KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Aurimas Norkus

Pamokų tvarkaraščio optimizavimas profiliuotoms mokykloms

Magistro darbas

Darbo vadovas
prof. habil. dr. J. Mockus

Kaunas, 2005

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

TVIRTINU
Katedros vedėjas
(parašas) doc. dr. E. Bareiša
2005 05

Pamokų tvarkaraščio optimizavimas profiliuotoms mokykloms

Magistro darbas

Kalbos konsultantė
Lietuvių kalbos katedros lektorė
(parašas) dr. J. Mikelionienė
2005 05

Vadovas
(parašas) prof. habil. dr. J. Mockus
2005 05

Recenzentas
(parašas)dr. G. Palubeckis
2005 05

Atliko
IFM-9/1 gr. stud.
(parašas) A. Norkus
2005 05

Kaunas, 2005

Turinys

Sutrumpinimų sąrašas

SA – *Simulated Annealing*

HC – *Hill Climbing*

TPSA – *Tenperature Parallel Simulated Annealing*

GA – *Genetic Algorithm*

TSP – *Traveling Salesman Problem*

PDF – *probability density function*

BA – *Bayesian Approach*

MCMC - *Markov Chain Monte Carlo*

Santrauka

Šiame darbe realizuoti pamokų perstatinėjimo, pamokų perstatinėjimo pritaikius *simulated annealing* (SA) ir pamokų perstatinėjimo panaudojant *Bayesian Approach* (BA) teoriją *simulated annealing* parametrų optimizavimui algoritmai. Algoritmai bei grafinė vartotojo sąsaja realizuota su JSP, kurios pagrindas yra Java objektinio programavimo kalba. Tam kad būtų galima įvertinti tvarkaraščio gerumą, yra skaičiuojami kiekvieno tvarkaraščio baudos taškai, kurie skiriami už tam tikrų apribojimų pažeidimą. Vartotojas turi galimybę įvesti baudos taškų kiekį, kuris skiriamas už apribojimo pažeidimą. Taip pat jis turi galimybę pasirinkti stochastinių algoritmų parametrus.

Atlikta teorijos apžvalga, kur apžvelgiama SA ir BA teorijos naudojimą įvairiuose stochastiniuose algoritmuose ir jų deriniuose.

Aprašytas sukurtas profiliuotų mokyklų tvarkaraščio optimizavimo algoritmas, kuris pasitelkia SA koncepciją: geriausio ieškojimas per blogesnius sprendinius, temperatūros funkcija, skirtumas tarp kokybės. Siekiant dar labiau optimizuoti tvarkaraščio sudarymą buvo sukurtas BA naudojantis algoritmas, kuris stengiasi prognozuoti SA parametrus. Eksperimentavimo tikslu vartotojas gali keisti SA elgesį keisdamas sistemos temperatūros arba atšaldymo greičio kintamuosius. Naudojant algoritmą pritaikytą BA, vartotojo įvesti SA parametrai neturi didelės reikšmės, nes pats algoritmas prognozuoja, jo manymu, geresnius parametrus, prie kurių SA turėtų veikti efektyviau.

Pasinaudojus darbo metu sukurta programa buvo atlikti išsamūs trijų stochastinių algoritmų praktiniai tyrimai. Eksperimentų metu analizuotas atsitiktinumo faktoriaus naudojimas optimizuojant tvarkaraštį, kai esant tiems patiems kintamiesiems gaunami

skirtingi tvarkaraščių variantai. Įvertinta tinkamų parametrų pasirinkimo įtaka stochastiniam algoritmui. Tirtas globalaus minimumo ieškojimas kai funkcija turi daug lokalių minimumų ir nagrinėta sąlyga, kada stochastinis algoritmas rado globaliai geriausią variantą. Taip pat įvertinta pradinio tvarkaraščio sudarymo įtaka vėlesniam optimizavimo procesui.

Visi eksperimentai buvo atlikinėjami su realioje mokykloje naudojamais tvarkaraščiais. Pradinio tvarkaraščio varianto sudarymui buvo naudoti paprastas-pamokų įdėjimo ir stochastinis, įvedantis atsitiktinumo faktorių ir atliekantis pamokų išmaišymą, algoritmai.

Summary

There are three implemented algorithms in this work: lessons permutation, lessons permutation with simulated annealing adjustment, lessons permutation using Bayesian approach theory to optimize SA parameters algorithms. Algorithms and graphical user interface are programmed with JSP which is based on Java object programming language. To evaluate schedule goodness algorithms are computing every penalty points which are given for some inconvenieces. User is able to define how much penalty points will be given if some inconveniece is satisfied. Also he is able to assign stochastic algorithm parameters.

There was accomplished theory, where was observed using of simulated annealing and Bayesian approch methods in other stochastic algorithms and their different combination.

There is a description of profiled school schedule optimization algorithm, which is based on SA searching methodology: searching for the optima through lower quality solutions, using temperature function which convergence, difference in quality. Algorythm which is using BA was created in case to improve SA searching methodology. User by changing systems temperature or annealing speed througth parameters can make big influence to SA behaviour. Passing parameters then using algorithm with BA meaner influence is made to behaviour because this method prognosticates, acording to him, better parameters with which SA should work effectively and changing them.

Researches with three stochastic algorithms were made after program was created. Analysed using randomization factor then optimizing schedule then different schedule variant are computed with same parameters. Exploring what big influence is made to stochatic algorithm then choosing correct parameters. Researched global minimum

searching then function has a lot of local minimums. Analysed condition then stochastic algorithm found global minimum, studing influnece of making initial schedule to later optimization process.

Whole experiments were performed with schedules from real Lithuanian school. For the calculation of initial schedule were used simple and stochastic, with coincidence factor mixing up lessons, algorithms.

1. Įvadas

Kiekvienas siekia racionaliai išnaudoti turimą laiką ir resursus, tuo siekdamas savo veiksmų efektyvumo, darbų našumo ir produktyvumo. Taigi mes planuojame, sudarinėjame darbotvarkę, kitaip sakant, kuriame tvarkaraščius kiekvienai savo organizuotai veiklai. Resursai - tai įrankiai, mašinos, medžiagos, darbo jėga, fiziniai pajėgumai. Savaime suprantama, tvarkaraščio sudarymas organizacijoms yra daug sudėtingesnis nei paprastam žmogui. Netinkamas laiko arba resursų paskirstymas daro įtaką neigiamo rezultato gavimui ir gali daug kainuoti.

Mokykla yra organizacija, kuri turi planuoti pamokas, jų vykimą, atsižvelgdama į mokinius, mokytojus, turimas klases ir inventorių (kompiuteriai ir kt.). Galime traktuoti pamokas kaip veiklą, o mokinius, mokytojus, klases ar inventorių kaip resursus, todėl žmogus, sudarinėjantis tvarkaraštį, sprendžia pakankamai sunkų klausimą, kaip jį sudaryti, kad tenkintų ir mokytojus, ir mokinius, ir neviršytų turimų išteklių, o ištekliai nėra begaliniai: mokytojų ir klasių skaičius yra ribotas, klasė gali talpinti tik tam tikrą kiekį mokinių, laiko apribojimai. Profiliuotoje mokykloje reikia sudarinėti tvarkaraštį kiekvienam mokiniui, kurių vidutiniškai apie 60-70 tenka 11 ir tiek pat 12 klasėms, ir kurie renkasi dalykus iš vidutiniškai 50-60 pasirenkamų dalykų (į tai įeina ir pogrupiai, ir skirtingi lygiai).

Palyginimui galima pasakyti, kad optimalaus tvarkaraščio sudarymas kiekvienai duotai pamokų sekai priklauso NP-complete uždavinių klasei. Norint rasti tikslų sprendinį tektų peržiūrėti 2 laipsnyje *m* tokių sekų, kur *m* – savaitinių pamokų skaičius. Todėl naudojamos euristikos, kai peržiūrimi ne visi leistini tvarkaraščiai, o tik tam tikra „pretendentų“ aibė. Anot S. H. Jacobson ir E. Yucesan vienas iš tokių euristikos algoritmų yra SA (*simulated annealing*), kurio esmė yra rasti geriausią galimą sprendinį atliekant kuo mažiau skaičiavimų. *Simulated annealing* algoritmas naudoja porą parametrų, nuo kurių labai priklauso rezultatas, todėl šių parametrų parinkimas yra svarbus optimizavimo procesui. Parametrus galima keisti, kad sumažinti vidutinį nukrypimą nuo globaliai geriausio varianto. Kaip teigia J. Mockus, dauguma atvejų vidutinis nukrypimas yra stochastinė euristinių parametrų funkcija, vadinasi parametrus galima optimizuoti. Bayeso euristikos algoritmas tinka šiai optimizacijai. Šis sprendinys nebus garantuotai optimalus ir paklaida priklausys nuo šio metodo kokybės ir efektyvumo. Todėl euristinių metodų tyrimas taikant juos tvarkaraščių optimizavimo uždaviniams yra svarbi praktinė ir mokslinė problema.

2. Teorijos apžvalga

2.1 SA teorijos apžvalga

Pirmasis, kuris pasiūlė naudoti *simulated annealing* kaip algoritmą spręsti kombinatorines optimizavimo problemas, buvo Kirckpatric. Jis pavadino savo sprendimo būdą kaip imituojantis atkaitinimas, nes algoritmas pamėgdžioja molekulių energijos sumažinimą metale. SA buvo sėkmingai taikomas spręsti keletui optimizavimo problemų tipų. Vis dėlto egzistuoja keletas didžiulių problemų: reikia vis tiek daug laiko tam, kad surasti optimaliausią sprendinį ir sunku apibrėžti tinkamą SA atšaldymo temperatūrą.

Grupavimas (*clustering*) turi daug pritaikymų tokie kaip augmenijos ir gyvūnijos sistematikos generavimas, koncepcijų formavimas, duomenų gavyba. Grupavimo sistemos dirba su objektų kolekcijomis, kurių kiekvienas yra aprašytas aibe iš *n* parametrų. Grupavimas stengiasi sukurti klases, jų aprašymus ir tuos objektus priskirti toms klasėms. *Knowledge representation scheme* (KRS) apibrėžia modelių ieškojimo erdvę, o kriterijus įvertina kiekvieno modelio gerumą. Paieškos algoritmas tyrinėja erdvę ir bando surasti modelį labiausiai atitinkantį kriterijų.

Ian Davidson pasiūlė naudotis SA ieškojimo algoritmo optimizavimui. Kiekvienu momentu egzistuoja vienas sprendinys, kuris šiek tiek keičiasi kiekvienos iteracijos metu. SA be jokių sąlygų priima geresnio modelio variantą ir su sąlyga *p* priima jei variantas blogesnis, kur :

$$p = e^{-\left(\frac{Difference\ in\ quality}{System tem\ temperature}\right)} \quad (1)$$

SA priimdamas blogesnį variantą, padeda išvengti lokaliai geriausio sprendinio problemos, kurią turi dauguma sudėtingų paieškos algoritmų. I. Davidson priskiria pradinę temperatūrą T_0, kur $T_0 = -\frac{C_0}{\log_e(0.5)}$, taigi su 50% tikimybe blogesnis variantas bus priimtas. Tikimybė mažėja kai temperatūra mažėja. Atšaldymo konstanta R mažina temperatūrą $T_k = T_{k-1}*R$. C_0 pradinio sprendinio gerumo įvertinimas.

Optimizavimo algoritmas :

Pasirinkti pradinį sprendinį ir pradinę temperatūrą

Optimizavimas : Generuoti naują sprendinį keičiant senąjį
Jei naujasis sprendinys geresnis už senąjį
senas sprendinys := naujasis sprendinys
kitaip su tikimybe *p*
senas sprendinys := nausis sprendinys
Jei gerumas ilgą laiką nepadidėjo
sumažinti temperatūrą
Jei temperatūra lygi 0 sustoti
Goto **Optimizavimas**

I. Davidson teigia, kad *simulated annealing* statistiškai garantuoja surasti globaliai geriausią variantą, bet tai be galo ilgas procesas ir reikalaujantis daug iteracijų. Vis dėlto SA metodas nėra idealus ir negarantuoja optimaliausio sprendinio kai reikia jį surasti per trumpą laiko tarpą.

Kadangi sprendžiant optimizavimo problemas reikalaujama daug iteracijų ir nemažai laiko, atsiranda paskata procesą sulygiagretinti. *Simulated annealing*, kuris yra efektyvus sprendžiant sudėtingas optimizavimo problemas, naudoja milžinišką skaičiavimo galią. Progresuojant paralelinėms kompiuterinėms sistemoms buvo pasiūlytas *temperature parallel simulated annealing* (TPSA), kuris galėjo rasti optimalųjį sprendinį ir galėjo sumažinti laiko sąnaudas šiam procesui.

