KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Gediminas Kairaitis

TVARKARAŠČIŲ SUDARYMO UŽDAVINIŲ IR JŲ ALGORITMŲ TYRIMAS

Magistro darbas

Vadovas
doc. dr. N. Listopadskis

KAUNAS, 2010

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. N. Listopadskis
2010 06 02

TVARKARAŠČIŲ SUDARYMO UŽDAVINIŲ IR JŲ ALGORITMŲ TYRIMAS

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
doc. dr. N. Listopadskis
2010 06 02

Recenzentas
doc.dr. D. Rubliauskas
2010 06 02

Atliko
FMMM 8 gr. stud.
G. Kairaitis
2010 06 02

KAUNAS, 2010

KVALIFIKACINĖ KOMISIJA

Pirmininkas: Leonas Saulis, profesorius (VGTU)

Sekretorius: Eimutis Valakevičius, docentas (KTU)

Nariai: Algimantas Jonas Aksomaitis, profesorius (KTU)

Vytautas Janilionis, docentas (KTU)

Vidmantas Povilas Pekarskas, profesorius (KTU)

Rimantas Rudzkis, habil. dr., vyriausiasis analitikas (DnB NORD Bankas)

Zenonas Navickas, profesorius (KTU)

Arūnas Barauskas, dr., vice-prezidentas projektams (UAB „Baltic Amadeus“)

Kairaitis G. Tvarkaraščių sudarymo uždavinių ir jų algoritmų tyrimas: taikomosios matematikos magistro darbas / darbo vadovas doc. dr. N. Listopadskis, taikomosios matematikos katedra, fundamentaliųjų mokslų fakultetas, Kauno technologijos universitetas, Kaunas, 2010. – 85 p.

SANTRAUKA

Tvarkaraščių sudarymo uždaviniai – viena iš sunkiau sprendžiamų problemų, kylančių įvairiose gamybinėse struktūrose, grupė. Darbo pradžioje supažindinama su bendrais tvarkaraščių sudarymo uždavinių bruožais ir jų sprendimo algoritmais. Detaliau nagrinėti šiame darbe parenkamas vienas sunkiausių gamybinių tvarkaraščių ir apskritai kombinatorinių optimizavimo uždavinių – darbo fabriko uždavinys (angl. job shop scheduling problem), kuris be abejo nėra tiksliai sprendžiamas per polinominį sprendimo laiką. Šio uždavinio pradiniai duomenys yra duotos darbų ir įrenginių aibės. Kiekvienas darbas apdorojamas specifine įrenginių tvarka. Uždavinio tikslas – minimizuoti visų darbų atlikimo laiką.

Šiam uždaviniui spręsti pristatėme du apytikslius tabu – atkaitinimo modeliavimo bei paieškos kintamose aplinkose algoritmus, priklausančius metaeuristinių metodų šeimai. Iš tabu – atkaitinimo modeliavimo galima nesunkiai gauti paprastą tabu paiešką, tad prie dviejų minėtų algoritmų galima pridėti ir paprastąją tabu paiešką. Šiame darbe atlikta minėtų algoritmų programinė realizacija. Pristatytų algoritmų efektyvumui įvertinti ir algoritmų parametrų parinkimo rekomendacijoms pateikti, buvo pasirinkti gerai literatūroje žinomi bei sunkiau sprendžiami etaloniniai darbo fabriko uždavinių pavyzdžiai. Darbo pabaigoje pateikiamos minėtų algoritmų parametrų parinkimo rekomendacijos ir aptariamas algoritmų efektyvumas, kuris nagrinėtuose uždaviniuose nebuvo pastovus minėtų trijų algoritmų atvejais, t.y. nebuvo tokio algoritmo, kuriuo būtų visada gaunami geresni rezultatai už kitų visuose nagrinėtuose uždaviniuose ir išryškėjo uždaviniai, kurie palankesni vienam ar kitam algoritmui, tačiau nagrinėjant vidutinį efektyvumą, geresnis jis buvo gaunamas su tabu – atkaitinimo modeliavimo ir paprastos tabu paieškos algoritmų kombinacija (spręstus uždavinius priskiriant vienam iš algoritmų) nei paieškos kintamose aplinkose algoritmo atveju.

Kairaitis G. Analysis of scheduling problems and their algorithms: Master‘s work in applied mathematics / supervisor dr. assoc. prof. N. Listopadskis; Department of Applied mathematics, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2010. – 85 p.

SUMMARY

At the beginning of this work we introduce to the combinatorial optimization, scheduling problems and methods used to solve them. In computer science scheduling problems is considered strongly NP-complete.

The combinatorial optimization problem considered in this paper is a static job shop problem scheduling arising in the manufacturing processes. In the static job shop scheduling problem, a finite number of jobs are to be processed by a finite number of machines. Each job consists of a predermination sequence of task operations, each of which needs to be processed without preemption for a given period of time on a given machine. Tasks of the same job cannot be processed concurrently and each job must visit each machine exactly once. A schedule is an assignment of operation to time slots on a machine. The makespan is the maximum completion time of the jobs and the objective of the job shop scheduling problem is to find a schedule that minimizes the makespan. When the size of problem increases, the computational time of the exact methods grows exponentially. Therefore, the recent research on job shop and other scheduling problems is focused on heuristic algorithms.

We also presented some meta-heuristic algorithms such as Tabu search – Simulated annealing (TS/SA), Tabu Search (TS), Variable Neighborhood Search (VNS) and showed their results on some job shop instances. At the end of this work we tell recommendations about choosing suitable parameters.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

1. ĮVADAS

Sprendžiant projektavimo ir vadybos uždavinius dažnai susiduriama su įvairiomis tvarkaraščių sudarymo ir kalendorinio planavimo problemomis. Tokių uždavinių pradiniai duomenys yra tam tikra darbų, susijusių nuoseklumo sąryšiais aibė. Darbai atliekami per tam tikrą laiką, sunaudojant jiems atlikti reikalingus išteklius. Ištekliai ir laikas, reikalingas projektui įgyvendinti, gali būti riboti. Reikia rasti tokį darbų atlikimo tvarkaraštį, kuris tenkintų nuoseklumo sąryšius, minimizuotų išteklius pagal tam tikrus kriterijus. Toks kriterijus dažniausiai yra laikas, reikalingas visiems darbams atlikti.

1.1 SPRENDŽIAMI UŽDAVINIAI IR TIKSLAI

Šiame darbe nagrinėsime vieną iš gamybinių tvarkaraščių sudarymo uždavinių. Visuose gamybinių tvarkaraščių sudarymo uždavinių pagrindiniai pradiniai duomenys – reikiamų atlikti darbų aibė ir įrenginių aibė, kuriais atliekami minėtieji darbai. Tarsime, kad abi šios aibės yra fiksuoto dydžio. Kiekvienam iš darbų galutinai užbaigti, dažniausiai reikia skirti tam tikros apibrėžtos įrenginių sekos darbo laiko ar kitokius išteklius. Tuomet pagrindinis gamybinių tvarkaraščių sudarymo uždavinių tikslas yra priskirti visus darbus įrenginiams taip, kad būtų minimizuojami sunaudoti ištekliai.

Šio darbo pagrindiniai tikslai yra:

- Atlikti keleto gamybinių tvarkaraščių sudarymo algoritmų programinę realizaciją.
- Atlikti realizuotų algoritmų tyrimus, apimančius optimalių ar rekomenduotinų algoritmų parametrų reikšmių ar jų intervalų nustatymą, kuriais būtų optimizuojamas algoritmų veikimo efektyvumas.
- Remiantis atliktais tyrimais, pateikti realizuotų algoritmų naudojimo rekomendacijas vartotojams.

1.2 TVARKARAŠČIŲ UŽDAVINIŲ SPRENDIMO METODIKA

Didžioji dalis tvarkaraščių sudarymo uždavinių priklauso NP – pilnajai uždavinių sudėtingumo grupei. Ne išimtis yra gamybinių tvarkaraščių sudarymo uždaviniai. Egzistuoja tik keletas gamybinių tvarkaraščių uždavinių atvejų, kai optimalų sprendinį galime gauti polinominio laiko algoritmu. Bendru atveju neribojant nei darbų, nei įrenginių skaičiaus, optimalų sprendinį galima gauti pilnu dvinariu perrinkimu arba pritaikius metodus, grindžiamus šakų ir ribų metodo idėjomis. Neribojant uždavinio apimties, minėtus tiksliuosius metodus dažniausiai neįmanoma

įgyvendinti realiu kompiuteriu per priimtiną skaičiavimų laiką, todėl tvarkaraščiams sudaryti taikomi apytiksliai euristiniai ar metaeuristiniai paieškos metodai. Minėtų apytikslių metodų pritaikymas negarantuoja optimalaus uždavinio sprendinio gavimo, bet apytiksliais metodais galima spręsti bet kokios apimties optimizavimo uždavinį ir per priimtiną skaičiavimų laiką (pageidaujamą arba laisvai pasirinktą) gauti aukštos kokybės sprendinius. Todėl šiame darbe nagrinėjamam vienam iš gamybinių tvarkaraščių sudarymo uždavinių spręsti sukurtoje programinėje įrangoje bus realizuoti du metaeuristiniams metodams priskiriami paieškos algoritmai.

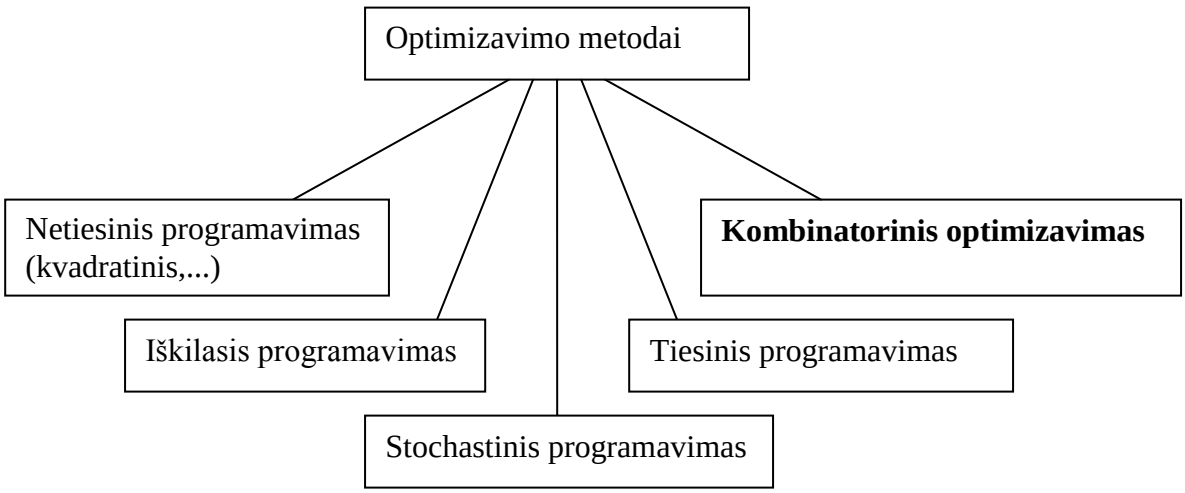
2. TEORINĖ DALIS

2.1 TVARKARAŠČIŲ SUDARYMO UŽDAVINIAI

Visų pirma bandysime surasti vietą, kurią matematikoje užima tvarkaraščių sudarymo uždaviniai ir jų sprendimo algoritmai. Tvarkaraščių sudarymo ar kitaip tariant optimizavimo uždaviniai ir jų sprendimo algoritmai priklauso optimizavimo metodų šeimai. Žodis „optimizuoti“ matematikoje reiškia funkcijos, kurią vadinsime tikslo funkcija, minimumo arba maksimumo paiešką tam tikroje srityje. Matematiškai optimizavimo uždaviniai formuluojami:

1. duota: funkcija f: A→ R, atvaizduojanti bet kurį aibės A elementą į realų skaičių.
2. ieškome: tokio $x_0 \in A$, kad $f(x_0) \leq f(x)$ bet kuriam $x \in A$ (minimizavimas) arba $f(x_0) \geq f(x)$ bet kuriam $x \in A$ (maksimizavimas).

Optimizavimo metodai yra plati matematikos šaka, jungianti įvairių sričių optimizavimo uždavinius. Pagal tai, kokio tipo uždaviniai nagrinėjami, optimizavimo metodus būtų galima suklasifikuoti:



2.1 pav. Optimizavimo metodų rūšių klasifikacija

Kombinatorinis optimizavimas nagrinėja tokias problemas, kurių visų galimų sprendimų aibė yra baigtinė ar bent jau skaiti. Nesunku suvokti, kad bet kurio tvarkaraščio (mokyklinio, transporto ar gamybinio...) visų galimų jo sudarymo variantų aibė yra baigtinė. Todėl tvarkaraščių optimizavimo uždavinys priskiriamas kombinatorinio optimizavimo problemoms. Bendra visų kombinatorinio optimizavimo uždavinių formulavimo schema galėtų būti nusakyta rinkiniu (*X*,*P*,*Y*,*f*,extr), kur:

- X – leistinoji sritis (kurioje f ir P yra apibrėžtos).

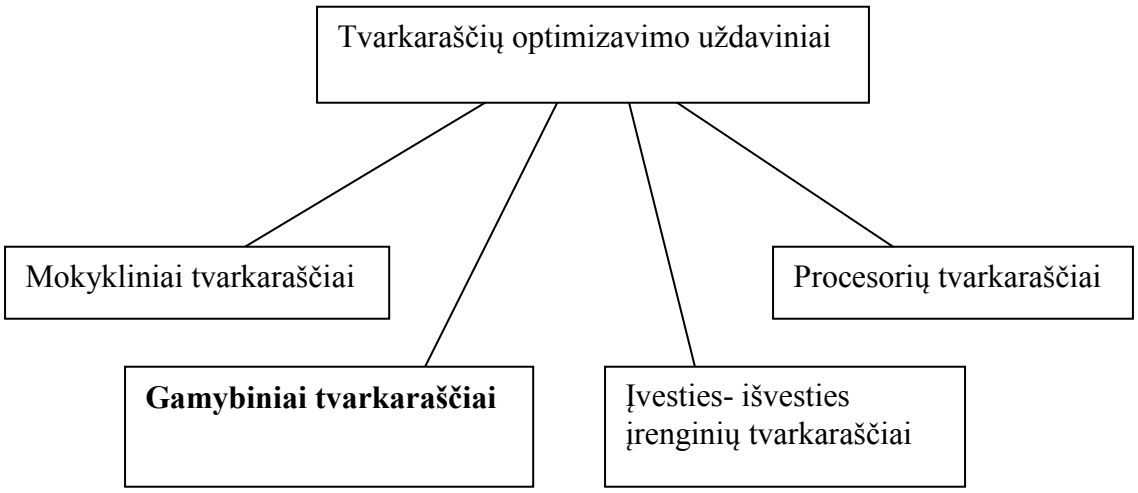
- P – įvykdomumo predikatas.
- Y – galimų sprendinių aibė.
- f – tikslo funkcija.
- extr – ekstremumas (minimumas arba maksimumas).

Tuomet bendriausia forma tvarkaraščių optimizavimo uždavinys galėtų būti apibrėžtas šitaip:

Duota:

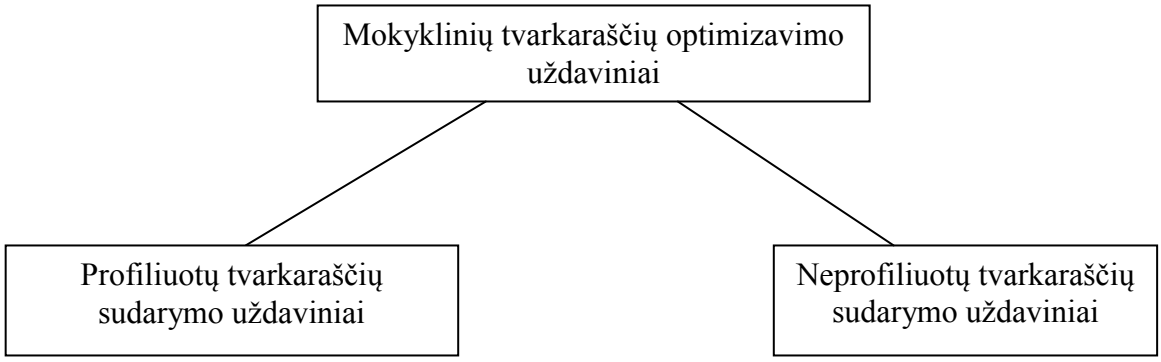
- aibė užduočių, kurios turi būti atliktos,
- aibė išteklių, kuriuos naudojant atliekamos užduotys,
- aibė apribojimų, kurie privalo būti tenkinami,
- tikslo funkcija, nusakanti tvarkaraščio charakteristikas.

Kaip geriausiai (tikslo funkcijos prasme) galima paskirti po vieną užduotį kiekvienam objektui tam tikrais laiko momentais, kad visi apribojimai būtų tenkinami ir visos užduotys atliktos? Pagal anksčiau minėtas aibes, kurios yra būtinos korektiškai tvarkaraščių optimizavimo uždavinio formuluotei, šio tipo uždavinius galima klasifikuoti į mokyklinius, gamybinius, procesorių, įvesties - išvesties įrenginių tvarkaraščių optimizavimo uždavinius:



2.2 pav. Tvarkaraščių optimizavimo uždavinių klasifikacija

Mokyklinių tvarkaraščių sudarymo uždavinius būtų galima suskirstyti į dvi grupes: profiliuotųjų ir neprofiliuotųjų tvarkaraščių sudarymo uždaviniai:



2.3 pav. Mokyklinių tvarkaraščių sudarymo uždavinių klasifikacija

Mokyklose, ypač profiliuotose, viena sudėtingiausių problemų – pamokų tvarkaraščio sudarymas. Profiliuotų mokyklų pamokų tvarkaraščio sudarymo problema tuo sudėtingesnė, kuo daugiau pasirinkimo laisvės duodama vienuoliktokams, tai yra kuo labiau yra tenkinami jų poreikiai. Tradicinių neprofiliuotų mokyklų tvarkaraščiai yra pastovūs kiekvienai klasei. Profiliuotoje klasėje gali būti mokinių, pasirinkusių skirtingus dalykus. Pavyzdžiui, kai kurie mokiniai gali rinktis tikybą, o kiti etiką. Taigi, klasės dalinamos į grupes pagal pasirinktus dalykus. Todėl tvarkaraštis vienai klasei tampa sudėtingesniu. Be to, klasės gali būti sujungiamos viena su kita dėl sutampančių dalykų arba padalinamos į grupes, kad dalykas atitiktų mokinio pasirinkimą.

Kadangi šiame darbe spręsime vieną iš gamybinių tvarkaraščių optimizavimo uždavinių, tai detaliau aptarsime šią tvarkaraščių sudarymo uždavinių grupę. Tokių uždavinių, kaip ir visų kitų rūšių tvarkaraščių sudarymo uždavinių, pradiniai duomenys yra reikiamų atlikti užduočių aibė. Kiekvienai užduočiai atlikti reikalingi apibrėžti ištekliai. Ištekliai skirstomi į atsinaujinančius (pavyzdžiui įrenginiai) ir neatsinaujinančius, kurie atlikus kažkokias operacijas yra sunaudojami. Neatsinaujinančių išteklių pavyzdys galėtų būti gamybinės žaliavos. Gamybinių tvarkaraščių sudarymo uždaviniuose dažniausiai sutinkamos prielaidos:

- Užduotis gali būti atliekamas keliais skirtingais būdais priklausomai nuo to, kokie ištekliai bus naudojami jį atliekant.
- Užduoties atlikimas gali reikalauti vieno ar daugiau skirtingų išteklių panaudojimo.
- Ištekliai gali turėti laikinus apribojimus.
- Užduoties vykdymas tvarkaraštyje gali būti nutrauktas ir atidėtas pagal tik tam darbui apibrėžtas nutraukimo taisykles. Taip pat užduoties nutraukimas gali būti draudžiamas.
- Išteklių pasiekiamumas gali būti keistis.

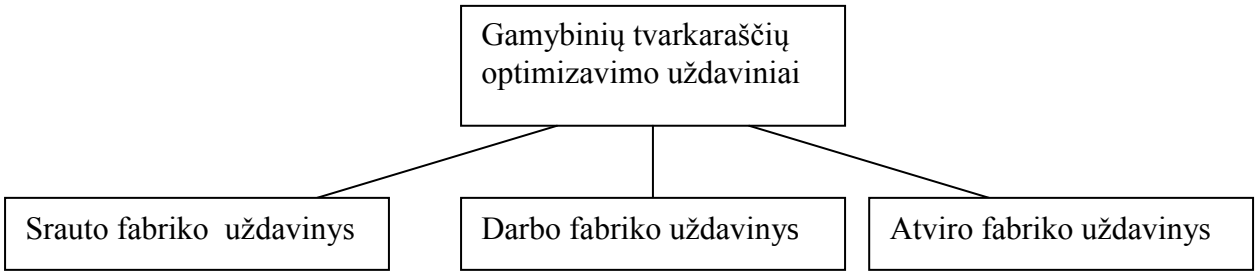
- Užduotys gali turėti pirmumą viena kitos atžvilgiu, t.y. yra užduočių, kurios negali būti pradedamos vykdyti, kol yra neužbaigtos kažkokios kitos, pirmumą prieš jas turinčios, užduotys.
- Galimas užduočių, kurios turi vienas kito atžvilgiu pirmumo sąryšius, vykdymo persidengimas, t.y. galima pradėti užduoties atlikimą, kai prieš tą užduotį pirmumą turinti užduotis yra dalinai užbaigta.

Tokios prielaidos paprastai vadinamos apribojimais. Apribojimai apibrėžiami konkretaus uždavinio formuluotėje. Tvarkaraščiu vadinsime bet kurį darbų priskyrimą ištekliams, kuris tenkina uždavinio apribojimus. Šio tipo uždavinių tikslas yra rasti tokį tvarkaraštį, kuris būtų optimalus pagal pasirinktus kriterijus. Tvarkaraščio optimalumui nusakyti sudaroma tikslo funkcija, galinti kiekvieną tvarkaraštį x atvaizduoti į realų skaičių: $f : x \to R$. Taigi, spręsdami gamybinių tvarkaraščių sudarymo uždavinius ieškosime sudarytos funkcijos f(x) globalaus ekstremumo (minimumo arba maksimumo). Pateiksime keletą dažniau pasitaikančių tikslo funkcijų pavyzdžių. Paprastai gamybinių tvarkaraščių optimizavimo uždavinių tikslo funkcija dažniausiai priklauso nuo i- ojo darbo atlikimo termino d_i ir realaus to darbo atlikimo laiko c_i. Pažymime:

- vėlavimas $L_i = c_i - d_i$; (2.1)
- ankstinimas $E_i = \max\{0, d_i - c_i\}$; (2.2)
- pavėlavimas $T_i = \max\{0, c_i - d_i\}$; (2.3)
- baudos taškas $u_i = \begin{cases} 0, jei & c_i < d_i \\ 1, jei & c_i > d_i \end{cases}$ (2.4)

Tikslo funkcijos išraiška galėtų būti c_{max} , L_{max}, T_{max}, $\sum_i c_i$, $\sum_i L_i$, $\sum_i T_i$, $\sum_i u_i$ ir priklausanti nuo to, pagal kokį požymį (bendrą visų darbų atlikimo laiką, didžiausią vėlavimą, bendrą visų darbų vėlavimų sumą, vėluojamų atlikti darbų skaičių...) norime optimizuoti tvarkaraštį. Nesunku pastebėti, kad sprendžiant optimizavimo uždavinius visais atvejais ieškoma pateiktų tikslo funkcijų minimumo. Realiuose gamybinių tvarkaraščių sudarymo uždaviniuose tvarkaraštį dažnai prireikia optimizuoti nebūtinai pagal vieną požymį. Tokiu atveju yra neišvengiami dviejų ar daugiau optimizavimo požymių konfliktai: gerinant tvarkaraščio charakteristikas pagal kažkokį vieną požymį, likusios charakteristikos neišvengiamai blogėja. Todėl optimizuojant tvarkaraščius pagal keletą požymių apibrėžiami ryšiai tarp konfliktuojančių požymių, kurie leidžia nuspręsti, kuris požymis svarbesnis norint gauti optimalų tvarkaraštį.

Įvairioje literatūroje dažniausiai nagrinėjami šie trys gamybinių tvarkaraščių optimizavimo uždavinių tipai: srauto fabrikas, darbo fabrikas, atviras fabrikas:



2.4 pav. Gamybinių tvarkaraščių optimizavimo uždavinių klasifikacija

Suformuluosime klasikinius šių uždavinių variantus. Visų trijų tipų uždavinius nusako du parametrai: tai darbų skaičius, kurį pažymėsime raide N, ir užduočių (įrenginių) skaičius, kurį pažymėsime raide M. Tarsime, kad įrenginių skaičius lygus užduočių skaičiui ir kiekvienas įrenginys gali atlikinėti tik vienos rūšies užduotį. Be to, vienu metu įrenginys gali atlikti vieną užduotį ir vienas darbas vienu metu gali būti viename įrenginyje. Bet kuris iš skaičiaus N darbų bus galutinai atliktas tada, kai su tuo darbu bus atliktos visos M užduotys. Todėl norint užbaigti visus darbus, reikės atlikti NM operacijų. Šios prielaidos būdingos visų trijų tipų (ir srauto, ir darbo, ir atviro fabriko) uždaviniams, o šie uždaviniai skiriasi tik darbų užduočių atlikimo tvarkos apribojimais.

Srauto fabriko uždavinyje su kiekvienu iš N darbų užduotys atliekamos tam tikra visiems darbams vienoda tvarka. Šio uždavinio tikslas yra rasti tokią darbų seką, sudarytą iš N elementų, kad pasirinktos tikslo funkcijos reikšmė būtų mažiausia. Šiuo atveju visų galimų tvarkaraščio sudarymo variantų skaičius yra lygus N!. Šį uždavinį galėtų gerai iliustruoti siuvykla, kurioje bet kuriam drabužiui pasiūti, tarsime, reikalinga vienoda užduočių seka: dažymas, kirpimas, siuvimas, lyginimas. Darbu laikysime bet kurį konkretų drabužį.

Darbo fabriko uždavinyje su kiekvienu darbu užduotys atliekamos tam tikra kiekvienam darbui būdinga tvarka, kuri kiekvienam darbui gali būti bet kokia. Šio uždavinio tikslas yra kiekvienam įrenginiui priskirti darbų užduotis taip, kad pasirinktos tikslo funkcijos reikšmė būtų mažiausia.

Atviro fabriko uždavinys nuo anksčiau nagrinėtų uždavinių skiriasi tuo, kad kiekvienam darbui nėra jokių užduočių atlikimo tvarkos apribojimų. Šio uždavinio tikslas yra kiekvienam įrenginiui priskirti darbų užduotis taip, kad pasirinktos tikslo funkcijos reikšmė būtų mažiausia.

Suformuluosime pateiktus uždavinius matematiškai. Tarkime, kad $O = \{1,2,...,NM\}$ yra visų reikiamų atlikti operacijų aibė. Tada pažymime:

- p_i - i-osios operacijos $i \in O$ vykdymo laikas.
- S_i - i-osios operacijos vykdymo pradžios laikas, $\forall S_i \in N_0$, čia: N_0 - natūraliųjų skaičių aibė su skaičiumi 0.
- C- aibė tokių porų (i, j), kur i-oji operacija turi pirmenybę prieš j-ąją, t.y. j-oji operacija negali būti pradedama, kol nepabaigta i-oji. Taigi, jei i-oji operaciją turi pirmenybę prieš j-ąją, tai $(i, j) \in C$.
- $D = \{(i, j) \mid m_i = m_j\}$, kur m_i - i-ajai operacijai atlikti reikalingas įrenginys. Taigi, aibė D apibrėžia tokią operacijų porų (i, j) aibę, kurioms atlikti reikia to paties įrenginio.

Pasirenkame konkrečią tikslo funkciją. Tarkime, kad tikslo funkcija lygi visų darbų atlikimo trukmei. Laikantis ankstesnių pažymėjimų, šiuo atveju tikslo funkcijos matematinė išraiška bus $\max_{i \in O}\{S_i + p_i\}$.

Todėl tiek srauto, tiek darbo fabriko klasikiniai uždaviniai formuluojami:

minimizuoti $\{ \max_{i \in O}\{S_i + p_i\} \}$

su sąlygomis:

$$S_i \in N_0,\ i \in O,$$

$$S_i + p_i \le S_j,\ (i, j) \in C, \quad (2.5)$$

$$S_i + p_i \le S_j \vee S_j + p_j \le S_i,\ (i, j) \in D. \quad (2.6)$$

Formaliai tvarkaraščiu vadinsime vektorių $S = (S_1, S_2, ..., S_{NM}) \in N_0 \times ... \times N_0$, kurio neneigiamos komponentės apibrėžia kiekvienos operacijos pradžios laiką ir tenkina 2.5 ir 2.6 sąlygas. Sąlyga 2.5 reikštų, kad visoms poroms $(i, j) \in C$ j-osios operacijos atlikimas negali prasidėti anksčiau nei i-osios operacijos atlikimo pabaiga. Sąlyga 2.6 reikštų, kad visoms poroms $(i, j) \in D$ (operacijoms i ir j atlikti reikia to paties įrenginio) j-osios operacijos atlikimas negali prasidėti anksčiau nei i-osios operacijos atlikimo pabaigos laikas arba i-osios operacijos atlikimas negali prasidėti anksčiau nei j-osios operacijos atlikimo pabaigos laikas, t.y. įrenginys negali vienu metu atlikinėti dviejų užduočių.

