybos planavimo pagal metodiką algoritmą, kuriame yra sukonstruojamas tvarkaraštis. Gavus tvarkaraštį yra skaičiuojama plano tikslo funkcija.

5-oje schemos dalyje yra apskaičiuojamos kiekvieno k-tojo individo tikimybės $\Delta\tau^k{}_{i,j}$, skaičiuojamas pagal formulę:

$$\Delta\tau^k{}_{i,j} = \begin{cases} \frac{Q}{L_k} \; jei \; k-toji \; skruzdė \; keliauja \; (i,j) \; briauna \\ 0 \end{cases}, (2).$$

čia L_k yra *k*-tojo individo kelio ilgis tarp laiko momentų *t* ir *t+n,* Q yra teigiama konstanta. Taigi, tikimybė Q/L yra perskaičiuojama kiekvienoje iteracijoje.

6-oje schemos dalyje skaičiuojamos individo tikimybės paliktas tikimybių atnaujinimo pavėlinimo principu. Jis išreiškiamas pagal funkciją:

$$\tau_{i,j}(t+n) = \rho \cdot \tau_{i,j} + \Delta\tau_{i,j} \; (3).$$

čia ρ yra koeficientas, parenkamas taip, kad $(1-\rho)$ reprezentuotų tikimybės nykimo (feromonų *išgaravimo koeficientą)* pagal briauną (*i,j*) tarp laiko momento *t* ir *t+n*. Kad tikimybės neaugtų per greitai, turėtų galioti $\rho \in (0,1)$ .Visos tikimybės, gautos *m* skruzdžių $\Delta\tau_{i,j}$ yra apskaičiuojamos tokiu būdu :

$$\Delta\tau_{i,j} = (1-p) * \tau_{i,j} + \sum_{k=1}^{m} \Delta\tau^k{}_{i,j} \; (4).$$

7-toje schemos dalyje yra išsaugomas geriausias surastas tvarkaraštis.

2.3 Tyrimai

Vykdant metaeuristikų tyrimą ribotų gamybos išteklių planavimo uždaviniams lanksčiai ir kompleksiškai spręsti buvo sumodeliuoti šie uždaviniai:

- *Dviejų darbo linijų uždavinys;*
- *PSPLIB bibliotekos ribotų išteklių tvarkaraščių sudarymo uždavinių sprendimas;*
- *Planavimo uždavinys su sąryšiais.*

Dviejų darbo linijų uždavinyje turime du darbo resursus (konvejerius), kuriems skirstome darbus (užsakymus). Planuojamų darbų kiekis yra 100 vienetų. Kiekvienam darbui yra sugeneruojama atsitiktinė trukmė pagal Gauso stabilųjį skirstinį. Jeigu darbą atlieka pirmoji darbo linija, jo atlikti antroji nebegali. Visus darbus konvejeriams reikia paskirti taip, kad minimizuotume užsakymų įvykdymo darbo laiką. Darbo linijos yra vienodo pajėgumo.
Taip pat buvo spręsta 600 uždavinių iš PSPLib bibliotekos, kur kiekviename uždavinyje yra po 120 procesų, kurie gali naudoti 4 resursus. Kiekvieno proceso įvykdymo laikas ir resursų panaudojimas yra skirtingas.

Trečiasis sumodeliuotas uždavinys yra su sąryšiais. Šis sudarytas uždavinys artimas realiems atvejams. Uždavinyje užsakymai turi bendrą technologinį medį su vienuolika procesų. Procesams buvo sugeneruotos atsitiktinės vykdymo trukmės pagal Gauso stabilųjį skirstinį, intervale nuo 100 iki 200 minučių. Taip pat kiekvienas užsakymas turėjo skirtingą pageidautiną užbaigimo datą, sugeneruotas realus darbo kalendorius.

2.3.1 Dviejų darbo linijų uždavinio tyrimas

Šie uždaviniai buvo spręsti dviem modeliuojamo atkaitinimo algoritmo modifikacijomis. Pirmojoje modifikacijoje naudojame Pareto stabilųjį dėsnį elementariųjų pertvarkymo kiekiui generuoti. Antroje modifikacijoje elementariųjų pertvarkymo kiekis yra konstanta.
Naudojant pirmąją modifikaciją, papildomai sukurtos dvi veikimo variacijos. Pirmoje variacijoje algoritmas atlieka elementariuosius pertvarkymus atsitiktinai, neatsižvelgiama į praeityje įvykusius pertvarkymus. Antroje variacijoje algoritmas, prieš atliekant elementariuosius pertvarkymus, tikrina, kad paimtas darbas nebūtų įdėtas atgal į buvusią darbo liniją.
Naudojant antrą modifikaciją, buvo tiriamos keturios variacijos. Jas vadiname 10, 20, 35, 50 variacijomis. Skaičius nurodo, kiek elementariųjų pertvarkymų įvyks vienoje iteracijoje. Kiekviena modeliuojamo atkaitinimo algoritmo variacija buvo kartojama po 2000 bandymų ir po 20000 iteracijų. Prieš atliekant bandymus, kiekvienai variacijai buvo parinkti algoritmo parametrai. Kad rezultatai būtų tikslesni, buvo atlikta po 2000 bandymų kiekvienu metodu. Prieš atliekant metodų rezultatų lyginimą tarpusavyje, buvo nugludinti kiekvienam metodui parametrai.

- „Atsitiktiniu“ metodu sprendžiant parametrai buvo: T0: 40 Daug: 15 alfa: 0,85 gama: 0,95 BK: 200;
- „Priverstiniu“ metodu sprendžiant parametrai buvo: T0: 35 Daug: 20 alfa: 0,85 gama: 0,95 BK: 100;

- „10“ elementariųjų pertvarkymų iteracijoje metodu sprendžiant parametrai buvo: T0: 40 Daug: 20 alfa: 0,85 gama: 0,95 BK: 300;
- „20“ elementariųjų pertvarkymų iteracijoje metodu sprendžiant parametrai buvo: T0: 40 Daug: 20 alfa: 0,85 gama: 0,95 BK: 150;
- „35“ elementariųjų pertvarkymų iteracijoje metodu sprendžiant parametrai buvo: T0: 40 Daug: 20 alfa: 0,85 gama: 0,95 BK: 100;
- „50“ elementariųjų pertvarkymų iteracijoje metodu sprendžiant parametrai buvo: T0: 40 Daug: 20 alfa: 0,85 gama: 0,95 BK: 200.

Variacijų rezultatai buvo registruojami šiose iteracijose: 1, 10, 100, 1000, 2000, 5000, 1000, 15000 ir 20000.

2.3.1.1 Šuolių vidurkis

Lentelėje Nr. 1 yra pateikti duomenys apie kiekvieno metodo šuolius (elementariuosius pertvarkymus) fiksuotose iteracijose. Fiksuotų metodų šuoliai nesikeičia, nes jie buvo užfiksuoti iš anksto. Siūlomų metodų šuolių vidurkis per 20000 iteracijų sumažėjo nuo 50 šuolių per iteraciją 7 kartais.

Lentelė Nr. 1. Šuolių vidurkio lentelė

Šuolių vidurkis									
	1	10	100	1000	2000	5000	10000	15000	20000
Atsitiktinis	49,47	49,53	48,86	34,39	24,61	14,57	9,49	8,16	7,35
Priverstinis	49,02	49,07	47,99	26,27	17,92	10,27	6,62	5,12	4,27
„10“	10	10	10	10	10	10	10	10	10
„20“	20	20	20	20	20	20	20	20	20
„35“	35	35	35	35	35	35	35	35	35
„50“	50	50	50	50	50	50	50	50	50

2.3.1.2 Globališkumo Parametras

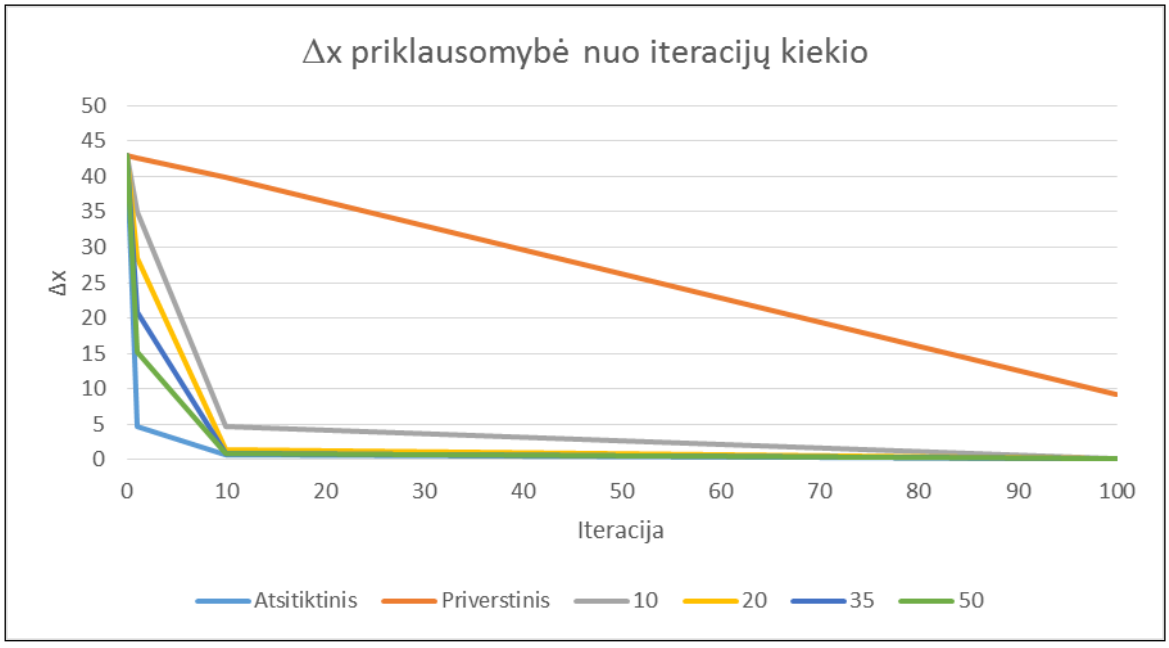
Globališkumo parametras – tai rodiklis, kuris teikia informacijos apie tai, kiek Metropolis‘o-Hastings‘o taisyklė leidžia pereiti į blogesnį rezultatą. Kaip matome pagal lentelę Nr. 2, kiekvienam tikrintame metode su atliktų iteracijų skaičiumi patekti į blogesnį rezultatą šansų vis mažėja.

Lentelė Nr. 2. Globališkumo parametro lentelė

Metropolis (globališkumo parametras)									
	1	10	100	1000	2000	5000	10000	15000	20000
Atsitiktinis	0	0,423	0,43	0,22	0,15	0,08	0,05	0,04	0,03
Priverstinis	0,95	0,48	0,46	0,26	0,18	0,1	0,06	0,04	0,03
„10“	0,001	0,17	0,43	0,27	0,19	0,1	0,06	0,05	0,03
„20“	0	0,28	0,44	0,26	0,18	0,10	0,07	0,05	0,04
„35“	0	3,44	4,4	0,26	0,18	0,1	0,06	0,05	0,04
„50“	0	0,37	0,44	0,26	0,18	0,1	0,06	0,05	0,04

2.3.1.3 Funkcijos minimumo tyrimas

Lentelėje Nr. 6 yra pateikti duomenys apie kiekvieno metodo Weibulo skirstinio λ reikšmę, K reikšmę, Standartinį nuokrypi, vidurkį medianą, funkcijos minimalią reikšmę ir maksimalią. Iš lentelėje Nr. 6 pateiktų duomenų pastebime, kad K reikšmė dažniausiai tampa 1. Tai reiškia, kad funkcija virsta eksponentine funkcija. Λ rodiklis nurodo funkcijos mastelį.