Tačiau priskyrus per aukštą arba per žemą temperatūrą paraleliniam procesui, SA nekonverguoja prie geriausio sprendinio, todėl tai sumažina TPSA efektyvumą. Tam kad būtų išspręstas šis efektyvumo praradimas, M. Miki, T. Hiroyasu ir J. Wako pasiūlė naują metodą, kuris nustato adaptuojančias atšaldymo temperatūras kiekvienam lygiagrečiam SA procesui: temperatūros dinamiškai keičiamos kiekvienam procesui priklausomai nuo rezultato kiekvienos iteracijos metu. Atšaldymo temperatūros prisitaikymą vykdo *genetic algortihm* (GA) pagalba. GA idėja yra panaši į natūralios atrankos procedūrą biologijos evoliucijoje: atliekami perėjimo ir mutacijos (keitimosi) veiksmai. Kiekvienas sprendinys-kandidatas yra vaizduojamas kaip simbolių eilutė. Aibė tokių sprendinių prie būsenos *m* yra vadinama *m*-tosios kartos populiacija. Perėjimo operacija padalina sprendinį-kandidatą į dvi dalis iki atsitiktinės pozicijos *s*, o mutacijos veiksmas vykdo

simbolių sukeitimą tarp abiejų dalių. GA įvertinimo funkcija nusprendžia mutavusių dalių tinkamumą.

Lygiagretaus SA su GA pritaikoma temperatūra (*parallel simulated annealing with adaptive temperature determined by* GA (PSA/AT(GA))) buvo naudojamas keliaujančio pardavėjo problemai (TSP) spręsti. TSP problema yra parinkti kuo optimalesnį maršrutą tarp atskaitos taškų ir išreiškiama:

$$\sum_{i=1}^{n-1} d(v_\pi, v_\pi(i+1)) + d(v_\pi(N), v_\pi(1)) \quad (2),$$

kur $v(i)$ yra i-tasis atskaitos taškas (pvz.: miestas) maršrute π, $d(v(i), v(j))$ atstumas tarp taškų ir $d(v(i), v(j)) = d(v(j), v(i))$. Pastebėta, kad egzistuoja temperatūros regionai, kurie priklauso nuo taškų skaičiaus N, t. y. geriausio varianto ieškojimas efektyvesnis su tam tikra temperatūra prie tam tikro atskaitos taškų N skaičiaus. Paraleliai GA generuoja skirtingas temperatūras ir atsimenamos tik tos, kurioms esant, buvo pasiekti pakankamai geri rezultatai. Taip artėjama prie optimaliausio sprendinio.

2.2 BA teorijos apžvalga

Bayeso statistika kilo iš nepasitenkinimo statistikos pamatais :

- Venn Limit netikslus tikimybės apibrėžimas.
- Reikšmingas testavimas (*significance testing*) nesuderinamas su sprendinių teorija.

Klasikinė tikimybė, kurios apibrėžimas yra santykis tarp tinkamų įvykių ir visų įvykių skaičiaus, pasidaro nenaudinga kai Venn Limit teorija taikoma nagrinėjant begalybė įvykių. Reikšmingas testavimas taip pat duoda netikslius atsakymus, jeigu priimami sprendimai kai yra nežinomybė.

Sprendinių teorija naudoja tikimybes (*probabilities*) ir naudingumus (*utilities*). Šioje teorijoje kiekvienas įvykis turi tam tikrą tikimybę, kuri apibrėžia kaip tikėtina, kad jis įvyks, o naudingumas apibūdina kaip norimas, tinkamas yra šitas įvykis. Tikimybė ir naudingumas elgiasi pagal tam tikras racionalias taisykles ir yra tarpusavyje susiję. Sprendinių teorijos svarbiausia idėja, kad iš susietų tikimybių ir naudingumų, esant

nežinomybei, sugebama pasiūlyti naują subjektyvią įvykio tikimybę. Nešališka tikimybė gali būti traktuojama kaip šansas.

Bayeso tikimybių teorija siejasi su sprendinių teorija ir įveda dar papildomus apibrėžimus :

- tikimybė yra nežinomybės įvertinimas.
- tikimybės turi būti tarpusavyje susietos.
- nežinomybė turi būti visą laiką įvertinta tikimybės pasiskirstymu.

Nežinomybė, įvertinta tikimybe $p(E)$, tenkina tris aksiomas, kurios yra būtinos ir pakankamos :

- *Additive Law*
  Nesuderinamiems įvykiams E_1 ir E_2 galioja $p(E_1 \text{ arba } E_2) = p(E_1) + p(E_2)$.
- *Multiplicative Law*
  Nepriklausomiems įvykiams E_1 ir E_2 galioja $p(E_1 \text{ ir } E_2) = p(E_1) * p(E_2)$.
- *Convexity Law*
  p yra apibrėžtas, egzistuoja a ir b tokie, kad $a \le p \le b$, pagal susitarimą $a = 0$, $b = 1$.

Tegu A ir B yra įvykiai. Tegu žinoma $p(A|B)$ ir norima sužinoti $p(B|A)$.

$$p(AB) = p(B|A)\,p(A) = p(A|B)\,p(B) \quad (3)$$

vadinasi :

$$p(B|A) = p(A|B)\,p(B) / p(A) \quad (4)$$

Tarkime, kad turime stebimą įvyki y, kuris, tarkime, kad priklauso nuo įvykių θ, kurio įvykimo tikimybės funkcija yra $p(\theta)$, tada iš lygybės (4) :

$$p(\theta|y) = p(y|\theta)\,p(\theta) / p(y) \quad (5)$$

Laikydami, kad $p(y|\theta)$ yra θ funkcija, kuri vadinama kaip *tikėtina įvykio tikimybė* (sąlyginė įvykio y tikimybė įvykus θ įvykiui), apibrėžkime $l(y|\theta)$. Taigi Bayeso teoremą galima užrašyti taip :

$$p(\theta|y) \propto l(y|\theta)\, p(\theta) \quad (6)$$

Kitaip tariant *Būsimas įvykis* $\propto$ *Tikėtinas* $*$ *Praeitas įvykis.*

Nors BA teorija pasiūlo užtikrinantį metodą nežinomybės įvertinimui, reikia išspręsti dvi problemas. Viena iš jų yra kaip apibrėžti tikėtinumo funkciją. Pagal S. Subbey, M. Christie, M. Sambridge, kadangi tikėtinumo modelis vertindamas nežinomybę įvertina einamojo modelio duomenis, yra būtinas poveikis prieš tai buvusio tikėtinumo modelio. Beje, tai yra įmanoma tik tada, kai ankstesnis tikėtinumas yra teisingai apibrėžtas. Netgi tuo atveju, kai praeitis yra miglota ir ne vienoda, prastai apibrėžtos tikėtinumo funkcijos rezultatas būtų prastas vėlesnis tikimybių pasiskirstymas. Jei tartumėm, kad tai geras modelis, tada bet koks prieštaringas tinkamumo įvertinimas lyginant su einamojo modelio rezultatu gali būti priskirtas klaidai. Priklausomai nuo klaidingo ar pakankamai teisingo rezultato vėlesnis tikėtinumas keičiamas.

Kita problema realizuojant BA yra apskaičiuoti pastovaus proporcingumo (6) lygtį. Užduotis modeliuojant net ir mažas problemas nėra paprasta. Vienas iš daugumos *Markov Chain Monte Carlo* (MCMC) algoritmas generuoja realizacijų sekas, kurie yra vėlesnio pasiskirstymo šablonai. MCMC metodas, eidamas per sugeneruotas sekas, kuria naujus modelius kurie yra arba priimami, arba atmetami priklausomai nuo pasirinkto kriterijaus. Tikslas, naudodamasis išrinktais baigtinio skaičiaus modeliais, sugeneruoti naują rinkinį modelių, kurių pasiskirstymas asimptotiškai bent kiek panašus į rinkinį modelių iš ankstesnių pasiskirstymų. Taigi BA metodą galima laikyti kaip pažinimo proceso modelį, kuris remiasi praeitų modelių patirtimi.

Sondquist ir Morgan (1963) pasiūlė originalų prognozuojantį CART medžio algoritmą kai duoti susieti tarpusavyje duomenys. Tai klasikinis statistinis metodas. Iš susietų duomenų galima sudaryti medžio tipo duomenų struktūrą, kur yra mazgai, lapai ir sąlyginiai ryšiai tarp mazgų ir lapų. Apibrėžti trys vaikščiojimo po medžio erdvę būdai : auginimas, kai pridedami mazgai arba lapai, keitimas, kai pagal duotą sąlygą pakeičiamas medis, genėjimas, kai mazgų arba lapų skaičius yra sumažinamas.

CART pagrindinė idėja :

- klasifikuoti besąlygiškas medžio dalis kiekviename mazge kai duoti aiškinamieji kintamieji;
- nuspėti medžio dalį, kurią galima skelti kai duoti aiškinamieji kintamieji;
- pasirinkti geriausią perskėlimą;

Kylančios problemos :

- jeigu spėjamieji kintamieji yra susieti tarpusavyje daugybe medžių duos gerą sprendinį;
- kada geriausia nustoti skaidyti medį į dalis;
- suskaidytos medžių dalys pasidaro per didelės;
- paprasta paieška suras tik lokaliai geriausią sprendinį;

CART algoritmui buvo pritaikytas BA :

- pradėti nuo bet kokio medžio (paprastai nuo tuščio).
- kartojama daug kartų :
  - atsitiktinai pasirinkti judesio tipą;
  - judėdamas atsitiktinai generuoti kandidatus;
  - stochastiškai priimti/atmesti kandidatą;
- generuoti skirtingas medžių sekas.

BA modelis naudojamas taip pat neuroniniuose ir Bayeso tinkluose.

3. *Simulated Annealing* teorinis tyrimas

Diskretinio optimizavimo probleminę sritį apibrėžia baigtinių sprendinių aibė ir funkcija, kuri yra asocijuota su kiekviena reikšme iš tos sprendinių aibės. Sakykime Ω yra sprendinių aibė. Apibrėžkime funkcija f: $\Omega \to [0, +\infty]$, kuri priskiria ne neigiamą reikšmę kiekvienam sprendinių aibės elementui. Kiekvienas *Hill Climbing* (HC) grupės algoritmas turi du svarbius komponentus: kaimyninė funkciją, η: $\Omega \to 2^{\Omega}$, kur $\eta(\omega) \subseteq \Omega$ visiems $\omega \in \Omega$ ir atsitiktinius skaičius R_k: Ω x $\Omega \to \Re$, k = 1, 2, … Kiekvienam sprendiniui $\omega \in \Omega$ kaimyninė funkcija $\eta(\omega)$ apibrėžia aibę sprendinių artimų ω. Taigi HC algoritmas kiekvienos iteracijos metu sugeneruoja atsitiktinį sprendinį tarp visų einamojo sprendinio kaimynų, kur atsitiktinių skaičių sekos yra nepriklausomos.

Minimizuojant, kai atsitiktinai sugeneruoto kaimyninio sprendinio funkcijos reikšmė yra didesnė nei einamojo sprendinio, jis priimamas (pasidaro einamuoju sprendiniu) jei skirtumas tarp funkcijos ir kaimyninės funkcijos reikšmių yra pakankamai mažas (pvz.: mažesnis už HC atsitiktinai sugeneruotą skaičių). Šitaip naudojantis kaimynine funkcija nustatomi ryšiai tarp sprendinių pačioje sprendinių aibėje. Funkcija f ir kaimyninė funkcija η leidžia sprendinių aibę Ω suskirstyti į tris nepersidengiančias aibes:

- globaliai optimalių sprendinių aibė $\mathbf{G} = \{\omega^* \in \Omega: f(\omega^*) \le f(\omega)$ visiems $\omega \in \Omega\}$;
- lokaliai optimalių sprendinių, kurie nėra globaliai optimalūs $\mathbf{L} \equiv \mathbf{L}(\eta) = \{ \omega \in \Omega \backslash \mathbf{G}: f(\omega) \le f(\omega')$ visiems $\omega' \in \eta(\omega)\}$;
- *hill* sprendinių aibė $\mathbf{H} = \Omega \backslash (\mathbf{G} \cup \mathbf{L})$;

Taigi $\mathbf{G} \cup \mathbf{L}$ yra lokaliai optimalūs sprendiniai Ω aibėje, kurie yra kaimyninės funkcijos η sprendiniai, kur $\Omega = \mathbf{G} \cup \mathbf{L} \cup \mathbf{H}$ kai $\mathbf{G} \cap \mathbf{L} = \varnothing$, $\mathbf{G} \cap \mathbf{H} = \varnothing$ ir $\mathbf{L} \cap \mathbf{H} = \varnothing$. Taip pat visiems $\omega \in \mathbf{G}$, $\eta(\omega) \cap \mathbf{L} = \varnothing$ ir visiems $\omega \in \mathbf{L}$, $\eta(\omega) \cap \mathbf{G} = \varnothing$ t. y. globaliai ir lokaliai optimalūs sprendiniai negali būti kaimynais.

HC algoritmas, skaičiuodamas kaimyninės funkcijos reikšmes, keliauja per sprendinių aibę įtraukdamas prastesnius variantus artėja prie globaliai optimalaus sprendinio. Kadangi geriausias variantas yra nežinomas, tai ir algoritmas nežinos ar jis konvergavo ar ne, todėl geriausias sprendinys nebūtinai bus paskutinis nagrinėtas variantas, bet bus tikrai vienas iš jau nagrinėtųjų.