Kadangi atviro fabriko uždavinyje nėra užduočių atlikimo tvarkos apribojimų, tai pastebime, kad šiuo atveju aibė $C = \varnothing$. Tačiau tiek srauto, tiek darbo, tiek atviro fabriko uždavinių

klasikiniuose variantuose taip pat reikalaujama, kad su vienu darbu negali būti vienu metu atliekamos kelios skirtingos operacijos. Tada dar pažymime aibę E:

- $E = \{(i, j) \mid d_i = d_j\}$, čia: d_i nusako darbą, kuriam priklauso i-oji operacija. Taigi, aibė E apibrėžia tokią operacijų porų (i, j) aibę, kurios priklauso tam pačiam darbui.

Todėl atviro fabriko uždavinio matematinė formuluotė:

minimizuoti $\{ \max_{i \in O} \{S_i + p_i\} \}$

su sąlygomis:

$$S_i \in N_0,\ i \in O,$$

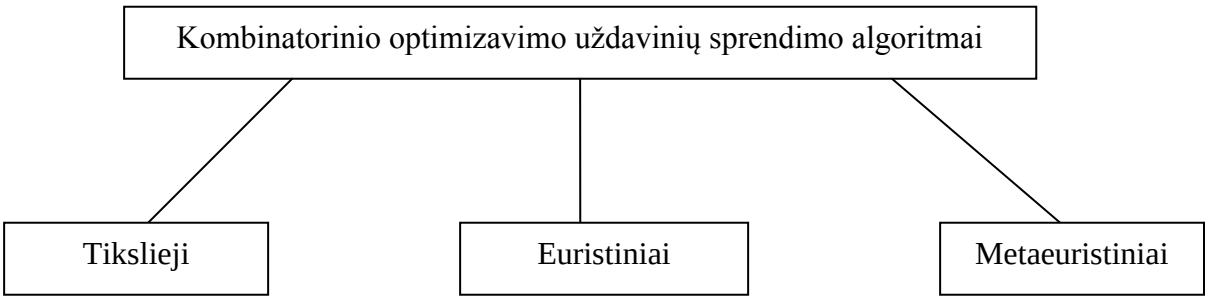
$$S_i + p_i \le S_j \vee S_j + p_j \le S_i,\ (i, j) \in E,$$

$$S_i + p_i \le S_j \vee S_j + p_j \le S_i,\ (i, j) \in D.$$

Aptarėme klasikinius srauto, darbo ir atvirojo fabriko uždavinių variantus ir jų matematines formuluotes. Visuose realiuose gamybinių tvarkaraščių sudarymo uždaviniuose dažnai sutinkama daugiau įvairių prielaidų ar galimybių, kurios paminėtos anksčiau. Aptartų bazinių uždavinių variantų konkretūs duomenys paprastai naudojami tyrinėjant konkrečių algoritmų gebėjimą spręsti atitinkamus gamybinių tvarkaraščių sudarymo uždavinius bei norint palyginti keleto algoritmų rezultatus sprendžiant konkrečius tvarkaraščių sudarymo uždavinius. Kita vertus, šių bazinių klasikinių uždavinių sudėtingumas dažniausiai būna ekvivalentus realių gamybinių tvarkaraščių sudėtingumui: abiejų šių tipų uždaviniai bendru atveju priklauso NP – pilnajai uždavinių sudėtingumo klasei.

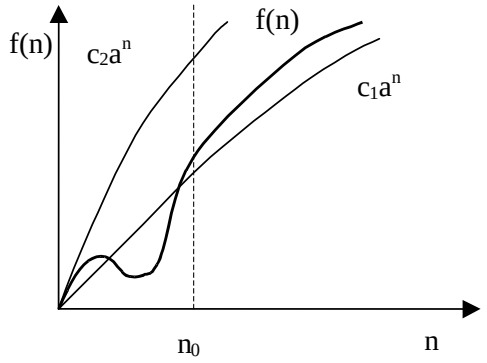
2.2 TVARKARAŠČIŲ SUDARYMO ALGORITMAI

Visus tvarkaraščių optimizavimo ar apskritai kombinatorinio optimizavimo uždavinių sprendimo metodus – algoritmus galima būtų suskirstyti į tris pagrindines grupes: tikslieji, euristiniai ir metaeuristiniai algoritmai.



2.5 pav. Kombinatorinio optimizavimo uždavinių sprendimo algoritmų rūšys

Tikslieji algoritmai garantuoja uždavinio optimalaus sprendinio radimą. Tačiau bendru atveju beveik visų kombinatorinio optimizavimo problemų, tarp jų žinoma ir tvarkaraščių optimizavimo uždavinių, tiksliųjų sprendimo algoritmų vykdymo laikas auga eksponentiškai, didėjant uždavinio apimčiai. Pavyzdžiui, gamybinių tvarkaraščių optimizavimo uždaviniuose uždavinio apimtimi n galime laikyti darbų skaičiaus ir užduočių skaičiaus sandaugą $n = NM$. Laikysime, kad algoritmo vykdymo laiko funkcija f(n), priklausanti tuo uždavinio apimties, auga eksponentiškai, didėjant uždavinio apimčiai, jeigu egzistuoja tokios konstantos c_1,c_2, n_0, kad visiems pakankamai dideliems n galiotų nelygybės: $c_1 a^n < f(n) < c_2 a^n, a > 1, n > n_0$. Funkcijų f(n) ir a^n santykį galėtume pavaizduoti:



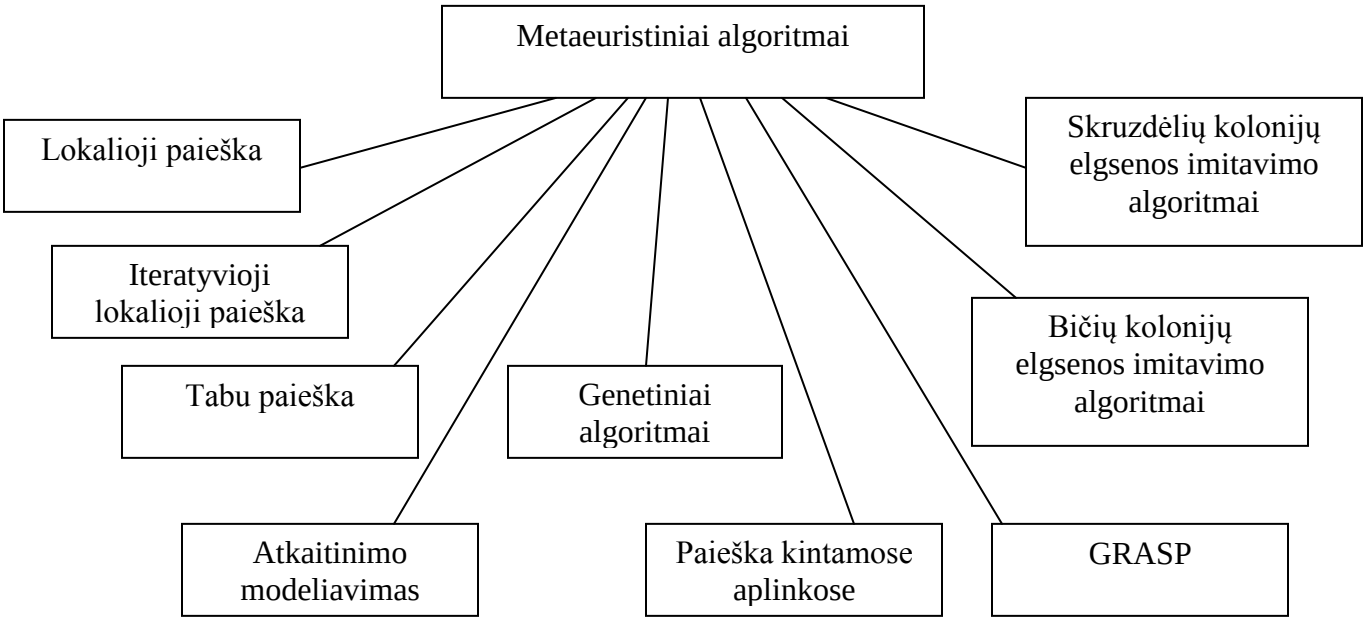
2.6 pav. Eksponentiškai auganti funkcija

Laikysime, kad algoritmo vykdymo laiko funkcija f(n) auga polinomiškai, didėjant uždavinio apimčiai, jeigu egzistuoja tokios konstantos c_1,c_2, ir pastovus realus skaičius k, kad visiems pakankamai dideliems n galiotų nelygybės: $c_1 n^k < f(n) < c_2 n^k, n > n_0$. Kai n→∞, bet kuri eksponentinė funkcija didėja žymiai greičiau už bet kurią polinominę funkciją. Jei neegzistuoja tikslus polinominio laiko tam tikro uždavinio sprendimo algoritmas, tai tas uždavinys priskiriamas NP – pilnajai uždavinių sudėtingumo klasei. Bendru atveju visos kombinatoriam optimizavimui priskiriamos problemos priklauso NP – pilnajai klasei. Laikoma, kad uždavinys yra išsprendžiamas kompiuteriu, jeigu jo algoritmo vykdymo laiko funkcija polinominė, ir laikoma, kad uždavinys yra kompiuteriu neišsprendžiamas, jei jo algoritmo vykdymo laiko funkcija auga eksponentiškai. Todėl per priimtiną skaičiavimų laiką tiksliai išspręsti pavyksta tik nedidelės apimties kombinatorinio optimizavimo uždavinius. Tai būtų bene vienintelis, tačiau reikšmingas tiksliųjų algoritmų trūkumas. Tarkime, kad darbo fabriko tvarkaraščių sudarymo uždavinio visų galimų sprendinių skaičius yra $N!^M$. Fiksuokime $M = 3$ ir N įgyja reikšmes 3,4,5,6, ir šių uždavinių operacijų skaičius atitinkamai yra 9,12,15,18. Tuomet šiš uždavinių sprendinių pilnojo perrinkimo laikas didėja eksponentiškai augant uždavinio operacijų skaičiui šiuolaikiniam kompiuteriui taip:

2.7 pav. Eksponentiškai augantis pilnojo perrinkimo skaičiavimų laikas

Euristiniu laikomas toks optimizavimo uždavinio sprendimo metodas, kuriuo siekiama rasti aukštos kokybės, bet nebūtinai optimalų sprendinį per priimtiną skaičiavimų laiką. Tai yra pagrindinis euristinių algoritmų skirtumas nuo anksčiau minėtų tikslių metodų. Šios rūšies algoritmai remiasi tam tikrų taisyklių, kurios vadinamos euristikomis, pritaikymu sprendžiant konkretų kombinatorinio optimizavimo uždavinį. Tokios taisyklės šiuo atveju nurodo sprendimą, kurį reikia priimti konkrečiose situacijose, o šio sprendimo pasirinkimas gali padėti gauti geresnį sprendinį. Euristiniai algoritmai yra pritaikomi tik tam tikriems konkretiems uždaviniams su tam tikrais apribojimais ar prielaidomis.

Metaeuristinius metodus apibrėšime kaip tam tikro aukšto abstrakcijos lygio nurodymų rinkinius. Priešingai nei euristinių algoritmų, tokių nurodymų paskirtis yra formaliai aprašyti kurios nors klasės uždavinių sprendimo idėją, principą. Taigi, sąvoka „metaeuristinis algoritmas“ yra talpesnė nei „euristinis metodas“. Kombinatorinio optimizavimo uždaviniams spręsti naudojami šie metaeuristiniai algoritmai: lokalioji paieška, iteratyvioji lokalioji paieška, tabu paieška, genetiniai algoritmai, godžiosios randomizuotos adaptyvios paieškos procedūros (GRASP), atkaitinimo modeliavimas, skruzdėlių kolonijų elgsenos imitavimo algoritmai, paieška kintamose aplinkose, bičių kolonijų elgsenos imitavimo algoritmai ir kt.



2.8 pav. Metaeuristniai algoritmai

Lokaliosios paieškos algoritmai remiasi teiginiu, kad turimas pradinis uždavinio sprendinys gali būti pagerintas atlikus mažus to sprendinio pakeitimus. Šiuose algoritmuose atliekami tik tokie nedideli pradinio sprendinio pakeitimai, kuriuos atlikus sumažėja pasirinktos tikslo funkcijos reikšmė. Lokaliosios paieškos algoritmo gautas sprendinys bus tik lokalusis optimumas tikslo funkcijos prasme toje aplinkoje, į kurią pataikė pradinis sprendinys. Šiuo atveju bet kokio pradinio sprendinio aplinka galėtume pavadinti tokią kitų sprendinių aibę, kurie gaunami atlikus visus galimus nedidelius pradinio sprendinio pakeitimus. Lokaliosios paieškos algoritmų gaunamų sprendinių kokybė labai priklauso nuo pradinio sprendinio.

Atkaitinimo modeliavimo algoritmai remiasi fizikinio proceso – atkaitinimo (grūdinimo) idėja. Vienas iš pagrindinių parametrų – temperatūra T, kuri nuolat mažėja su kiekvienu šio proceso imitavimo žingsniu. Įvedama speciali temperatūros atnaujinimo (aušinimo) funkcija, taip pat specialus nuo temperatūros priklausomas tikimybinis patvirtinimo kriterijus. Jo esmė – kiekviename šio metodo žingsnyje patvirtinti rastus geresnius nei esamas sprendinius ir su nedidele tikimybe patvirtinti blogesnius sprendinius. Tai suteikia galimybę ištrūkti iš lokalių optimumo taškų ieškant globalaus optimumo.

Tabu paieškos metodas remiasi tam tikrų sprendinių uždraudimo idėja. Pagrindinis jo parametras – tabu sąrašas, kuris iš pradžių yra tuščias. Į jį proceso eigoje vis įtraukiami tie sprendiniai (ar jų aplinkos), kuriuos mes jau aplankėme. Tokiu būdu šis metodas kuriam laikui uždraudžia grįžimą prie jau buvusių sprendinių (jų aplinkų), kol iš fiksuoto ilgio ir nuolat kintančio

tabu sąrašo bus išstumtas draudimas. Taigi, tabu paieškos algoritmuose priešingai nei likusiuose saugoma dalis paieškos proceso praeities, kas teoriškai turėtų suteikti ir suteikia pranašumą prieš kitus algoritmus. Tabu paieškos algoritmų įvairios modifikacijos yra vieni iš efektyviausių metodų, naudojamų spręsti kombinatorinio optimizavimo uždavinius.

Esminis genetinių algoritmų skirtumas nuo anksčiau aptartų algoritmų yra tas, kad čia operuojama ne su vienu sprendiniu, o su sprendinių populiacija – chromosomų rinkiniu. Prie tokių metodų priskiriami genetiniai algoritmai. Šie algoritmai remiasi genetikos mokslo idėjomis. Nauji sprendiniai iš turimų gaunami pritaikius genetines operacijas - kryžminimą, mutaciją, atranką. Taip gaunamos naujos sprendinių populiacijos su „geresnėmis“ savybėmis, kurios turi įtakos apibrėžtos tikslo funkcijos gerėjimui.

Godžioji randomizuota adaptyvi paieškos procedūra (GRASP) yra konstruktyvios euristikos ir lokaliosios paieškos derinys. Kiekviena GRASP iteracija susideda iš dviejų dalių: sprendinio generavimo ir sprendinio gerinimo naudojant lokaliąją paiešką. Gamybinių tvarkaraščių sudarymo GRASP algoritmuose sprendinys paprastai sugeneruojamas dedant į tvarkaraštį nuosekliai vieną po kitos operacijas. Dedama operacija atsitiktinai pagal įvairias strategijas atrenkama iš tos grupės kandidatų, kuriuos įdėjus dalinio tvarkaraščio tikslo funkcijos reikšmė būna mažiausia (jei spręsdami uždavinį ieškome minimumo). Sugeneruotas sprendinys nebūtinai yra lokalusis minimumas, todėl turimam sprendiniui pagerinti dar pritaikoma lokalioji paieška.

Paieškos kintamose aplinkose algoritmuose vieną iteraciją paprastai sudaro dvi fazės: suvirpinimo procedūra ir lokalioji paieška. Suvirpinimo procedūroje atliekama atsitiktinių turimo sprendinio perstatymų, kurie gali ir pagerinti, ir pabloginti sprendinį. Tuo siekiama, kad algoritmu gaunami sprendiniai neužsiciklintų kažkurioje lokalaus minimumo aplinkoje. Po suvirpinimo procedūros seka lokaliosios paieškos fazė.

Tvarkaraščių ar apskritai įvairių optimizavimo uždavinių sprendimo algoritmuose galima pritaikyti praktiškai visų minėtų metaeuristinių nurodymų idėjas. Tai būtų pagrindinis metaeuristinių metodų pranašumas prieš kokiam nors vienam konkrečiam uždaviniui spręsti tinkamus euristinius algoritmus. Norint rasti geresnės kokybės sprendinius, dėl metaeuristinių metodų universalumo, dažnai galime apjungti kelių skirtingų rūšių metaeuristinių algoritmų idėjas. Todėl šiuo metu metaeuristiniai algoritmai ir įvairios jų ir kelių metaeuristinių metodų idėjų kombinacijos, pavyzdžiui genetiniai algoritmai ir lokalioji paieška, tabu paieška ir atkaitinimo modeliavimas ir kt., yra pagrindiniai tyrinėjimų objektai, kuriais siekiama sukurti algoritmus, kuo efektyviau sprendžiančius kombinatorinio optimizavimo, tarp jų ir tvarkaraščių sudarymo, uždavinius, t.y. per tam tikrą skaičiavimų laiką gauti optimalius ar jiems artimus sprendinius.

2.3 DARBE SPRENDŽIAMI UŽDAVINIAI IR JŲ ALGORIMAI

Šiame darbe nagrinėsime algoritmus bei kursime programines priemones darbo fabriko uždaviniui spręsti, kurį, kaip jau minėjome, nusako reikiamų atlikti darbų skaičius N ir kiekvienam darbui būtinų atlikti užduočių skaičius M. Be to, kiekvienas darbas gali turėti specifinę užduočių atlikimo tvarkos seką. Tvarkaraščio optimizavimo kriterijumi pasirenkame darbų trukmę, tai reikštų, kad bandysime sudaryti tokį tvarkaraštį, kurio visų darbų atlikimo trukmė būtų trumpiausia. Bene akivaizdžiausias tikslus šio uždavinio sprendimas būtų visų galimų tvarkaraščių sudarymo variantų perrinkimas ir liktų išrinkti tą tvarkaraštį, kuris yra „geriausias“ pasirinktos tikslo funkcijos atžvilgiu. Tačiau bet kurio tokio pilno perrinkimo algoritmo vykdymo laiko funkcija yra eksponentinė. Šis uždavinys bendru atveju priklauso NP – pilnajai uždavinių sudėtingumo klasei ir nėra tiksliai išsprendžiamas per polinominį sprendimo laiką. Yra tik keli atskirieji šio uždavinio atvejai, kai jį pavyksta tiksliai išspręsti per polinominį laiką:

- darbų skaičius yra lygus dviem,
- darbui užbaigti reikalingų atlikti užduočių skaičius yra lygus dviem, ir kiekvienos užduoties trukmė lygi vienetui.

Praktikoje, žinoma, kyla gerokai sudėtingesnių uždavinių nei aprašyti atskirieji atvejai, todėl šiame darbe darbų skaičiaus N ir užduočių skaičiaus M neribosime. Be to, nagrinėsime klasikinį darbo fabriko uždavinį, nes tai bene sunkiausiai sprendžiama gamybinių tvarkaraščių sudarymo problema. O klasikinį uždavinio variantą pasirenkame todėl, kad būtent pagal šio varianto etaloninius uždavinius daugelis literatūros šaltinių autorių testuoja bei lygina savo algoritmais gaunamus rezultatus, taip pat yra žinomos minėtų etaloninių darbo fabriko problemų optimalios darbų trukmės ar jų įverčiai. Taigi, todėl galėsime įvertinti gautų sprendinių darbų trukmių skirtumą nuo optimalios darbų trukmės ar jos įverčio.

Kadangi konstantos N ir M galės būti bet kokio dydžio, tai šiam uždaviniui spręsti tiksliųjų pilnojo perrinkimo algoritmų atsisakome dėl jų vykdymo laiko eksponentinės funkcijos, žinodami, kad tokie algoritmai per priimtiną sprendimo laiką tiksliai išspręstų tik labai nedidelės apimties uždavinius. Kaip minima [10] šaltinyje, patys efektyviausi šakų ir ribų metodai dar per priimtiną laiką išsprendžia 225 operacijų uždavinius, ir tai yra riba šio tipo metodams. Užuot perrinkinėję visus galimus tvarkaraščio sudarymo variantus, naudosime apytikslius paieškos metodus, kurie nebūtinai, o dažniausiai ir nesuras optimalaus uždavinio sprendinio. Tačiau su apytiksliais paieškos algoritmais galėsime spręsti iš esmės bet kokios apimties uždavinį ir gauti aukštos kokybės sprendinius per priimtiną skaičiavimų laiką. Pagrindinis reikalavimas darbe naudosimiems algoritmams ir kuriamai tvarkaraščių optimizavimo programinei įrangai būtų rasti bet kokios

apimties tvarkaraščių sudarymo uždavinio kiek galima geresnį sprendinį per tam tikrą ir ribotą skaičiavimų laiką. Todėl pasirenkame keletą algoritmų, kurie priklauso metaeuristinių metodų rūšiai: paieška kintamose aplinkose, tabu – atkaitinimo modeliavimo paieškos algoritmas. Paminėsime, kad antrame pasirinktame algoritme bus bandomos sujungti atskirų metaeuristinių metodų – tabu paieškos ir atkaitinimo modeliavimo – idėjos. Plačiau apie realizuotus algoritmus kituose skyreliuose.

2.4 DARBO FABRIKO UŽDAVINYS IR JO DUOMENŲ STRUKTŪROS

Visų pirma nusakysime tikslią pradinių duomenų struktūrą darbo fabriko tvarkaraščio optimizavimo uždaviniui, kuri tiks visiems naudosimiems algoritmams. Kaip minėjome, darbo fabriko uždavinį nusako darbų skaičius N ir užduočių skaičius M, todėl šiame darbe uždavinio apimčiai nusakyti naudosime tokią formą: N×M, kurioje pirmas skaičius reikš darbų skaičių, o antras skaičius – užduočių skaičių. Pavyzdžiui, darbo fabriko uždavinio 3×3 pradinių duomenų lentelė, visiškai nusakanti šio uždavinio apribojimus tvarkaraščiui, galėtų atrodyti šitaip:

2.1 Lentelė

Darbų ir jų užduočių tvarkos ir trukmės lentelė

Darbas	(užduotis(įrenginys), laikas)	(užduotis(įrenginys), laikas)	(užduotis(įrenginys), laikas)
d1	3,1	1,3	2,6
d2	2,8	3,5	1,10
d3	2,3	1,8	3,1

Iš uždavinio formato matome, kad turime tris darbus ir norint pabaigti kiekvieną darbą, reikia su kiekvienu iš darbų atlikti po tris užduotis. 2.1 lentelės pirmąją eilutę mes interpretuosime taip: pirmam darbui „d1“ pirmiausiai turės būti atlikta trečioji užduotis (ar kitaip sakant darbas „d1“ pirmiausiai turi aplankyti trečiąjį įrenginį), kuri truks vieną laiko vienetą, po to darbui „d1“ turi būti atlikta pirmoji užduotis, kuri truks tris laiko vienetus ir galiausiai darbui „d1“ turės būti atlikta antroji užduotis, truksianti šešis laiko vienetus.

Taip pat nusakysime optimizavimo algoritmais gaunamų sprendinių duomenų struktūrą. Kaip jau minėjome teorinėje dalyje, formaliai tvarkaraščiu (uždavinio sprendiniu) laikysime vektorių $S = (S_1, S_2, ..., S_{NM})$, kurio kiekviena komponentė nusako atitinkamos operacijos pradžios laiką ir tenkina uždavinio apribojimus. Siekdami lengvesnio gauto sprendinio interpretavimo ir bendresnės jo formos, galutinio sprendinio kiekvieną iš NM operacijų galutinai nusakysime taip:

$$(darbas_i, irenginys_j, pradžia, pabaiga), i = \overline{1, N},\ j = \overline{1, M}.$$

Ši eilutė reiškia, kad j- asis įrenginys apibrėžtam laikui priskirtas i – ajam darbui. Ir tokie gauti sprendiniai paprastai vaizduojami Ganto diagrama.

Kaip ir visais atvejais sprendžiant panašius uždavinius, algoritmų realizacijose turimi tvarkaraščiai bus nusakomi sveikųjų skaičių sekomis. Pavyzdžiui, nagrinėto 3×3 formos pavyzdžio darbams „d1“, „d2“, „d3“ priskiriame atitinkamai sveikuosius skaičius 1,2,3. Duomenų lentelę nuskaitę stulpeliais ir priskyrę darbams atitinkamus numerius, gautume tokią seką:

(1,3,1) (2,2,8) (3,2,3) (1,1,3) (2,3,5) (3,1,8) (1,2,6) (2,1,10) (3,3,1);

Kiekvienas sekos narys apibūdina kažkurią iš 9 operacijų ir nario vieta sekoje parodo jį atitinkančios operacijos įdėjimo į tvarkaraštį prioritetą. Pavyzdžiui, sudarant tvarkaraštį iš pateiktos sekos, pradedant iš kairės įdėjus į tvarkaraštį pirmąjį narį (1,3,1), trečiajam įrenginiui bus priskirtas pirmasis darbas ir šio priskyrimo pradžios laikas yra lygus 0, o pabaigos laikas lygus vienetui. Visos kitos operacijos dedamos į tvarkaraštį jau atsižvelgiant į anksčiau įdėtas operacijas, ir jų pradžios ir pabaigos laikai parenkami tokie, kad nepažeistų uždavinio apribojimų. Tačiau darbo fabriko uždavinio atveju, ne visos sekų kombinacijos (jų šiuo atveju yra 9! variantų) nusako darbo užduočių atlikimo apribojimus atitinkančius tvarkaraščius. Stulpeliais nuskaityta pavyzdžio seka visuomet tenkins uždavinio apribojimus, todėl kiekvieno algoritmo realizacijoje įvairūs sprendinio pakeitimai bus pradedami atlikti nuo šios iš duomenų stulpeliais nuskaitytos pradinės sekos. Sukeitus pradinės sekos elementus taip, kad sprendinys taps netenkinančiu apribojimų, sekos darbų (tų darbų, kurių užduočių tvarka netenkina apribojimų) tvarka bus pakeičiama taip, kad tenkintų apribojimus.

2.5 PAIEŠKA KINTAMOSE APLINKOSE

Paieška kintamose aplinkose – vienas iš geriau žinomų lokaliosios paieškos metodų, skirtų įvairioms kombinatorinio optimizavimo problemoms spręsti. Kaip ir daugumoje metaeuristinių metodų, paieškos kintamose aplinkose algoritme tyrinėjama turimo sprendinio ribota aplinka ir tuo metu tikintis, kad joje bus rasti geresnę (mažesnę ar didesnę) tikslo funkcijos reikšmę turintys sprendiniai. Šiame algoritme kiekvieną iteraciją paprastai sudaro dvi fazės: sužadinimo procedūra ir lokalioji paieška. Sužadinimo procedūroje atliekama atsitiktinių turimo sprendinio perstatymų, kurie gali ir pagerinti, ir pabloginti sprendinį. Tuo siekiama, kad algoritmu gaunami sprendiniai neužsiciklintų kažkurioje lokalaus minimumo apibrėžtoje aplinkoje, nes dėl tvarkaraščių sudarymo uždavinių ypatumų, dažniausiai ištyrę turimo sprendinio apibrėžtą ribotą aplinką, geresnio sprendinio tikslo funkcijos prasme šioje aplinkoje galime nerasti, nors tokie sprendiniai egzistuoja. Taigi, norint to ištrūkti iš izoliuotų lokaliųjų minimumų aplinkų, algoritmuose būtinos panašios procedūros. Po sužadinimo procedūros seka lokaliosios paieškos fazė, kurioje tyrinėjama jau sužadinto sprendinio aplinka.

Paieškos kintamose aplinkose algoritmo schema:

1. **Pradžia.** Atsitiktinai generuojame pradinį sprendinį x. Apskaičiuojama x tikslo funkcijos reikšmė (darbų trukmė).

Pagalbiniai kintamieji $x' = x$, $x_{\min} = x$, $j = 0$. Įvedame parametrus S, K.

2. **Kartoti:**

   a) Lokalioji paieška. Pritaikoma turimam sprendiniui x' lokaliosios paieškos procedūra ir grąžinamas iš tvarkaraščio x' gautas sprendinys x.

   b) Pagerinta ar ne. Jei $f(x) < f(x_{\min})$, tai $x_{\min} = x$.

   čia: $f(x)$ - sprendinio x tikslo funkcijos reikšmė (darbų trukmė).

   c) Sužadinimo procedūra. Jei $j \bmod K = 0$, tai iš sprendinio x generuojamas sužadintas sprendinys $x' \in A^S(x)$, čia $A^S(x)$ sprendinio x sužadinimo funkcija.

   d) $j = j + 1$. Grįžtama į antrojo žingsnio pradžią.

3. **Pabaiga.** Grąžiname mažiausią tikslo funkcijos reikšmę turintį sprendinį $x_{\min}$.

