KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Lina Rajeckaitė

GAMYBINIŲ TVARKARAŠČIŲ SUDARYMO UŽDAVINIŲ IR ALGORITMŲ ANALIZĖ

Magistro darbas

Vadovas
doc. dr. N. Listopadskis

KAUNAS, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. N. Listopadskis
2009 06 05

GAMYBINIŲ TVARKARAŠČIŲ SUDARYMO UŽDAVINIŲ IR ALGORITMŲ ANALIZĖ

Matematikos magistro baigiamasis darbas

Vadovas
doc. dr. N. Listopadskis
2009 06 03

Recenzentas
doc. dr. D.Rubliauskas
2009 06 01

Atliko
FMMM 7 gr. stud.
L. Rajeckaitė
2009 05 22

KAUNAS, 2009

KVALIFIKACINĖ KOMISIJA

Pirmininkas: Leonas Saulis, profesorius (VGTU)

Sekretorius: Eimutis Valakevičius, docentas (KTU)

Nariai: Algimantas Jonas Aksomaitis, profesorius (KTU)

Arūnas Barauskas, Vice-prezidentas projektams (UAB „Baltic Amadeus“)

Vytautas Janilionis, docentas (KTU)

Zenonas Navickas, profesorius (KTU)

Vidmantas Povilas Pekarskas, profesorius (KTU)

Rimantas Rudzkis, valdybos pirmininko patarėjas („DnB NORD“ bankas)

Rajeckaitė L. Gamybinių tvarkaraščių sudarymo uždavinių ir algoritmų analizė: Matematikos magistro darbas / vadovas dr. doc. N. Listopadskis; Taikomosios matematikos katedra, Fundamentaliųjų mokslų fakultetas, Kauno technologijos universitetas. – Kaunas, 2009. – 93 p.

SANTRAUKA

Šiame darbe nagrinėjamas nuoseklios gamybos tvarkaraščių sudarymo uždavinys. Tai vienas iš kombinatorinės optimizacijos uždavinių. Tokio tipo problemos iškyla gamyboje, paskirstant išteklius, logistikoje ir pan. Duotos darbų ir mašinų (aptarnaujančių įrenginių) aibės. Kiekvienas darbas susideda iš vienodo skaičiaus operacijų. Mašinos dirba su pertraukomis. Uždavinio tikslas – minimizuoti visų darbų apdorojimo pabaigos laiko momentą. Darbe pateikiama kombinatorinės optimizacijos, tvarkaraščių sudarymo uždavinių ir jiems spręsti naudojamų metodų apžvalga. Taip pat pateikiamas ir realizuotas vienas tikslusis algoritmas – šakų ir ribų (branch and bound), bei du meta-euristiniai algoritmai: modeliuojamasis atkaitinimas (Simulated Annealing) ir paieška su draudimais (Tabu Search). Atlikta šių trijų algoritmų analizė. Rezultatai parodė, jog šakų ir ribų algoritmas efektyvus tiktai sprendžiant nesudėtingus uždavinius. Tuo tarpu sudėtingesniems uždaviniams spręsti paieška su draudimais yra efektyvesnė nei modeliuojamasis atkaitinimas. Darbo pabaigoje aptariami nagrinėti algoritmai, pateikiamos rekomendacijos, kokius parametrus pasirinkti, sprendžiant uždavinius.

Rajeckaitė L. Analysis of shop scheduling problems and algorithms: Master‘s work in mathematics / supervisor dr. assoc. prof. N. Listopadskis; Department of Applied mathematics, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2009. – 93 p.

SUMMARY

The combinatorial optimization problem considered in this paper is flow shop scheduling problem arising in logistics, management, business, manufacture and etc. A set of machines and a set of jobs are given. Each job consists of a set of operations. Machines are working with unavailability intervals. The task is to minimize makespan, i.e. the overall length of the schedule. There is overview of combinatorial optimization, scheduling problems and methods used to solve them. There is also presented and realized one exact algorithm – Branch and Bound, and two meta-heuristics: Simulated Annealing (SA) and Tabu Search (TS). Analysis of these three algorithms is made. The results showed that Branch and Bound is effective solving simple problems. While solving more complicated problems Tabu Search is more effective than Simulated Annealing. At the end of this paper we tell the decision about algorithms analyzed and recommendations about choosing suitable parameters.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Tvarkaraščiai naudojami įvairiose srityse (logistikoje, vadyboje, versle, gamyboje ir kt.). Sudarytas tvarkaraštis smarkiai įtakoja produkcijos gamybą, gaunamą pelną bei verslo sėkmę, todėl labai svarbu sudaryti tokį, kuris su mažiausiomis laiko bei išteklių sąnaudomis duotų geriausius rezultatus. Tvarkaraščių sudarymo uždaviniai sprendžiami įvairiais kombinatorinio optimizavimo metodais.

Šiame darbe nagrinėjami vienas tikslusis – šakų ir ribų, bei du meta-euristiniai, modeliuojamojo atkaitinimo (Simulated Annealing) ir paieškos su draudimais (Tabu Search), metodai, sprendžiant nuoseklios gamybos (flow shop) uždavinį. Padaryta programa, realizuojanti šiuos algoritmus. Pirmoje darbo dalyje pateikiama teorinė dalis apie optimizavimą, kombinatorinį optimizavimą, tvarkaraščių sudarymo uždavinius bei jų sprendimo algoritmus, pateikiama medžiaga, susijusi su gamybos tvarkaraščių sudarymo uždaviniais, aprašomi darbe tiriami metodai. Antroje dalyje – atliktas naudojamų metodų tyrimas, pateikiami gauti rezultatai. Trečioje dalyje supažindinama su programa, pateikiama instrukcija vartotojui. Toliau pateikiama rezultatų apibendrinimas ir rekomendacijos, susijusios su metodų naudojimu.

Atliekant metodų analizę, tiriama sudaryto tvarkaraščio visų atliekamų darbų pabaigos empirinio vidurkio bei empirinės dispersijos priklausomybė nuo metodo parametrų. Pateikiamos rekomendacijos apie metodų naudojimą, parametrų pasirinkimą.

Darbas pristatytas konferencijoje „Matematika ir matematikos dėstymas – 2009“.

1. TEORINĖ DALIS

1.1 OPTIMIZAVIMO UŽDAVINIŲ KLASIFIKACIJA

Optimizacija vadinamas problemos studijavimas, kurio metu siekiama minimizuoti arba maksimizuoti tikslo funkciją *f*, sistemingai renkantis realiųjų ar sveikaskaičių kintamųjų reikšmes iš leistinosios srities *A* [7], [15], [24].

Optimizavimo uždavinys užrašomas taip:

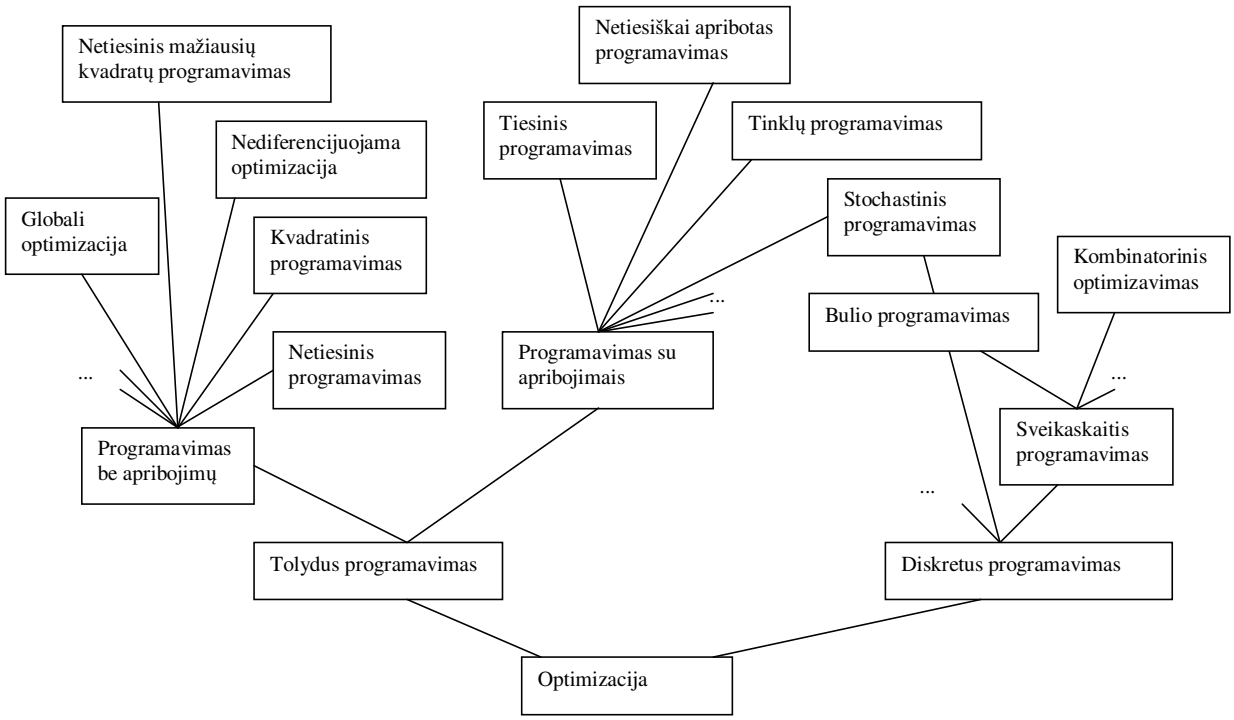
Duota: tikslo funkcija $f : A \longrightarrow \mathbf{R}$.

Rasti: elementą x_0 aibėje *A*, tokį, kad $f(x_0) \leq f(x) \ \ \forall x \in A$ (minimizacija), arba tokį, kad $f(x_0) \geq f(x) \ \ \forall x \in A$ (maksimizacija).

Aibė *A* yra Euklido erdvės $\mathbf{R}^n$ poaibis, apibrėžiamas lygybiniais ar nelygybiniais apribojimais, kuriuos tenkina aibės *A* elementai. Aibės *A* elementai vadinami sprendiniais. Sprendinys, kuris minimizuoja (maksimizuoja) tikslo funkciją, vadinamas optimaliu sprendiniu.

Kai leistinoji sritis ar tikslo funkcija nėra iškila, tuomet gali egzistuoti keli lokalieji minimumo ar maksimumo taškai. Taškas *x** vadinamas funkcijos *f* lokalaus minimumo tašku, jei egzistuoja tokia to taško aplinka *δ(x)*, kad $f(x^*) \leq f(x), \quad x \in \delta(x)$.

Priklausomai nuo to, kaip apibrėžiama tikslo funkcija bei leistinoji sritis, optimizacija yra skirstoma įvairiais aspektais. Optimizacijos klasifikacijos fragmentas pateikiamas 1.1 paveiksle.



1.1 pav. Optimizacijos klasifikacijos fragmentas

Toliau trumpai apžvelgiamos kai kurioms klasėms būdingos savybės, taikymo sritis ir sprendimo metodai.

Tiesinis programavimas (TP) – tiesinė tikslo funkcija, tiesinėmis lygybėmis bei nelygybėmis apibrėžta leistinoji sritis [2].

Matematiškai užrašoma taip [15]:

$\min\{c^T x : Ax = b, x \geq 0\}$, čia $x \in R^n$ yra kintamųjų vektorius, $c \in R^n$ koeficientų vektorius, o $A \in R^{m \times n}$ koeficientų matrica.

Tiesinis programavimas naudojamas sprendžiant verslo, ekonominius, transporto, energijos, telekomunikacijų, gamybos uždavinius, naudingas sprendžiant įvairias planavimo, tvarkaraščių sudarymo, paskirstymo problemas.

Tiesinio programavimo uždaviniams spręsti naudojami simplekso (simplex), elipsių (ellipsoid), vidinių taškų (interior point) metodai (barjerų funkcija (barrier function), kelio sekimo (path-following)) ir kt.

Kvadratinis programavimas - kvadratinė tikslo funkcija, tiesinėmis lygybėmis bei nelygybėmis apibrėžta leistinoji sritis [15], [24].

Matematiškai užrašoma:

$\min\left\{\frac{1}{2} x^T Q x + c^T x : Ax \leq b, Ex = d, x \in R^n\right\}$, kur Q $n \times n$ matavimo simetrinė matrica, $c - n \times 1$ matavimo vektorius.

Kvadratinio programavimo uždaviniams spręsti naudojami vidinių taškų (interior poin), aktyvios aibės (active set), jungtinių gradientų (conjugate gradient) ir kiti metodai.

Netiesinis programavimas – tiek tikslo funkcija, tiek leistinąją sritį apibrėžiančios lygybės ir nelygybės gali būti netiesinės [7], [15].

Matematiškai užrašoma:

$\max_{x \in X} f(x)$ arba $\min_{x \in X} f(x)$, kur $f : R^n \to R$, $X \subseteq R^n$.

Netiesinio programavimo uždaviniai sprendžiami šakų ir ribų (branch and bound), gradientinio nusileidimo (reduced-gradient), Gauso-Niutono (Gauss-Newton) ir daugeliu kitų metodų.

Stochastinis programavimas nagrinėja uždavinius, kuriuose turi būti įvertinti įvairūs neapibrėžtumai. Tokio tipo uždavinių stilius yra panašus, tikimybių skirstiniai yra žinomi arba gali būti įvertinti. Tikslas – rasti tokią strategiją, kuri būtų tinkama visiems (ar beveik visiems) duomenų rinkiniams ir maksimizuotų lūkesčių funkciją bei atsitiktinius kintamuosius. Plačiausiai taikomi ir nagrinėjami dviejų pakopų tiesinio programavimo stochastiniai uždaviniai. Pirmoje pakopoje priimamas kažkoks sprendimas, po kurio įvyksta šį sprendimą įtakojantis atsitiktinis įvykis. Tuomet antroje pakopoje priimamas toks sprendimas, kad būtų atitaisyta galimai padaryta žala.

Tokio tipo uždaviniai dažnai pasitaiko ekonomikoje, biologijoje.

Kombinatorinis optimizavimas nagrinėja uždavinius, kurių sprendiniai yra diskretūs. Leistinoji aibė yra baigtinė. Kombinatorinio optimizavimo algoritmai sprendžia uždavinius, ieškodami optimalaus sprendinio didelėje sprendinių aibėje, sumažindami jos tikrąjį dydį ir efektyviai ją tirdami [2], [15], [24].

Kombinatorinio optimizavimo uždavinys formaliai gali būti užrašomas kaip sistema (X,P,Y,f,extr), kur

X – leistinoji aibė (joje apibrėžta f ir P);

P – tinkamumo požymis;

Y – sprendinių aibė;

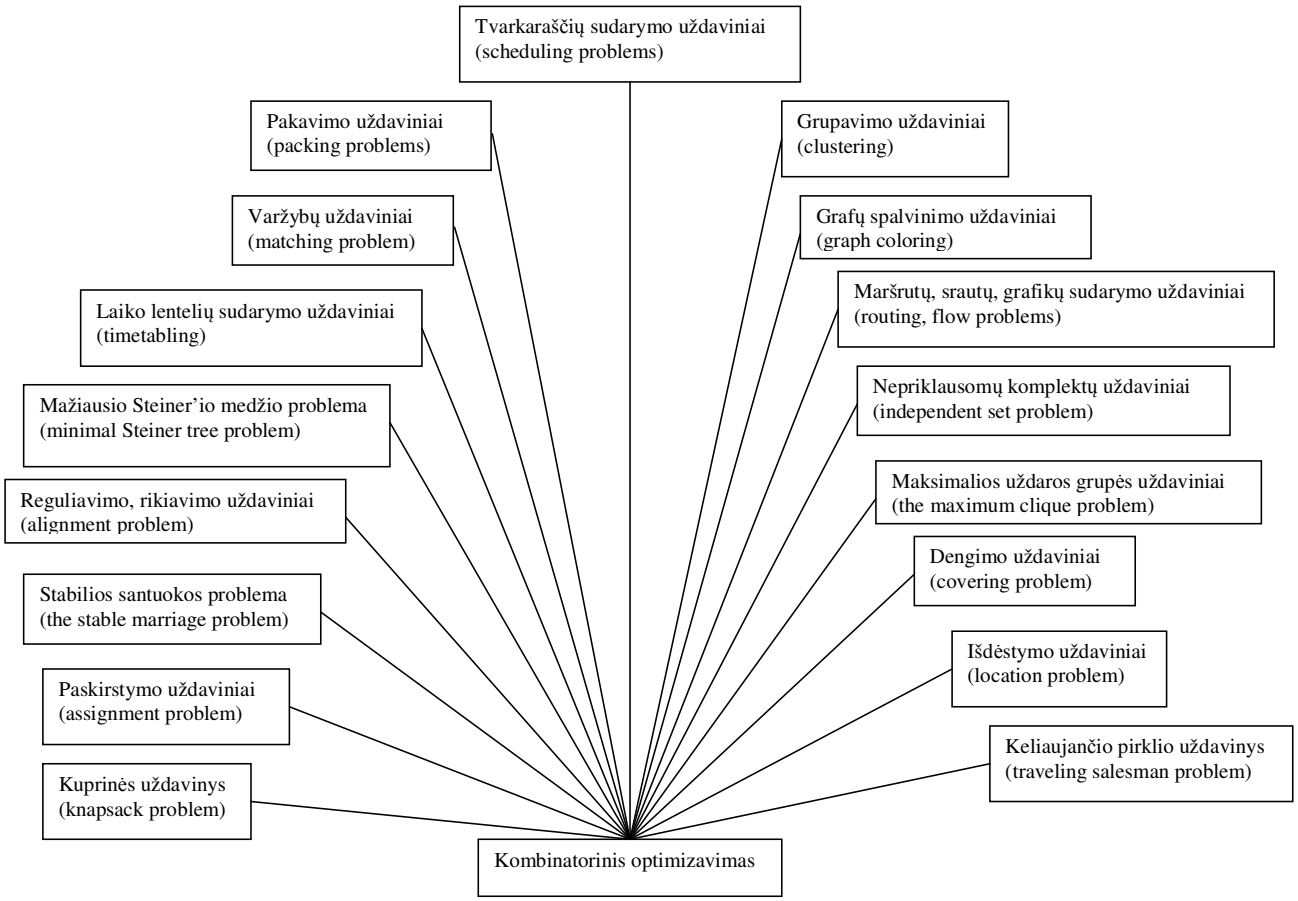
f - tikslo funkcija;

extr – ekstremumas (min arba max).

Kombinatorinio optimizavimo uždaviniams spręsti naudojami godžios atsitiktinės adaptyviosios paieškos (greedy randomized adaptive search), lokaliosios paieškos (local search), modeliuojamojo atkaitinimo (simulated annealing), kvantinio atkaitinimo (quantum annealing), paieškos su draudimais (tabu search), šakų ir ribų (branch and bound) metodai, spiečių (swarm), genetiniai (genetic) algoritmai, skruzdžių kolonijos optimizacija (ant colony optimization) ir daugelis kitų.

1.2 KOMBINATORINIO OPTIMIZAVIMO UŽDAVINIŲ KLASIFIKACIJA

Kiekvienas kombinatorinio optimizavimo uždavinys, atsižvelgiant į jo formuluotę ir pageidaujamą sprendinį, gali būti priskirtas vienai iš klasių, pateiktų 1.2 paveiksle [4].



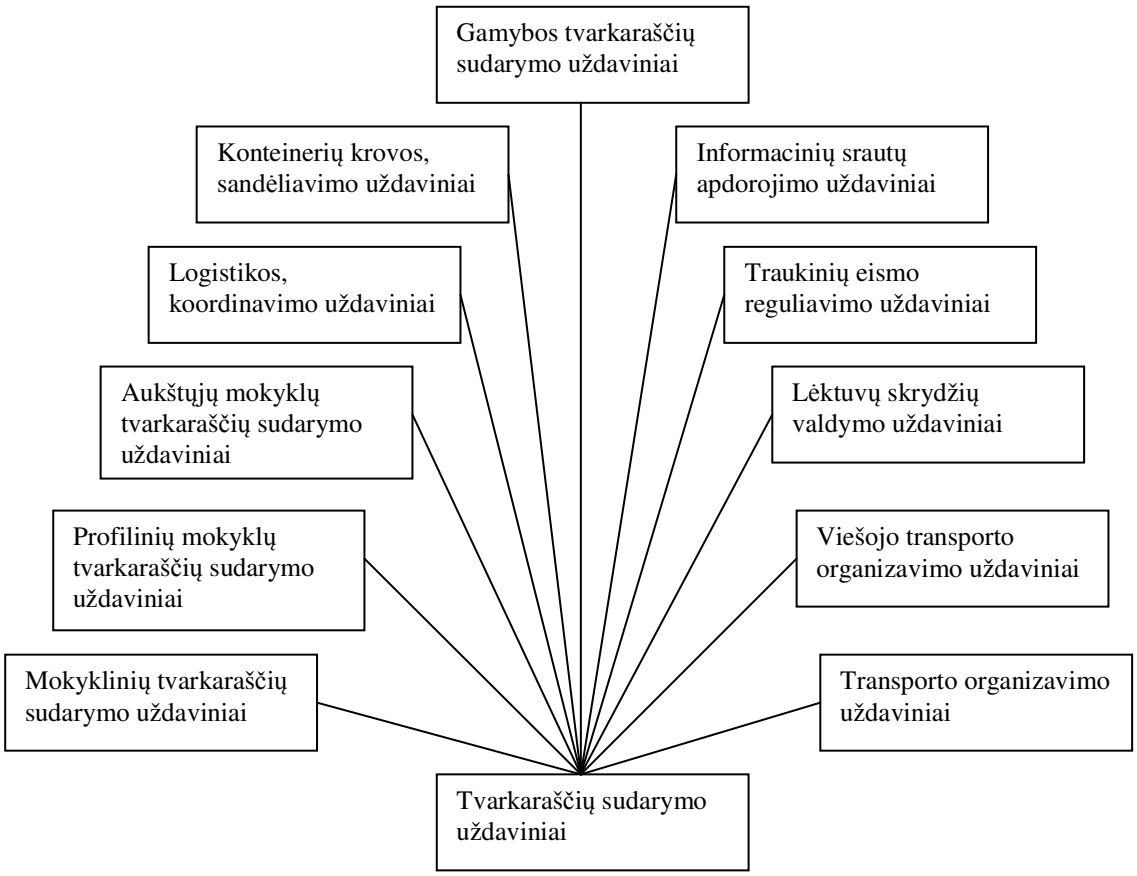
1.2 pav. Kombinatorinio optimizavimo uždavinių klasifikacija

1.3 TVARKARAŠČIŲ SUDARYMO UŽDAVINIŲ KLASIFIKACIJA

Tvarkaraščių sudarymo uždaviniai yra svarbūs gamybos, inžinerijos srityse, kur grafikas smarkiai įtakoja produkcijos eigą. Gamyboje naudojamų grafikų esmė – sumažinti laiką ir kainą, skirtą produkcijai pagaminti, įrenginiams nurodant, ką ir kada daryti bei personalą. Produkcijos grafiko tikslas – maksimizuoti veikimo efektyvumą ir sumažinti kainą.

Tvarkaraščių radimo uždaviniai iškyla įvairiose veiklos srityse: gamyboje, ryšių ir kompiuterių tinklų architektūrai parinkti bei informacijos srautams valdyti, logistikoje, vadyboje, versle, projektavime, ekonomikoje ir kitose srityse [1].

Tvarkaraščių sudarymo uždavinių klasifikacija pateikiama 1.3 paveiksle.



1.3 pav. Tvarkaraščių sudarymo uždavinių klasifikacija

Gamyboje sprendžiami uždaviniai, kai įmonei reikia optimizuoti gamybos planą, nusprendžiant kiek ir kokios produkcijos vienetų pagaminti, kad pelnas būtų maksimalus.

Telekomunikacijų tinkluose dažnai sprendžiami įvairių informacinių srautų apdorojimo uždaviniai. Tokie uždaviniai ypač svarbūs GSM tinkluose, kai tenka apdoroti įvairius duomenų paketus, analizuoti pliūpsninius srautus, kontroliuoti pranešimų srautus, kad jie patikimai patektų pas gavėjus su didžiausia sparta ir mažiausiu vėlavimu. Su panašiais uždaviniais susiduriama ir kompiuterių tinkluose, kai reikia apdoroti siunčiamų duomenų srautus.

Transporto organizavimo, logistikos, koordinavimo uždaviniai taip pat labai aktualūs. Šio tipo uždaviniams priklauso transporto priemonių parinkimo ir paskirstymo uždaviniai, kai iš įvairių galimų krovinio vežimo priemonių bei maršrutų reikia parinkti minimizuojančius vežimo išlaidas ar laiką.

Viešajam transportui mieste organizuoti reikia spręsti pakankamai sudėtingus kombinuotus uždavinius, kuriuose reikia optimizuoti transporto priemonių maršrutus, minimizuojant sąnaudas ir maksimizuojant teritorijos padengiamumą, optimizuojant transporto priemonių kiekį keleivių srautams, sudarant tinkamas laiko lenteles, tuo minimizuojant laukimo laiką.

Su panašiais tik sudėtingesniais uždaviniais susiduriama traukinių eismo reguliavimo, lėktuvų skrydžių valdymo uždaviniuose, kur ypač svarbu tikslumas ir patikimumas.

Laivų uostuose dažnai susiduriama su konteinerių krovos, sandėliavimo, jų tvarkymo, išvežimo uždaviniais.

Be tvarkaraščių sudarymo neišsiversime ir mokymo įstaigose.

Ypač sudėtingi yra aukštųjų ir profilinių mokyklų užsiėmimų tvarkaraščių sudarymo uždaviniai. Dažniausiai tokiuose uždaviniuose reikia suplanuoti ir optimizuoti savaitinius užsiėmimų tvarkaraščius studentų ar moksleivių grupėms, priskiriant grupei ar pogrupiui dalyko dėstytoją ar mokytoją, patalpas, tenka atsižvelgti į įvairius ribojimus, minimizuoti „langų“ skaičių per dieną studentams ar dėstytojams [1].

1.4 TIPINIAI TVARKARAŠČIŲ UŽDAVINIAI

Tvarkaraščių uždaviniai, kurie iškyla įvairiose srityse, dažnai turi bendrų ypatybių ir atsiribojant nuo antraeilių kiekvienos srities detalių galima formuluoti tipinius tvarkaraščių uždavinius. Uždavinių formalizavimas kiekvienoje konkrečioje srityje padeda priskirti juos tipinių uždavinių grupei ir parinkti tinkamus sprendimo metodus. Dažnai būna, jog iš pažiūros skirtingų taikomųjų sričių uždaviniai gali būti suvedami į tą pačią uždavinių klasę. Tvarkaraščių uždaviniai pasižymi didele įvairove, todėl, priklausomai nuo informacijos, reikalingos tvarkaraščiui apibūdinti, juos galima įvairiai klasifikuoti. Informacija, reikalinga tvarkaraščiams sudaryti, apima duomenis apie užduočių trukmes, jų nuoseklumo sąryšius, naudojamus išteklius užduotims atlikti, išteklių atsargas bei ribojimus, procesorių (mašinų ar įrenginių), atliekančių užduotis, pajėgumą, tvarkaraščio realizavimo ypatybes ir panašiai.

Dažniausiai išskiriami tokie kriterijai tvarkaraščiams klasifikuoti [1]:

- Statinis arba dinaminis duomenų srauto pobūdis,
- Išteklių tipai,
- Ribojimų pobūdis,
- Duomenų apibrėžtumas.

1.4.1 STATINIAI IR DINAMINIAI TVARKARAŠČIAI

Tvarkaraščių planavimas gali būti statinis arba dinaminis. Statinio planavimo atveju naudojant fiksuotus ir iš anksto žinomus duomenis apie darbų srautą, reikia parinkti darbų eiliškumą, atsižvelgiant į ribojimus, susijusius su darbų nuoseklumo sąryšiais, medžiagų, žaliavų ar finansų ištekliais, vykdytojų gebėjimus atlikti darbus ir pan. Tokie uždaviniai sprendžiami tinklinio arba kalendorinio planavimo metodais.

Dinaminiuose uždaviniuose tenka priimti sprendimus atsižvelgiant į turimus darbo jėgos, techninių įrengimų, laiko bei finansinius ribojimus, apdorojant duomenų srautą apie iškylančius darbus. Tokio tipo uždaviniai dažnai iškyla transporto, krovos, gamybos organizavimo metu,

telekomunikacijų ir ryšių tinklų valdymo metu ir pan. Dinaminiuose uždaviniuose informacija apie darbus gali būti iš anksto nežinoma ir gali paaiškėti tik proceso vykdymo metu [1].

1.4.2 TVARKARAŠČIAI SU ĮVAIRIAIS IŠTEKLIŲ TIPAIS

Tvarkaraščių uždavinius galima suskirstyti pagal tai, su kokiais ištekliais operuojama uždavinyje. Išskirkime šias grupes:

- Uždaviniai su ribojimais vykdymui – apima uždavinius su darbų pertraukimais ir darbo jėgos arba įrengimų ribojimais. Tokiuose uždaviniuose yra tiriamas vykdymo resursų skirstymas statinių ir dinaminių uždavinių atvejais.
- Uždaviniai su medžiagų ribojimais – išskiriami uždaviniai tiek su medžiagų stygiumi, tiek su jų pertekliumi, taip pat su medžiagų kokybės tikrinimu. Su šiais uždaviniais dažniausiai susiduriame, kai reikia spręsti racionalaus medžiagų atsargų skirstymo uždavinius.
- Uždaviniai su ribotais žinių ištekliais – apima uždavinius, kuriuose tenka priimti sprendimus, kai nepakanka žinių turimai informacijai apdoroti arba žinios yra pasenusios ir nebetinka naujai situacijai. Žinių ištekliais gali būti faktai, technologijos, strategijos. Svarbu žinių išteklius vystyti ir tobulinti, nes tai garantuotų įvairių tvarkaraščių uždavinių sprendinio suradimą.

Uždaviniai su išteklių ribojimais dažnai yra sprendžiami pasirinkus ekonominius kriterijus ir atsižvelgiant į finansinę ribojimų išraišką [1].

1.4.3 TVARKARAŠČIŲ RIBOJIMŲ TIPAI

Ribojimai tvarkaraščiuose gali būti labai įvairūs. Gali būti, jog tam tikru laiko momentu nepakaks kokių nors išteklių ar bus nusižengta nuoseklumo sąryšiams, gali būti reikalaujama paisyti tam tikrų laiko terminų ir panašiai.

Pagal ribojimų sritis (sferas), tvarkaraščių uždavinius galima suskirstyti į šias grupes [1]:

- Tvarkaraščiai su laiko ribojimais – apima uždavinius su laiko reikalavimais darbų pradžiai ar pabaigai. Su vienokiais ar kitokiais laiko ribojimais susiduriama visuose tvarkaraščių uždaviniuose.
- Tvarkaraščiai su išteklių ribojimais – apima nekintamus, ilgalaikius ar fizinius ribojimus medžiagoms, darbo jėgai bei techniniams įrengimams.
- Tvarkaraščiai su darbų nuoseklumo ribojimais – apima uždavinius su darbų eiliškumo ribojimais. Atskiru atveju gali būti nagrinėjami nepriklausomų užduočių tvarkaraščiai, kuriuose nėra ribojimų užduočių eiliškumui. Dažniausiai yra taip, jog kuo daugiau yra nuoseklumo sąryšių, tuo sunkiau sudaryti gerą tvarkaraštį.

Pagal ribojimų ypatybes tvarkaraščių uždavinius galime suskirstyti į šias grupes:

- Tvarkaraščiai su laikinais ribojimais – ribojimai tokiuose uždaviniuose yra nusakyti tik tam tikriems laiko periodams ir suvaržo procesų pradžios, pabaigos laiko momentus, trukmes, taip pat ir resursų paskyrimus. Kai užduotis tokiuose uždaviniuose jau apdorojama, laikini ribojimai laikomi patenkintais ir naikinami arba atnaujinami.
- Tvarkaraščiai su nenumatytais ribojimais – ribojimai tokiuose uždaviniuose nusako netikėtus ar iš anksto nenumatytus ribojimus, kurie keičia įvairius prioritetus ir išteklių skirstymą. Iškilus naujiems ribojimams, atsižvelgiama į turimus ir einama prie tvarkaraščio perdarymo.
- Tvarkaraščiai su nuolatiniais (ilgalaikiais) ribojimais – dažniausiai tai fiziniai ribojimai, įtakojantys tvarkaraščio sudarymo procesą, resursų skirstymą. Įprastai šių ribojimų neįtakoja kiti ribojimai.

Ribotų išteklių tvarkaraščių sudarymo uždaviniai yra vieni iš svarbiausių tvarkaraščių teorijos uždavinių, kurie plačiai taikomi praktikoje. Mokslinėje literatūroje pažymima, jog sudarant tokių uždavinių efektyvius sprendimo algoritmus lieka dar daug neišspręstų klausimų.

1.4.4 TVARKARAŠČIŲ DUOMENŲ APIBRĖŽTIS

Tvarkaraščių uždaviniai gali būti klasifikuojami pagal duomenų apibrėžties pobūdį [1]:

- Deterministiniai uždaviniai – šiuose uždaviniuose yra žinoma visa informacija, reikalinga tvarkaraščiui sudaryti.
- Stochastiniai uždaviniai – šiuose uždaviniuose duomenys yra aprašomi tikimybiniais modeliais.
- Nedeterministiniai (nestochastiniai) uždaviniai – šiuose uždaviniuose dalis duomenų yra neapibrėžti ir neapibrėžtumas yra aprašomas netikimybiniais modeliais (blausioji logika, intervalinė aritmetika ir pan.).
- Nepilnai apibrėžti uždaviniai – šiuose uždaviniuose dalis duomenų gali būti nežinoma, praleista arba turėti klaidų.

Šiuo metu yra labiausiai ištirti įvairių tipų deterministiniai tvarkaraščių sudarymo uždaviniai.

Apžvalgose yra pažymima, jog tiriant stochastinių ir nedeterministinių (nestochastinių) tvarkaraščių sudarymo metodus lieka dar daug neišspręstų uždavinių. Nepilnai apibrėžti uždaviniai yra sprendžiami suvedant juos į jau minėtus deterministinius arba nedeterministinius uždavinius.

1.5 TVARKARAŠČIŲ UŽDAVINIŲ SPRENDIMO ALGORITMAI

Tvarkaraščiams sudaryti yra naudojami įvairūs kombinatorinio optimizavimo algoritmai, apdorojantys formalizuotus tvarkaraščio pradinius duomenis. Šie algoritmai skirstomi į tiksliuosius, euristinius ir meta-euristinius.

Tiksliaisiais vadinami tokie algoritmai, kurie randa tikslo funkcijos minimumo tašką. Šie algoritmai taikytini tik mažos apimties uždaviniams, nes sprendžiant didelės apimties uždavinius, tiksliam sprendiniui rasti reiktų labai daug laiko (skaičiuojama metais).

Tikslieji algoritmai:

- kertančiųjų plokštumų (cutting planes);
- šakų ir ribų (branch and bound);
- šakų ir rėžių (branch and cut).

Euristika – kokia nors taisyklė, principas ar metodika, nusakanti kaip ir kokį veiksmą reikia atlikti, kad būtų surastas koks nors sprendinys ar priimtas koks nors sprendimas. Euristinių metodų tikslas – greitai gauti gerą sprendinį, nesuteikiant jokios garantijos sprendinio kokybei.

Euristiniai metodai:

- lokaliosios paieškos (local search);
- prisitaikančios atminties (adaptive memory);
- nuolatinio judėjimo (sweep).

Meta-euristikos yra aukšto lygio iteracinės procedūros, kurios valdo paprastas euristikas, sujungdamos skirtingas leistinosios srities tyrinėjimo idėjas, sprendinių, artimų globaliajam minimumui, paieškai. Kiekvienoje iteracijoje meta-euristiniai algoritmai manipuliuoja vienu gautu sprendiniu ar jų populiacija. Daugelyje praktinių uždavinių meta-euristiniai algoritmai yra efektyvūs ir našūs, lankčiai pritaikomi prie įvairių uždavinio modifikacijų. Ta pati meta-euristika gali būti pritaikyta daugeliui uždavinių, net jeigu tie uždaviniai turi labai skirtingas kombinatorines struktūras. Meta-euristiniai algoritmai sistemingai ieško sprendinio, retkarčiais priimdami atsitiktinius sprendimus ir taip praplėsdami paieškų sritį naujais regionais. Kai kurie algoritmai naudoja atmintį, kad išvengtų sprendinių pasikartojimo [12], [16], [22].

Šiuolaikiniai meta-euristiniai algoritmai:

- modeliuojamas atkaitinimas (simulated annealing);
- genetiniai algoritmai (genetic algorithms);
- paieška su apribojimais (tabu search);
- godžioji atsitiktinė adaptyvioji paieška (greedy randomized adaptive search);
- išbarstytoji, išsklaidytoji paieška (scatter search);
- skruzdžių kolonijos optimizacija (ant colony optimization);
- spiečių algoritmai (swarm algorithms);

- kintamos aplinkos paieška (variable neighbourhood search);
- dirbtinių neuroninių tinklų metodai (neural networks);
- memetiniai algoritmai (memetic algorithms);
- kiti algoritmai bei įvairūs hibridai.

Meta-euristiniai algoritmai gali būti suklasifikuoti įvairiais būdais, atsižvelgiant į jų techniką (įkvėpti gamtos ir negamtiniai), sprendinių, kuriais manipuliuojama kiekvienoje iteracijoje, skaičių (vienas sprendinys ir sprendinių populiacija), kaimynystės ar tikslo funkcijos aprašymą (pastovi, kintanti), naudojamą atmintį (su atmintimi ir be atminties).

Šakų ir ribų (branch and bound), modeliuojamojo atkaitinimo (simulated annealing) ir paieškos su draudimais (tabu search) metodai plačiau aptariami atitinkamai 1.8, 1.9 ir 1.10 skyreliuose.