SA yra tipinis HC grupei priklausantis algoritmas, kur $R_k(\omega(i), \omega) = -t(k) \ln(v_k)$, $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, $k = 1, 2, \ldots$, kur $t(k)$ yra temperatūros parametras (taigi $t(k) \to 0$

kai $k \to \infty$) ir v_i nepriklausomai pasiskirstę atsitiktiniai skaičiai. Priimti varianto pagerinimą arba *Hill Climbing* judesio tikimybė $R_1(\omega(i), \omega) \geq \delta(\omega(i), \omega)$ *simulated annealing* naudojama tokia tikimybe $v_i \leq e^{-\frac{\sigma(\omega(i),\omega)}{t(k)}}$. Hajek(1988) parodo, kad SA konverguoja su tikimybe prie globaliai geriausio varianto jeigu tik $\sum_{k=1}^{+\infty} e^{-(d^*/t(k))} = +\infty$, kur *t(k)* yra temperatūra, kuri artėja prie 0 kai $k \to \infty$ ir d* skirtumas tarp vietinių geriausių sprendinių.

Naudojama tokia euristika, kuri tinkama *simulated annealing* :

$$h_i = v(m) - v(\overline{m}), m = 0,1.$$

čia

$v(m)$ yra geriausias rastas sprendinys po *i*-tosios iteracijos.

$v(\overline{m})$ yra einamasis sprendinys po *i*-tosios iteracijos.

Tada atsitiktinė perstatymo euristika yra :

$$r_i(m) = \begin{cases} e^{\frac{-h_i(m)}{x_1 / \ln(1+x_2 N)}}, if\ h_i(m) < 0 \\ 1 \qquad\qquad\qquad\quad , otherwise \end{cases} \quad (7)$$

kur

i – iteracijos skaičius;

x_1 – pradinė temperatūra;

x_2 – atšaldymo greitis;

N – iteracijų skaičius;

Šiuo atveju *t(N)* yra temperatūros funkcija, kuri artėja prie 0, kai $N \to \infty$. Įvedami x_1 ir x_2 parametrai tam, kad būtų galima reguliuoti temperatūros konvergavimą prie 0: pradinė sistemos temperatūra ir jos atšalimo greitis.

4. *Bayesian Approch* teorija

Kiekvieną kartą, kai pradinis tvarkaraštis yra optimizuojamas pasinaudojus SA gaunami variantai su skirtingais baudos taškais, taigi baudos taškų funkcija *f* turi daug minimumų visoje reikšmių srityje **Ω**, todėl galima pritaikyti stochastinę globalaus optimizavimo Bayeso sprendinių teoremą, kuri prognozuoja *simulated annealing* parametrus. Taip pat kiekvienas stochastinis algoritmas veiks neefektyviai jei bus parinkti netinkami parametrai, todėl pritaikoma BA tam, kad būtų išvengta žmogiškojo faktoriaus įvedant parametrus. Bayeso teorijos pagrindas yra kaip įvertinti nežinomybę skaitine verte, remiantis jau atliktais modeliavimais. Vienas iš Bayeso priėjimo(BA) prie nežinomybės naudingumų, kad kiekvienam modeliavimo aspektui priskiriama tikimybės tankio funkcija (PDF), kuri įvertina kiek yra žinoma nežinomybė. Nežinomybė išreiškiama tikimybine tankio funkcija $p(\theta)$, kai yra priskiriama modelio tikimybė, bet neatsižvelgiama į praeitus duomenis. Priskirti sekančiam modeliui paprastą tikimybę nėra geriausias variantas, todėl modeliuojant vėlesnių tikimybinės tankio funkcijų $p(\theta \mid y)$ išraiškas yra atsižvelgiama į praeitus duomenis. Kitas svarbus parametras - kiekvieno modelio tikėtinumas. Tikėtinumo tikimybė $p(y \mid \theta)$ yra apskaičiuojama lyginant esamuosius duomenis su gautais viso proceso metu duomenimis. Vėliau esamieji duomenys laikomi abejotinais. Taigi pagal Bayeso teoremą:

$$p(\theta \mid y) \propto p(y \mid \theta)\, p(\theta) \quad (8)$$

Pagrindinis BA privalumas prieš kitus tradicinius nežinomybės analizavimo metodus yra tai, kad jis leidžia naudoti pasirinktą tikimybės pasiskirstymą, ne vien tik Gauso pasiskirstymą, ir pasirenka nežinomybės įvertį, ne vien tik variacijas. Taip pat BA sugeba analizę praplėsti iki aukštesnio laipsnio interpretacijos nei paprasti algoritmai, t. y. atmesti tam tikrus ir pasirinkti tinkamus modelius. Vertinant nežinomybę pasitelkiama automatiškai generuojamais praeities modeliais, kurie remiasi *ankstesniaisiais* ir *tikėtinumo* modeliais.

Algoritmas naudoja *probability* kintamąjį, kuris apsprendžia ar gerinti pasirinktam mokytojui tvarkaraštį, o SA naudoja X_1 (sistemos temperatūra) ir X_2 (atkaitinimo kintamasis). Pradines reikšmes įveda vartotojas. Prie šių parametrų apskaičiuojamas tvarkaraštis ir BA metodas analizuoja gautą rezultatą ir pagal jį prognozuoja naujus parametrus, prie kurių turėtų būti gaunamas geresnis optimalesnis rezultatas. Taip keičiant *simulated annealing* parametrus, galima rasti tvarkaraščio su mažiausiu baudos taškų kiekiu visoje reikšmių srityje, t. y. rasti globalų minimumą.

5. Programinė realizacija

Siekiant žmogui palengvinti tvarkaraščio sudarymo darbą, kuriama programa, kuri sudarinėtų optimizuotus tvarkaraščius Lietuvos mokykloms. Programa realizuota JSP (*Java Server Page*) programavimo kalba. JSP tai *client-server* sistemos architektūros paremta programavimo kalba, kurios pagrindas yra JAVA. Sistemą sudaro *tomcat* serveris, kuris atlieka visus skaičiavimus, ir klientas. Klientas - tai internetinė naršyklė, tokia kaip *Opera*, *Mozila*, *Internet Explorer,* arba kitos. Tokia sistema išnaudoja interneto privalumus: platformos nepriklausomumą ir pasiekiamumą iš visur, kur yra kompiuteris su internetu. Taip pat visus skaičiavimus atlieka serveris, o ne kliento kompiuteris, todėl pakanka galingo serverio, o vartotojas gali turėti ir ne tokį galingą kompiuterį.

5.1 Algoritmo aprašymas

Algoritmas atlieka pradinio tvarkaraščio sudarymą, o po to optimizavimą, kuris galimai sumažintų „langų“ skaičių bei kitus nepatogumus. Žinoma, tvarkaraštis turi tenkinti tam tikrus fizinius apribojimus :

- vienu metu mokytojas gali būti tik vienoje vietoje;
- vienu metu mokinys gali būti tik vienoje vietoje;

Taip pat egzistuoja ir normatyviniai apribojimai :

- mokinių „langai“ yra neleistini (kai mokinys neturi pamokos, prieš kurią, arba po kurios jis ją turėjo);
- tuščios mokytojų pamokos yra neleistinos (kai mokytojas neturi pamokos, prieš kurią, arba po kurios jis ją turėjo);
- „dvigubos“ pamokos (iš eilės einančios to pačio dalyko pamokos) yra neleistinos (yra išimčių);
- pamokų skaičius turi neviršyti tam tikro skaičiaus;

Esant tokioms pusėms (mokytojai, mokiniai) ir sąlygoms (pasirenkamieji dalykai, individualūs tvarkaraščius, resursų ribotumas, fiziniai ir normatyviniai apribojimai) galime teigti, kad tvarkaraščio, visiškai tenkinančio visas puses, nėra, nes gerinant vienam mokytojui arba mokiniui sąlygas, yra prastinamos sąlygos kitam mokytojui arba mokiniui. Kadangi fizinių apribojimų negalima niekaip pažeisti, įvedamas baudos taškų

skyrimas už normatyvinių apribojimų pažeidimą. Baudų taškai leidžia apibrėžti geriausią tvarkaraštį, t. y. geriausias tvarkaraštis turės mažiausiai baudos taškų, o tai reiškia, labiausiai tenkins visas puses ir mažiausiai pažeis normatyvinius apribojimus. Taigi, baudos taškai yra duodami :

- už mokytojo „langus“;
- už mokinio „langus“;
- už pamokas per mokytojų laisvadienius;
- už neleistiną pamokų seką;
- už didesnį pamokų skaičių nei norima;

Tarkime, kad:

1. c_i bauda už *i-tojo* mokytojo „langą“;
2. c_k bauda už *k-tojo* mokinio „langą“;
3. c_{il} bauda už *l-tąją* dieną, kuri netinka *i-tajam* mokytojui;
4. c_s bauda už netinkamą pamokų seką;
5. c_m bauda, jei pamokų skaičius viršija norimą pamokų per dieną skaičių;

Taigi, bendra baudų skaičiavimo funkcija yra:

$$C_c = \sum_i c_i L_i + \sum_k c_k L_k + \sum_i \sum_l c_{il} L_{il} + \sum_s c_s L_s + \sum_m c_m L_m \quad (9),$$

kur L_i – *i*-tojo mokytojo tuščių pamokų skaičius; L_k – *k*-tojo mokinio tuščių pamokų skaičius; L_{il} – netinkamų *i*-tajam mokytojui dienų skaičius; L_s – netinkamų pamokų sekų skaičius; L_m – dienų skaičius, kuriose yra daugiau pamokų, nei norima.

Algoritmas susidaro iš dviejų dalių: pradinio tvarkaraščio sudarymo ir pradinio tvarkaraščio optimizavimo. Visų pirma, duomenys yra nuskaitomi iš duomenų failo. Jei vartotojas yra nustatęs pamokas, kurios gali eiti viena po kitos, tai jos yra sudedamos į tvarkaraščio priekį. Po šių perstatymų išrenkamas tuščias langas (diena ir valanda) ir pamoka, tada tikrinama, ar ši pamoka gali vykti, jei įstatysime į pasirinktą langą (tenkinami fiziniai apribojimai, kad nei mokinys, nei mokytojas vienu momentu negali būti dvejose vietose). Jei gali vykti, tada įstatoma pamoka į langą, jei negali – sekanti pamoka pasirenkama ir taip daroma tol, kol randama tinkanti pamoka. Kai sudarinėjamas

pradinis tvarkaraštis nekreipiamas dėmesys į mokytojų ar mokinių tuščius langus ir norima, kad pamokų skaičius per dieną neviršytų pasirinkto skaičiaus. Jei atsitinka, kad sudarius tvarkaraštį pamokų skaičius per dieną viršija nustatytą skaičių, tai tvarkaraštis išmaišomas (eilutės sukeičiamos vietomis) ir sudarinėjamas iš naujo. Maišymas vykdomas 100 kartų ir nutraukiamas jei pamokos sutilpo. Jei ir tai nepadėjo, skaičius, kurį nustatė vartotojas, padidinamas vienetu ir už tokį apribojimo sulaužymą pridedame 10 baudos taškų. Po pradinio tvarkaraščio sudarymo iš galo atkeliamos pamokos, kurias lanko daugiau mokinių, ir sukeičiamos su priekinėmis pamokomis, kurias lanko mažiau mokinių, tam kad būtų sumažintas mokinių „langų“ skaičius.

Kita dalis yra optimizuoti pradinį tvarkaraštį, kad jis turėtų kuo mažiau baudos taškų. Vartotojas gali pasirinkti vieną iš trijų algoritmų: optimizuoti perstatinėjant pamokas, optimizuoti naudojant *simulated annealing* algoritmą ir optimizuoti naudojant Bayeso sprendimų teorijos algoritmą, kuris parenka *simulated annealing* paremetrus.

5.2 Optimizavimas naudojant pamokų perstatinėjimą

1. Iteracijos skaitliukas padidinamas vienetu.
2. Pasirenkamas sekantis mokytojas iš mokytojų sąrašo.
3. Pasirenkamas atsitiktinis dydis kuris $\in [0..1]$.
4. Jei atsitiktinis dydis yra mažesnis už vartotojo įvestą tikimybę, tada pasirinkto mokytojo tvarkaraštis nagrinėjamas (gerinamas), kitaip grįžtama prie 2-ojo žingsnio.
5. Ieškomas laisvas „langas“ mokytojui, jei randamas toks „langas“, ieškoma mokytojo pamoka iš visų dienų, kuri gali būti įdėta į tuščią „langą“ (tikrinama, ar įdėjus pamoką nebus pažeisti fiziniai apribojimai).
6. Pamoka įdedama į tuščią „langą“.
7. Grįžtama į 2-ąjį žingsnį, kai išnagrinėjami visi tušti mokytojo „langai“.
8. Lyginamas einamojo tvarkaraščio baudos taškai su geriausio tvarkaraščio baudos taškais kai mokytojų sąrašas peržiūrėtas. Jei einamasis tvarkaraštis turi daugiau baudos taškų, grįžtama prie 1-ojo žingsnio, kitaip sukeičiami geriausias tvarkaraštis su einamuoju ir įsidėmimi jo baudos taškai. Grįžtama prie 1-ojo žingsnio.

Pamokų perstatinėjimo algoritmas gerina pradinį tvarkaraštį pasitelkdamas atsitiktinumo faktorių, tik tam, kad nereikėtų nagrinėti visų tvarkaraščio variantų.