Toliau aptarsime pateikto paieškos kintamose aplinkose algoritmo lokaliosios paieškos ir sužadinimo procedūras išsamiau. Šiose dviejose pagrindinėse algoritmo fazėje atlikinėjami tam tikri turimo sprendinio pakeitimai apibrėžtoje aplinkoje.

Pirmiausiai nusakysime pasirinktą aplinką. Sprendinio x aplinka vadinsime kitų sprendinių aibę $A(x)$ su nusakyta taisykle, kuria naudojantis galima gauti visus aibės $A(x)$ elementus iš turimo sprendinio x. Tokios taisyklės paprastai apibrėžiamos kaip maži turimo sprendinio perstatymai, ir šiuo atveju kiekvieną sprendinio x aplinkos elementą $x' \in A_S(x)$ galime gauti sprendinyje x atlikę kažkurį apibrėžtą perstatymą. Sprendžiant tvarkaraščių sudarymo uždavinius, tokias turimo sprendinio perstatymo taisykles patogu nusakyti galimų sukeisti (kai perstatymo taisyklė yra elementų sukeitimas) elementų porų aibe. Pavyzdžiui, 2.1 uždavinio sprendinys gali būti nusakytas seka:

(1,3,1) (2,2,8) (3,2,3) (1,1,3) (2,3,5) (3,1,8) (1,2,6) (2,1,10) (3,3,1);

Bene paprasčiausia tokio sprendinio aplinka gali būti nusakyta aibe, kurios bet kurį elementą galime gauti sukeitus pradinio duoto sprendinio bet kuriuos 2 elementus (vektorius) vietomis. Nuo aplinkos pasirinkimo dažnai labai stipriai priklauso algoritmų efektyvumas sprendžiant atitinkamus uždavinius. Pasirinkus tyrinėti didesnę sprendinio aplinką, tam bus naudojama daugiau kompiuterio skaičiavimo laiko, ir gali būti lėtai artėjama prie ieškomų sprendinių, o pasirinkus mažesnę, kompiuterio skaičiavimo laikas sumažės, bet ši aplinka gali būti per maža norint joje surasti geresnę tikslo funkcijos reikšmę turintį sprendinį.

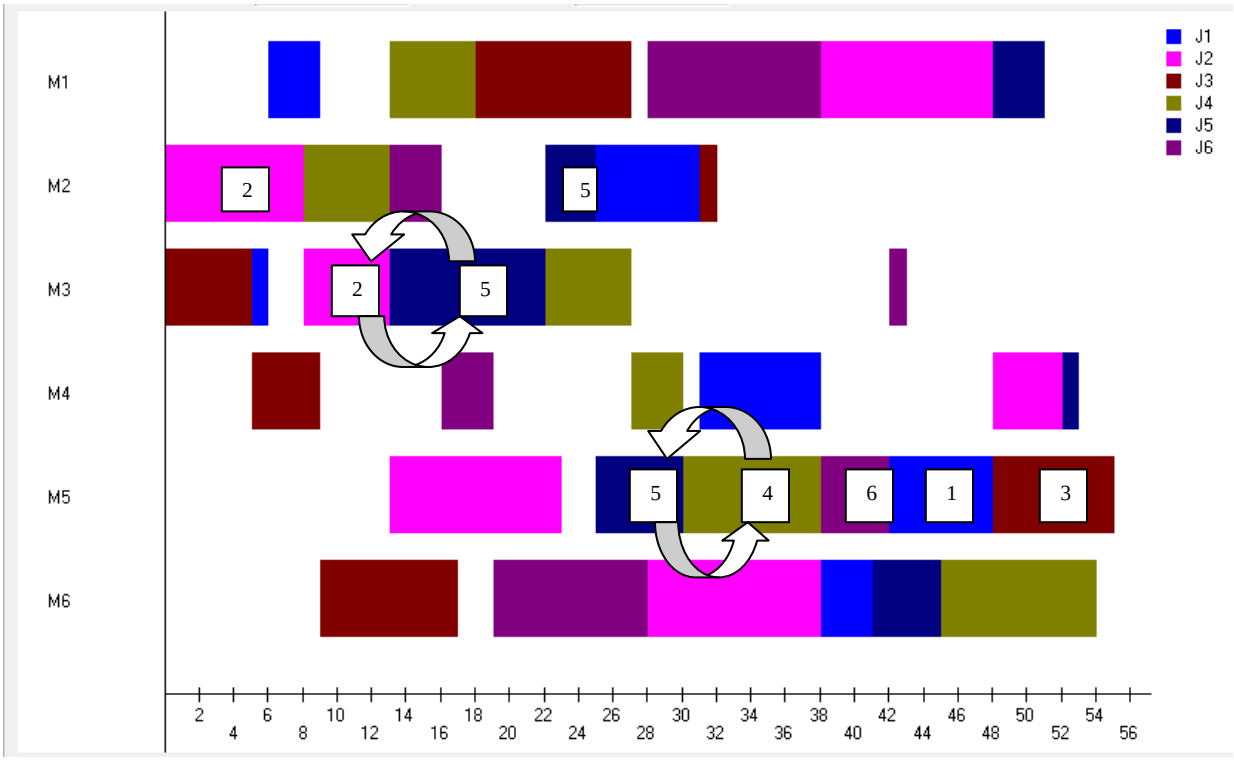
Šiame algoritme naudosime bene efektyviausią aplinką, naudojamą sprendžiant darbo fabriko tvarkaraščių sudarymo uždavinius, kuri nusakoma tam tikrų operacijų, esančių tvarkaraščio ilgiausioje operacijų sekoje, sukeitimais vietomis. Pirmiausiai apibrėšime tvarkaraščio ilgiausią operacijų seką. Tarkime, kad turime kažkokį klasikinio darbo fabriko sprendžiamo uždavinio sprendinį, t.y. uždavinio visų reikiamų atlikti operacijų aibės $O = \{o_k\}, k = \overline{1, NM}$ elementai išdėstyti tam tikra tvarka ir visoms operacijos priskirti jų pradžios bei pabaigos laikai, ir tvarkaraščio visų darbų atlikimo trukmė lygi m. Tuomet turimo sprendinio ilgiausia operacijų seka vadinsime tokį atliekamų operacijų poaibį (elementai pernumeruoti naujai nuo 1 iki n):

$$O' = \{o_k\}, k = \overline{1, n},$$

tenkinantį sąlygas:

- $c_i = s_{i+1}, i = \overline{1, n-1}, c_n = m,$ čia: c_i, s_i - atitinkamai i- osios operacijos pabaigos ir pradžios laikai.
- $m_i = m_{i+1} \vee d_i = d_{i+1}, i = \overline{1, n-1};$ čia d_i, m_i atitinkamai i-osios operacijos darbo ir įrenginio numeris.

Šios sąlygos reiškia, kad ilgiausia operacijų seka vadinsime tokią operacijų aibę, kurios vykdymo trukmė lygi viso tvarkaraščio vykdymo trukmei ir bet kuriuo laiko momentu vykdoma bent viena iš operacijų. Pastebime, kad gretimų ilgiausios operacijų sekos operacijų pradžios ir pabaigos laikai sutampa ir šios operacijos priklauso arba tam pačiam įrenginiui, arba darbui. Iš šių sąlygų galima pastebėti, kad gali egzistuoti daugiau nei viena operacijų seka, tenkinanti minėtas kritinio operacijų kelio sąlygas. Šiuo atveju pasirenkama bet kuri viena seka ir joje atliekami algoritmuose numatyti pertvarkymai, ir po to galime gauti, kad pasirinkta seka jau netenkins ilgiausios operacijų sekos sąlygų, ir bus imama kita seka... Norėdami nusakyti naudojamą aplinką, pasinaudosime pavyzdžiu. Pavyzdžiui, bet kurį sprendinį galima pavaizduoti Ganto diagrama, rodančia darbų užduočių priskyrimą laike atitinkamiems įrenginiams:



2.9 pav. Sprendinio aplinkos elementų sudarymas

Vertikalioje ašyje sužymėti įrenginiai M1,M2,...,M6. Tvarkaraštyje atliekami šeši J1,J2,...,J6 darbai Skirtingiems darbams priklausančios užduotys yra skirtingų spalvų. Operacijos, priklausančios tvarkaraščio ilgiausiai operacijų sekai, sužymėtos skaičiais, šie skaičiai nurodo darbo numerį, kuriam priklauso operacija. Iš 2.9 paveikslo pastebime, kad ilgiausiai operacijų sekai priklauso devynios iš 36 operacijų. Apibrėžę ilgiausią operacijų seką, nusakysime aplinką, kurią pasirinkome paieškos kintamose aplinkose algoritmo realizacijoje. Šiame algoritme pasirenkame, kad bus galimos sukeisti tokios gretimos ilgiausios operacijų sekos užduočių poros, kurios priklauso tam pačiam įrenginiui ir prieš tokią porą ar po tokios poros (taip pat gali būti tenkinamos ir abi sąlygos) atliekama gretima ilgiausios operacijų sekos užduotis priklauso jau kitam įrenginiui. 2.9 paveiksle rodyklėmis pažymėtos dvi operacijų poros, tenkinančios suformuluotas sąlygas. Taigi, Ganto diagrama pavaizduotam sprendiniui galima atlikti du perstatymus ir gauti 2 sprendiniai sudarys pradinio nagrinėto sprendinio aplinką.

Lokaliosios paieškos fazės schema:

1. Imame pradinį sprendinį x'. min = paskutinė skaičiuota darbų trukmė.
2. Priimame $j = 0$.
3. Kol $j < k_{\max}$
   - Atsitiktinai parenkame $x'' \in A(x')$. Apskaičiuojama darbų trukmė $f(x'')$.

- Jei $f(x'') < \min$, tai $\min = f(x'')$, $x' = x''$; j = 0, priešingu atveju j = j + 1.

4. Grąžiname x'.

Čia: $A(x)$- funkcija, grąžinanti atsitiktinai parinktą sprendinio x aplinkos elementą. k_{max} - turimo sprendinio operacijų porų skaičius, kurias pagal aplinkos apibrėžimą galima sukeisti vietomis. Atkreipsime dėmesį, kad lokaliosios paieškos žingsnyje grąžintas sprendinys x' nebūtinai bus lokalusis minimumas, t.y. tvarkaraščio x' aplinkoje gali būti mažesnę tikslo funkcijos reikšmę turinčių sprendinių. Taip yra todėl, kad lokaliosios paieškos fazės metu turimo sprendinio aplinkos elementai k_{max} kartų generuojami nepriklausomai vienas nuo kito ir gali pasitaikyti vienodų. Tad su tam tikra tikimybe galimas įvykis, kad grąžintas sprendinys nebus lokalusis minimumas. Tokios lokaliosios paieškos žingsnio savybės gerokai sumažina galimybes algoritmo generuojamai sprendinių sekai užsiciklinti blogos kokybės lokaliame minimume.

Sužadinimo fazės schemoje parenkame sprendinį $x' \in A^S(x)$, kas reiškia, kad sprendiniui x atliekame S atsitiktinių operacijų sukeitimų vietomis:

1. Imame sprendinį x; $j = 0$.
2. Kol $j < S$
   - Atsitiktinai parenkamas $x' \in A(x)$. Apskaičiuojama darbų trukmė $f(x')$.
   - $x = x'$; $j = j + 1$.
3. Grąžinamas x'.

Svarbūs parametrai šiam algoritmui yra sužadinimų skaičius S ir skaičius K, reiškiantis, kad kas K – ąją iteraciją bus atliekamas algoritmu generuojamos sprendinių sekos sužadinimas. Vėliau atlikdami tyrimus bandysime rasti optimalias šių parametrų reikšmes. Jau iš anksto galima paminėti, kad esant per mažai S reikšmei, algoritmu generuojama sprendiniu seka gali neištrūkti iš blogos kokybės lokaliojo minimumo. Esant per didelei S reikšmei, algoritmu generuojama sprendinių seka gali pavirsti į atsitiktinį klaidžiojimą sprendinių aibėje su minimaliomis galimybėmis rasti aukštesnės kokybės sprendinį. Didėjant parametro K reikšmei, didėja tikimybė, kad kiekvienai sužadinimo procedūrai tenkantis sprendinys x yra lokalusis minimumas nagrinėjamoje aplinkoje. Atkreipsime dėmesį, kad jei kažkurioje iteracijoje pradedant vykdyti lokaliosios paieškos procedūrą prieš tai buvusioje iteracijoje nebuvo vykdyta sužadinimo procedūra, lokaliosios paieškos pradžioje patvirtinamas atsitiktinai parinktas šiai fazei pateikto sprendinio aplinkos elementas, ir jis negali būti blogesnis už prieš tai vykusioje lokaliosios paieškos fazėje paskutinį atsitiktinai generuotą sprendinį (galimas tik geresnio patvirtinimas).

Vienas iš trūkumų, kurį galima įžvelgti aptartame paieškos kintamose aplinkose algoritme, tai yra galimybė grįžti prie neseniai nagrinėtų sprendinių. Šiame algoritme iteracija po iteracijos vis nuosekliai atliekamos lokaliosios paieškos ir sužadinimo procedūros. Taigi, po kiekvienos sužadinimo procedūros gautam blogesniam sprendiniui pritaikant aprašytą lokaliosios paieškos procedūrą, galime grįžti į tą patį lokalųjį minimumą, kuris buvo gautas dar prieš minėtą sužadinimą atliktoje lokaliosios paieškos procedūroje.

2.6 TABU – ATKAITINIMO MODELIAVIMO PAIEŠKOS ALGORITMAS

Tabu paieška – vienas efektyviausių metaeuristinių metodų, naudojamų kombinatorinio optimizavimo uždaviniams spręsti. Šio metodo esmė – laikinas draudimas grįžti į prieš tai buvusius sprendinius. Šiuo tikslu sudaromas tam tikro ilgio (pastovaus ar kintamo) vadinamasis tabu sąrašas, kuris optimizavimo procedūros pradžioje yra tuščias, o vėliau iteracinio proceso metu nuolat atnaujinamas. Dar viena specifinė tabu paieškos algoritmų savybė – tai aspiracijos kriterijus, kuris nusako kriterijų, kada galima atlikti tabu sąrašo draudžiamą sprendinio pakeitimą. Literatūroje [6], [7] bene populiariausias aspiracijos kriterijus leidžia atlikti tokį sprendinio sukeitimą, esantį tabu sąraše, kurį atlikus gaunamas mažesnę tikslo funkcijos reikšmę turintis sprendinys už iki šiol geriausią rastą sprendinį. Literatūros šaltinyje [7] pastebima, kad tabu paieškos algoritmų gaunamų sprendinių kokybė priklauso nuo pradinio sprendinio.

Atkaitinimo modeliavimas nėra efektyviausiems metaeuristiniams metodams priskiriama technika. Atkaitinimo modeliavimo algoritmuose taip pat generuojamos tam tikros sprendinių sekos. Pradėjus nuo kažkokiu būdu sugeneruoto pradinio sprendinio, kiti algoritmu gaunamos sprendinių sekos elementai atrenkami iš esamo sprendinio pasirinktos aplinkos tokiais kriterijais:

- visuomet patvirtinamas geresnis sprendinys už esamą,
- su tam tikra tikimybe patvirtinami blogesni sprendiniai už esamą.

Antrasis blogesnių sprendinių patvirtinimo kriterijus sudaro galimybes ištrūkti iš lokaliųjų minimumų aplinkų. Pati blogesnių sprendinių priėmimo tikimybė paprastai turi eksponentinę išraišką, t.y. blogesnių sprendinių patvirtinimo tikimybė eksponentiškai mažėja didėjant blogo sprendinio ir esamo sprendinio tikslo funkcijų skirtumui. Be to, blogesnių sprendinių patvirtinimo tikimybės išraiška priklauso ir nuo temperatūros, kuri iteracinio proceso metu paprastai mažėja bei mažėja tam, kad blogesnių sprendinių patvirtinimo tikimybė tęsiantis paieškos procesui (ir artėjant prie optimalaus sprendinio) vis didėtų. Blogesnių sprendinių patvirtinimo tikimybės pavyzdžiu galėtų būti išraiška

$$P = \frac{e^{f(x)-f(x')}}{Temp}$$

čia: x - esamas, x' - blogesnis sprendinys, Temp – temperatūra. Literatūros šaltinyje [7] pastebėta, kad atkaitinimo modeliavimo algoritmais gaunami optimizavimo uždavinių sprendimo rezultatai mažai priklauso nuo pradinio sprendinio, todėl prie tabu paieškos pridėdami atkaitinimo modeliavimo elementus tikėsimės, kad gauto junginio rezultatai sprendžiant optimizavimo uždavinius mažiau priklausys nuo pradinio sprendinio, nei paprasčiausios tabu paieškos atveju.

Tabu – atkaitinimo modeliavimo paieškos algoritme pradedama taip pat nuo atsitiktinai sugeneruoto pradinio sprendinio. Toliau sprendinių seka iteracija po iteracijos generuojama parenkant geriausią (tvarkaraščio darbų trukmės prasme) tabu sąrašu nedraudžiamą aplinkos elementą arba tabu draudžiamą sprendinį, kuris yra geresnis už iki šiol rastą geriausią. Sprendinių aplinka apibrėžiama lygiai taip pat, kaip ankstesnio paieškos kintamose aplinkose algoritmo atveju. Parinkus kitą sekos sprendinį, fiksuoto ilgio tabu sąrašas atnaujinamas. Taip procesas tęsiamas tol, kol po paskutinio minimalaus sprendinio pagerinimo sugeneruojama Maxiter iteracijų. Praėjus po minimalaus sprendinio pagerinimo Maxiter iteracijų, paieškos procesas su tuščiu tabu sąrašu pradedamas nuo pirmojo elitinių sprendinių sąrašo L tvarkaraščio, jį iš to sąrašo pašalinus. Su naujai paimtu sprendiniu iš elitinių sprendinių sąrašo paieškos procesas tęsiamas Maxiter iteracijų nuo paskutinio minimalaus sprendinio pagerinimo arba nuo paskutinio sprendinio paėmimo iš elitinių sprendinių sąrašo, po to vėl imamas pirmasis sąrašo sprendinys ir t.t. Aptarsime elitinių sprendinių sąrašą L ir jo formavimą. Šio algoritmo paieškos proceso metu saugomas tam tikra geriausių, dar vadinamų elitiniais, sprendinių aibė. Maksimalus šio sprendinių sąrašo ilgis laikomas konstanta ir lygus 50. Į šį elitinių tvarkaraščių sąrašą patenka bet kuris paieškos proceso metu gautas sprendinys, tuo metu buvęs geresnis už iki šiol rastą geriausią, taip pat su tam tikra tikimybe gali patekti blogesni. Blogesnio sprendinio x patekimo į elitinių tvarkaraščių sąrašą L tikimybė:

$\frac{e^{f(x_{\min})-f(x)}}{Temp}$, čia $Temp = \frac{f(x_{\min})}{T}$, $f(x)$ – sprendinio x tikslo funkcijos reikšmė, T – parametras. (2.7)

Pateiktuose žymėjimuose simboliu $f(x_{\min})$ žymima gauto geriausio sprendinio darbų trukmė. Pastebėsime, kad ši reikšmė uždavinio sprendimo proceso metu mažėja. Kadangi T parametras nustatomas prieš pradedant algoritmo skaičiavimus ir vėliau nekeičiamas, tai iš pateiktos Temp išraiškos pastebime, kad Temp reikšmė tęsiantis uždavinio sprendimo procesui mažės, todėl blogesnių sprendinių priėmimo į sąrašą L tikimybė tuo pat metu didės.

Svarbūs trys tabu – atkaitinimo modeliavimo algoritmo efektyvumą lemiantys parametrai yra T, Maxiter, tabu sąrašo ilgis. Mažėjant T reikšmei, blogesnių sprendinių patekimo į sąrašą L tikimybė mažėja. Esant per mažai T reikšmei, sąrašas L bus apytuštis, tad paieška apie turimą lokalų minimumą bus neefektyvi, vyks netoli turimo geriausio sprendinio. Esant per didelei T reikšmei, sąrašas L persipildys žemos kokybės sprendiniais, ir juos tyrinėjant bus sugaišta daugiau kompiuterio laiko. Esant per mažai tabu sąrašo ilgio reikšmei, paieškos procesas dažnai patenka į ciklus, o esant per didelei, paieškos procesas įgyja per daug apribojimų. Esant didesnei Maxiter reikšmei, didėja galimybės rasti geresnį sprendinį nuosekliai tabu paieškos procesu tyrinėjant L elementus, tačiau per tą patį skaičiavimų laiką bus išnagrinėta mažiau L sprendinių. Tiriamojoje darbo dalyje bandysime įvertinti rekomenduotinas aptartų trijų parametrų reikšmes ar jų intervalus.

Tabu – atkaitinimo modeliavimo paieškos algoritmo schema:

1. **Pradžia.** Atsitiktinai generuojamas pradinis sprendinys x'.

Pagalbiniai kintamieji x', $x_{\min}$, j, a. Įvedamos parametrų MaxIter, T reikšmės. Fiksuojamas tabu sąrašo ilgis. Tabu sąrašas tuščias. $x_{\min} = x'$, $a = j = 0$.

2. **Kartoti:**

- Generuojami turimo sprendinio x' visi aplinkos elementai.
- Jei tenkinamas aspiracijos kriterijus t.y. kažkurio aplinkos sprendiniui y galioja $f(y) < f(x_{\min})$, priskiriame $x' = y$, priešingu atveju x' priskiriama geriausias sugeneruotos aplinkos sprendinys, nedraudžiamas tabu. Jei visi aplinkos sprendiniai draudžiami, atliekamas seniausias tabu draudžiamas sukeitimas ir x' priskiriamas atitinkamas sprendinys.
- Tabu sąrašo elementus perstūmus per vieną vietą, atlaisvinama pirmoji jo vieta, ir atlikto sprendinio sukeitimo duomenys (sukeistos operacijų poros darbo ir įrenginių numeriai) įterpiami į draudimų sąrašo pirmąją vietą.
- Apskaičiuojama tikslo funkcijos reikšmė $f(x')$ bei nustatoma temperatūra $Temp = \frac{f(x_{\min})}{T}$. Generuojamas atsitiktinis skaičius $at \sim T[0,1]$.
- Jei $f(x') < f(x_{\min})$ arba $\frac{e^{f(x_{\min}) - f(x')}}{Temp} > at$, tai sprendinys x' įterpiamas į sąrašo L pirmąją vietą.
- $j = j + 1$. Jei $f(x')$ reikšmės periodiškai kartojasi (paieškoje ciklas), tai sprendiniui x' sukeičiamos atsitiktinai parinktos bet kurios (nebūtinai priklausančios ilgiausiai

operacijų sekai) viena po kitos einančios operacijų poros, priklausančios tam pačiam įrenginiui.

- Jei $j - a > Maxiter$:

  Sprendiniui x' priskiriamas pirmasis sąrašo L elementas (jei sąrašas L nėra tuščias), kuris po priskyrimo iš minėto sąrašo pašalinamas. Jei sąrašas L tuščias, sprendiniui x' priskiriamas turimas geriausias sprendinys $x_{\min}$. Iš tabu sąrašo pašalinami visi elementai, $a = j$.

Saugomas minimalus sprendinys. Jei $f(x') < f(x_{\min})$, tai $x_{\min} = x'$, $a = j$.

čia: $f(x)$ - sprendinio x tikslo funkcijos reikšmė.

- Grįžtama į antrojo žingsnio pradžią.

3. **Pabaiga.** Grąžinamas gautas geriausias sprendinys $x_{\min}$.

Pastebėsime, kad įvedus pakankamai didelę parametro Maxiter reikšmę, galima gauti vien tik tabu paieška paremtą algoritmą t.y. šiuo atveju paieškos proceso metu nebus nagrinėjami sąrašo L sprendiniai ir vientisas tabu paieškos procesas tęsis nuo pat pradinio sprendinio tam tikrą norimą iteracijų skaičių.

3. TIRIAMOJI DALIS

Priminsime, kad pagrindinis šio darbo tiriamosios dalies tikslas – realizuotų algoritmų efektyvumo įvertinimas ir optimizavimas sprendžiant konkrečius darbo fabriko tvarkaraščių sudarymo uždavinius. Algoritmų efektyvumą optimizuosime bandydami surasti kiek galima tinkamesnes jų parametrų reikšmes ar bent intervalus.

3.1 REKOMENDUOTINŲ ALGORITMŲ PARAMETRŲ REIKŠMIŲ RADIMO SCHEMA

Aptarsime abiejų realizuotų paieškos kintamose aplinkose bei tabu – atkaitinimo modeliavimo algoritmų rekomenduotinų parametrų įvertinimo proceso bendruosius bruožus. Visų pirma, prieš nagrinėdami abu algoritmus pasirenkame 10 įvairios apimties, sunkiau sprendžiamų, etaloninių darbo fabriko problemų. Uždavinius pasirenkame iš Éric Taillard puslapio [8] ir [9] šaltinio. Įvairiuose literatūros šaltiniuose ([5], [7], [10] ir kt.) kažkokių nagrinėjamų algoritmų efektyvumas vertinamas sprendžiant uždavinius, esančius pateiktuose uždavinių rinkiniuose. Be to, yra surasti visų šių etaloninių uždavinių minimalaus sprendinio darbų trukmės įverčiai, bei įvairiuose ([5], [7], [10] ir kt.) šaltiniuose galima rasti geriausių rastų sprendinių darbų trukmes, tad abejais algoritmais gaunamų sprendinių darbų trukmės bus lyginamos su šiomis reikšmėmis. Pasirinktų uždavinių lentelė:

3.1 Lentelė

Sprendžiamos darbo fabriko problemos

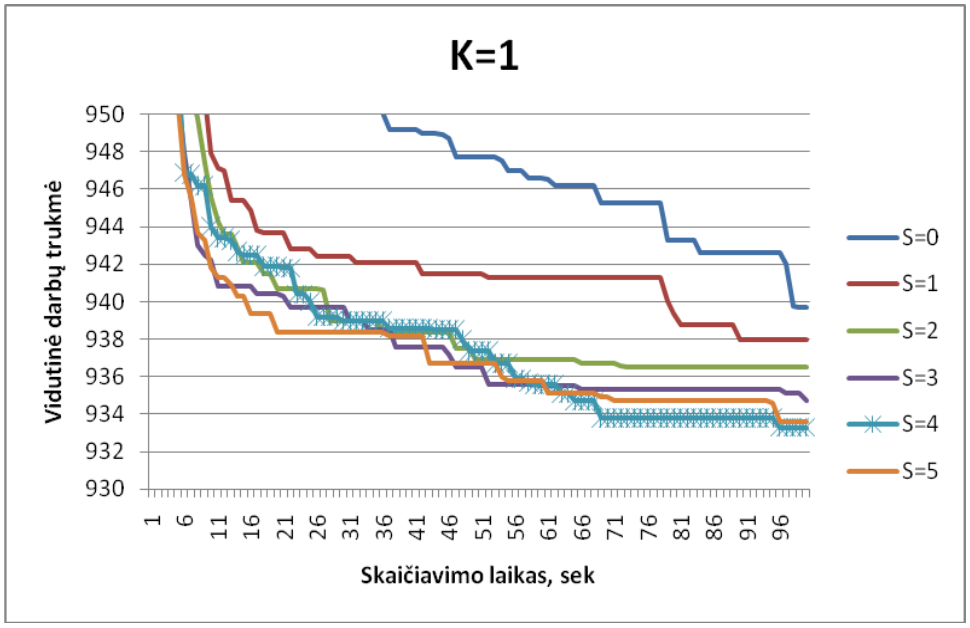
Uždavinio pavadinimas	Apimtis (NxM)	Minimalaus sprendinio darbų trukmės įvertis	Geriausio rasto sprendinio darbų trukmė
ft10	10x10	930	930
la25	15x10	977	977
la40	15x15	1222	1222
la29	20x10	1142	1152
abz9	20x15	661	679
abz8	20x15	645	665
YN4	20x20	918	968
TAI41	30x20	1859	2018
SWV15	50x10	2885	2904
TAI62	50x20	2869	2869

Pasirinkę konkrečius uždavinius ir turėdami tikslą rasti kiek galima geresnes algoritmų parametrų reikšmes, atliksime šių uždavinių sprendimo eksperimentus abejais algoritmais, keisdami

realizuotų algoritmų parametrų reikšmes. Tabu – atkaitinimo paieškos algoritmo parametrą Maxiter keisime intervale [1500,4500] žingsniu 1000, kitą šio algoritmo parametrą T bandysime intervale [250,450] žingsniu 50, trečią šio algoritmo parametrą „tabu sąrašo ilgį“ bandysime intervale [5,10] žingsniu 1. Tabu paieškos algoritmas bus gaunamas įvedus Maxiter reikšmę $5 \cdot 10^6$, tai yra pakankama reikšmė visiems uždaviniams, kad 0 – 2000 sekundžių vyktų vientisas tabu paieškos procesas (mažiausio nagrinėto uždavinio FT10 atveju per sekundę atliekama maždaug 2500 iteracijų, o didesnių uždavinių iteracijų per sekundę bus atliekama mažiau proporcingai jų dydžiui). Paieškos kintamose aplinkose algoritmo parametrą S bandysime intervale [0,3] žingsniu 1, o kitą šio algoritmo parametrą K bandysime intervale [1,3] žingsniu 1. Kartais išbandysime ir keletą reikšmių už nurodytų intervalų ribos. Toliau abiejų algoritmų parametrų kombinacijas išbandysime su kiekvienu pasirinktu uždaviniu atlikdami po 5 – 10 eksperimentų, trunkančių po 100 – 900 sekundžių. Kiekvieno eksperimento metu kas sekundę fiksuosime iki tol gauto geriausio sprendinio darbų trukmę. Taigi, atlikus vienu iš algoritmų su tam tikru parametrų rinkiniu nurodytus 5 – 10 eksperimentų galėsime rasti įvairias darbų trukmių charakteristikas kiekvienu skaičiavimų laiko momentu. Laikysime, kad geriausias iš nagrinėtų parametrų rinkinių sprendžiamam uždaviniui yra tas, kuriam esant eksperimentų metu gauta empirinė vidutinė darbų trukmė yra mažiausia po nagrinėto skaičiavimų laiko. Eksperimentuose naudoto kompiuterio duomenys: procesorius Intel Pentium E5200, dirbantis 3,75 GHz dažniu, operatyviosios atminties kiekis 4 GB.