1.6 PAGRINDINIAI GAMYBOS TVARKARAŠČIŲ SUDARYMO UŽDAVINIAI

Gamybos tvarkaraščių sudarymo uždaviniai yra plačiai ir įvairiapusiškai išnagrinėti. Klasikinis tokio uždavinio modelis susideda iš mašinų (procesorių, įrenginių) ir darbų. Kiekvienas darbas susideda iš tam tikros aibės operacijų. Kiekvienai operacijai atlikti yra nurodyta mašina ir operacijos apdorojimo trukmė. Darbo operacijų apdorojimo laikai negali persidengti, t.y., kol nepabaigta apdoroti viena operacija, kita negali būti pradėta. Jei operacija jau pradėta apdoroti, jos vykdymas negali būti nutrauktas. Kiekviena mašina bet kuriuo laiko momentu daugiausiai gali apdoroti vieną operaciją. Kiekviena operacija atliekama viena mašina [8].

Trys gerai išnagrinėti modeliai yra gamybos be nuoseklumo sąryšių (open shop), dalinai nuoseklios gamybos (job shop) ir nuoseklios gamybos (flow shop) uždaviniai.

Gamybos be nuoseklumo sąryšių (open shop) uždavinyje darbo operacijos gali būti apdorojamos bet kokia tvarka. Tarkime, mašinų skaičius $m \in Z^+$, darbų aibė *J*. Kiekvienas darbas $j \in J$ susideda iš m operacijų $o_{i,j}$, $1 \le i \le m$ (operacija $o_{i,j}$ apdorojama *i*-tąja mašina), ir kiekvienos operacijos apdorojimo trukmės $l_{i,j} \in N$. Tvarkaraštis, sudarytas darbų aibei *J*, yra vienos mašinos tvarkaraščių rinkinys, t.y. mašinai i, $1 \le i \le m$, randama tokia atitiktis $f_i : J \to N$, kur $f_i(j) > f_i(j')$ reiškia $f_i(j) \ge f_i(j') + l_{i,j'}$, o kiekvienam darbui $j \in J$ intervalai $\left[f_i(j), f_i(j) + l_{i,j}\right)$ yra nesikertantys. Tokio tvarkaraščio darbų pabaiga $\max_{1 \le i \le m,\ j \in J} f_i(j) + l_{i,j}$. Uždavinio esmė – rasti operacijų atlikimo sekas kiekvienam darbui ir darbų atlikimo seką kiekvienai iš mašinų [6], [14].

Dalinai nuoseklios gamybos (job shop) uždavinyje darbo operacijos atliekamos griežtai tam tikra, nuo darbo priklausančia, tvarka. Tarkime, mašinų skaičius $m \in Z^+$, darbų aibė *J*. Kiekvienas darbas $j \in J$ susideda iš operacijų $o_{i,j}$, $1 \le i \le m$, sekos n_j, kiekvieną operaciją apdorojančios

mašinos $p_{i,j} \in [1..m]$ ir apdorojimo trukmės $l_{i,j} \in N$. Tvarkaraštis, sudarytas darbų aibei *J*, yra vienos mašinos tvarkaraščių rinkinys, t.y. kiekvienai mašinai randama atitiktis $f_p : \{o_{i,j} : p_{i,j} = p\} \to N$, kur $f_p(o_{i,j}) > f_p(o_{i',j'})$ reiškia $f_p(o_{i,j}) \geq f_p(o_{i',j'}) + l_{i',j'}$, o $f_p(o_{i+1,j}) \geq f_p(o_{i,j}) + l_{i,j}$. Tokio tvarkaraščio darbų pabaiga $\max_{j \in J} f_p(o_{n_j,j}) + l_{n_j,j}$. Uždavinio esmė – rasti operacijų atlikimo seką kiekvienai mašinai [6], [14].

Nuoseklios gamybos (flow shop) uždavinys yra atskiras dalinai nuoseklios gamybos (job shop) uždavinio atvejis, kai kiekvieno darbo operacijų atlikimo tvarka yra vienoda, o operacijų skaičius yra lygus mašinų skaičiui. Tarkime, mašinų skaičius $m \in Z^+$, darbų aibė *J*. Kiekvienas darbas $j \in J$ susideda iš *m* operacijų $o_{i,j}$, $1 \leq i \leq m$ (operacija $o_{i,j}$ apdorojama *i*-tąja mašina), ir kiekvienos operacijos apdorojimo trukmės $l_{i,j} \in N$. Tvarkaraštis, sudarytas darbų aibei *J*, yra toks, kad kiekvienam darbui $j \in J$ ir $1 \leq i \leq m$, $f_{i+1}(j) \geq f_i(j) + l_{i,j}$. Tokio tvarkaraščio darbų pabaiga $\max_{j \in J} f_m(j) + l_{m,j}$. Uždavinio esmė – rasti darbų atlikimo seką kiekvienai mašinai [6], [14].

Dažniausiai užduotis yra minimizuoti visų tvarkaraščio darbų pabaigą (makespan) su aukščiau aprašytais nuoseklumo apribojimais. Taip pat gali būti ieškoma tokia darbų bei operacijų atlikimo seka, kad laikas, reikalingas visiems darbams atlikti, būtų mažesnis už nurodytą laiką T. Gamyboje taip pat svarbu kuo labiau sumažinti mašinų prastovų laiką, t.y. laiką, kai mašina, apdorojusi operaciją, laukia kitos operacijos apdorojimo pradžios.

Realiai sprendžiami uždaviniai yra šių pagrindinių uždavinių įvairios modifikacijos. Pavyzdžiui, gali būti kelios identiškos mašinos nurodomos mašinos tipu, gali būti vienodo tipo mašinų, besiskiriančių aptarnavimo sparta, gali būti mašinų, gebančių apdoroti kelias operacijas vienu metu ir t.t. Gali būti leidžiamas operacijų apdorojimo persidengimas, t.y. aprašant kiekvieną operaciją, nurodomas laiko momentas, kada gali būti pradėta vykdyti kita šio darbo operacija (vadinamasis lag time). Mašinos gali dirbti ne visą laiką, o tik nurodytais intervalais. Operacijos apdorojimas gali būti nutraukiamas arba ne. Nutraukus operacijos apdorojimą, ji toliau gali būti apdorojama ta pačia arba jau kita to paties tipo mašina, arba apdorojimas turi būti pradėtas iš naujo. Kai kuriuose uždaviniuose reikia atsižvelgti į laiko sąnaudas, reikalingas žaliavoms ar pusfabrikačiams nugabenti iš vienos gamybos vietos į kitą ir pan. Uždaviniai taip pat skiriasi darbų parinkimo pirmenybe: darbų vykdymo seka parenkama atsitiktinai, atsižvelgiant į eilę (FIFO – pirmas atėjo, pirmas išėjo), atsižvelgiant į darbo pabaigos terminą (deadline), pirmiausiai vykdomi tie darbai, kuriems atlikti reikia mažiausiai laiko ir t.t.

1.7 GAMYBOS TVARKARAŠČIŲ SUDARYMO SU RIBOTU MAŠINŲ PARENGTUMU APŽVALGA

Tvarkaraščių sudarymo teorijos pagrindiniame modelyje tariama, kad visos mašinos be perstojo yra pasirengusios aptarnauti. Tokia prielaida kai kuriais atvejais gali būti taikoma, tačiau ji neaprėpia atvejų, kai mašinos yra nepasirengusios aptarnauti dėl atliekamų remonto darbų, gedimų ar kitokių apribojimų [13], [17], [18].

Tokių apribojimų pavyzdžių galima rasti daugelyje sričių. Tvarkaraščiai, sudaryti neatsižvelgiant į ribotą mašinų parengtumą, gali netekti prasmės, nes realaus pasaulio išteklių planavimo uždaviniai yra dinaminiai. Tai reiškia, kad informacija yra nuolat atnaujinama. Vienas iš būdų susidoroti su dinamiškai besikeičiančia aplinka – iš naujo pradėti planuoti, kai tik pasikeičia duomenys. Tačiau dėl daugelio reikalingų atlikti darbų, pavyzdžiui, operacijos paruošimo apdorojimui, būtina atsižvelgti į ankstesnio plano rezultatus, o tai riboja išteklių panaudojimą sudarant naują planą. Pavyzdžiui, gamyboje fiksuojami darbų apdorojimo pradžios ir pabaigos laiko momentai ir mašina, kuri šį darbą apdoroja. Kai ceche (stage) atsiranda naujų darbų, ankstesnių darbų apdorojimo pradžios ir pabaigos laiko momentai bei apdorojančios mašinos jau yra paskirstyti. Tuo tarpu nauji darbai turi būti apdoroti iki tam tikro nurodyto laiko.

Dar vienas riboto mašinų parengtumo pavyzdys – operacinės sistemos, turinčios vieną ar daugiau procesorių, kur didesnio prioriteto paprogramė susiduria su dabar veikiančia programa. Panašios problemos iškyla kompiuterių sistemose, turinčiose daug vartotojų, kur apkrautumas keičiasi priklausomai nuo vartojimo. Didelėse paraleliose sistemose patogu keisti procesoriaus dalį tarp skirtingų tipų vartotojų priklausomai nuo jų kompiuterio poreikių. Kaita, susijusi su apdorojimo pajėgumu, gali būti modeliuojama procesoriaus skirtingo pajėgumo intervalais. Apstu kitų pavyzdžių, kuriuose ribotas mašinų parengtumo nagrinėjimas yra labai svarbus, ir šios problemos sprendimo praktinis poreikis pasireiškia komercinės programinės įrangos paketų augančia paklausa. Šių problemų analizė taip pat pritraukė nemažai tyrėjų.

Mašinų sistema su ribotu parengtumu yra mašinų (procesorių), kurios dirba su pertraukomis, aibė; kiekviena mašina pasirengusi apdoroti tik tam tikrais laiko intervalais. Tegu $T = \{T_j \mid j = 1,\ldots,n\}$ yra užduočių aibė, o $P = \{P_i \mid i = 1,\ldots,m\}$ – mašinų (procesorių) aibė, kur mašina P_i gali apdoroti tik per S_i duotų laiko intervalų $[B_i^s, F_i^s)$, $s = 1,...,S_i$, o $B_i^{s+1} > F_i^s$ visiems $s = 1,...,S_i$. B_i^s žymi s-tojo intervalo pradžios laiko momentą, o F_i^s pabaigos laiko momentą, kai mašina P_i yra pasirengusi dirbti. Gali būti pilna ar nepilna informacija apie darbingumo intervalus. Kai kuriais atvejais visi B_i^s ir F_i^s yra žinomi iš anksto; kitais atvejais žinoma tik jų dalis. Gali būti ir taip, kad iš anksto apie mašinos parengtumą iš viso nieko nežinoma.

Kiekvienai užduočiai T_j atlikti reikia p_j laiko vienetų. Aibėje *T* gali būti nustatyti pirmumo apribojimai. $T_i < T_j$ reiškia, kad T_i turi būti pabaigta prieš pradedant T_j. Užduotys aibėje *T* vadinamos priklausomomis, jei apdorojimo tvarka tokiais sąryšiais yra nustatyta bent dvejoms užduotims. Tada šie sąryšiai gali būti modeliuojami naudojant pirmumo grafą. Kitu atveju užduotys vadinamos nepriklausomomis.

Kiekviena mašina vienu laiko momentu gali apdoroti tik vieną užduotį, o kiekviena užduotis vienu laiko momentu gali būti apdorojama tik viena mašina. Mašinos gali būti paralelios, t.y. atlieka tokias pačias funkcijas, arba paskirtos, t.y. specializuotos tik tam tikroms užduotims atlikti. Jei visų mašinų P_i iš aibės *P* užduočių apdorojimo sparta yra vienoda, tuomet jos vadinamos identiškomis. Paskirtų mašinų atveju yra trys užduočių apdorojimo modeliai: nuoseklios gamybos (flow shop), dalinai nuoseklios gamybos (job shop) ir nenuoseklios gamybos (open shop). Vėliau nagrinėsime nuoseklios gamybos (flow shop) atvejį. Tokiose sistemose tariame, kad užduotys suformuoja *n* poaibių (grandinių). Kiekvienas poaibis vadinamas darbu. Tai yra, darbas J_j suskaidomas į *m* operacijų, T_{1j}, T_{2j}, ..., T_{mj}, o operacija T_{ij} bus apdorota P_i mašina. Be to, operacija T_{i-1j} turi būti apdorojama prieš T_{ij} visiems $i = 2, \ldots, m$ ir visiems $j = 1, 2, \ldots, n$. Darbų aibė žymima *J*.

Norime rasti tokį tinkamą tvarkaraštį (laiku paremtą mašinų iš *P* paskirstymą užduotims iš aibės *T* taip, kad būtų tenkinami visi apribojimai), jei toks egzistuoja, kad visos užduotys būtų atliekamos per duotus mašinų parengtumo intervalus, optimizuojant kažkokį vykdymo kriterijų. Tokie kriterijai gali būti užbaigimo laikas, užbaigimo laikas susijęs su nustatytu laiku, maksimalus užbaigimo laikas, užbaigimo laikų suma ir maksimalus vėlavimas.

Tvarkaraščių sudarymo literatūroje paralelių mašinų sistemoms aprašomi skirtingi mašinų parengtumo modeliai: pastovus, zigzaginis, mažėjantis, didėjantis ir laiptuotas. Tegul $0 = t_1 < t_2 < \ldots < t_j < \ldots < t_q$ yra laiko momentai, kai pasikeičia konkrečios mašinos parengtumas, o $m(t_j)$ mašinų, galinčių apdoroti laiko intervale $[t_j, t_{j+1})$, skaičius, $m(t_j) > 0$. Laikoma, kad mašinų parengtumo modelis nesikeičia be galo daug kartų per bet kurį baigtinį laiko intervalą. Remiantis šiais faktais, parametras $\alpha_3 \in \{\varnothing, NC_{zz}, NC_{inc}, NC_{dec}, NC_{inczz}, NC_{deczz}, NC_{sc}, NC_{win}\}$ apibūdina mašinų parengtumą.

1. Jei visos mašinos dirba be perstojo ($t = 0$), tuomet modelis vadinamas pastoviu ($\alpha_3 = \varnothing$).
2. Jei kiekviename intervale yra tik *k* arba $k-1$ pasirengusių aptarnauti mašinų, tuomet modelis vadinamas zigzaginiu ($\alpha_3 = NC_{zz}$).

3. Modelis vadinamas didėjančiu (mažėjančiu), jei visiems j iš IN_+ $m(t_j) \geq \max_{1 \leq u \leq j-1} \{m(t_u)\}$ ($m(t_j) \leq \max_{1 \leq u \leq j-1} \{m(t_u)\}$), t.y. pasirengusių aptarnauti intervale $[t_{j-1}, t_j)$ mašinų skaičius yra nedidesnis (nemažesnis) negu šis skaičius yra intervale $[t_j, t_{j+1})$ ($\alpha_3 \in \{NC_{inc}, NC_{dec}\}$).
4. Modelis vadinamas didėjantis (mažėjantis) zigzaginis, jei visiems j iš IN_+ $m(t_j) \geq \max_{1 \leq u \leq j-1} \{m(t_u) - 1\}$ ($m(t_j) \leq \max_{1 \leq u \leq j-1} \{m(t_u) + 1\}$) ($\alpha_3 \in \{NC_{inczz}, NC_{deczz}\}$).
5. Modelis vadinamas laiptuotas, jei visiems intervalams mašinos P_i parengtumas nusako mašinos P_{i-1} parengtumą ($\alpha_3 = NC_{sc}$).

Atkreikite dėmesį, kad 1 – 4 modeliai yra atskiri laiptuoto modelio atvejai.

6. Modelis vadinamas sutartiniu, jei netaikomos 1 – 5 sąlygos ($\alpha_3 = NC_{win}$).

1 – 5 modeliai yra atskiri 6 modelio atvejai.

Tokiems uždaviniams spręsti taikomi skirtingi algoritmai. Informacija apie mašinų parengtumą gali būti pilna arba nepilna. Tiesioginio (on-line) darbo režimo atveju mašinų parengtumas iš anksto nėra žinomas. Netikėti mašinų gedimai yra tipiniai tiesioginio darbo režimo pavyzdžiai. Kartais tvarkaraščių sudarytojai turi dalinę informaciją apie parengtumus, t.y. jie turi galvoti apie ateitį. Jie gali žinoti, kuriai mašinai per ateinantį laiko intervalą reikalingas remontas ar kada sugedusi mašina vėl galės aptarnauti. Netiesioginio (off-line) darbo režimo atveju tariama, kad visa informacija žinoma iš anksto, t.y. visa informacija apie mašinų parengtumus žinoma prieš sudarant tvarkaraštį. Dėl šių galimybių galime pritaikyti skirtingo tipo algoritmus.

- Algoritmas yra pritaikytas tiesioginio darbo režimui (on-line), jei informacija apie esamą situaciją nuosekliai atnaujinama ir kiekvienu laiko momentu t reikia žinoti tik pasirengusių aptarnauti laiko momentu t mašinų skaičių, paruoštų apdorojimui užduočių skaičių momentu t, jų likusius apdorojimo laikus ir galutinį terminą (deadline) ar nustatytus laikus (due-dates).
- Algoritmas yra beveik pritaikytas tiesioginio darbo režimui (nearly on-line), jei be viso to laiko momentu t dar reikia žinoti, kada pasirodys kitas įvykis, t.y. kada bus paruošta apdorojimui nauja užduotis ar pasikeis aptarnaujančių mašinų skaičius.
- Algoritmas yra pritaikytas netiesioginio darbo režimui (off-line), jei visa informacija turi būti žinoma iš anksto. Tai yra, laiko momentu 0 reikalinga visa informacija apie mašinų parengtumus ir užduotis.

Jei mašinos neparengtumas susijęs su netikėtais gedimais, tuomet reikia naudoti pritaikytus tiesioginio darbo režimui (on-line) algoritmus. Jei laiko momentai, kada pasikeis mašinų parengtumas,

žinomi truputį į priekį, tuomet naudojami beveik pritaikyti tiesioginio darbo režimui (nearly on-line) algoritmai. Kitais atvejais naudojami pritaikyti netiesioginio darbo režimui (off-line) algoritmai.

1.8 ŠAKŲ IR RIBŲ ALGORITMAS

Šakų ir ribų algoritmas yra tikslusis algoritmas skirtas įvairiems optimizavimo uždaviniams, ypač diskretaus ir kombinatorinio optimizavimo uždaviniams, spręsti. Pirmą kartą šis metodas buvo pasiūlytas A. H. Land ir A. G. Doig 1960 m. [24].

Trumpai apžvelkime pagrindines šakų ir ribų (branch and bound) algoritmo sudedamąsias dalis. Tarkime, reikia išspręsti tokį kombinatorinio optimizavimo uždavinį: duota baigtinė diskrečioji aibė *X*, funkcija $f : X \to R$, aibė *S*, $S \subseteq X$ ir optimalus sprendinys $x^* \in S$ toks, kad $f(x^*) = \min\{f(x) | \forall x \in S\}$. Paprastai aibė *S* yra apribota aibė ir vadinama galimų sprendinių sritimi, t.y. visi elementai $x \in S$ yra galimi sprendiniai. Tariame, kad *S* yra baigtinė arba tuščia.

Sprendinio radimas šakų ir ribų metodu susideda iš aibės *S* elementų išvardijimo, nagrinėjant tik galimų sprendinių poaibius. Kiti sprendiniai, jei negali duoti galimo ar optimalaus sprendinio, yra atmetami.

Sprendinių išvardijimas – tai paieškos medžio, kurio viršūnės yra sprendinių poaibiai, konstravimas. Medžio dydis, t.y. sugeneruotų viršūnių skaičius, priklauso nuo paieškos medžio konstravimo strategijos.

Šakų ir ribų algoritmas gali būti aprašytas taip:

- konstruoti paieškos medį;
- genėti (mažinti) šakas;
- baigti paiešką, kai tenkinamos tam tikros sąlygos.

1.8.1 PAIEŠKOS MEDŽIO KONSTRAVIMAS

Šakojimo operacija skaido aibę *X* į vis mažesnius poaibius tol, kol gauname tokį uždavinį, kurį galime išspręsti. Paieškos medžio viršūnės reprezentuoja poaibius (dalinius sprendinius), o briaunos – ryšį tarp tėvo-poaibio ir jo vaikų-poaibių, gautų šakojant viršūnę.

Medžio tyrinėjimo strategija parenka vieną viršūnę iš daugelio nebaigtų nagrinėti viršūnių pagal iš anksto nustatytas taisykles. Viršūnės parinkimo prioritetas dažniausiai paremtas arba paieškos medžio viršūnės gyliu („pirmas giliausias“ paieškos strategija), arba numanomu viršūnės gebėjimu duoti gerą sprendinį („pirmas geriausias“ paieškos strategija). Paieškos algoritmas „pirmas giliausias“ ieškoti pradeda nuo šakninės viršūnės, kuri yra šakojama; tuomet pasirenkama pirma viršūnė-vaikas ir šaka nagrinėjama kiek galima giliau tol, kol randamas ieškomas sprendinys arba pasiekiama vaikų neturinti viršūnė. Tuomet paieška grįžta tuo pačiu keliu ir nagrinėja kitą dar nebaigtą tirti šaką. Paieškos strategija „pirmas geriausias“ ieškoti pradeda nuo šakninės viršūnės, kuri yra šakojama,

tuomet iš visų viršūnių-vaikų pasirenkama ta, kuri pagal tam tikrus kriterijus (dažniausiai įvertinami apatiniai ar viršutiniai rėžiai) žada geriausią sprendinį [3], [5]. Mes naudojame šių abiejų strategijų kombinaciją.

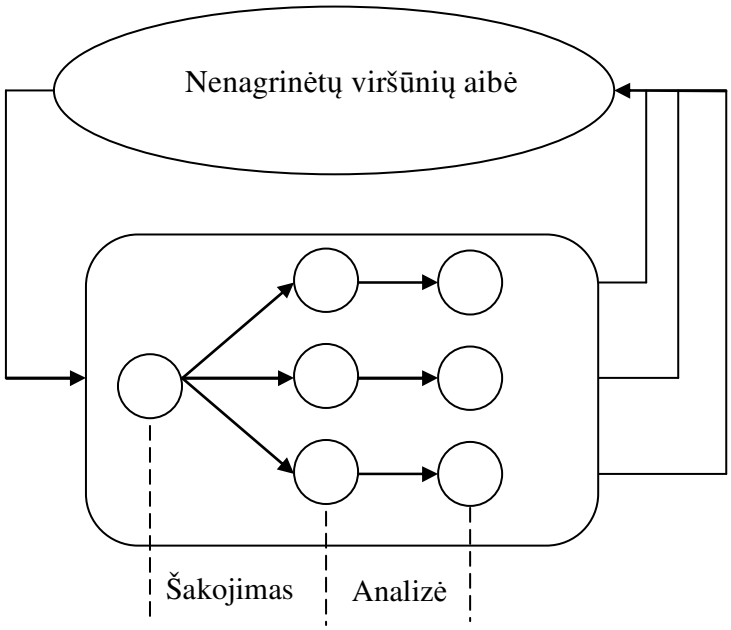
1.8.2 ŠAKŲ GENĖJIMAS

Kiekvienai viršūnei, gautai šakojimo metu, skaičiuojamas apatinis rėžis. Tokiu būdu sumažinama paieška, kadangi tik tos viršūnės, kurių apatinis rėžis patenka į tam tikrą paieškos intervalą, yra tiriamos toliau, o kitos yra pašalinamos. Viršutinis rėžis, apskaičiuojamas naudojant žinomą galimą arba euristiniu būdu gautą sprendinį, priklauso paieškos intervalui. Viršutinis rėžis periodiškai yra atnaujinamas, kai tik randamas naujas geriausias iki šiol rastas sprendinys. Mūsų realizacijos atveju pradinis sprendinys randamas naudojant SPT taisyklę (SPT – shortest processing time first). Surastam sprendiniui apskaičiuojamas pradinis viršutinis rėžis.

1.8.3 PAIEŠKOS BAIGIMO SĄLYGOS

Šios sąlygos nurodo, kada uždavinys yra išspręstas ir rastas optimalus sprendinys. Tai atsitinka, kai visos viršūnės yra išnagrinėtas arba atmestos.

1.8.4 ŠAKŲ IR RIBŲ ALGORITMO PSEUDO KODAS



1.4 pav. Grafiškai pavaizduotas šakų ir ribų algoritmas

Šakų ir ribų algoritmas, kaip parodyta 1.4 paveiksle, susideda iš kelių paprastų operacijų, taikomų skirtingo ar vienodo prioriteto viršūnėms, kurios sudaro viršūnių aibę, dažnai vadinamą pirmumo eile:

- *ištrinti_min* - pasirinkti ir ištrinti aukščiausią prioritetą turintį elementą;
- *įterpti* - įterpti naują elementą, kurio prioritetas žinomas;
- *ištrinti_didesnį* - ištrinti elementus, kurie netenkina tam tikrų sąlygų.

1.5 paveiksle pateiktas šakų ir ribų algoritmo pseudo kodas.

```
Viršūnė šaknis;       // šaknies viršūnė.
PirmumoEilė eilė;     // eilė, kurioje prioriteto tvarka išrikiuotos
                      // dar nenagrinėtos viršūnės
Viršūnė geriausias;   // geriausias iki šiol rastas sprendinys

eilė.įterpti(šaknis);
kol(¬eilė.tuščia()) {
  Viršūnė v = aibė.ištrinti_min();
  kiekvienam(Viršūnė s sūnus v) { // šakojimas
    s.analizuoti(); // viršūnės s įvertinimas
    jei(s.yraSprendinys() ∧ geriausias.kaina() > s.kaina()) {
      geriausias = s;
      eilė.ištrinti_didesnį(geriausias.kaina());
    }
    kitaip jei(s.galimasSprendinys() ∧ geriausias.kaina() > s.analizuoti())
      eilė.įterpti(s);
  }
}
```

1.5 pav. Šakų ir ribų algoritmo pseudo kodas

1.8.5 UŽDAVINYS

Toliau pateiktas uždavinys ir jo sprendimo būdas yra aprašytas [23] su tam tikrais pakeitimais. Nagrinėjamas nuoseklios gamybos (flow shop) uždavinys, susidedantis iš $m \geq 2$ cechų (mašinų centrų) $[Z_1, Z_2, \ldots, Z_m]$, kurių kiekviename yra $m_j \geq 1$ lygiagrečių (parallel) mašinų $\{M_{j1}, M_{j2}, \ldots, M_{jm_j}\}$. Mašinos M_{jk} spartą nusako g_{jk}, o vidutinė *j*-tojo cecho mašinų apdorojimo sparta $ag_j = \sum_{k=1}^{m_j} g_{jk} / m_j$. Reikia apdoroti *n* darbų $\{J_i \mid 1 \leq i \leq n\}$. Kiekvienas darbas J_i susideda iš *m* operacijų, kurių kiekvienai atlikti reikalingas laikas p_{ij} $(1 \leq i \leq n,\ 1 \leq j \leq m)$. Visi darbai atliekami ta pačia tvarka, operacija turi būti apdorota viena iš ceche esančių lygiagrečių mašinų. Tariame, kad kiekvienos mašinos nedarbingumo intervalai nepersidengia ir yra išrikiuoti didėjančia tvarka pagal nedarbingumo intervalo pradžios momentus. Mašinos M_{jk} *l*-tojo nedarbingumo intervalo pradžios momentą ir ilgį pažymėkime atitinkamai $s_{jk,l}$ ir $h_{jk,l}$. Kiekviena mašina vienu laiko momentu gali aptarnauti tik vieną operaciją, o kiekviena operacija vienu laiko momentu gali būti apdorojama daugiausia viena mašina. Uždavinio esmė – minimizuoti visų darbų pabaigos laiką (makespan), t.y. maksimalų laiką, reikalingą visoms operacijoms atlikti.

Duota darbų aibė *N*, t.y. $N = \{J_i \mid 1 \le i \le n\}$. Kai visi darbai priskiriami mašinoms visuose cechuose, lengva pastebėti, kad reikia nagrinėti tik tuos tvarkaraščius, kuriuose visi darbai apdorojami kiek galima anksčiau. Visų galimų tvarkaraščių skaičius lygus $\prod_{j=1}^{m}(n!m_j^n)$.

Tariame, kad nuo 1-ojo iki $j-1$-ojo cecho sudarytas dalinis tvarkaraštis, paskirsčius šiuose cechuose visus darbus visoms mašinoms. Toliau, priskirdami visus darbus, sudarinėjame tvarkaraštį *j*-tajame ceche. Duotas surikiuotas aibės *N* poaibis *A*. Tegu $S_j(A)$ yra dalinis tvarkaraštis, sudarytas nuo 1-ojo iki $j-1$-ojo cecho, kartu su poaibio *A* darbų, paskirtų apdoroti *j*-tajame ceche, seka. Mašinos M_{jk} dalinio tvarkaraščio $S_j(A)$ pabaigos laiko momentą pažymėkime $C_k(S_j(A))$. Tegu $C_k(S_j(A)) = 0$, jei mašinai M_{jk} daliniame tvarkaraštyje $S_j(A)$ dar nėra paskirtas darbas. Pažymėkime $JC_j(J_q)$ darbo J_q pabaigos laiko momentą *j*-tajame ceche. Tegu $JC_0(J_q) = 0$ kiekvienam darbui J_q iš *N*.

Imame darbą $J_q \notin A$. Tegu $A' = A \cup \{J_q\}$, o $S_j(A')$ yra papildytas dalinis tvarkaraštis, gautas prie $S_j(A)$ *j*-tajame ceche pridėjus darbą J_q. Akivaizdu, kad anksčiausias laiko momentas, kai darbas J_q gali būti pradėtas apdoroti mašina M_{jk} yra

$$ET(S_j(A), J_q) = \max\{C_k(S_j(A)), JC_{j-1}(J_q)\}. \quad \textbf{(1.1)}$$

Tačiau, kadangi mašinos dirba ne visą laiką, o tik tam tikrais laiko intervalais, darbo apdorojimas gali būti atidėtas. Iš šių samprotavimų išplaukia, kad

$$C_k(S_j(A')) = \begin{cases} ET(S_j(A), J_q) + \Delta_{qj}(S_j(A)) + p_{qj}/g_{qj}, & \textit{jei } J_q \textit{ priskirtas } M_{jk}; \\ C_k(S_j(A)), \textit{ kitu atveju}, \end{cases} \quad \textbf{(1.2)}$$

čia $\Delta_{qj}(S_j(A))$ apibūdina darbo J_q uždelsimo laiką dėl mašinos M_{jk} nedarbingumo apribojimų. Tai reiškia, kad darbas J_q mašina M_{jk} gali būti nebaigtas apdoroti tarp $ET(S_j(A), J_q)$ ir nedarbingumo intervalų, prasidedančių po $ET(S_j(A), J_q)$.

Jei tarp $ET(S_j(A), J_q)$ ir $ET(S_j(A), J_q) + p_{qj}/g_{qj}$ mašina M_{jk} dirba be perstojo, t.y. nėra nedarbingumo intervalų, tuomet darbo J_q nereikia atidėti ir $\Delta_{qj}(S_j(A)) = 0$. Kitu atveju, darbo J_q apdorojimas bus atidėtas. Remiantis tokia situacija, pirmas nedarbingumo intervalas, prasidedantis po $ET(S_j(A), J_q)$ yra

$$l_1 = \min\{l : 0 \le s_{jk,l} - ET(S_j(A), J_q) < p_{qj}/g_{qj}\}. \quad \textbf{(1.3)}$$

Tačiau darbas J_q ir toliau gali būti atidedamas dėl mašinos M_{jk} nedarbingumo intervalų, prasidedančių po l_1 intervalo. Tegu l_2 yra pirmas nedarbingumo intervalas, po kurio darbas J_q gali būti nedelsiant pradėtas apdoroti. Jis gali būti apskaičiuotas taip

$$l_2 = \min\left\{l : l \geq l_1, s_{jk,l+1} - \left(s_{jk,l} + h_{jk,l}\right) \geq p_{qj} / g_{qj}\right\}. \quad \textbf{(1.4)}$$

Taigi darbo J_q uždelsimo laikas

$$\Delta_{qj}\left(S_j(A)\right) = \left(s_{jk,l_2} + h_{jk,l_2}\right) - ET\left(S_j(A), J_q\right). \quad \textbf{(1.5)}$$

Remiantis šiais žymėjimais, maksimalus visų darbų pabaigos laiko momentas (makespan) minimizuojamas tada ir tik tada, kai minimizuojamas $\max_k\left\{C_k\left(S_m(N)\right)\right\}$. Vadinasi, uždavinio tikslo funkcija yra

$$C_{\max} = \max_k\left\{C_k\left(S_m(N)\right)\right\}. \quad \textbf{(1.6)}$$

1.8.6 APATINIAI RĖŽIAI

Toliau pateiksime pabaigos laiko momento apatinių rėžių skaičiavimą.

1.8.6.1 MAŠINOS APATINIS RĖŽIS

Duotas dalinis tvarkaraštis $S_j(A)$. Neapdorotų darbų (*N-A*) apkrova *j*-tajame ceche gali būti panaudota apskaičiuojant pabaigos laiko momento apatinį rėžį visiems darbams *j*-tajame ceche, taigi ir bet kuriame ceche.

Pažymėkime K_1 Z_j cecho mašinų, kurioms daliniame tvarkaraštyje $S_j(A)$ yra priskirta darbų, aibę, o K_2 - likusias Z_j cecho mašinas. Anksčiausias laiko momentas, kada mašina $k \in K_1$ gali pradėti apdoroti aibei *N-A* priklausiančius darbus, yra

$$T_{1k} = \max\left\{\min_{J_q \in N-A} JC_{j-1}\left(J_q\right), C_k\left(S_j(A)\right)\right\}. \quad \textbf{(1.7)}$$

Anksčiausias laiko momentas, kada mašina $k \in K_2$ gali pradėti apdoroti aibei *N-A* priklausiančius darbus, yra

$$T_2 = \min_{J_q \in N-A} JC_{j-1}\left(J_q\right). \quad \textbf{(1.8)}$$

Nesunku pastebėti, kad visų darbų pabaigos laiko momentas *j*-tajame ceche yra mažiausiai

$$T_3 = \left(\sum_{k \in K_1} T_{1k} + |K_2| T_2 + \sum_{J_q \in N-A} p_{qj} / ag_j\right) / m_j. \quad \textbf{(1.9)}$$

Tačiau dėl nedarbingumo intervalų, darbų pabaigos laiko momentas gali būti didesnis. Pažymėkime $\Delta_j\left(S_j(A)\right)$ bergždžiai praleistą laiką *j*-tajame ceche dėl mašinų nedarbingumo intervalų.

Tuomet dalinio tvarkaraščio $S_j(N)$, gauto papildžius dalinį tvarkaraštį $S_j(A)$, darbų pabaigos laiko momento apatinis rėžis gali būti užrašytas

$$ACT\left(S_j(A)\right)=T_3+\Delta_j\left(S_j(A)\right)/m_j=$$

$$=\left(\sum_{k\in K_1}T_{1k}+|K_2|T_2\right)/m_j+\left(\sum_{J_q\in N-A}p_{qj}/ag_j\right)/m_j+\Delta_j\left(S_j(A)\right)/m_j=$$

$$=\left(\sum_{k\in K_1}\max\left\{\min_{J_q\in N-A}JC_{j-1}\left(J_q\right),C_k\left(S_j(A)\right)\right\}+|K_2|\min_{J_q\in N-A}JC_{j-1}\left(J_q\right)\right)/m_j+ \quad \textbf{(1.10)}$$

$$+\left(\sum_{J_q\in N-A}p_{qj}/ag_j\right)/m_j+\Delta_j\left(S_j(A)\right)/m_j.$$

Pirmasis dešinėje pusėje esantis dėmuo nusako anksčiausio galimo laiko momento, nuo kurio *j*-tojo etapo mašinos gali pradėti aibėje *N-A* esančių darbų apdorojimą, vidurkį. Antrasis dėmuo – likusį vidutinį neapdorotų darbų (esančių *N-A*) krūvį *j*-tajame ceche. Paskutinis dėmuo nusako vidutinį bergždžiai praleistą laiką *j*-tajame ceche dėl mašinų nedarbingumo intervalų. Mašinos nedarbingumo laikas gali būti tiksliai apskaičiuotas, pridedant po vieną neapdorotus darbus prie dalinio tvarkaraščio $S_j(A)$. Tačiau tiksliems skaičiavimams atlikti reikia daug laiko, todėl panaudodami apatinį rėžį galime greitai įvertinti $\Delta_j\left(S_j(A)\right)$. Manome, kad $\Delta_j\left(S_j(A)\right)$ susideda tik iš dviejų dėmenų: 1) K_1 aibei priklausančių mašinų nedarbingumo intervalų, prasidedančių tarp T_{1k} ir T_3, ilgių sumos; 2) K_2 aibei priklausančių mašinų nedarbingumo intervalų, prasidedančių tarp T_2 ir T_3, ilgių sumos. Taigi $\Delta_j\left(S_j(A)\right)$ įvertis gali būti užrašytas taip

$$\Delta_j\left(S_j(A)\right)=\sum_{k\in K_1}\left(\sum_{l:T_{1k}\le s_{jk,l}<T_3}h_{jk,l}\right)+\sum_{k\in K_2}\left(\sum_{l:T_2\le s_{jk,l}<T_3}h_{jk,l}\right). \quad \textbf{(1.11)}$$

Remiantis padarytomis prielaidomis, gauname, kad $ACT\left(S_j(A)\right)$ yra bet kurio dalinio tvarkaraščio $S_j(N)$, gauto papildžius dalinį tvarkaraštį $S_j(A)$, darbų pabaigos laiko momento apatinis rėžis, t.y.

$$ACT\left(S_j(A)\right)\le\max_k C_k\left(S_j(N)\right). \quad \textbf{(1.12)}$$

Pagaliau, tvarkaraščių, gautų papildžius dalinį tvarkaraštį $S_j(A)$ ir kuriuose visi darbai visuose cechuose yra paskirstyti, apatinis rėžis gali būti apskaičiuotas taip

$$LBM\left(S_j(A)\right)=ACT\left(S_j(A)\right)+\min_{J_q\in N}\sum_{j'=j+1}^{m}\left(p_{qj'}/ag_{j'}\right). \quad \textbf{(1.13)}$$

1.8.6.2 DARBO APATINIS RĖŽIS

Duotas dalinis tvarkaraštis $S_j(A)$. Informacija apie aibėje *A* esančius darbus gali būti panaudota ieškant anksčiausio nepriskirtų darbų apdorojimo pradžios laiko momento, taigi ir visų darbų visuose cechuose apdorojimo pabaigos laiko momento apatinio rėžio. Tokiu darbo apatinio rėžio pavyzdžiu galėtų būti

$$LBJ_0\left(S_j(A)\right) = \min_k C_k\left(S_j(A)\right) + \max_{J_q \in N-A} \sum_{j'=j}^{m} \left(p_{qj'} / ag_{j'}\right). \qquad \textbf{(1.14)}$$

Jei kai kurioms mašinoms nėra priskirtas joks darbas, tuomet anksčiausio nepriskirtų darbų apdorojimo pradžios laiko momento įvertis bus lygus 0, taigi ir visų darbų visuose cechuose pabaigos laiko momento įvertis bus mažas. Atsižvelgiant į nepriskirtų darbų apdorojimo pabaigos laiko momentus likusiuose cechuose, apdorojimo pabaigos momentų apatinio rėžio įvertis padidės. Taigi, darbo apatinis rėžis gali būti pakeistas į

$$LBJ\left(S_j(A)\right) = \begin{cases} \min\left\{ \min\limits_{J_q \in N-A} JC_{j-1}\left(J_q\right), \max\limits_{k \in K_1} C_k\left(S_j(A)\right) \right\} + \\ \quad + \max\limits_{J_q \in N-A} \sum\limits_{j'=j}^{m} \left(p_{qj'} / ag_{j'}\right), \; jei \; K_2 \neq \varnothing \\ \min\limits_k C_k\left(S_j(A)\right) + \max\limits_{J_q \in N-A} \sum\limits_{j'=j}^{m} \left(p_{qj'} / ag_{j'}\right), \; jei \; K_2 = \varnothing \end{cases} . \qquad \textbf{(1.15)}$$

1.8.6.3 JUNGTINIS APATINIS RĖŽIS

Jei kuriame nors ceche apdorojamų darbų skaičius yra daug didesnis nei jame esančių mašinų skaičius, tuomet mašinos apatinis rėžis bus artimesnis sudaryto tvarkaraščio apdorojimo pabaigos laiko momentui. Kita vertus, jei apdorojamų darbų skaičius yra daug mažesnis nei kažkuriame ceche esančių mašinų skaičius, tai darbo apatinis rėžis bus artimesnis sudaryto tvarkaraščio apdorojimo pabaigos laiko momentui.