5.3 Optimizavimas naudojant SA algoritmą

1. Iteracijos skaitliukas padidinamas vienetu.
2. Pasirenkamas sekantis mokytojas iš mokytojų sąrašo.
3. Pasirenkamas atsitiktinis dydis kuris ∈ [0..1].
4. Jei atsitiktinis dydis yra mažesnis už vartotojo įvestą tikimybę, tada pasirinkto mokytojo tvarkaraštis nagrinėjamas (gerinamas), kitaip grįžtama prie 2-ojo žingsnio.
5. Ieškomas laisvas „langas“ mokytojui, jei randamas toks „langas“, ieškoma mokytojo pamoka iš visų dienų, kuri gali būti įdėta į „langą“ (tikrinama, ar įdėjus pamoką nebus pažeisti fiziniai apribojimai).
6. Pamoka įdedama į „langą“.
7. Grįžtama į 2-ąjį žingsnį, kai išnagrinėjami visi mokytojo „langai“.
8. Kai baigiasi mokytojų sąrašas, yra sugeneruojamas atsitiktinis skaičius, kuris ∈ [0..1], ir jis lyginamas *simulated annealing* „atšaldymo“ kintamuoju: atsitiktinis skaičius yra didesnis, tada grįžtama prie 1-ojo žingsnio, kitaip sukeičiami geriausias tvarkaraštis su einamuoju ir įsidėmimi jo baudos taškai, ir grįžtama prie 1-ojo žingsnio.

Simulated annealing sąlyga padeda spręsti tvarkaraščio visiško perrinkimo problemą, kai besąlygiškai yra priimamas geresnis tvarkaraštis (kuris turi mažiausiai baudos taškų už geriausiąjį) ir priimamas blogesnis tvarkaraštis (kuris turi daugiau baudos taškų už geriausiąjį) su tam tikra tikimybe r, kur :

Jeigu $h_{i+1} > 0$, *tai*

$$r_{i+1} = e^{\frac{-h_{i+1}}{x_1 / \ln(1+x_2 N)}} \quad (10)$$

kitaip $r_{i+1} = 1$

Čia :

h_{i+1} = *einamojo tvarkaraščio baudų skaičius – geriausio tvarkaraščio baudų skaičius*, iš pradžių *einamojo tvarkaraščio baudų skaičius = geriausio tvarkaraščio baudų skaičius = pradinio tvarkaraščio baudų skaičius*.

r_{i+1} – *simulated annealing* „atšaldymo“ kintamasis;

x_1 – „sistemos temperatūra“, kurią turi įvesti vartotojas;

x_2 – „atšaldymo greitis“, kurį turi įvesti vartotojas;

N – iteracijų skaičius, kurį vartotojas pasirenka;

Taigi, pasinaudojus *simulated annealing* sąlyga su tam tikra tikimybe, mes galime geriausiam tvarkaraščiui priskirti blogesnį variantą. Šitoks sprendimo būdas vadinamas geriausio varianto ieškojimas per blogesnius. Toks algoritmas įgalina mus surasti mažiausią sprendinį, bet tik tam tikroje apibrėžtoje reikšmių srityje, nes parametrai nesikeičia ir tai negarantuoja, kad bus minimumas visoje reikšmių srityje kai nedaug iteracijų.

5.4 Optimizavimas naudojant Bayeso sprendinių teoriją

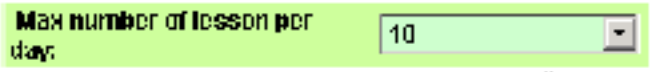
1. Naudojantis Bayeso sprendinių teorija prognozuojami parametrai x_1, x_2 ir *probability* (iš pradžių imami vartotojo įvesti parametrai) ir padidinamas prognozavimo skaitliukas.
2. Iteracijos skaitliukas padidinamas vienetu.
3. Pasirenkamas sekantis mokytojas iš mokytojų sąrašo.
4. Pasirenkamas atsitiktinis dydis kuris $\in [0..1]$.
5. Jei atsitiktinis dydis yra mažesnis už *probability* (parenka Bayeso sprendinių teorijos algoritmas), tada pasirinkto mokytojo tvarkaraštis nagrinėjamas (gerinamas), kitaip grįžtama prie 3-iojo žingsnio.
6. Ieškomas laisvas „langas“ mokytojui, jei randamas toks „langas“, ieškoma mokytojo pamoka iš visų dienų, kuri gali būti įdėta į tuščią „langą“ (tikrinama, ar įdėjus pamoką, nebus pažeisti fiziniai apribojimai).
7. Pamoka įdedama į tuščią „langą“.
8. Grįžtama į 3-iąjį žingsnį, kai išnagrinėjami visi mokytojo „langai“.
9. Kai baigiasi mokytojų sąrašas, apskaičiuojamas *simulated annealing* pagal x_1 ir x_2 „atšaldymo“ kintamasis *cnt_r*. Sugeneruojamas atsitiktinis skaičius $x \in [0..1]$.

10. Jei *x* mažesnis už *cnt_r*, sukeičiami geriausias tvarkaraštis su einamuoju ir įsidėmimi jo baudos taškai.
11. Jei iteracijos skaitliukas mažesnis už įvestą iteracijų skaičių, grįžtama prie antrojo žingsnio.
12. Kai iteracijos skaitliukas lygus įvestajam iteracijų skaičiui, einama prie pirmo žingsnio tam, kad naudojantis Bayeso sprendinių teorija, būtų galima prognozuoti SA parametrus.
13. Jei prognozavimo skaitliukas lygus iteracijų skaičiui, nutraukiamas algoritmas.

6. Tvarkaraščio optimizavimo eksperimentai

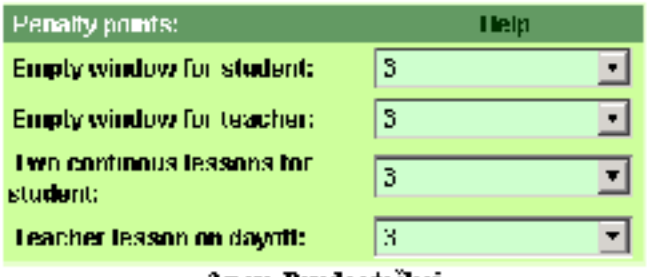
6.1 Programos parametrai

Pasirenkamas maksimalus pamokų skaičius per dieną :



1 pav. Maksimalus pamokų per dieną skaičius

Pasirenkami baudos taškai už kiekvieną normatyvinį pažeidimą :



2 pav. Baudos taškai

6.2 Eksperimentai su pastoviu pradiniu tvarkraščiu

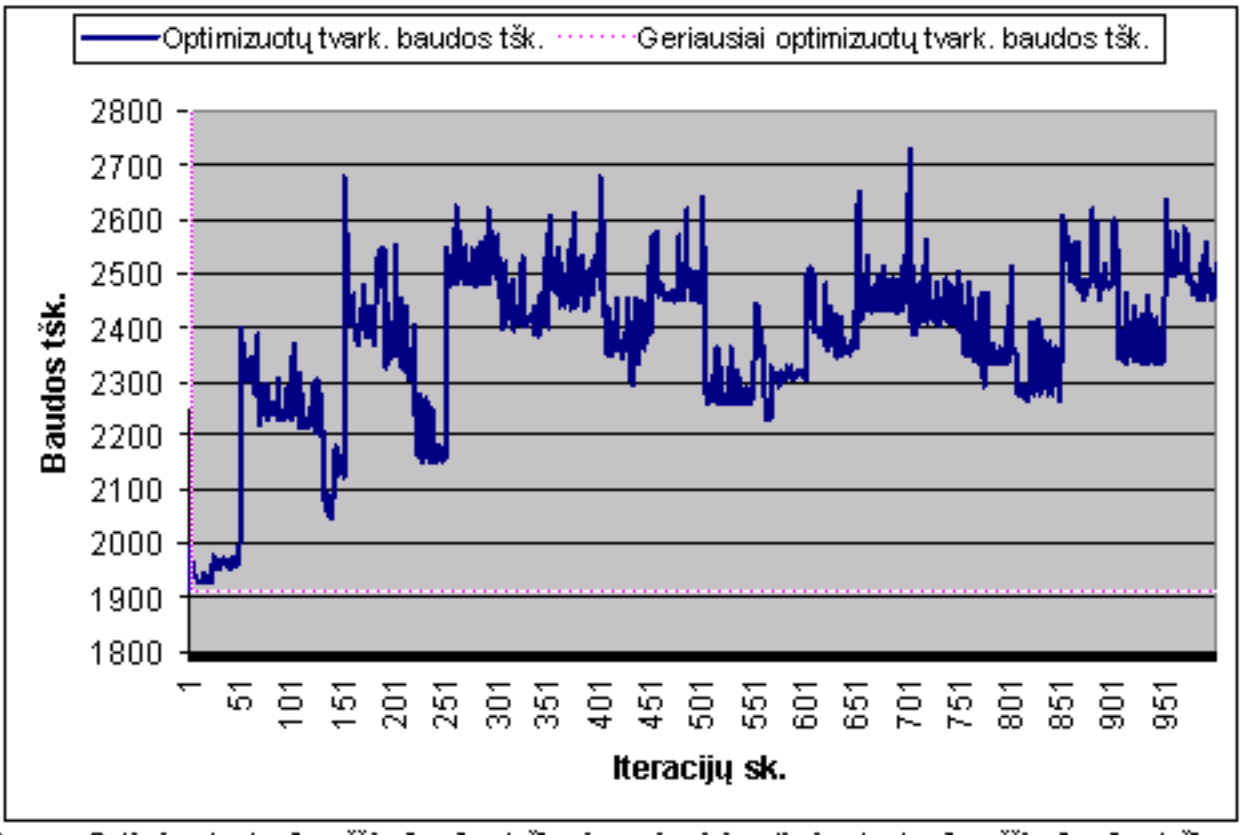
Eksperimentai buvo atliekami su pradiniu tvarkaraščiu, kuris buvo sudaromas naudojant paprastą-pamokų sudėjimo algoritmą, ir turėjo visą laiką 3105 baudos taškus. Tyrimo tikslas yra išsiaiškinti, kuris algoritmas susitvarko geriausiai su užduotimi, kai pradinės sąlygos vienodos.

6.2.1 Eksperimentai su pamokų perstatinėjimo algoritmu

Eksperimentų buvo atlikta apie 200, su įvairiomis iteracijų ir *probability* tikimybių, kurių intervalas yra nuo 0 iki 1, reikšmėmis. Didinant *probability* didėja tikimybė, kad bus nagrinėjama daugiau mokytojų tvarkaraščių kiekvienos iteracijos metu. Pats geriausias šiuo metodu surastas tvarkaraštis turėjo 1548 baudos taškus, o pats prasčiausias turėjo 2148 taškus. Vidurkis buvo 1871 baudos taškai.

Toliau pateikiami grafikai atskirų optimizavimo variantų, kur galima pamatyti optimizuotų baudos taškų funkcijos ir geriausiai optimizuotų baudos taškų funkcijų elgsenos, sistemos temperatūrą-triukšmingumą, konvergavimo greitį.

Pasirinktas iteracijų skaičius 50, tikimybė 0.5, atlikta 20 optimizavimo bandymų.



3 pav. Optimizuotų tvarkaraščių baudos taškų ir geriausiai optimizuotų tvarkaraščių baudos taškų funkcijų grafikas

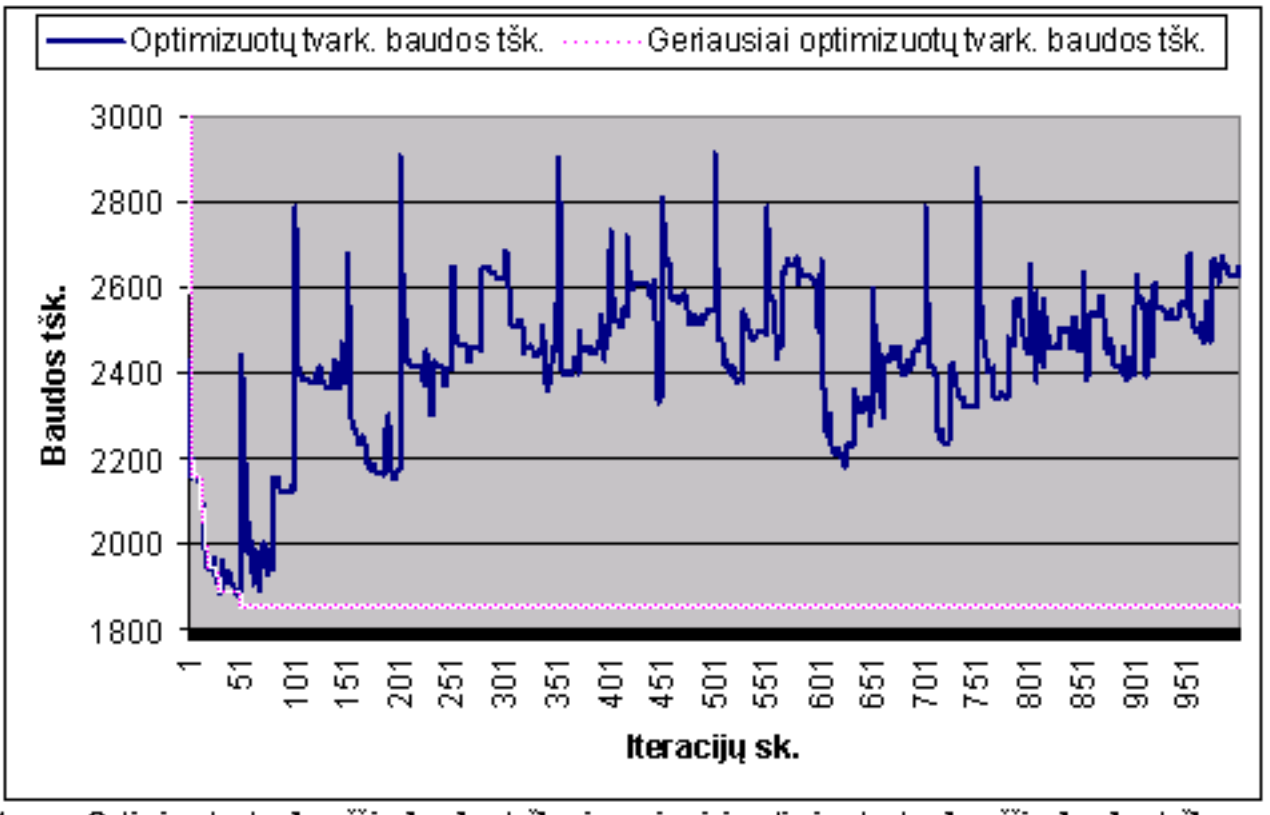
1 lentelė. Bandymų ir baudos taškų lentelė