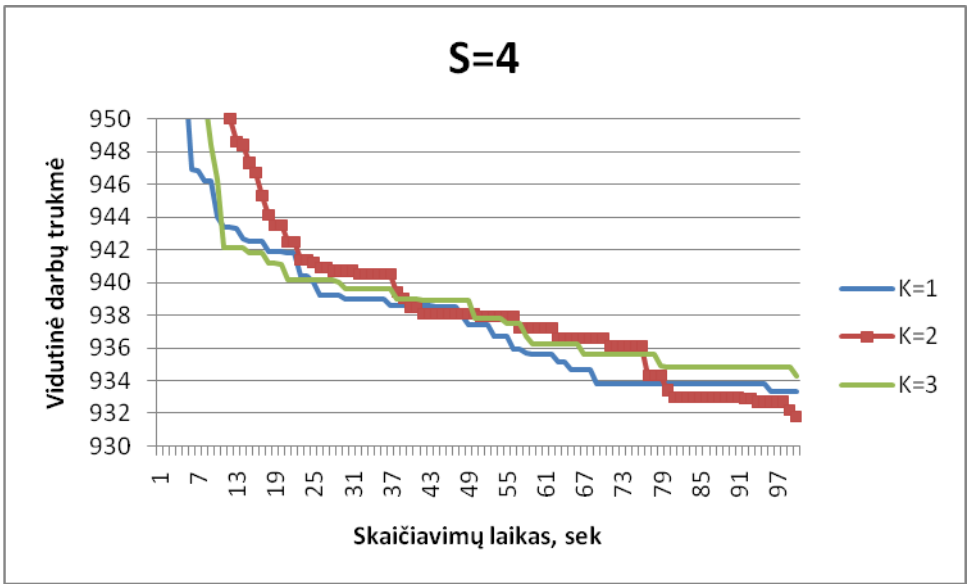
3.2 PAIEŠKOS KINTAMOSE APLINKOSE ALGORITMO REKOMENDUOTINŲ PARAMETRŲ RADIMAS

Šiame skyrelyje keletui nagrinėtų darbo fabriko tvarkaraščių sudarymo uždavinių pateiksime atliktų tyrimų rezultatus, atliktus paieškos kintamose aplinkose algoritmui, iš kurių pagal ankstesniame skyrelyje nusakytus kriterijus atrinksime rekomenduotiną parametrų rinkinį. Imame pirmąjį nagrinėjamą uždavinį FT10, kurio apimtis 10 × 10 (10 darbų ir 10 įrenginių). Vieno eksperimento kompiuterio skaičiavimų trukmę fiksuojame ties 100 sekundžių. Kadangi paieškos kintamose aplinkose algoritmas turi 2 parametrus S ir K, tai visų pirma fiksuojame $K = 1$, o parametro S reikšmę didiname nuo 0 iki 5. Gauname 6 skirtingas parametrų kombinacijas ir su kiekviena iš jų atliekame po 10 eksperimentų. Kaip minėjome anksčiau, kiekvieno eksperimento metu kas sekundę išsaugoma gauto geriausio sprendinio darbų trukmė. Paveiksle ir pateiktos šios empirinės vidutinės darbų trukmės kiekvienai iš parametrų kombinacijų.



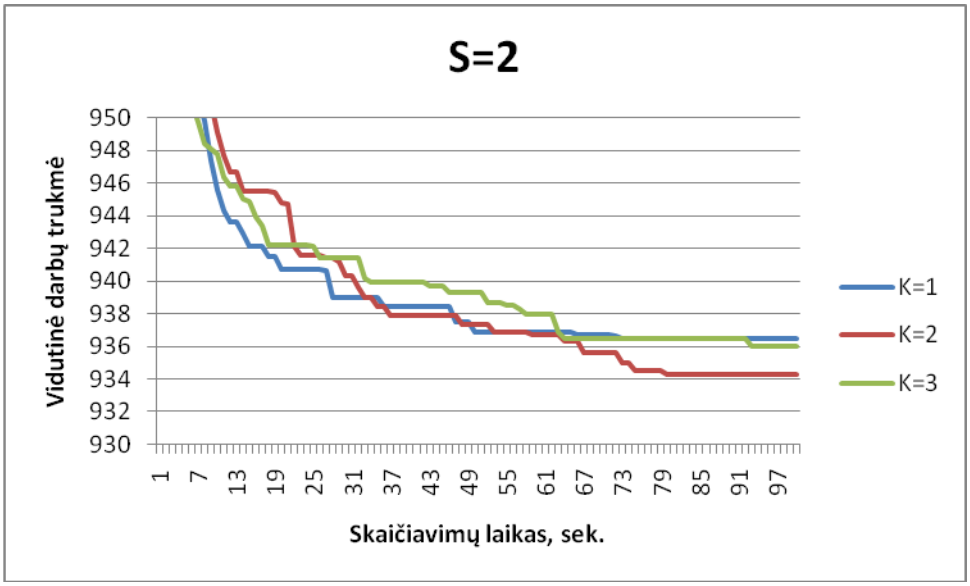
3.1 pav. Darbų trukmių empirinių vidurkių palyginimas kintant S

Iš pateiktų grafikų pastebime, kad mažiausia empirinė vidutinė darbų trukmė po nagrinėto 100 sekundžių skaičiavimų laiko yra parametrams $K=1, S=4$. Toliau fiksuojame parametrą $S=4$, kitą parametrą K keičiame nuo 2 iki 3 (su reikšme 1 jau eksperimentai atlikti). Atliktų eksperimentų empirinės vidutinės darbų trukmių grafikai gautoms trims parametrų kombinacijomis.

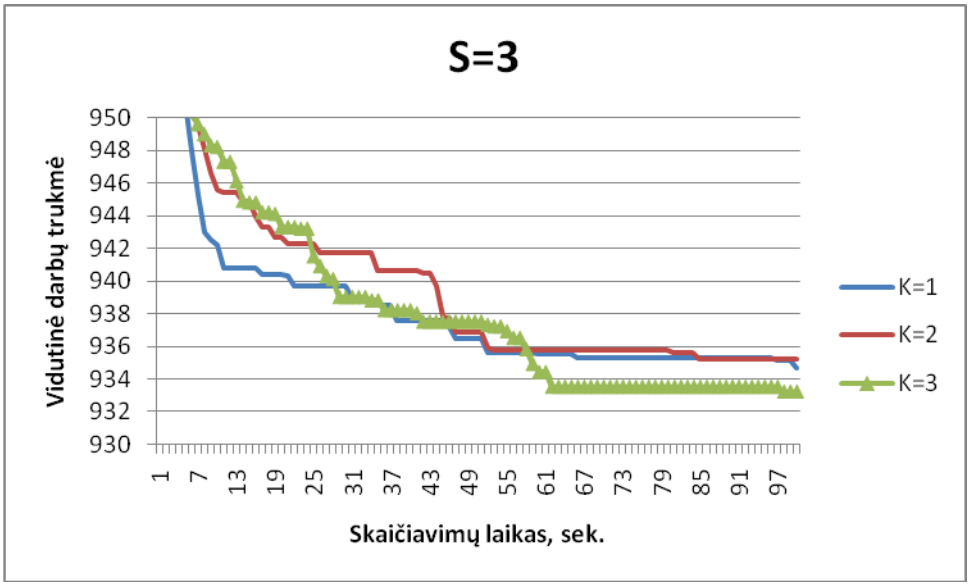


3.2 pav. Darbų trukmių empirinių vidurkių palyginimas kintant K

Iš pateiktų grafikų pastebime, kad mažiausia vidutinė darbų trukmė po 100 sekundžių gaunama su $K=2, S=4$. Dar panagrinėjame likusias parametrų kombinacijas:



3.3 pav. Darbų trukmių empirinių vidurkių palyginimas kintant K



3.4 pav. Darbų trukmių empirinių vidurkių palyginimas kintant K

Iš pateiktų grafikų pastebime, kad atlikus likusių kombinacijų eksperimentus, vidutinė gauta trukmė nebuvo sumažinta, ir gauname, kad šio uždavinio rekomenduotinas rinkinys yra $K = 2, S = 4$. Paminėsime, kad sprendžiant šį FT10 10×10 uždavinį sprendinio tikslo funkcija skaičiuojama buvo vidutiniškai 1,71·106 kartų per 100 sekundžių. Taigi, iš viso perrinkta buvo nedaugiau nei $1{,}71 \cdot 10^6$ variantų, kas yra žymiai mažiau už visų galimų uždavininio sprendinių skaičių $10!^{10} = 3{,}96 \cdot 10^{65}$.

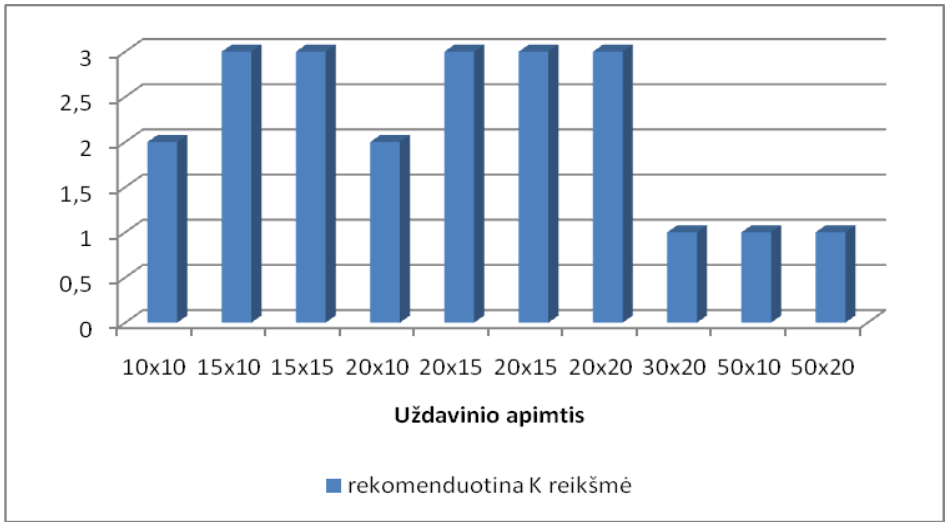
Analogiškai eksperimentus atliekame su visais nagrinėjamais uždaviniais ir jiems randame rekomenduotinas sprendimo parametrų reikšmes. Surastų rekomenduotinų parametrų reikšmių lentelė visiems nagrinėtiems uždaviniams:

3.2 Lentelė

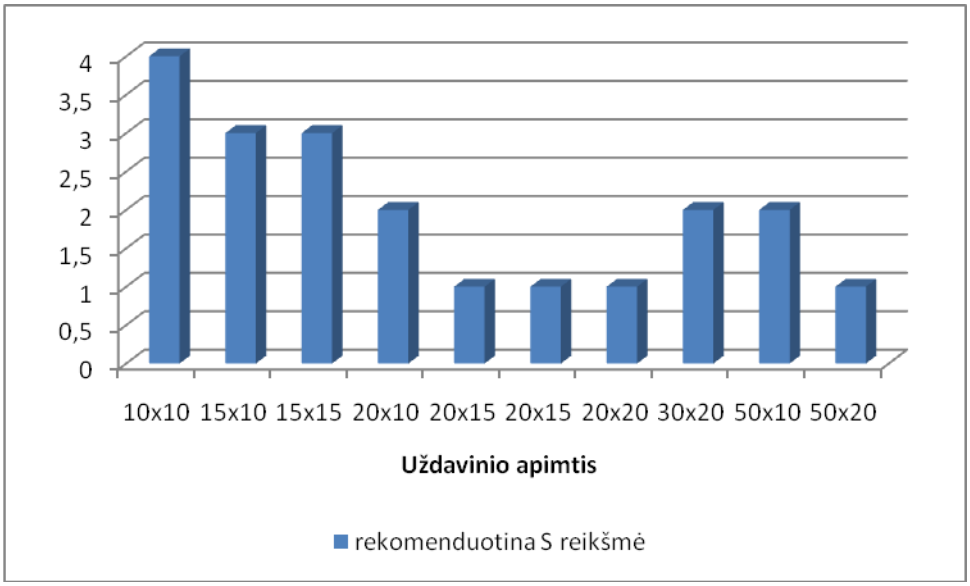
Rastos rekomenduotinos paieškos kintamose aplinkose algoritmo parametrų reikšmės

Uždavinys	Apimtis (NxM)	Paieškos kintamose aplinkose rekomenduotini parametrai	
		S	K
ft10	10x10	4	2
la25	15x10	3	3
la40	15x15	3	3
la29	20x10	2	2
abz9	20x15	1	3
abz8	20x15	1	3
YN4	20x20	1	3
TAI41	30x20	2	1
SWV15	50x10	2	1
TAI62	50x20	1	1

Nesunku pastebėti, kad pateiktos rekomenduotinos parametrų reikšmės kinta įvairiuose uždaviniuose. Taigi, kažkokios vienos parametrų kombinacijos, kuri geriausiai spręstų bent jau pateiktus uždavinius nėra. Pateikiame abiejų paieškos kintamose aplinkose gautų rekomenduotinų parametrų reikšmių stulpelines diagramas kintant uždavinio apimčiai:



3.5 pav. Rekomenduotinų parametro K reikšmių stulpelinė diagrama



3.6 pav. Rekomenduotinų parametro S reikšmių stulpelinė diagrama

Toliau pateikiame nagrinėtų uždavinių sprendimo rezultatus su rastomis rekomenduotinomis parametrų reikšmėmis:

3.3 Lentelė

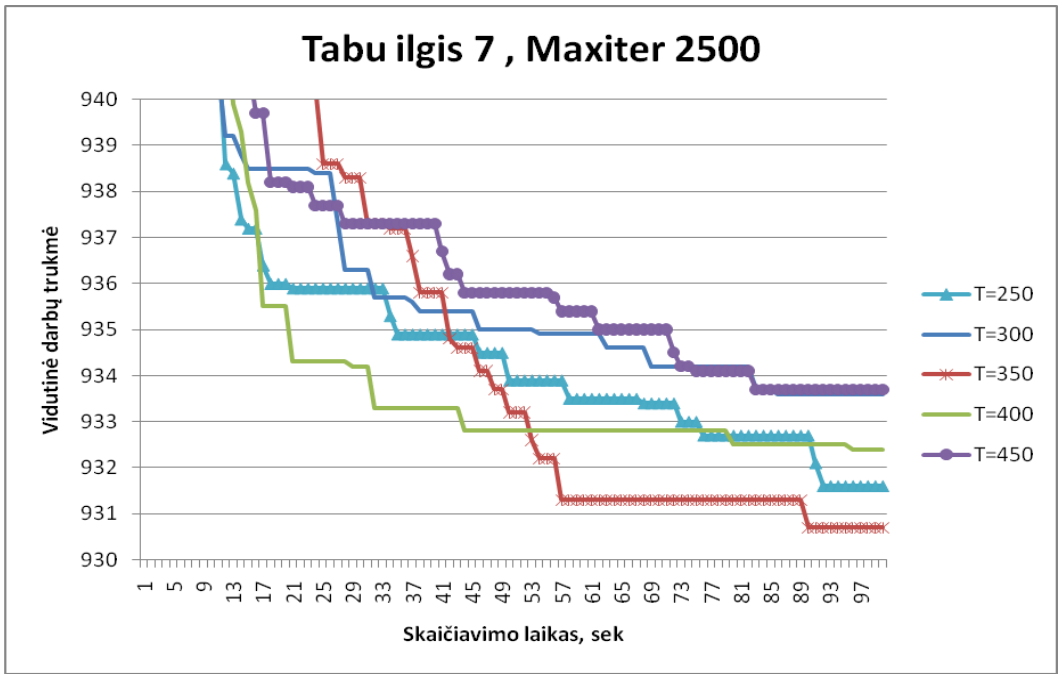
Uždavinių sprendimo paieškos kintamose aplinkose rezultatų suvestinė

Uždavinys	Apimtis (NxM)	ODT	GŽS	Paieška kintamose aplinkose				Skaičiavimų laikas, sek.
				VDT	Ev, %	MDT	Em, %	
ft10	10x10	930	930	931,8	0,19	930	0,00	100
la25	15x10	977	977	979,3	0,24	977	0,00	100
la40	15x15	1222	1222	1231,6	0,79	1228	0,49	100
la29	20x10	1152	1152	1169,7	1,53	1167	1,30	300
abz9	20x15	661	679	691,8	4,66	688	4,08	300
abz8	20x15	645	665	681,1	5,60	673	4,34	300
YN4	20x20	918	968	985,4	7,34	976	6,32	300
TAI41	30x20	1859	2018	2070,4	11,37	2065	11,08	900
SWV15	50x10	2885	2904	2937	1,80	2917	1,11	900
TAI62	50x20	2869	2869	2887,8	0,66	2881	0,42	900
Vidutinė					3,42%		2,91%	

čia: ODT – optimalios darbų trukmės įvertis; GŽS – geriausio žinomo sprendinio darbų trukmė; VDT – darbų trukmių empirinis vidurkis; Ev – VDT santykinė paklaida nuo ODT; MDT – eksperimentuose gauta minimali darbų trukmės reikšmė; Em – MDT santykinė paklaida nuo ODT.

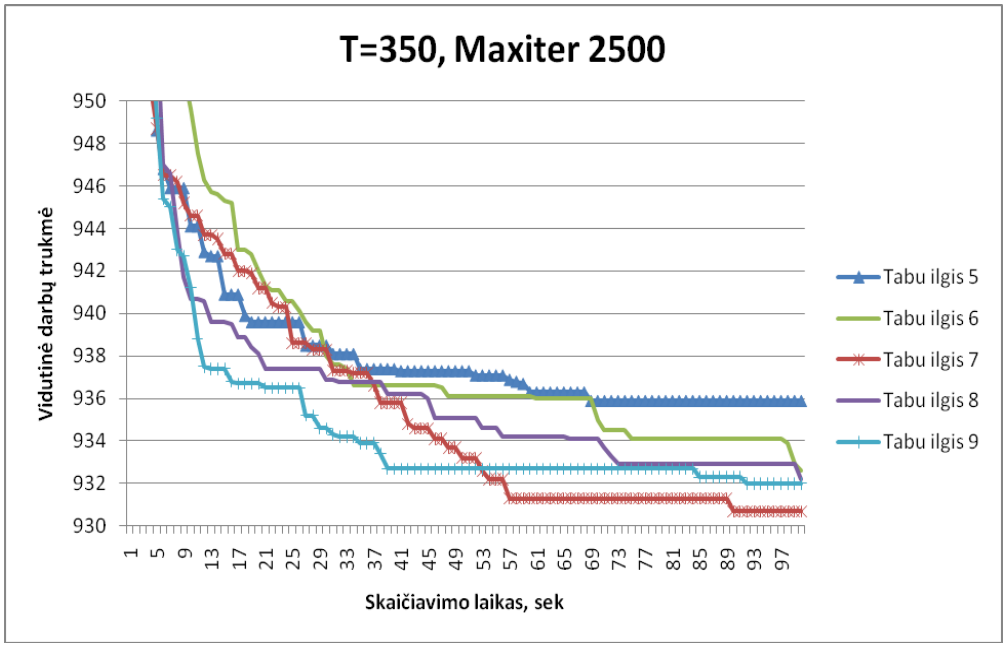
3.3 TABU – ATKAITINIMO MODELIAVIMO IR TABU PAIEŠKOS ALGORITMŲ REKOMENDUOTINŲ PARAMETRŲ RADIMAS IR ALGORITMŲ PALYGINIMAS

Šiame skyrelyje keletui nagrinėtų darbo fabriko tvarkaraščių sudarymo uždavinių pateiksime atliktų tyrimų rezultatus, atliktus tabu – atkaitinimo modeliavimo paieškos algoritmui. Imame pirmąjį nagrinėjamą uždavinį FT10, kurio apimtis 10 × 10. Vieno eksperimento kompiuterio skaičiavimų trukmę fiksuojame ties 100 sekundžių. Kadangi tabu – atkaitinimo modeliavimo paieškos algoritmas turi 3 parametrus Maxiter, T ir tabu sąrašo ilgis, tai visų pirma fiksuojame $Maxiter = 2500$, tabu sąrašo ilgio reikšmę priimame lygią 7, o parametro T reikšmę didiname nuo 250 iki 450 žingsniu 50. Gauname 5 skirtingas parametrų kombinacijas ir su kiekviena iš jų atliekame po 10 eksperimentų. Paveiksle ir pateiktos šios empirinės vidutinės darbų trukmės kiekvienai iš parametrų kombinacijų:



3.7 pav. Empirinių vidutinių darbų trukmių grafikai FT10 uždaviniui kintant parametrui T

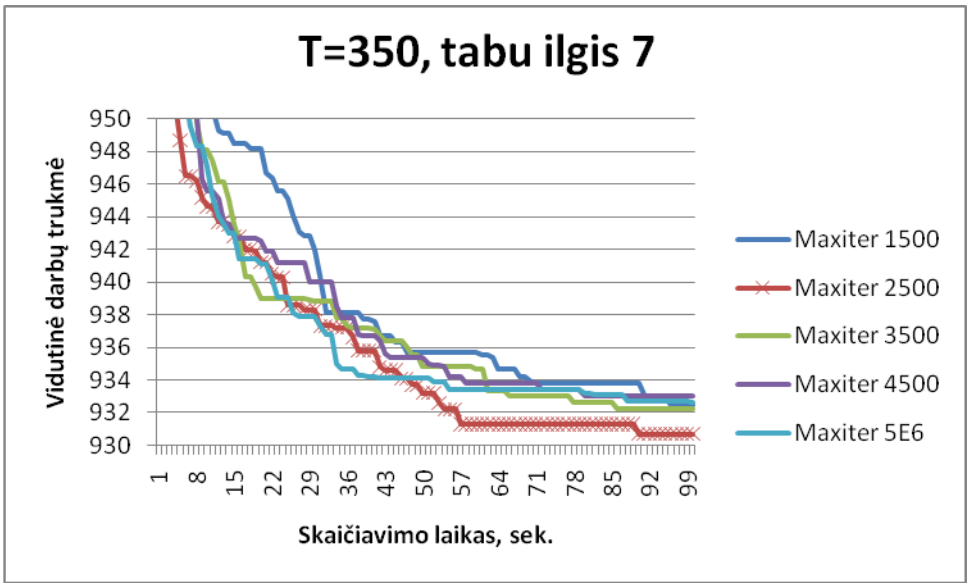
Iš grafikų pastebime, kad mažiausias empirinis darbų trukmių vidurkis po 100 sekundžių buvo gautas, kai $T = 350$. Nežymiai didesnė vidutinė darbu trukmė buvo gauta su $T = 250$, parametro T kitimas labai didelės įtakos galutinėms vidutinėms darbų trukmėms neturėjo – su visomis T reikšmėmis gautos vidutinės darbų trukmės nėra didesnės už optimalią darbų trukmę daugiau nei 0,5%. Toliau fiksuojame geriausią reikšmę $T = 350$, pradinę $Maxiter = 2500$ ir keičiame tabu sąrašo ilgio reikšmę nuo 5 iki 9 žingsniu 1. Paveiksle pateiktos empirinės vidutinės darbų trukmės kiekvienai iš parametrų kombinacijų:



3.8 pav. Empirinių vidutinių darbų trukmių grafikai FT10 uždaviniui kintant tabu sąrašo ilgiui

Iš grafikų pastebime, kad ankstesniuose eksperimentuose gauta vidutinė darbų trukmė po 100 sekundžių su tabu sąrašo ilgiu lygiu 7, nebuvo pagerinta, tačiau pasirinku tabu sąrašo ilgį lygų 6, 8, ar 9 gaunamos vidutinė darbų trukmės yra nežymiai didesnės, bet su tabu ilgio reikšme, lygia 5, gaunamas darbų trukmių empirinis vidurkis po 100 sekundžių pastebimai didesnis.

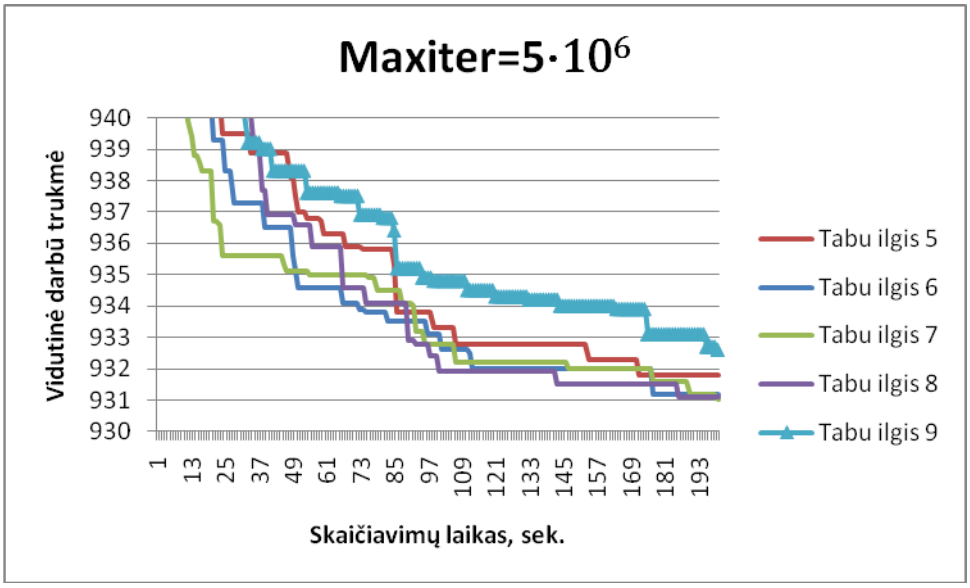
Norėdami ištirti parametro Maxiter parinkimo įtaką, fiksuojame geriausių gautų likusių dviejų parametrų reikšmes $T = 350$ ir tabu sąrašo ilgį lygų 7. Parametrą Maxiter keičiame nuo 1500 iki 4500 žingsniu 1000 ir šis parametras dar įgyja reikšmę $5 \cdot 10^6$ (vien tik tabu paieškos atvejis). Paveiksle pateiktos empirinės vidutinės darbų trukmės kiekvienai iš parametrų kombinacijų:



3.9 pav. Empirinių vidutinių darbų trukmių grafikai FT10 uždaviniui kintant parametrui Maxiter

Iš grafikų pastebime, kad iki šiol geriausia laikytos parametrų kombinacijos $T = 350$, $Maxiter = 2500$, tabu sąrašo ilgis 7 empirinė vidutinė darbų trukmė nebuvo sumažinta, nors keičiant parametrą Maxiter gaunamos tik nežymiai didesnės vidutinės darbų trukmės.

Norėdami atlikti vien tik tabu paieškos efektyvumo įvertinimą sprendžiant pasirinktus darbo fabriko uždavinius ir rasti optimalią jau tik vieno parametro – tabu sąrašo ilgio reikšmę, fiksuojame parametrą $Maxiter = 5 \cdot 10^6$ (parametro T reikšmė šiuo atveju jau neturi jokios įtakos) ir keičiant tabu sąrašo ilgį nuo 5 iki 9 žingsniu 1, gauname tokias empirinių darbų trukmių vidurkių reikšmes kiekvienam parametrų rinkiniui:



3.10 pav. Empirinių darbų trukmių vidurkių grafikai FT10 uždaviniui kintant tabu ilgiui

Iš paveiksle esančių grafikų pastebime, kad vien tik tabu paieškos atveju, darbų trukmių empirinio vidurkio artėjimas prie optimalios reikšmės lėtesnis nei geriausios tabu – atkaitinimo modeliavimo parametrų kombinacijos atveju. Dar galime pastebėti, kad tabu paieškos atveju su beveik visomis tabu ilgio reikšmėmis gaunamos panašios vidutinės darbų trukmės, tik su tabu sąrašo ilgio reikšme, lygia 9, gaunami rezultatai akivaizdžiai išsiskiria iš likusių ir yra blogesni. Sprendžiant šį FT10 10×10 uždavinį sprendinio tikslo funkcija skaičiuojama buvo vidutiniškai $1{,}6 \cdot 10^6$ kartų per 100 sekundžių. Taigi, iš viso perrinkta buvo nedaugiau nei $1{,}6 \cdot 10^6$ variantų, kas yra žymiai mažiau už visų galimų uždavininio sprendinių skaičių $10!^{10} = 3{,}96 \cdot 10^{65}$.