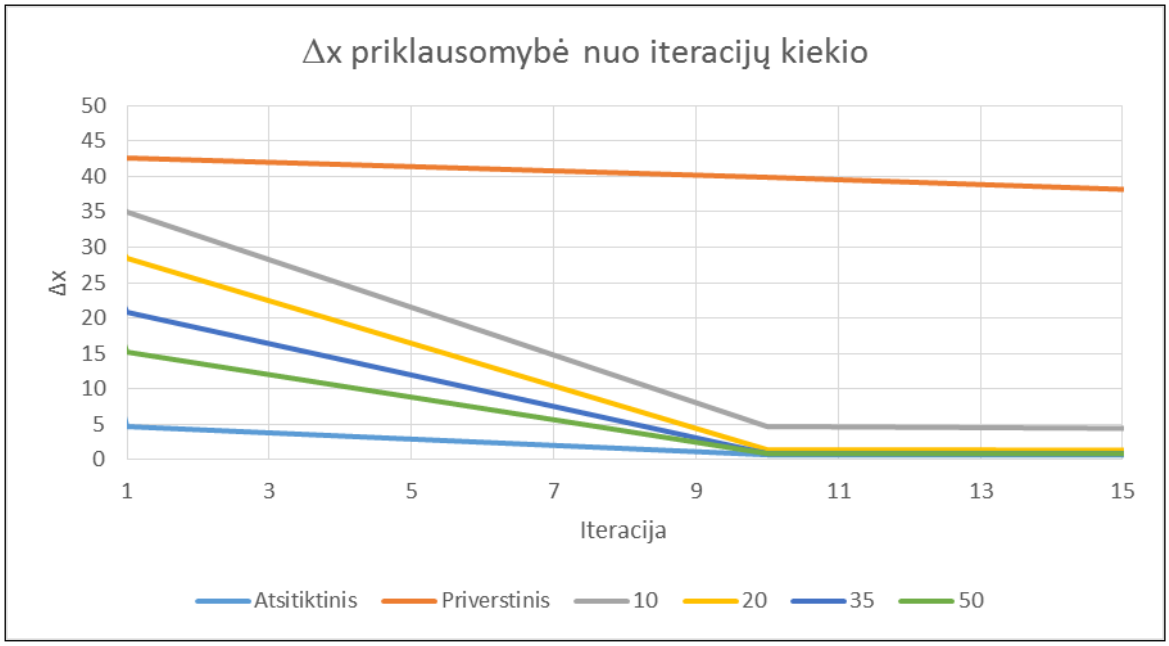


Pav. 8. Δx priklausomybės nuo iteracijų kiekio iki 100 iteracijos grafikas

Pav. 8 pateikiamas 6 variacijų *Δx* priklausomybės nuo iteracijų kiekio grafikas. *Δx* tai :

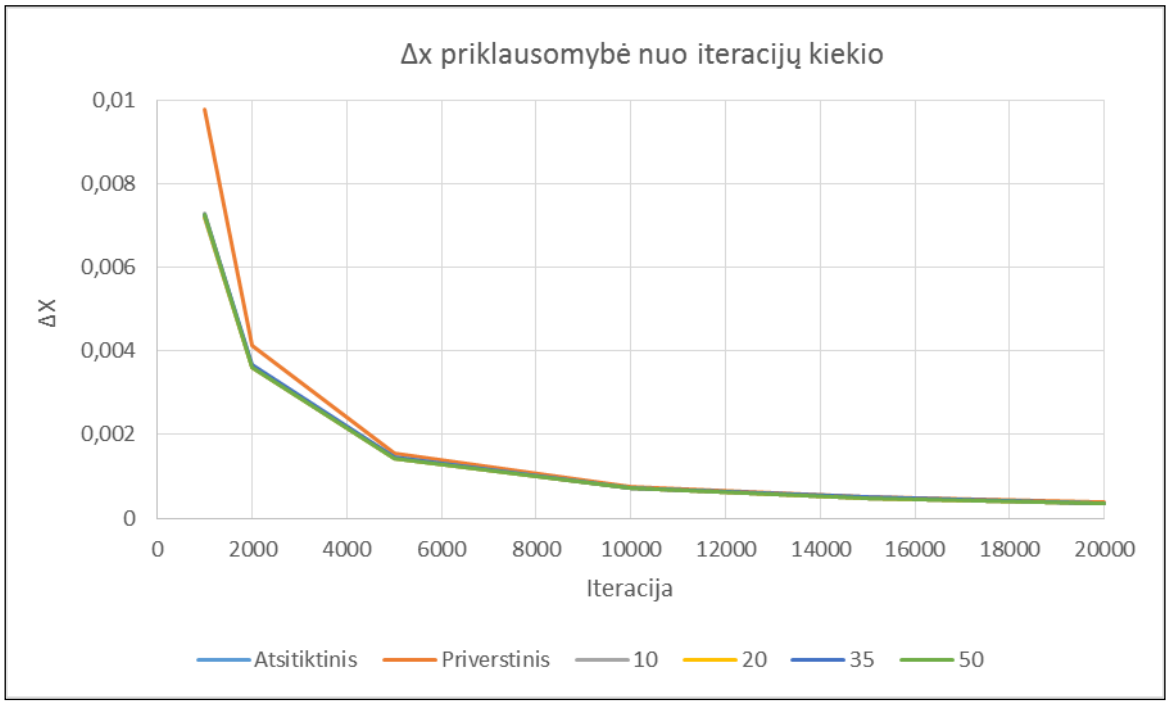
$$\Delta x = \frac{\sum_{i=0}^{n} F(x) - opt}{n},$$

čia *F(x)* yra tikslo funkcijos reikšmė, *opt* – tai sprendinio optimumo reikšmė, o *n* – atliktų bandymų skaičius. Pav. 8 matoma, kad sukurta modeliuojamo atkaitinimo „priverstinė“ variacija nuo „10“ variacijos po šimtosios iteracijos skiriasi 113 kartų. Visos variacijos, išskyrus „priverstinė“, po šimtosios iteracijos teikia *Δx* < 0,082. „Atsitiktinė“ variacija per 100 iteracijų sugeba priartėti 99,83 % prie optimalios reikšmės nuo pradinio rezultato.



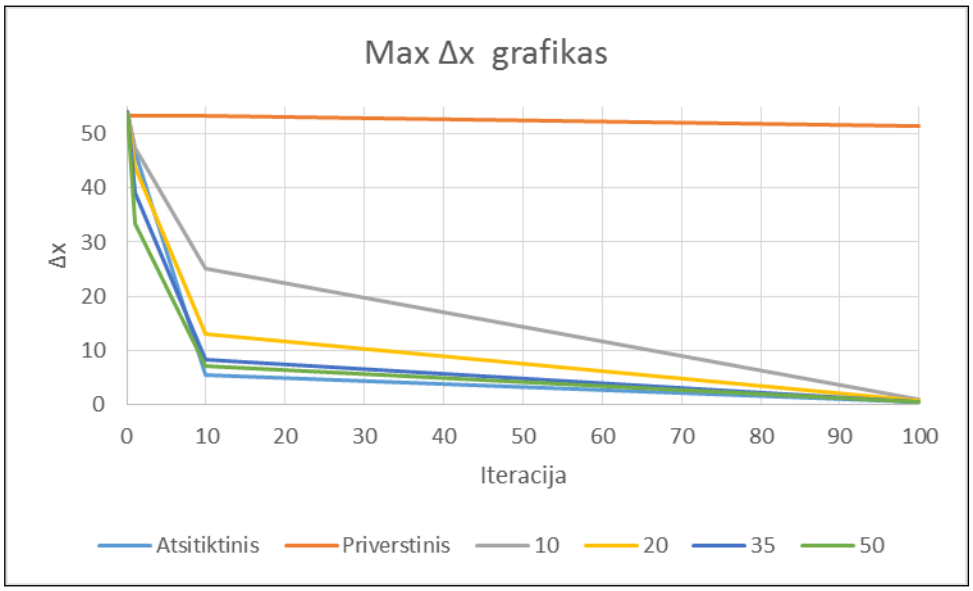
Pav. 9. Δx priklausomybės nuo 1 iki 15 iteracijos grafikas

Pagal pav. 9 grafiką matoma, kad „atsitiktinė“ variacija po 10 iteracijos pateikia geriausią rezultatą Δx = 0,6719. Po dešimtosios veikimo iteracijos „atsitiktinė“, „50“ ir „35“ variacijos Δx < 0,9. Variacija „20“ Δx >1,3. Variacija „20“ demonstruoja 44 % prastesnius rezultatus.



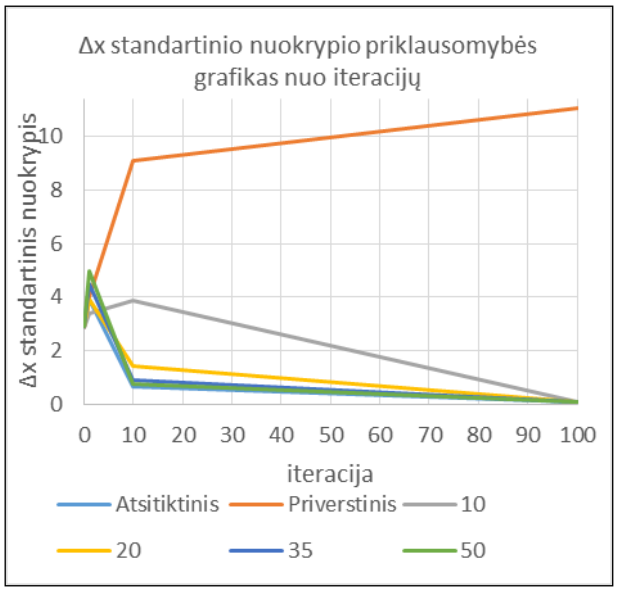
Pav. 10. Δx priklausomybės nuo 1000 iki 20000 iteracijų grafikas

Pav. 10 pateikiamas Δx priklausomybės nuo 1000 iki 20000 iteracijų grafikas. Po 5000 iteracijų teikiami variacijų duomenys skiriasi vieni nuo kitų per mažiau negu 1,06 * 10^{-4} vienetų. Pagal pav. 8, pav. 9 ir pav. 10 informaciją, galima daryti išvadą, kad visos modeliuojamo atkaitinimo variacijos konverguoja, nes Δx rezultatai artėja į 0. Modeliuojamo atkaitinimo „atsitiktinė“ variacija iki 2000 iteracijos demonstruoja geriausią rezultatą, tačiau nuo 2000 iki 20000 iteracijos visos variacijos demonstruoją labai artimus rezultatus.

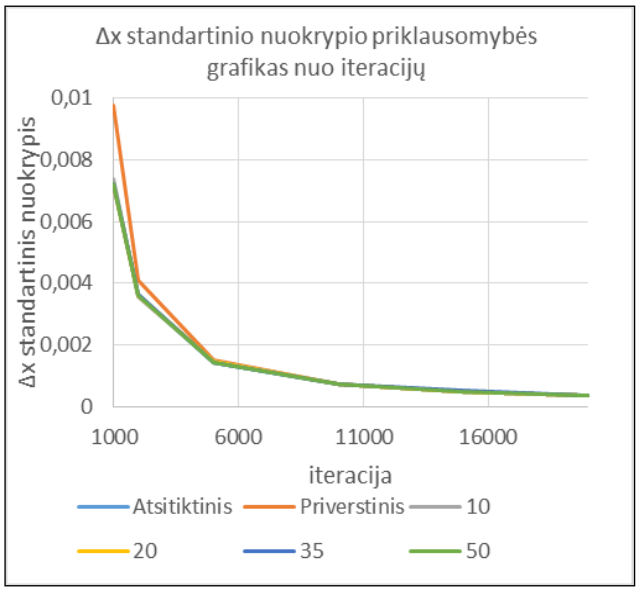


Pav. 11. Minimalus Δx pagerėjimo priklausomybės grafikas iki 100 iteracijos

Pav. 11 yra pateiktas iš 2000 atliktų bandymų pasiektos blogiausios Δx reikšmės fiksuotose iteracijose kitimo grafikas. Tai rodo prasčiausią užfiksuotą Δx reikšmę registruotose vykdymo iteracijose. Pastebima, kad „atsitiktinė“ variacija nuo 10 iki 100 iteracijos veikia efektyviau, lyginant su kitomis variacijomis. Variacija, „50“ nuo 1 iki 9 iteracijose pateikė geriausius rezultatus.

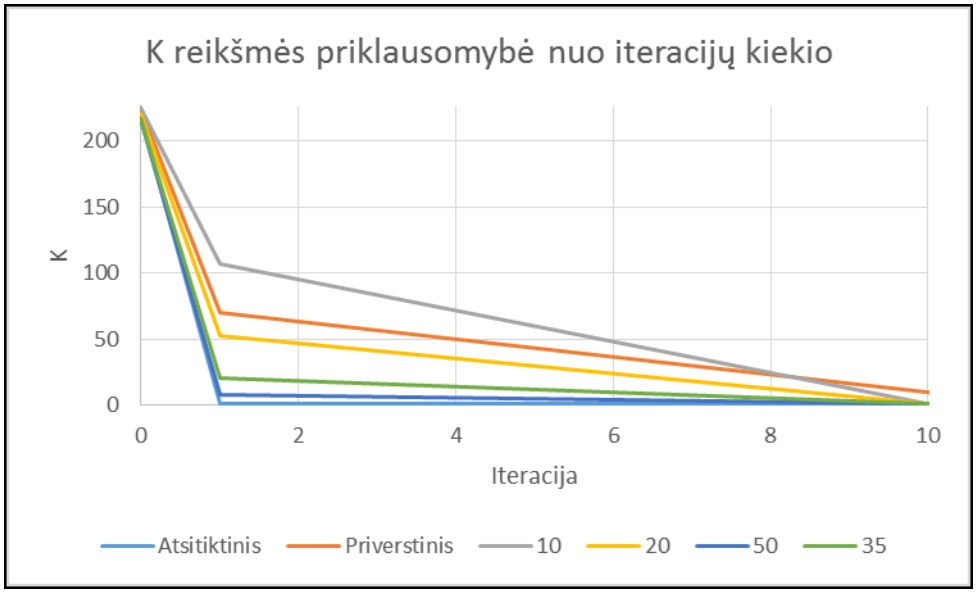


Pav. 12.Δx standartinio nuokrypio priklausomybės iki 100 iteracijos grafikas



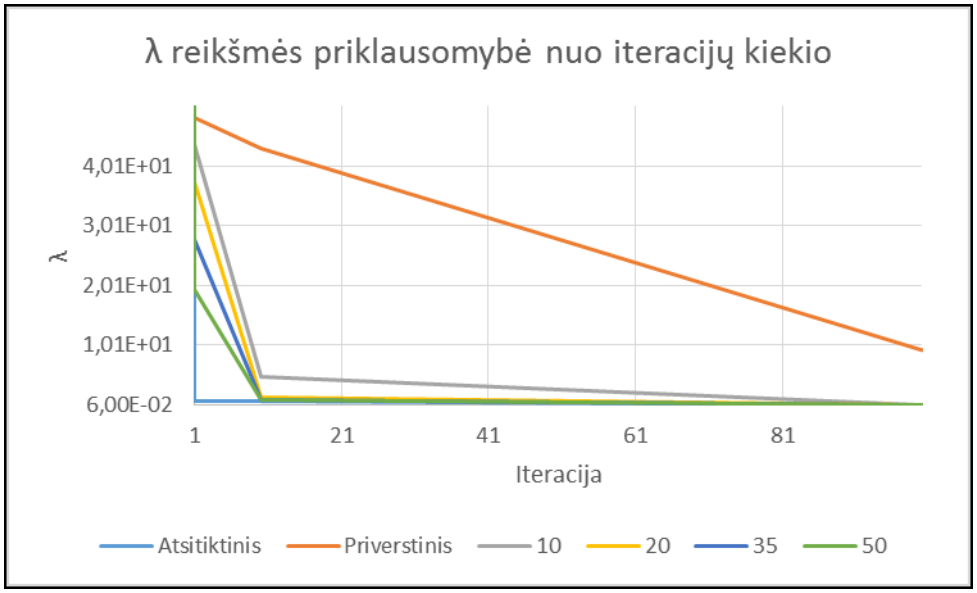
Pav. 13. Δx standartinio nuokrypio priklausomybės nuo 1000 iki 20000 iteracijos grafikas