Bandymas	Geriausio varianto baudos taškai
1	1911
2	1911
3	1911
4	1911
5	1911
6	1911
7	1911
8	1911
9	1911
10	1911
11	1911
12	1911
13	1911
14	1911
15	1911
16	1911
17	1911
18	1911
19	1911
20	1911

Kaip matome, algoritmas tik pradėdamas perstatinėti pamokas rado geriausią variantą, ir visi kiti bandymai nepagerino tvarkaraščio. Pasirinkus tokius parametrus

algoritmas su 0.5 tikimybe nagrinėja ir gerina pasirinkto mokytojo tvarkaraštį, t. y. maždaug pusės mokytojų tvarkaraščiai bus nagrinėjami.

Pasirinktas iteracijų skaičius 50, tikimybė 0.9, atlikta 20 bandymai.



4 pav. Optimizuotų tvarkaraščių baudos taškų ir geriausiai optimizuotų tvarkaraščių baudos taškų funkcijų grafikas

2 lentelė. Bandymų ir baudos taškų lentelė

Bandymas	Geriausio varianto baudos taškai
1	1875
2	1854
3	1854
4	1854
5	1854
6	1854
7	1854
8	1854
9	1854
10	1854
11	1854
12	1854
13	1854
14	1854
15	1854
16	1854
17	1854
18	1854
19	1854
20	1854

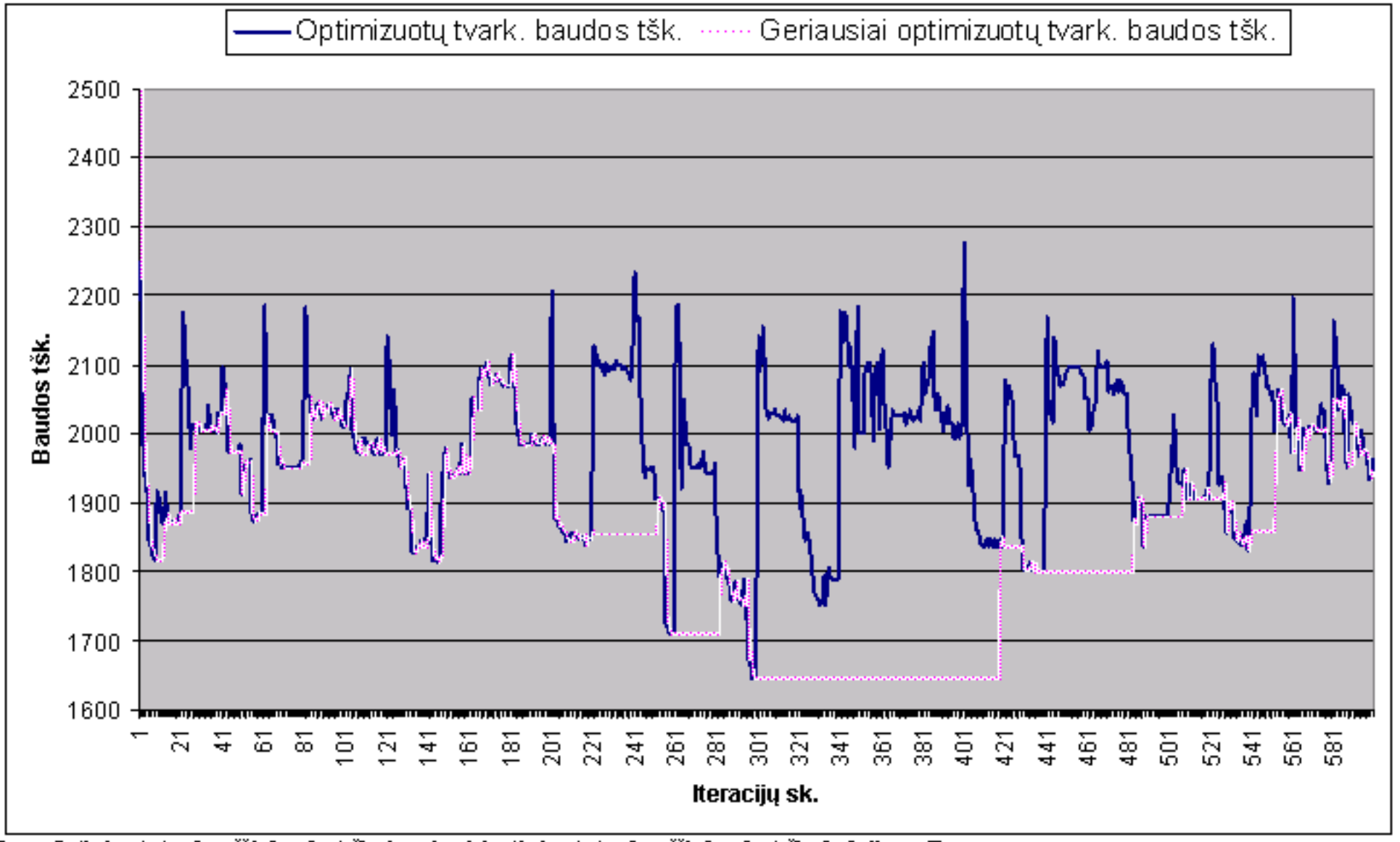
Kaip matome, pamokų perstatinėjimo algoritmas konvergavo jau per antrąjį bandymą. Buvo pasirinkta 0.9 tikimybė tam, kad 90% mokytojų būtų nagrinėjama. Šiuo eksperimentu geriausiai optimizuotų tvarkaraščių baudos taškų funkcija konvergavo jau antruoju bandymu. Optimizuotų tvarkaraščių baudos taškų funkcijos yra triukšminga, t. y. po kiekvienos iteracijos šis algoritmas tai pagerina, tai pablogina situaciją, bet per tolesnius bandymus nekonverguoja.

6.2.2 Eksperimentai su pamokų perstatinėjimo algoritmu pritaikius SA

Eksperimentų buvo atlikta apie 200, su įvairiomis iteracijų, X_1, X_2 ir *probability* tikimybės, kurios intervalas yra nuo 0 iki 1, reikšmėmis. Didinant X_1 SA temperatūra didėja, todėl didėja tikimybė priimti blogesnį variantą, t. y. sistema darosi nenuspėjama. Taip pat X_2 SA atšaldymo greitis turi didelę įtaką. Jį didinant „atšaldomas“ SA procesas ir sistema pasidaro labiau nuspėjama. Pats geriausias šiuo metodu surastas tvarkaraštis turėjo 1569 baudos taškus, o pats prasčiausias turėjo 2160 taškus. Vidurkis buvo 1804 baudos taškai.

Toliau pateikiami grafikai atskirų optimizavimo variantų, kur galima pamatyti optimizuotų baudos taškų funkcijos ir geriausiai optimizuotų baudos taškų funkcijų elgesys, sistemos temperatūrą-triukšmingumą, konvergavimo greitį.

Pasirinktas iteracijų skaičius 20, tikimybė 0.5, X_1 = 140, X_2 = 0.5.



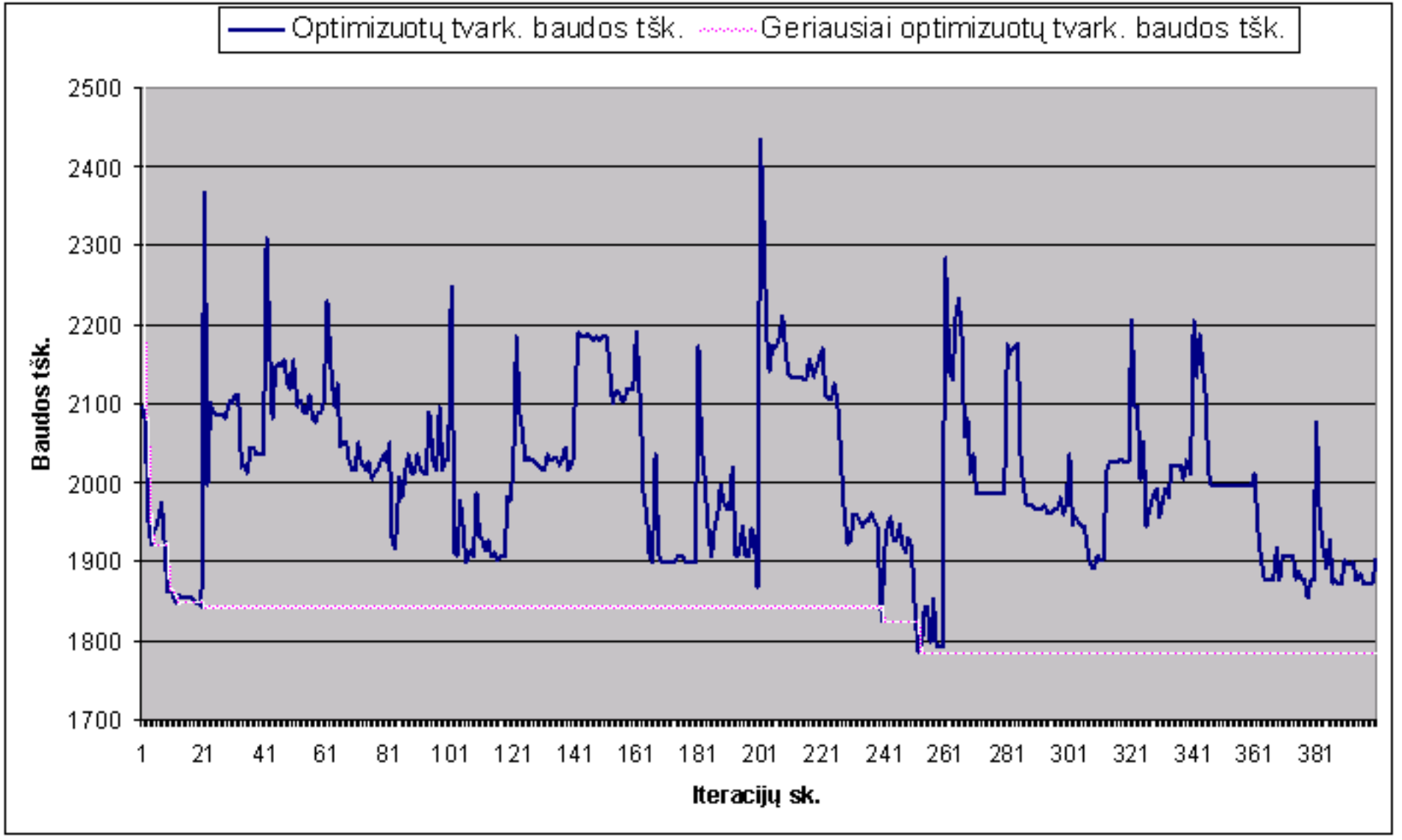
5 pav. Optimizuotų tvarkaraščių baudos taškų ir geriausiai optimizuotų tvarkaraščių baudos taškų funkcijų grafikas

3 lentelė. Bandymų ir baudos taškų lentelė

Bandymas	Geriausio varianto baudos taškai
1	1887
2	2031
3	1884
4	1956
5	2010
6	1971
7	1848
8	1941
9	2070
10	1983
11	1854
12	1854
13	1710
14	1710
15	1647
16	1647
17	1647
18	1647
19	1647
20	1647
21	1836
22	1800
23	1800
24	1800
25	1881
26	1905
27	1860
28	1974
29	2046
30	1944

Buvo pasirinkti labai didelė sistemos „temperatūra“ X_1 ir labai mažas „atšaldymo“ greitis X_2. Surasti variantai tai gerėjo, tai blogėjo ir geriausią variantą algoritmas surado per 15-20 bandymus. Kadangi sistemos „temperatūra“ aukšta, o „atšaldymo“ temperatūra maža, sistema „nestabili“, ir algoritmas dažnai pereina prie blogesnio varianto. Tokiu elgesiu algoritmas ieško geresnio varianto per blogesnius ir bando surasti globaliai geriausią variantą.