Kombinuodami mašinos apatinį rėžį ir darbo apatinį rėžį, gauname jungtinį apatinį rėžį

$$LBC\left(S_j(A)\right) = \max\left\{LBM\left(S_j(A)\right), LBJ\left(S_j(A)\right)\right\}. \qquad \textbf{(1.16)}$$

1.9 MODELIUOJAMOJO ATKAITINIMO ALGORITMAS

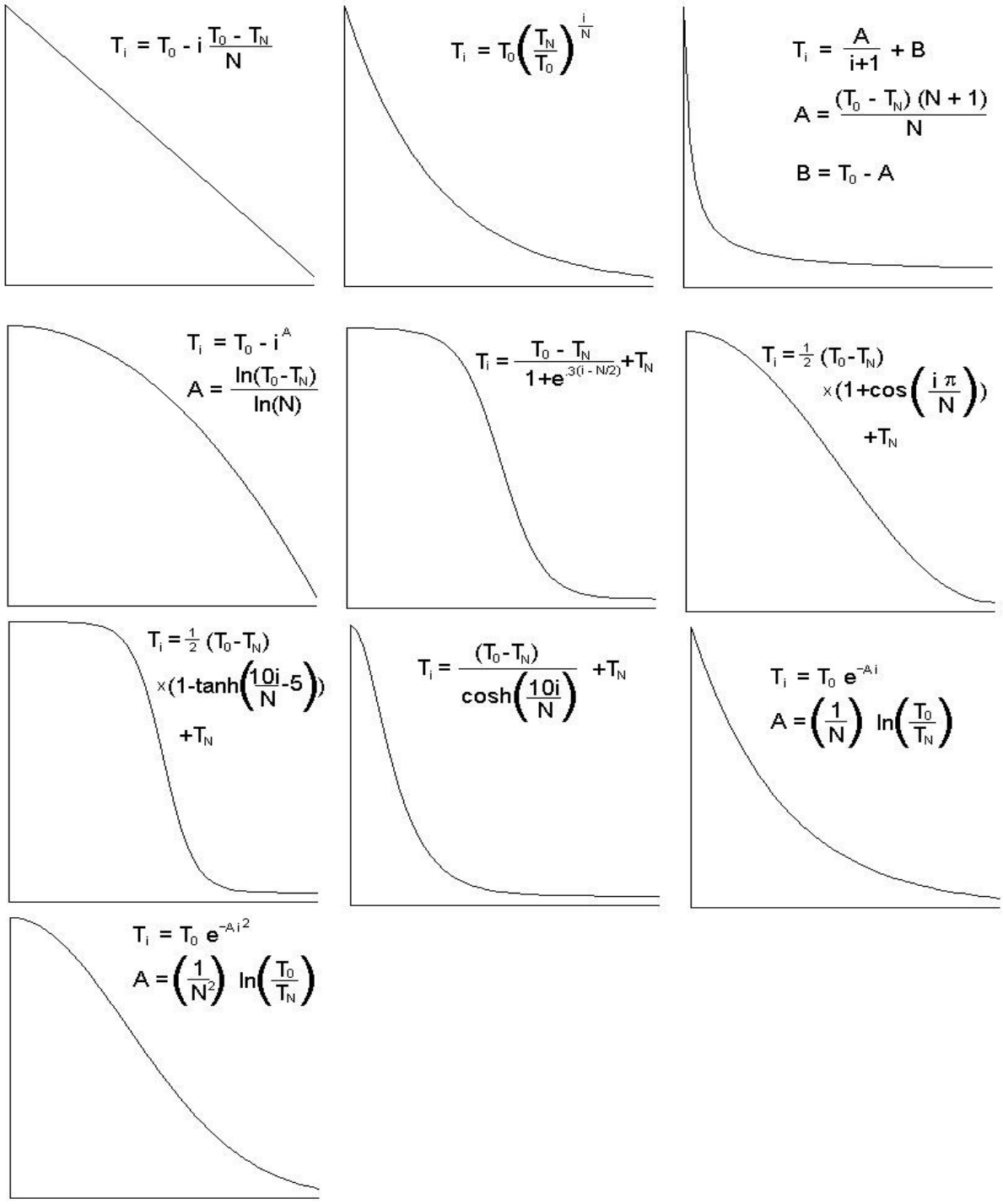
Modeliuojamasis atkaitinimas (Simulated Annealing) yra plataus naudojimo tikimybinis meta-euristinis algoritmas, skirtas rasti gerą globalaus optimumo artinį duotajai funkcijai, kai ieškojimų erdvė yra plati. Jis nepriklausomai buvo pristatytas S. Kirkpatrick, C. D. Gelatt ir M. P. Vecchi 1983 m., bei V. Černý 1985 m. Šis metodas sukurtas kaip Monte Carlo metodo apibendrinimas, skirtas tirti sistemos, sudarytos iš *n* kūnų, būsenos lygtis ir sustingusias būsenas [24]. Modeliuojamojo atkaitinimo metodas yra naudojamas įvairiems kombinatorinio optimizavimo uždaviniams, pavyzdžiui, grupavimo, tvarkaraščių sudarymo, spręsti.

Šio metodo idėja ir pavadinimas kilo iš metalurgijos, kur naudojamas atkaitinimas. Atkaitinimu vadinamas procesas, kai metalai yra kaitinami ir lėtai vėsinami, siekiant gauti didesnius su mažesniais defektais kristalus. Taip gaunamas tvirtas lydinys, kurio vidinė energija yra mažesnė nei buvo prieš tai. Atkaitinimo procese kietasis kūnas yra kaitinamas tol, kol išsilydo ir tuomet lėtai vėsinamas taip, kad kiekvienu laiko momentu sistema būtų beveik termodinaminėje pusiausvyros būsenoje. Kaitinant atomai išsilaisvina iš savo pradinės būsenos (lokalaus pradinės energijos minimumo) ir gali laisvai šokinėti po aukštesnės energijos būsenas. Vykstant lėtam vėsinimui, atomai gali rasti mažesnės vidinės energijos padėtį, todėl sistema tampa labiau sutvarkyta ir artėja prie būsenos, turinčios mažesnę vidinę energiją. Sistemos, kurios yra lėtai vėsinamos (atkaitinamos) gali pasiekti vidinės energijos globalų minimumą, o ne sukietėti lokaliame energijos minimume, nes lėto vėsinimo procesas suteikia pakankamai laiko molekulėms palaipsniui persiorientuoti į mažesnės energijos būsenas.

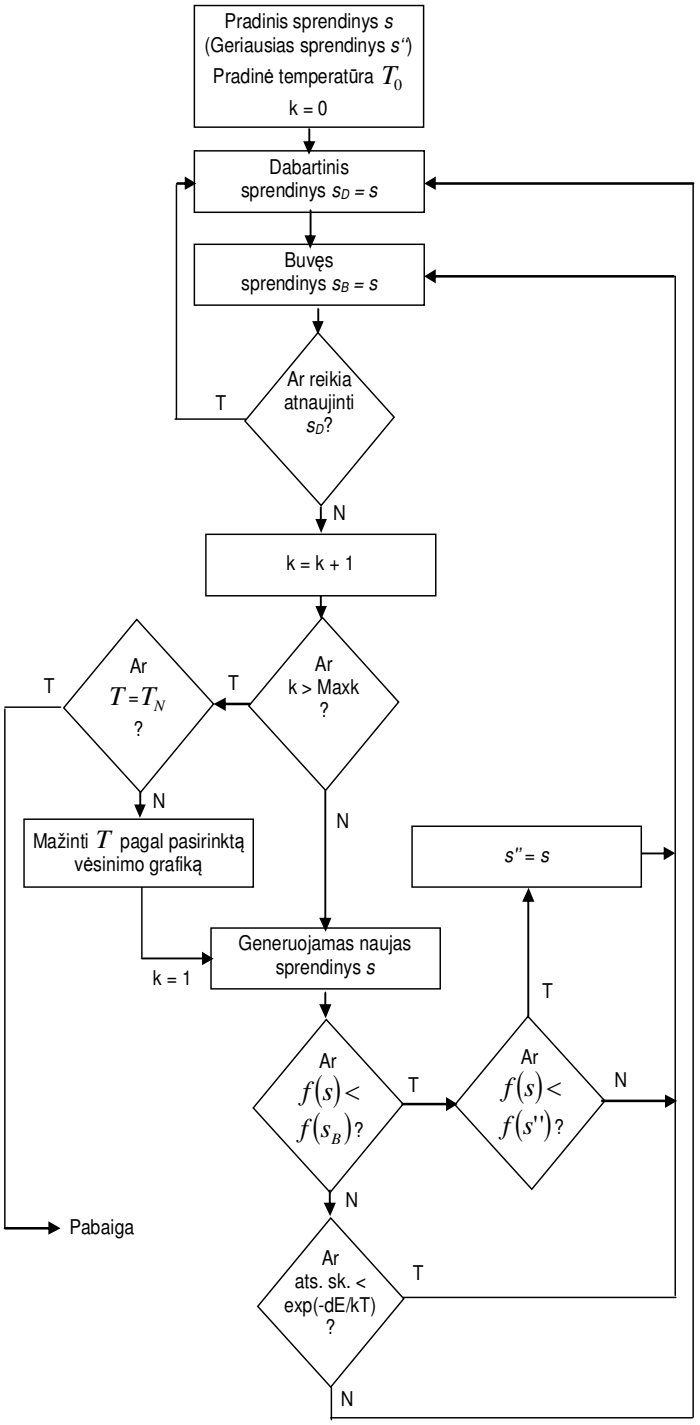
Naudojant modeliuojamo atkaitinimo metodą, uždavinys aprašomas naudojant termodinaminės sistemos, kurios energija *E* ir temperatūra *T*, išraiškas. Energija čia vadinama tikslo funkcijos reikšmė, o temperatūra naudojama tikimybės skaičiavimui. Pakeitus pradinę temperatūrą *T*, pradinis sprendinys keičiamas nauju, skaičiuojamas tikslo funkcijos pokytis *dE*. Jei pokytis yra neigiamas, naujas sprendinys yra priimamas. Jei tikslo funkcijos pokytis yra teigiamas, naujas sprendinys yra priimamas su tam tikra tikimybe, kuri paskaičiuojama naudojant Bolcmano konstantą *k*. Šis procesas kartojamas, kol atliekama pakankamai tyrimų tam tikroje temperatūroje *T*. Kiekvieno tyrimo metu naujas sprendinys yra atsitiktinai parenkamas iš „kaimyninių“ pradiniam sprendinių. Tuomet temperatūra *T* yra sumažinama ir trikdymo procesas vykdomas, kol pasiekiama pabaigos temperatūra, dažniausiai *T=0*. Kai *T* yra didelė, pradinis sprendinys keičiasi beveik atsitiktinai, bet kai *T* artėja prie 0 vis labiau eina žemyn („downhill“). Leidimas kilti aukštyn („uphill“) apsaugo metodą nuo sustojimo lokaliame minimumo taške [24], [9].

Būdas, kaip temperatūra yra mažinama, vadinamas vėsinimo grafiku. Praktikoje dažniausiai naudojami du vėsinimo grafikai: tiesinis ($T_{nauja} = T_{sena} - dT$) ir proporcingas ($T_{nauja} = C \cdot T_{sena}$, $C < 1{,}0$). Daugiau vėsinimo grafikų yra pateikta 1.6 paveiksle.

Modeliuojamojo atkaitinimo algoritmas pateiktas 1.7 paveiksle [19].



1.6 pav. Įvairūs vėsinimo grafikai. T_i **- i-tojo ciklo temperatūra,** $i = \overline{0, N}$ **. Pradinė (**T_0**), galinė (**T_N**) temperatūros, bei žingsnių skaičius N yra pasirenkami**



1.7 pav. Modeliuojamojo atkaitinimo algoritmas

Modeliuojamojo atkaitinimo optimizacija pradedama nuo uždavinio atsitiktinio pradinio sprendinio *s*. Tai yra ir geriausias iki šiol žinomas sprendinys s''. Pradedama skaičiuoti nuo pradinės temperatūros T_0. Sprendinys *s* tampa *Dabartiniu* s_D bei *Buvusiu* s_B sprendiniu. Iteracijų skaičius *k* nustatomas 0.

Padidinus iteracijų skaičių, tikrinama, ar jau pasiektas maksimalus iteracijų skaičius *Maxk*. Jeigu taip, tuomet tikrinama esama temperatūra *T*. Jei ji lygi galutinei temperatūrai T_N, tuomet ieškojimas baigiamas, o *Geriausias* sprendinys s'' ir yra ieškotas sprendinys. Jei dabartinė temperatūra aukštesnė

už galutinę T_N, tuomet ji mažinama pagal pasirinktą vėsinimo grafiką, o iteracijų skaičius nustatomas 0.

Jei iteracijų skaičius nėra didesnis už maksimalų iteracijų skaičių *Maxk*, ar temperatūra buvo sumažinta, *Buvęs* sprendinys s_B yra keičiamas ir sugeneruojamas „kaimyninis“ *Naujas* sprendinys *s*. Jei $f(s)$ (čia f – tikslo funkcija) yra mažesnė nei $f(\mathrm{s_B})$, tikrinama, ar tai yra *Geriausias* iki šiol rastas sprendinys s''. Jeigu taip, jis išsaugomas. Nepriklausomai nuo to, ar tai *Geriausias* sprendinys, ar ne, jis tampa nauju *Buvusiu* sprendiniu. Kai tik *Buvęs* sprendinys s_B yra atnaujinamas, jis tampa *Dabartiniu* sprendiniu s_D.

Jei $f(s)$ yra didesnė nei $f(\mathrm{s_B})$ dydžiu *dE*, tikimybė $e^{-\frac{dE}{kT}}$ (čia *k* –Bolcmano konstanta, *T* – dabartinė temperatūra) yra skaičiuojama. Jei ši tikimybė yra didesnė už atsitiktinį skaičių ($ats.sk. \in [0;1]$), tuomet *Naujas* sprendinys *s* yra priimamas ir tampa *Buvusiu* s_B kitai iteracijai ir *Dabartiniu* sprendiniu s_D. Jei tikimybė yra mažesnė už atsitiktinį skaičių, *Naujas* sprendinys *s* yra atmetamas, o *Dabartinis* s_D */Buvęs* s_B sprendinys lieka tas pats.

1.10 PAIEŠKOS SU DRAUDIMAIS ALGORITMAS

Paieškos su draudimais (Tabu Search) metodas priskiriamas lokaliosios paieškos metodų klasei [24]. Kaip 1986 m. F. Glover apibūdino, paieška su draudimais yra meta-euristika, kuri valdo kitą euristiką. Pagrindinis tikslas yra išvengti ciklų susidarymo, uždraudžiant ar skiriant baudos taškus ėjimams, kurie parenka iš sprendinių aibės sprendinius, artimus neseniai tikrintiems. Paieškos su draudimais metodas iš dalies remiasi žmogaus elgesiu dirbant su atsitiktiniais elementais, kai priešingas veiksmas duoda panašius rezultatus, t.y. naudojamos dirbtinio intelekto savybės. Kaip nurodo F. Glover, polinkis nukrypti nuo sudaryto kurso gali būti atsisakytas kaip klaidingas sprendinio šaltinis, bet tai gali būti kelias į geresnį sprendinį. Paieškos su draudimais metodas veikia aprašytu būdu su sąlyga, kad naujos kryptys parenkamos ne atsitiktinai, o griežtai nustatytu būdu. Jei paieška pradedama nuo to paties pradinio sprendinio, tai ir optimalus sprendinys bus toks pat [10]. Paieška su draudimais vykdoma atsižvelgiant į prielaidą, kad nėra prasmės priimti naujo (blogesnio) sprendinio, nebent tik siekiant išvengti jau ištirto sprendinių kelio. Tai garantuoja, kad bus ištirtos naujos sprendinių aibės sritys išvengiant lokalaus minimumo, ir galų gale bus rastas ieškomas sprendinys.

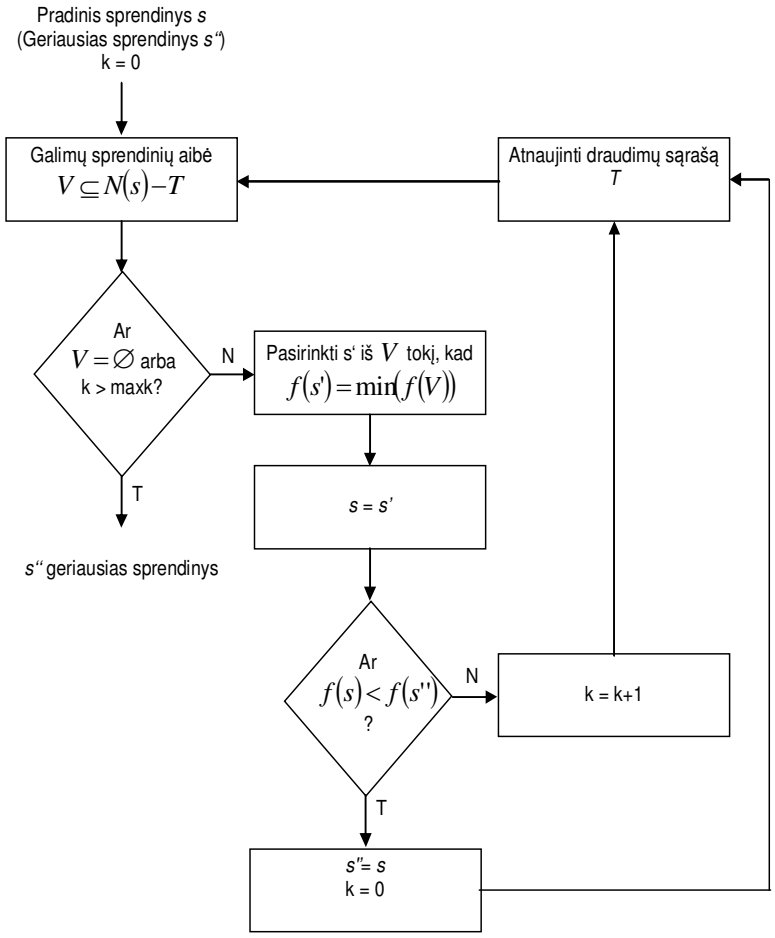
Paieška su draudimais pradedama žingsniuojant nuo pradinio sprendinio link lokalaus minimumo, pasirenkant „kaimyninius“ sprendinius. Kad būtų išvengta vedančių atgal žingsnių, metodas įsimena paskutinius atliktus ėjimus viename ar keliuose draudimų sąrašuose (tabu list), kiekviename žingsnyje keisdamas „kaimyninių“ sprendinių aibę. Sprendinys, įtrauktas į draudimų sąrašą yra uždraustas (Tabu active). Pagrindinis sąrašo tikslas yra ne apsisaugoti nuo veiksmo pakartojimo, o apsidrausti nuo atvirkštinio veiksmo, t.y. grįžimo atgal, ir užtikrinti, kad bus išvengta

sustojimo lokalaus optimumo taške. Draudimo sąrašai savo pobūdžiu yra istoriniai ir formuoja paieškos su draudimais trumpalaikę atmintį (short-term memory). Pagrindinė šios atminties paskirtis yra nustatyti esamo sprendinio „kaimyninių“ sprendinių aibę. Vykdant algoritmą, atminties vaidmuo gali kisti. Iš pradžių grubiai apžiūrima sprendinių erdvė. Tai vadinama sprendinių įvairinimu (diversification). Kai randama patraukli aplinka, paieška sutelkiama optimalaus lokalaus sprendinio radimui. Tai vadinama suintensyvėjimu (intensification).

Daugeliu atvejų įvairios paieškos su draudimais realizacijos skiriasi draudimų atminties dydžiu, kintamumu ir pritaikymu tam tikrai uždavinių sričiai. Paprasčiausiu atveju draudimų sąrašas susideda iš sprendinių, kurie buvo nagrinėjami pastaruoju metu (mažiau nei prieš n žingsnių, čia n – draudimų sąrašo ilgis). Sprendiniai, esantys draudimų sąraše neįtraukiami į „kaimyninių“ sprendinių aibę. Taip pat naudojami draudimų sąrašai, draudžiantys tam tikrus bruožus turinčius sprendinius (pavyzdžiui, keliaujančio pirklio uždavinyje sprendinius, kuriuose yra konkretūs keliai tarp miestų) arba draudžiantys konkrečius veiksmus (pavyzdžiui, keliaujančio pirklio uždavinyje į maršrutą įtrauktas kelias tarp miestų negali būti pašalintas iš maršruto n veiksmų) [20], [21], [24].

Draudimų sąrašai, saugantys sprendinius užima daug vietos. Daug efektyvesni yra tam tikrus sprendinio bruožus saugantys sąrašai, nors dėl to iškyla naujų problemų. Kai nors vienas atributas yra uždraustas, paprastai daugiau nei vienas sprendinys tampa uždraustas. Kai kurie iš šių sprendinių gali būti labai geri ir iki šiol neištirti. Šiai problemai spręsti yra įvedamas siekimo kriterijus (aspiration criteria), kuris suteikia galimybę įtraukti sprendinį į galimų sprendinių aibę. Dažniausiai naudojamas kriterijus, leidžiantis priimti sprendinius, kurie yra geresni nei iki šiol rasti [24].

Paieškos su draudimais algoritmas pateiktas 1.8 paveiksle.



1.8 pav. Paieškos su draudimais algoritmas

Pradinis sprendinys *s* yra pasirenkamas atsitiktinai. Šis sprendinys yra ir geriausias iki šiol rastas sprendinys s''. Jis išsaugomas draudimų sąraše *T*. Iteracijų skaičius *k* nustatomas 0.

Galimų sprendinių aibė *V* – tai nedraudžiamų „kaimyninių“ sprendinių aibė ($N(s)$ - visi „kaimyniniai“ *s* sprendiniai, *T* – uždrausti sprendiniai). „Kaimyniniais“ sprendiniais vadinami tokie sprendiniai, kurie gali būti gauti nežymiai pakeitus esantį sprendinį. Tariama, kad sprendiniai, esantys draudimų sąraše, yra nepasiekiami.

Jei aibė *V* yra tuščia arba pasiektas maksimalus iteracijų skaičius *maxk*, tuomet paieška yra baigiama, o geriausias rastas sprendinys s'' ir yra ieškotas sprendinys. Jei aibė *V* nėra tuščia ir iteracijų skaičius mažesnis už *maxk*, tuomet pasirenkamas „kaimyninis“ sprendinys, kurio tikslo funkcijos reikšmė yra mažiausia.

Rastasis sprendinys tampa dabartiniu sprendiniu. Jei šio sprendinio tikslo funkcijos reikšmė yra mažesnė nei iki šiol rasto geriausio, tuomet jis tampa geriausiu sprendiniu, o iteracijų skaičius *k* nustatomas 0. Jei šio sprendinio tikslo funkcijos reikšmė nėra mažesnė nei iki šiol rasto geriausio sprendinio, tuomet didinamas iteracijų skaičius. Nepriklausomai nuo to, ar tai buvo geriausias sprendinys, ar ne, atnaujinamas draudimų sąrašas ir tęsiama paieška.

1.11 TIRIAMO UŽDAVINIO, NAUDOJAMŲ METODŲ APRAŠYMAS

Šiame darbe nagrinėjamas nuoseklios gamybos (flow shop) tvarkaraščių sudarymo uždavinys. Kiekviena mašina (procesorius, įrenginys) nusakoma mašinos tipu (viename ceche dirba vieno tipo mašinos), sparta ir nedarbingumo intervalais. Nedarbingumo intervalai nusakomi nurodant kalendorių, pagal kurį dirba mašina. Ceche gali būti kelios mašinos. Mašinos sparta nusakoma kanoninės mašinos atžvilgiu. Kiekvieno darbo operacijų skaičius yra vienodas ir lygus cechų skaičiui. Visų darbų operacijų apdorojimo tvarka yra vienoda. Operacija aprašoma mašinos tipu (kokio tipo mašina ją turi apdoroti), jos apdorojimo trukme bei laiku, po kurio gali būti pradėta vykdyti kita operacija (lag time). Taip pat nurodoma, ar operacija apdorojimo metu gali būti nutraukta, ar ne. Nutraukus operaciją, ji toliau apdorojama ta pačia mašina, kai tik mašina vėl pradeda dirbti, t.y. pasibaigus nedarbingumo intervalui. Operacijos apdorojimo trukmė bei laikas, po kurio gali būti pradėta vykdyti kita operacija, nusakomi kanoninės mašinos atžvilgiu. Reikia rasti tokią darbų atlikimo seką kiekviename ceche, kad laikas, reikalingas visiems darbams atlikti, būtų minimalus.

Tyrimui naudojami šakų ir ribų (branch and bound), modeliuojamojo atkaitinimo (simulated annealing) bei paieškos su draudimais (tabu search) algoritmai. Sprendinių aibę sudaro visos galimos darbų sekos kiekviename ceche. Modeliuojamojo atkaitinimo ir paieškos su draudimais algoritmuose naudojamas „kaimyninis“ sprendinys. Šiuo atveju, „kaimyniniu“ sprendiniu vadinama darbų seka, kurioje du darbai tame pačiame ceche yra sukeisti vietomis.

Naudojant šakų ir ribų algoritmą, ieškoma ne tik darbų seka kiekviename ceche, bet ir mašinos, kurios tuos darbus apdoros, kad būtų geriau įvertinti apatiniai rėžiai. Tuo tarpu modeliuojamojo atkaitinimo ir paieškos su draudimais algoritmų atveju ieškoma darbų seka kiekviename ceche, o mašinos parenkamos pagal tai, kuri pirmoji pabaigs apdoroti priskirtą darbą.

Modeliuojamojo atkaitinimo metodas naudoja proporcingą vėsinimo grafiką (žr. 1.9 skyrelį). Temperatūra mažinama tol, kol tampa mažesnė už 10^{-4}.

Paieškos su draudimais metodas naudoja trumpalaikę atmintį. Draudimo sąraše saugomos pastaruoju metu sukeistų darbų poros ir cechas, kuriame tie darbai buvo sukeisti.

Kadangi naudojant modeliuojamojo atkaitinimo ir paieškos su draudimais metodus pradinis sprendinys parenkamas atsitiktinai, bei šie abu metodai priklauso nuo tam tikrų parametrų parinkimo (iteracijų skaičiaus, temperatūros, draudimų sąrašo ilgio ir pan.), todėl rastas sprendinys yra stochastinio pobūdžio. Dėl šios priežasties bus tiriama laiko, reikalingo visiems darbams atlikti, vidurkio bei dispersijos priklausomybė nuo parametrų. Šakų ir ribų algoritmas yra tikslusis ir randa optimalų sprendinį (jei toji šaka nebuvo nukirsta), tačiau didesnės apimties uždaviniams išspręsti gali prireikti labai daug laiko.

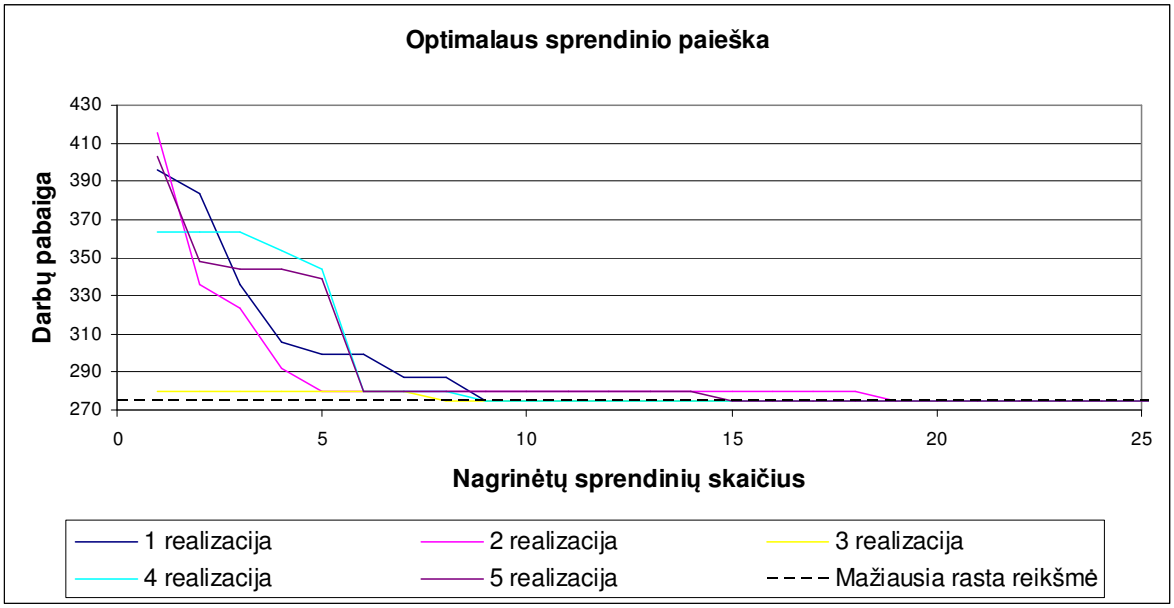
2. TIRIAMOJI DALIS IR REZULTATAI

Naudojamas kompiuteris, kurio resursai Intel(R) Core(TM)2 Duo T7300 @ 2.00 GHz procesorius, 2.00 Gb RAM darbinė atmintis, 32-bit operacinės sistemos tipas.

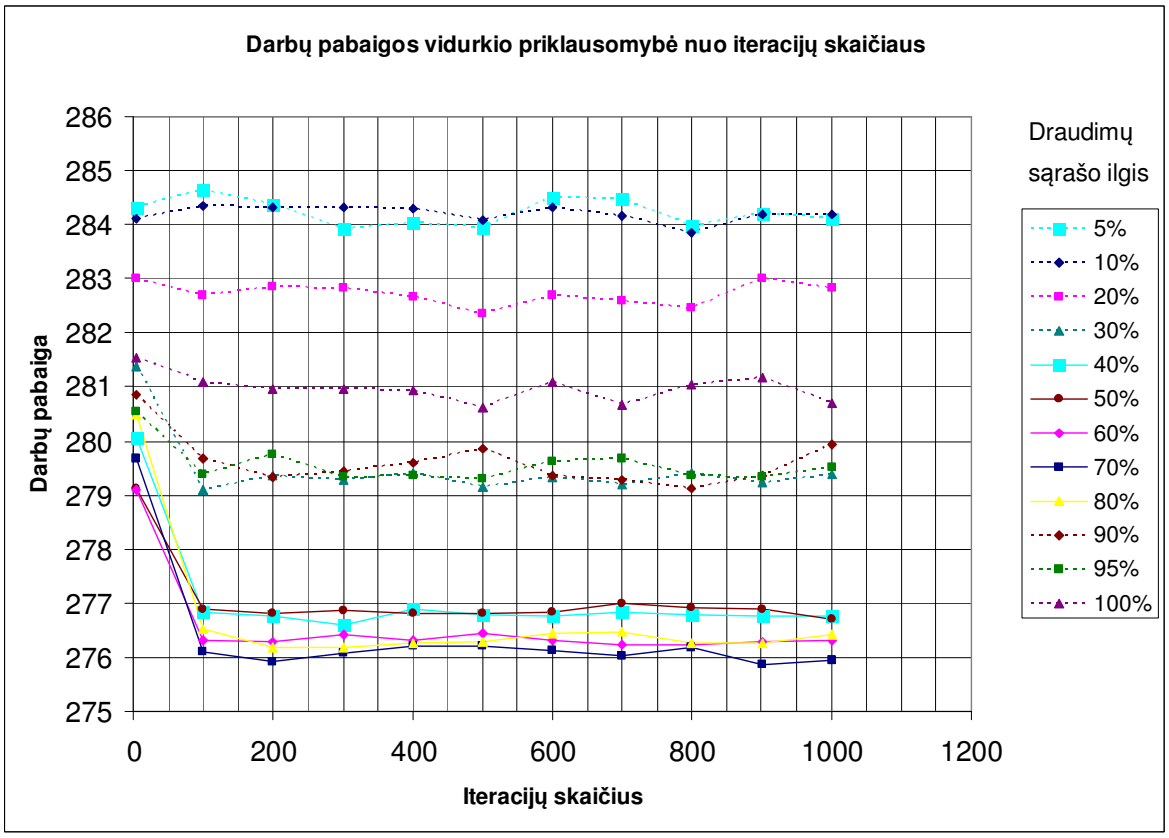
Naudojant sukurtą programą, tiriamas metodų efektyvumas, gaunamų rezultatų priklausomybė nuo parinktų parametrų. Kad gautus rezultatus būtų lengviau analizuoti ir daryti išvadas, nuspręsta, kad duomenų failuose visos mašinos bus skirtingų tipų, vienodos spartos ir dirbs visą laiką, operacijos negalima nutraukti, o kita operacija galės būti pradėta vykdyti tik šiai pasibaigus. Tyrimui naudoti šiai programai pritaikyti duomenų failai iš J E Beasley tinklalapio [11]. Jie pateikiami B priede.

Kadangi modeliuojamojo atkaitinimo ir paieškos su draudimais metoduose pradinis uždavinio sprendinys generuojamas atsitiktinai, o modeliuojamojo atkaitinimo metode dar naudojama ir tikimybė, todėl rastas darbų pabaigos laikas yra atsitiktinis. Dėl šios priežasties tvarkaraštis sudaromas 100 kartų, išsaugomas geriausias rastas rezultatas (mažiausias darbų pabaigos laiko momentas), apskaičiuojama darbų pabaigos empirinis vidurkis, empirinė dispersija ir vidutiniškai per kiek laiko sudaromas vienas tvarkaraštis. Kiekvienam parametrų rinkiniui eksperimentas pakartojamas 5 kartus ir kiekvienam stebimam dydžiui apskaičiuojamas vidurkis. Šakų ir ribų metodas yra tikslus ir nepriklauso nuo jokių parametrų, todėl tvarkaraštis sudaromas tik vieną kartą.