Pav. 12 ir pav. 13 yra pateikiamas *Δx* standartinio nuokrypio priklausomybės nuo iteracijų kiekio grafikas. Pav. 12 variacijos: „atsitiktinis“, „35“, „50“ nuo 10 iki 100 iteracijos *Δx* standartinio nuokrypio reikšmė mažesnė už 1. *Δx* standartinio nuokrypio reikšmė nuo 1-10 iteracijos kinta nuo 5 iki 0,08. Visų variacijų standartinis nuokrypis pirmoje iteracijoje neviršija 5. „Priverstinės“ variacijos sklaida didžiausia, nuo 100 iteracijos ji lenkia kitus 131 kartą. Pav. 13 nuo 5000 iteracijos visų variacijų *Δx* standartinis nuokrypis mažesnis negu 0,0001. Pagal gautus duomenis, galime tvirtinti, kad visi mūsų 2000 kartų vykdyti bandymai duoda vienas kitam artimus rezultatus. Visų vykdytų bandymų rezultatai konverguoja, artėja link optimumo.



Pav. 14. Weibull‘o skirstinio parametro K reikšmės priklausomybės nuo iteracijų kiekio grafikas

Pav. 14 pavaizduotas parametro K reikšmės kitimas nuo pradinio sprendinio (0 iteracijos) iki 10 iteracijos. Pastebime, kad pirmosiose iteracijose K reikšmės krenta nuo ~225 iki artimo 1. Taip pat pav. 14 pastebime, kad visos modeliuojamo atkaitinimo variacijos po pirmosios iteracijos sugeba sumažinti K reikšmę per 118,6744 vienetus, kas sudaro 52,7 % sumažėjimo. Dešimtoje iteracijoje parametro K reikšmė yra artima vienetui. Tai reiškia, kad funkcija virsta eksponentine funkcija.



Pav. 15. Weibull'o skirstinio parametro λ reikšmės priklausomybės nuo iteracijų kiekio grafikas

Pav. 15 yra pateikiamas Weibull'o skirstinio λ parametro reikšmės priklausomybės nuo iteracijų kiekio grafikas. Pastebime, kad didžiausias reikšmės kitimas įvyksta pirmosiose algoritmo veikimo iteracijose. Pav. 10 pateikiamas λ reikšmės pirmuose 100 iteracijose grafikas. Nuo pradinės reikšmės $1*10^{10}$ sumažėja iki ~50 vienetų po pirmosios iteracijos. „Atsitiktinio" metodo λ reikšmė po pirmosios iteracijos yra 0,69. Λ rodiklis nurodo funkcijos mastelį. Šiuo atveju parametras rodo, kad skirtiniai yra glodūs.

Funkcijos minimumo vidurkio reikšmes galima vaizduoti ir grafiniu metodu. Grafinis atvaizdavimas pateikiamas pav. 16 - 20. Kartu su funkcijos tikslo reikšmėmis yra pateikiamos ir Weibull'o skirstinio (angl. Cumulative distribution function) reikšmės. Jos yra skaičiuojamos pagal formulę:

$$F(x;k,\lambda) = \begin{cases} 1 - e^{-(x/\lambda)^k}, & x \geq 0, \\ 0, & x < 0; \end{cases}$$

Dažnis grafikuose vaizduojamas *Y* koordinačių ašyje ir yra skaičiuojamas pagal:

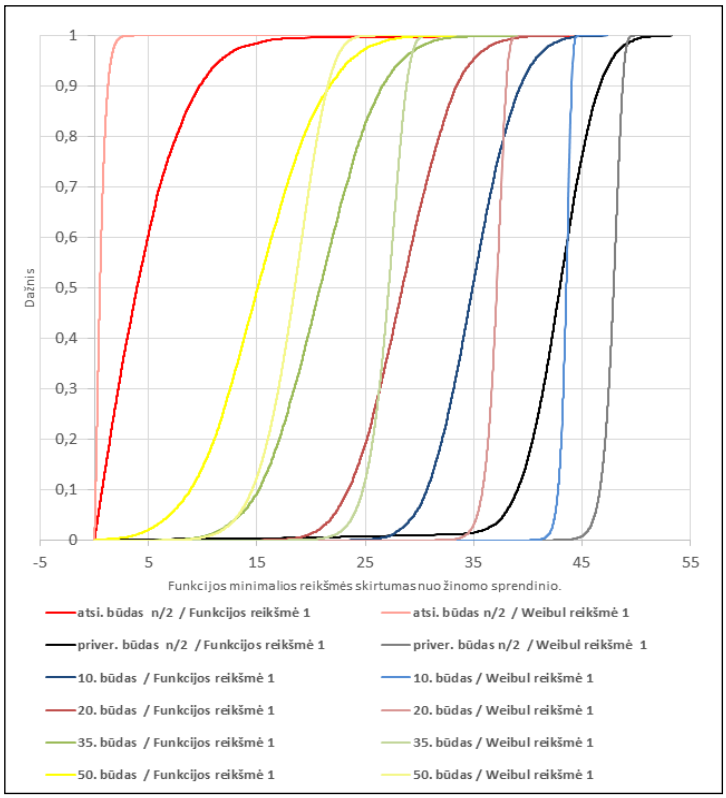
- *Surikiuojamos skirstinio reikšmės nuo mažiausios iki didžiausios.*
- *Kiekvienam skirstinio nariui priskiriama reikšmė pagal formulę:*

$$y = \frac{n}{i}, ;$$

čia i reprezentuoja funkcijos reikšmės vietą rikiuotoje aibėje, o *n* yra visas aibės narių skaičius.

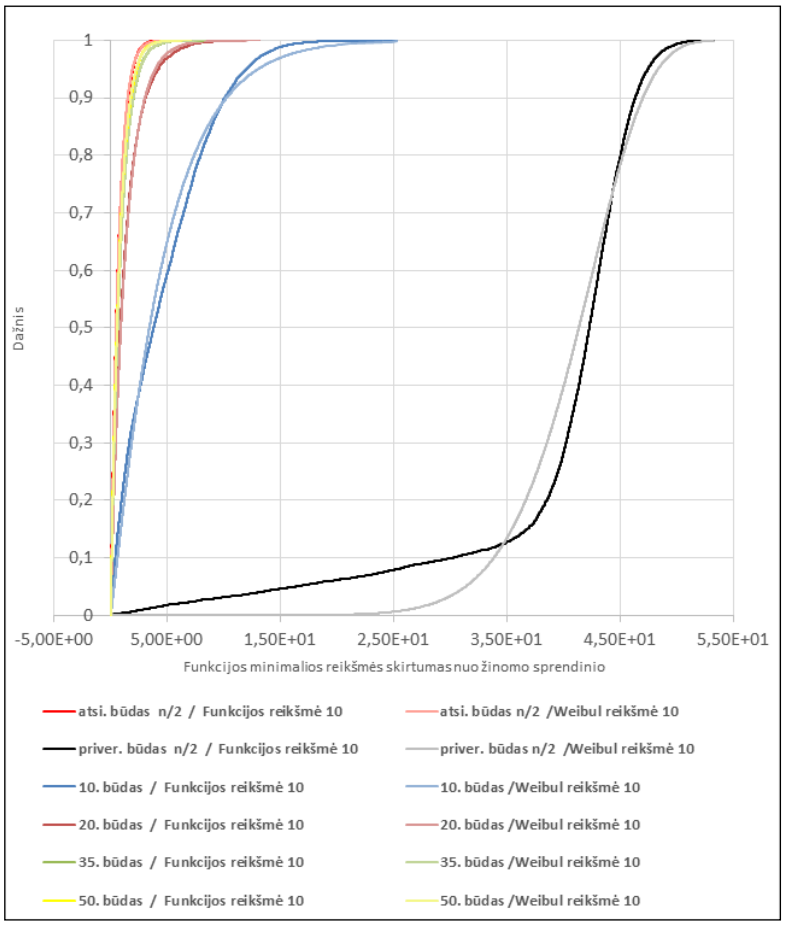
Pav. 16 yra pateiktas funkcijos Δx reikšmės po 1 iteracijos su teoriniu Weibull'o skirstiniu grafikas. Pastebime, kad geriausius rezultatus teikia modeliuojamo atkaitinimo algoritmas, kuris naudoja gylio parametrą, apskaičiuojamą pagal Pareto stabilųjį dėsnį. Teoriniai Weibull'o skirstiniai grafiškai

mažai skiriasi nuo funkcijos minimo reikšmių. Daugelyje skaičiavimų vyrauja funkcijos ilgos tankio uodegos.



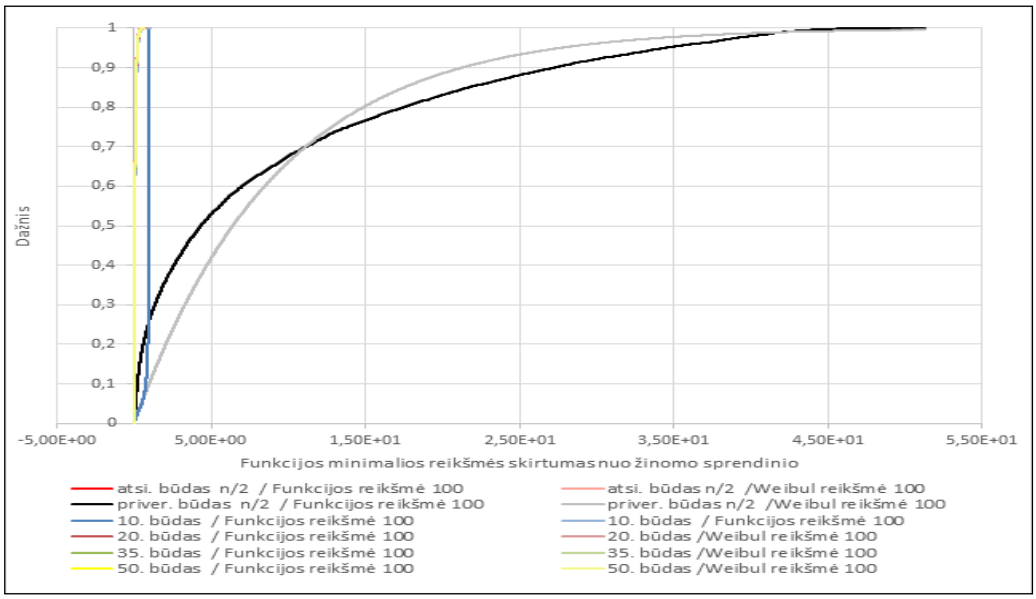
Pav. 16. Empirinis Δx reikšmės po 1 iteracijos kartu su Weibulo skirstiniais grafikas.

Pav. 17 yra pateiktas empirinis funkcijos minimumo reikšmės po 10 iteracijų kartu su Weibulo skirstiniais grafikas. Visi metodai, išskyrus „priverstinis“ būdas, panaikino ilgas tankio uodegas. Šiame etape taip pat galime pastebėti, kad Weibull‘o skirstiniai yra labai artimi funkcijos reikšmės grafikams. Taip pat po dešimt iteracijų vis dar galima grafiškai matyti, kad sukurtas „atsitiktinis metodas“ lenkia kitus metodus.