Pasirinktas iteracijų skaičius 20, tikimybė 0.5, X_1 = 0.5, X_2 = 0.5.



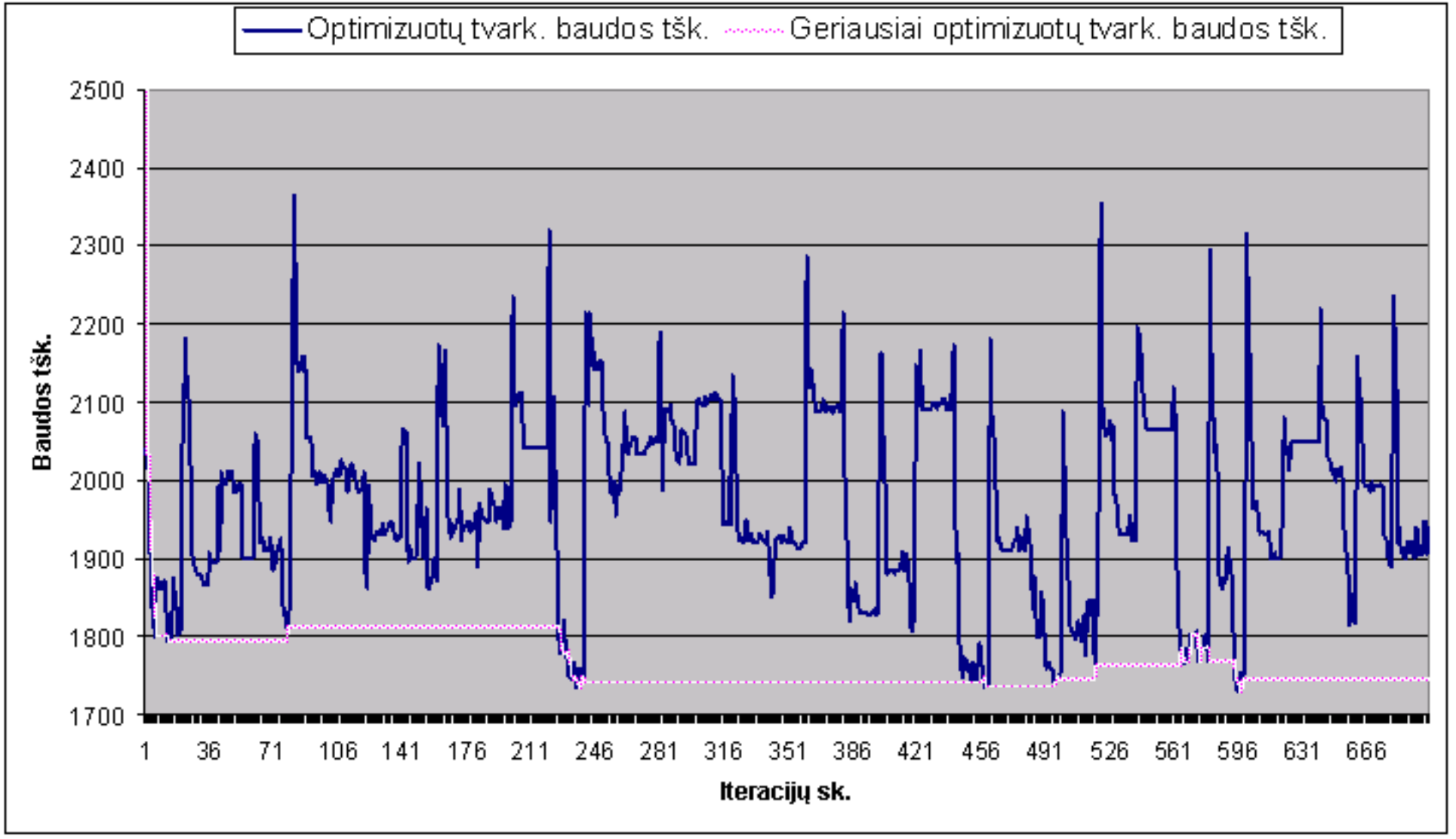
6 pav. Optimizuotų tvarkaraščių baudos taškų ir geriausiai optimizuotų tvarkaraščių baudos taškų funkcijų grafikas

4 lentelė. Bandymų ir baudos taškų lentelė

Bandymai	Geriausio varianto baudos taškai
1	1842
2	1842
3	1842
4	1842
5	1842
6	1842
7	1842
8	1842
9	1842
10	1842
11	1842
12	1842
13	1785
14	1785
15	1785
16	1785
17	1785
18	1785
19	1785
20	1785

Buvo pasirinkti maža sistemos „temperatūra“ ir mažas „atšaldymo“ greitis. Grafikas parodo, kad geriausių optimizuotų tvarkaraščių baudos taškų kreivė yra lygi ir labai greitai nusistovi. Algoritmas geriausią variantą surado tryliktu bandymu. Sistema elgiasi tolygiai ir nerizikuoja ieškoti geresnio varianto, eidama per blogesnius tvarkaraščius. Bet deja, rastas sprendinys yra lokalus minimumas ir tikrai nėra geriausias.

Pasirinktas iteracijų skaičius 20, tikimybė 0.5, X_1 = 120, X_2 = 500.



7 pav. Optimizuotų tvarkaraščių baudos taškų ir geriausiai optimizuotų tvarkaraščių baudos taškų funkcijų grafikas

5 lentelė. Bandymų ir baudos taškų lentelė

Bandymai	Geriausio varianto baudos taškai
1	1794
2	1794
3	1794
4	1812
5	1812
6	1812
7	1812
8	1812
9	1812
10	1812
11	1812
12	1743
13	1743
14	1743
15	1743
16	1743
17	1743
18	1743
19	1743
20	1743
21	1743
22	1743
23	1737
24	1737
25	1746
26	1746
27	1746
28	1746
29	1770
30	1746
31	1746
32	1746
33	1746
34	1746
35	1746

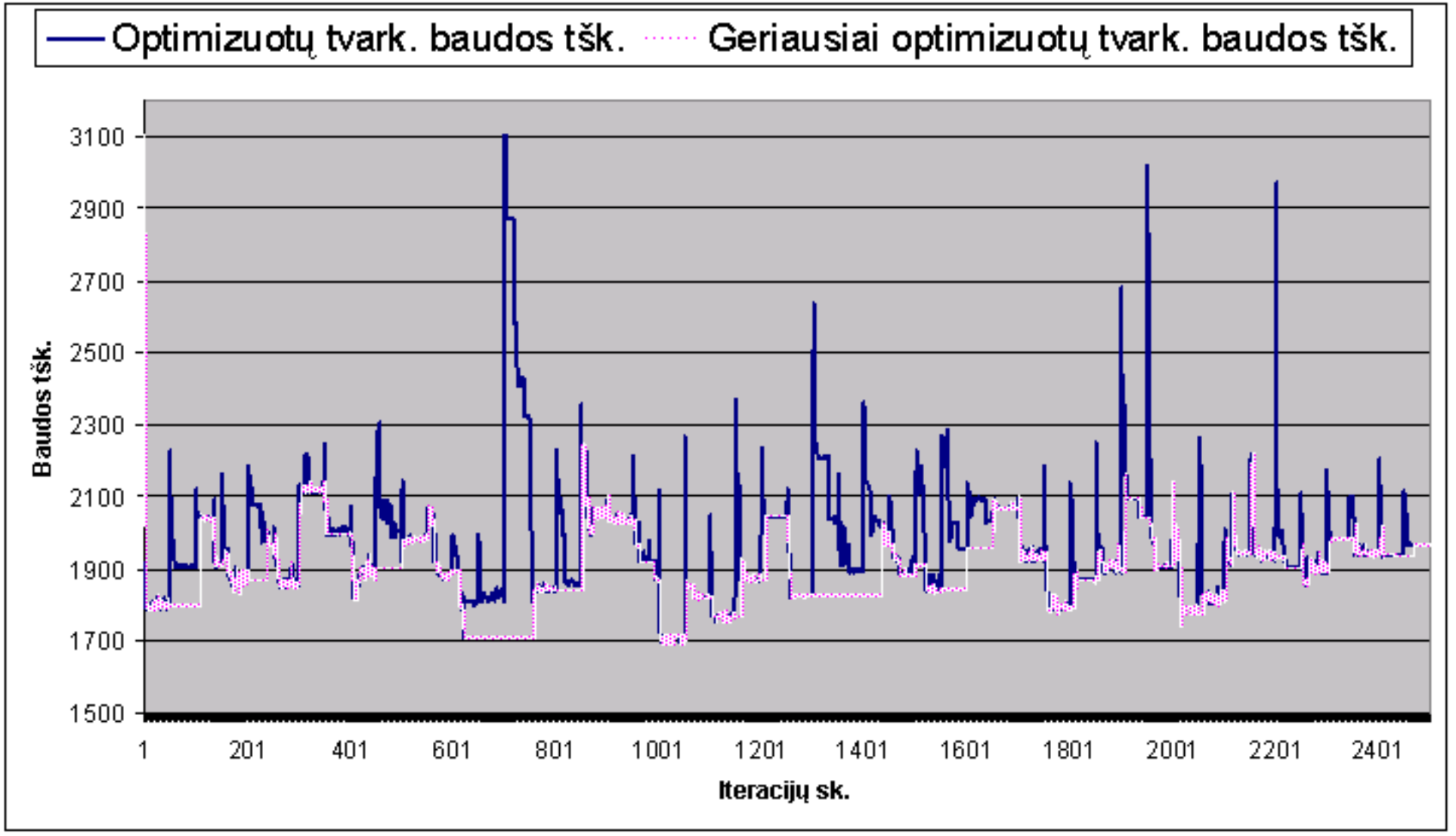
Šiuo bandymu pasirinktas ne tik didelė sistemos „temperatūra“, bet ir pakankamai didelis „atšaldymo“ greitis. Iš grafiko matyti, kad geriausių tvarkaraščių baudos taškų funkcija yra nei labai triukšminga, nei labai rami. Minimumas buvo surastas septintuoju bandymu. Sistema tolygiai artėjo prie minimumo, bet kelis kartus perėjo prie blogesnio tvarkaraščio su intencija rasti geresnįjį, kaip ir pavyko.

6.2.3 Eksperimentai su pamokų perstatinėjimo algoritmu panaudojant *Bayes Approach*

Eksperimentų buvo atlikta apie 200, su įvairiomis iteracijų, X_1, X_2, max X_1, max X_2 ir *probability* tikimybės, kurios intervalas yra nuo 0 iki 1, reikšmėmis. Šiam algoritmui pradinės tikimybės ir SA parametrų reikšmės nėra labai svarbios, nes jis pats po to jas prognozuoja ir keičia. Svarbūs yra max X_1 ir max X_2 parametrai, kurie apibrėžia, atitinkamai, sistemos temperatūros ir atšaldymo kintamųjų kitimo intervalus. Pats geriausias šiuo metodu surastas tvarkaraštis turėjo 1482 baudos taškus, o pats prasčiausias turėjo 2019 taškus. Vidurkis buvo 1711 baudos taškai.

Toliau pateikiami grafikai atskirų optimizavimo variantų, kur galima pamatyti optimizuotų baudos taškų funkcijas ir geriausiai optimizuotų baudos taškų funkcijų elgesį, sistemos temperatūrą-triukšmingumą, konvergavimo greitį.

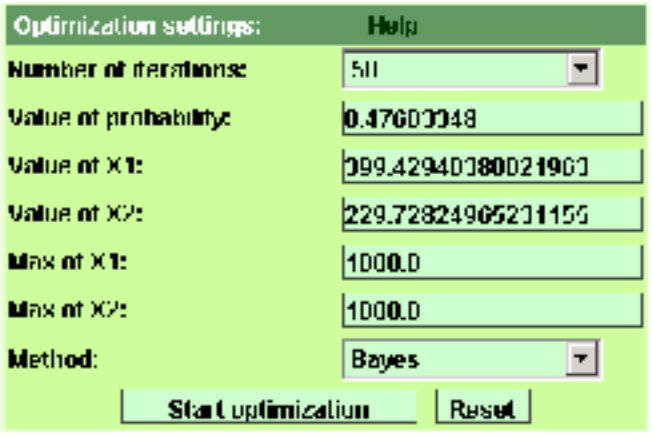
Pasirinktas iteracijų skaičius 50, sistemos pradinė „temperatūra“ X_1 = 500, sistemos „atšaldymo“ pradinis greitis X_2 = 400, pradinė tikimybė 0.5, „temperatūros“ kitimo intervalas [0 .. 1000], „atšaldymo“ greičio kitimo intervalas pasirinktas [0 .. 1000].