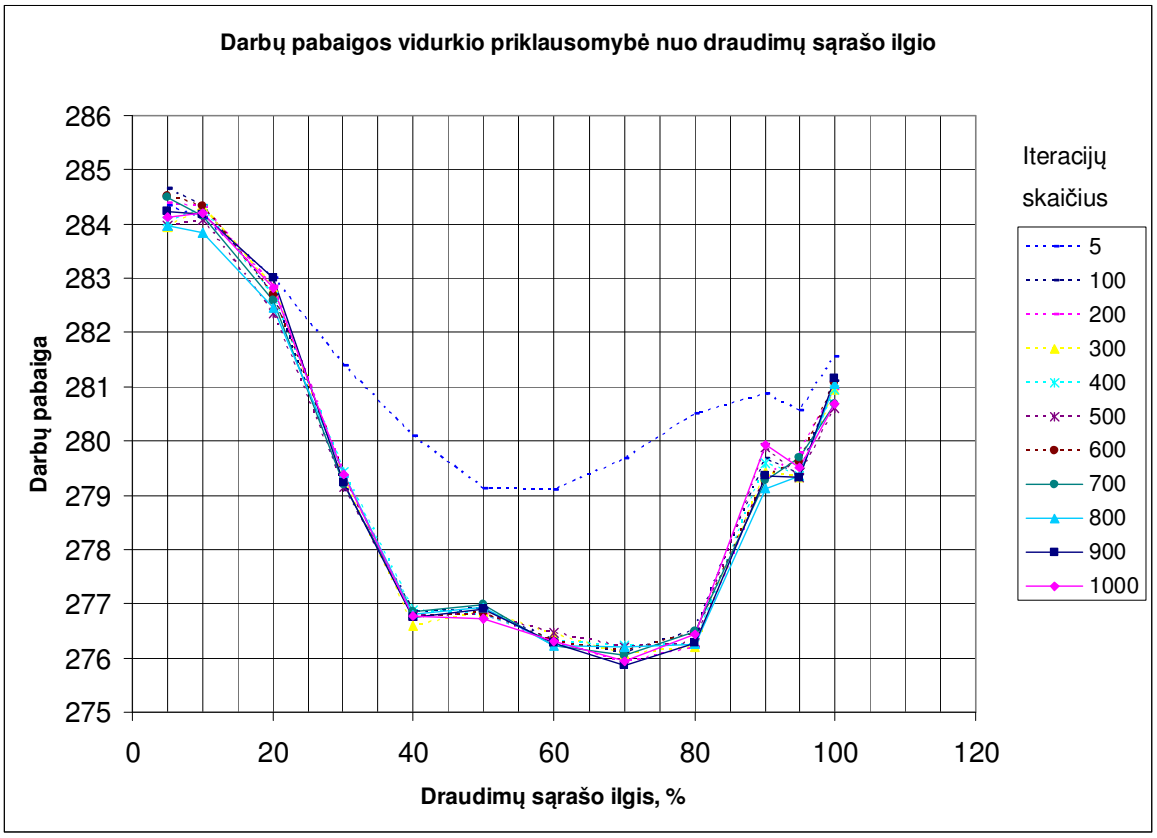
Visais trim metodais trasuojamas failas „darbai3-3.txt“. Rezultatai, gauti naudojant modeliuojamąjį atkaitinimą bei paiešką su draudimais, pateikti A priede atitinkamai A1 ir A5 lentelėse. Naudojant šakų ir ribų metodą gauta tikslo funkcijos reikšmė (darbų pabaigos laiko momentas) yra 275. Tvarkaraštis sudaromas greičiau nei per 1 ms. Kaip matyti iš A1 lentelėje esančių rezultatų, naudojant modeliuojamąjį atkaitinimą optimalus sprendinys randamas maždaug per 60 ms su mažomis parametrų reikšmėmis (iteracijų skaičius – 100, pradinė temperatūra – 5, temperatūros daugiklis – 0,9). Su šiomis parametrų reikšmėmis 2.1 paveiksle pateikta optimalaus sprendinio paieška. Kaip matyti iš A5 lentelės, naudojant paiešką su draudimais rezultatai yra labiau išsibarstę. Darbų pabaigos vidurkio priklausomybė nuo iteracijų skaičiaus bei draudimų sąrašo ilgio pateikta 2.2 ir 2.3 paveiksluose. Geriausi rezultatai gaunami, kai iteracijų skaičius yra 900, o draudimų sąrašo ilgis – 70%. Su šiomis parametrų reikšmėmis 2.4 paveiksle pateikta optimalaus sprendinio paieška.



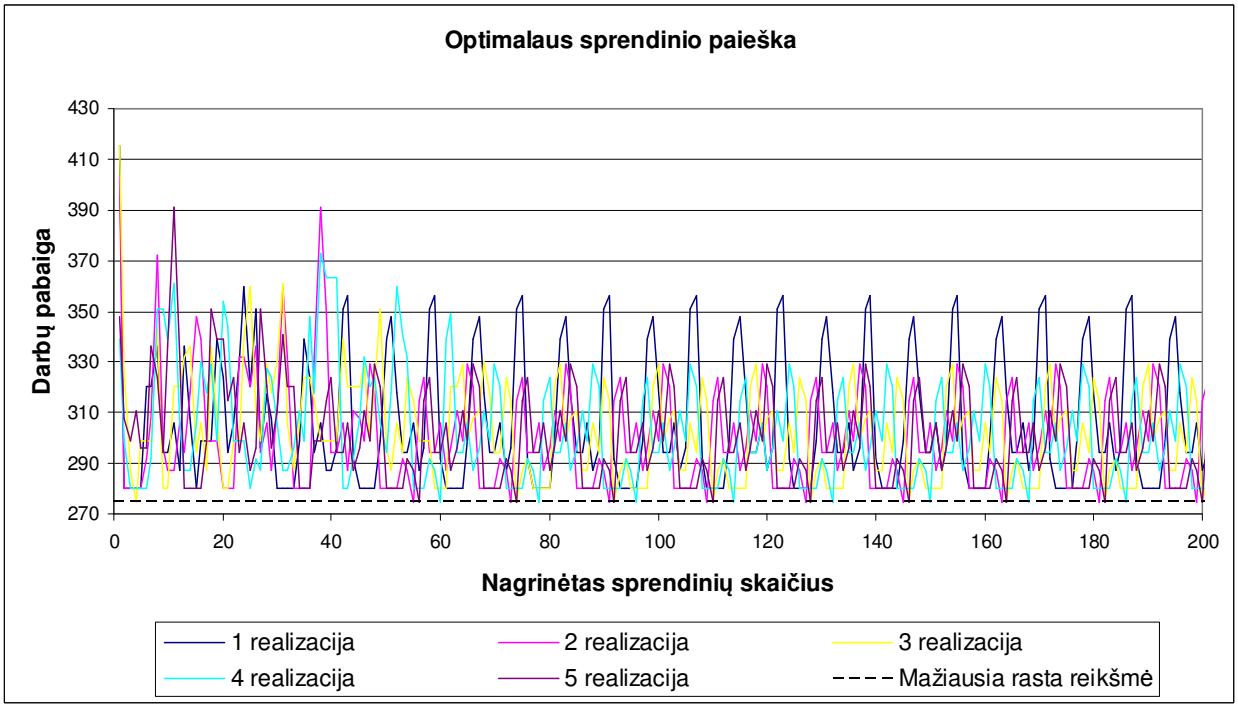
2.1 pav. Optimalaus sprendinio paieška modeliuojamojo atkaitinimo metodu („darbai3-3.txt")



2.2 pav. Darbų pabaigos vidurkio priklausomybė nuo iteracijų skaičiaus („darbai3-3.txt")



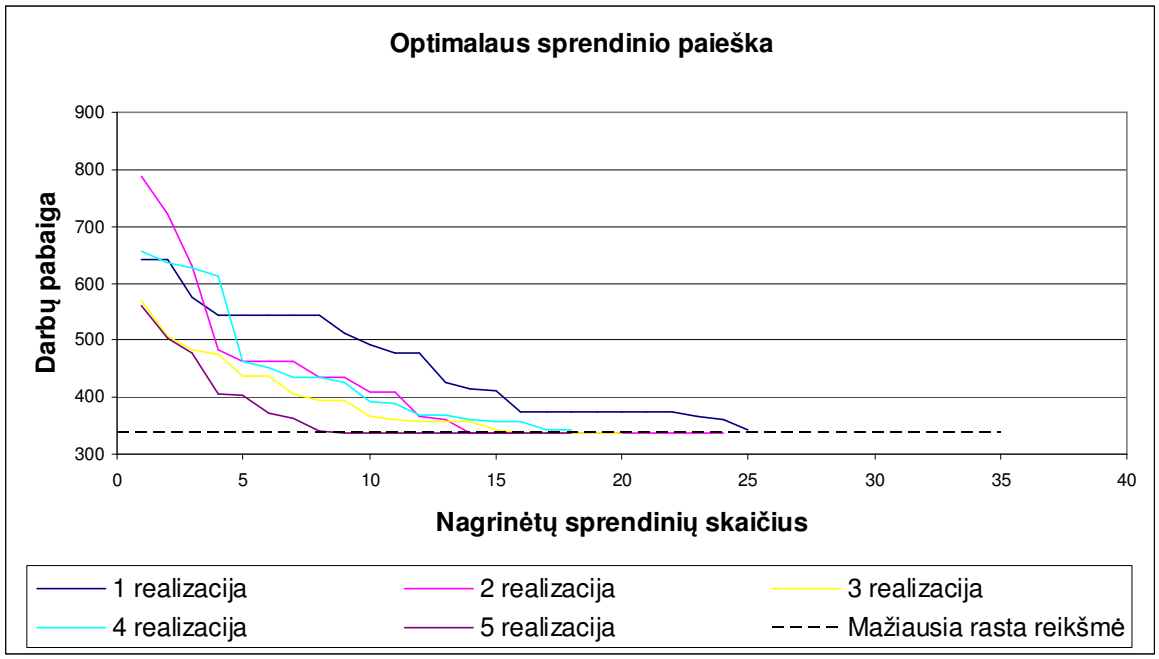
2.3 pav. Darbų pabaigos vidurkio priklausomybė nuo draudimų sąrašo ilgio („darbai3-3.txt")



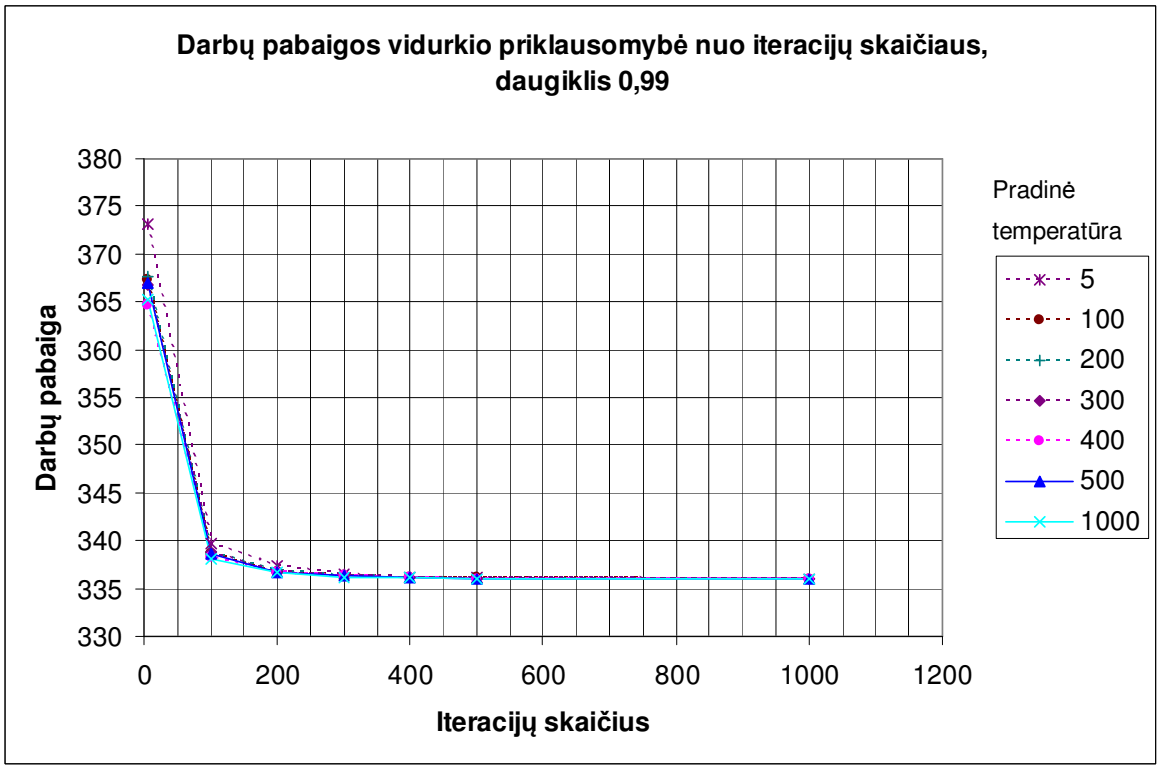
2.4 pav. Optimalaus sprendinio paieška paieškos su draudimais metodu („darbai3-3.txt")

Trasuojamas failas „darbai4-4.txt". Naudojant šakų ir ribų metodą gauta tikslo funkcijos reikšmė yra 336. Tvarkaraštis sudaromas maždaug per 62 ms. Rezultatai, gauti naudojant modeliuojamąjį

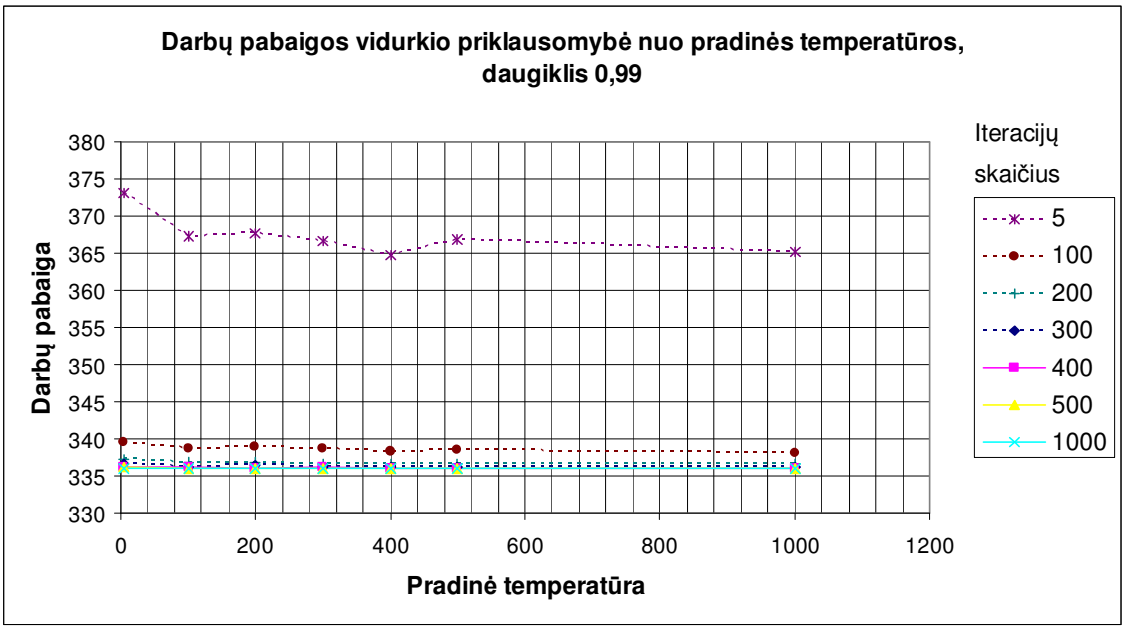
atkaitinimą, pateikti A2 lentelėje. Kaip matyti iš šios lentelės, naudojant modeliuojamąjį atkaitinimą rezultatai yra labiau išsibarstę, o optimalus sprendinys visada gaunamas, kai iteracijų skaičius yra 1000, pradinė temperatūra – 100, temperatūros daugiklis – 0,99. Su šiomis parametrų reikšmėmis 2.5 paveiksle pateikta optimalaus sprendinio paieška. Darbų pabaigos vidurkio priklausomybė nuo iteracijų skaičiaus bei pradinės temperatūros pateikiama 2.6 ir 2.7 paveiksluose.



2.5 pav. Optimalaus sprendinio paieška modeliuojamojo atkaitinimo metodu („darbai4-4.txt")

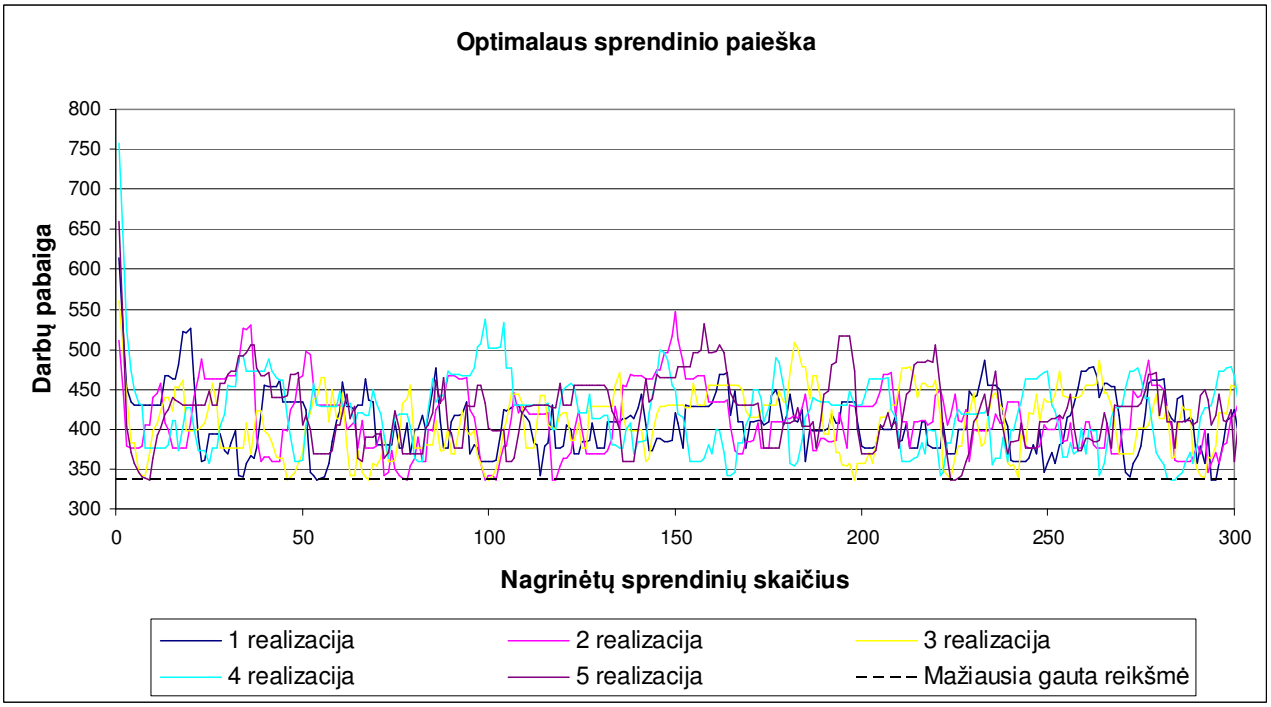


2.6 pav. Darbų pabaigos vidurkio priklausomybė nuo iteracijų skaičiaus („darbai4-4.txt")



2.7 pav. Darbų pabaigos vidurkio priklausomybė nuo pradinės temperatūros („darbai4-4.txt")

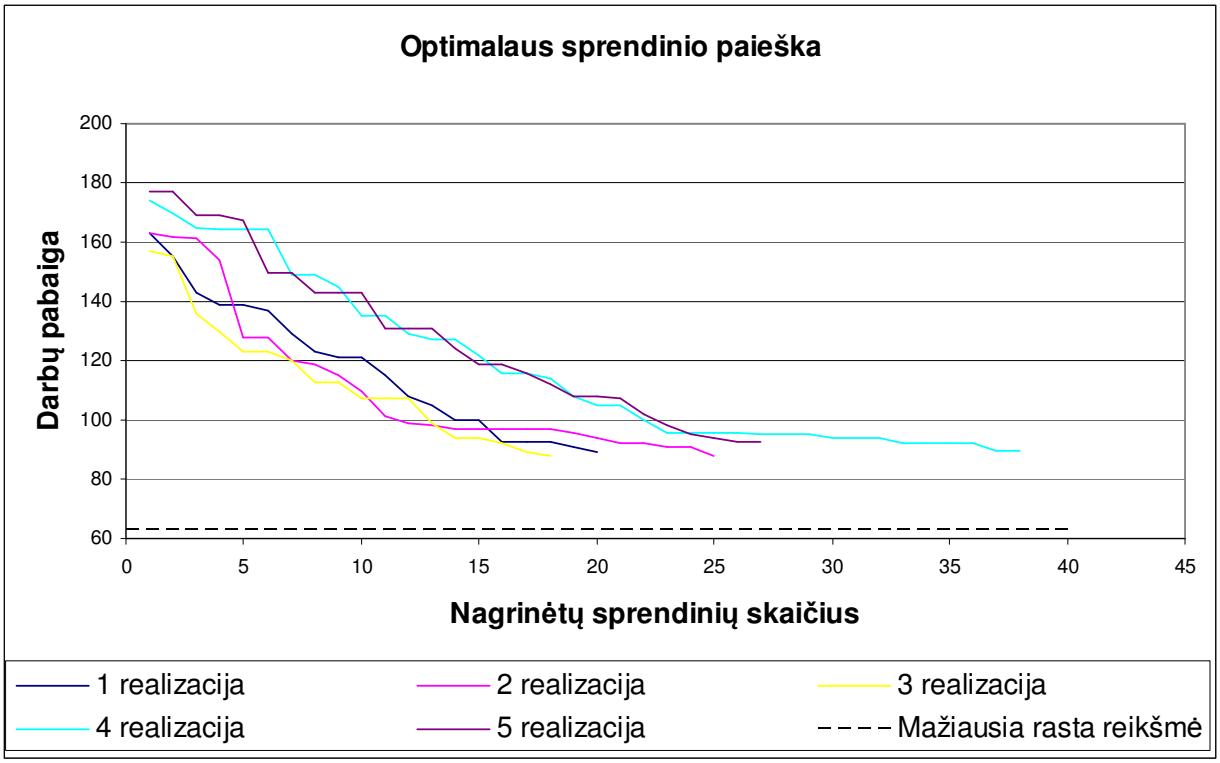
Paieškos su draudimais rezultatai pateikti A6 lentelėje. Darbų pabaigos priklausomybė nuo iteracijų skaičiaus bei nuo draudimų sąrašo ilgio pateikta A priedo A.5 ir A.6 paveiksluose. Geriausi rezultatai gauti, kai iteracijų skaičius yra 1000, o draudimų sąrašo ilgis – 40%. Su šiomis parametrų reikšmėmis 2.8 paveiksle pateikta optimalaus sprendinio paieška.



2.8 pav. Optimalaus sprendinio paieška paieškos su draudimais metodu („darbai4-4.txt")

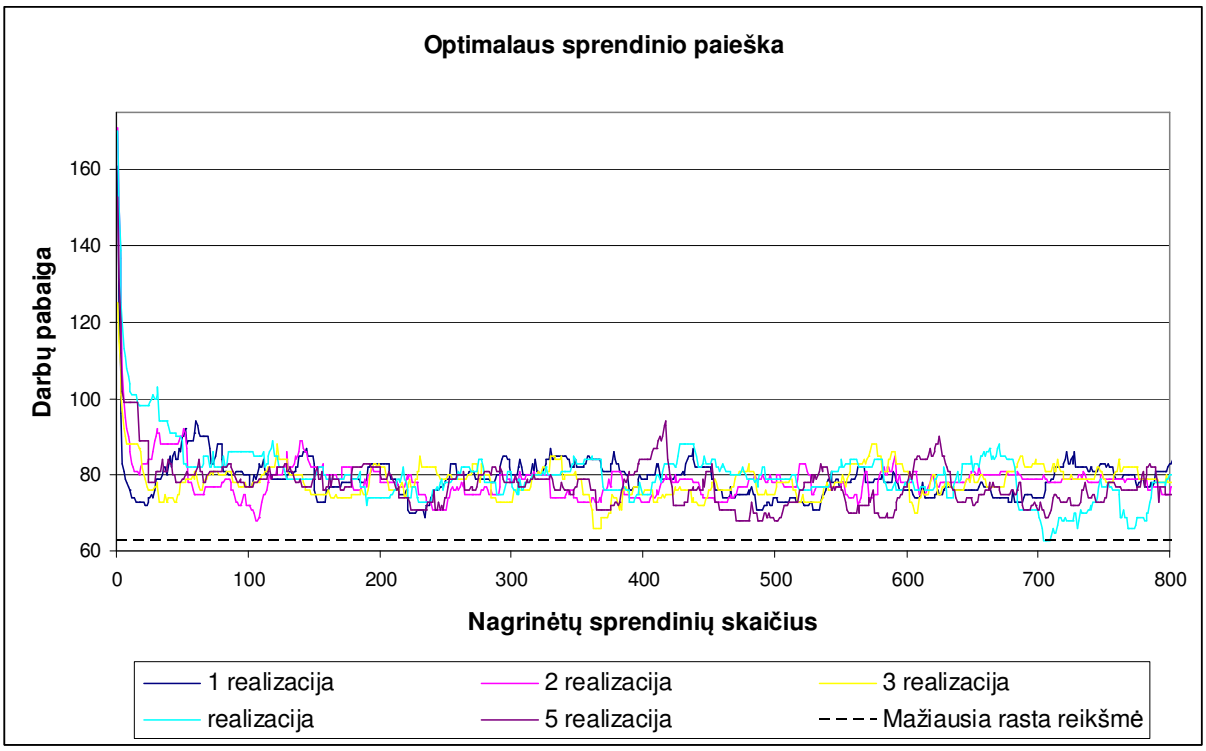
Trasuojamas failas „darbai6-6.txt". Naudojant šakų ir ribų metodą gauta tikslo funkcijos reikšmė yra 63. Tvarkaraštis buvo sudarytas maždaug per 2 paras. Rezultatai, gauti naudojant modeliuojamąjį atkaitinimą, pateikti A3 lentelėje. Kaip matyti iš šios lentelės, rezultatai yra išsibarstę, darbų pabaigos

vidurkis yra tolokai nuo rastos mažiausios reikšmės. Geriausi rezultatai gaunami, kai iteracijų skaičius yra 1000, pradinė temperatūra – 1000, temperatūros daugiklis – 0,99. Su šiomis parametrų reikšmėmis 2.9 paveiksle pateikta optimalaus sprendinio paieška. Darbų pabaigos vidurkio priklausomybė nuo iteracijų skaičiaus bei pradinės temperatūros pateikiama A priedo A.1 ir A.2 paveiksluose.



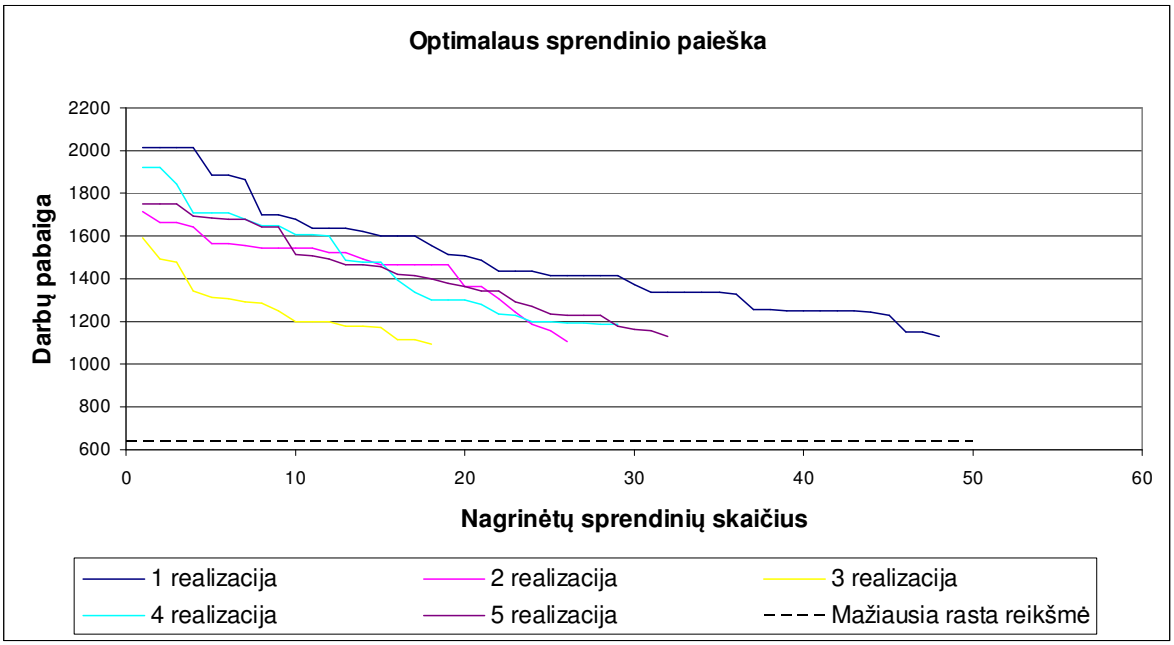
2.9 pav. Optimalaus sprendinio paieška modeliuojamojo atkaitinimo metodu („darbai6-6.txt")

Paieškos su draudimais rezultatai pateikti A7 lentelėje. Darbų pabaigos priklausomybė nuo iteracijų skaičiaus bei nuo draudimų sąrašo ilgio pateikta A priede A.7 ir A.8 paveiksluose. Geriausi rezultatai gauti, kai iteracijų skaičius yra 1000, o draudimų sąrašo ilgis – 30%. Su šiomis parametrų reikšmėmis 2.10 paveiksle pateikta optimalaus sprendinio paieška.



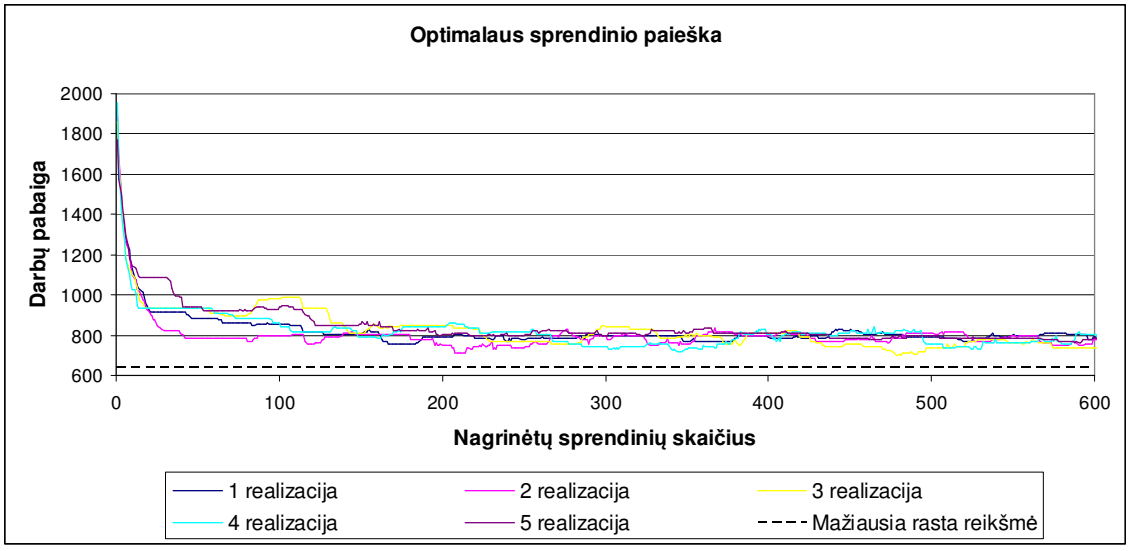
2.10 pav. Optimalaus sprendinio paieška paieškos su draudimais metodu („darbai6-6.txt")

Trasuojamas failas „darbai10-5.txt". Kadangi šis uždavinys yra dar sudėtingesnis, o literatūroje nerekomenduojama sudėtingų uždavinių spręsti naudojant šakų ir ribų metodą, todėl nurodytas failas šiuo metodu nebuvo trasuojamas. Rezultatai, gauti naudojant modeliuojamąjį atkaitinimą, pateikti A4 lentelėje. Darbų pabaigos vidurkis yra tolokai nuo rastos mažiausios reikšmės (637). Geriausi rezultatai gaunami, kai iteracijų skaičius yra 1000, pradinė temperatūra – 1000, temperatūros daugiklis – 0,99. Su šiomis parametrų reikšmėmis 2.11 paveiksle pateikta optimalaus sprendinio paieška. Darbų pabaigos vidurkio priklausomybė nuo iteracijų skaičiaus bei pradinės temperatūros pateikiama A priede A.3 ir A.4 paveiksluose.



2.11 pav. Optimalaus sprendinio paieška modeliuojamojo atkaitinimo metodu („darbai10-5.txt")

Paieškos su draudimais rezultatai pateikti A8 lentelėje. Darbų pabaigos priklausomybė nuo iteracijų skaičiaus bei nuo draudimų sąrašo ilgio pateikta A.9 ir A.10 paveiksluose. Geriausi rezultatai gauti, kai iteracijų skaičius yra 1000, o draudimų sąrašo ilgis – 30%. Su šiomis parametrų reikšmėmis 2.12 paveiksle pateikta optimalaus sprendinio paieška.

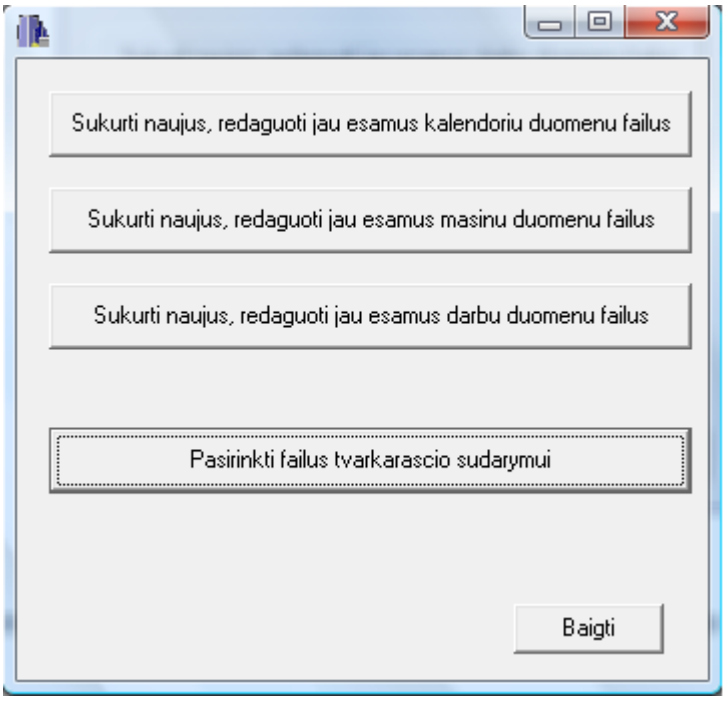


2.12 pav. Optimalaus sprendinio paieška paieškos su draudimais metodu („darbai10-5.txt")

3. PROGRAMINĖ REALIZACIJA IR INSTRUKCIJA VARTOTOJUI

Programa parašyta „Borland C++ 5.0 Builder“ programavimo kalba. Ji skirta nuoseklios gamybos tvarkaraščiams sudaryti, naudojant šakų ir ribų (Branch and Bound), modeliuojamojo atkaitinimo (Simulated Annealing) ir paieškos su draudimais (Tabu Search) metodus.

Paleidus programą, matomas 3.1 paveiksle pateiktas langas su galimais veiksmais.



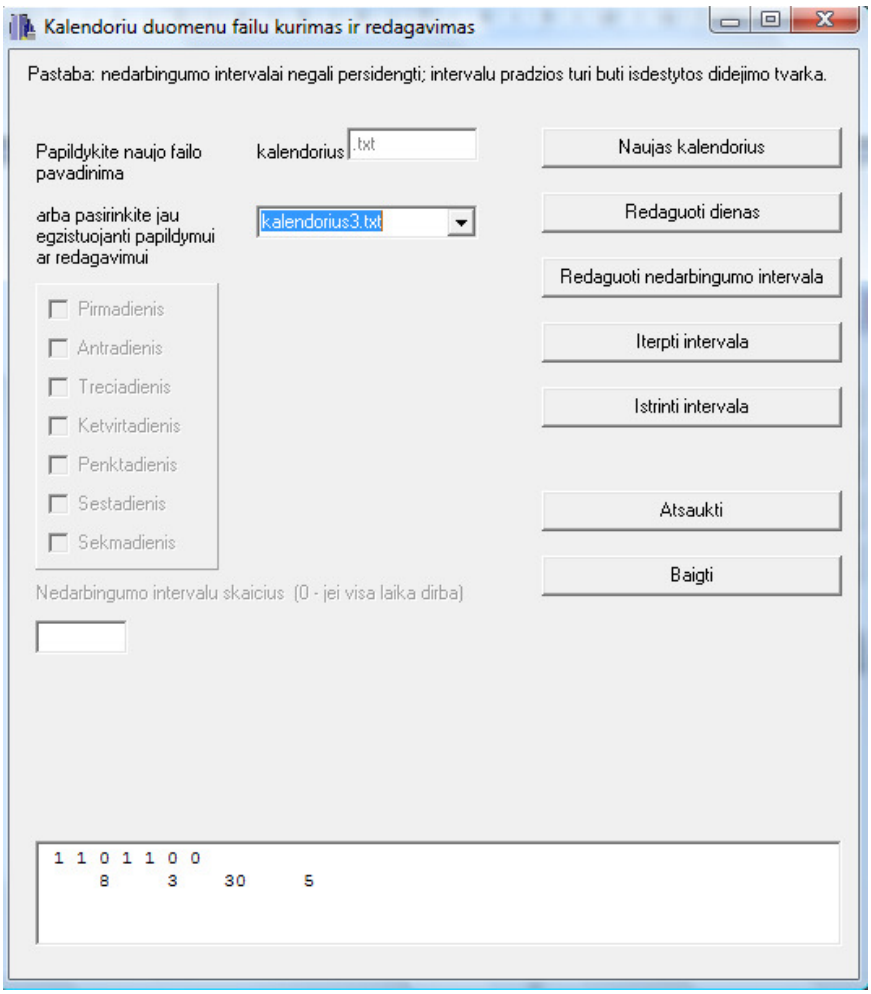
3.1 pav. Programos pradžios langas

Naudojantis programa galima sukurti naujus bei redaguoti jau esamus kalendorių, mašinų ir darbų duomenų failus.

Mygtukų paskirtis:

- „Sukurti naujus, redaguoti jau esamus kalendorių duomenų failus“ – iškviečiamas langas kalendoriaus duomenų failo kūrimui ir redagavimui;
- „Sukurti naujus, redaguoti jau esamus mašinų duomenų failus“ – iškviečiamas langas mašinų duomenų failų kūrimui ir redagavimui;
- „Sukurti naujus, redaguoti jau esamus darbų duomenų failus“ – iškviečiamas langas darbų duomenų failų kūrimui ir redagavimui;
- „Pasirinkti failus tvarkaraščio sudarymui“ – iškviečia langą duomenų failui pasirinkti, paskui – tvarkaraščiui sudaryti;
- „Baigti“ – uždaromas langas.