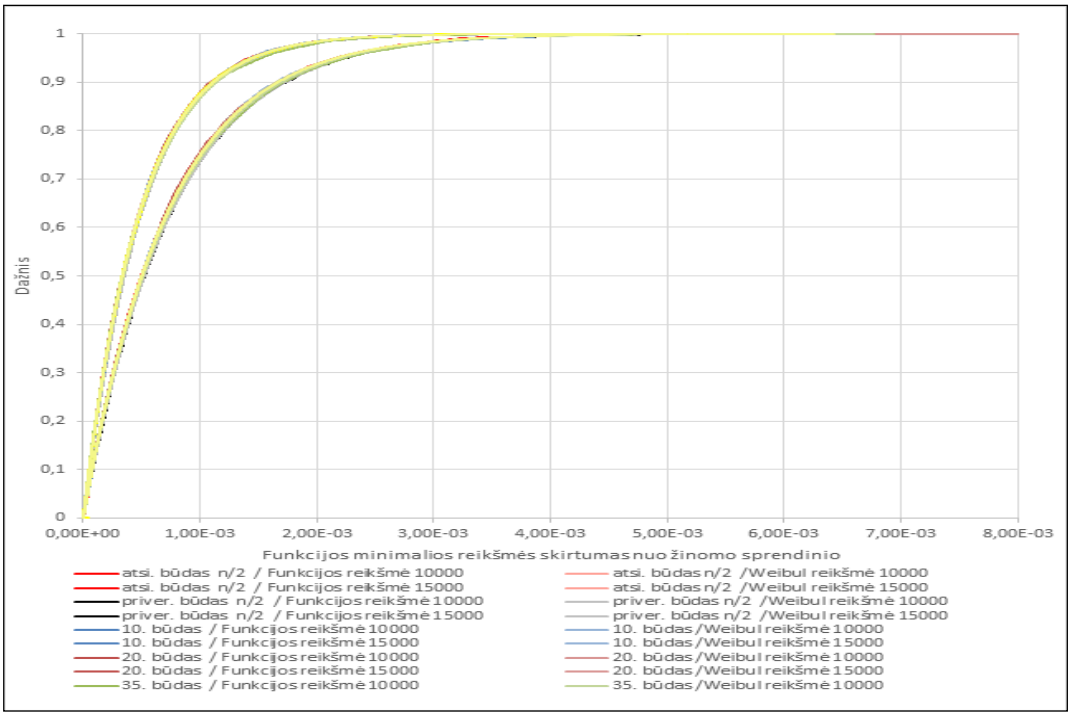
Pav. 17. Empirinis Δx reikšmės po 10 iteracijų kartu su Weibulo skirstiniais grafikas

Pav. 18 vis dar galime pastebėti, kad priverstinio būdo algoritmas atsilieka nuo kitų metodų. Likusių metodų reikšmių, kurios teikia iteracijoje geriausius rezultatus, grafike sunku atskirti, geriausią algoritmo variaciją. Funkcijos reikšmių ir Weibull‘o skirstinio grafikai sutampa vienas su kitu.



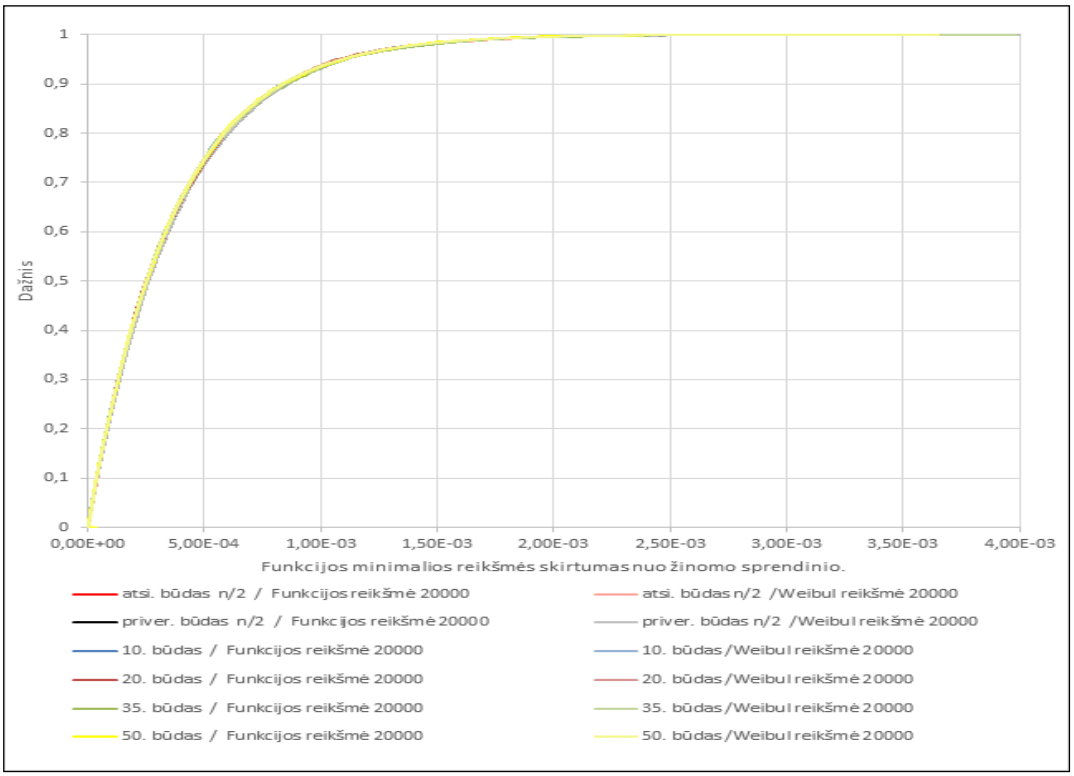
Pav. 18. Funkcijos Δx reikšmės po 100 iteracijų kartu su Weibulo skirstiniais grafikas

Pav. 19 pavaizduota situacija yra po 10000 ir 15000 iteracijų. Šiais atvejais priverstinio būdo algoritmas sutampa su likusių metodų grafikais. Taip pat matome, kad iš grafiko sunku pasakyti, kuris metodas teikia geresnius rezultatus.



Pav. 19. Funkcijos Δx reikšmės tarp 10000 ir 15000 iteracijų kartu su Weibulo skirstiniais grafikas

Pav. 20 yra pateikiamas grafikas po 20 000 iteracijų. Visų metodų funkcijos reikšmių grafikai yra arti vienas kito. Weibull‘o skirstinių grafikai sutampa su funkcijos grafikais. Tikslo funkcijos skirtumo tarp metodų iš grafiko išskirti negalime. Visu metodų reikšmės yra arti vienos kitos.



Pav. 20. Funkcijos Δx reikšmės po 20000 iteracijų kartu su Weibulo skirstiniais grafikas

2.4 PSPLIB uždavinys

Testuojant sukurtus tvarkaraščių algoritmus bei tiriant jų efektyvumą, yra pasinaudojama standartinių tęstinių pavyzdžių duomenų bazė ir apie sukurtų algoritmų privalumus yra sprendžiama, lyginant didelio skaičiaus tęstinių uždavinių sprendimo rezultatus.

Bibliotekoje PSPLib galima rasti skirtingų uždavinių rinkinių įvairaus tipo tvarkaraščių su ribotais ištekliais sudarymo uždaviniams spręsti. Kartu su uždavinio formalizuotu aprašymu yra pateikiami uždavinio sprendiniai, gauti įvairių autorių įvairiais algoritmais. Duomenų rinkiniai gali būti naudojami sprendimo procedūroms tikrinti. [31].

Spręsta 600 uždavinių iš PSPLib bibliotekos, kur kiekviename uždavinyje yra po 120 procesų, kurie gali naudoti 4 resursus. Kiekvieno proceso įvykdymo laikas ir resursų panaudojimas yra skirtingas. Pirmosios dvi modeliuojamo atkaitinimo algoritmo versijos, kai elementariųjų pertvarkymo kiekis yra konstanta (1 ir 10). Tai yra, kad iteracijoje leidžiame pakeisti tik nurodytą elementų skaičių. Trečioji algoritmo versija yra naudojanti Pareto stabilųjį dėsnį, norint įvertinti gylio parametrą.

Lentelėje Nr. 3 yra pateikiami tyrimo rezultatai. Pastebime, kad modeliuojamo atkaitinimo algoritmo variacijos sugeba gauti žinomą optimalią reikšmę daugiau negu 43 % visų uždavinių, t. y išsprendžiama virš 260 uždavinių iš 600. Optimali reikšmė išspręstuose uždaviniuose buvo randama apie 5000 iteraciją. Tai vidutiniškai užtrukdavo apie 42 sekundes. Variacija, kuri naudoja Pareto stabilųjį dėsnį elementariesiems pertvarkiams, teikia geriausius rezultatus. Ši variacija sugebėjo pasiekti 46,3 % uždaviniuose optimalią reikšmę, o ši reikšmė vidutiniškai pasiekiama po 4390 iteracijos.

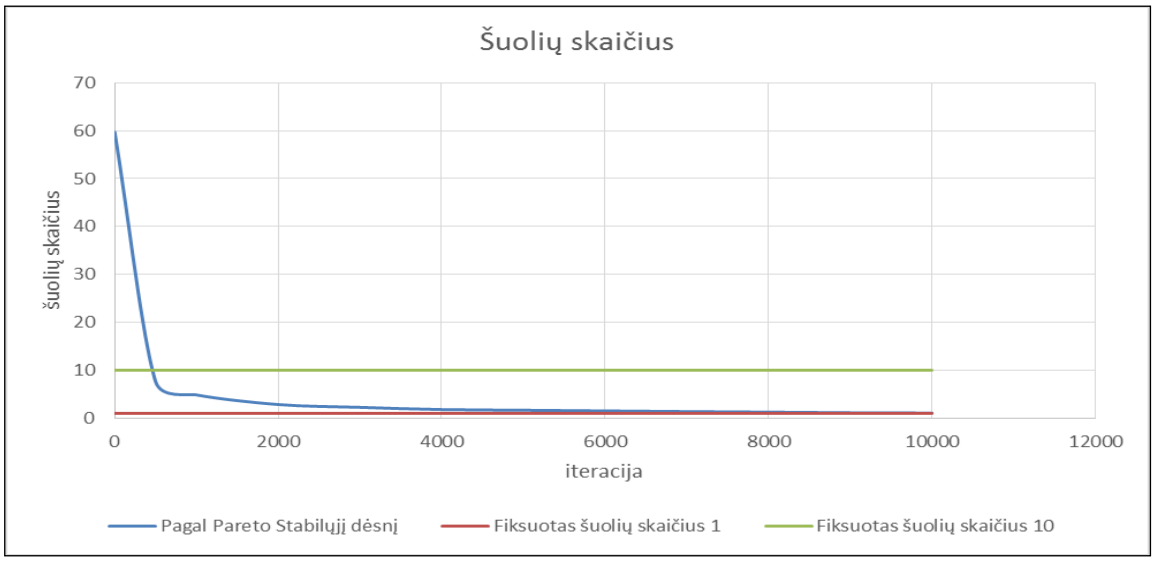
Lentelė Nr. 3. PSPLib rezultatų lentelė

	1	10	Atsitiktinis
Pavyko išspręsti pagal žinomą optimumą (vnt.)	260	268	278
Išspręstų užduočių procentas	43,3 %	44,67 %	46,33 %
Vidutiniškai 10000 iteracijų atliekama per (s)	89,32	81,56	95,25
Vidutiniškai optimumas buvo pasiekiamas (iteracija)	5222	5184,36	4390,35

Algoritmas, naudojant anksčiau nurodytus parametrus, per 4000 iteracijų pereina nuo 60 procesų iki 2 procesų per iteraciją. Detalesnė informacija pateikiama lentelėje Nr. 4. Taip pat visų trijų variacijų šuolių (elementariųjų pertvarkymų) diagrama pavaizduota pav. 13.

Lentelė Nr. 4. Šuolių lentelė

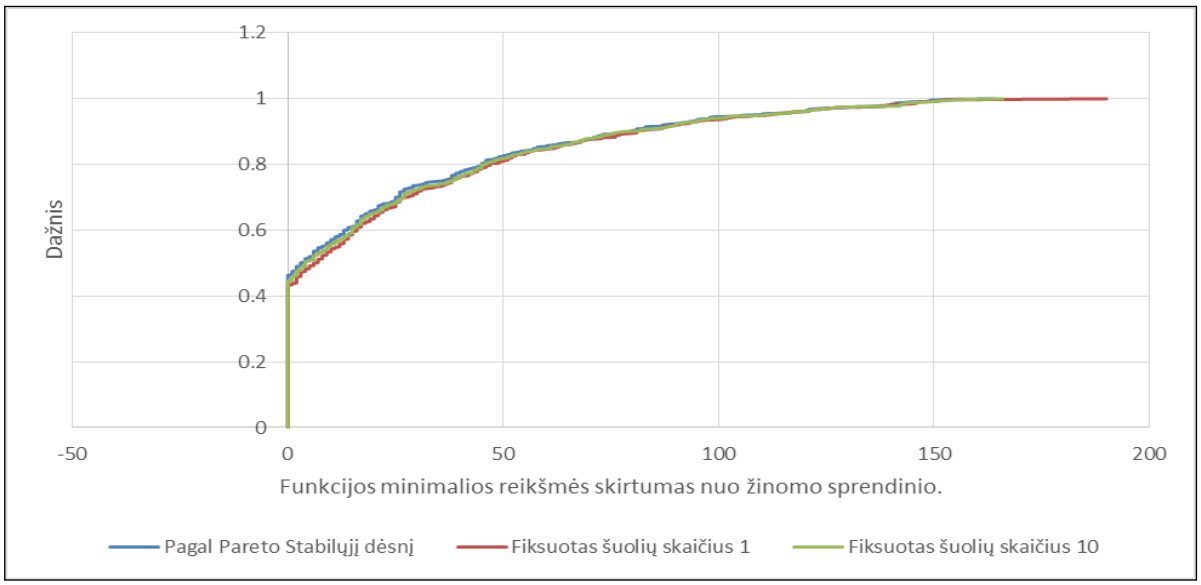
1	500	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
59.73	7.66	4.85	2.84	2.25	1.79	1.66	1.5	1.37	1.26	1.13	1.06



Pav. 21. Šuolių grafikas

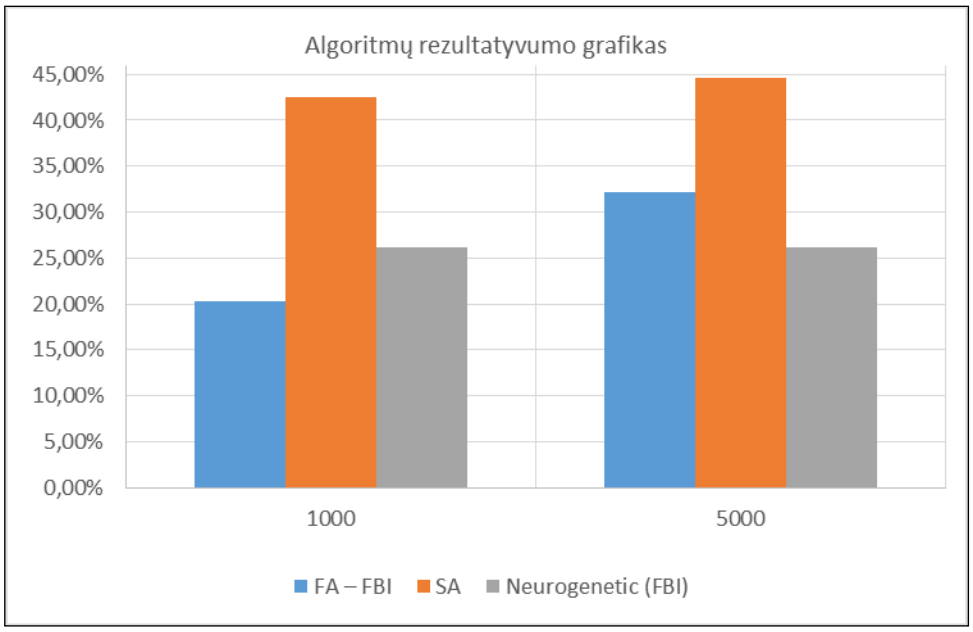
Pav. 21 yra pateikiamas elementariųjų pertvarkymų priklausomybės nuo iteracijų kiekio grafikas. Pastebime, kad per 10000 iteracijų modeliuojamo atkaitinimo variacija, naudojanti Pareto stabilųjį dėsnį, sumažina elementariųjų pertvarkymų kiekį nuo 60 iki 1.