Pratęsę aprašytą eksperimentų procesą su likusiais devyniais didesnės apimties darbo fabriko uždaviniais, gauname likusių uždavinių rekomenduotinų parametrų kombinacijas. Tyrimo procesas parodė, kad uždaviniuose, turinčiuose 400 operacijų ir daugiau, parametrų reikšmėms T ir Maxiter įgyjant atitinkamai 450 ir 4500, t.y. maksimalioms tirtoms, visada buvo gaunami mažiausi empiriniai darbų trukmių vidurkiai. Todėl buvo kiek praplėsti minėtų dviejų parametrų įgyjamų

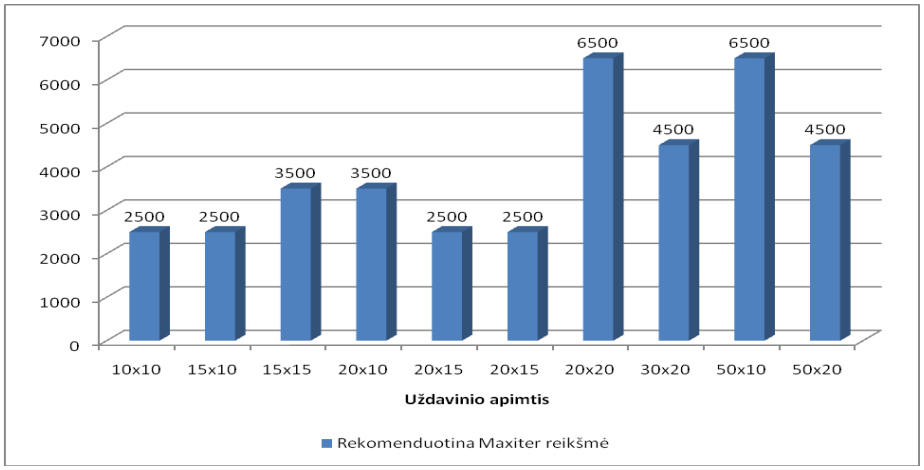
reikšmių intervalai. Tyrime parametro įgyjamos reikšmės T dar praplėstas iki 650, 850, ar net 1200, 1800 kai stebimas ženklus gerėjimas su mažesnėmis. Parametras Maxiter dar buvo bandomas su 6500 ir 8500 reikšmėmis. Eksperimentų su likusiais uždaviniais pagrindinius rezultatus galite rasti prieduose. Pateikiame surastų rekomenduotinų parametrų reikšmių lentelę nagrinėtiems darbo fabriko uždaviniams:

3.4 Lentelė

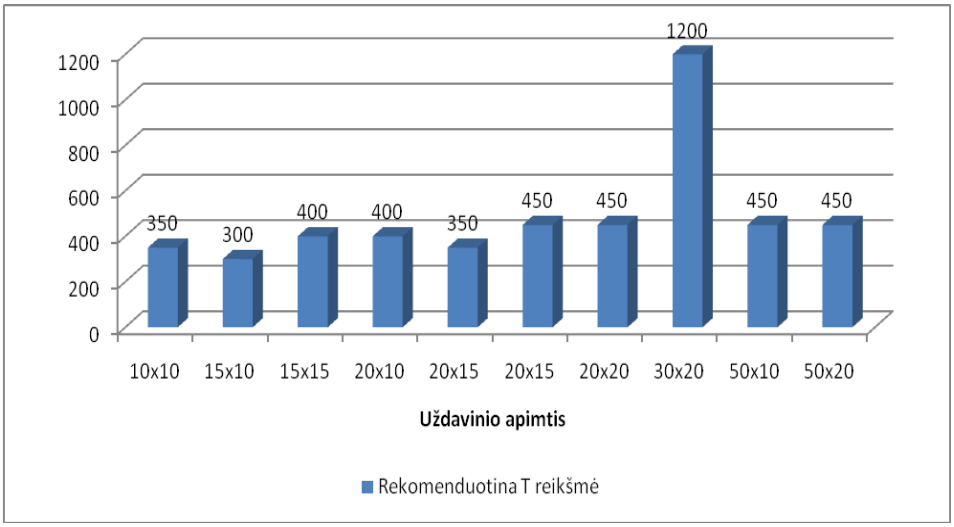
Rastos rekomenduotinos tabu – atkaitinimo modeliavimo paieškos algoritmo ir tabu paieškos parametrų reikšmės

Uždavinys	Apimtis (NxM)	Tabu - atkaitinimo modeliavimo algoritmo parametrų rekomenduotinos reikšmės			Tabu paieškos parametro rekomenduotinos reikšmės
		Maxiter	T	Tabu sąrašo ilgis	Tabu sąrašo ilgis
ft10	10x10	2500	350	7	7
la25	15x10	2500	300	9	5
la40	15x15	3500	400	7	6
la29	20x10	3500	400	9	5
abz9	20x15	2500	350	6	7
abz8	20x15	2500	450	8	8
YN4	20x20	6500	450	9	5
TAI41	30x20	4500	1200	9	5
SWV15	50x10	6500	450	9	10
TAI62	50x20	4500	450	7	8

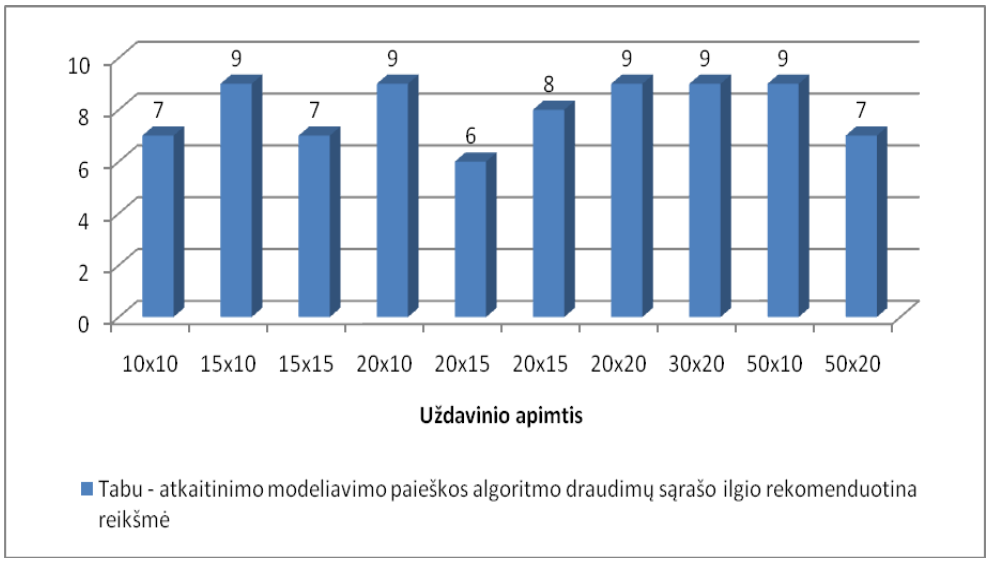
Pateikiame visų trijų tabu – atkaitinimo modeliavimo gautų rekomenduotinų parametrų reikšmių stulpelines diagramas kintant uždavinio apimčiai:



3.11 pav. Rekomenduotinų parametro Maxiter reikšmių stulpelinė diagrama



3.12 pav. Rekomenduotinų parametro T reikšmių stulpelinė diagrama

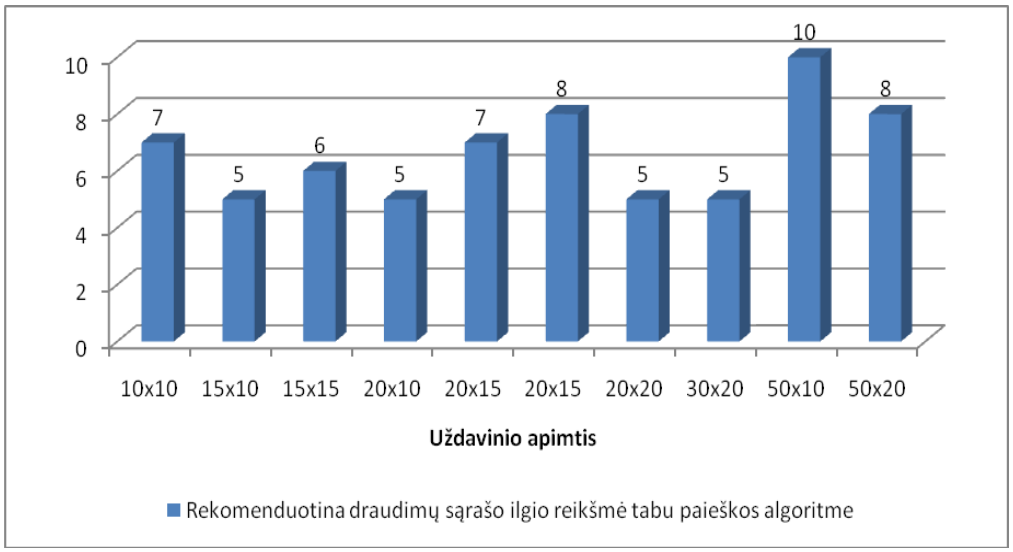


3.13 pav. Rekomenduotinų tabu sąrašo ilgio reikšmių stulpelinė diagrama

Iš pateiktų diagramų nesunku pastebėti, kad didėjant sprendžiamo uždavinio apimčiai rekomenduotinos parametrų Maxiter ir T reikšmės didėja. Be to, didžiausiems keturiems uždaviniams, t.y. turintiems 400 operacijų ir daugiau, rekomenduotinos parametrų Maxiter ir T reikšmės įgyja atitinkamai 4500 - 6500 ir 450 – 1200. Dar 3.12 diagramoje pastebime, kad iš parametro T rekomenduotinų reikšmių aiškiai išsiskiria uždavinio TAI41 atvejis, kuris, kaip bus galima pamatyti algoritmų rezultatų suvestinėse, yra sprendžiamas žymiai sunkiau nei kiti nagrinėjami darbo fabriko uždavinių pavyzdžiai. Iš 3.13 diagramos pastebime, kad didėjant sprendžiamo uždavinio apimčiai, tabu – atkaitinimo modeliavimo paieškos algoritmo

rekomenduotina draudimų sąrašo ilgio reikšmė dėsningai nekinta, tačiau net 5 kartus įgyjama maksimali tirta reikšmė.

Vienintelio tabu paieškos parametro draudimų sąrašo ilgio rekomenduotinų reikšmių stulpelinė diagrama:



3.14 pav. Tabu paieškos algoritmo rekomenduotinų draudimų sąrašo ilgio reikšmių stulpelinė diagrama

Iš 3.14 paveikslo pastebime, kad didėjant sprendžiamo uždavinio apimčiai, tabu paieškos algoritmo rekomenduotina draudimų sąrašo ilgio reikšmė dėsningai nekinta, tačiau kiek išsiskiria du nagrinėti uždaviniai su darbų skaičiumi $N = 50$ bei įrenginių skaičiais $M = 10$ ir $M = 20$, šiuose uždaviniuose didinant tabu sąrašo ilgį algoritmo efektyvumas pakankamai stabiliai gerėja. Kituose uždaviniuose dažnai su keliomis tabu sąrašo ilgio reikšmėmis buvo gaunami tik labai nežymiai besiskiriantys darbų trukmių empiriniai vidurkiai, ir tokio monotoniško rezultatų gerėjimo didinant tabu sąrašo ilgį nepastebėjome. Todėl visai pagrįstai galime teigti, kad optimalus tabu sąrašo ilgis gali būti tiesiogiai proporcingas santykiui N/M, o ne sprendžiamo uždavinio operacijų skaičiui. Tam dar labiau įsitikinti, reiktų paimti daugiau uždavinių su skirtingomis santykio N/M reikšmėmis ir atlikti analogišką optimalaus tabu sąrašo ilgio nustatymą.

Toliau pateikiame nagrinėtų uždavinių sprendimo rezultatus su rastomis rekomenduotinomis parametrų reikšmėmis:

3.5 Lentelė

Uždavinių sprendimo tabu paieškos algoritmais rezultatų suvestinė

Uždavinys	Apimtis (NxM)	ODT	GŽS	Tabu- atkaitinimo modeliavimas				Tabu paieška				Skaičiavimų laikas, sek.
				VDT	Ev, %	MDT	Em, %	VDT	Ev, %	MDT	Em, %	
ft10	10x10	930	930	930,7	0,08	930	0,00	932,8	0,30	930	0,00	100
la25	15x10	977	977	978,8	0,18	977	0,00	978,8	0,18	977	0,00	100
la40	15x15	1222	1222	1230,8	0,72	1224	0,16	1233,3	0,92	1228	0,49	100
la29	20x10	1152	1152	1169,4	1,51	1157	0,43	1170,3	1,59	1166	1,22	300
abz9	20x15	661	679	691,5	4,61	683	3,33	690,9	4,52	689	4,24	300
abz8	20x15	645	665	679,8	5,40	669	3,72	684,6	6,14	679	5,27	300
YN4	20x20	918	968	980	6,75	975	6,21	981,6	6,93	974	6,10	300
TAI41	30x20	1859	2018	2079	11,83	2061	10,87	2061,4	10,89	2049	10,22	900
SWV15	50x10	2885	2904	3040,4	5,39	3016	4,54	2960	2,60	2946	2,11	900
TAI62	50x20	2869	2869	2911,6	1,48	2901	1,12	2888,8	0,69	2877	0,28	900
Vidutinė					3,8%		3,04%		3,48%		2,99%	

čia: ODT – optimalios darbų trukmės įvertis; GŽS – geriausio žinomo sprendinio darbų trukmė; VDT – darbų trukmių empirinis vidurkis; Ev – VDT santykinė paklaida nuo ODT; MDT – eksperimentuose gauta minimali darbų trukmės reikšmė; Em – MDT santykinė paklaida nuo ODT.

Apibendrinant tabu – atkaitinimo modeliavimo ir paprastos tabu paieškos algoritmų efektyvumą galime pasakyti, kad tabu ir atkaitinimo modeliavimo kombinacija (kai parametrai parenkami iš anksčiau apibrėžtų intervalų) kiek geriau sprendžiami mažesnės apimties uždaviniai nei paprastos tabu paieškos algoritmu, t.y. uždaviniuose, turinčiuose 300 operacijų ir mažiau, pastebime geresnius tiek empirinių darbų trukmių vidurkių, tiek minimalių reikšmių rezultatus, bet uždaviniuose, turinčiuose 500 ir daugiau operacijų, jau efektyvesnis paprastas tabu paieškos algoritmas. Dar neaptartam (300,500) operacijų intervalui priklauso vos vienas uždavinys, turintis 400 operacijų. Šiame uždavinyje gautų darbų trukmių empirinis vidurkis kiek mažesnis tabu – atkaitinimo modeliavimo atveju, bet su paprasta tabu paieška gauta mažesnė minimali darbų trukmė, todėl kategoriškai nuspręsti, kad operacijų intervalo (300,500) uždaviniams geriausiai tinka tik tabu – atkaitinimo modeliavimo algoritmas arba tik tabu paieška, negalime. Tad galime teigti, kad uždaviniuose, kurių operacijų skaičius grupuojasi apie intervalo (300,500) vidurį, abiejų tabu paieška paremtų algoritmų efektyvumas bus apylygis. Lygindami šių algoritmų bendrus rezultatus su paieškos kintamose aplinkose algoritmu pastebime, kad visų uždavinių vidutinė darbų trukmių empirinių vidurkių santykinė paklaida tiek tabu paieškos (paklaidos reikšmė 3,48%), tiek tabu ir atkaitinimo modeliavimo (paklaidos reikšmė 3,8%) algoritmais gaunama didesnė už paieškos kintamose aplinkose atitinkamą paklaidą (paklaidos reikšmė 3,42%). Eksperimentuose gautų

minimalių reikšmių vidutinė visų uždavinių santykinė paklaida paieškos kintamose aplinkose algoritmo yra mažiausia, ir jos reikšmė yra 2,91% (3.3 lentelė).

Jei tarsime, kad tabu – atkaitinimo modeliavimo algoritmą naudojame uždaviniuose turinčiuose mažiau nei 400 operacijų, o tabu paieškos algoritmą – didesniuose, tai tabu ir atkaitinimo modeliavimo bei paieškos kintamose aplinkose algoritmais gaunami rezultatai būtų tokie:

3.6 Lentelė

Uždavinių sprendimo tabu paieškos ir paieškos kintamose aplinkose algoritmais rezultatų suvestinė

Uždavinys	Apimtis (NxM)	ODT	GŽS	Tabu – atkaitinimo modeliavimas ir tabu paieška				Paieška kintamose aplinkose				Skaič. laikas, sek.
				VDT	Ev, %	MDT	Em, %	VDT	Ev, %	MDT	Em, %	
ft10	10x10	930	930	930,7	0,08	930	0,00	931,8	0,19	930	0,00	100
la25	15x10	977	977	978,8	0,18	977	0,00	979,3	0,24	977	0,00	100
la40	15x15	1222	1222	1230,8	0,72	1224	0,16	1231,6	0,79	1228	0,49	100
la29	20x10	1152	1152	1169,4	1,51	1157	0,43	1169,7	1,53	1167	1,30	300
abz9	20x15	661	679	691,5	4,61	683	3,33	691,8	4,66	688	4,08	300
abz8	20x15	645	665	679,8	5,40	669	3,72	681,1	5,60	673	4,34	300
YN4	20x20	918	968	981,6	6,93	974	6,10	985,4	7,34	976	6,32	300
TAI41	30x20	1859	2018	2061,4	10,89	2049	10,22	2070,4	11,37	2065	11,08	900
SWV15	50x10	2885	2904	2960	2,60	2946	2,11	2937	1,80	2917	1,11	900
TAI62	50x20	2869	2869	2888,8	0,69	2877	0,28	2887,8	0,66	2881	0,42	900
Vidutinė					3,36%		2,64%		3,42%		2,91%	

čia: ODT – optimalios darbų trukmės įvertis; GŽS – geriausio žinomo sprendinio darbų trukmė; VDT – darbų trukmių empirinis vidurkis; Ev – VDT santykinė paklaida nuo ODT; MDT – eksperimentuose gauta minimali darbų trukmės reikšmė; Em – MDT santykinė paklaida nuo ODT. Tikslus eksperimentų skaičius, atliktas uždaviniams: FT10, LA25, LA40, ABZ8, YN4 uždaviniams po 10, LA29, ABZ9 uždaviniams po 20, TAI41, SVW15, TAI62 uždaviniams po 5.

Iš lentelėje pateiktų rezultatų pastebime, kad tabu – atkaitinimo paieškos algoritmui paskyrus uždavinius, turinčius mažiau nei 400 operacijų, o tabu paieškos algoritmui paskyrus uždavinius, turinčius 400 ir daugiau operacijų, gaunami ir empirinių darbų trukmių vidurkių santykinių paklaidų vidurkis 3,36%, ir minimalių reikšmių santykinių paklaidų vidurkis 2,64% yra mažesni už paieškos kintamose aplinkose algoritmu gautas atitinkamas paklaidas. Tiesa, empirinių darbų trukmių vidurkių santykinių paklaidų vidurkiai skiriasi nežymiai 3,36% ir 3,42%, bet minimalių reikšmių santykinių paklaidų vidurkiai skiriasi ženkliau. Bet vienareikšmiškai teigti, kad abu tabu paieškos algoritmai yra visada geresnis pasirinkimas, negalime, nes SVW15 uždavinio atveju tiek vidutiniai, tiek minimalūs rezultatai yra pastebimai geresni. 3.6 lentelėje pastebime, kad šešiuose uždaviniuose,

t.y. FT10, LA25, LA40, LA29, ABZ9, TAI62, abejais algoritmais gautų darbų trukmių empiriniai vidurkiai skiriasi labai nežymiai (jų santykinių paklaidų skirtumas mažesnis ar labai artimas 0,1%). Likusiuose 3 – iuose uždaviniuose tabu paieškos algoritmais gaunami empiriniai darbų trukmių vidurkiai jau ženkliau mažesni. Nagrinėjant eksperimentuose gautas minimalias reikšmes, pastebimas ryškesnis tabu paieškos ir tabu – atkaitinimo modeliavimo paieškos algoritmo pranašumas, šiais algoritmais buvo gauti geresni rezultatai septyniuose uždaviniuose, kai paieškos kintamose aplinkose algoritmu gauta minimali reikšmė buvo mažesnė tik minėtame SVW15 uždavinyje. Minimalios reikšmės sutapo lengviausiai sprendžiamuose uždaviniuose.

3.4 SKAIČIAVIMŲ LAIKO PARINKIMO ĮTAKA SPRENDINIO KOKYBEI

Šiame skyrelyje atsakysime į klausimą: ar padidinus kompiuterio skaičiavimų laiką, galėtume gauti žymiai geresnius sprendinius? O gal toliau didindami skaičiavimų laiką, visus uždavinius išspręstume optimaliai? Kaip galime pastebėti 3.6 lentelėje algoritmų rezultatų palyginime, ankstesniuose eksperimentuose vienos realizacijos skaičiavimų laikas kito nuo 100 sekundžių mažiausiuose uždaviniuose iki 900 sekundžių didžiausiuose. Norėdami turėti daugiau duomenų apie algoritmais gaunamų sprendinių empirinių darbų trukmių artėjimą prie optimalios darbų trukmės įverčio, skaičiavimų laiką ir eksperimentų skaičių visiems uždaviniams nustatome šiuos:

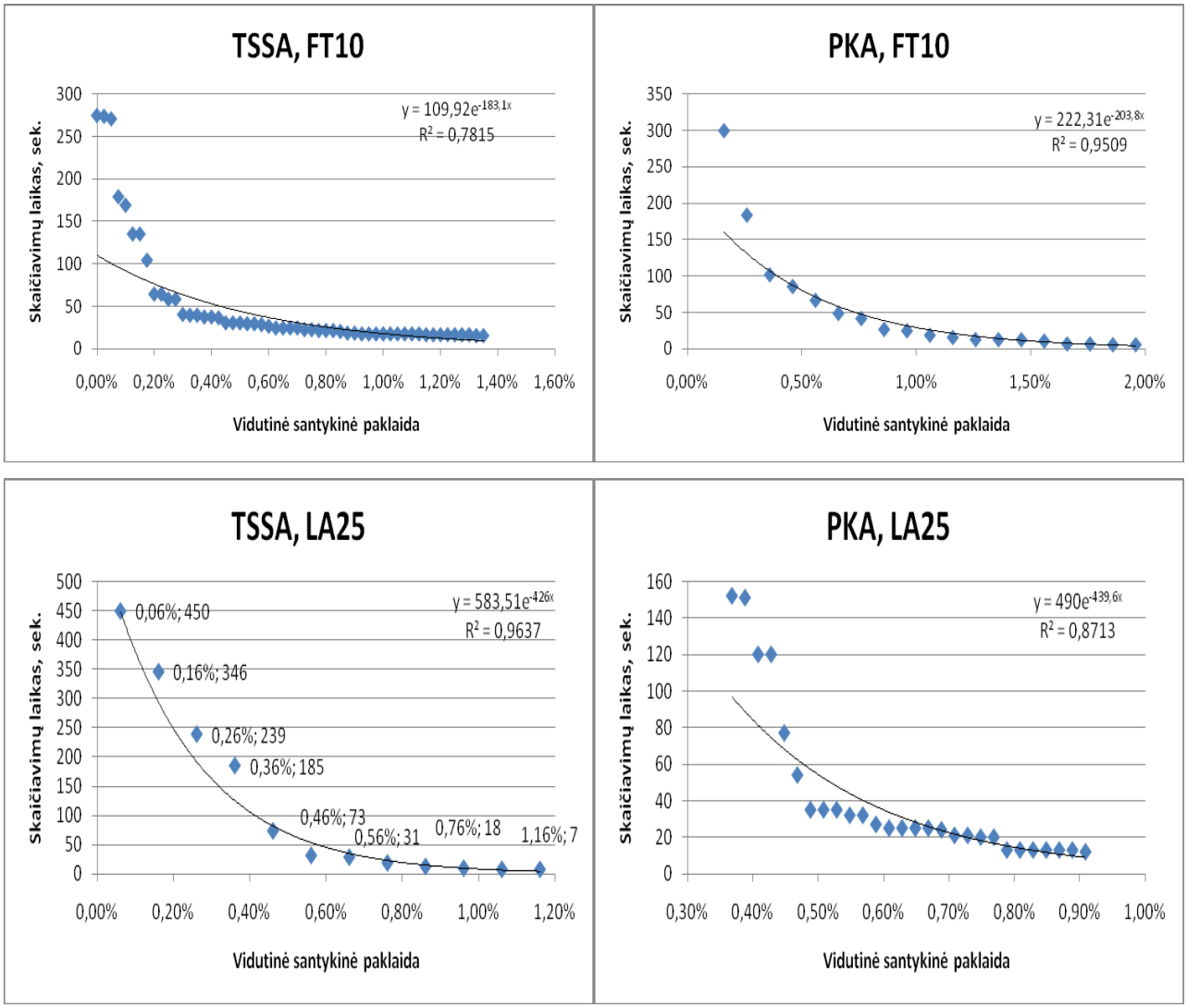
3.7 Lentelė

Uždavinių sprendimo laikų lentelė

Uždavinys	Apimtis (NxM)	Optimalios darbų trukmės įvertis	Geriausio žinomo sprendinio darbų trukmė	Testuotas skaičiavimų laikas, sek.	Eksperimentų skaičius
ft10	10x10	930	930	300	10
la25	15x10	977	977	450	10
la40	15x15	1222	1222	600-900	10
la29	20x10	1152	1152	900-1200	10
abz9	20x15	661	679	900	10
abz8	20x15	645	665	900	10
YN4	20x20	918	968	900-1200	10
TAI41	30x20	1859	2018	1800	5
SWV15	50x10	2885	2904	1800	5
TAI62	50x20	2869	2869	1800	5

Eksperimentai atliekami su gautomis rekomenduotinomis algoritmų parametrų reikšmėmis, kurias galite rasti 3.2 ir 3.4 lentelėse. Kiekvieno eksperimento metu kas sekundę saugoma iki šiol geriausio rasto sprendinio darbų trukmė, tad atlikus visus eksperimentus galime rasti kiekvienu algoritmu gaunamų geriausių sprendinių darbų trukmių empirinį vidurkį bet kuriuo skaičiavimo laiko momentu. Gautą empirinio vidurkio reikšmę palyginame su optimalios darbų trukmės įverčiu, apskaičiuodami empirinio vidurkio santykinę paklaidą nuo minėto įverčio. Tokiu būdu iš eksperimentų rezultatų galime sužinoti apie vidutinę santykinę paklaidą bet kuriuo skaičiavimo laiko momentu. Pateikiame dviejų mažiausių uždavinių skaičiavimų laiko taškines diagramas kintant

vidutinei santykinei paklaidai, tabu – atkaitinimo modeliavimo ir paieškos kintamose aplinkose algoritmų atveju:

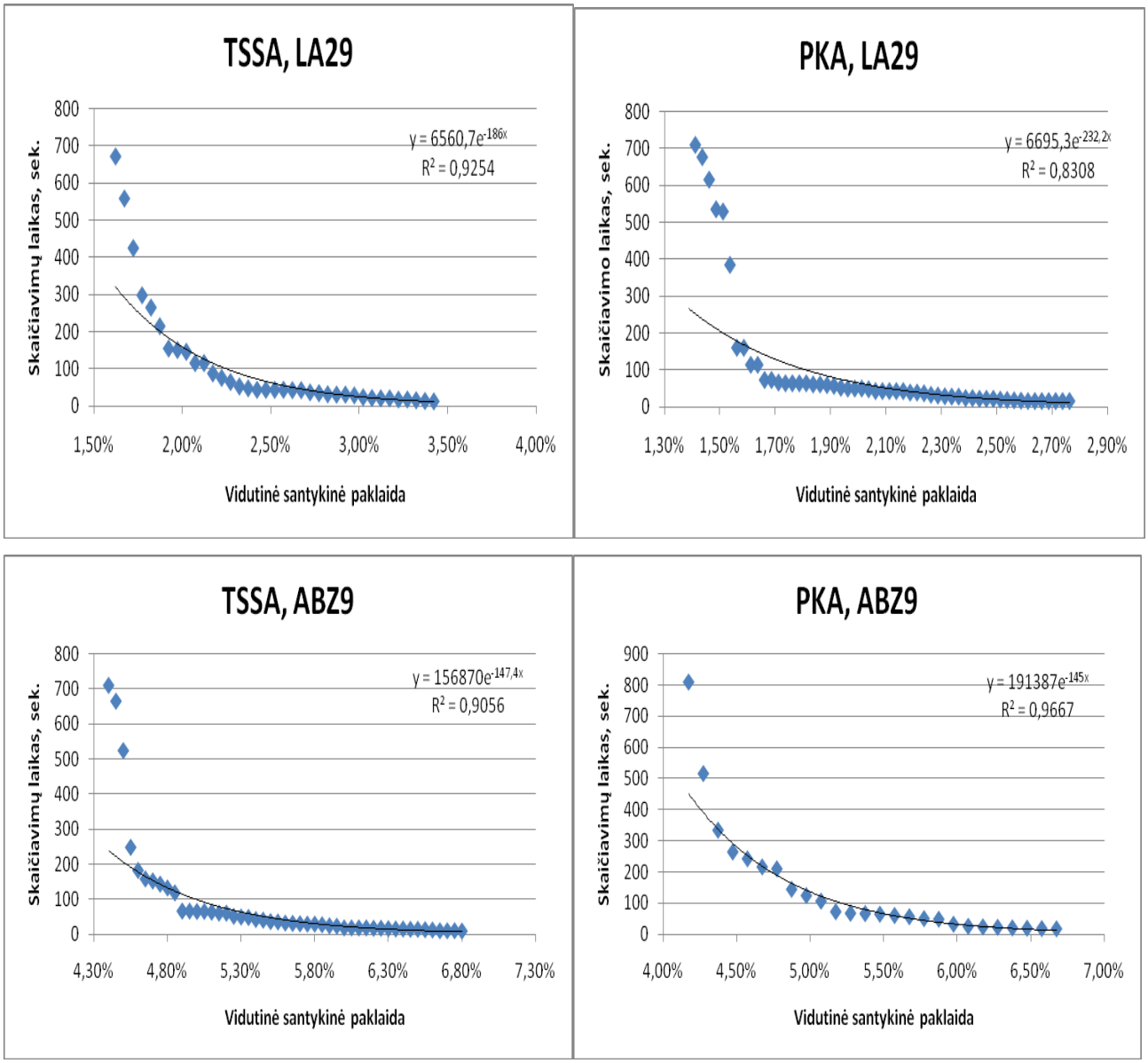


3.15 pav. Uždavinių FT10 ir LA25 skaičiavimų laiko taškinės diagramos kintant santykinei paklaidai

čia: TSSA – tabu ir atkaitinimo modeliavimo paieškos algoritmo rezultatai, PKA – paieškos kintamose aplinkose algoritmo rezultatai, vėliau TS – tabu paieška.