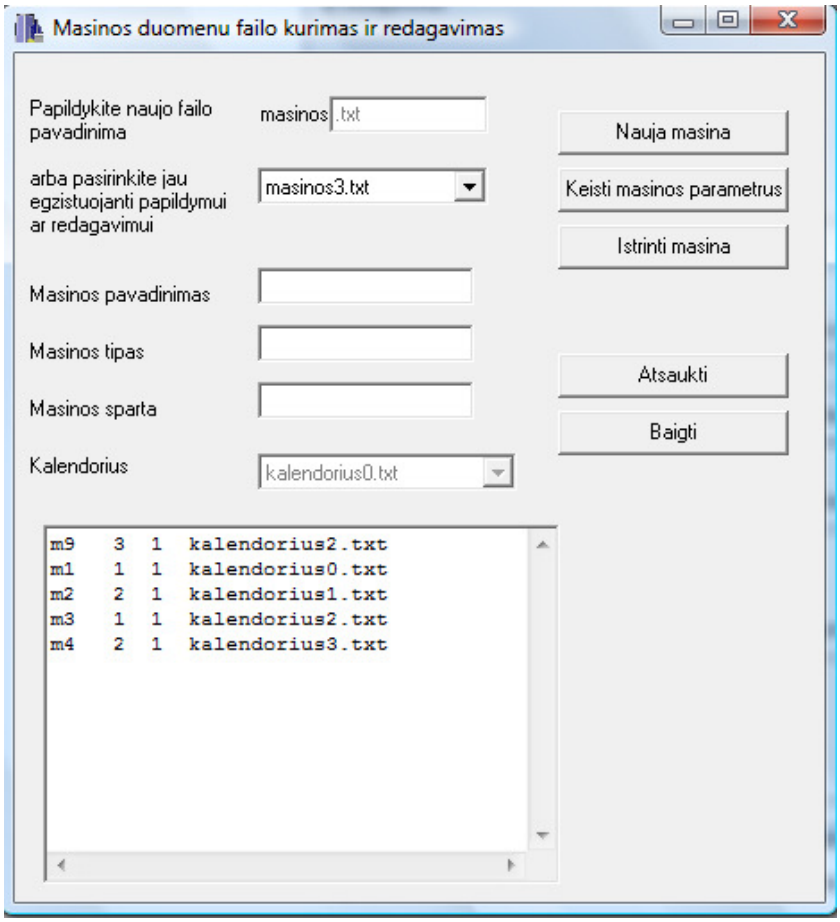
Programos langai atitinkamai pateikiami 3.2, 3.3 ir 3.4 paveiksluose.



3.2 pav. Kalendoriaus duomenų failų kūrimo ir redagavimo langas

Mygtukų paskirtis:

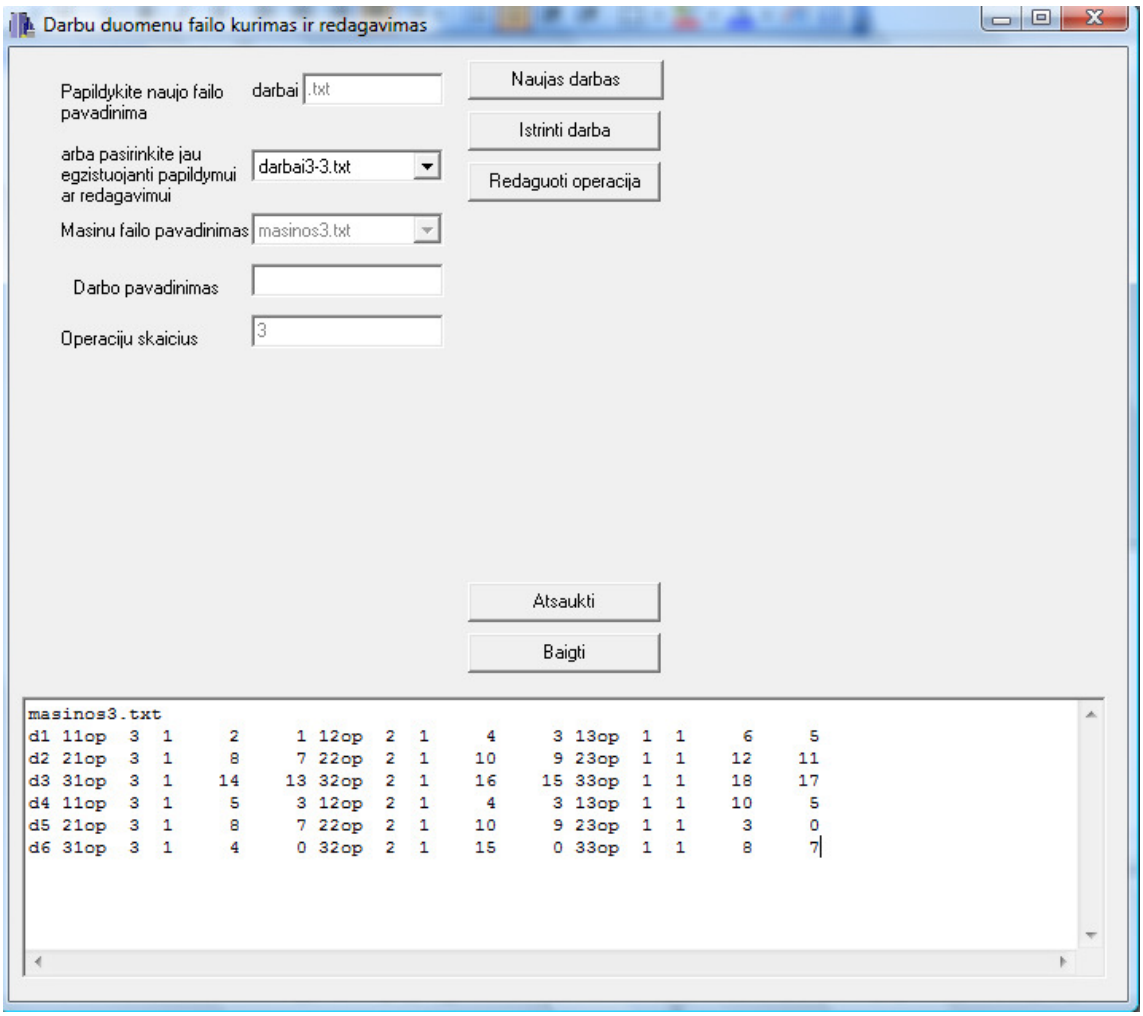
- „Naujas kalendorius“ – kuriamas naujas kalendorius. Jei keičiamas naujo failo pavadinimui skirtas laukelis, sukuriamas naujas failas, jei ne – tuomet kalendorius išsaugomas pasirinktame faile. Įvedama informacija apie darbo dienas ir nedarbingumo intervalus išsaugoma;
- „Redaguoti dienas“ – keičiamos darbo dienos;
- „Redaguoti nedarbingumo intervalą“ – redaguojamas nedarbingumo intervalas;
- „Įterpti intervalą“ – pasirinktame faile nurodytoje vietoje įterpiamas naujas intervalas;
- „Ištrinti intervalą“ – iš pasirinkto failo ištrinamas pageidaujamas intervalas;
- „Atšaukti“ – nutraukiami vykdomi veiksmai;
- „Baigti“ – uždaromas langas.



3.3 pav. Mašinų duomenų failų kūrimo ir redagavimo langas

Mygtukų paskirtis:

- „Nauja mašina“ – paruošiamas papildymui pasirinktas failas arba, jei keičiamas naujo failo pavadinimui skirtas laukelis, sukuriamas naujas failas. Įvedus reikiamą informaciją apie mašiną, ją galima išsaugoti;
- „Keisti mašinos parametrus“ – pasirinktame faile, jei galima, keičiami pageidaujamos mašinos parametrai;
- „Ištrinti mašiną“ – iš pasirinkto failo, jei galima, ištrinama pageidaujama mašina;
- „Atšaukti“ – nutraukiami vykdomi veiksmai;
- „Baigti“ – uždaromas langas.



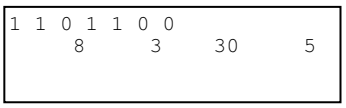
3.4 pav. Darbų duomenų failų kūrimo ir redagavimo langas

Mygtukų paskirtis:

- „Naujas darbas“ – paruošiamas papildymui pasirinktas failas arba, jei keičiamas naujo failo pavadinimui skirtas laukelis, sukuriamas naujas failas. Įvedama informacija apie darbą ir operacijas išsaugoma;
- „Ištrinti darbą“ – iš pasirinkto failo ištrinamas pageidaujamas darbas;
- „Redaguoti operaciją“ – pasirinktame faile keičiama informacija apie nurodyto darbo pageidaujamą operaciją;
- „Atšaukti“ – nutraukiami vykdomi veiksmai;
- „Baigti“ – uždaromas langas.

Kalendoriai aprašomi failuose „kalendorius*.txt“ (čia * reiškia bet kokį skaičių). Kiekviena mašina dirba savaitės periodu, t.y. dirba arba nedirba tomis pačiomis dienomis kiekvieną savaitę, o kiekvieną darbo dieną nedirba tuo pačiu laiku, pvz. naktį, per pietų pertrauką, profilaktinio patikrinimo metu ir pan. Pirmoji failo eilutė nurodo darbo (1) ir nedarbo (0) dienas, o antroje aprašomi nedarbingumo intervalai – intervalo pradžia ir nedarbo trukmė. Nedarbingumo intervalai negali kirstis

ir turi būti išrikiuoti pagal pradžią didėjimo tvarka. Laiko vienetas yra 10 minučių. Valandoje yra 60 minučių, paroje – 24 valandos, savaitėje 7 dienos. Failo pavyzdys pateiktas 3.5 paveiksle.

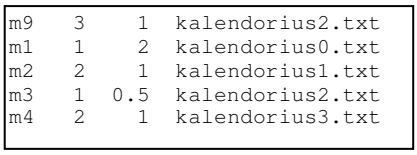


3.5 pav. Kalendoriaus duomenų failo pavyzdys

Mašinos aprašomos failuose „masinos*.txt“ (čia * reiškia bet kokį skaičių). Kiekviena eilutė reprezentuoja mašiną. Eilutėje turi būti pateikta tokia informacija:

- mašinos pavadinimas,
- tipas,
- vykdymo sparta,
- kalendoriaus, pagal kurį dirba mašina, failo pavadinimas.

Duomenų failo pavyzdys pateikiamas 3.6 paveiksle.



3.6 pav. Kalendoriaus duomenų failo pavyzdys

Darbai aprašomi failuose „darbai*.txt“ (čia * reiškia bet kokį skaičių). Pirmoje eilutėje nurodomas mašinų, reikalingų darbams atlikti, failo pavadinimas. Kiekviena kita eilutė reprezentuoja darbą. Visi darbai atliekami tokia pačia tvarka ir turi vienodą skaičių operacijų. Eilutėje turi būti bent vienas įrašas, kuris susideda iš:

- darbo pavadinimo,
- operacijos pavadinimo,
- operacijai atlikti reikalingo mašinos tipo,
- indekso, nurodančio, ar operacija apdorojimo metu gali būti nutraukta (1 – gali, 0 – ne);
- operacijos apdorojimo trukmės,
- laiko, kada gali būti pradėta vykdyti kita operacija (jei kita operacija gali būti pradėta tik pirmajai pasibaigus, rašomas 0).

Duomenų failo pavyzdys pateikiamas 3.7 paveiksle.

```
masinos3.txt
d1 11op  3  1     2     1 12op  2  1     4     3 13op  1  1     6     5
d2 21op  3  0     8     7 22op  2  1    10     9 23op  1  1    12    11
d3 31op  3  1    14    13 32op  2  0    16    15 33op  1  0    18    17
d4 11op  3  1     5     3 12op  2  1     4     0 13op  1  0    10     5
d5 21op  3  0     8     7 22op  2  1    10     9 23op  1  0     3     0
d6 31op  3  1     4     0 32op  2  0    15     0 33op  1  1     8     7
```

3.7 pav. Darbų duomenų failo pavyzdys

Pasirinkus darbų duomenų failą (mašinų failas nurodomas darbų faile) bei pasirinkus norimą metodą ir, jei reikia, parametrus, galima sudaryti tvarkaraštį. Jis sudaromas tokiu būdu: pagal sudarytą seką imamas darbas, kiekvienai jo operacijai atlikti surandama reikalingo tipo mašina. Jei yra kelios vienodo tipo mašinos ir nenurodyta, kuri mašina turi apdoroti operaciją, tuomet operacija priskiriama tai mašinai, kuri greičiau ją apdoros. Jei operacijos apdorojimo pabaigos sutampa, tuomet ji priskiriama pirmajai mašinai. Sudarytas tvarkaraštis vaizduojamas Ganto diagrama lange. Vartotojas pats gali pasirinkti rodomų ekrane mašinų skaičių bei pageidaujamą matyti laiko intervalą. Tvarkaraščio sudarymo langas pateikiamas 3.8 paveiksle.

3.8 pav. Tvarkaraščių sudarymo langas

Mygtukų paskirtis:

- „Sudaryti tvarkaraštį“ – tvarkaraštis sudaromas naudojant pasirinktą metodą su nurodytais parametrais, jei parametrus galima pasirinkti;
- „Atlikti analizę“ – sudaromas nurodytas skaičius tvarkaraščių, lange pateikiami gauti rezultatai;
- „Baigti“ – uždaromas langas.

Sudarytas tvarkaraštis išsaugomas faile „Tvarkarastis.txt“. Faile pateikiama darbų atlikimo seka kiekviename ceche (etape), darbų pabaiga, bei visų mašinų. Kiekviena eilutė reprezentuoja mašinos apkrautumą. Eilutė susideda iš:

- mašinos pavadinimo,
- darbo pavadinimo,
- operacijos pavadinimo,
- operacijos apdorojimo pradžios momento,
- operacijos apdorojimo pabaigos momento.

Tvarkaraščio failo pavyzdys pateikiamas 3.9 paveiksle.

```
Darbu atlikimo seka:
0 5 2 3 4 1  0 5 3 2 1 4  0 5 3 2 1 4
Darbu pabaiga: 75
m1 d1 13op 4 10 d6 33op 26 34 d4 13op 37 47 d3 33op 52 70 d5 23op 72 75
m3 d2 23op 62 74
m2
m4 d1 12op 1 5 d6 32op 11 26 d4 12op 28 37 d3 32op 37 53 d2 22op 53 63 d5 22op 63 73
m9 d1 11op 0 2 d6 31op 2 9 d3 31op 9 23 d4 11op 23 30 d5 21op 30 38 d2 21op 38 46
```

3.9 pav. Sudaryto tvarkaraščio failo pavyzdys

Programoje informacijai apie kalendorius, mašinas, darbus bei tvarkaraštį saugoti naudojami vienmačiai dinaminiai masyvai bei vektoriai.

Pasirinkus norimą metodą ir parametrus, galima sudaryti norimą skaičių tvarkaraščių analizei atlikti. Programos lange parodoma darbų atlikimo pabaiga, dažnis, apskaičiuotas empirinis vidurkis ir empirinė dispersija bei kiek laiko vidutiniškai reikėjo vienam tvarkaraščiui sudaryti. Visa informacija išsaugoma faile „Analizes rezultatai.txt“. Šio failo pavyzdys pateikiamas 3.10 paveiksle.

```
Tvarkarastis sudarytas paieskos su draudimais metodu 100 kartu.

Darbu pabaiga  Santykinis daznis
   73.00         0.44
   74.00         0.21
   75.00         0.18
   76.00         0.03
   77.00         0.13
   81.00         0.01

Vidurkis 74.26
Dispersija 2.33
Tvarkarastis sudaromas vid. per 134.47 ms
```

3.10 pav. Analizės rezultatų failo pavyzdys

Programos aprašymas pateiktas C priede.

DISKUSIJA

Sprendžiant „darbai3-3.txt“ faile aprašytą uždavinį šakų ir ribų bei modeliuojamojo atkaitinimo metodais (iteracijų skaičius – 100, pradinė temperatūra – 5, temperatūros daugiklis – 0,9) greitai gaunamas optimalus sprendinys. Naudojant paiešką su draudimais (iteracijų skaičius – 900, draudimų sąrašo ilgis – 70%) optimalus sprendinys gaunamas ne visada, tačiau vienam tvarkaraščiui sudaryti reikia mažesnių laiko sąnaudų nei naudojant modeliuojamąjį atkaitinimą.

Sprendžiant faile „darbai4-4.txt“ aprašytą uždavinį šakų ir ribų metodu pakankamai greitai (apie 62 ms) gaunamas optimalus sprendinys. Naudojant modeliuojamojo atkaitinimo metodą (iteracijų skaičius – 1000, pradinė temperatūra – 100, temperatūros daugiklis – 0,99) taip pat gaunamas optimalus sprendinys, tačiau vienam tvarkaraščiui sudaryti reikia daug didesnių laiko sąnaudų nei naudojant paiešką su draudimais (iteracijų skaičius – 1000, draudimų sąrašo ilgis – 40%). Pasirinkus pastarąjį metodą optimalus sprendinys gaunamas ne visada.

Faile “darbai6-6.txt” aprašytas uždavinys yra sudėtingesnis nei anksčiau spręstieji, todėl jį sprendžiant šakų ir ribų metodu reikia pakankamai daug laiko (apie 2 paras) optimaliam sprendiniui rasti. Naudojant modeliuojamojo atkaitinimo metodą (iteracijų skaičius – 1000, pradinė temperatūra – 1000, temperatūros daugiklis – 0,99) darbų pabaigos vidurkis yra tolokai nuo mažiausios rastos reikšmės. Tuo tarpu naudojant paiešką su draudimais (iteracijų skaičius – 1000, draudimų sąrašo ilgis – 30%) gaunami geresni rezultatai, ir vienam tvarkaraščiui sudaryti reikia mažiau laiko nei modeliuojamojo atkaitinimo metodui.

“darbai10-5.txt” faile aprašytą uždavinį sprendžiant modeliuojamojo atkaitinimo metodu (iteracijų skaičius – 1000, pradinė temperatūra – 1000, temperatūros daugiklis – 0,99) darbų pabaigos vidurkis yra beveik dvigubai nutolęs nuo mažiausios rastos. Naudojant paiešką su draudimais (iteracijų skaičius – 1000, draudimų sąrašo ilgis – 30%) darbų pabaigos vidurkis yra artimas mažiausiai gautai reikšmei, o vienam tvarkaraščiui sudaryti reikia maždaug 5 kartus mažiau laiko nei modeliuojamojo atkaitinimo metodui.

Šakų ir ribų metodas yra labai jautrus uždavinio sudėtingumui laiko, reikalingo optimaliam sprendiniui rasti, atžvilgiu, kadangi sparčiai didėja nagrinėtinų paieškos medžio viršūnių (dalinių sprendinių) skaičius. Tikslieji algoritmai – toks yra šakų ir ribų algoritmas - taikytini tik mažos apimties uždaviniams spręsti, nes sprendžiant didelės apimties uždavinius, tiksliam sprendiniui rasti reiktų labai daug laiko (skaičiuojama metais).

Naudojant modeliuojamąjį atkaitinimą, galima pastebėti, kad sprendinio vidurkio gerėjimui didesnę įtaką turi iteracijų skaičiaus didinimas nei pradinės temperatūros kėlimas. Iteracijų skaičius nusako, kaip nuodugniai bus ištirtas sprendinių poaibis prie tam tikros temperatūros. Taip pat kuo didesnis temperatūros daugiklis, tuo mažiau iteracijų reikia atlikti, nes sprendinio ieškoma nuosekliau tiriant sprendinių sritį. Kaip matyti iš gautų rezultatų, sudėtingėjant uždaviniui, metodas labai lėtai

artėja prie minimalios funkcijos reikšmės, be to, sparčiai didėja skaičiavimų laikas. Taip yra todėl, kad naujas sprendinys pasirenkamas atsitiktinai iš visų „kaimyninių“ sprendinių ir yra priimamas arba atmetamas.

Naudojant paiešką su draudimais, galima pastebėti, kad geriausi rezultatai gaunami prie tam tikrų parametrų reikšmių. Kai draudimų sąrašo ilgis yra per trumpas, gali būti, kad paieška apsistoja ties lokaliu minimumu. Kai per ilgas – metodas „nukeliauja“ per toli nuo optimalaus sprendinio. Kai iteracijų skaičius yra per didelis, sprendinio ieškoma plačioje sprendinių erdvėje. Ieškant naujo sprendinio jis parenkamas „apgalvotai“, t.y. priimamas tas sprendinys, kurio tikslo funkcijos reikšmė yra mažiausia.

Skaičiuojant tiek modeliuojamojo atkaitinimo, tiek paieškos su draudimais metodais, parametrų pasirinkimas priklauso nuo uždavinio sudėtingumo. Jei uždavinys nėra sudėtingas, tuomet patartina pasirinkti neaukštą pradinę temperatūrą (apie 100), iteracijų skaičių apie 100, o temperatūros daugiklį 0,9. Uždaviniui sudėtingėjant, patartina pasirinkti didesnį temperatūros daugiklį (0,99) bei didinti iteracijų skaičių (500 ar daugiau), pradinę temperatūrą taip pat rekomenduojama kelti, kad būtų iššvalgyta platesnė sprendinių sritis. Sprendžiant nesudėtingus uždavinius paieškos su draudimais metodu, patartina draudimų sąrašo ilgį rinktis apie 40% – 70%, o iteracijų skaičių apie 900. Sudėtingėjant uždaviniui, reikia mažinti draudimų sąrašo ilgį (apie 30%) ir didinti iteracijų skaičių (apie 1000).

IŠVADOS

1. Šakų ir ribų algoritmas labai jautrus uždavinio sudėtingumui laiko, reikalingo optimaliam sprendiniui rasti, atžvilgiu. Šį algoritmą patartina naudoti tik mažos apimties uždaviniams spręsti.
2. Naudojant modeliuojamojo atkaitinimo algoritmą, geriausi rezultatai gaunami, kai iteracijų skaičius yra nuo 100 iki 1000, pradinė temperatūra apie 1000, o temperatūros daugiklis – 0,95-0,99.
3. Naudojant paieškos su draudimais algoritmą, geriausi rezultatai gaunami, kai iteracijų skaičius yra 900 - 1000, draudimų sąrašo ilgis – 30% - 70%.
4. Remiantis gautais rezultatais bei skaičiavimams atlikti reikalingu laiku, galima teigti, kad tikslusis metodas – šakų ir ribų – yra efektyvus tik nesudėtingiems uždaviniams spręsti. Sprendžiant sudėtingesnius uždavinius efektyvesni yra meta-euristiniai metodai.
5. Iš dviejų nagrinėtų meta-euristinių metodų, modeliuojamojo atkaitinimo ir paieškos su draudimais, pastarasis yra efektyvesnis. Galima sakyti, kad naudojantis dirbtinio intelekto savybes metodas yra efektyvesnis nei naudojantis atsitiktinai generuojamus sprendinius.

REKOMENDACIJOS

Norint padidinti šakų ir ribų metodo efektyvumą, galima griežčiau įvertinti paieškos medžio viršūnių (dalinių sprendinių) apatinius rėžius. Tokiu būdu, būtų nukertama daugiau šakų, tariant, kad tose šakose negali būti optimalaus sprendinio. Gali būti, kad bus nukertama ir ta šaka, kurioje yra optimalus sprendinys, tačiau, nukirtus daugiau šakų, būtų sumažinamas paieškos medis, tuo pačiu ir sprendimo laikas, nes reikėtų nagrinėti mažiau viršūnių, o rastas sprendinys nebūtų optimalus, tačiau pakankamai geras.

Taip pat galima padaryti modeliuojamojo atkaitinimo bei paieškos su draudimais algoritmų kombinaciją, norint paspartinti optimalaus sprendinio paiešką. Pavyzdžiui, iš pradžių taikomas modeliuojamojo atkaitinimo algoritmas, kad būtų iššvalgyta pakankama sprendinių sritis, o tolesnei paieškai pritaikomas paieškos su draudimais algoritmas.

PADĖKOS

Nuoširdžiai dėkoju savo vadovui, Narimantui Listopadskiui, už vertingus patarimus ir pagalbą rašant šį darbą.

NAUDOTA LITERATŪRA

1. FELINSKAS, G. Euristinių metodų tyrimas ir taikymas ribotų išteklių tvarkaraščiams optimizuoti. *Daktaro disertacija*. VDU Matematikos ir informatikos institutas, 2007. Prieiga per internetą: <http://www.mii.lt/files/mii_dis_07_felinskas.pdf>
2. ŽILINSKAS, A. *Matematinis programavimas*. 2005. p. 7-176.
3. BROCKMANN, K.; DANGELMAIER, W. A parallel branch & bound algorithm for makespan optimal sequencing in flow shops with parallel machines. [žiūrėta 2009-05]. Prieiga per internetą: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.8621>
4. Combinatorial Optimization [interaktyvus]. [žiūrėta 2007-11]. Prieiga per internetą: <http://www.iitp.ru/mslevin/CO.HTM>
5. CRAINIC, T. G. Parallel branch and bound algorithms. [žiūrėta 2009-05]. Prieiga per internetą: <http://media.wiley.com/product_data/excerpt/18/04717210/0471721018.pdf>
6. CRESCENZI, P.; KANN, V. A compendium of NP optimization problems. [žiūrėta 2009-05]. Prieiga per internetą: <http://www.csc.kth.se/~viggo/wwwcompendium/wwwcompendium.html>
7. Crystal Ball Risk Analysis Software & Solutions [interaktyvus]. [žiūrėta 2007-11]. Prieiga per internetą: <http://www.decisioneering.com/optimization.html>
8. GARG, N.; JAIN, S.; SWAMY, C. A Randomized Algorithm for Flow Shop Scheduling. [žiūrėta 2009-05]. Prieiga per internetą: < http://www.math.uwaterloo.ca/~cswamy/papers/049.pdf >
9. GAUL:Genetic Algorithm Utility Library [interaktyvus]. [žiūrėta 2009-05]. Prieiga per internetą: <http://gaul.sourceforge.net/tutorial/sa.html>
10. HAGEMAN, J. A.; STREPPEL, M.; WEHRENS, R.; BUYDENS, L. M. C. Wavelength selection with Tabu Search. Journal Of Chemometrics, 2003, 17, p. 427-437. [žiūrėta 2009-05]. Prieiga per internetą: <http://www.cac.science.ru.nl/research/publications/PDFs/hageman2003a.pdf>
11. Home page J E Beasley [interaktyvus]. [žiūrėta 2009-05]. Prieiga per internetą: <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/flowshop1.txt >
12. YOUNES, A. Adapting Evolutionary Approaches for Optimization in Dynamic Environments. *A thesis presented to the University of Waterloo in fulfillment of the thesis requirement for the degree of Doctor of Philosophy in Systems Design Engineering*. Waterloo, Ontario, Canada, 2006. [žiūrėta 2009-05]. Prieiga per internetą: <http://etd.uwaterloo.ca/etd/ayounes2006.pdf>
13. LIAO, L. – W.; SHEEN G. – J. Parallel machine scheduling with machine availability and eligibility constraints. [žiūrėta 2009-05]. Prieiga per internetą: <http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6VCT-4MX4VTR-2-

3W&_cdi=5963&_user=5674488&_orig=search&_coverDate=01%2F16%2F2008&_sk=998159997&view=c&wchp=dGLzVlz-zSkWz&md5=fd64d8a9884bda877be1eb7af79a8928&ie=/sdarticle.pdf>

14. Metaheuristics Network [interaktyvus]. [žiūrėta 2009-05]. Prieiga per internetą: <http://www.metaheuristics.org/index.php?main=4&sub=43>
15. NEOS Guide Optimization Tree [interaktyvus]. [žiūrėta 2007-11]. Prieiga per internetą: <http://www-fp.mcs.anl.gov/otc/Guide/OptWeb/index.html>
16. RESENDE, M. G. C.; DE SOUSA, J. P. *Metaheuristics: computer decision – making*. Dordrecht, The Netherlands, 2004. [žiūrėta 2009-05]. Prieiga per internetą: <http://www.research.att.com/~mgcr/mic2001/>
17. SCHMIDT, G. Scheduling with limited machine availability. [žiūrėta 2009-05]. Prieiga per internetą: <http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6VCT-3Y9MCF4-1-9&_cdi=5963&_user=5674488&_orig=na&_coverDate=02%2F15%2F2000&_sk=998789998&view=c&wchp=dGLzVlz-zSkzk&md5=6d78d5a4b6a385a803759b6cffa5ae27&ie=/sdarticle.pdf>
18. SHEEN, G. – J.; LIAO L. – W.; LIN C. – F. Optimal parallel machines scheduling with machine availability and eligibility constraints. [žiūrėta 2009-05]. Prieiga per internetą: <http://www.springerlink.com/content/d26252v22h3254m4/fulltext.pdf>
19. Simulated Annealing [interaktyvus]. [žiūrėta 2007-11]. Prieiga per internetą: <http://members.aol.com/btluke/simann1.htm>
20. Tabu Search [interaktyvus]. [žiūrėta 2009-05]. Prieiga per internetą: <http://www.cs.sandia.gov/opt/survey/ts.html>
21. Tabu Search [interaktyvus]. [žiūrėta 2009-05]. Prieiga per internetą <http://www.idsia.ch/~monaldo/tabusearch.html>
22. VACIC, V.; SOBH, T. M. Vehicle Routing Problem with Time Windows. *Independent study*. University of Bridgeport Department of Computer Science and Engineering. 2002. [žiūrėta 2009-05]. Prieiga per internetą: <http://www.bridgeport.edu/sed/projects/cs399/vacic/vrptw.html>
23. WANG, X.; XIE, J. Branch and bound algorithm for flexible flowshop with limited machine availability, Asian Information-Science-Life, Volume 1, Number 3, pp. 241-248. [žiūrėta 2009-05]. Prieiga per internetą: <http://faculty.math.tsinghua.edu.cn/~jxie/papers/AISF2002.pdf>
24. Wikipedia The Free Encyclopedia [interaktyvus]. [žiūrėta 2009-05]. Prieiga per internetą: <http://en.wikipedia.org/wiki>

A PRIEDAS. TYRIMO REZULTATAI

A.1 lentelė

Modeliuojamojo atkaitinimo metodu gauti rezultatai (failas „darbai3-3.txt“)

Temperatūros daugiklis	Pradinė temperatūra	Įvertis	Iteracijų skaičius						
			5	100	200	300	400	500	1000
0.9	5	min	275	275	275	275	275	275	275
		Vidurkis	276.2	275	275	275	275	275	275
		Dispersija	4.552	0	0	0	0	0	0
		ms	2.432	59.094	104.148	154.408	233.254	292.25	557.424
	100	min	275	275	275	275	275	275	275
		Vidurkis	275.55	275	275	275	275	275	275
		Dispersija	2.45	0	0	0	0	0	0
		ms	2.776	63.118	142.894	222.896	270.506	335.964	785.432
	200	min	275	275	275	275	275	275	275
		Vidurkis	275.62	275	275	275	275	275	275
		Dispersija	2.69	0	0	0	0	0	0
		ms	2.996	78.56	142.184	203.362	310.63	391.564	736.888
	300	min	275	275	275	275	275	275	275
		Vidurkis	275.51	275	275	275	275	275	275
		Dispersija	2.278	0	0	0	0	0	0
		ms	3.212	68.516	152.036	242.456	294.902	360.424	824.122
	400	min	275	275	275	275	275	275	275
		Vidurkis	275.49	275	275	275	275	275	275
		Dispersija	2.206	0	0	0	0	0	0
		ms	3.12	82.806	152.384	213.756	321.52	416.93	780.286
	500	min	275	275	275	275	275	275	275
		Vidurkis	275.21	275	275	275	275	275	275
		Dispersija	1.006	0	0	0	0	0	0
		ms	3.56	70.106	152.504	254.248	306.288	367.664	839.846
	1000	min	275	275	275	275	275	275	275
		Vidurkis	275.33	275	275	275	275	275	275
		Dispersija	1.542	0	0	0	0	0	0
		ms	3.212	87.984	163.552	224.424	335.096	445.292	835.576
0.95	5	min	275	275	275	275	275	275	275
		Vidurkis	275.07	275	275	275	275	275	275
		Dispersija	0.346	0	0	0	0	0	0
		ms	4.678	100.902	214.5	365.948	445.1	523.474	1171.69
	100	min	275	275	275	275	275	275	275
		Vidurkis	275.09	275	275	275	275	275	275
		Dispersija	0.442	0	0	0	0	0	0
		ms	5.556	156.312	290.442	391.31	583.164	789.834	1485.318
	200	min	275	275	275	275	275	275	275
		Vidurkis	275.01	275	275	275	275	275	275
		Dispersija	0.05	0	0	0	0	0	0
		ms	6.77	134.378	281.114	488.378	611.554	695.67	1590.554
	300	min	275	275	275	275	275	275	275
		Vidurkis	275.05	275	275	275	275	275	275
		Dispersija	0.25	0	0	0	0	0	0
		ms	6.394	164.392	321.55	418.988	613.27	849.612	1644.906
	400	min	275	275	275	275	275	275	275
		Vidurkis	275.02	275	275	275	275	275	275
		Dispersija	0.098	0	0	0	0	0	0
		ms	6.99	144.21	293.408	508.408	660.54	733.332	1634.39
	500	min	275	275	275	275	275	275	275
		Vidurkis	275.04	275	275	275	275	275	275
		Dispersija	0.2	0	0	0	0	0	0
		ms	6.21	168.576	339.33	439.328	628.06	889.796	1789.022
	1000	min	275	275	275	275	275	275	275
		Vidurkis	275.05	275	275	275	275	275	275
		Dispersija	0.25	0	0	0	0	0	0
		ms	7.396	152.102	308.856	537.642	699.294	785.344	1814.228
0.99	5	min	275	275	275	275	275	275	275
		Vidurkis	275	275	275	275	275	275	275
		Dispersija	0	0	0	0	0	0	0
		ms	21.464	595.766	1211.846	1588.028	2235.556	3125.758	6391.792
	100	min	275	275	275	275	275	275	275
		Vidurkis	275	275	275	275	275	275	275
		Dispersija	0	0	0	0	0	0	0
		ms	32.948	673.956	1347.66	2342.262	3064.608	3438.606	7347.526
	200	min	275	275	275	275	275	275	275
		Vidurkis	275	275	275	275	275	275	275
		Dispersija	0	0	0	0	0	0	0
		ms	29.514	800.814	1605.062	2146.666	2990.818	4198.27	8744.976

	300	min	275	275	275	275	275	275	275
		Vidurkis	275	275	275	275	275	275	275
		Dispersija	0	0	0	0	0	0	0
		ms	34.726	740.038	1503.228	2509.464	3331.87	4044.946	8326.522
	400	min	275	275	275	275	275	275	275
		Vidurkis	275	275	275	275	275	275	275
		Dispersija	0	0	0	0	0	0	0
		ms	30.204	831.392	1703.28	2238.898	3130.938	4499.692	9153.672
	500	min	275	275	275	275	275	275	275
		Vidurkis	275	275	275	275	275	275	275
		Dispersija	0	0	0	0	0	0	0
		ms	35.536	766.87	1560.914	2576.634	3493.584	4026.7	8666.388
	1000	min	275	275	275	275	275	275	275
		Vidurkis	275	275	275	275	275	275	275
		Dispersija	0	0	0	0	0	0	0
		ms	31.826	872.574	1828.24	2425.068	3292.712	4798.56	9505.142

A.2 lentelė

Modeliuojamojo atkaitinimo metodu gauti rezultatai (failas „darbai4-4.txt“)

Temperatūros daugiklis	Pradinė temperatūra	Įvertis	Iteracijų skaičius						
			5	100	200	300	400	500	1000
0.9	5	min	346	336	336	336	336	336	336
		Vidurkis	418.59	360.702	352.37	349.376	346.254	344.594	339.976
		Dispersija	969.936	170.21	127.656	100.9	81.902	65.678	23.498
		ms	4.618	104.774	211.13	318.178	413.216	538.982	1072.222
	100	min	344.5	336	336	336	336	336	336
		Vidurkis	416.014	356.122	351.5	346.292	344.228	342.704	339.124
		Dispersija	834.904	156.334	116.358	79.932	64.366	51.588	17.594
		ms	5.492	135.752	298.648	407.564	533.428	697.358	1457.768
	200	min	340	336	336	336	336	336	336
		Vidurkis	411.462	357.046	348.544	346.284	344.764	342.326	338.668
		Dispersija	773.364	138.674	98.624	79.066	68.398	46.684	15.288
		ms	5.992	138.31	278.526	445.012	587.684	750.614	1544.192
	300	min	340.5	336	336	336	336	336	336
		Vidurkis	413.576	357.082	350.078	346.464	343.538	341.978	338.968
		Dispersija	795.684	141.444	100.136	76.288	55.4	41.006	14.578
		ms	6.176	153.91	300.736	464.882	603.726	790.798	1554.736
	400	min	342.25	336	336	336	336	336	336
		Vidurkis	410.104	355.876	348.654	345.844	343.22	341.15	338.538
		Dispersija	814.76	148.666	102.524	79.04	52.924	36.77	14.61
		ms	6.084	148.826	310.35	465.07	600.228	773.3	1621.35
	500	min	342.75	336	336	336	336	336	336
		Vidurkis	409.494	355.904	349.356	345.172	343.762	341.744	338.648
		Dispersija	782.312	135.472	102.792	71.952	57.148	42.554	12.544
		ms	6.056	155.72	308.136	463.386	609.654	778.508	1615.138
	1000	min	342	336	336	336	336	336	336
		Vidurkis	408.732	355.506	348.48	345.016	343.37	341.652	338.606
		Dispersija	822.402	125.396	100.146	75.374	54.86	41.346	14.102
		ms	6.24	154.348	320.77	479.202	624.938	821.874	1653.144
0.95	5	min	344.75	336	336	336	336	336	336
		Vidurkis	401.718	353.334	345.286	342.984	341.322	340.172	337.596
		Dispersija	630.088	121.78	76.43	49.654	37.226	24.448	6.912
		ms	8.738	219.712	445.38	745.748	882.5	1150.662	2258.33
	100	min	337.5	336	336	336	336	336	336
		Vidurkis	397.226	350.522	343.796	341.648	340.18	339.336	337.046
		Dispersija	690.352	108.7	60.66	38.058	25.062	18.242	4.566
		ms	11.2	285.106	560.544	836.2	1106.484	1418.89	2889.702
	200	min	338.5	336	336	336	336	336	336
		Vidurkis	398.958	349.544	343.074	340.484	339.836	338.516	336.992
		Dispersija	568.832	113.506	56.366	27.784	21.196	13.062	4.03
		ms	11.638	285.886	570.994	872.452	1169.728	1528.87	3118.518
	300	min	341.25	336	336	336	336	336	336
		Vidurkis	398.056	348.586	343.11	341.034	339.38	338.918	336.94
		Dispersija	613.37	101.174	52.226	34.06	18.864	16.344	3.972
		ms	12.104	296.746	592.99	891.856	1188.758	1554.33	3101.238
	400	min	336	336	336	336	336	336	336
		Vidurkis	394.984	350.802	343.606	340.742	339.528	338.432	336.804
		Dispersija	638.806	105.68	55.442	27.688	21.45	12.674	2.872
		ms	12.294	303.982	613.426	905.12	1193.502	1573.58	3188.63
	500	min	338.5	336	336	336	336	336	336
		Vidurkis	395.596	348.658	343.392	340.806	339.376	338.388	336.858
		Dispersija	605.44	100.152	56.872	32.406	18.184	11.41	3.144
		ms	12.356	297.772	626.154	925.768	1246.198	1612.15	3195.744
	1000	min	346.25	336	336	336	336	336	336