Pav. 22 yra pavaizduota trijų variantų empirinio skirstinio diagrama. Pastebime, kad nuo 1 iki 50 funkcijos minimo reikšmės atkaitinimo variacija, naudojanti Pareto stabilųjį dėsnį, teikia kokybiškesnius rezultatus, nes grafikas yra arčiau Y koordinačių ašies. Taip pat dažniau bandymuose pasikartoja, kad ši variacija pasieka optimumą.



Pav. 22. 10000 iteracijos empirinio skirstinio diagrama

2.4.1 **PSPLIB palyginimas su kitais autoriais**



Pav. 23. Algoritmų rezultatyvumo grafikas [33],[34]

Pav. 23 yra pateiktas trijų autorių skirtingų metaeuristinių algoritmų palyginimas. Grafike lyginame, kiek uždavinių iš 600 vnt. sprėstų pavyko rasti optimumą išreikštą procentais. Sukurtam algoritmui (SA) pavyko rasti optimumą 42,5 % po 1000 iteracijų ir 44,56 % po 5000 iteracijų iš 600 spręstų uždavinių. Kitų autorių rezultatai neviršija 33 %. Sukurtas algoritmas sugeba net 16 % daugiau uždavinių rasti optimumą per 1000 iteracijų.

Lentelėje Nr. 5 pateiktos rezultatų vidutinio nuokrypio reikšmės, kurios apskaičiuojamos pagal vidutinio nuokrypio formulę. Vidutinis nuokrypis apskaičiuojamas pagal formulę:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - X_{vid})^2}$$

Pateikiami 31 autoriaus sukurti algoritmai. Sukurto modeliuojamo atkaitinimo algoritmo rezultatas po 1000 iteracijų patenka į geriausiųjų 15, atsilikdamas nuo geriausio rezultato 12 %. Po 5000 iteracijoje algoritmo rezultatai patenka į 12 vietą ir atsilieka nuo geriausio rezultato 13,5 %.

Lentelė Nr. 5. PSPLib uždavinio autorių palyginimo vidutinio nuokrypio su 120 užduotimis atvejo analizė [32]

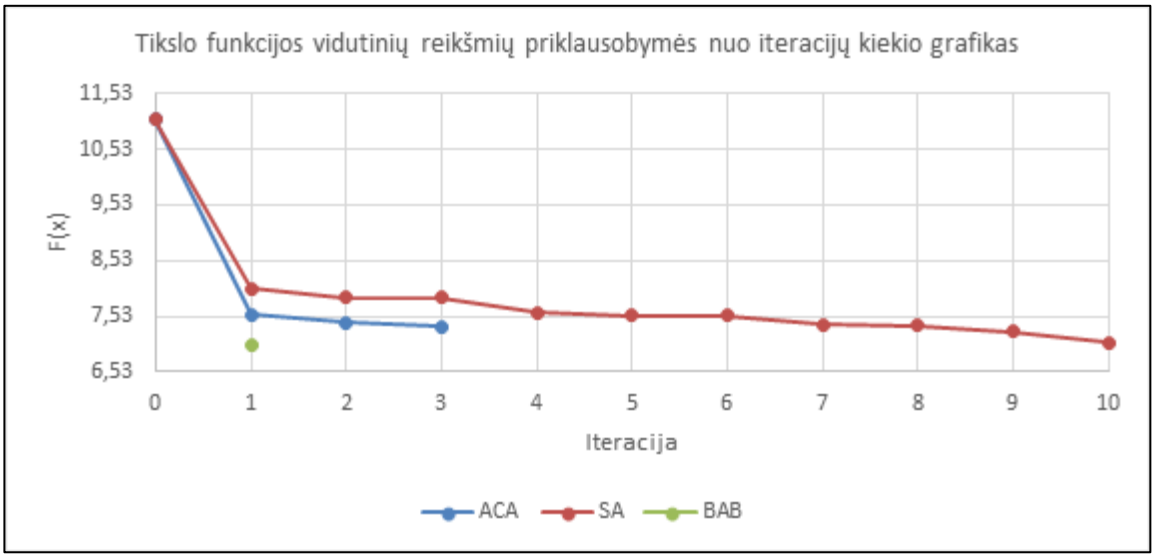
Autorius	Metodas	Iteracijų skaičius

		1000	5000
Valls, et al.	GA – hybrid, FBI	34,07	32,54
Alcaraz, et al.	GA – forw.–backw., FBI	36,53	33,91
Debels, et al.	Scatter search – FBI	35,22	33,1
Valls, et al.	GA–FBI	35,39	33,24
Kochetov and Stolyar	GA,TS–path relinking	34,36	33,36
Valls, et al.	Population –based –FBI	35,18	34,02
Chen, Ruey-Maw	JPSO	35,71	33,88
Hartmann	GA –self -adapting	37,19	35,39
Tormos and Lova	Sampling –LFT, FBI	35,01	34,41
Merkle, et al.	Ant system	-	35,43
Hartmann	GA –activity list	39,37	36,74
Tormos and Lova	Sampling –LFT, FBI	36,24	35,56
Tormos and Lova	Sampling –LFT, FBI	36,49	35,81
Alcaraz and Maroto	GA–forw.–backw.	39,36	36,57
Nonobe and Ibaraki	TS –activity list	40,86	37,88
	GRASP-JUST-SS	38,16	37,3
Coelho and Tavares	GA–late join	39,97	38,41
Valls, et al.	Sampling –random, FBI	38,21	37,47
Bouleimen and Lecocq	SA–activity list	42,81	37,68
Hartmann	GA–priority rule	39,93	38,49
Schirmer	Sampling –adaptive	39,85	38,7
Kolisch	Sampling –LFT	39,6	38,75
Kolisch	Sampling –WCS	39,65	38,77
Hartmann	GA –random key	45,82	42,25
Kolisch and Drexl	Sampling –adaptive	41.37	40.45
Coelho and Tavares	Sampling –global	41,36	40,46
Leon and Ramamoorthy	GA –problem space	42,91	40,69
Kolisch	Sampling –LFT	42,84	41,84
Kolisch	Sampling –random	44,46	43,05
Kolisch	Sampling –random	49,25	47,61

L. Sakalauskas, and D. Kavaliauskas (sukurtas algoritmas)	SA	37,922	36,94

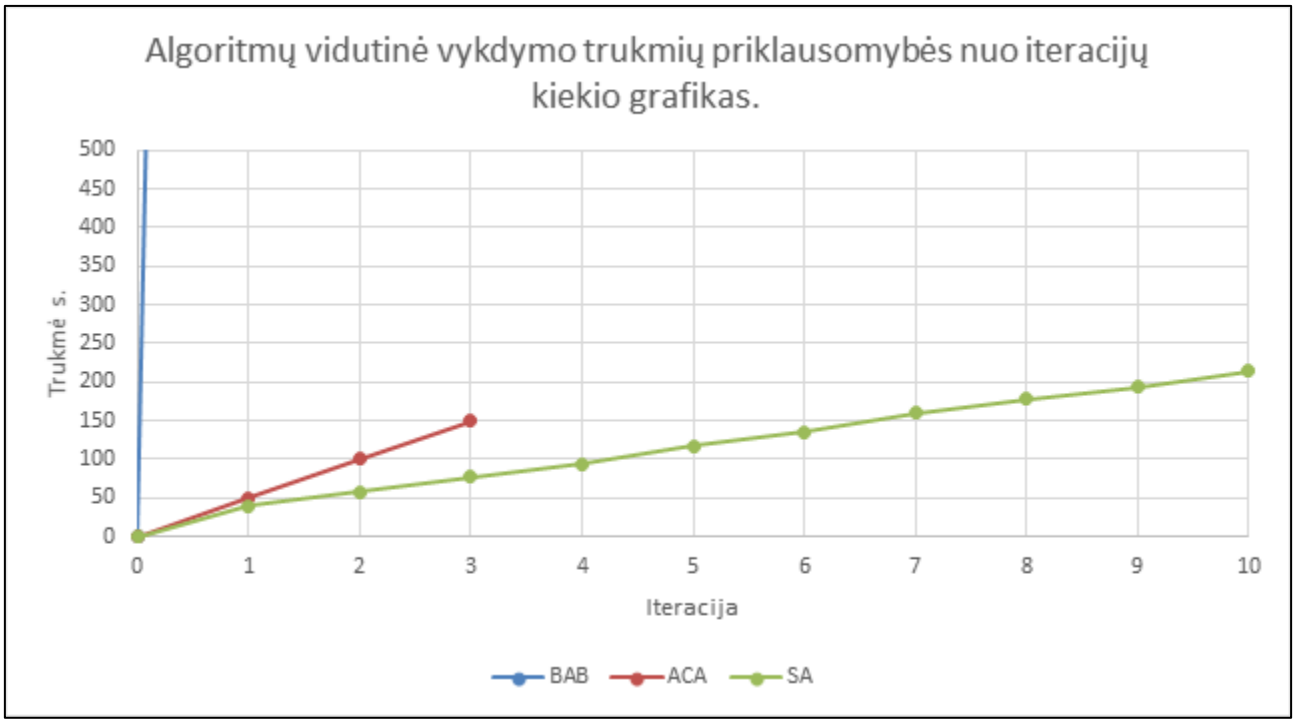
2.5 *Uždavinys su sąryšiais*

Šį uždavinį sprendžiame modeliuojamo atkaitinimo algoritmo algoritmu, kai naudojame Pareto stabilųjį dėsnį elementariųjų pertvarkymo kiekiui generuoti . BAB algoritmas yra tikslusis algoritmas. Šis algoritmas kiekvieno bandymo metu pateikią tą pati rezultatą, todėl jis buvo atliekamas tik vieną kartą. Modeliuojamo atkaitino ir skruzdžių kolonijos algoritmai buvo kartojami po 100 kartų.



Pav. 24 Tikslo funkcijos vidutinių reikšmių priklausomybės nuo iteracijų kiekio grafikas.

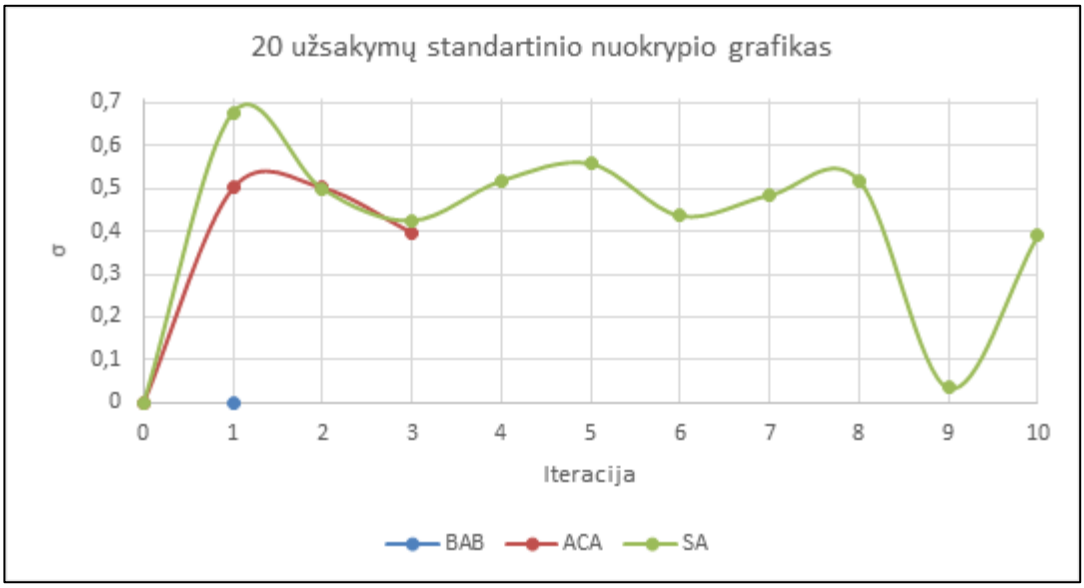
Pav. 24 yra pateiktas 20 vnt. užsakymų uždavinio, 100 bandymų registruotų tikslo funkcijos vidutinės reikšmės priklausomybė nuo iteracijų kiekio. Sprendinį visi algoritmai pagerina daugiau negu 36 %. Šakų ir ribų algoritmas teikia geriausius rezultatus tai yra 36,57 % trumpesnis planas. Visi algoritmai po pirmos iteracijos sugeba sutrumpinti planą daugiau negu 3 paromis.