Iš pateiktų grafikų pastebime, kad tik sprendžiant uždavinį LA25 tabu – atkaitinimo modeliavimo algoritmu, skaičiavimų laikas didėja eksponentiškai mažėjant santykinei paklaidai. Visais kitais atvejais skaičiavimų laikas didėja greičiau nei eksponentė, mažėjant santykinei paklaidai. Taip pat galime pastebėti, kad naudojant tabu – atkaitinimo modeliavimo paieškos algoritmą šiem uždaviniams, eksperimentuose gautų darbų trukmių empirinis vidurkis greičiau artėjo prie optimalios reikšmės, ir nustačius didesnį skaičiavimų laiką galima tikėtis optimalaus sprendinio radimo.

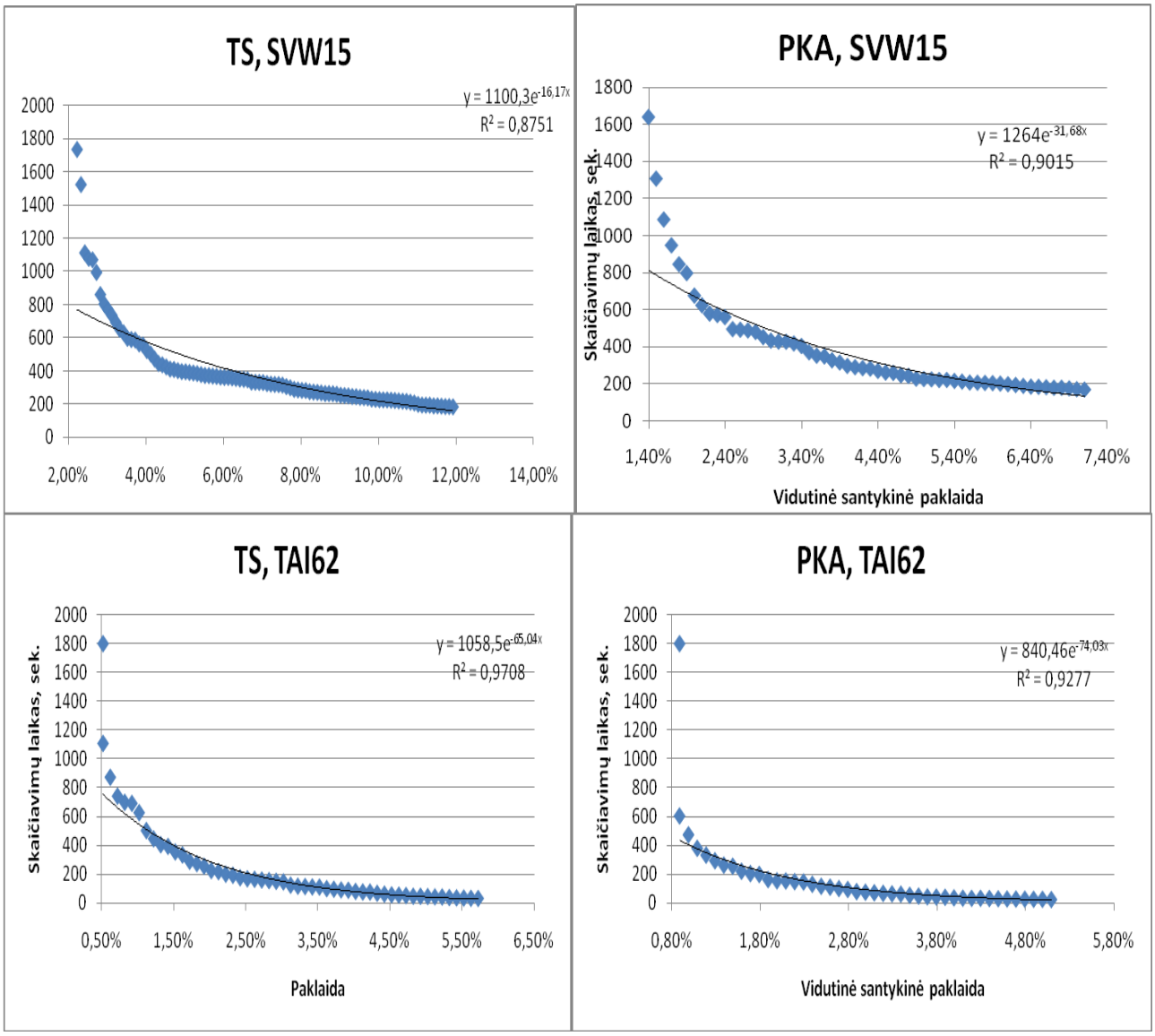
Pateikiame dviejų didesnių uždavinių LA29 ir ABZ9 analogiškas taškines diagramas:



3.16 pav. Uždavinių LA29 ir ABZ9 skaičiavimų laiko taškinės diagramos kintant santykinei paklaidai

Iš pateiktų grafikų pastebime, kad abiem algoritmais sprendžiant tiek LA29, tiek ABZ9, skaičiavimų laikas, pasiekus tam tikras santykines paklaidas, pradeda didėti greičiau nei eksponentė, mažėjant santykinei paklaidai. Tad apytiksliai galima prognozuoti, kad norint vidutinę sprendinio darbų trukmę pagerinti 0,1% - 0,2% dalimis, skaičiavimų laiką tektų didinti mažiausiai keletą kartų, ir pasiekti optimalų sprendinį per priimtiną skaičiavimų laiką sunkiai tikėtina.

Pateikiame dviejų didžiausių uždavinių SVW15 ir TAI62 analogiškas taškines diagramas:



3.17 pav. Uždavinių SVW15 ir TAI62 skaičiavimų laiko taškinės diagramos kintant santykinei paklaidai

Ir vėl nesunku pastebėti panašią į ankstesnes diagramų taškų išsidėstymo formą. Tik skirtumas tas, kad dėl didesnės uždavinių apimties, greitesnis už eksponentinę funkciją skaičiavimo laiko augimas prasideda vėliau nei mažesnės apimties uždaviniams.

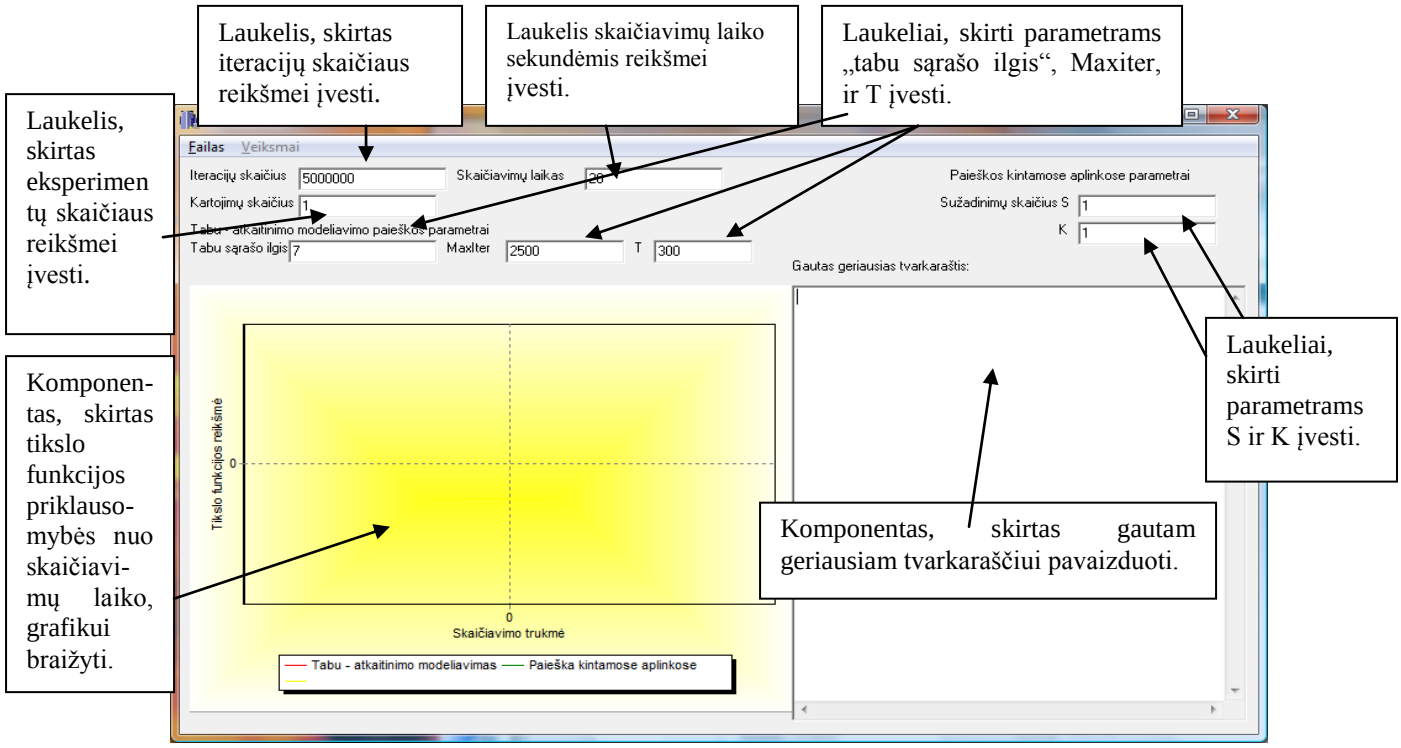
Visų kitų nagrinėtų darbo fabriko uždavinių tokias taškines diagramas galite rasti trečiame priede.

Apibendrinant visų uždavinių sprendimo efektyvumą tabu paieškos ir paieškos kintamose aplinkose algoritmais galima paminėti, kad bene visais atvejais (viena išimtis 3.15 paveiksle) abiejų algoritmų vykdymo laikas, fiksuotu dydžiu mažinant paklaidą, pradeda didėti greičiau nei eksponentinė funkcija, esant tam tikrai vidutinei santykinės paklaidos reikšmei, kuri priklauso nuo konkretaus uždavinio, ir svyruoja tarp 0 - 1% lengviau sprendžiamuose uždaviniuose, o tarp 3 – 6% sunkiau sprendžiamuose, ir net 10% siekia sunkiausiai sprendžiamame TAI41. Todėl atsižvelgiant į

itin mažą prognozuotiną vidutinės santykinės paklaidos mažėjimą, dar parenkant didesnį skaičiavimų laiką nei 3.7 lentelėje, galime teigti, kad norint gauti kuo geresnį sprendinį, gan logiškas sprendimas būtų skaičiavimų laiko daug kartų nedidinti, bet geriau su mažesniu vieno eksperimento skaičiavimų laiku atlikti daugiau eksperimentų ir išrinkti geriausią rezultatą.

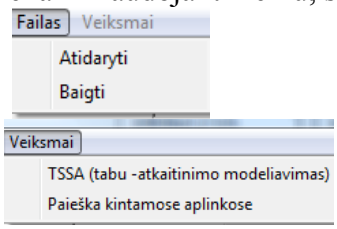
VARTOTOJO SĄSAJA

Šiame skyriuje aprašysime sukurtas darbo fabriko uždavinio tvarkaraščių optimizavimo priemones bei jų taikymą. Kompiuterinė programa parašyta C++ builder aplinkoje. Pradiniai duomenys t.y. kiekvieno darbo užduočių tvarka ir būtinos toms užduotims atlikti laiko trukmės bus nuskaitomos iš tekstinės duomenų bylos. Įvedus pradinius duomenis iš pasirinktos duomenų bylos, bus galima vienu iš dviejų algoritmų optimizuoti tvarkaraštį. Po optimizavimo procedūros programa atspausdins geriausią gautą tvarkaraštį, taip pat nubraižys optimizavimo algoritmo efektyvumą nusakantį grafiką, kuriame bus atidėtos geriausio tvarkaraščio tikslo funkcijos reikšmės kas vieną skaičiavimų sekundę. Programos langas atrodo taip:



3.18 pav. Programos langas

Visi vartotojo veiksmai atliekami naudojant meniu, šios programos meniu struktūra:



3.19 pav. Komandų „Failas“ ir „Veiksmai“ skleistinės

Paspaudus punktą „Atidaryti“, leidžiama pasirinkti norimą atidaryti tekstinių duomenų bylą. Duomenys, pavyzdžiui 6×6 darbo fabriko tvarkaraščių optimizavimo uždavinio, tekstinėje byloje turėtų būti tokio formato:

```
M1 M2 M3 M4 M5 M6 (išvardinami įrenginiai)
J1 3 1 1 3 2 6 4 7 6 3 5 6(vieno darbo įrenginių tvarka ir trukmės)
J2 2 8 3 5 5 10 6 10 1 10 4 4
J3 3 5 4 4 6 8 1 9 2 1 5 7
J4 2 5 1 5 3 5 4 3 5 8 6 9
J5 3 9 2 3 5 5 6 4 1 3 4 1
J6 2 3 4 3 6 9 1 10 5 4 3 1
```

Duomenų byloje šiuo atveju turi būti septynios eilutės. Pirmoje eilutėje turi būti išvardintos užduotys (įrenginiai). Vienas nuo kito įrenginių pavadinimai skiriami bent vienu tarpu. Jiems programa priskiria po skaičių: įrenginiui „M1“ šiuo atveju būtų priskirtas skaičius 1, įrenginiui „M2“ būtų priskirtas skaičius 2... Tolesnėse eilutėse išvardijamas norimų atlikti darbų sąrašas. Vienam darbui skiriama viena duomenų bylos eilutė. Pirmoje eilutės vietoje privalo būti darbo pavadinimas, o toliau toje pačioje eilutėje nusakoma to darbo įrenginių aplankymo tvarka ir trukmės. Pavyzdžiui, antroji mūsų nagrinėjamos duomenų bylos eilutė reiškia, kad darbui, kurio pavadinimas „J1“ iš pradžių reikės vienam laiko vienetui priskirti įrenginį, kuriam priskirtas skaičius 3 t.y. įrenginį „M3“, po to reikės trims laiko vienetams priskirti įrenginį, kurio numeris yra lygus 1 t.y. įrenginį „M1“. Likusi eilutės dalis bei likę darbai interpretuojami analogiškai.

Meniu punktas „Baigti“ uždaro programos langą.

Įvedus duomenis iš pasirinktos tekstinės bylos, galime naudotis komandos „Veiksmai“ punktais: „TSSA (tabu – atkaitinimo modeliavimas)“, „Paieška kintamose aplinkose“. Pasirinkus bet kurį iš šių punktų, norimam uždaviniui spręsti pritaikomas atitinkamas optimizavimo algoritmas. Taikant bet kurį optimizavimo algoritmą, programa į tekstinę bylą „reikšmės.txt“ kas vieną skaičiavimų sekundę spausdina geriausio iki tol gauto tvarkaraščio tikslo funkcijos reikšmę. Pagal šiuos duomenis nubraižomas tikslo funkcijos reikšmės priklausomybės nuo skaičiavimo laiko grafikas. Atlikus visus numatytus skaičiavimus tikslo funkcijos prasme geriausias tvarkaraštis atspausdinamas į tekstinę bylą „tvarkaraštis.txt“ ir pavaizduojamas pagrindiniame programos lange. Galutinis tvarkaraštis į rezultatų bylą spausdinamas eilutėmis ir kiekviena eilutė turi tokį formatą:
Darbas, užduotis, vykdymo pradžia, vykdymo pabaiga.

Jei minėtų tekstinių bylų, į kurias spausdinami programos vykdymo metu gaunami rezultatai, programos vykdomosios bylos kataloge nėra, tai jos vykdomosios bylos kataloge sukuriamos naujai. Programos būsimų skaičiavimų trukmė nusakoma dviem parametrais: iteracijų skaičiumi ir bendru skaičiavimų laiku. Programa vykdo tam tikro optimizavimo algoritmo skaičiavimus tol, kol pasiekiamas maksimalus iteracijų skaičius arba nurodyta maksimali skaičiavimų trukmė

sekundėmis. Programos tekstą galima rasti priedų kompaktiniame diske adresu „D:\Programa\Darbo fabrikas\Unit.cpp“.

IŠVADOS

1. Paieškos kintamose aplinkose algoritmo parametro S rekomenduotina reikšmė mažesnės apimties uždaviniams yra didesnė nei didelės apimties problemoms, t.y. uždaviniuose, turinčiuose 225 operacijų ir mažiau, rekomenduotina S reikšmė kito 2 – 4 reikšmių ribose, o uždaviniuose, turinčiuose daugiau nei 225 operacijų, rekomenduotina S reikšmė įgijo 1 – 2 reikšmes.
2. Paieškos kintamose aplinkose algoritmo parametro K rekomenduotina reikšmė mažesnės apimties uždaviniams taip pat yra didesnė nei didelės apimties problemoms, t.y. uždaviniuose, turinčiuose 400 operacijų ir mažiau, rekomenduotina K reikšmė kito 2 – 3 reikšmių ribose, o uždaviniuose, turinčiuose daugiau nei 400 operacijų, rekomenduotina K reikšmė įgijo reikšmę 1.
3. Didėjant sprendžiamo uždavinio apimčiai, parametro Maxiter optimali reikšmė didėja. Uždaviniuose, turinčiuose 300 operacijų ir mažiau, rekomenduotina parametro reikšmė Maxiter kito 2500 – 3500 ribose, o uždaviniuose, turinčiuose 500 ir daugiau operacijų, geriausi rezultatai gaunami su kiek norima (pvz. $5 \cdot 10^6$) didele Maxiter reikšme, ir taip iš tabu – atkaitinimo modeliavimo algoritmų kombinacijos pašalinant atkaitinimo modeliavimą. Uždaviniuose, turinčiuose daugiau nei 300 ir mažiau nei 500 operacijų, gaunami apylygiai rezultatai ir su kiek norima didele parametro Maxiter reikšme, ir su 4500 – 6500 šio parametro reikšmėmis.
4. Didėjant sprendžiamo uždavinio apimčiai, parametro T rekomenduotina reikšmė didėja. Uždaviniuose, turinčiuose 150 operacijų ir mažiau, rekomenduotina T reikšmė įgijo 300 – 350 reikšmes. Uždaviniuose, turinčiuose daugiau nei 150 ir mažiau nei 400 operacijų, rekomenduotina T reikšmė kito 350 – 450 ribose.
5. Didėjant sprendžiamo uždavinio apimčiai, fiksuoto ilgio tabu sąrašo optimalus ilgis neturi nei mažėjimo, nei didėjimo tendencijų. Daugumoje uždavinių optimalus tabu sąrašo ilgis kito 6 – 9 reikšmių ribose.
6. Daugumoje uždavinių vidutinės abiem algoritmais gaunamų sprendinių darbų trukmės yra apylygės, bet bendra minimalių eksperimentuose stebėtų tabu paieškos (tabu – atkaitinimo modeliavimo ir paprastos tabu paieškos) algoritmais gaunamų darbų trukmių paklaida (2,64%) yra pastebimai mažesnė už paieškos kintamose aplinkose algoritmo atitinkamą paklaidą (2,91%).

REKOMENDACIJOS

Skyrelyje 3.4 išryškėjo pakankamai rimtas abiem realizuotais algoritmais gaunamų sprendinių darbų trukmių artėjimo prie optimalios reikšmės įverčio trūkumas. Norint pagreitinti realizuotais algoritmais gaunamų sprendinių artėjimą prie optimalios reikšmės įverčio, dar galima ištirti kitokias algoritmų modifikacijas, pavyzdžiui, tiek tabu paieškos, tiek tabu – atkaitinimo modeliavimo algoritmuose dar galėtume tirti dinaminio tabu sąrašo ilgio parinkimo poveikį gaunamų sprendinių kokybei, nes šiame darbe buvo tiriamas tik fiksuoto ilgio draudimų sąrašas. Taip pat tabu – atkaitinimo modeliavimo paieškos algoritme dar galima būtų ištirti platesnius parametrų parinkimo intervalus. Taip pat galima ištirti dinaminės T reikšmės parinkimo įtaką gaunamų sprendinių kokybei, pavyzdžiui, nerandant tam tikrą iteracijų skaičių geresnio sprendinio tabu – atkaitinimo modeliavimo paieškos algoritmu, didinti parametro T reikšmę, kad blogesnių sprendinių patvirtinimo tikimybė didėtų ir galbūt taip pavyktų ištrūkti iš lokalaus minimumo. Šiame darbe realizuotame tabu – atkaitinimo modeliavimo paieškos algoritme, T laikyta parametru ir algoritmo vykdymo metu yra pastovi. Paieškos kintamose aplinkose algoritme dar galima ištirti sužadinimo procedūroje parenkamų sprendinių kokybės įtaką galutiniam geriausiam algoritmo rezultatui, galbūt sužadinimo procedūrai uždraudus patvirtinti blogiausius aplinkos sprendinius (ar kažkokį blogiausią aplinkos sprendinių darbų trukmių intervalą), gautume geresnės kokybės sprendinius ir artėjimą? Šiame darbe sužadinimo procedūroje atsitiktinai patvirtinamas apskritai bet koks aplinkos sprendinys. Taip pat abiejų algoritmų atveju galima būtų naudoti kitokias sprendinių aplinkas.

PADĖKA

Nuoširdžiai dėkoju savo magistro baigiamojo darbo vadovui doc. dr. Narimantui Listopadskiui už pagalbą ir patarimus.

LITERATŪRA

1. Matthew Bartschi Wall, A Genetic Algorithm for Resource-Constrained Scheduling, 1996.
2. Bibo Yang, Joseph Geunes, William J. O'Brien, Resource-Constrained Project Scheduling: Past Work and New Directions, 2001.
3. Miloš Šeda, Mathematical Models of Flow Shop and Job Shop Scheduling Problems, 2007, p.122-127.
4. Christian Blum, Metaheuristics for Group Shop Scheduling, 2002.
5. Mehmet Sevkli, Variable Neighbourhood Search for Job Shop Scheduling Problems, 2006, p. 34-39.
6. ANANT SINGH JAIN, BALASUBRAMANIAN RANGASWAMY, SHEIK MEERAN, New and "Stronger" Job-Shop Neighbourhoods: A Focus on the Method of Nowicki and Smutnicki, Journal of Heuristics, 6: 457–480 (2000).
7. ChaoYong Zhang, PeiGen Li,YunQing Rao, ZaiLin Guan, A very fast TS/SA algorithm for the job shop scheduling problem, Computers & Operations Research 35 (2008) 282 – 294.
8. Darbo fabriko uždavinių pavyzdžiai, http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html
9. Darbo fabriko uždavinių pavyzdžiai, http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/jobshop1.txt
10. R. J. M. Vaessens, E. H. L. Aarts, J.K. Lenstra, Job Shop Scheduling by Locar Search.
11. Misevičius, A., Blonskis, J., Bukšnaitis, V. Kombinatorinis optimizavimas ir metaeuristiniai metodai: teoriniai aspektai / Informacijos mokslai, 2007, t. 42-43, ISSN 1392-0561, p. 213-219.
12. Listopadskis, N., Kairaitis, G., Tvarkaraščių sudarymo uždavinių ir jų algoritmų tyrimas / Matematika ir matematinis modeliavimas / Kauno technologijos universitetas, 2010.
13. Listopadskis, N., Kairaitis, G., Tvarkaraščių sudarymo uždavinių ir jų algoritmų tyrimas / VIII studentų konferencija / Kauno technologijos universitetas, 2010.
14. Kairaitis, G., Tvarkaraščių sudarymo uždavinių ir jų algoritmų analizė. Bakalauro darbas. Kauno technologijos universitetas, 2008.

1. PRIEDAS: TERMINŲ ŽODYNAS

Tabu search – tabu paieška

Tabu list – tabu sąrašas

Variable neighbourhood search – paieška kintamose aplinkose

Makespan – tvarkaraščio darbų trukmė

Job shop scheduling problem – darbo fabriko tvarkaraščių uždavinys

Flow shop scheduling problem – srauto fabriko tvarkaraščių uždavinys

Open shop scheduling problem – atviro fabriko tvarkaraščių uždavinys

Local search – lokalioji paieška

Iterated local search – iteratyvioji lokalioji paieška

Greedy randomized adaptive search procedures (GRASP) – godžiosios randomizuotos adaptyvios paieškos procedūros

Simulated annealing – atkaitinimo modeliavimas

Ant colony optimization – skruzdėlių kolonijų elgsenos imitavimo algoritmai bičių kolonijų

Bee colony optimization – elgsenos imitavimo algoritmai

Critical path – ilgiausia tvarkaraščio operacijų seka

Lower bound – optimalios tikslo funkcijos reikšmės įvertis

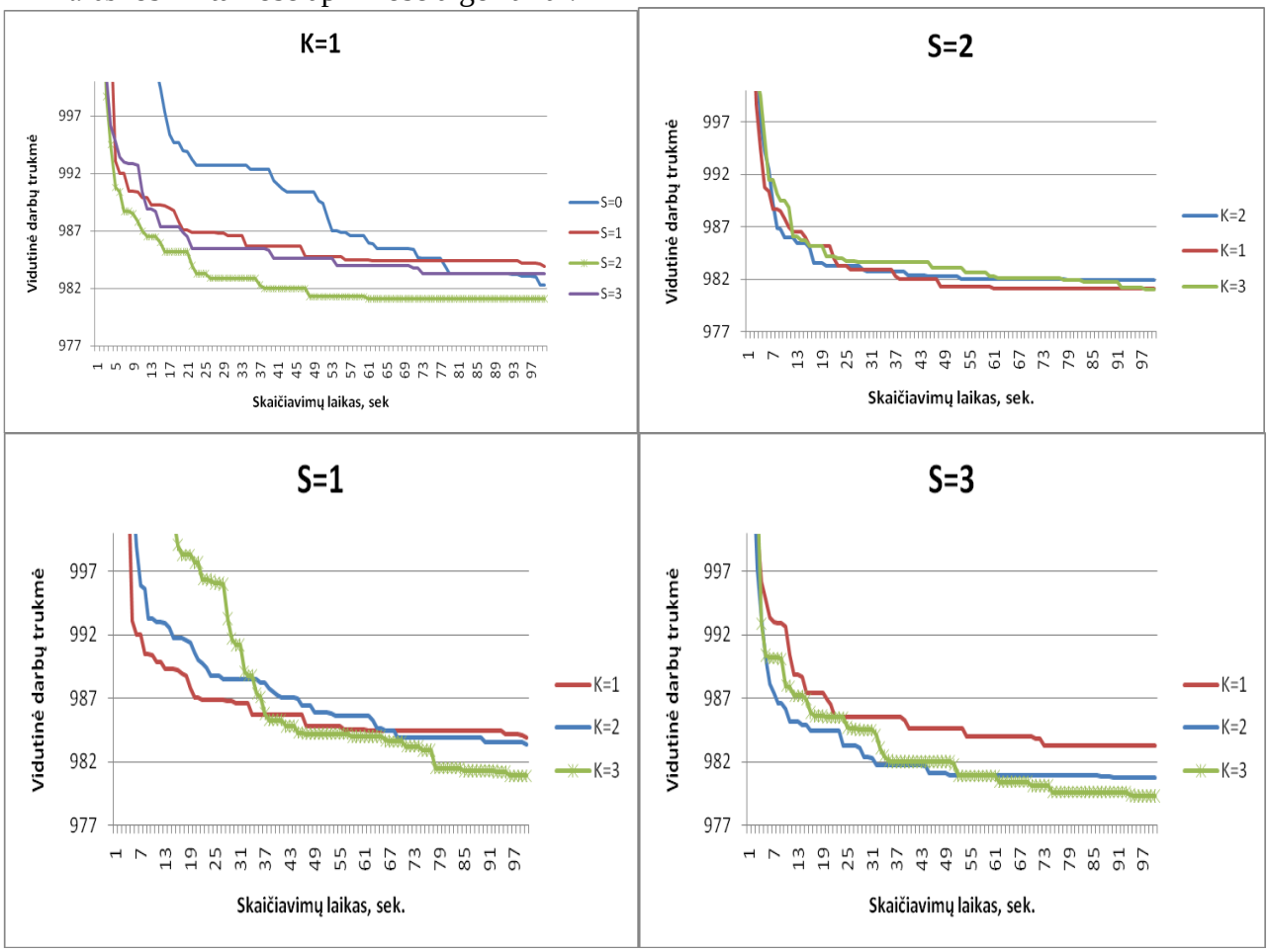
Upper bound – geriausio žinomo sprendinio tikslo funkcijos reikšmė

Shake procedure – sužadinimo procedūra

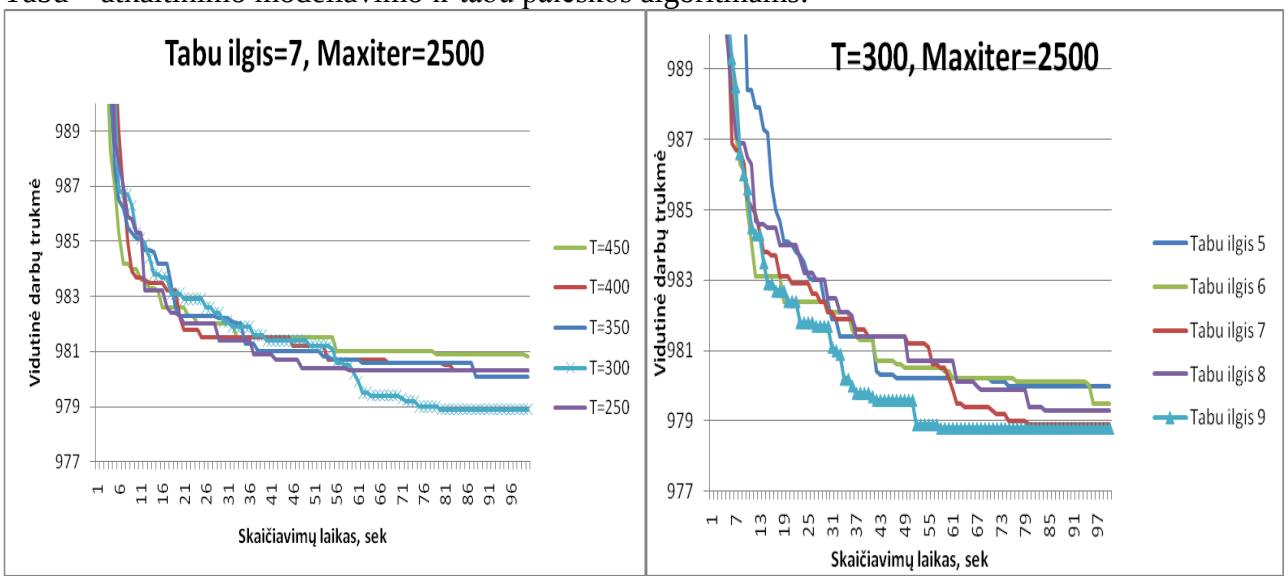
2. PRIEDAS: VISŲ SPRĘSTŲ UŽDAVINIŲ TYRIMO REZULTATAI

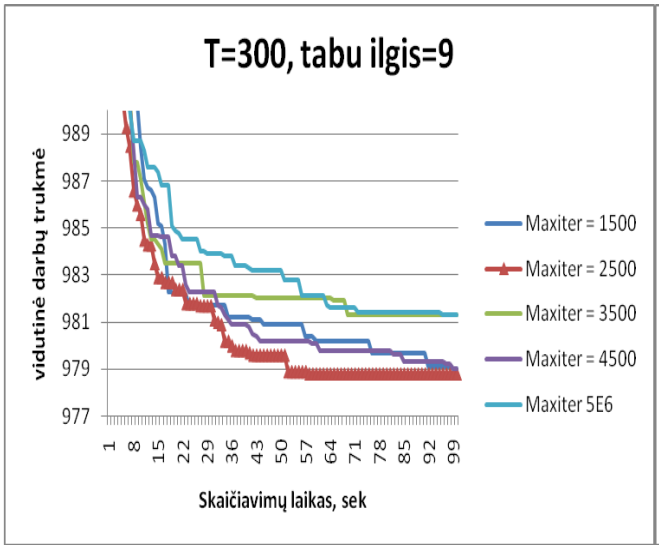
1. Uždavinys LA25 (15×10).