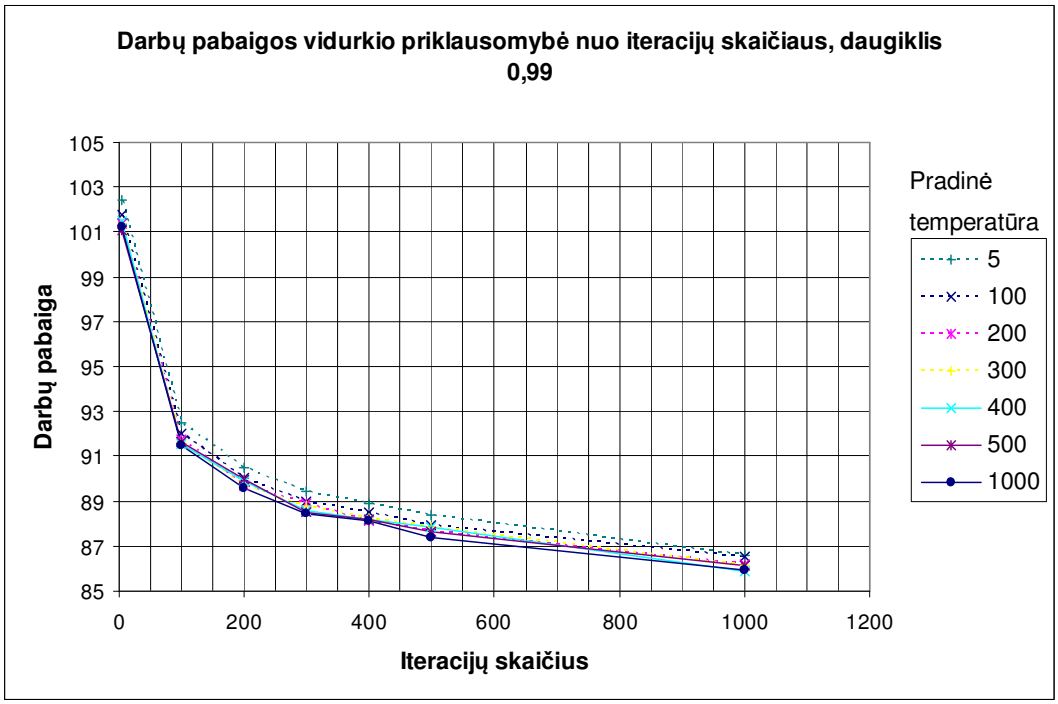
		Vidurkis	396.522	348.68	343.222	340.61	339.042	338.488	336.75
		Dispersija	615.028	101.494	55.744	30.524	16.642	11.78	3.01
		ms	12.73	323.606	649.122	971.356	1315.494	1683.03	3291.09
0.99	5	min	337.75	336	336	336	336	336	336
		Vidurkis	373.132	339.67	337.336	336.574	336.23	336.172	336.008
		Dispersija	295.156	22.83	6.224	2.404	0.834	0.57	0.032
		ms	43.678	1101.058	2203.014	3287.75	4464.498	6747.292	11636.99
	100	min	336	336	336	336	336	336	336
		Vidurkis	367.324	338.676	336.778	336.35	336.164	336.084	336
		Dispersija	271.778	13.638	3.156	1.26	0.586	0.316	0
		ms	55.786	1384.418	2801.718	4177.178	6818.71	7787.57	14911.79
	200	min	336	336	336	336	336	336	336
		Vidurkis	367.612	338.876	336.86	336.37	336.108	336.062	336
		Dispersija	253.35	16.16	3.32	1.378	0.382	0.192	0
		ms	59.87	1481.76	2972.194	4453.95	5952.688	7562.862	15916.28
	300	min	336	336	336	336	336	336	336
		Vidurkis	366.74	338.72	336.732	336.302	336.154	336.022	336
		Dispersija	279.302	13.656	2.952	1.106	0.544	0.082	0
		ms	61.092	1513.862	3076.682	4520.538	6179.73	7850.472	16499.98
	400	min	336	336	336	336	336	336	336
		Vidurkis	364.766	338.316	336.734	336.336	336.13	336.044	336
		Dispersija	224.304	11.644	3.204	1.198	0.508	0.136	0
		ms	63.12	1554.332	3129.816	5350.238	6112.898	8030.306	16184.89
	500	min	336	336	336	336	336	336	336
		Vidurkis	366.916	338.58	336.752	336.302	336.114	336.032	336
		Dispersija	259.012	10.664	3.11	1.216	0.448	0.102	0
		ms	64.308	1609.866	3148.662	5635.188	6550.606	8262.158	15913.76
	1000	min	336	336	336	336	336	336	336
		Vidurkis	365.21	338.156	336.646	336.242	336.106	336.038	336
		Dispersija	205.756	10.316	2.338	0.844	0.368	0.122	0
		ms	66.11	1630.024	3272.182	6062.792	6972.402	8683.514	16603.5

A.3 lentelė

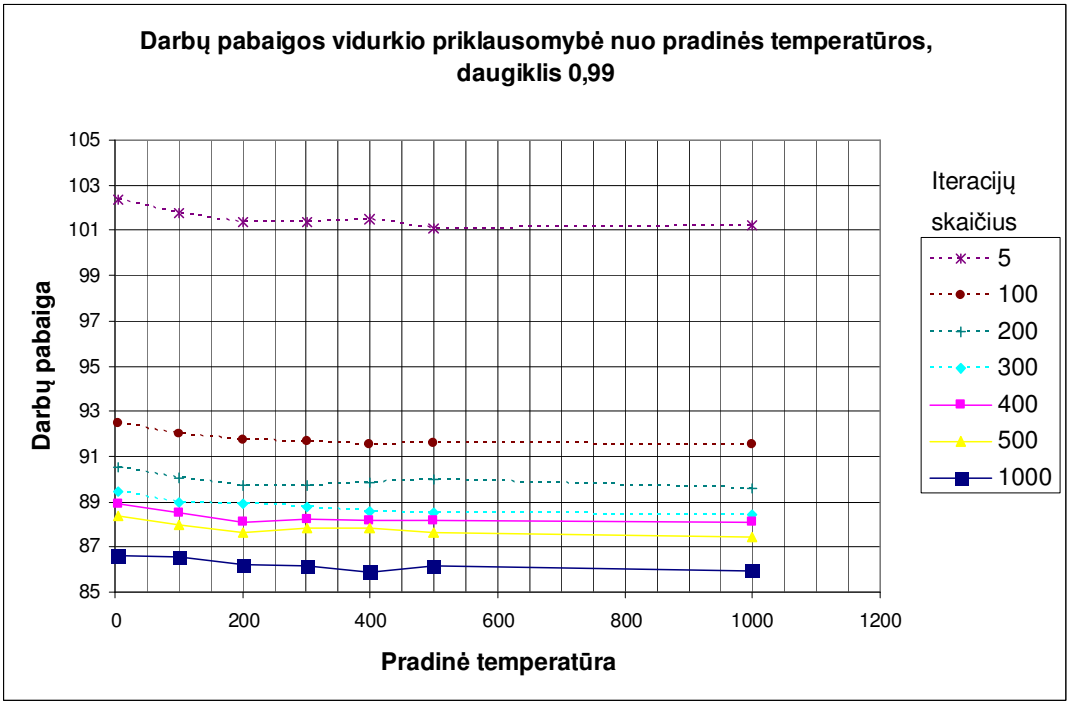
Modeliuojamojo atkaitinimo metodu gauti rezultatai (failas „darbai6-6.txt“)

Temperatūros daugiklis	Pradinė temperatūra	Įvertis	Iteracijų skaičius						
			5	100	200	300	400	500	1000
0.9	5	min	95.25	86.25	85.25	85.25	86.75	83.5	83
		Vidurkis	112.176	99.214	97.224	96.186	95.35	94.682	92.792
		Dispersija	35.484	16.62	13.356	12.542	10.674	12.064	9.374
		ms	6.77	161.492	323.796	485.79	647.842	820.282	1625.096
	100	min	96	87.75	86.25	84.5	82.75	83	81
		Vidurkis	111.234	98.568	96.314	95.43	94.618	93.564	92.112
		Dispersija	27.426	13.268	13.27	12.228	11.056	12.62	9.962
		ms	9.576	206.454	410.004	624.222	830.766	1040.338	2081.178
	200	min	95.5	85.75	87.75	82.75	81.5	80.75	83.25
		Vidurkis	110.886	98.192	96.662	95.376	94.21	93.724	91.986
		Dispersija	29.624	15.076	11.494	12.644	12.404	11.924	9.412
		ms	8.924	216.06	434.93	650.928	871.862	1093.91	2173.876
	300	min	92.5	87.75	86.5	82.75	84.25	84.5	79.5
		Vidurkis	110.304	98.644	96.31	95.03	94.284	93.618	91.812
		Dispersija	32.662	14.206	12.122	12.566	11.012	10.592	11.508
		ms	9.234	222.33	444.788	670.65	898.908	1122.21	2234.682
	400	min	94.25	86.25	84	84.75	83.5	83.75	83
		Vidurkis	110.646	98.284	96.248	94.83	94.106	94.066	91.744
		Dispersija	28.788	14.528	12.28	12.028	10.518	10.408	9.456
		ms	9.578	225.516	457.74	686.312	916.164	1108.572	2287.006
	500	min	97	86.25	83.75	84	81.5	81	79
		Vidurkis	110.662	98.158	96.036	95.05	94.046	93.442	91.53
		Dispersija	26.004	14.606	13.276	12.01	11.95	12.184	10.37
		ms	9.766	227.322	466.91	700.786	927.678	1153.222	2343.012
	1000	min	94.25	87	84.25	84.25	84.5	83.25	81.5
		Vidurkis	110.542	97.992	95.972	94.714	94.04	93.494	91.55
		Dispersija	31.968	14.344	13.712	12.072	10.486	11.262	10.018
		ms	10.076	240.398	485.942	725.904	965.992	1204.486	2403.974
0.95	5	min	91.5	86	85.75	83	83.75	81.75	80.75
		Vidurkis	109.16	97.19	95.218	93.926	93.014	92.294	90.782
		Dispersija	29.854	14.686	11.126	12.17	11.752	10.958	10.366
		ms	13.574	331.814	672.832	1011.136	1340.608	1652.828	3321.822
	100	min	92.75	83.5	85.25	83.25	84.25	81.25	79
		Vidurkis	108.024	96.064	94.2	93.256	92.42	91.724	89.942
		Dispersija	24.96	14.81	11.324	10.336	9.896	10.294	9.504
		ms	17.472	430.934	858.474	1290.378	1718.26	2131.94	4276.922
	200	min	91.5	85.75	82.75	82.5	83	83.25	81.5
		Vidurkis	107.516	96.406	94.224	93.26	92.706	91.852	90.074

		Dispersija	27.516	11.998	11.832	11.942	9.316	8.794	8.672
		ms	18.158	446.852	915.694	1361.638	1825.898	2237.242	4496.544
	300	min	92.25	86	79.75	84.5	83	83	80.5
		Vidurkis	107.488	96.502	93.96	93.048	92.168	91.788	89.882
		Dispersija	24.28	11.936	13.896	9.65	10.298	9.726	9.486
		ms	18.936	461.734	924.962	1403.574	1875.038	2304.728	4610.326
	400	min	94.5	84.5	84	82	83.25	82.75	80.75
		Vidurkis	107.6	96.3	94.17	93.068	92.27	91.522	89.956
		Dispersija	23.224	11.878	10.832	9.54	9.904	11.088	8.496
		ms	18.938	469.218	942.994	1410.468	1884.27	2349.564	4728.578
	500	min	88.75	85.25	82.5	83.5	80.75	80	81
		Vidurkis	107.04	96.358	94.066	92.902	92.086	91.634	89.89
		Dispersija	27.606	11.792	11.896	10.064	10.736	10.346	8.782
		ms	19.654	472.374	947.772	1427.66	1904.21	2375.272	4721.934
	1000	min	94.25	81.75	83.25	81.5	81.75	80	80.75
		Vidurkis	107.304	96.192	94.122	92.902	91.98	91.558	90.07
		Dispersija	22.916	13.948	11.23	11.392	10.57	10.426	7.252
		ms	20.502	493.958	988.142	1492.678	1991.262	2489.586	4966.666
0.99	5	min	89.75	81.25	80.25	80	79.25	80.25	77.25
		Vidurkis	102.402	92.506	90.512	89.418	88.928	88.356	86.616
		Dispersija	18.288	12.202	10.1	10.68	9.42	7.292	7.606
		ms	69.262	1694.232	3385.128	5087.662	6982.448	8780.112	17288.28
	100	min	90.25	81	81	78	78.25	78.5	79.75
		Vidurkis	101.768	92.018	90.042	88.984	88.49	87.942	86.546
		Dispersija	15.422	11.102	8.29	9.8	8.59	7.9	6.934
		ms	91.698	2184.48	4365.906	6526.644	8921.694	11020.26	21658.306
	200	min	86.25	80.5	81.25	78.25	80	76.75	77.5
		Vidurkis	101.338	91.762	89.706	88.904	88.102	87.624	86.182
		Dispersija	19.504	11.56	8.97	9.722	8.412	8.632	7.198
		ms	91.73	2264.948	4598.94	6851.19	9343.34	11376.53	22592.284
	300	min	87.25	80.5	80	79.5	78.75	78	77.25
		Vidurkis	101.346	91.654	89.706	88.744	88.242	87.83	86.15
		Dispersija	19.252	11.664	9.784	9.384	7.602	8.444	7.25
		ms	95.91	2315.68	4759.158	7162.944	9996.39	12089.05	23321.9
	400	min	88.5	82.25	81.25	78	77.75	80	77.5
		Vidurkis	101.508	91.534	89.88	88.588	88.166	87.822	85.858
		Dispersija	16.43	10.08	9.002	10.196	9.168	7.816	7.314
		ms	98.406	2435.426	4917.118	7438.408	9916.516	12287.32	23740.076
	500	min	87.5	81.5	80.75	79.5	78.75	78	78.5
		Vidurkis	101.068	91.596	89.964	88.52	88.146	87.634	86.136
		Dispersija	16.98	10.394	8.076	9.238	7.612	8.386	7.278
		ms	98.248	2489.248	5073.342	7516.44	10073.7	12288.85	24069.206
	1000	min	88.5	81.75	78.75	79.5	77.75	79.25	77.75
		Vidurkis	101.216	91.504	89.59	88.464	88.106	87.404	85.954
		Dispersija	15.996	10.196	9.658	9.12	8.372	7.43	7.24
		ms	103.178	2529.744	5198.544	7864.042	10061.69	12486.79	26038.064



A.1 pav. Darbų pabaigos vidurkio priklausomybė nuo iteracijų skaičiaus („darbai6-6.txt“)



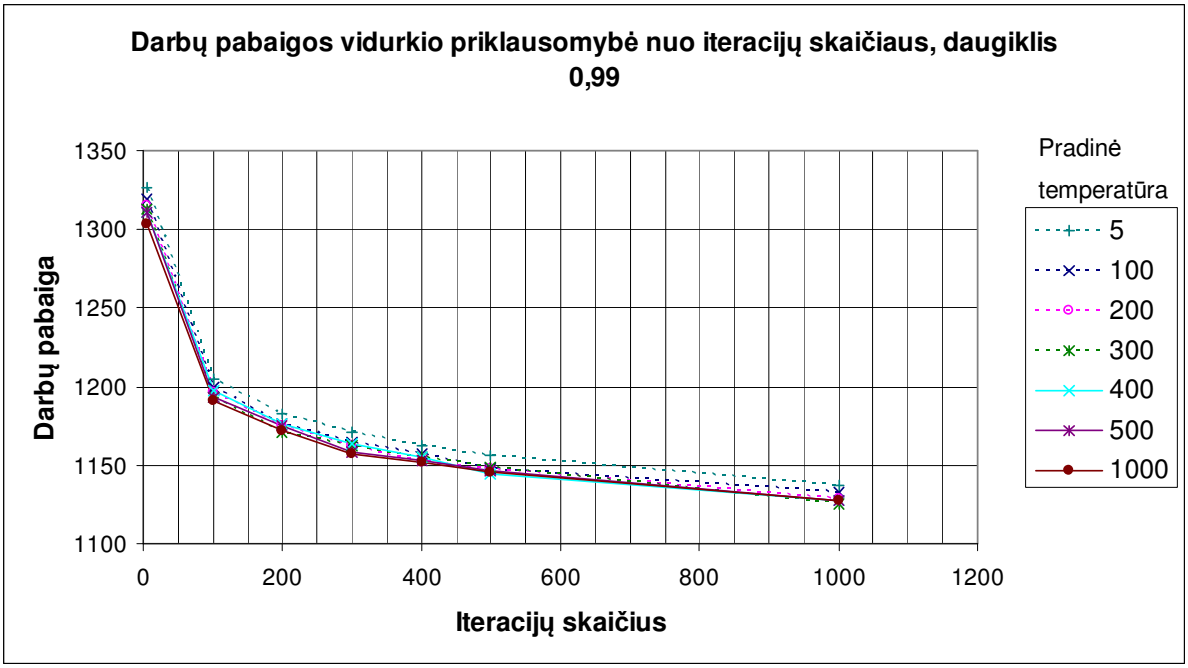
A.2 pav. Darbų pabaigos vidurkio priklausomybė nuo pradinės temperatūros („darbai6-6.txt“)

A.4 lentelė

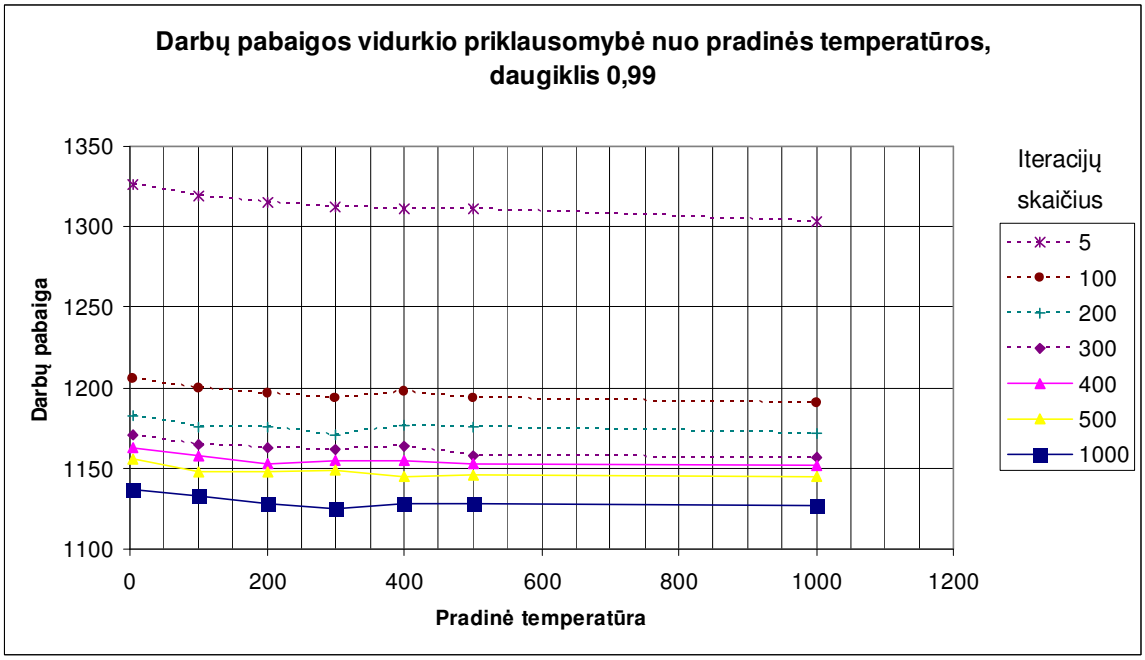
Modeliuojamojo atkaitinimo metodu gauti rezultatai (failas „darbai10-5.txt“)

Temperatūros daugiklis	Pradinė temperatūra	Įvertis	Iteracijų skaičius						
			5	100	200	300	400	500	1000
0.9	5	min	1227.75	1122.25	1114	1103	1089	1099.75	1097.75
		Vidurkis	1447.61	1290.108	1263.376	1248.722	1238.45	1227.5	1207.254
		Dispersija	5322.392	2706.252	1940.778	1894.482	1900.774	1696.95	1416.132
		ms	10.328	249.072	497.736	746.278	1002.87	1241.142	2509.43
	100	min	1225.5	1117.5	1105.75	1101.25	1080	1123.75	1079.5
		Vidurkis	1437.058	1279.94	1258.048	1240.162	1227.926	1222.774	1199.316
		Dispersija	4630.076	2250.316	1698.992	1672.744	1843.99	1285.428	1515.592
		ms	13.2	318.992	645.626	961.31	1274.716	1623.286	3192.53
	200	min	1208.5	1133.5	1104.25	1110	1117	1069.25	1066.75
		Vidurkis	1422.258	1274.706	1252.24	1240.416	1227.71	1218.426	1197.096
		Dispersija	4861.856	2239.938	1676.15	1670.876	1619.624	1913.382	1331.056
		ms	13.634	334.808	661.474	1000.344	1347.036	1668.742	3357.612
	300	min	1218.25	1135.5	1109.5	1119	1085.25	1079	1075.5
		Vidurkis	1435.782	1275.39	1251.386	1237.476	1226.17	1219.11	1197.994
		Dispersija	4398.882	2006.494	1895.28	1628.382	1826.732	1841.582	1416.328
		ms	13.758	340.178	695.488	1037.28	1375.402	1735.074	3448.746
	400	min	1199.25	1138.25	1131	1112.75	1094	1077.5	1093.75
		Vidurkis	1430.672	1274.922	1249.134	1236.424	1228.172	1220.906	1195.792
		Dispersija	4639.296	2067.312	1965.934	1739.764	1821.596	1603.302	1388.738
		ms	14.042	350.472	699.504	1050.698	1413.086	1757.04	3535.418
	500	min	1245.75	1122.5	1115.75	1123	1111	1104.25	1085.75
		Vidurkis	1428.792	1278.154	1250.146	1235.416	1224.874	1217.69	1194.744
		Dispersija	4815.836	1842.49	1818.296	1717.398	1460.2	1615.068	1616.364
		ms	14.324	357.12	716.232	1070.762	1421.698	1799.066	3581.034
	1000	min	1211	1121	1109.5	1093	1058.5	1092.75	1023.5
		Vidurkis	1423.95	1274.498	1246.952	1232.088	1222.512	1219.762	1191.136
		Dispersija	4770.84	2089.166	1961.854	1827.362	1893.236	1406.568	1650.526
		ms	14.978	370.312	742.968	1115.5	1492.306	1846.712	3733.54
0.95	5	min	1220.5	1116.5	1124.25	1089	1098.75	1088.75	1067.5
		Vidurkis	1404.408	1262.904	1235.022	1222.532	1213.194	1206.418	1185.834
		Dispersija	4243.012	1977.94	1894.224	1692.262	1720.384	1710.01	1371.392
		ms	20.28	508.846	1025.58	1535.364	2089.632	2621.41	5246.44
	100	min	1251	1096.5	1094.75	1037.25	1091.25	1054.25	1038
		Vidurkis	1400.486	1252.376	1229.42	1218.264	1205.478	1198.482	1175.796
		Dispersija	3393.402	2220.466	1718.148	1760.246	1528.786	1555.512	1455.136
		ms	27.424	657.356	1316.15	1971.606	2700.22	3329.872	6856.088
	200	min	1177.25	1085.75	1095	1084.25	1081.25	1083	1059.5
		Vidurkis	1394.362	1248.684	1226.106	1214.006	1205.068	1198.206	1175.01
		Dispersija	4318.492	1897.502	1672.238	1518.098	1567.094	1443.026	1327.142
		ms	27.33	695.766	1385.134	2079.994	2727.022	3553.546	6994.212
	300	min	1196.25	1140.75	1081.25	1081.75	1089.75	1079.25	1061.5
		Vidurkis	1389.12	1253.082	1227.506	1209.752	1202.31	1199.088	1175.83
		Dispersija	3981.384	1676.512	1855.544	1493.006	1553.208	1482.116	1383.91
		ms	28.264	705.436	1413.308	2120.114	2829.514	3835.158	7061.884
	400	min	1179.75	1078.5	1120.5	1062.5	1074	1058.25	1058.25
		Vidurkis	1385.098	1254.948	1227.122	1213.322	1202.356	1195.342	1173.242
		Dispersija	3830.06	1882.79	1564.35	1669.878	1484.082	1663.77	1501.024
		ms	28.77	718.884	1450.184	2165.39	2876.444	3728.768	7294.512
	500	min	1191	1129.25	1093.5	1089.25	1072.25	1085.25	1077.75
		Vidurkis	1386.868	1244.838	1226.552	1210.14	1202.666	1194.486	1174.028
		Dispersija	3788.376	1702.906	1544.888	1672.964	1555.108	1498.174	1185.996
		ms	29.642	733.3	1484.316	2177.866	2943.178	3708.424	7434.044
	1000	min	1192.5	1132.5	1073.25	1080.25	1070.5	1076	1051
		Vidurkis	1382.368	1249.644	1220.962	1209.8	1202.138	1193.628	1171.402
		Dispersija	3624.522	1647.276	1898.726	1467.918	1326.728	1410.808	1371.924
		ms	32.01	775.072	1764.25	2452.71	3108.472	3934.908	7729.102
0.99	5	min	1171	1073.75	1061.5	1040.5	1032.2	1051	1021.4
		Vidurkis	1326.38	1205.094	1183.154	1171.094	1162.896	1156.144	1137.188
		Dispersija	2666.19	1634.474	1391.19	1475.182	1216.236	1234.52	1155.766
		ms	105.58	2868.608	6083.632	7967.438	10858.32	13099.06	26355.03
	100	min	1146.5	1084	1067	1055.6	1036.4	1042.8	1037.6
		Vidurkis	1319.59	1199.128	1176.086	1164.466	1157.46	1148.044	1132.74
		Dispersija	2527.336	1361.802	1341.5	1151.052	1289.444	1256.414	1000.102
		ms	132.942	3415.956	6679.496	10139.69	13437.86	17591.14	33838.95
	200	min	1151	1088	1065	1038.8	1037	1028.2	1013.4
		Vidurkis	1315.468	1196.364	1175.342	1162.618	1152.38	1147.552	1128.242
		Dispersija	2520.112	1360.962	1364.128	1348.236	1335.158	1313.168	1158.892
		ms	141.712	3581.532	4582.8	10768.06	14283.83	17441.16	34815.56
	300	min	1180.5	1055.75	1060.25	1058.8	1043.2	1052	1037.2

		Vidurkis	1312.638	1193.258	1171.072	1162.054	1155.276	1148.794	1125.198
		Dispersija	2326.176	1569.658	1127.99	1138.308	1126.43	1121.41	1064.22
		ms	148.606	3708.734	7317.344	10886.09	14973.26	18714.72	36359.81
	400	min	1161.5	1110.75	1046.5	1046.2	1038.2	1022.8	1022.2
		Vidurkis	1310.742	1197.368	1176.284	1163.502	1155.038	1144.948	1127.918
		Dispersija	2589.766	1192.684	1319.854	1325.964	1180.914	1226.628	1047.908
		ms	177.154	4109.192	7874.684	11255.04	15656.32	18573.98	37098.6
	500	min	1125.75	1085.75	1052.5	1038.6	1031.2	1033	1010.2
		Vidurkis	1311.238	1193.604	1175.562	1158.008	1152.826	1146.196	1127.594
		Dispersija	2679.096	1395.242	1269.192	1401.862	1166.192	1219.732	1108.402
		ms	178.466	4185.224	8409.732	12089.77	16041.99	18566.52	37818.33
	1000	min	1147	1071.5	1041	1048.6	1039.2	1031	1014.75
		Vidurkis	1302.882	1190.892	1171.848	1157.24	1152.168	1145.12	1127.376
		Dispersija	2397.838	1520.746	1345.046	1212.646	1219.572	1217.058	1136.314
		ms	157.188	3878.592	7781.016	13365.54	15892.38	19361.07	38524.89



A.3 pav. Darbų pabaigos vidurkio priklausomybė nuo iteracijų skaičiaus („darbai10-5.txt“)



A.4 pav. Darbų pabaigos vidurkio priklausomybė nuo pradinės temperatūros („darbai10-5.txt“)

A.5 lentelė

Paieškos su draudimais metodu gauti rezultatai (failas „darbai3-3.txt“)

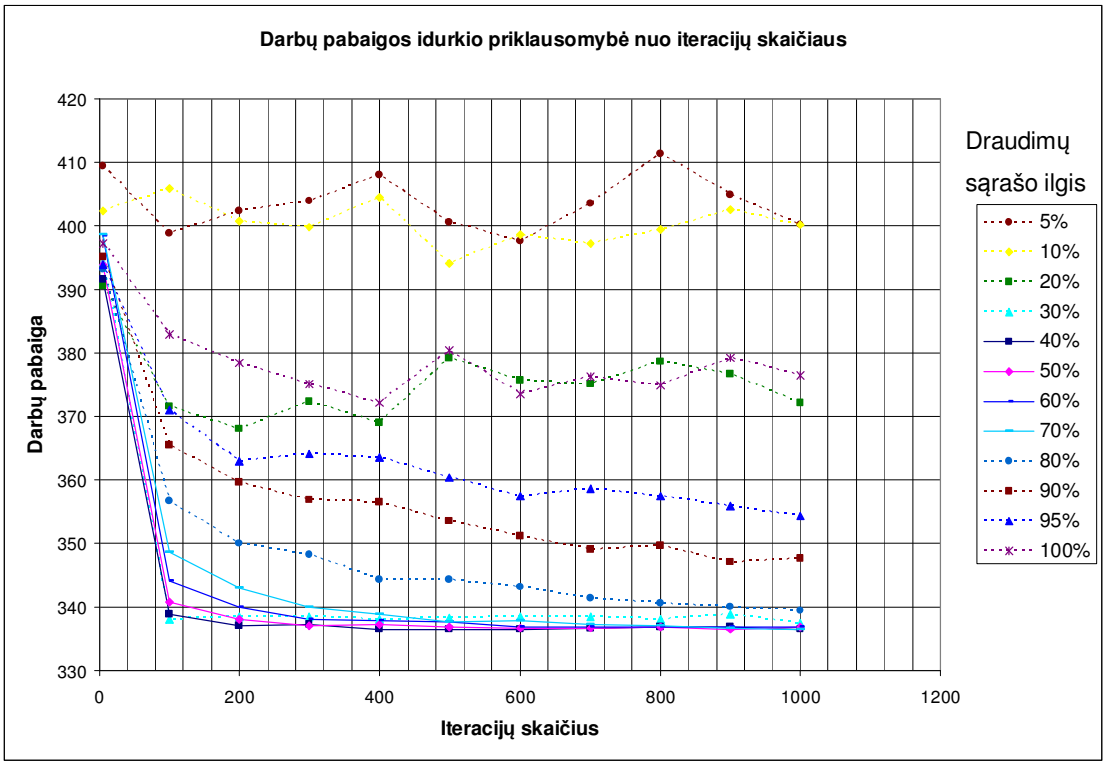
Sąrašo ilgis, %	Įvertis	Iteracijų skaičius										
		5	100	200	300	400	500	600	700	800	900	1000
5	min	275	275	275	275	275	275	275	275	275	275	275
	Vidurkis	284.326	284.632	284.382	283.93	284.044	283.962	284.506	284.476	283.976	284.222	284.12
	Dispersija	47.938	44.312	47.638	45.912	45.06	42.226	43.32	45.784	47.118	45.512	45.704
	ms	0.342	4.432	9.048	13.57	18.002	22.43	26.642	30.608	34.724	39.062	43.804
10	min	275	275	275	275	275	275	275	275	275	275	275
	Vidurkis	284.1	284.344	284.314	284.31	284.282	284.074	284.324	284.152	283.84	284.182	284.198
	Dispersija	50.48	49.782	46.142	46.382	49.082	46.74	49.46	48.354	45.266	47.736	47.34
	ms	0.374	5.146	9.048	13.358	17.442	21.75	26.24	31.264	35.974	40.094	44.024
20	min	275	275	275	275	275	275	275	275	275	275	275
	Vidurkis	283.002	282.694	282.842	282.83	282.668	282.348	282.682	282.586	282.456	283.002	282.826
	Dispersija	19.208	19.634	20.168	19.654	18.332	19.942	20.122	20.534	20.124	19.262	19.676
	ms	0.342	5.178	7.894	11.982	16.096	20.406	23.9	27.27	31.106	34.32	38.844
30	min	275	275	275	275	275	275	275	275	275	275	275
	Vidurkis	281.39	279.09	279.35	279.28	279.42	279.15	279.32	279.19	279.41	279.23	279.39
	Dispersija	14.616	3.678	2.816	3.046	2.554	3.5	2.926	3.362	2.592	3.236	2.66
	ms	0.312	3.776	7.52	10.418	13.948	17.314	20.778	24.96	28.486	32.73	34.88
40	min	275	275	275	275	275	275	275	275	275	275	275
	Vidurkis	280.074	276.84	276.76	276.6	276.88	276.78	276.75	276.85	276.79	276.75	276.77
	Dispersija	17.38	5.748	5.666	5.382	5.81	5.716	5.602	5.796	5.72	5.656	5.7
	ms	0.344	3.402	6.394	9.644	12.67	15.446	18.846	20.842	23.992	27.05	30.016
50	min	275	275	275	275	275	275	275	275	275	275	275
	Vidurkis	279.122	276.9	276.81	276.86	276.81	276.81	276.83	276.99	276.92	276.89	276.71
	Dispersija	14.038	5.848	5.706	5.79	5.728	5.716	5.73	5.938	5.88	5.872	5.58
	ms	0.372	2.934	5.554	8.142	10.198	12.914	15.224	18.344	20.998	22.87	25.804
60	min	275	275	275	275	275	275	275	275	275	275	275
	Vidurkis	279.088	276.31	276.29	276.43	276.31	276.45	276.31	276.23	276.23	276.29	276.3
	Dispersija	14.098	4.812	4.744	5.09	4.796	5.128	4.828	4.606	4.612	4.782	4.736
	ms	0.314	2.368	4.522	6.456	8.988	10.766	12.886	14.914	16.57	19.154	21.152
70	min	275	275	275	275	275	275	275	275	275	275	275
	Vidurkis	279.674	276.09	275.92	276.07	276.22	276.2	276.13	276.03	276.19	275.86	275.95
	Dispersija	15.292	4.25	3.728	4.16	4.596	4.53	4.326	4.082	4.504	3.538	3.828
	ms	0.314	2.06	3.682	5.274	6.77	8.33	10.11	11.606	13.7	14.854	16.224
80	min	275	275	275	275	275	275	275	275	275	275	275
	Vidurkis	280.488	276.52	276.19	276.19	276.26	276.28	276.45	276.48	276.26	276.27	276.43
	Dispersija	18.936	5.214	4.514	4.524	4.632	4.74	5.098	5.104	4.67	4.696	5.092
	ms	0.216	1.498	2.746	3.932	5.02	6.238	7.082	8.392	9.388	10.298	11.92
90	min	275	275	275	275	275	275	275	275	275	275	275
	Vidurkis	280.862	279.664	279.328	279.43	279.588	279.864	279.35	279.278	279.126	279.356	279.936
	Dispersija	21.546	17.996	17.146	15.228	16.082	19.884	16.074	17.394	16.778	18.65	19.848
	ms	0.25	0.906	1.654	2.182	2.872	3.436	4.212	4.87	5.492	6.332	6.8
95	min	275	275	275	275	275	275	275	275	275	275	275
	Vidurkis	280.546	279.386	279.76	279.328	279.348	279.316	279.622	279.68	279.35	279.328	279.508
	Dispersija	20.7	18.074	17.532	18.028	19.278	16.938	19.636	17.206	16.178	15.814	19.346
	ms	0.282	0.872	1.592	2.248	3.028	3.584	4.148	4.772	5.494	6.148	7.428
100	min	275	275	275	275	275	275	275	275	275	275	275
	Vidurkis	281.548	281.096	280.952	280.95	280.936	280.616	281.08	280.678	281.038	281.16	280.694
	Dispersija	21.344	23.554	24.314	22.468	23.466	21.816	20.272	22.246	21.37	25.048	23.52
	ms	0.248	0.282	0.28	0.282	0.218	0.254	0.248	0.282	0.278	0.28	0.278