Pav. 25 Algoritmų vidutinė vykdymo trukmių priklausomybės nuo iteracijų kiekio grafikas.

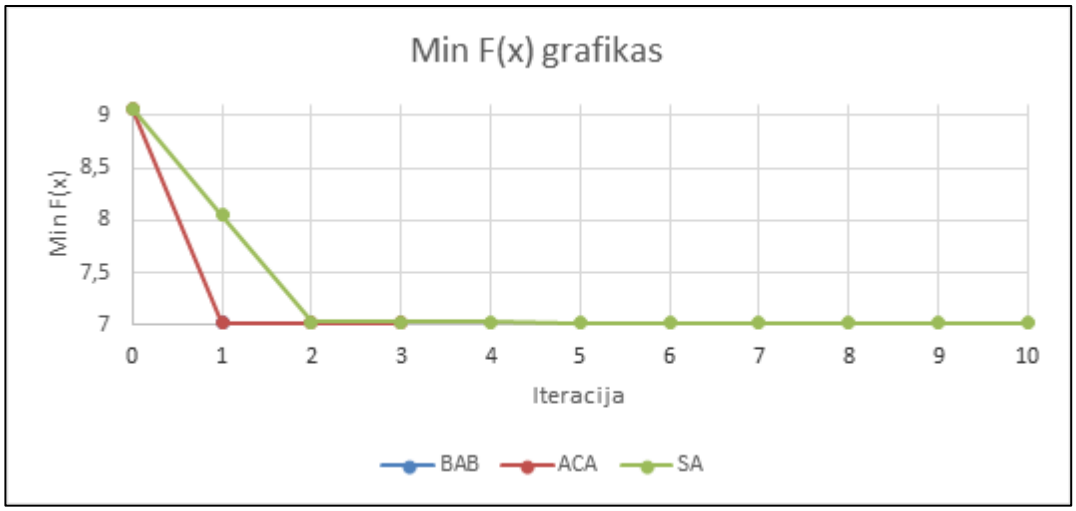
Pav. 25 yra pateikta vykdymų vidutinių bandymų trukmių priklausomybė nuo iteracijų kiekio grafikas. Šakų ir ribų algoritmas skaičiavimus atlieka per 107,8 min. Tuo tarpu skruzdžių kolonijos algoritmas atsakymą pateikia per mažiau nei 3 min., modeliuojamo atkaitinimo algoritmas neužtrunka nei 4 min.

Nors ACA algoritmas pateikė atsakymą greičiau nei SA algoritmas, tačiau atsakymas šiuo nagrinėjamu atveju buvo 5 proc. prastesnis.



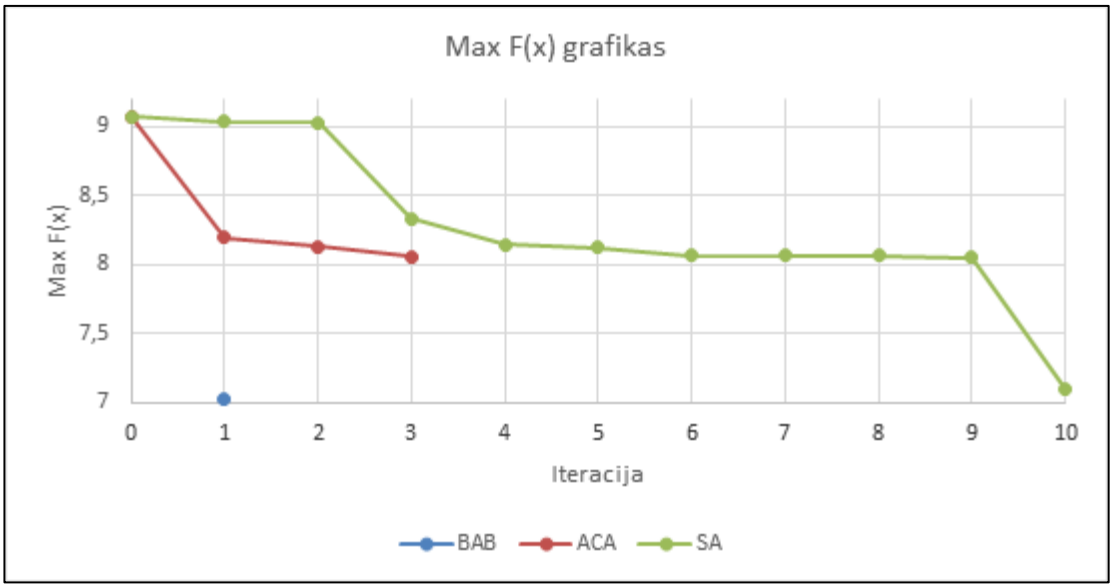
Pav. 26 20 užsakymų standartinio nuokrypio priklausomybės nuo iteracijų kiekio grafikas.

Pav. 26 yra pateiktas standartinio nuokrypio priklausomybė nuo iteracijų kiekio grafikas. Po pirmosios iteracijos ACA algoritmo sugeneruotų duomenų σ <0,51, o modeliuojamo atkaitinimo algoritmo σ <0,7. Šių algoritmų standartinis nuokrypis neviršija 9 % nuo vidutinės reikšmės. Nuo 2 iki 9 iteracijos modeliuojamo atkaitinimo standartinio nuokrypio amplitudė ~0,2. Pav. 14 rodo, kad norint sumažinti standartinio nuokrypio reikšmę, reikia didinti iteracijų kiekį algoritmuose.



Pav. 27 20 užsakymų geriausio pagerėjimo priklausomybės nuo iteracijų kiekio grafikas.

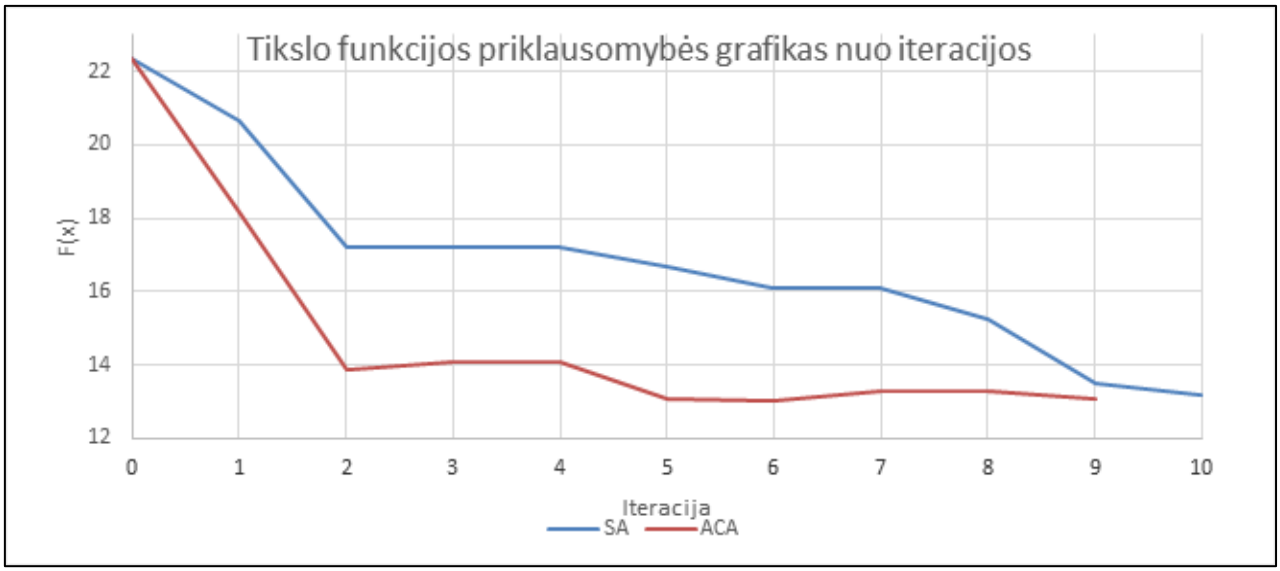
Pav. 27 yra pateikta iš 100 atliktų bandymų pasiekta geriausia tikslo funkcijos reikšmė fiksuotose iteracijose kitimo grafikas. Tai rodo, geriausią užfiksuotą tikslo funkcijos reikšmę registruotose vykdymo iteracijose. Pastebime, kad skruzdžių kolonijos algoritmas jau po pirmos iteracijos 22,6 % pateikė geresnį rezultatą. Tuo tarpu modeliuojamo algoritmas po pirmos iteracijos - 11.24 %. Po aštuntos iteracijos modeliuojamo atkaitinimo algoritmui pavyksta pasiekti skruzdžių kolonijos algoritmo rezultatą. Trečioje iteracijoje skruzdžių kolonijos algoritmo maksimalaus pagerėjimo reikšmė skiriasi nuo tikslo funkcijos vidutinės reikšmės 0,33. Tai yra 4,5% geresnis rezultatas negu tikslo funkcijos vidutinė reikšmė. Modeliuojamo atkaitinimo algoritmo maksimalaus pagerėjimo reikšmė 10 iteracijoje lenkia vidutinės reikšmės tikslo funkcijos reikšmę 0,23. Tai sudaro 3,17 % kokybiškesnį gamybos planą. Geriausio bandymo metu modeliuojamo atkaitinimo algoritmo reikšmė po antros iteracijos yra 0,14% prastesnė negu po 10 iteracijos, o skruzdžių kolonijos algoritmas sugebėjo gauti geriausią reikšmę po pirmos iteracijos.



Pav. 28 Minimalus tikslo funkcijos pagerėjimo priklausomybės nuo iteracijų grafikas.

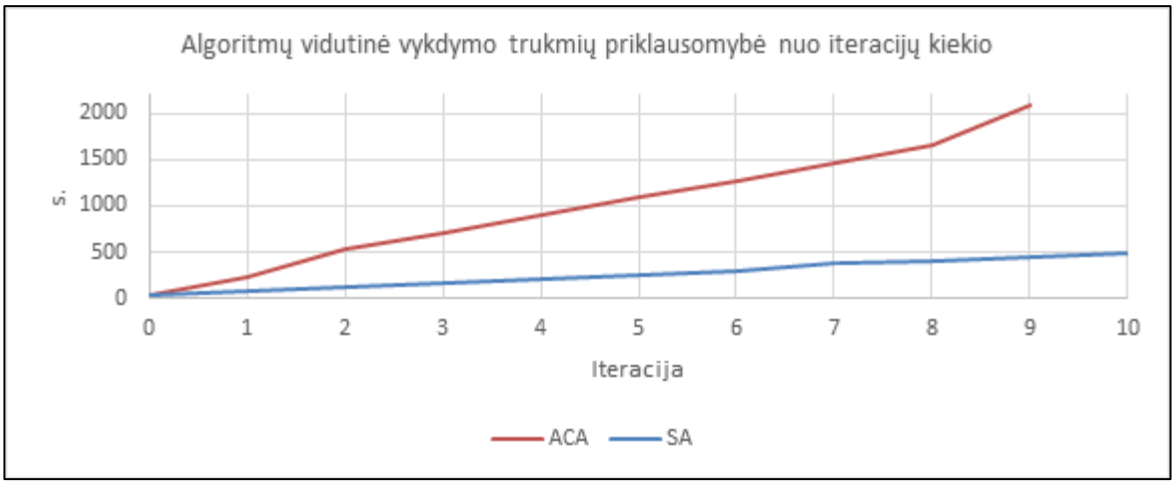
Pav. 28 yra pateikta iš 100 atliktų bandymų pasiekta blogiausia tikslo funkcijos reikšmės fiksuotose iteracijose kitimo grafikas. Tai rodo, prasčiausią užfiksuotą tikslo funkcijos reikšmę registruotose vykdymo iteracijose. Pastebime, kad modeliuojamo atkaitinimo algoritmas pageriną rezultatą 0,74 arba 8,12 %. Tuo tarpu skruzdžių kolonijos algoritmas pirmoje iteracijoje rezultatą pageriną 0,88, tai sudaro 9,68%. ACA algoritmas vienoje iteracijoje atlieka 3 nepriklausomus vienas nuo kito sprendimus. Skruzdžių kolonijos algoritmo rezultatas trečioje iteracijoje atsilieka nuo tikslo funkcijos vidurkio 0,7 arba 9,6 % blogesnis rezultatas. Modeliuojamo atkaitinimo algoritmo minimalaus pagerėjimo reikšmė 10 iteracijoje skiriasi nuo tikslo funkcijos vidurkio 0,48. Tai yra 0,68 % blogesnis rezultatas. Algoritmų visų bandymų rezultatai rodo, kad prasčiausiu skaičiavimo metu sprendinio rezultatas pagerinamas per 1, kas sudaro 11,2%.