8 pav. Optimizuotų tvarkaraščių baudos taškų ir geriausiai optimizuotų tvarkaraščių baudos taškų funkcijų grafikas

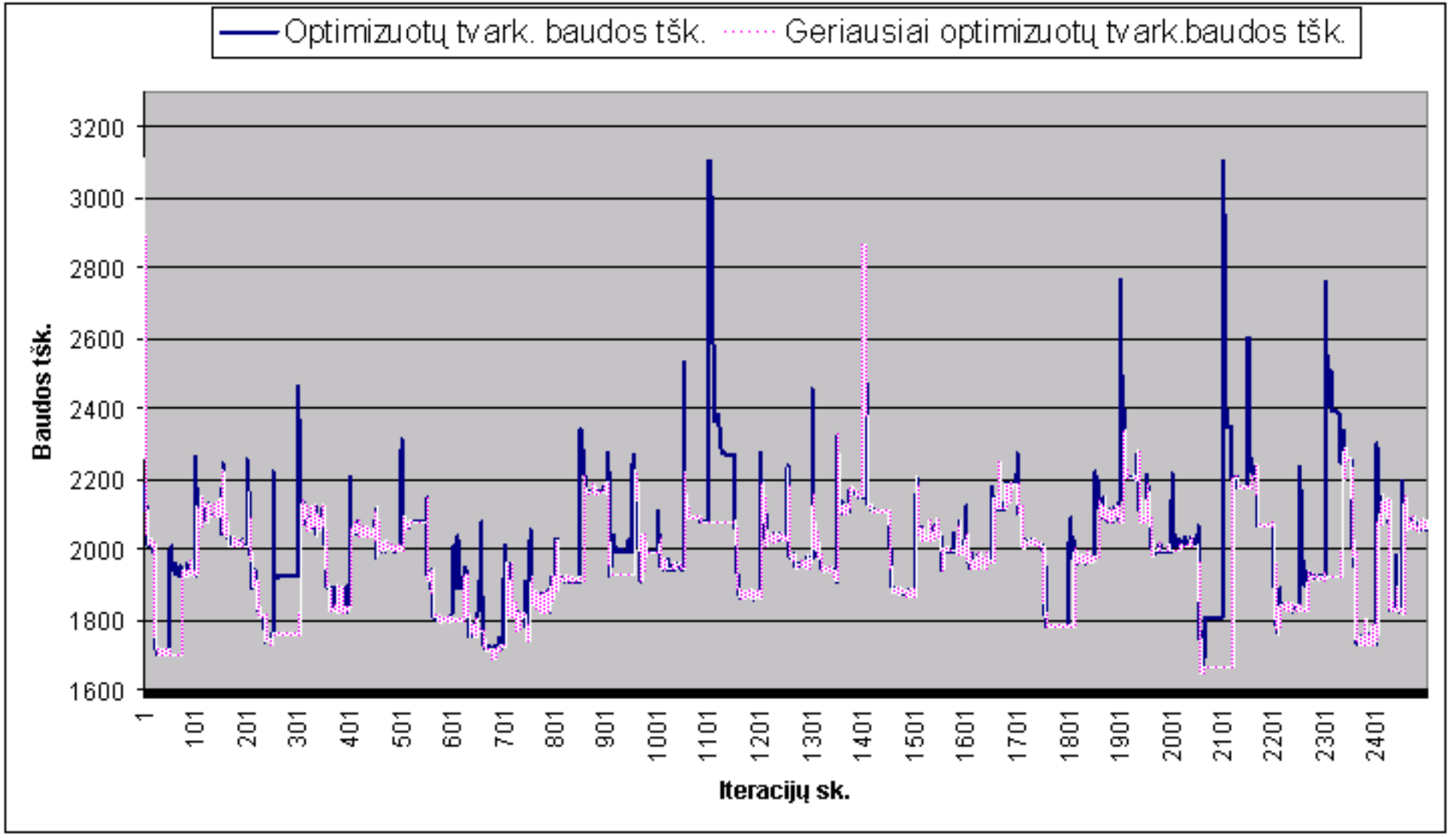
Kaip matosi iš grafiko, pasirinkus pakankamai didelius sistemos „temperatūros“ ir „atšaldymo“ kintamųjų intervalus, geriausių tvarkaraščių baudos taškų funkcija yra triukšminga ir kai kuriuose intervaluose net sutampa su optimizuotų variantų baudos taškų funkcija. Galutinis geriausias tvarkaraštis turi 1965 baudos taškus, o geriausių optimizuotų variantų baudos taškų minimumas buvo 1695 baudos taškai.

Galutiniai parametrai nustatyti tokie :



9 pav. Galutiniai BA prognozuoti parametrai

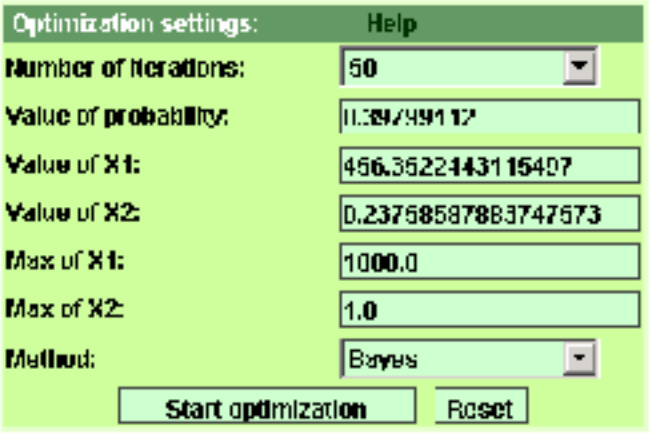
Pasirinktas iteracijų skaičius 50, sistemos pradinė „temperatūra“ X_1 = 400, sistemos „atšaldymo“ pradinis greitis X_2 = 0.5, pradinė tikimybė 0.5, „temperatūros“ kitimo intervalas [0 .. 1000], „atšaldymo“ greičio kitimo intervalas pasirinktas [0 .. 1].



10 pav. Optimizuotų tvarkaraščių baudos taškų ir geriausiai optimizuotų tvarkaraščių baudos taškų funkcijų grafikas

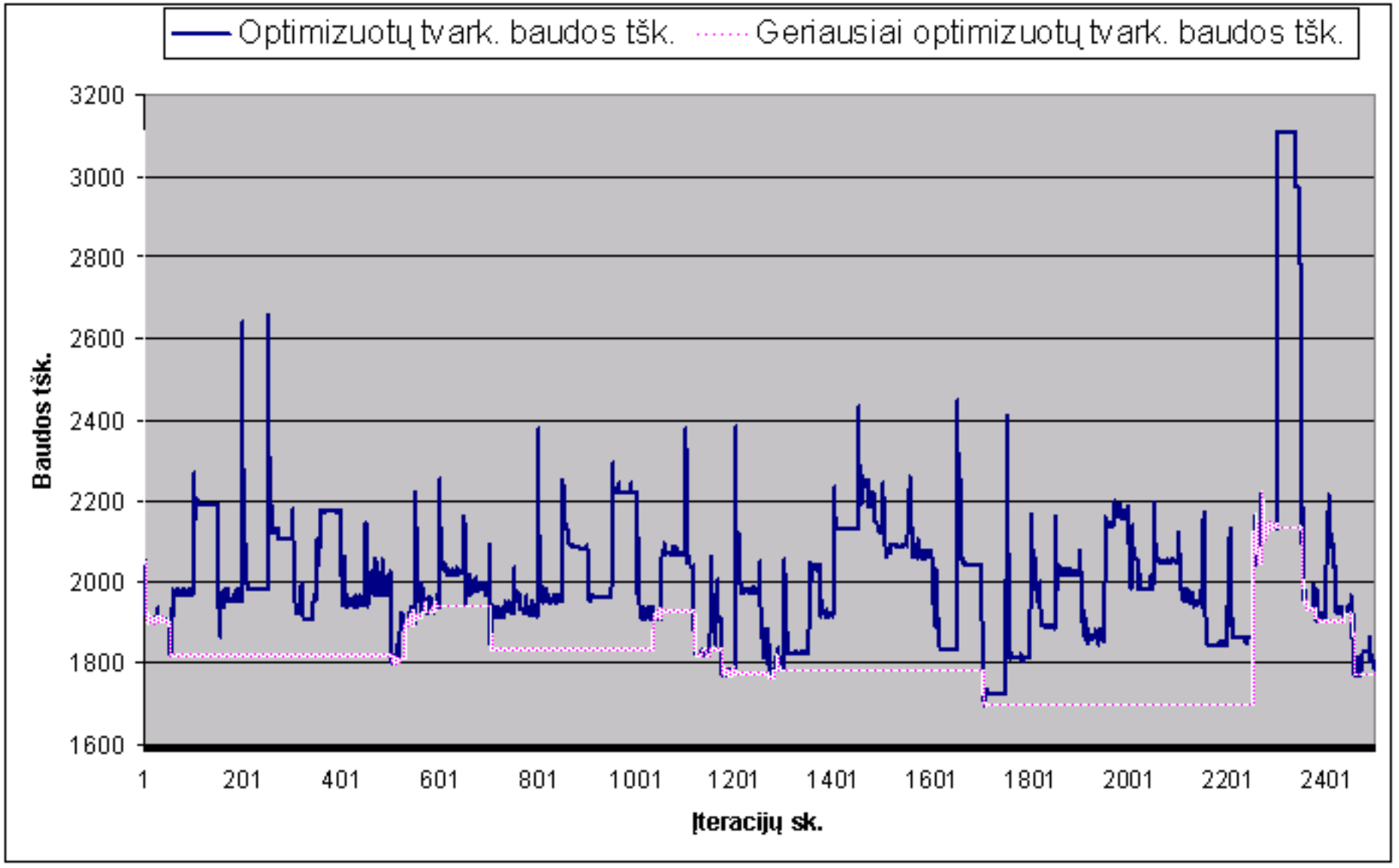
Kaip matosi iš grafiko, pasirinkus labai didelius sistemos „temperatūros“ ir labai mažą „atšaldymo“ kintamųjų intervalus, geriausių tvarkaraščių baudos taškų funkcija yra labai triukšminga ir beveik visuose intervaluose sutampa su optimizuotų variantų baudos taškų funkcija. Algoritmas, kai labai didelis skirtumas tarp einamojo tvarkaraščio ir geriausiojo tvarkaraščio baudos taškų, nepriima blogesnio varianto. Galutinis geriausias tvarkaraštis turi 2058 baudos taškus, o geriausių optimizuotų variantų baudos taškų minimumas buvo 1650 baudos taškai.

Galutiniai parametrai nustatyti tokie :



11 pav. Galutiniai BA prognozuoti parametrai

Pasirinktas iteracijų skaičius 50, sistemos pradinė „temperatūra“ X_1 = 50, sistemos „atšaldymo“ pradinis greitis X_2 = 10, pradinė tikimybė 0.5, „temperatūros“ kitimo intervalas [0 .. 110], „atšaldymo“ greičio kitimo intervalas pasirinktas [0 .. 20] .



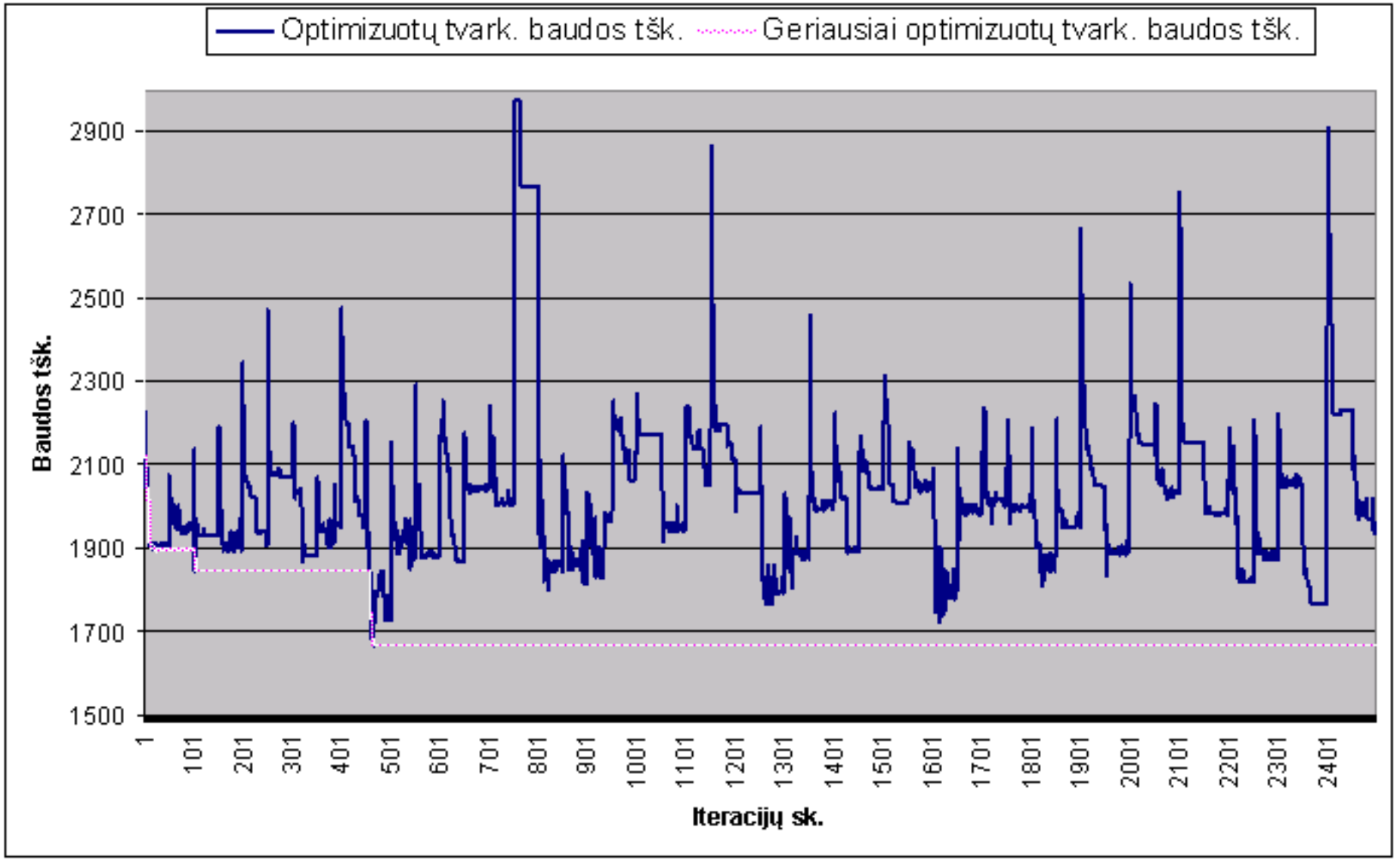
12 pav. Optimizuotų tvarkaraščių baudos taškų ir geriausiai optimizuotų tvarkaraščių baudos taškų funkcijų grafikas

Kaip matosi iš grafiko, pasirinkus nelabai didelius sistemos „temperatūros“ ir pakankamai mažą „atšaldymo“ kintamųjų intervalus, geriausių tvarkaraščių baudos taškų funkcija yra rami ir tik kartais algoritmas surizikuoja pereiti prie blogesnio tvarkaraščio. Galutinis geriausias tvarkaraštis turi 1788 baudos taškų, o geriausių optimizuotų variantų baudos taškų minimumas buvo 1698 baudos taškai.