A.6 lentelė

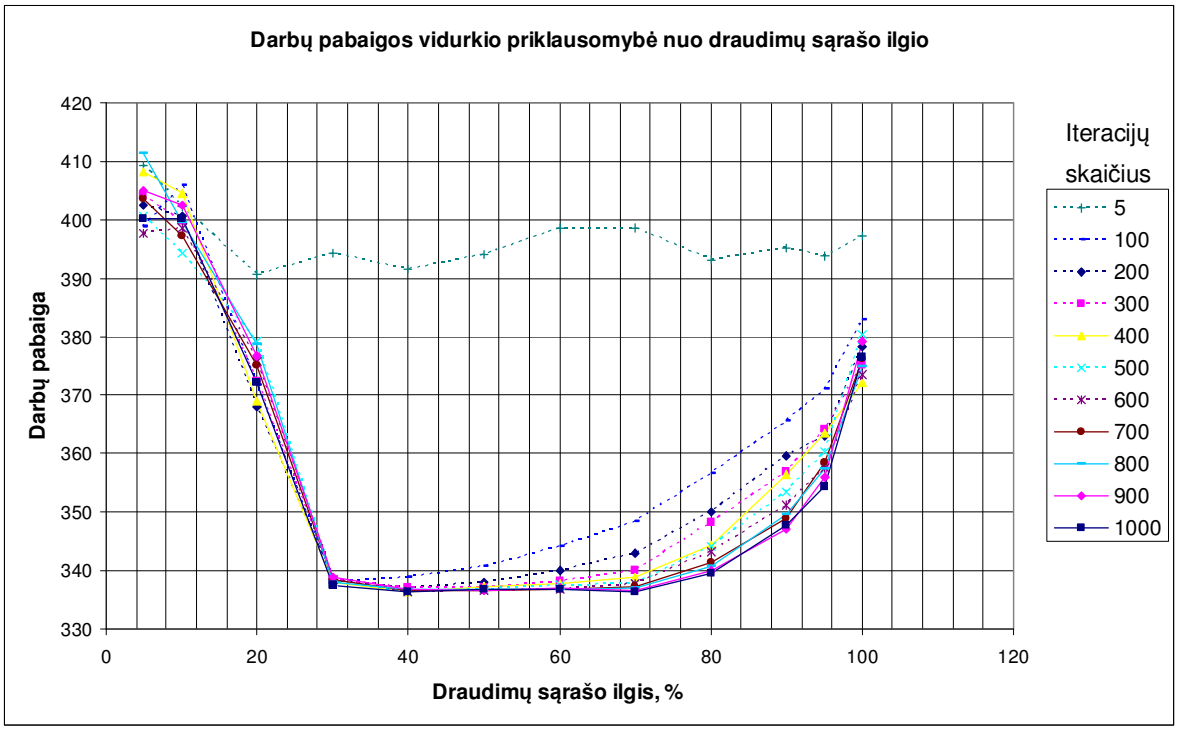
Paieškos su draudimais metodu gauti rezultatai (failas „darbai4-4.txt“)

Sąrašo ilgis, %	Įvertis	Iteracijų skaičius										
		5	100	200	300	400	500	600	700	800	900	1000
5	min	336	336	336	336	336	336	336	336	336	336	336
	Vidurkis	409.388	398.75	402.4	403.94	408.06	400.662	397.734	403.582	411.314	404.944	400.178
	Dispersija	3463.304	3440.888	3546.524	3449.96	3576.502	3252.746	3083.098	3483.24	3851.084	3768.472	3307.604
	ms	1.87	18.628	35.382	52.48	69.516	86.332	101.338	119.338	136.312	153.348	170.384
10	min	336	336	336	336	336	336	336	336	336	336	336
	Vidurkis	402.326	405.854	400.708	399.766	404.42	394.168	398.562	397.208	399.47	402.458	400.178
	Dispersija	3656.672	3844.808	3412.092	3565.506	3534.498	3229.074	3133.432	3104.336	3355.336	3583.882	3434.09
	ms	1.968	18.284	34.41	50.202	67.078	81.84	96.906	114.316	130.386	144.706	162.148
20	min	336	336	336	336	336	336	336	336	336	336	336
	Vidurkis	390.52	371.66	368.014	372.268	369.006	379.282	375.774	375.062	378.692	376.74	372.188
	Dispersija	2979.85	2661.498	2406.654	2910.624	2702.506	3488.612	3377.516	2993.906	3382.28	3351.194	2811.772
	ms	1.84	17.316	32.2	46.926	61.342	74.722	91.042	105.74	119.872	138.062	153.942
30	min	336	336	336	336	336	336	336	336	336	336	336
	Vidurkis	394.312	337.992	338.368	338.448	337.952	338.246	338.43	338.504	337.97	338.824	337.5
	Dispersija	3399.226	51.628	78.28	66.03	62.724	62.126	67.8	70.146	52.868	73.324	42.958
	ms	1.718	19.872	32.386	46.114	60.402	73.008	84.742	98.124	114.786	123.864	136.906

40	min	336	336	336	336	336	336	336	336	336	336	336
	Vidurkis	391.498	338.858	336.99	337.172	336.386	336.466	336.414	336.644	336.838	336.9	336.386
	Dispersija	2997.634	42.972	15.526	65.57	3.38	4.044	5.24	20.818	22.18	27.706	1.44
	ms	1.684	18.754	32.604	45.488	56.974	67.486	79.186	92.384	101.46	115.098	126.048
50	min	336	336	336	336	336	336	336	336	336	336	336
	Vidurkis	393.992	340.784	338.022	337.154	337.29	336.862	336.674	336.686	336.888	336.528	336.79
	Dispersija	3044.11	54.356	22.752	9.668	18.246	9.302	5.488	7.892	16.672	6.252	8.922
	ms	1.716	15.814	29.454	40.56	51.322	60.778	69.108	80.186	93.446	99.372	108.638
60	min	336	336	336	336	336	336	336	336	336	336	336
	Vidurkis	398.472	344.144	340.016	338.106	337.79	337.58	336.796	336.936	336.77	336.958	336.938
	Dispersija	3359.166	119.456	42.976	21.384	15.978	15.832	7.548	10.118	9.702	10.4	13.704
	ms	1.714	13.73	25.116	36.504	47.644	56.534	67.046	75.534	87.86	93.196	101.496
70	min	336	336	336	336	336	336	336	336	336	336	336
	Vidurkis	398.574	348.556	342.962	339.946	338.832	337.712	337.776	337.206	336.972	336.666	336.416
	Dispersija	3118.106	180.85	91.314	46.652	24.282	12.584	13.782	9.3	6.154	5.124	2.522
	ms	1.684	11.14	21.344	31.136	40.436	50.326	57.91	65.582	76.284	85.08	89.012
80	min	336	336	336	336	336	336	336	336	336	336	336
	Vidurkis	393.188	356.612	350.084	348.264	344.386	344.226	343.108	341.358	340.616	339.918	339.508
	Dispersija	3014.578	282.274	192.036	155.828	103.214	94.092	91.886	63.48	50.822	34.772	36.28
	ms	1.75	7.642	14.162	20.312	26.206	33.258	39	47.456	57.688	61.28	67.862
90	min	336	336	336	336	336	336	336	336	336	336	336
	Vidurkis	395.068	365.494	359.614	356.954	356.4	353.494	351.084	349.008	349.56	347.124	347.712
	Dispersija	3132.246	616.476	400.008	316.376	270.046	242.364	204.002	167.074	158.23	150.638	154.616
	ms	2.186	5.712	9.236	13.388	17.91	21.124	25.832	30.702	33.448	38.592	44.522
95	min	336	336	336	336	336	336	336	336	336	336	336
	Vidurkis	393.826	371.076	362.97	364.086	363.476	360.344	357.54	358.576	357.406	355.872	354.272
	Dispersija	2888.898	965.52	682.434	558.55	537.836	445.118	354.136	359.272	297.25	278.762	221.766
	ms	1.652	4.616	7.364	10.294	13.196	15.728	18.098	22.278	25.086	26.364	30.36
100	min	336	336	336	336	336	336	336	336	336	336	336
	Vidurkis	397.302	382.872	378.37	375.174	372.176	380.41	373.544	376.36	374.942	379.176	376.41
	Dispersija	3182.832	1815.814	1618.428	1429.162	1247.024	1617.912	1403.566	1644.144	1357.868	1653.442	1393.682
	ms	1.714	2.466	2.4	2.34	2.372	2.37	2.31	2.372	2.434	2.402	2.372



A.5 pav. Darbų pabaigos vidurkio priklausomybė nuo iteracijų skaičiaus („darbai4-4.txt“)



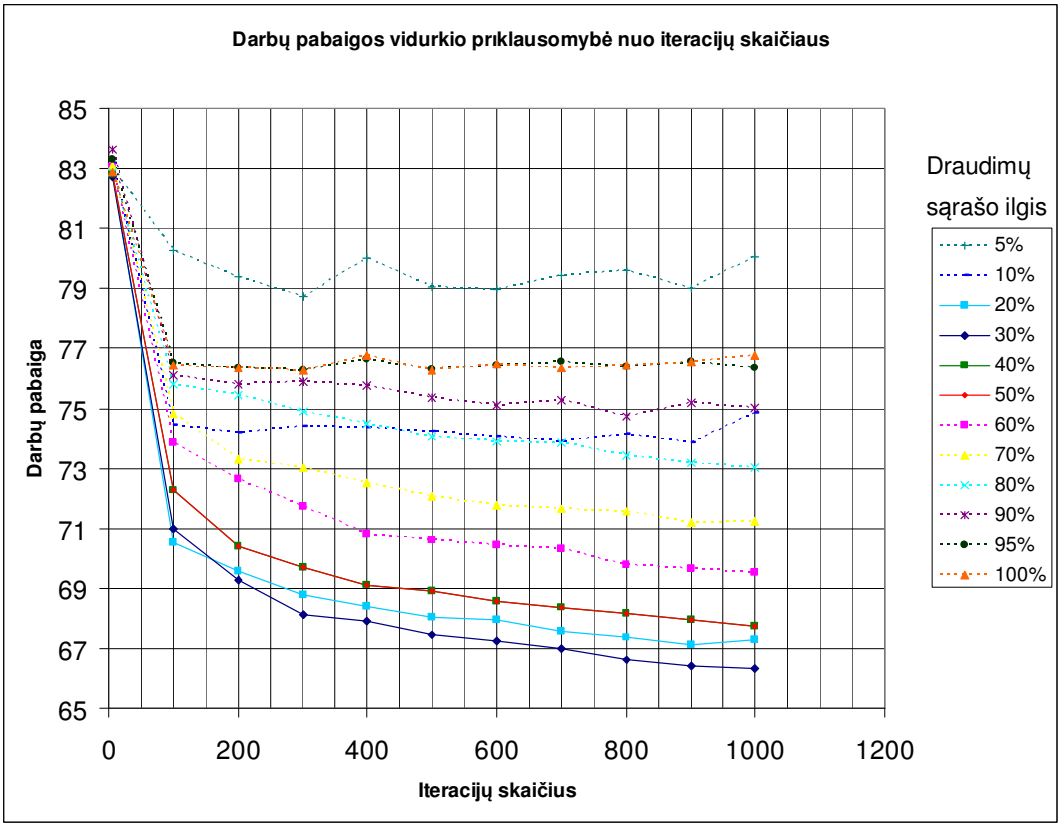
A.6 pav. Darbų pabaigos vidurkio priklausomybė nuo draudimų sąrašo ilgio („darbai4-4.txt“)

A.7 lentelė

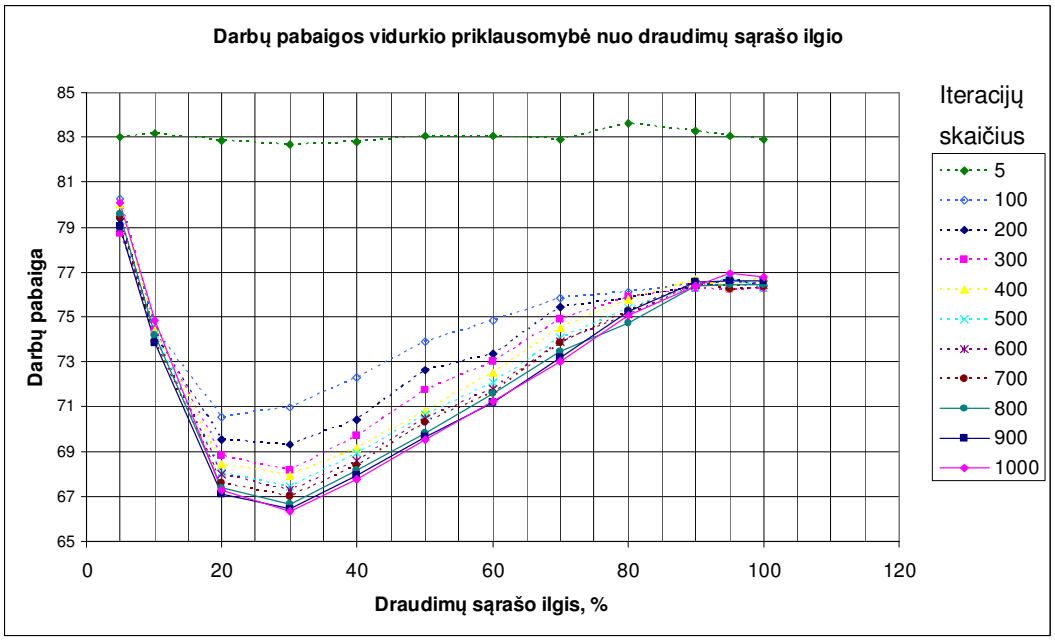
Paieškos su draudimais metodu gauti rezultatai (failas „darbai16-6.txt“)

Sąrašo ilgis, %	Įvertis	Iteracijų skaičius										
		5	100	200	300	400	500	600	700	800	900	1000
5	min	67.4	65	64	63.8	63.8	63.4	63.6	64.8	64.2	63.6	63.2
	Vidurkis	83.016	80.266	79.37	78.708	80.03	79.044	78.976	79.426	79.6	79.028	80.068
	Dispersija	67.09	70.876	66.762	64.086	69.874	68.532	68.412	68.254	72.488	67.59	75.148
	ms	33.13	171.82	306.668	445.288	583.91	713.828	859.316	994.288	1123.394	1261.49	1394.96
10	min	67	63	63.2	63.4	63.2	63	63	63.4	63	63	63.2
	Vidurkis	83.18	74.424	74.194	74.396	74.368	74.222	74.064	73.908	74.154	73.856	74.858
	Dispersija	69.13	43.334	40.348	38.914	42.806	42.528	41.554	39.756	44.728	37.698	48.31
	ms	25.366	201.96	346.634	482.604	627.344	751.642	892.512	1029.668	1149.514	1277.804	1409.626
20	min	67.4	63	63	63	63	63	63	63	63	63	63
	Vidurkis	82.832	70.542	69.556	68.8	68.408	68.036	67.958	67.596	67.372	67.108	67.292
	Dispersija	69.62	20.314	20.634	18.298	15.73	16.482	19.156	17.71	15.976	15.91	16.044
	ms	24.524	240.616	418.83	627.344	792.484	992.166	1173.784	1361.078	1503.412	1785.464	1934.286
30	min	67.8	64.8	63.2	63	63	63.2	63	63	63	63	63
	Vidurkis	82.704	70.978	69.284	68.122	67.914	67.438	67.252	66.978	66.636	66.412	66.336
	Dispersija	65.248	9.102	9.354	5.716	5.838	4.356	4.318	3.888	3.738	3.104	3.182
	ms	23.336	215.094	399.986	584.72	758.164	942.032	1126.386	1310.504	1498.262	1654.296	1825.682
40	min	66.6	64.8	63.8	63.4	63.4	63.4	63.4	63.2	63.6	63.2	63.4
	Vidurkis	82.782	72.278	70.412	69.684	69.126	68.906	68.586	68.388	68.168	67.954	67.764
	Dispersija	64.898	11.386	6.748	5.746	5.61	4.876	4.142	4.164	3.88	4.242	3.27
	ms	23.712	176.874	332.97	477.678	633.242	777.352	933.948	1064.048	1220.272	1354.558	1562.13
50	min	68	66	65.4	65	64.6	64.4	64.2	65.2	64.6	64.6	64.2
	Vidurkis	83.066	73.874	72.638	71.742	70.816	70.622	70.454	70.302	69.784	69.658	69.524
	Dispersija	68.052	13.318	9.452	6.642	7.2	5.79	5.402	4.606	4.688	4.088	4.168
	ms	23.212	139.464	261.458	388.16	508.592	620.478	755.076	897.758	1017.876	1093.878	1258.21
60	min	67	66.2	65.2	66.2	66.4	64.4	64.6	64.8	65.6	64.8	65.4
	Vidurkis	83.09	74.82	73.332	73.026	72.516	72.064	71.768	71.648	71.564	71.206	71.244
	Dispersija	68.768	14.092	11.198	8.488	7.72	7.726	6.634	6.172	5.81	5.274	5.01
	ms	23.462	117.532	213.626	304.14	396.056	501.854	599.824	682.532	801.344	905.802	1008.486
70	min	66.4	67	66.2	66.2	65.8	66.4	66.6	66.4	65.8	66.4	65.2
	Vidurkis	82.898	75.83	75.444	74.89	74.49	74.058	73.912	73.85	73.45	73.196	73.026
	Dispersija	60.956	17.658	13.644	12.902	10.85	10.046	8.934	8.546	9.268	7.422	7.204
	ms	23.618	99.432	157.622	222.894	283.672	360.144	433.932	498.05	558.516	648.092	706.994
80	min	67	65.2	65.2	65.8	65.8	66.4	66	65.8	65	66.6	66
	Vidurkis	83.616	76.096	75.83	75.904	75.768	75.374	75.094	75.274	74.71	75.204	75.038
	Dispersija	71.638	20.45	19.178	17.988	16.556	14.448	14.638	13.856	13.726	11.986	11.54
	ms	23.182	88.296	128.328	168.478	213.41	255.404	306.7	340.956	377.618	437.428	482.354
90	min	67.6	66.6	65.4	66.4	65.8	64.4	65.6	67	66.6	66.2	65.2
	Vidurkis	83.294	76.5	76.332	76.288	76.654	76.324	76.446	76.548	76.384	76.544	76.37

	Dispersija	75.05	25.372	24.862	22.226	24.164	24.626	21.67	20.592	21.318	22.138	22.578
	ms	23.462	78.096	99.466	119.346	140.902	162.24	183.798	204.048	227.886	247.852	268.542
95	min	67.4	64.8	66	65.8	66	66	65.6	66.2	66.2	66.4	66.2
	Vidurkis	83.07	76.598	76.636	76.234	76.332	76.716	76.37	76.22	76.434	76.592	76.958
	Dispersija	71.75	23.206	23.49	22.396	22.736	23.99	23.67	23.302	24.114	19.882	24.182
	ms	23.808	74.79	89.046	104.204	118.716	133.756	147.826	164.364	177.34	190.822	207.668
100	min	66.2	66.6	66	65.2	66.2	65.2	65.8	66.2	67	65.2	66.8
	Vidurkis	82.886	76.422	76.362	76.278	76.78	76.252	76.458	76.348	76.416	76.58	76.768
	Dispersija	66.312	21.686	22.42	23.132	21.992	23.338	23.766	23.19	21.52	23.81	23.556
	ms	23.428	68.356	69.234	68.392	69.016	68.55	68.64	69.262	68.076	68.672	68.764



A.7 pav. Darbų pabaigos vidurkio priklausomybė nuo iteracijų skaičiaus („darbai16-6.txt“)

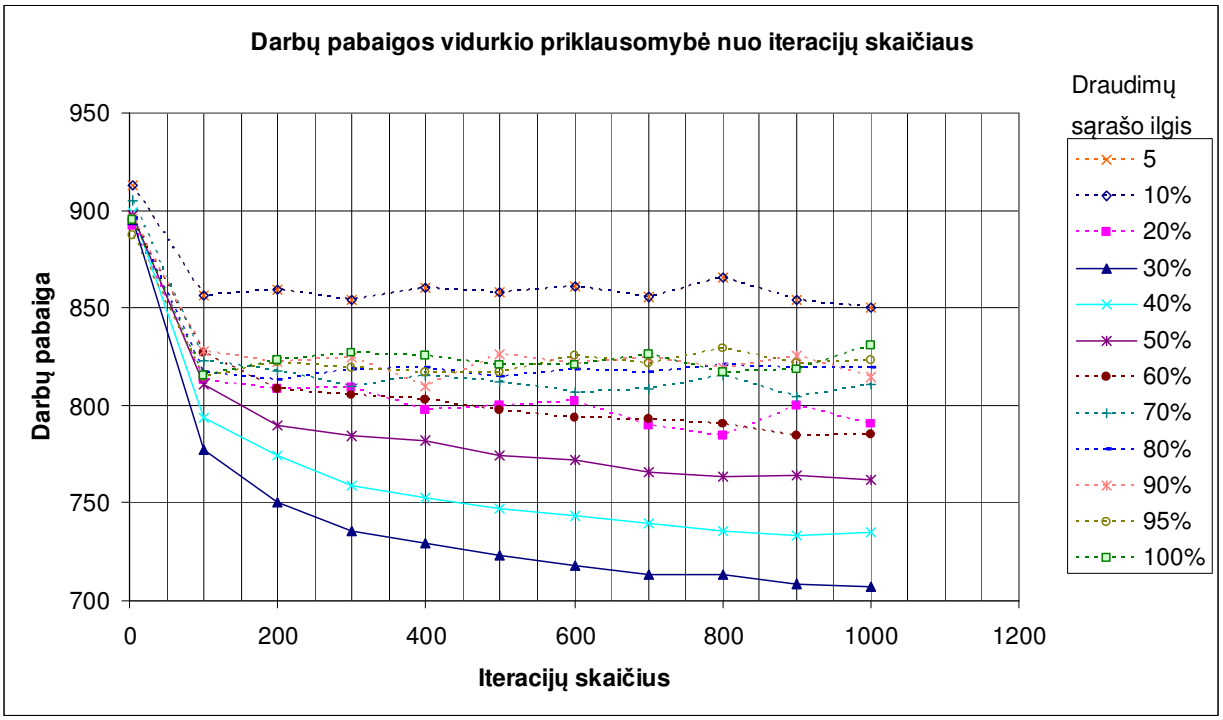


A.8 pav. Darbų pabaigos vidurkio priklausomybė nuo draudimų sąrašo ilgio („darbai16-6.txt“)

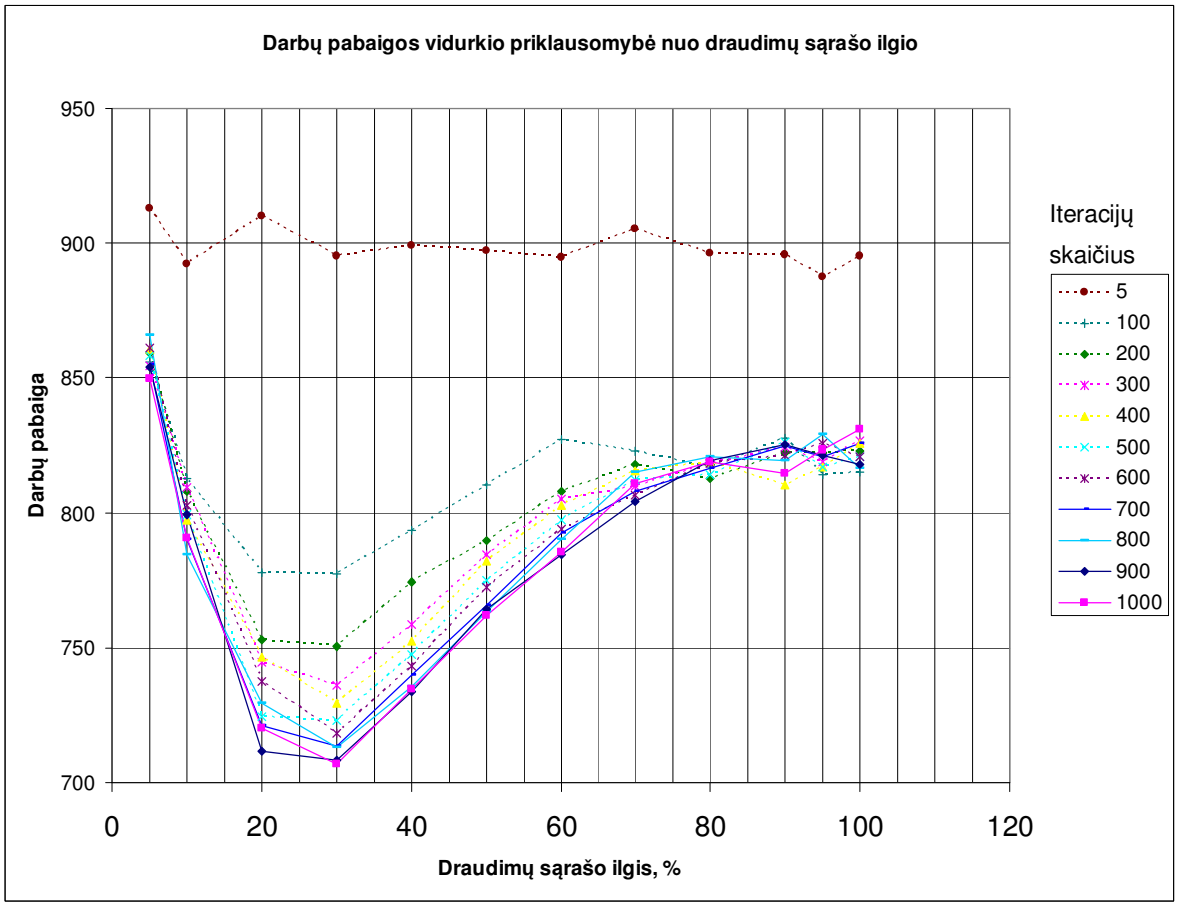
A.8 lentelė

Paieškos su draudimais metodu gauti rezultatai (failas „darbai10-5.txt“)

Sąrašo ilgis, %	Įvertis	Iteracijų skaičius										
		5	100	200	300	400	500	600	700	800	900	1000
5	min	715.2	704	701.4	708.2	693	668.4	666	678.2	653.8	690.2	678
	Vidurkis	913.14	856.184	859.284	854.302	860.59	858.124	861.282	855.19	865.896	854.056	849.92
	Dispersija	7140.972	5274.98	6970.532	5325.968	6482.168	6862.546	6450.54	6129.742	7173.802	7571.676	7533.18
	ms	145.484	825.748	1440.076	2069.384	2508.5	3203.258	3857.406	4288.56	4813.504	4457.854	4497.164
10	min	727.6	651	656.6	646.8	644.6	646.4	653.4	644.6	646	646.8	639
	Vidurkis	892.306	812.926	807.976	809.166	797.42	799.47	802.492	790.012	784.312	799.546	790.632
	Dispersija	5572.238	5121.706	5616.084	6045.55	6650.218	5499.494	4911.666	6198.574	4919.468	7141.462	5741.448
	ms	139.404	981.9	1708.43	2483.1	3188.752	3683.936	4250.122	5008.35	5736.876	4783.834	5410.866
20	min	768	662	663.8	646.4	653.8	652.6	647.2	639.2	646.8	646.8	637.8
	Vidurkis	910.11	777.788	752.554	744.582	746.33	724.682	737.636	720.874	729.464	711.672	720.114
	Dispersija	5421.006	3939.698	3572.284	3863.398	3943.946	3107.106	4413.016	3411.616	4129.082	2904.594	4371.97
	ms	140.37	1319.394	2337.206	3164.352	3795.224	5758.212	5629.638	6713.348	8946.47	7045.066	7614.748
30	min	725.8	688	677.6	661.4	663.6	661.6	661.8	661.8	662.2	661.6	656.4
	Vidurkis	895.412	777.116	750.49	735.942	729.032	723.268	718.122	713.516	712.824	708.246	706.892
	Dispersija	6351.894	2137.124	1684.81	1447.454	1186.536	1139.68	843.056	748.266	911.806	577.952	458.168
	ms	144.768	1108.854	2058.652	3010.884	3892.444	4622.842	5508.894	6447.612	6629.574	6089	6692.6
40	min	751	701.4	708.8	699	671.8	671.8	690.2	678.2	673.2	675	689
	Vidurkis	898.912	793.646	774.54	758.64	752.482	747.274	742.974	739.668	735.704	733.424	734.666
	Dispersija	5219.536	1945.136	1066.584	818.358	885.336	788.042	640.774	681.026	648.164	553.91	398.514
	ms	142.99	915.322	1667.556	2352.434	2834.286	3463.194	4168.754	4757.252	5278.36	4473.048	4791.852
50	min	743.6	693.4	699.6	703	696.8	674.4	708.6	691.2	690.6	694.2	681
	Vidurkis	897.032	810.376	789.714	784.594	781.862	774.64	772.274	765.96	763.208	764.514	761.94
	Dispersija	5187.124	2326.892	1208.438	1268.018	871.074	963.622	700.188	679.518	849.764	630.44	627.316
	ms	144.612	770.02	1223.734	1771.202	2225.262	2789.702	3321.354	3696.942	4342.504	3612.172	4164.29
60	min	747	718.2	697.2	687.2	718.8	695	670.6	705.2	701.4	708.2	700.8
	Vidurkis	894.892	827.248	808.092	805.04	802.732	797.316	793.858	792.684	790.412	784.47	785.32
	Dispersija	6040.834	1963.338	1934.152	1487.556	1225.202	1125.51	1212.522	943.46	855.866	821.528	842.826
	ms	147.766	715.422	1031.324	1310.44	1713.108	2140.116	2557.574	3016.532	3393.086	2821.528	3239.27
70	min	729	714.2	694.4	687.2	712.8	705.2	699	691.8	726.6	703.8	720.2
	Vidurkis	905.23	822.844	817.97	810.092	815.646	812.398	806.52	808.064	814.97	804.152	810.872
	Dispersija	7820.476	2235.308	2283.174	2003.054	1795.932	1380.348	1555.802	1287.73	1211.838	1347.362	934.332
	ms	140.618	660.38	908.27	1099.558	1403.698	1716.822	1897.41	2169.88	2481.602	2104.206	2437.824
80	min	766.8	714	717.4	692.6	707	716.6	717	676.6	708.4	705.2	722
	Vidurkis	896.192	816.966	812.822	818.484	819.03	814.414	818.668	816.648	820.726	819.564	819.114
	Dispersija	5764.124	2521.142	2011.21	2480.294	1985.522	2070.148	2168.204	2250.282	1880.514	2151.452	1977.862
	ms	142.15	648.124	824.62	949.67	1143.112	1263.702	1476.986	1619.634	1807.49	1526.716	1666.996
90	min	728.8	684.2	711.8	715.6	680.6	723.2	687.2	737.8	703.4	702	692.6
	Vidurkis	895.706	827.458	822.2	824.386	810.18	826.414	821.478	824.66	819.552	825.244	814.586
	Dispersija	6396.972	3142.202	2416.86	2595.806	2118.986	2119.76	2131.042	1994.402	2473.048	2709.484	2836.998
	ms	142.146	648.84	774.982	855.044	970.732	1065.954	1166.854	1275.308	1412.464	1142.68	1224.89
95	min	726.8	695.6	713.4	716.2	718	700.2	695.8	704.4	703.2	702	699.2
	Vidurkis	887.486	814.318	822.45	819.206	816.928	817.082	825.49	821.142	828.96	821.176	823.196
	Dispersija	6228.152	2452.658	2972.066	2382.514	2178.248	2378.826	2527.282	2853.926	2009.364	2729.596	2977.99
	ms	147.546	647.182	754.39	806.088	896.132	971.356	1054.726	1120.868	1237.774	992.79	1052.008
100	min	743.8	688.6	709	723.2	718.8	698.2	714.4	714.2	716	692.8	693.4
	Vidurkis	895.174	815.126	822.892	826.768	825.74	821.11	820.684	825.782	816.598	818.09	830.96
	Dispersija	6705.854	2495.95	2084.812	2999.2	2657.512	2711.518	2222.77	2246.938	2081.798	2375.046	2258.314
	ms	141.586	646.472	710.772	687.4	701.6	687.716	687.684	684.438	691.584	519.016	527.906



A.9 pav. Darbų pabaigos vidurkio priklausomybė nuo iteracijų skaičiaus („darbai10-5.txt“)



A.10 pav. Darbų pabaigos vidurkio priklausomybė nuo draudimų sąrašo ilgio („darbai10-5.txt“)

B PRIEDAS. TYRIMUI NAUDOTI DUOMENŲ FAILAI

Failas „darbai3-3.txt”

```
masinos3.txt
d1 1op 1 0 55 0 2op 2 0 40 0 3op 3 0 64 0
d2 1op 1 0 24 0 2op 2 0 12 0 3op 3 0 19 0
d3 1op 1 0 81 0 2op 2 0 90 0 3op 3 0 30 0
```

Failas „masinos3.txt”

```
m1 1 1 kalendorius0.txt
m2 2 1 kalendorius0.txt
m3 3 1 kalendorius0.txt
```

Failas „darbai4-4.txt”

```
masinos4.txt
d1 1op 1 0  3 0 2op 2 0 64 0 3op 3 0 92 0 4op 4 0 79 0
d2 1op 1 0 71 0 2op 2 0 63 0 3op 3 0 89 0 4op 4 0  2 0
d3 1op 1 0  4 0 2op 2 0 39 0 3op 3 0 31 0 4op 4 0 64 0
d4 1op 1 0 33 0 2op 2 0  3 0 3op 3 0 86 0 4op 4 0 65 0
```

Failas „masinos4.txt”

```
m1 1 1 kalendorius0.txt
m2 2 1 kalendorius0.txt
m3 3 1 kalendorius0.txt
m4 4 1 kalendorius0.txt
```

Failas „darbai6-6.txt”

```
masinos6.txt
d1 1op 1 0 1 0 2op 2 0 3 0 3op 3 0  6 0 4op 4 0  7 0 5op 5  3 0 6op 6 0 6 0
d2 1op 1 0 8 0 2op 2 0 5 0 3op 3 0 10 0 4op 4 0 10 0 5op 5 10 0 6op 6 0 4 0
d3 1op 1 0 5 0 2op 2 0 4 0 3op 3 0  8 0 4op 4 0  9 0 5op 5  1 0 6op 6 0 7 0
d4 1op 1 0 5 0 2op 2 0 5 0 3op 3 0  5 0 4op 4 0  3 0 5op 5  8 0 6op 6 0 9 0
d5 1op 1 0 9 0 2op 2 0 3 0 3op 3 0  5 0 4op 4 0  4 0 5op 5  3 0 6op 6 0 1 0
d6 1op 1 0 3 0 2op 2 0 3 0 3op 3 0  9 0 4op 4 0 10 0 5op 5  4 0 6op 6 0 1 0
```

Failas „masinos6.txt”

```
m1 1 1 kalendorius0.txt
m2 2 1 kalendorius0.txt
m3 3 1 kalendorius0.txt
m4 4 1 kalendorius0.txt
m5 5 1 kalendorius0.txt
m6 6 1 kalendorius0.txt
```

Failas „darbai10-5.txt”

```
masinos5.txt
d1  1op 1 0 72 0 2op 2 0 87 0 3op 3 0 95 0 4op 4 0 66 0 5op 5 0 60 0
d2  1op 1 0  5 0 2op 2 0 35 0 3op 3 0 48 0 4op 4 0 39 0 5op 5 0 54 0
d3  1op 1 0 46 0 2op 2 0 20 0 3op 3 0 21 0 4op 4 0 97 0 5op 5 0 55 0
d4  1op 1 0 59 0 2op 2 0 19 0 3op 3 0 46 0 4op 4 0 34 0 5op 5 0 37 0
d5  1op 1 0 23 0 2op 2 0 73 0 3op 3 0 25 0 4op 4 0 24 0 5op 5 0 28 0
d6  1op 1 0 28 0 2op 2 0 45 0 3op 3 0  5 0 4op 4 0 78 0 5op 5 0 83 0
d7  1op 1 0 53 0 2op 2 0 71 0 3op 3 0 37 0 4op 4 0 29 0 5op 5 0 12 0
d8  1op 1 0 12 0 2op 2 0 87 0 3op 3 0 33 0 4op 4 0 55 0 5op 5 0 38 0
d9  1op 1 0 49 0 2op 2 0 83 0 3op 3 0 40 0 4op 4 0 48 0 5op 5 0  7 0
d10 1op 1 0 65 0 2op 2 0 17 0 3op 3 0 90 0 4op 4 0 27 0 5op 5 0 23 0
```

Failas „masinos5.txt”

```
m1 1 1 kalendorius0.txt
m2 2 1 kalendorius0.txt
m3 3 1 kalendorius0.txt
m4 4 1 kalendorius0.txt
m5 5 1 kalendorius0.txt
```

Failas „kalendorius0.txt”

```
1 1 1 1 1 1 1
-1
```

C PRIEDAS. PROGRAMOS APRAŠYMAS

Failas „Unit1.h“

```
//---------------------------------------------------------------------------
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Forms.hpp>
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Dialogs.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:   // IDE-managed Components
        TButton *Button1;
        TButton *Button2;
        TButton *Button3;
        TButton *Button5;
        TOpenDialog *OpenDialog1;
        TButton *Button4;

        //-- Iskviecia forma masinu duomenu failams kurti bei redaguoti -------------
        void __fastcall Button1Click(TObject *Sender);

        //-- Uzdaro forma ------------------------------------------------------------
        void __fastcall Button2Click(TObject *Sender);

        //-- Iskviecia forma darbu duomenu failams kurti bei redaguoti --------------
        void __fastcall Button3Click(TObject *Sender);

        //-- Leidzia pasirinkti duomenu faila tvarkarascio sudarymui, iskviecia forma
        //-- tvarkarascio sudarymui --------------------------------------------------
        void __fastcall Button5Click(TObject *Sender);

        //-- Iskviecia forma kalendoriu duomenu failams kurti bei redaguoti ---------
        void __fastcall Button4Click(TObject *Sender);

private:       // User declarations
public:        // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Failas „Ivesti_kalendorius.h“