Papildomai buvo tirta šio uždavinio modifikacija su 45 užsakymais. Skruzdžių kolonijos algoritmui iteracijoje leista atlikti 5 nepriklausomus vienas nuo kito sprendinius.



Pav. 29 SA ir ACA algoritmų tikslo funkcijos vidutinės reikšmės priklausomybės grafikas nuo iteracijų kiekio.

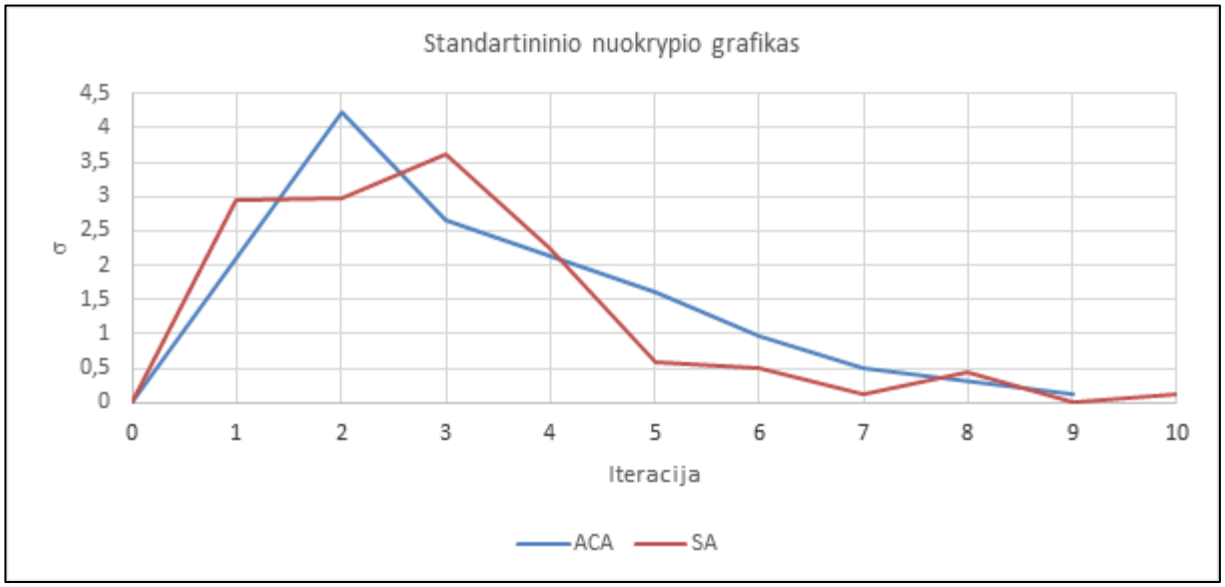
Šakų ir ribų algoritmas šiam uždaviniui išspręsti truko ilgiau negu 4 valandas. Todėl jis buvo nenagrinėjamas. Pav. 29 pateikiamas 45 uždavinių tvarkaraščio trukmės F(x) vidurkio iš 100 bandymo priklausomybės nuo iteracijų kiekio grafikas. Pastebime, kad skruzdžių kolonijos algoritmas per pirmąsias dvi iteracijas atlieką didesnį pokyti negu modeliuojamo atkaitinimo algoritmas ir abu algoritmai sugeba daugiau nei 41 % pateiktį geresni rezultatą per savo vykdymo procesą. ACA algoritmas atsakymą pateikė po 9 iteracijų ir rezultatas 0,44 %. geresnis lyginant su SA. Modeliuojamo atkaitinimo algoritmas atsakymą pateikė po 10 iteracijų.



Pav. 30 Algoritmų vidutinė vykdymo trukmių priklausomybės nuo iteracijų kiekio grafikas.

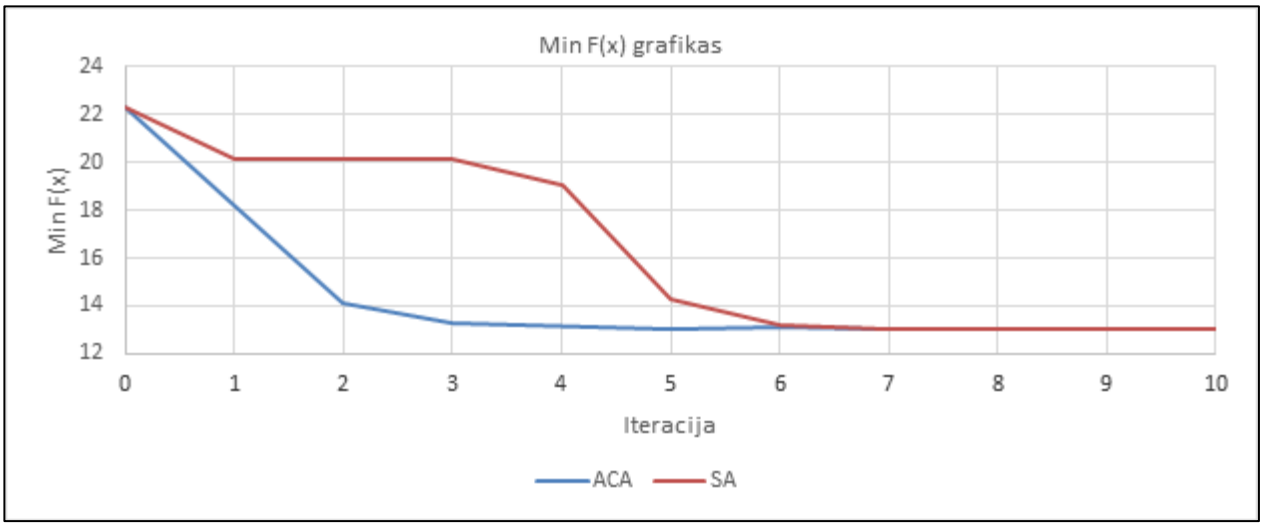
Pav. 30 yra pateikiamas 45 užsakymų algoritmų vykdymo vidutinis bandymų trukmių priklausomybė nuo iteracijų kiekio. Abiejų algoritmų vykdymo laikas auga tiesine progresija. Skruzdžių kolonijos

algoritmas skaičiavimus vykdė 34,8 minutės, o modeliuojamo atkaitinimo algoritmas 8,2 minutės. Nors skruzdžių kolonijos algoritmas 0,28 % pasiekė geresnį rezultatą negu modeliuojamo atkaitino algoritmas, tačiau tam sugaišo 4,2 kartus ilgiau.



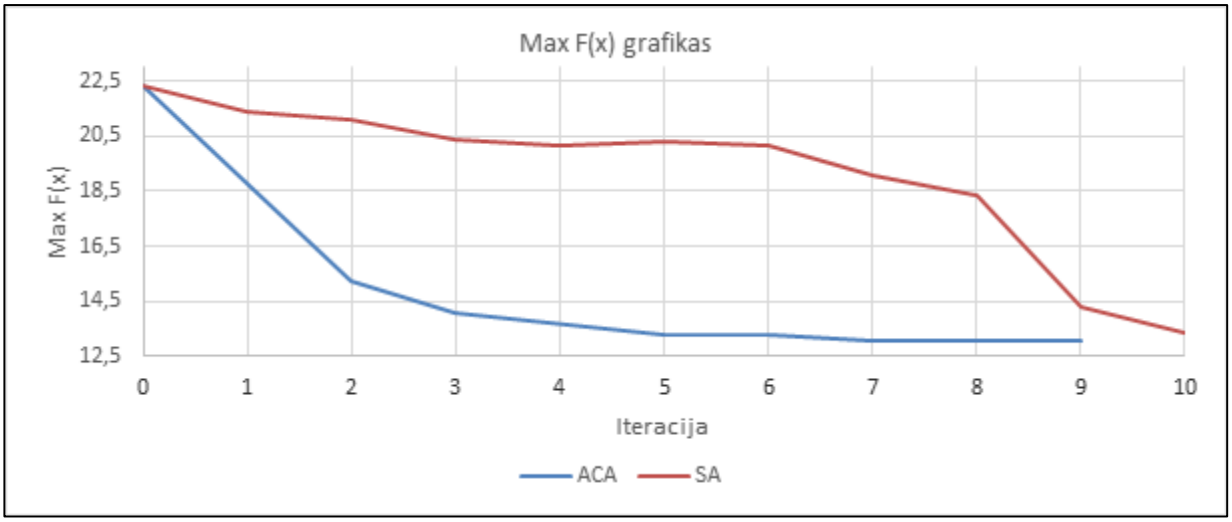
Pav. 31 45 užsakymų tikslo funkcijos standartinio nuokrypio priklausomybės nuo iteracijų kiekio grafikas.

Pav. 31 yra pateiktas standartinio nuokrypio priklausomybė nuo iteracijų kiekio grafikas. Abiejų algoritmų sklaida neviršija 4,3 vienetų. Didžiausią sklaidos koeficientą pasiekia skruzdžių kolonijos algoritmas po antros iteracijos - 4,22. Modeliuojamo atkaitinimo algoritmo standartinis nuokrypis po penktos iteracijos tampa mažesnis negu 1, o skruzdžių kolonijos algoritmo tik po 6 iteracijos. Atliekant didesnį kiekį iteracijų abiejų algoritmo standartinis nuokrypis mažėja tiesine regresija.



Pav. 32 didžiausia tikslo funkcijos pagerėjimo priklausomybės nuo iteracijų kiekio grafikas.

Pav. 32 yra pateikta iš 100 atliktų bandymų pasiekta geriausią tikslo funkcijos reikšmė, fiksuotose iteracijose, kitimo grafikas. Tai rodo, geriausią užfiksuotą tikslo funkcijos reikšmę registruotose vykdymo iteracijose. Pastebime, kad skruzdžių kolonijos algoritmas po pirmųjų 2 iteracijų pagerina tikslo funkcijos reikšmę per 9,02 arba 40,4%. Tuo tarpu modeliuojamo atkaitino algoritmas sugeba sumažinti tikslo funkcijos reikšmę tik per 8,08, tai yra 36,21%. Skruzdžių kolonijos algoritmas po 2 iteracijų skiriasi nuo tikslo funkcijos vidurkio reikšmės 0,57. Tai yra maksimalus pagerėjimo planas 4,08 % geresnis. Tuo tarpu modeliuojamo atkaitinimo algoritmo maksimali pagerėjimo reikšmė skiriasi nuo tikslo funkcijos vidurkio po 2 iteracijų 2,95. Tai yra 2,52 % geresnis rezultatas nuo tikslo funkcijos vidurkio. Rezultatai atskleidžia, kad gali egzistuoti metaeuristinių algoritmų parametrų rinkinys, kuris teiks greičiau rezultatus.



Pav. 33 Minimalus tikslo funkcijos pagerėjimo priklausomybės nuo iteracijų kiekio grafikas.

Pav. 33 pateikta iš 100 atliktų bandymų pasiekta blogiausią tikslo funkcijos reikšmė fiksuotose iteracijose kitimo grafikas. Tai rodo, prasčiausią užfiksuotą tikslo funkcijos reikšmę registruotose vykdymo iteracijose. Matoma, kad skruzdžių kolonijos algoritmas po 2 iteracijos atlieka didžiausią pokyti. Jis pagerina planą 8,24 arba 36,89 %. Modeliuojamo atkaitinimo algoritmo minimalus tikslo funkcijos pagerėjimo reikšmė po 2 iteracijų yra geresnė 1,96 arba 8,8 %. Po devintos iteracijos skruzdžių kolonijos algoritmas minimalaus tikslo funkcijos pagerėjimo grafikas skiriasi nuo tikslo funkcijos vidurkio 0,007 prastesniu rezultatu. Tai yra 0,05 % prastesnis rezultatas. Modeliuojamo atkaitinimo algoritmo minimalaus tikslo funkcijos reikšmė po dešimtos iteracijoje mažesnė 0,13 nuo tikslo funkcijos vidurkio. Tai yra 0,98 % prastesnis rezultatas. Tyrimas atskleidžia, kad algoritmų minimalaus pagerėjimo charakteristika nepriklauso nuo užsakymų kiekio. Algoritmas abiejuose uždaviniuose elgėsi identiškai.

Kadangi pastebėta, kad šakų ir ribų algoritmas skaičiavimus vykdo ilgiau, nuspręsta ištirti algoritmo vykdymo trukmės priklausomybę nuo užsakymo kiekio. Pav. 34 yra pateikiamas šios priklausomybės grafikas. Grafike yra pateikiamos vidutinės reikšmės iš dešimties bandymų. Pagal tyrimo duomenis matome, kad didinant užsakymų skaičių nuo 25 vienetų algoritmo vykdymo laikas auga eksponentiškai. Šis algoritmas tampa netinkamas realiose sąlygose, jeigu uždavinys turi didesnį užsakymų kiekį negu 20 vienetų.

Pav. 34 Šakų ir ribų algoritmo vykdymo laiko priklausomybės grafikas nuo užsakymo kiekio grafikas.