Galutiniai parametrai nustatyti tokie :

13 pav. Galutiniai BA prognozuoti parametrai

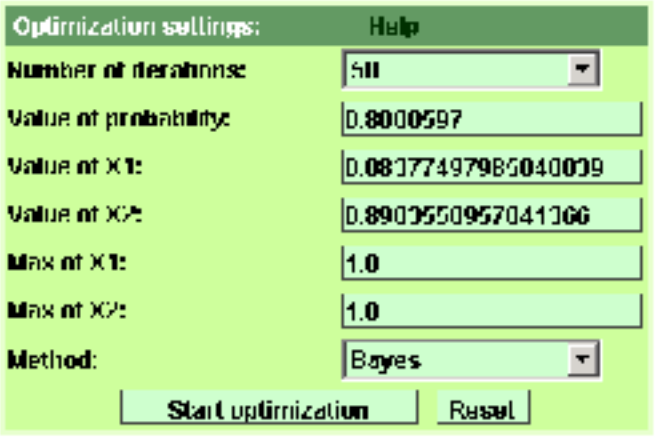
Pasirinktas iteracijų skaičius 50, sistemos pradinė „temperatūra“ X_1 = 0.5, sistemos „atšaldymo“ pradinis greitis X_2 = 0.5, pradinė tikimybė 0.5, „temperatūros“ kitimo intervalas [0 .. 1], „atšaldymo“ greičio kitimo intervalas pasirinktas [0 .. 1].



14 pav. Optimizuotų tvarkaraščių baudos taškų ir geriausiai optimizuotų tvarkaraščių baudos taškų funkcijų grafikas

Kaip matosi iš grafiko, pasirinkus pakankamai mažus sistemos „temperatūros“ ir „atšaldymo“ kintamųjų intervalus, geriausių tvarkaraščių baudos taškų funkcija yra rami ir tolygiai konverguoja prie geriausio sprendinio. Geriausias rastas tvarkaraštis turi 1668 baudos taškus.

Galutiniai parametrai nustatyti tokie :



15 pav. Galutiniai BA prognozuoti parametrai

6.3 Eksperimentai su ne pastoviu pradiniu tvarkaraščiu

Eksperimentai buvo atlikinėjami su pradiniu tvarkaraščiu, kuris buvo sudarinėjamas įvedant atsitiktinumo faktorių ir išmaišant pamokas, todėl kiekvienas pradinis variantas visą laiką turi skirtingus baudos taškus. Tyrimo tikslas yra išsiaiškinti, kaip įtakoja vėlesnį optimizavimo procesą pradinio tvarkaraščio sudarymas. Bandymai atlikinėjami su tokiais pat parametrais kaip ir 6.1 skyriuje.

Atlikti bandymai su pamokų perstatinėjimo algoritmu:

6 lentelė. Tvarkaraščių optimizavimo naudojant pamokų perstatinėjimo algoritmą lentelė

N, iteracijų Nr. skaičius		Pradinio tvarkaraščio baudų skaičius	Optimizuoto tvarkaraščio baudų skaičius	1	2
1	100	3042	1596	1446	1006,875
2	100	2625	1908	717	
3	100	3060	1821	1239	
4	100	2661	1401	1260	
5	100	2514	1506	1008	
6	100	2583	1632	951	
7	100	2067	1506	561	
8	100	2478	1605	873	
9	50	2586	1692	894	936
10	50	2334	1638	696	
11	50	2526	1677	849	
12	50	3129	2112	1017	
13	50	2778	1512	1266	
14	50	2823	1836	987	
15	50	2682	1908	774	
16	50	2979	1974	1005	
17	20	3009	1884	1125	885,75
18	20	2622	1734	888	
19	20	2616	1992	624	
20	20	2949	1887	1062	
21	20	1947	1239	708	
22	20	2214	1461	753	
23	20	2574	1635	939	
24	20	3021	2034	987	

Stulpelis ‘1’ – *pradinio tvarkaraščio baudos taškai – optimizuoto tvarkaraščio baudos taškai*

Stulpelis ‘2’ – vidutinis panaikintų baudos taškų skaičius.

Geriausias tvarkaraštis turėjo 1239 baudos taškus.

Atlikti bandymai su SA algoritmu:

7 lentelė. Tvarkaraščių optimizavimo naudojant SA algoritmą lentelė

N, iteracijų Nr. skaičius		Pradinio tvarkaraščio baudų skaičius	Optimizuoto tvarkaraščio baudų skaičius	1	2
1	100	2415	1338	1077	971,25
2	100	3006	2106	900	
3	100	3018	1863	1155	
4	100	2850	1626	1224	
5	100	2865	2163	702	
6	100	2793	1863	930	
7	100	2520	1800	720	
8	100	2889	1827	1062	
9	50	2892	1959	933	939
10	50	2319	1305	1014	
11	50	2583	1488	1095	
12	50	2127	1296	831	
13	50	2448	1524	924	
14	50	2193	1590	603	
15	50	2634	1368	1266	
16	50	2766	1920	846	
17	20	2472	1572	900	913,875
18	20	2643	1698	945	
19	20	2460	1614	846	
20	20	2970	1704	1266	
21	20	2406	1869	537	
22	20	2109	1632	477	
23	20	2460	1353	1107	
24	20	3027	1794	1233	

Stulpelis ‘1’ – *pradinio tvarkaraščio baudos taškai – optimizuoto tvarkaraščio baudos taškai*

Stulpelis ‘2’ – vidutinis panaikintų baudos taškų skaičius.

Geriausias tvarkaraštis turėjo 1296 baudos taškus.

Atlikti bandymai su BA algoritmą :

8 lentelė. Tvarkaraščių optimizavimo naudojant Bayeso algoritmą lentelė

N, iteracijų Nr. skaičius		Pradinio tvarkaraščio baudų skaičius	Optimizuoto tvarkaraščio baudų skaičius	1	2
1	100	2448	1302	1146	1303,875
2	100	2928	1494	1434	
3	100	2673	1731	942	
4	100	2676	1269	1407	
5	100	3120	1890	1230	
6	100	2946	1083	1863	
7	100	2934	1749	1185	
8	100	2310	1086	1224	
9	50	2691	1542	1149	1169,25
10	50	2931	1698	1233	
11	50	2799	1491	1308	
12	50	2946	2010	936	
13	50	2730	1665	1065	
14	50	2439	1461	978	
15	50	3258	1653	1605	
16	50	2727	1647	1080	
17	20	2043	1320	723	1086
18	20	2706	1467	1239	
19	20	2457	1626	831	
20	20	2436	1365	1071	
21	20	2991	1854	1137	
22	20	3147	1710	1437	
23	20	2607	1518	1089	
24	20	3057	1896	1161	

Stulpelis ‘1’ – *pradinio tvarkaraščio baudos taškai – optimizuoto tvarkaraščio baudos taškai*

Stulpelis ‘2’ – vidutinis panaikintų baudos taškų skaičius.

Geriausias tvarkaraštis turėjo 1083 baudos taškus.

7. Išvados

1. Ištirta *simulated annealing* ir *Bayesian Approach* metodų įtaka paprastiems modelių paieškos algoritmams.
2. Pastebėtas pradinio tvarkaraščio sudarymo poveikis tolesniam optimizavimo procesui. Sudarinėjant pradinį variantą, naudojant stochastinį algoritmą, buvo pasiekti geresni optimizavimo rezultatai, nei gerinant pastovų pirminį tvarkaraštį, kurį sudarinėjo algoritmas be jokio atsitiktinumo faktoriaus.
3. Optimizuojant tvarkaraščius pastebėta stochastinio algoritmo esmė: prie tų pačių parametrų gaunamos skirtingos funkcijos reikšmės. Tai įtakoja atsitiktinumo faktoriaus taikymas šiuose algoritmuose. Pasitvirtino, kad kuo dažniau naudojamas atsitiktinumo faktorius, tuo geresnis rezultatas gaunamas.
4. Išanalizuota sprendinio paieškos modelių erdvėje problema, kai funkcija visoje reikšmių srityje turi daug lokalių minimumų arba maksimumų, ir įvertintas *simulated annealing* požiūris šiai problemai spręsti.
5. Ištirta didelė *simulated annealing* parametrų reikšmė pačiam modelių paieškos procesui : norint kad būtų vykdoma geriausio sprendinio paieška per blogesnius (SA idėja) variantus, reikia aukštos temperatūros ir mažo atšaldymo greičio. Atšaldymo greitis priklauso nuo iteracijų skaičiaus *N*, taip suformuojami tam tikri temperatūros regionai.
6. Tiriant *simulated annealing* ir pamokų perstatinėjimo algoritmus, pastebėta, kad pamokų perstatinėjimo algoritmas gan dažnai surasdavo geresnį variantą nei algoritmas su SA. Tai rodo, kad SA veikia neefektyviai kai parametrai yra paduodami neoptimalūs. Taip pat, norint surasti gerą variantą naudojant SA, reikia daug iteracijų ir daug laiko.
7. Pastebėta, kad algoritmas kai *Bayesian Approach* prognozuoja *simulated annealing* parametrus veikia geriausiai abiem pradinio tvarkaraščio sudarymo atvejais. Taigi parametrų parinkimas, išvengiant žmogiškojo faktoriaus, stochastiniam algoritmui daro didelę įtaką metodo efektyvumui. Taigi, šis algoritmas konverguoja prie pakankamai gero varianto prie mažo iteracijų skaičiaus.
8. Visi trys algoritmai konverguoja labai greitai, bet jie tikrai nepriartėja prie globaliai geriausio varianto. Pamokų perstatinėjimo algoritmas ir nepriartės prie jo, nes suradęs lokalų minimumą ten ir sustos. SA ir naudojantis BA algoritmai turi teorinę galimybę, kai iteracijų skaičius didelis ir *simulated annealing*

temperatūra aukšta, o atšaldymo greitis mažas. Didesnis iteracijų skaičius įtakoja ilgesnį laiko tarpą atlikti iteracijoms. Algoritmas naudojantis BA visiškai pasiteisina: duoda gerus rezultatus esant mažam iteracijų skaičiui.

9. Paduodami pradiniai parametrai, algoritmo su BA proceso neįtakoja smarkiai, nes BA juos pats pakeičia kitais parametrais, jo požiūriu, geresniais. Max X_1 ir max X_2 kintamieji apibrėžia intervalus, iš kurių BA gali prognozuoti SA parametrus. Kuo didesnis bus paduodamas max X_1, tuo SA temperatūra gali būti didesnė, kuo didesnis bus max X_2, tuo SA atšaldymo greitis gali būti didesnis.

8. Literatūra

- Mockus J. A set of examples pf global and discrete optimization 2. Home work for graduate students.
- Mockus J. Bayesian Approach to Optimization of Heuristic Parameters.
- Davidson I. Clustering using the Minimum Message Length Criterion and Simulated Annealing
- Jacobson S. H., Yucesan E. Performance measures for generalised hill climbing algorithms. JOURNAL OF GLOBAL OPTIMIZATION. 173-190, 2004.
- Hajek B. Cooling Schedules for optimal annealing. MATHEMATICS OF OPERATIONS RESEARCH, 1988. 311-329 psl.
- Miki M. Hiroyasu T. Wako J. Adaptive Temperature Schedule Determined by Genetic Algorithm for Parallel Simulated Annealing