```
//---------------------------------------------------------------------------
#ifndef Ivesti_kalendoriusH
#define Ivesti_kalendoriusH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <string>
#include <fstream>
#include "iomanip.h"
using namespace std;
//---------------------------------------------------------------------------
class TForm5 : public TForm
{
__published:   // IDE-managed Components
        TMemo *Memo1;
        TLabel *Label1;
        TLabel *Label2;
        TComboBox *ComboBox1;
        TEdit *Edit1;
        TButton *Button1;
        TLabel *Label3;
        TButton *Button2;
        TButton *Button3;
        TButton *Button4;
        TButton *Button5;
        TPanel *Panel1;
        TCheckBox *CheckBox1;
        TCheckBox *CheckBox2;
        TCheckBox *CheckBox3;
        TCheckBox *CheckBox4;
        TCheckBox *CheckBox5;
        TCheckBox *CheckBox6;
        TCheckBox *CheckBox7;
        TLabel *Label4;
        TEdit *Edit2;
        TEdit *Edit3;
        TEdit *Edit4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TEdit *Edit5;
        TEdit *Edit6;
        TButton *Button6;
        TButton *Button7;
        TRadioGroup *RadioGroup1;
        TLabel *Label9;

        //-- Uzdaromas formos langas ---------------------------------------------
```

```
        void __fastcall Button5Click(TObject *Sender);

        //-- Nutraukiami vykdomi veiksmai ------------------------------------------
        void __fastcall Button4Click(TObject *Sender);

        //-- Forma paruosiama darbui -----------------------------------------------
        void __fastcall FormActivate(TObject *Sender);

        //-- Parodomas lange pasirinktas kalendoriaus failas -----------------------
        void __fastcall ComboBox1Click(TObject *Sender);

        //-- Forma paruosiama naujo kalendoriaus kurimui. Issaugomas sukurtas kalendorius
        void __fastcall Button1Click(TObject *Sender);

        //-- Formos komponentu savybiu keitimas, jei bus kuriamas naujas kalendorius
        void __fastcall Edit1Change(TObject *Sender);

        //-- Jei ivedami duomenys, aktyvinamas mygtukas intervalo saugojimui -------
        void __fastcall Edit4Change(TObject *Sender);

        //-- Forma paruosiama dienu redagavimui. Issaugomi pakeitimai --------------
        void __fastcall Button2Click(TObject *Sender);

        //-- Forma paruosiama nedarbingumo intervalo redagavimui. Issaugomi pakeitimai
        void __fastcall Button3Click(TObject *Sender);

        //-- Forma paruosiama naujo intervalo iterpimui. Issaugomi pakeitimai -----
        void __fastcall Button6Click(TObject *Sender);

        //-- Forma paruosiama intervalo istrynimui. Issaugomi pakeitimai -----------
        void __fastcall Button7Click(TObject *Sender);

private:          // User declarations
  string fv;      // failo pavadinimas;
  int n,          // intervalu skaicius
      k,          // irasomo intervalo numeris
      p;          // kintamasis, nurodantis pries ar po nurodyto intervalo iterpiamas naujas intervalas
  int dienos[7];  // masyvas informacijai apie dienas saugoti
  float pradzia,  // intervalo pradzia
        trukme,   // intervalo trukme
        pradzia2, // intervalo pradzia
        trukme2;  // intervalo trukme

//-- Sudaromas kalendoriu failu sarasas ------------------------------------
  void FailuSarasas();

//-- Atzymimi visi CkeckBox ------------------------------------------------
  void ParuostiCheckBox();

//-- Aktyvinami visi CheckBox ----------------------------------------------
  void EnablePanel();

//-- Visi CheckBox padaromi pasyvus ----------------------------------------
```

```
  void DisablePanel();

//-- Tikrinama, ar pazymeta bent viena diena -------------------------------
  bool TikrintiDienas();

//-- I masyva surasoma, kuriomis dienomis dirbama, o kuriomis ne -----------
  void Dienos();

//-- Faile issaugomos dienos -----------------------------------------------
//-- string fv - failo pavadinimas
//-- int n - nurodo nedarbingumo intervalu skaiciu.
  void SaugotiDienas(string fv, int n);

//-- Faile issaugomas intervalas -------------------------------------------
// string fv - falo pavadinimas
// float pr - intervalo pradzia
// float tr - intervalo trukme
  void SaugotiIntervala(string fv, float pr, float tr);

//-- I faila irasoma informacija apie dienas -------------------------------
//-- string fv - failo pavadinimas
  void RedaguotiDienas(string fv);

//-- Intervalu skaicius nurodytame faile -----------------------------------
// string fv - failo pavadinimas
// return int - intervalu skaicius
  int IntervaluSkaicius(string fv);

//-- Tikrinama, ar duotame faile egzistuoja nurodytas intervalas -----------
// string fv - failo pavadinimas
// float pradzia - intervalo pradzia
// float trukme - intervalo trukme
// return bool - true, jei toks intervalas yra, kitu atveju false
  bool IntervalasEgzistuoja(string fv, float pradzia, float trukme);

//-- Redaguojamas nurodytas intervalas -------------------------------------
// string fv - failo pavadinimas
// float pradzia - redaguojamo intervalo pradzia
// float trukme - redaguojamo intervalo trukme
// float pradzia2 - intervalo pradzia
// float trukme2 - intervalo trukme
  void RedaguotiIntervala(string fv, float pradzia, float trukme, float pradzia2, float trukme2);

//-- I nurodyta faila irasomas intervalas ----------------------------------
// string fv - failo pavadinimas
// float pradzia - intervalo pradzia
// float trukme - intervalo trukme
  void IrasytiIntervala(string fv, float pradzia, float trukme);

//-- Po ar pries nurodyta intervala iterpiamas naujas intervalas -----------
// string fv - failo pavadinimas
// float pradzia - intervalo, prie/po kurio iterpiamas naujas intervalas, pradzia
// float trukme - intervalo, prie/po kurio iterpiamas naujas intervalas, trukme
```

```
// float pradzia2 - naujo intervalo pradzia
// float trukme2 - naujo intervalo trukme
// int p - nurodo pries ar po nurodyto intervalo terpiamas naujas intervalas. 0 - pries, 1 - po
  void IterptiIntervala(string fv, float pradzia, float trukme, float pradzia2, float trukme2, int p);

//-- Is failo istrinamas nurodytas intervalas ------------------------------
// string fv - failo pavadinimas
// float pradzia - trinamo intervalo pradzia
// float trinamo intervalo trukme
// int n - intervalu skaicius
  void IstrintiIntervala(string fv, float pradzia, float trukme, int n);

public:        // User declarations
        __fastcall TForm5(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm5 *Form5;
//---------------------------------------------------------------------------
#endif
```

Failas „Ivesti_masinas.h“

```
//---------------------------------------------------------------------------
#ifndef Ivesti_masinasH
#define Ivesti_masinasH
//---------------------------------------------------------------------------
#include <Forms.hpp>
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <ExtCtrls.hpp>
#include <string>
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
//---------------------------------------------------------------------------
class TForm2 : public TForm
{
__published:   // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TEdit *Edit2;
        TEdit *Edit3;
        TEdit *Edit4;
        TButton *Button1;
        TButton *Button2;
        TComboBox *ComboBox1;
        TMemo *Memo1;
        TButton *Button3;
        TLabel *Label5;
```

```
        TEdit *Edit1;
        TButton *Button4;
        TButton *Button5;
        TLabel *Label6;
        TComboBox *ComboBox2;

        //-- Uzdaroma forma ------------------------------------------------------
        void __fastcall Button1Click(TObject *Sender);

        //-- Forma paruosiama naujos masinos ivedimui. Ivedama nauja masina ---------
        void __fastcall Button2Click(TObject *Sender);

        //-- Forma paruosiama darbui ---------------------------------------------
        void __fastcall FormActivate(TObject *Sender);

        //-- Lange parodomas pasirinktas duomenu failas ----------------------------
        void __fastcall ComboBox1Click(TObject *Sender);

        //-- Forma paruosiama esancios masinos duomenu keitimui ir, jei leidziama,
        //-- jie pakeiciami ------------------------------------------------------
        void __fastcall Button3Click(TObject *Sender);

        //-- Forma paruosiama naujo failo kurimui ----------------------------------
        void __fastcall Edit1Change(TObject *Sender);

        //-- Forma paruosiama masinai istrinti ir, jei leidziama, ji istrinama-------
        void __fastcall Button4Click(TObject *Sender);

        //-- Nutraukiami vykdyti veiksmai ------------------------------------------
        void __fastcall Button5Click(TObject *Sender);

private:       // User declarations
  string fv,       // failo pavadinimas
         pav;      // masinos pavadinimas
  int t;           // masinos tipas
  float s;         // masinos sparta
  string kal;      // masinos darbingumo kalendorius

//-- I nurodyta faila irasomi naujos masinos duomenys -----------------------
//-- string fv - failo pavadinimas
//-- string pav - masinos pavadinimas
//-- int t - masinos tipas
//-- float s - masinos sparta
//-- string kal - kalendoriaus failo pavadinimas
  void IvestiMasina(string fv, string pav, int t, float s, string kal);

//-- Jei leidziama, keiciami masinos parametrai -----------------------------
//-- string fv - failo pavadinimas
//-- string pav - masinos pavadinimas
//-- int t - masinos tipas
//-- string kal - kalendoriaus failo pavadinimas
  void Koreguoti(string fv, string pav, int t, float s, string kal);
```

```
//-- Tikrina, ar faile tokio tipo masina yra paskutine.
//-- string fv - failo pavadinimas
//-- string mpav - masinos pavadinimas
//-- return bool - true, jei nurodyta masina yra paskutine siame faile, kitu atveju - false
  bool Paskutine(string fv, string mpav);

//-- Tikrinama, ar sio tipo masina yra reikalinga kuriam nors darbui,
//-- aprasytam kuriame nors susietame faile, atlikti ------------------------
//-- string fv - failo pavadinimas
//-- string mpav - masinos pavadinimas
//-- return bool - true, jei nurodytos masinos tipas reikalingas, kitu atveju - false
  bool Naudojama(string fv, string mpav);

//-- Istrinama nurodyta masina ----------------------------------------------
//-- string fv - failo pavadinimas
//-- string pav - masinos pavadinimas
  void Trinti(string fv, string pav);

//-- Surandami ir i ComboBox surasomi visi masinu failai. --------------------
//-- ComboBox rodo nurodyta faila -------------------------------------------
//-- string f - failo pavadinimas
  void FailuSarasas(string failas);

public:        // User declarations
        __fastcall TForm2(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm2 *Form2;
//---------------------------------------------------------------------------
#endif
```

Failas „Ivesti_darbus.h“

```
//---------------------------------------------------------------------------
#ifndef Ivesti_darbusH
#define Ivesti_darbusH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <string>
using namespace std;

// saugoma programos direktorija
const  AnsiString duomenu_dir = GetCurrentDir() + "\\Duomenys\\";
//---------------------------------------------------------------------------
class TForm3 : public TForm
{
__published:   // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
```

```
TLabel *Label3;
TLabel *Label4;
TLabel *Label6;
TLabel *Label7;
TEdit *Edit2;
TEdit *Edit3;
TEdit *Edit4;
TButton *Button1;
TButton *Button2;
TEdit *Edit7;
TLabel *Label8;
TComboBox *ComboBox1;
TComboBox *ComboBox2;
TLabel *Label9;
TComboBox *ComboBox3;
TEdit *Edit1;
TLabel *Label10;
TMemo *Memo1;
TButton *Button3;
TButton *Button5;
TButton *Button7;
TRadioGroup *RadioGroup2;
TLabel *Label5;
TEdit *Edit5;

//-- Uzdaroma forma ------------------------------------------------------
void __fastcall Button1Click(TObject *Sender);

//-- Forma paruosiama naujo darbo ivedimui. Ivedamas naujas darbas ----------
void __fastcall Button2Click(TObject *Sender);

//-- Forma paruosiama darbui ---------------------------------------------
void __fastcall FormActivate(TObject *Sender);

//-- Parodomas lange pasirinktas darbu failas ir pagal faile nurodyta masinu
//-- faila suformuojamas masinu tipu sarasas -----------------------------
void __fastcall ComboBox1Click(TObject *Sender);

//-- Forma paruosiama, jei kuriamas naujas failas ------------------------
void __fastcall Edit1Change(TObject *Sender);

//-- Forma paruosiama darbo istrynimui ir istrinamas darba ---------------
void __fastcall Button3Click(TObject *Sender);

//-- Forma paruosiama operacijos redagavimui. Redaguojama operacija ---------
void __fastcall Button5Click(TObject *Sender);

//-- Nutaukiami vykdomi veiksmai -----------------------------------------
void __fastcall Button7Click(TObject *Sender);

//-- Mygtukas aktyvuojamas, jeigu ivedamas naujas darbas -----------------
void __fastcall Edit3Change(TObject *Sender);
```

```
private:       // User declarations
  int n,         // naujo darbo operaciju skaicius
      k;         // irasomos operacijos numeris
  string fv,     // failo pavadinimas
         pav,    // darbo pavadinimas
         mpav,   // masinos pavadinimas
         op_pav; // operacijos pavadinimas
  int m,         // reikalingos masinos tipas
      nutr;      // nurodoma, ar operacija gali buti nutraukta
  float t;       // darbo trukme
  float  a;      // laikas, po kurio gali buti pradeta vykdyti kita operacija
  int* dTipai;   // masyvas operacijoms apdoroti reikalingu masinu tipams saugoti

//-- I faila, jei reikia, irosomas darbo pavadinimas bei duomenys apie operacija
//-- string fv - failo pavadinimas
//-- string pav - darbo pavadinimas
//-- string op_pav - operacijos pavadinimas
//-- int b - kintamasis, nurodantis, ar reikia rasyti darbo pavadinima, ar reikia kursoriu kelti i
nauja eilute;
//--   0 - pirmoji operacija (yra daugiau nei viena oparacija); rasomas darbo pavadinimas
//--   2 - paskutine operacija (yra daugiau nei viena operacija); kursorius keliamas i nauja eilute
//--   3 - vienintele operacija; rasomas darbo pavadinimas, kursorius keliamas i nauja eilute
//-- int mtipas - operacijai apdoroti reikalingas masinos tipas
//-- int nutr - nurodoma, ar operacija gali buti nutraukta; 0 - nenutraukiama, 1 - nutraukiama
//-- float trukme - operacijos trukme, jei ja apdorotu kanonine masina
//-- float pers -- laikas, po kurio gali buti pradeta vykdyti kita operacija
  void IvestiOperacija(string fv, string pav, string op_pav, int b, int mtipas, int nutr, float trukme,
float pers);

//-- Formuojamas nurodytame masinu duomenu faile esanciu masinu tipu sarasas
  void FormuotiTipuSarasa();

//-- Istrinamas darbas ------------------------------------------------------
//-- string fv - failo pavadinimas
//-- string pav - darbo pavadinimas
  void IstrintiDarba(string fv, string pav);

//-- Redaguojama nurodyto darbo nurodyta operacija --------------------------
//-- string fv - failo pavadinimas
//-- string pav - darbo pavadinimas
//-- string op - operacijos pavadinimas
//-- int nutr - nurodoma, ar operacija gali buti nutraukta; 0 - nenutraukiama, 1 - nutraukiama
//-- float t - operacijos trukme, jei ja apdorotu kanonine masina
//-- float s -- laikas, po kurio gali buti pradeta vykdyti kita operacija
  void RedaguotiOperacija(string fv, string pav, string op, int nutr, float t, float s);

//-- Surandami ir i ComboBox surasomi visi darbu failai. --------------------
//-- ComboBox rodo nurodyta faila -------------------------------------------
//-- string f - failo pavadinimas
  void FailuSarasas(string failas);

//-- I masyva surasomi masinu tipai, reikalingi darbams atlikti -------------
//-- string fv - failo pavadinimas
```

```
  void DarboTipai(string fv);

public:        // User declarations
        __fastcall TForm3(TComponent* Owner);

//-- Grazina eiluciu skaiciu faile ------------------------------------------
//-- string failas - failo pavadinimas
//-- return int - eiluciu skaicius faile
  int EilSk(string failas);

//-- Grazina nurodytame faile esanciu operaciju skaiciu ---------------------
//-- string failas - failo pavadinimas
//-- return int - operaciju skaicius
  int OperacijuSk(string failas);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm3 *Form3;
//---------------------------------------------------------------------------
#endif
```

Failas „tvarkarastis.h“

```
//---------------------------------------------------------------------------
#ifndef tvarkarastisH
#define tvarkarastisH
#include <math.h>
#include "CSPIN.h"
#include <Classes.hpp>
#include <Controls.hpp>
#include <ExtCtrls.hpp>
#include <StdCtrls.hpp>
#include <iostream>
#include <list>
#include <vector>
#include <algorithm>
using namespace std;
randomize();
//---------------------------------------------------------------------------

// grafiko vidinio remelio nuotolis nuo isorinio remelio
const int krastas = 30;

// Bolcmano konstanta
const double Bolc_konst = 1.3806505*pow(10,-23);

const int lv_valandoje = 6; // laiko vienetai valandoje
const int v_par = 24;       // valandu paroje
const int p_sav = 7;        // paru savaiteje
const string tvf = "Tvarkarastis.txt";
const string anf = "Analizes rezultatai.txt";
//---------------------------------------------------------------------------

// struktura tabu sarasui saugoti (2 darbu numeriai ir cechas(etapas))
```

```
struct pora
{
 int pora[3];
};
//---------------------------------------------------------------------------

class AAA
{
   public:
      int* sekos;  // suplanuotu darbu seka
      int* m_apk;  // nurodo, kurios masinos kuriuos darbus aptarnauja
      int gylis;   // nurodo, kiek darbu jau suplanuota
      float lb;    // virsunes apatinis rezis

//-- Konstruktorius -------------------------------------------------------
      AAA(){};

//-- Virsunes kopijavimo konstruktorius -----------------------------------
// const AAA &copyin - kopijuojama virsune
      AAA(const AAA &);

//-- Destruktorius --------------------------------------------------------
      ~AAA(){};

//-- Virsunes priskyrimo operatorius --------------------------------------
// const AAA &rhs - priskiriamas virsune
      AAA &operator=(const AAA &rhs);
};

// dalinius sprendinius surikiuoja didejancia tvarka pagal apatinius rezius
// (medzio nagrinejimas platyn)
struct Sort
{
  bool operator () (const AAA & lhs , const AAA & rhs) const {
    if( lhs.lb == rhs.lb && lhs.gylis > rhs.gylis) return true;
    if( lhs.lb < rhs.lb ) return true;
    return false;
  }
};

// dalinius sprendinius surikiuoja mazejancia tvarka pagal gyli
// (madzio nagrinejimas gilyn)
struct Sort2
{
  bool operator () (const AAA & lhs , const AAA & rhs) const {
    if( lhs.gylis == rhs.gylis && lhs.lb < rhs.lb) return true;
    if( lhs.gylis > rhs.gylis ) return true;
    return false;
  }
};
//---------------------------------------------------------------------------

class TForm4 : public TForm
```

```
{
__published:   // IDE-managed Components
        TButton *Button1;
        TButton *Button2;
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label6;
        TScrollBar *ScrollBar1;
        TScrollBar *ScrollBar2;
        TImage *Image1;
        TImage *Image2;
        TCSpinEdit *CSpinEdit1;
        TCSpinEdit *CSpinEdit2;
        TGroupBox *GroupBox1;
        TLabel *Label3;
        TEdit *Edit1;
        TLabel *Label4;
        TEdit *Edit2;
        TLabel *Label5;
        TEdit *Edit3;
        TMemo *Memo1;
        TRadioGroup *RadioGroup1;
        TButton *Button3;
        TLabel *Label9;
        TEdit *Edit6;
        TComboBox *ComboBox1;
        TLabel *Label7;

       //-- Patikrinama, ar duomenys yra korektiski; optimizuotas tvarkarastis -----
       //-- issaugomas faile ir pavaizduojamas ekrane ------------------------------
        void __fastcall Button1Click(TObject *Sender);

       //-- Uzdaroma forma ---------------------------------------------------------
        void __fastcall Button2Click(TObject *Sender);

       //-- Forma paruosiama darbui ------------------------------------------------
        void __fastcall FormActivate(TObject *Sender);

       //-- Kvieciamas metodas grafikui braizyti -----------------------------------
        void __fastcall ScrollBar1Change(TObject *Sender);

       //-- Kvieciamas metodas grafikui braizyti -----------------------------------
        void __fastcall ScrollBar2Change(TObject *Sender);

       //-- Kvieciamas metodas grafikui braizyti -----------------------------------
        void __fastcall CSpinEdit1Change(TObject *Sender);

       //-- Kvieciamas metodas grafikui braizyti -----------------------------------
        void __fastcall CSpinEdit2Change(TObject *Sender);

       //-- Reguliuoja, etikeciu antrastes ir laukeliu aktyvuma pagal tai, ---------
       //-- kuris metodas yra pasirinktas ------------------------------------------
        void __fastcall RadioGroup1Click(TObject *Sender);
```

```
        //-- Sudaromas nurodytas kiekis tvarkarasciu, apibendrinami gauti rezultatai,
        //-- issaugomi faile ir parodomi ekrane ------------------------------------
         void __fastcall Button3Click(TObject *Sender);

        //-- Ekrane parodomas ComboBox'e pasirinktas failas -------------------------
         void __fastcall ComboBox1Click(TObject *Sender);

        //-- Naikinami visi sudaryti masyvai ----------------------------------------
         void __fastcall FormClose(TObject *Sender, TCloseAction &Action);

private:       // User declarations
  int masinu_sk;               // masinu skaicius
  int darbu_sk;                // darbu skaicius
  int* masinu_kiekisEt;        // masinu kiekis etapuose
  AAA best;                    // geriausias rastas sprendinys (saku ribu metodui)
  vector<AAA> nodeList;        // daliniu sprendiniu sarasas (saku ribu metodui)
  float* tvarka;               // tvarkarastis masinu atzvilgiu. Saugojam operaciju apdorojimo pradzios
ir pabaigos
  int* d_tvarka;               // tvarkarastis masinu atzvilgiu. Saugojam darbu ir operaciju indeksai
  int* apt_nr;                 // nurodo kiekvienos masinos apdorojamu operaciju skaiciu
  float makespan;              // sudaryto tvarkarascio darbu apdorojimo pabaigos laiko momentas
  int opSk;                    // operaciju skaicius
  string *darbu_pav;           // darbu pavadinimai
  string *op_pav;              // operaciju pavadinimai
  int *darbu_tipai;            // darbams atlikti reikalingi masinu tipai
  float *darbu_trukmes;        // operaciju trukmes
  float *darbu_prsdgms;        // nurodoma, po kiek laiko gali buti pradeta vykdyti kita operacija
  int *darbu_nutr;             // nurodoma, ar operacija nutraukiama, ar ne
  string *masinu_pav;          // masinu pavadinimai
  int *masinu_tipai;           // masinu tipai
  float *masinu_sparta;        // masinu spartos
  float *vid_sparta;           // vidutines masinu spartos etapuose
  int *masinos_surik;          // indeksu masyvas
  int* m_kal;                  // kalendoriu indeksai
  vector<string> kal_failai;   // kalendoriu failu pavadinimai
  vector<float> nedarb_pr;     // nedarbingumo intervalu pradzios
  vector<float> nedarb_trukme; // nedarbingumo intervalu trukmes
  int *nedarb_int_sk;          // nedarbingumo intervalu skaicius
  int *dienos;                 // informacija apie dienas
  string dir;                  // failu direktorija

//-- I ComboBox surasomi darbu, masinu ir kalendoriu failai -----------------
  void FailuSarasas(string mfailas);

//-- Suranda ir grazina siame ceche (etape) nesuplanuotu darbu sarasa
// int* darbuSeka - suplanuotu darbu seka
// int etapas - nurodo, kuriam cechui (etapui) ieskomi nesuplanuoti darbai
// int kiek - jau suplanuotu darbu skaicius siame ceche (etape)
// return int* - siame ceche (etape) nesuplanuotu darbu sarasa
  int* NepriskirtiDarbai(int* darbuSeka, int etapas, int kiek);

//-- Padaroma duotos darbu sekos kopija -------------------------------------
```

```
// int* darbuSekos - duota darbu seka
// int etapas - nurodo, kuriam cechui (etapui) planuojami darbai
// int darbas - nurodo, kiek darbu suplanuota dabartiniam ceche (etape)
// return int* - darbu sekos kopija
  int* DarbuSekosKopija(int* darbuSeka, int etapas, int darbas);

//-- Tvarkarastis optimizuojamas saku ir ribu metodu ------------------------
  void SakuRibu();

//-- Atliekama virsunes sakojimo operacija ----------------------------------
// int* darbuSekos - iki siol suplanuotu darbu seka
// int* m_apk - nurodo, kurios masinos apdoroja suplanuotus darbus
// int virsunesGylis - nurodo, kiek darbu jau suplanuota
  void SakojimoOperacija(int* darbuSeka, int* m_apk, int virsunesGylis);

//-- Sudaromas tvarkarastis, grazinamas darbu pabaigos laiko momentas -------
// int* seka - suplanuotu darbu seka
// int* m_apk - nurodo, kurios masinos apdoroja suplanuotus darbus
// return float - sudaryto tvarkarascio darbu apdorojimo pabaigos laiko momentas arba NULL,
// jei tvarkarastis nesudarytas
  float SudarytiGrazintiPabaigaBB(int* seka, int* m_apk);

//-- Sudaromas tvarkarastis, kai duota darbu seka ---------------------------
// int* seka -iki siol sudarytu darbu seka
// float* darbu_pr - operaciju apdorojimo pradzios laiko momentai
// float* darbu_pb - operaciju apdorojimo pabaigos laiko momentai
// int* masinu_apkrautumas - nurodo, kurios masinos apdoroja iki siol suplanuotus darbus
// float* masinu_uzimtumas - nurodo, kada masinos gali anksciausiai pradeti aptarnauti
// int gylis - nurodo, kiek darbu jau suplanuota
// return bool - nurodo, ar pavyko sudaryti tvarkarasti. true- jei taip, false - kitu atveju
  bool SudarytiTvarkarastiBB(int *seka, float* darbu_pr, float* darbu_pb, int* masinu_apkrautumas,
float* masinu_uzimtumas, int gylis);

//-- Apskaiciuojamas virsunes apatinis rezis --------------------------------
// int* darbuSekos - iki siol suplanuotu darbu seka
// int* m_apk - nurodo, kurios masinos apdoroja suplanuotus darbus
// int gylis - nurodo, kiek darbu jau suplanuota
// return float - apskaiciuotas apatinis rezis
  float LB(int* darbuSekos, int* m_apk, int gylis);

//-- Sukuriami masyvai masinu kiekiui etapuose ir vidutinei masinu spartai
//-- etapuose saugoti -------------------------------------------------------
  void MasinuKiekisEtapuose();

//-- Apskaiciuojama galima operacijos apdorojimo pradzia --------------------
// int* seka - iki siol sudarytu darbu seka
// float* darbu_pr - operaciju apdorojimo pradzios laiko momentai
// float* darbu_pb - operaciju apdorojimo pabaigos laiko momentai
// int* masinu_apkrautumas - nurodo, kurios masinos apdoroja iki siol suplanuotus darbus
// float* masinu_uzimtumas - nurodo, kada masinos gali anksciausiai pradeti aptarnauti
// int i - etapas
// int darbo_nr - darbo numeris
```

```
  float GalimaPradzia(int *seka, float* darbu_pr, float* darbu_pb, int* masinu_apkrautumas, float*
masinu_uzimtumas, int etapas, int darbo_nr);

//-- Apskaiciuojama nedarbingumo intervalu, esanciu tarp nurodytu laiko momentu, ilgiu suma
// float pradzia - pradzios laiko momentas
// float pabaiga - pabaigos laiko momentas
// int m_nr - masinos numeris
// return float - intervalu ilgiu suma
  float IntervaluIlgis(float pradzia, float pabaiga, int m_nr);

//-- Padaroma masinu, apdorojanciu suplanuotus darbus, masyvo kopija --------
// int* darbuSeka - duota darbu seka
// int* m_apk - masinos, apdorojancios suplanuotus darbus
// int etapas - nurodo, kuriam cechui (etapui) planuojami darbai
// int darbas - nurodo, kiek darbu suplanuota dabartiniam ceche (etape)
// return int* - masinu, apdorojanciu suplanuotus darbus, masyvo kopija
  int* MasinuApkrautumoKopija(int* darbuSeka, int* m_apk, int etapas, int darbas);

//-- Is nurodyto failo skaitoma informacija apie masinas, sukuriami masyvai
//-- informacijai apie masinas saugoti. I vektoriu surasomi naudojamu
//-- kalendoriu failu pavadinimai ------------------------------------------
// string fm - masinu failo pavadinimas
  void SkaitytiMasinas(string fm);

//-- Is nurodyto failo skaitoma informacija apie darbus, sukuriami masyvai
//-- informacijai apie darbus saugoti. Grazina susijusio masinu failo varda
// string fd - darbu failo pavadinimas
// return string - masinu failo pavadinimas
  string SkaitytiDarbus(string fd);

//-- Tvarkarastis optimizuojamas modeliuojamo atkaitinimo metodu ------------
// return int* - optimalaus tvarkarascio darbu seka arba NULL, jei ji nesudaryta
  int* Atkaitinimas(int* darbu_seka);

//-- Grazinamas gretimas sprendinys duotajam --------------------------------
// return int* - darbu seka
  int* GretimasSpr(int *darbu_seka);

//-- Grazinama temperatura --------------------------------------------------
// return float - temperatura
  float Temp(float t, float td);

//-- Grazinama tikimybe -----------------------------------------------------
// return float - tikimybe
  float Tikimybe(float Bpabaiga, float Npabaiga, float T);

//-- Braizomas tvarkarascio grafikas ir nedarbingumo intervalai ------------
  void Grafikas();

//-- Tikrina ar duomenys yra korektiski, t.y. ar masinu faile yra visi
//-- darbams atlikti reikalingi masinu tipai --------------------------------
// return bool - true, jei duomenys geri, false kitu atveju
  bool GeriDuomenys();
```

```
//-- Spausdinami analizes rezultatai ----------------------------------------
// int n - kiek kartu sudarytas tvarkarastis
// vector<float> rez - sudarytu tvarkarasciu darbu apdorojimo pabaigos laiko momentai
// vector<float> daznis - daznis
// float vid - empirinis vidurkis
// float disp - empirine dispersija
// long int uztruko - per kiek laiko (milisekundemis) buvo sudarytas vienas tvarkarastis
  void SpausdintiAnalizesRez(int n, vector<float> rez, vector<float> daznis, float vid, float disp,
long int uztruko);

//-- Grazina spalvos koda ---------------------------------------------------
  int Spalva(int darbu_sk, int d_nr);

//-- Tvarkarastis optimizuojamas tabu paieskos metodu -----------------------
// return int* - optimalaus tvarkarascio darbu seka arba NULL, jei ji nesudaryta
  int* Tabu(int* darbu_seka);

//-- Grazinamas poru skaicius (deriniai is n po 2 kiekvienam etapui) --------
// int n - darbu skaicius
// int s - etapu skaicius
// return int - poru skaicius
  int PoruSkaicius(int n, int s);

//-- Sudaroma darbu seka, pagal trumpiausia operaciju apdorojimo laika
//-- (ShortestSprocessinTime taisykle) --------------------------------------
// return int* - sudaryta darbu seka
  int* TrumpiausiasApdorojimoLaikas();

//-- Masinos surikiuojamos pagal tipa ---------------------------------------
// int* masinu_tipai - masinu tipu masyvas
  void SurikiuotiPagalTipa(int* masinu_tipai);

//-- Pagal suplanuotus darbus sudaromas tvarkarastis, parenkant tas masinas, kurios
//-- greiciausiai apdoroja operacija ----------------------------------------
// int *seka - suplanuotu darbu seka
// float* darbu_pr - operaciju apdorojimo pradzios laiko momentai
// float* darbu_pb - operaciju apdorojimo pabaigos laiko momentai
// int* masinu_apkrautumas - kada masina gali anksciausiai pradeti operacijos apdorojima
// return bool - true, jei tvarkarasti pavyko sudaryti, false - kitu atveju
  bool SudarytiTvarkarasti(int *seka, float* darbu_pr, float* darbu_pb, int* masinu_apkrautumas);

//-- Grazinama visu darbu apdorojimo pabaigos laiko momentas ----------------
// float* darbu_pb - operaciju apdorojimo pabaigos laiko momentai
// return float - visu darbu apdorojimo pabaigos laiko momentas
  float DarbuPabaiga(float* darbu_pb);

//-- Kuriami masyvai informacijai apie kalendorius saugoti ------------------
  void SkaitytiKalendorius();

//-- Generuojamas pradinis sprendinys ---------------------------------------
// float int* - darbu seka
  int* PradinisSpr();
```

```
//-- Sudaromas tvarkarastis. Grazinamas darbu apdorojimo pabaigos laiko momentas arba NULL
// int* seka - suplanuotu darbu seka
// return pabaiga - darbu apdorojimo pabaigos laiko momentas arbe NULL, jei
// tvarkarastis nesudarytas
  float SudarytiGrazintiPabaiga(int* seka);

//-- Sudaromas tvarkarastis, vaizduojamas ekrane, atspausdinamas faile,
//-- grazinamas visu darbu apdorojimo pabaigos laiko momentas ---------------
// int* seka - suplanuoti darbai
// return float - visu darbu apdorojimo pabaigos laiko momentas arba NULL,
// jei tvarkastis nesudarytas
  float SudarytiSpausdintiVaizduotiGrazinti(int* seka);

//-- Sudaromas tvarkarastis, vaizduojamas ekrane, atspausdinamas faile,
//-- grazinamas visu darbu apdorojimo pabaigos laiko momentas ---------------
// int* seka - suplanuoti darbai
// int* m_apk - masinos, kurios apdoroja paskirtus darbus
// return float - visu darbu apdorojimo pabaigos laiko momentas arba NULL,
// jei tvarkastis nesudarytas
  float SudarytiSpausdintiVaizduotiGrazinti(int* seka, int* m_apk);

//-- Spausdinamas sudarytas tvarkarastis masinu atzvilgiu -------------------
// int *seka - suplanuotu darbu seka
// float* darbu_pb - operaciju apdorojimo pabaigos laiko momentai
// float* tvarka - tvarkarastis masinu atzvilgiu (operaciju apdorojimo pradzios ir pabaigos laiko
momentai)
// int* d_tvarka - tvarkarastis masinu atzvilgiu (darbu ir operaciju indeksai)
// int* apt_nr - nurodo, kiek kuri masina apdoroja operaciju
  void Spausdinti(int *seka, float* darbu_pb, float* tvarka, int* d_tvarka, int* apt_nr);

//-- Spausdinamas ir vaizduojamas sudarytas tvarkarastis --------------------
// int* seka - siplanuotu darbu seka
// float* darbu_pr - operaciju apdorojimo pradzios laiko momentai
// float* darbu_pb - operaciju apdorojimo pabaigos laiko momentai
// int* masinu_apkrautumas - masinos, kurios apdoroja paskirtus darbus
  void SpausdintiVaizduotiTvarkarasti(int *seka, float* darbu_pr, float* darbu_pb, int*
masinu_apkrautumas);

//-- Tvarkarastis sudaromas masinu atzvilgiu --------------------------------
// int* seka - siplanuotu darbu seka
// float* darbu_pr - operaciju apdorojimo pradzios laiko momentai
// float* darbu_pb - operaciju apdorojimo pabaigos laiko momentai
// int* m_ap - masinos, kurios apdoroja paskirtus darbus
// float* tvarka - tvarkarastis masinu atzvilgiu (operaciju apdorojimo pradzios ir pabaigos laiko
momentai)
// int* d_tvarka - tvarkarastis masinu atzvilgiu (darbu ir operaciju indeksai)
// int* apt_nr - nurodo, kiek kuri masina apdoroja operaciju
  void TvarkarastisMasinuAtz(int *seka, float* darbu_pr, float* darbu_pb, int* m_ap, float* tvarka,
int* d_tvarka, int* apt_nr);

//-- Apdorojanciu masinu skaicius -------------------------------------------
// return int - masinu, kurioms paskirta aptarnauti bent viena operacija, skaicius
```

```
  int DirbanciuMasinuSkaicius();

//-- Braizomi nedarbingumo intervalai ---------------------------------------
// int m_nr - masinos numeris
// float pradzia - laiko momentas, nuo kurio vaizduojami intervalai
// float pabaiga - laiko momentas, iki kurio vaizduojami intervalai
// float plotis - laiko vieneto plotis
// float aukstis - intervalo aukstis
// int dm - vaizduojamos masinos vieta
  void VaizduotiNedarbingumoIntervalus(int m_nr, float pradzia, float pabaiga, float plotis, float
aukstis, int dm);

//-- Braizomas vienas nedarbingumo intervalas -------------------------------
// float x1 - intervalo (staciakampio) pradzia
// float y1 - intervalo (staciakampio) virsus
// float x2 - intervalo (staciakampio) pabaiga
// float y2 - intervalo (staciakampio) apacia
  void VaizduotiIntervala(float x1, float y1, float x2, float y2);

//-- Sugraduoja vaizduojama tvarkarasti
// int p1 - laiko momentas, nuo kurio graduojama
// int hkiek - laiko vienetu skaicius
// float plotis - laiko vieneto plotis
// int vkiek - vaizduojamu masinu skaicius
// float aukstis - vienai masinai skirtas aukstis
  void Graduoti(int p1, int hkiek, float plotis, int vkiek, float aukstis);

public:        // User declarations
        __fastcall TForm4(TComponent* Owner);
   string dfailas;   // pilnas darbu failo pavadinimas (su direktorija)
};
//---------------------------------------------------------------------------
extern PACKAGE TForm4 *Form4;
//---------------------------------------------------------------------------
#endif
```