3 Literatūra

1. *Leonidas Sakalauskas. (2002). Nonlinear Stochastic Programming by Monte-Carlo estimators. EJOR, V. 137, no. 3, p. 558–573.*
2. *Jonathan Pengelly, (2002). Monte Carlo Methods. [interaktyvus][žiūrėta 2017-04-19]. Prieiga internete: https://pdfs.semanticscholar.org/5a97/bc0b7c3de780ae1b71162c3e0e37a97e4757.pdf*
3. *V. H. Coria, S. Maximov, F. Rivas-Davalos, C. L. Melchor-Hernandez, (2016). Perturbative method for maximum likelihood estimation of the Weibull distribution parameters.*

[interaktyvus][žiūrėta 2017-04-19]. Prieiga internete: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5069269/pdf/40064_2016_Article_3500.pdf

4. *V. H. Coria, S. Maximov, F. Rivas-Davalos, C. L. Melchor-Hernandez, (2016). Perturbative method for maximum likelihood estimation of the Weibull distribution parameters. [interaktyvus][žiūrėta 2017-04-19]. Prieiga internete: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5069269/pdf/40064_2016_Article_3500.pdf*
5. *George Karabatsos, (2017). Marginal maximum likelihood estimation methods for the tuning parameters of ridge, power ridge, and generalized ridge regression. Communications in Statistics - Simulation and Computation, 47(6), p. 1632–1651. Prieiga internete: https://doi:10.1080/03610918.2017.1321119*
6. *Gražvydas Felinskas,(2007). Euristinių metodų tyrimas ir taikymas ribotų išteklių tvarkaraščiams optimizuoti. Daktaro disertacija. P. 58-72. [interaktyvus] [žiūrėta 2017-03-10]. Prieiga internete: http://www.mii.lt/files/mii_dis_07_felinskas.pdf*
7. *Kirkpatrick, S.; Gelatt C.D. Jr.; Vecchi, M. P. (1983). Optimization by simulated annealing, Science. Vol. 220, p. 671-680.*
8. *Altaleb, A., Chauveau, D. (2002). Bayesian analysis of the Logit model and comparison of two. Metropolis-Hastings strategies. Computational Statistics and Data Analysis, 39: 137–152*
9. *Ilker Yildirim , (2012). Bayesian Inference: Metropolis-Hastings Sampling. [interaktyvus] [žiūrėta 2017-03-10]. Prieiga per interneta: http://www.mit.edu/~ilkery/papers/MetropolisHastingsSampling.pdf*
10. *Rainer Kolisch, Sönke Hartmann, (1999). Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis. Published Springer Science + Business Media. New York, USA.*
11. *Enda Ridge, (2007). Design of Experiments for Tuning of Optimisation Algorithms. PhD Thesis. [interaktyvus][žiūrėta 2017-04-10]. Prieiga internete: https://pdfs.semanticscholar.org/fab5/db46541cf40cb89d6feee58a8cd3c7db4484.pdf*
12. A. H. Land and A. G. Doig (1960). "An automatic method of solving discrete programming problems". *Econometrica*. **28** (3). pp. 497–520.
13. *Michael Pinedo, (2001). „Scheduling. Theory, algorithms, and systems“. [interaktyvus][žiūrėta 2017-04-10]. Prieiga internete: http://web-static.stern.nyu.edu/om/faculty/pinedo/scheduling/shakhlevich/handout06.pdf*

14. *P. Sivakumar and K. Elakia,(2016). A Survey of Ant Colony Optimization. International Journal of Advanced Research in Computer Science and Software Engineering, Volume 6, Issue 3, March 2016 [interaktyvus][žiūrėta 2017-04-19]. Prieiga internete: http://ijarcsse.com/Before_August_2017/docs/papers/Volume_6/3_March2016/V6I3-0126.pdf*
15. *P.-P. Grassé, (1959). La reconstruction du nid et les coordinations inter-individuelles chez Belicositermes natalensis et Cubitermes sp. La théorie de la Stigmergie : Essai d'interprétation du comportement des termites constructeurs. Insectes Sociaux, numéro 6, p. 41-80.*
16. *J.L. Denebourg, J.M. Pasteels et J.C. Verhaeghe,(1983). Probabilistic Behaviour in Ants : a Strategy of Errors?. Journal of Theoretical Biology, numéro 105.*
17. *S. Goss, S. Aron, J.-L. Deneubourg et J.-M. Pasteels,(1989). Self-organized shortcuts in the Argentine ant. Naturwissenschaften, volume 76, pages 579-581.*
18. *M. Ebling, M. Di Loreto, M. Presley, F. Wieland, et D. Jefferson, (1989). An Ant Foraging Model Implemented on the Time Warp Operating System. Proceedings of the SCS Multiconference on Distributed Simulation.*
19. *M. Dorigo, (1992). Optimization, Learning and Natural Algorithms. PhD thesis, Politecnico di Milano, Italy.*
20. *M. Dorigo, V. Maniezzo, et A. Colorni, (1996). Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics--Part B , volume 26, numéro 1, pages 29-41.*
21. *T. Stützle et H.H. Hoos, (2000). MAX MIN Ant System. Future Generation Computer Systems, volume 16, pages 889-914.*
22. *M. Dorigo et L.M. Gambardella, (1997). Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, volume 1, numéro 1, pages 53-66.*
23. *M. Dorigo, (1998). ANTS' 98. From Ant Colonies to Artificial Ants : First International Workshop on Ant Colony Optimization. ANTS 98, Bruxelles, Belgique.*
24. *T. Stützle,(1998). Parallelization Strategies for Ant Colony Optimization. Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature, Springer-Verlag, volume 1498, pages 722-731.*
25. *W.J. Gutjahr, (2000). A graph-based Ant System and its convergence. Future Generation Computer Systems, volume 16, pages 873-888.*

26. *S. Iredi, D. Merkle et M. Middendorf,(2001). Bi-Criterion Optimization with Multi Colony Ant Algorithms, Evolutionary Multi-Criterion Optimization. First International Conference (EMO'01), Zurich, Springer Verlag, pages 359-372.*
27. *L. Bianchi, L.M. Gambardella et M.Dorigo, (2002). An ant colony optimization approach to the probabilistic traveling salesman problem. PPSN-VII, Seventh International Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science, Springer Verlag, Berlin, Allemagne.*
28. *M. Zlochin, M. Birattari, N. Meuleau, et M. Dorigo, (2004). Model-based search for combinatorial optimization: A critical survey. Annals of Operations Research, vol. 131, pp. 373-395.*
29. *B. Prabhakar, K. N. Dektar, D. M. Gordon, (2012). The regulation of ant colony foraging activity without spatial information. PLOS Computational Biology. [interaktyvus][žiūrėta 2017-04-19]. Prieiga internete: http://www.ploscompbiol.org/article/info%3Adoi%2F10.1371%2Fjournal.pcbi.1002670*
30. *Mladineo, Marko; Veza, Ivica; Gjeldum, Nikola (2017). Solving partner selection problem in cyber-physical production networks using the HUMANT algorithm. International Journal of Production Research. 55(9): 2506–2521.*
31. *Kolisch R., Sprecher A., (1996). PSPLIB - A project scheduling library. European Journal of Operational Research, Vol. 96, p. 205-216*
32. *Qiang Jiang, (2004). A genetic algorithm for multiple resource-constrained project sheduling.). [interaktyvus][žiūrėta 2017-04-19]. Prieiga internete: http://ro.uow.edu.au/cgi/viewcontent.cgi?article=1256&context=theses*
33. *Anurag Agarwal , Selcuk Colak , Selcuk Erenguc, (2012). Using Firefly Algorithm to Solve Resource Constrained Project Scheduling Problem. Advances in Intelligent Systems and Computing book series , Vol. 201 p. 417-428. Prieiga internete: https://doi.org/10.1007/978-81-322-1038-2*
34. *Anurag Agarwal , Selcuk Colak , Selcuk Erenguc, (2006).Resource Constrained Project Scheduling: a Hybrid Neural Approach. International Series in Operations Research & Management Science Vol. 92 p 297-318. Prieiga internete: https://doi.org/10.1007/978-0-387-33768-5*

Priedai

Lentelė Nr. 6. Dviejų procesorių modeliuojamo atkaitinimo algoritmo veikimo rezultatų lentelė

		0	**1**	**10**	**100**	**1000**	**2000**	**5000**	**10000**	**15000**	**20000**
„Atsitiktinis“	K	218,647	1,368055	1,066529	1,011397	1	1,005168	1	1,002648	1,00395	1,004997
	lamda	1E+10	0,689448	0,689448	0,071389	0,007185	0,003601	0,00144	0,000724	0,000487	0,000368
	min	32,6312	0,001429	6,44E-05	1,33E-05	5,45E-08	5,45E-08	5,4E-08	1,4E-09	1,4E-09	1,4E-09
	max	54,0775	46,69519	5,602822	0,576072	0,068511	0,033089	0,01199	0,006589	0,00495	0,004351
	vidurkis	42,9501	4,752823	0,671943	0,07105	0,007185	0,003594	0,00144	0,000723	0,000486	0,000367
	mediana	42,9442	3,930204	0,477623	0,049268	0,004952	0,0025	0,00099	0,0005	0,000337	0,000255
	standartinis nuokrypis	2,89775	3,939143	0,632097	0,070353	0,007279	0,003589	0,00144	0,000721	0,000486	0,000369
„Priverstinis“	K	224,914	70,09204	9,382963	1	1	1,007146	1,00763	1,002536	1,015331	1,008637
	lamda	1E+10	48,17074	42,99362	9,232867	0,009768	0,004156	0,00155	0,00075	0,0005	0,000376
	min	32,9373	0,122654	0,023137	0,000593	8,03E-07	8,03E-07	3,3E-09	3,26E-09	3,26E-09	1,86E-09
	Max	53,2373	53,23733	53,23733	51,31539	0,08667	0,038316	0,01822	0,007776	0,005402	0,004283
	Vidurkis	43,0115	42,69646	39,79792	9,232867	0,009768	0,004144	0,00154	0,000749	0,000497	0,000374
	Mediana	43,0317	42,95844	42,26027	4,322775	0,006778	0,002879	0,00108	0,000522	0,00035	0,00026
	Standartinis nuokrypis	2,86108	3,999854	9,091187	11,05686	0,00976	0,004104	0,00153	0,000748	0,000495	0,000373
„10“	K	225,849	107,1746	1,096006	1	1	1,005069	1	1	1,008128	1
	lamda	1E+10	43,70323	4,779636	0,081107	0,007287	0,00362	0,00146	0,000727	0,000486	0,000364

	min	31,4842	23,48041	2,57E-05	1,82E-07	1,82E-07	1,78E-07	1,6E-08	1,63E-08	1,63E-08	1,63E-08
	max	53,893	47,32566	25,21964	1,017152	0,107292	0,031729	0,01464	0,00822	0,005616	0,003255
	vidurkis	42,9742	34,95891	4,624992	0,081107	0,007287	0,003612	0,00146	0,000727	0,000484	0,000364
	mediana	42,9353	34,8998	3,800287	0,054785	0,005037	0,002546	0,001	0,000505	0,00034	0,000253
	standartinis nuokrypis	2,85403	3,367249	3,850339	0,084269	0,007362	0,003557	0,00147	0,000736	0,000483	0,000364
„20“	K	220,708	52,19045	1	1	1	1	1	1	1	1
	lamda	1E+10	37,40695	1,317392	0,075918	0,007196	0,003601	0,00144	0,000723	0,000481	0,000365
	min	32,3199	15,17583	0,00016	1,08E-05	1,53E-07	1,53E-07	7E-08	6,98E-08	6,98E-08	6,98E-08
	max	53,6939	43,7573	13,12164	0,78122	0,062596	0,037627	0,01624	0,008461	0,004483	0,003556
	vidurkis	42,9885	28,46221	1,317392	0,075918	0,007196	0,003601	0,00144	0,000723	0,000481	0,000365
	mediana	42,9712	28,42256	0,853016	0,05282	0,004938	0,002518	0,00098	0,000502	0,000334	0,00025
	standartinis nuokrypis	2,88662	3,911735	1,418382	0,075934	0,007191	0,003606	0,00145	0,000732	0,000485	0,000365
„35“	k	216,758	20,49033	1,036936	1,009243	1,018606	1,013089	1,0196	1	1	1
	lamda	1E+10	27,65816	0,900462	0,074761	0,007318	0,003675	0,00147	0,000736	0,000493	0,000367
	min	32,2229	5,32508	0,000128	2,82E-05	6,92E-07	8,8E-08	8,8E-08	8,8E-08	8,8E-08	8,8E-08
	max	54,0738	38,98677	8,306712	0,548471	0,069115	0,03244	0,01231	0,006773	0,00476	0,003985
	vidurkis	42,9787	20,85611	0,887107	0,074475	0,007261	0,003655	0,00146	0,000736	0,000493	0,000367
	mediana	43,0123	20,81923	0,631292	0,052089	0,005088	0,002529	0,00102	0,000508	0,000334	0,000252

	standartinis nuokrypis	2,91278	4,502528	0,868915	0,073015	0,007211	0,00363	0,00143	0,000741	0,000508	0,000373
50	K	217,027	7,873428	1,065176	1,008914	1,002595	1	1,00459	1	1,000717	1
	lamda	1E+10	19,37427	0,806219	0,072792	0,007267	0,00361	0,00144	0,000722	0,000483	0,000363
	min	29,7838	0,010914	0,000103	8,95E-06	3,85E-07	3,85E-07	9,3E-10	9,31E-10	9,31E-10	9,31E-10
	max	53,7025	33,36743	7,065595	0,601102	0,066661	0,031218	0,01228	0,006425	0,005164	0,003649
	vidurkis	42,9962	15,17976	0,785978	0,072519	0,007259	0,00361	0,00144	0,000722	0,000483	0,000363
	mediana	42,9665	15,07537	0,561829	0,050664	0,005063	0,002477	0,00101	0,000499	0,000335	0,000252
	standartinis nuokrypis	2,91358	4,994123	0,748341	0,072319	0,00721	0,003604	0,00143	0,000724	0,000483	0,000362

.