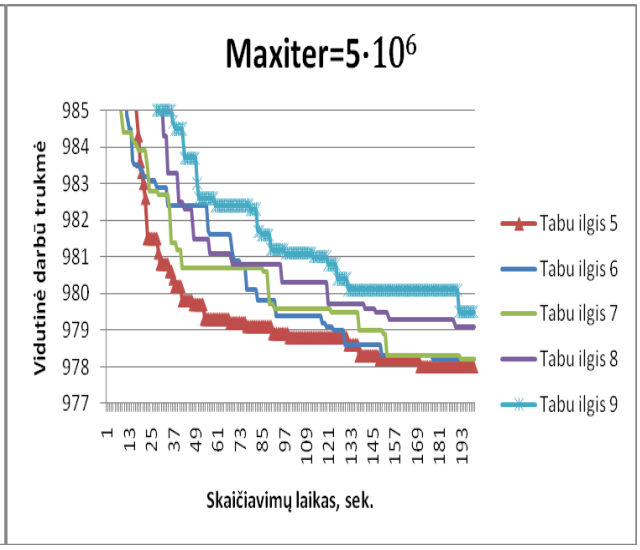
Paieškos kintamose aplinkose algoritmui:



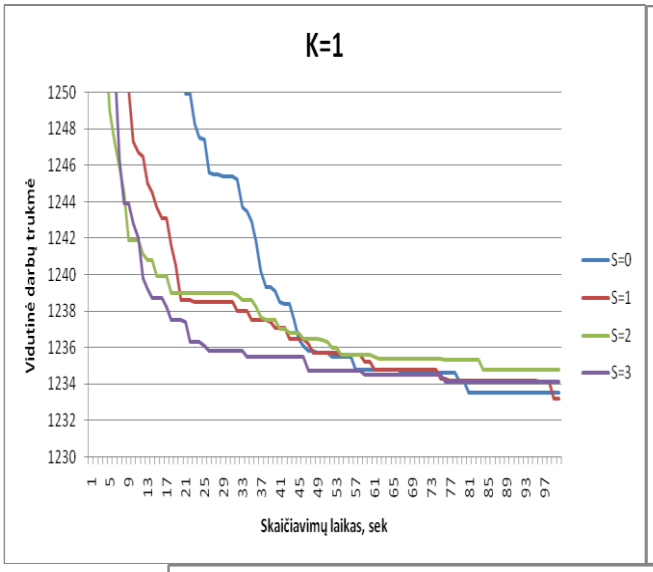
Tabu – atkaitinimo modeliavimo ir tabu paieškos algoritmams:

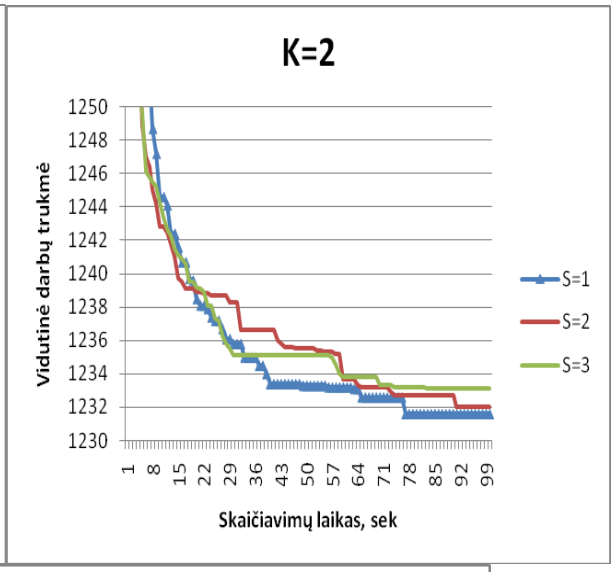


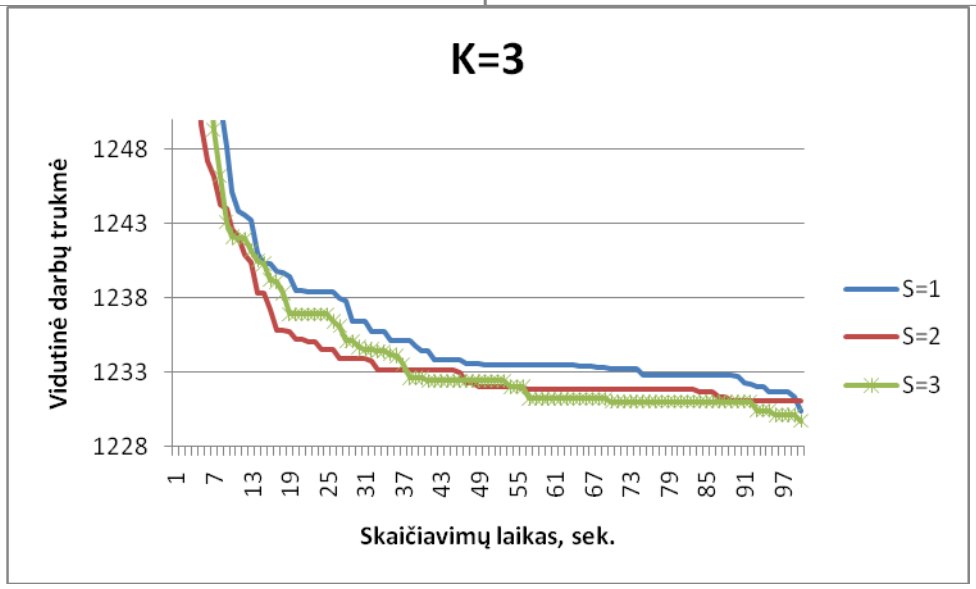




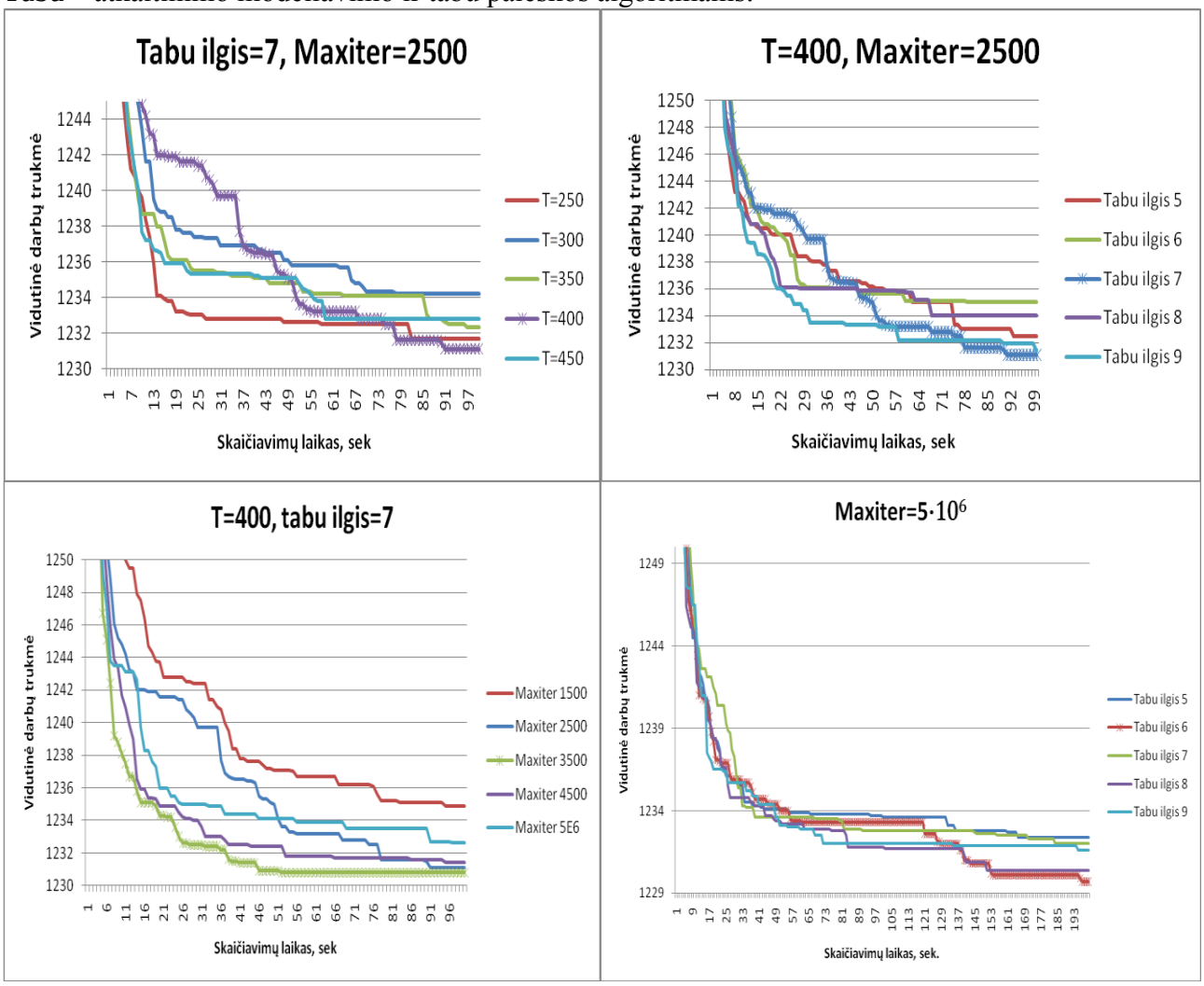
2. Uždavinys LA40 (15×15).
Paieškos kintamose aplinkose algoritmui





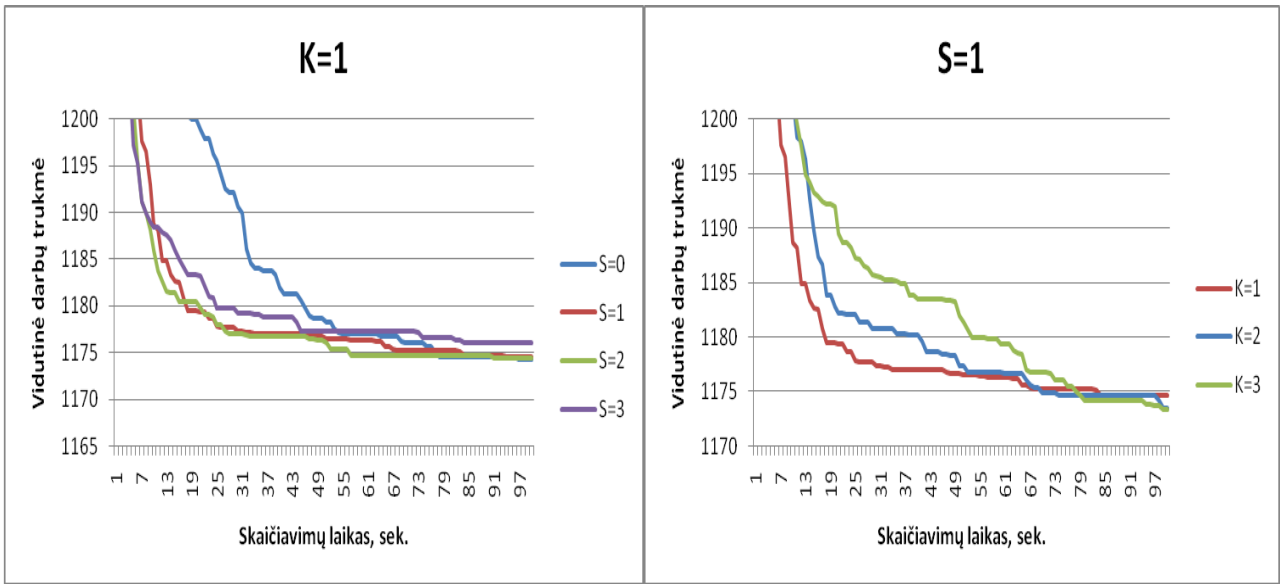


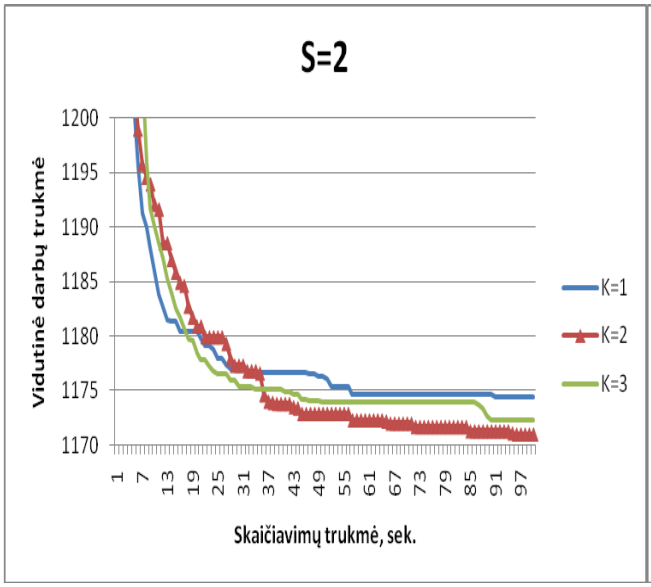
Tabu – atkaitinimo modeliavimo ir tabu paieškos algoritmams:

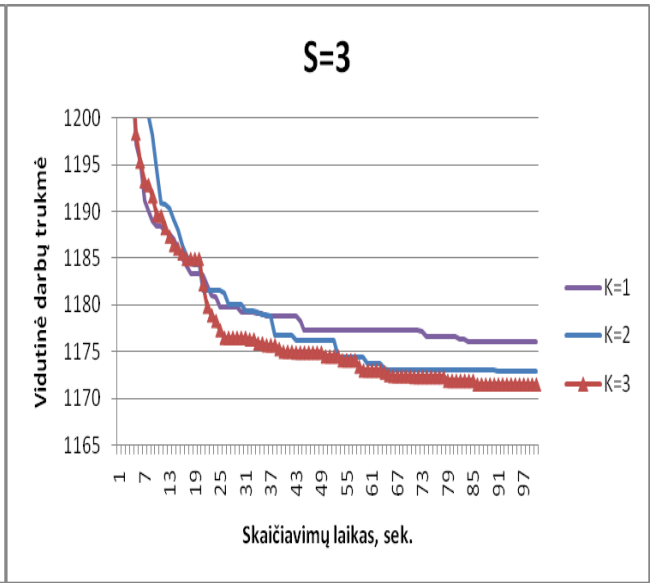


3. Uždavinys LA29 (20×10).

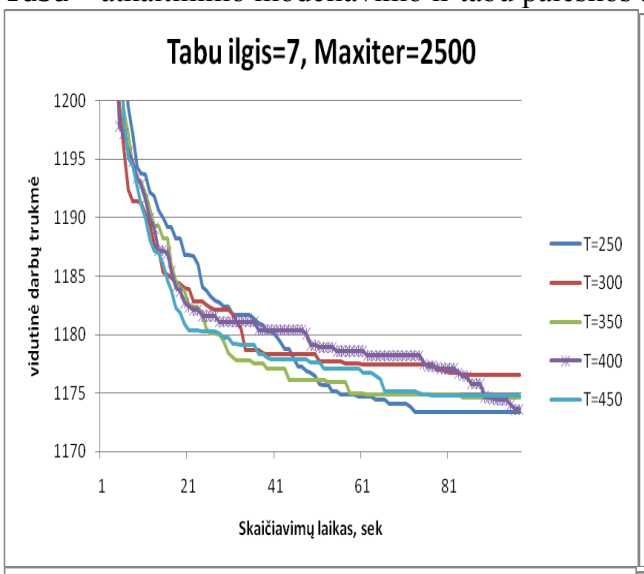
Paieškos kintamose aplinkose algoritmui:

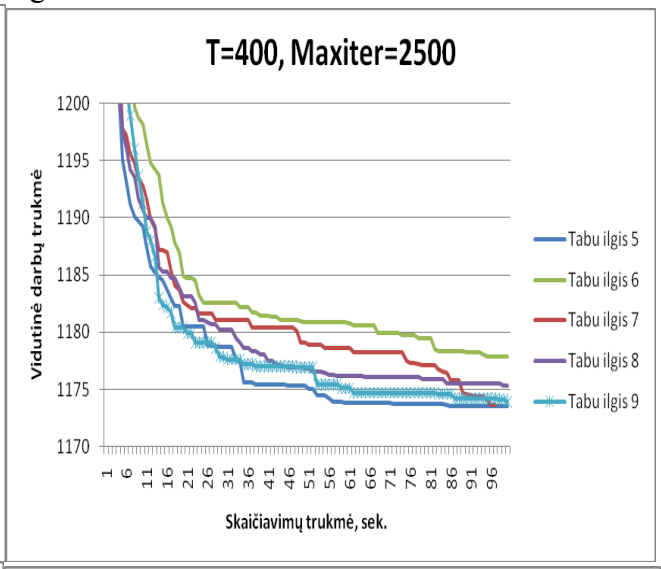


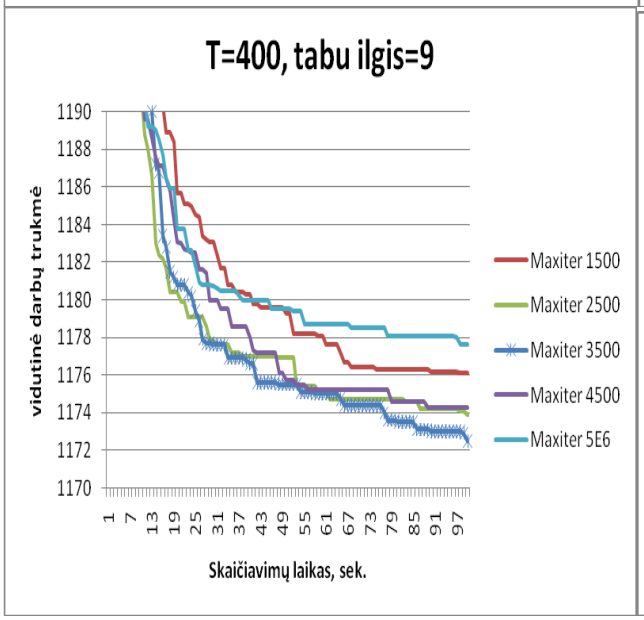


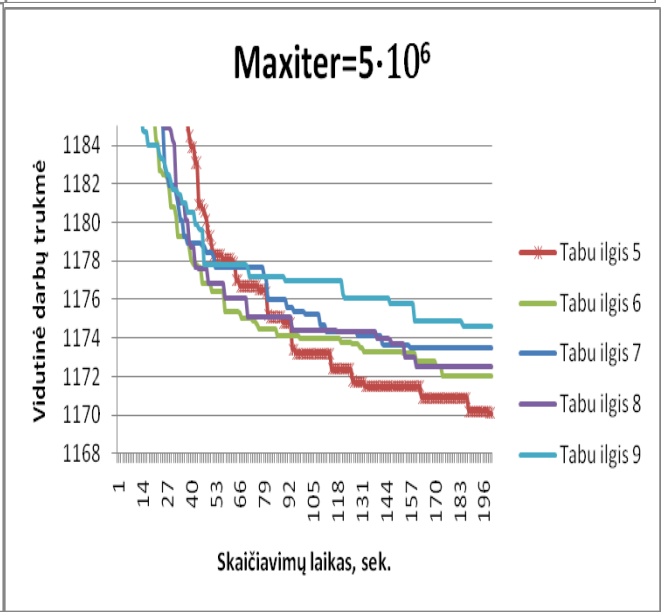


Tabu – atkaitinimo modeliavimo ir tabu paieškos algoritmams:



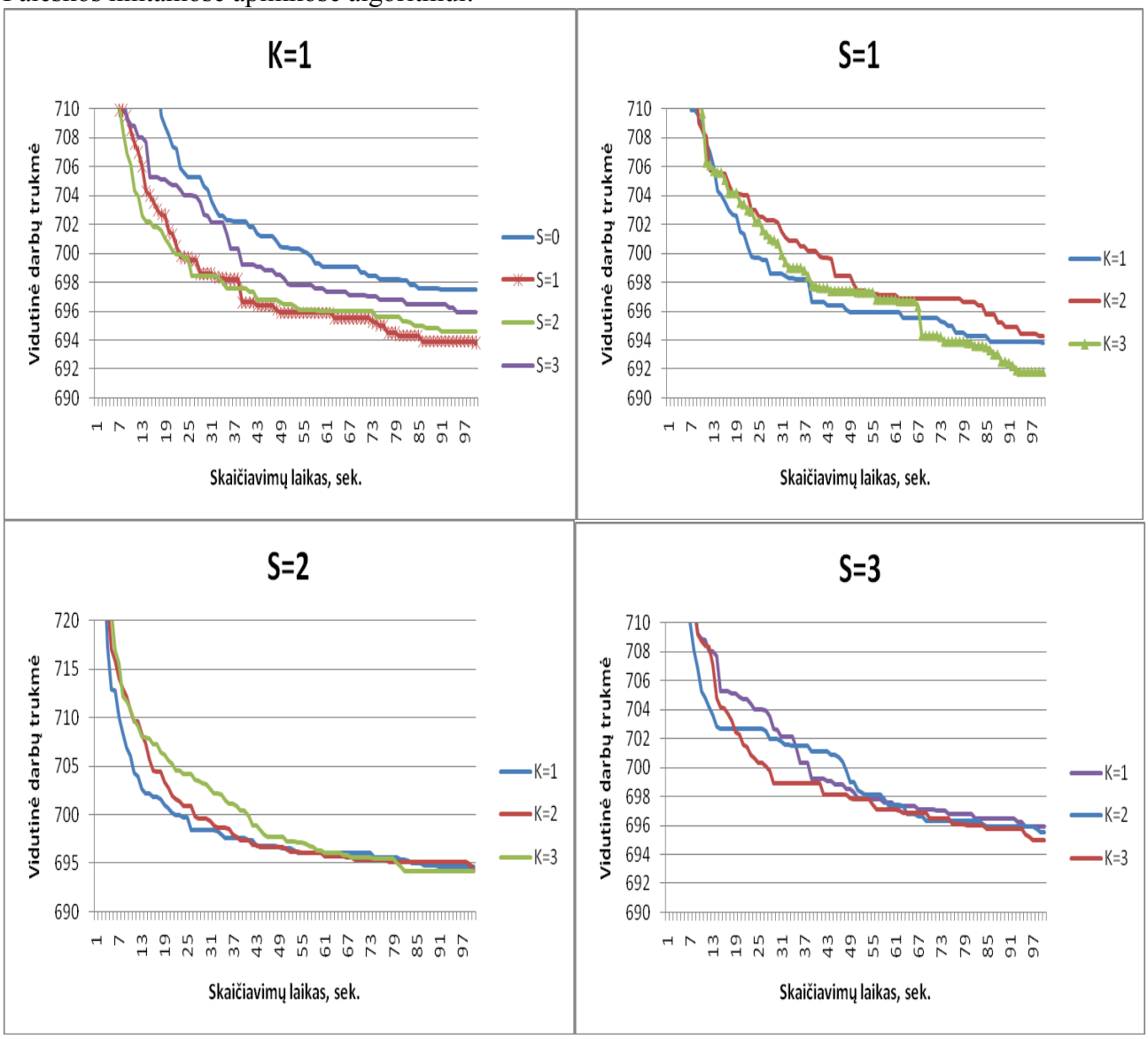




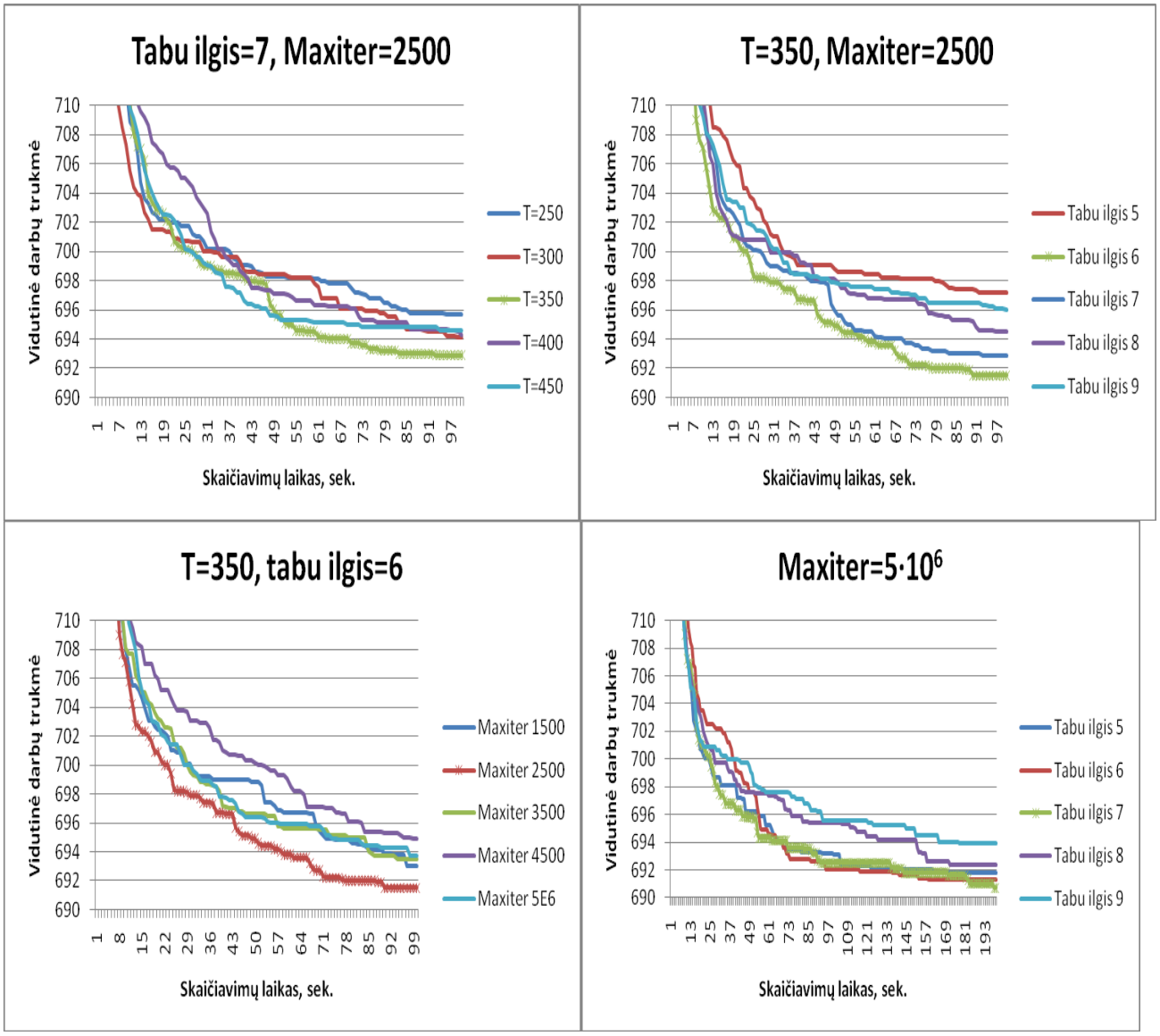


4. Uždavinys ABZ9 (20×15).

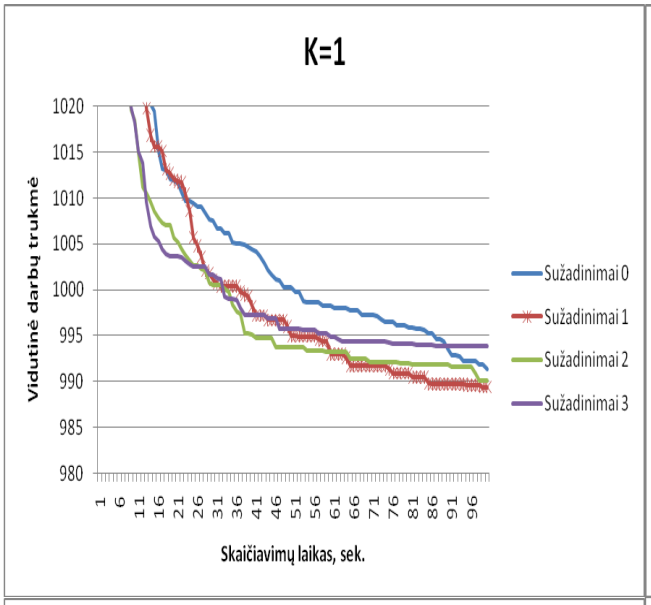
Paieškos kintamose aplinkose algoritmui:

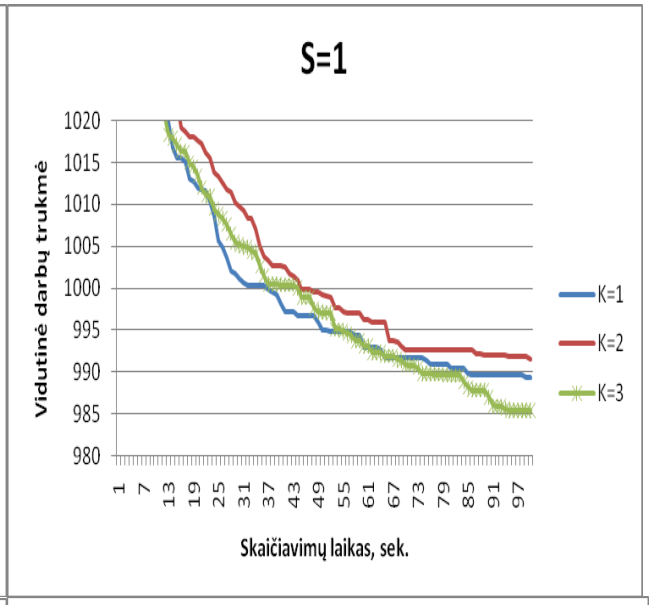


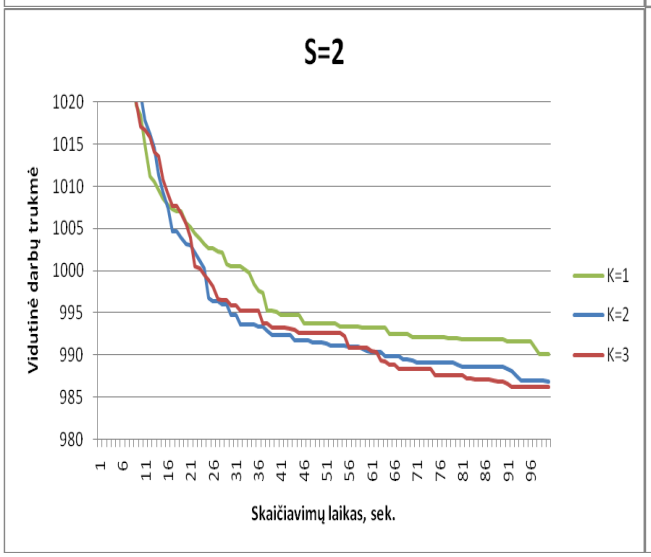
Tabu – atkaitinimo modeliavimo ir tabu paieškos algoritmams:

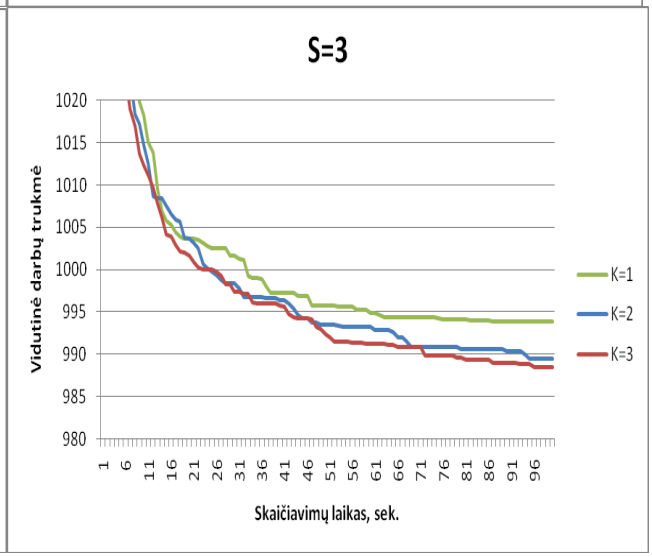


5. Uždavinys YN4 (20×20).
   Paieškos kintamose aplinkose algoritmui:

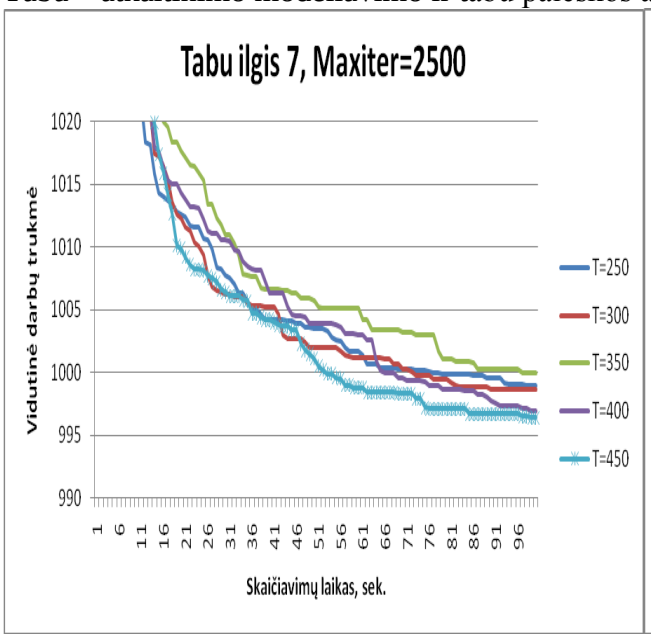


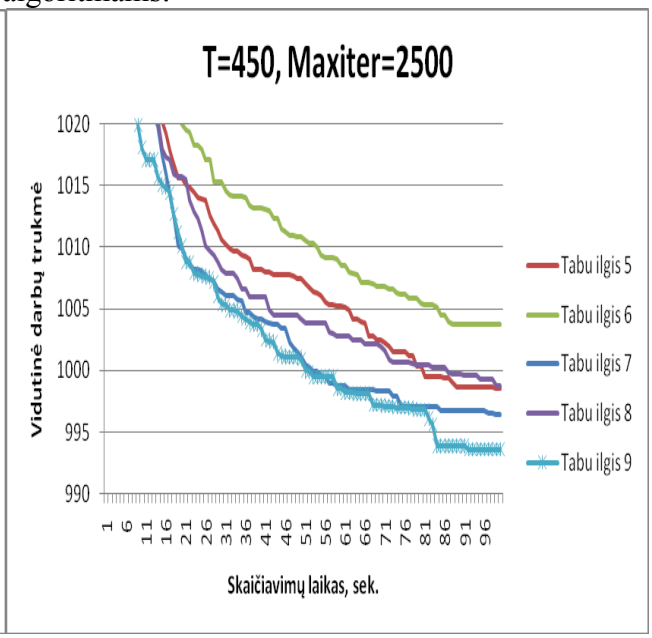


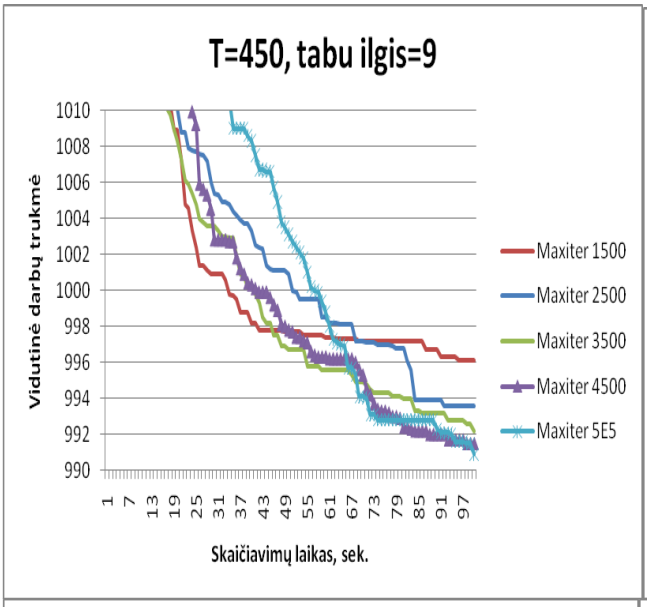


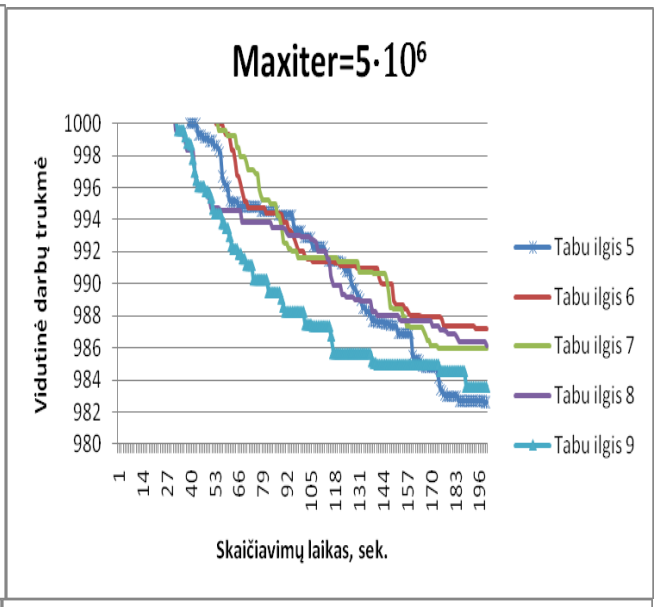


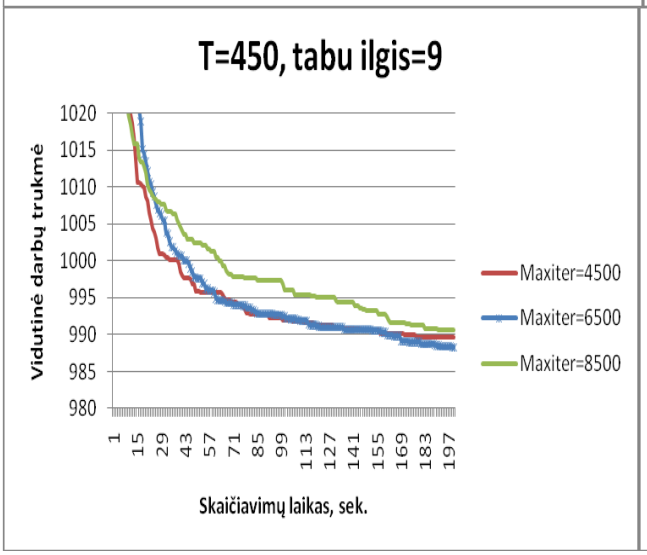
Tabu – atkaitinimo modeliavimo ir tabu paieškos algoritmams:

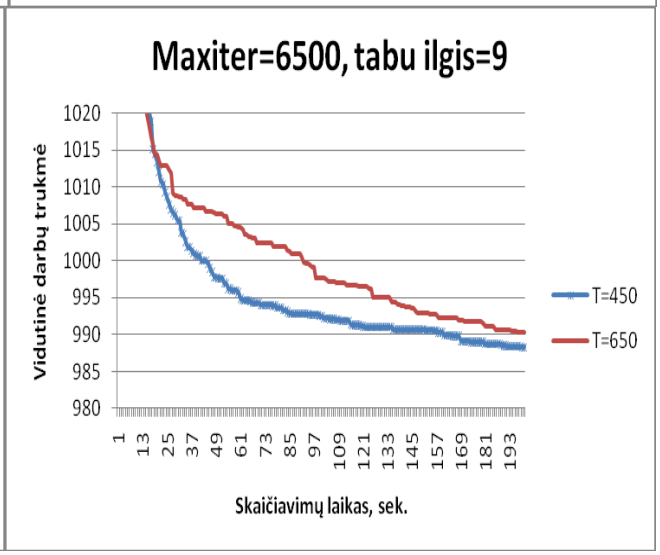






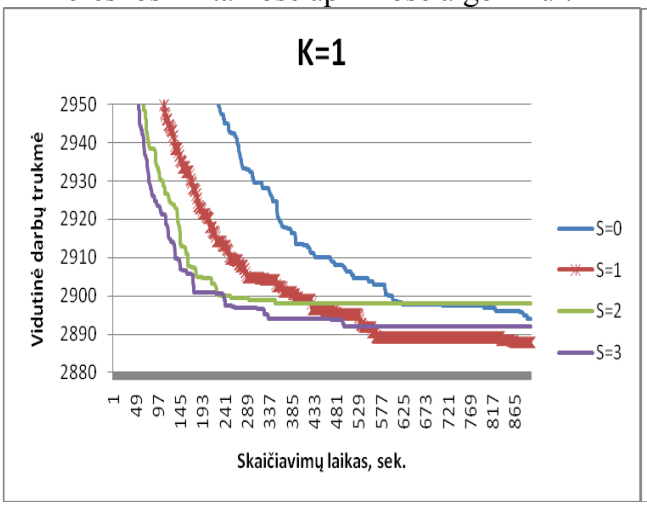


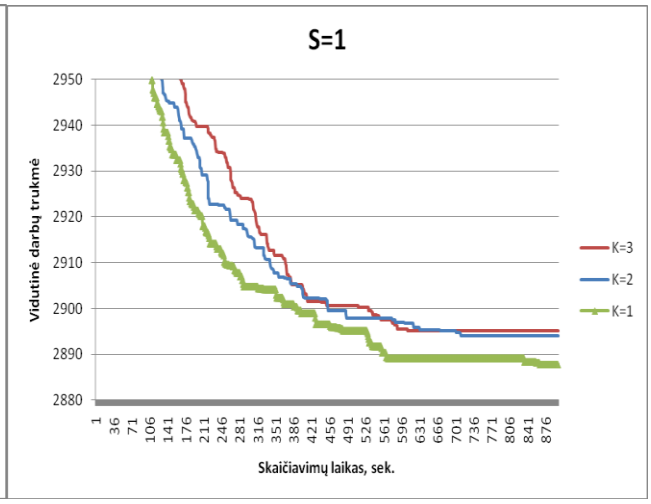


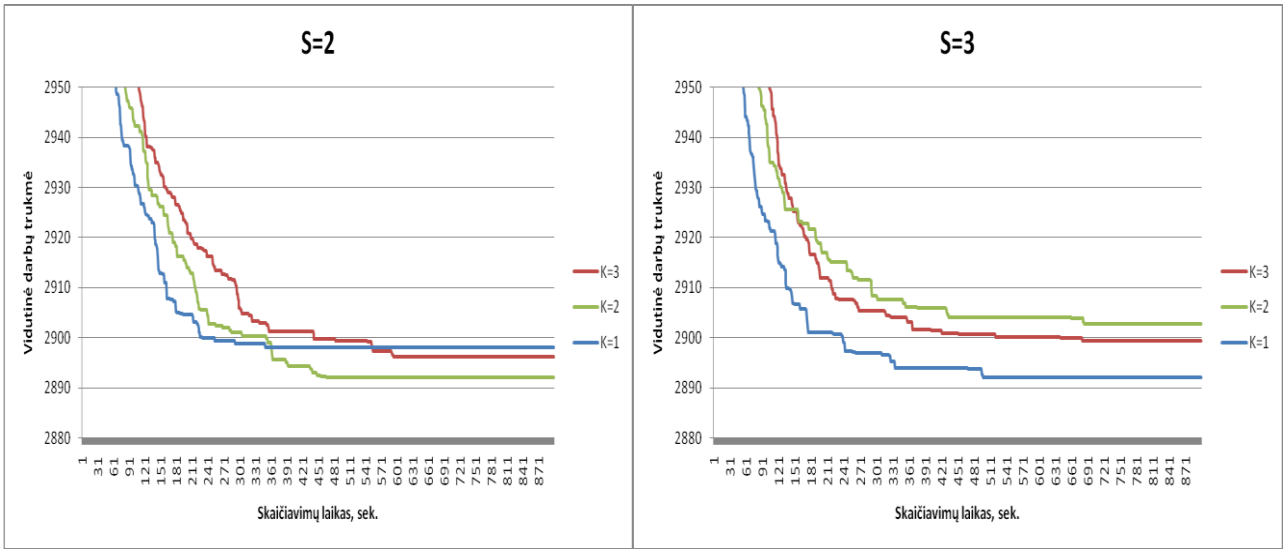


6. Uždavinys TAI62 (50×20).

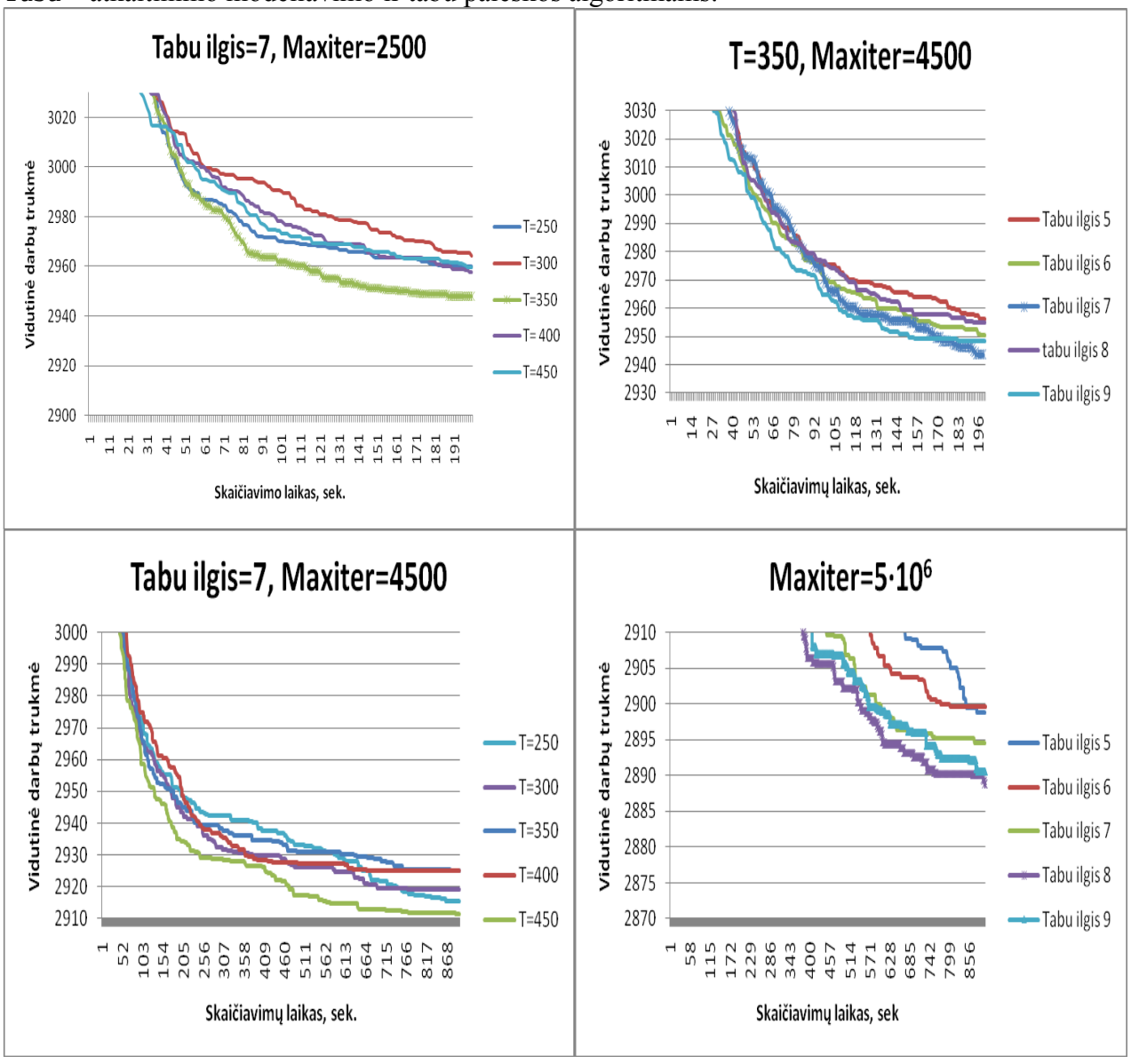
Paieškos kintamose aplinkose algorimui:

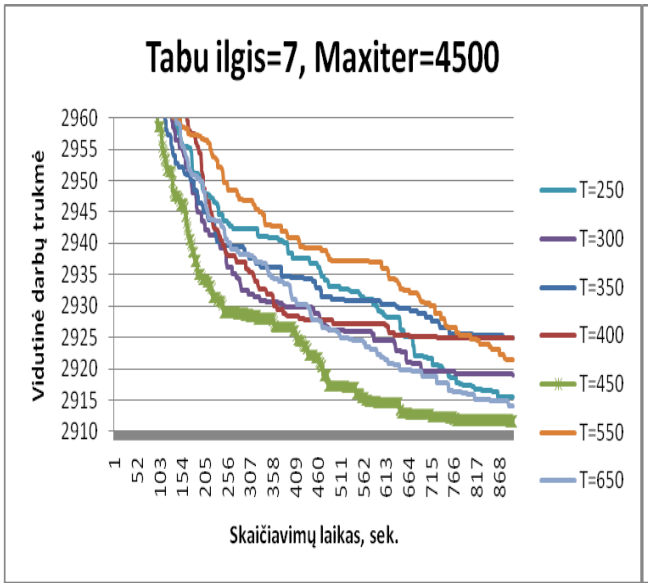


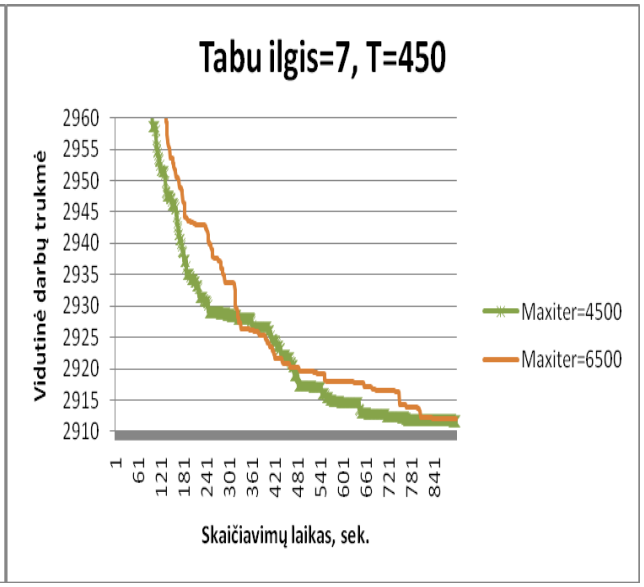




Tabu – atkaitinimo modeliavimo ir tabu paieškos algoritmams:

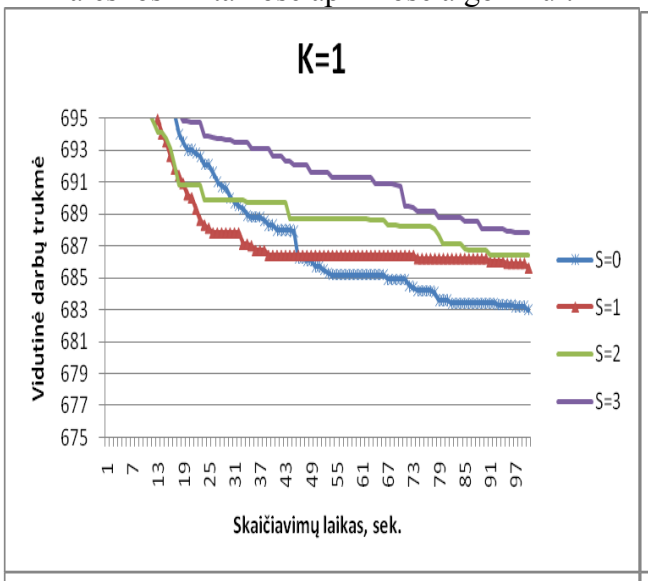


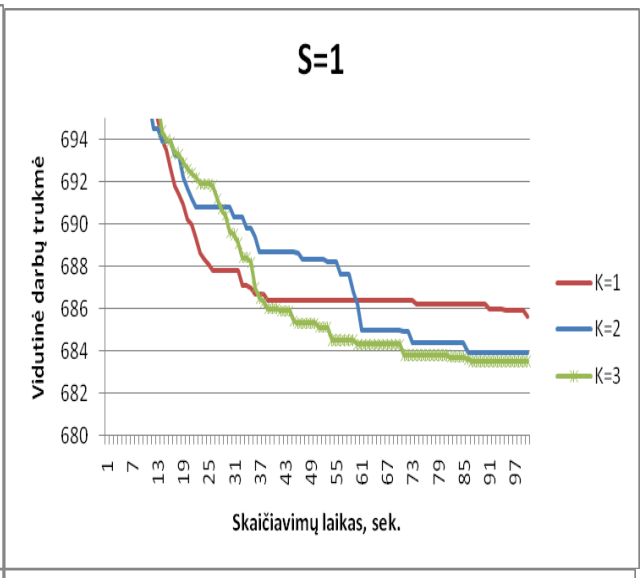


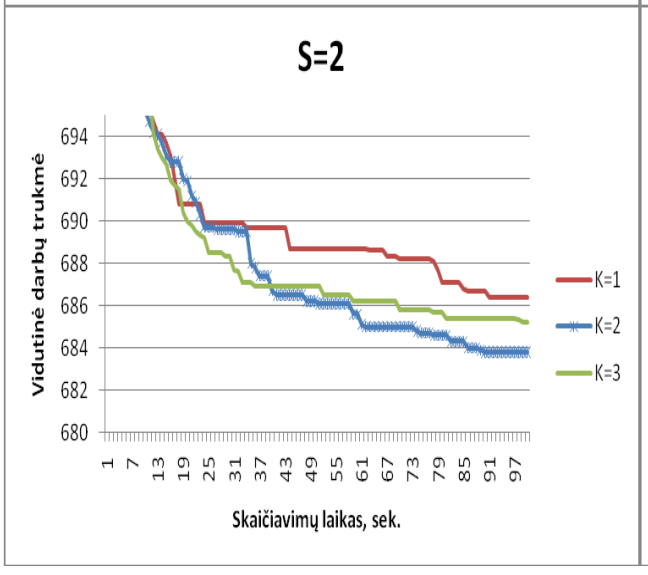


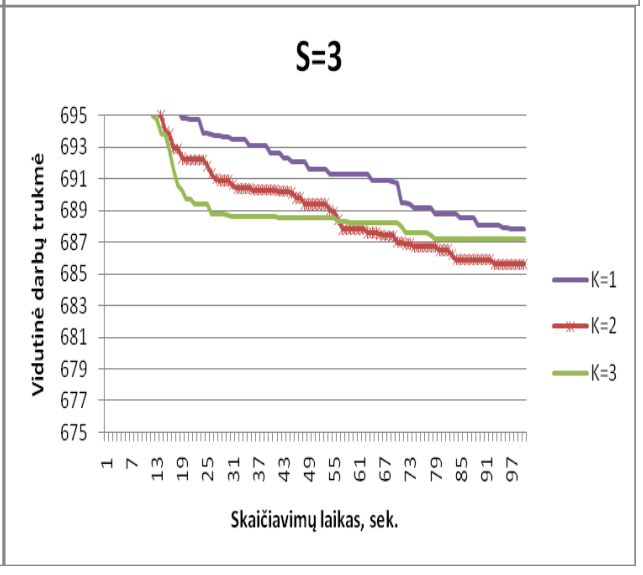
7. Uždavinys ABZ8 (20×15).

Paieškos kintamose aplinkose algorimui:

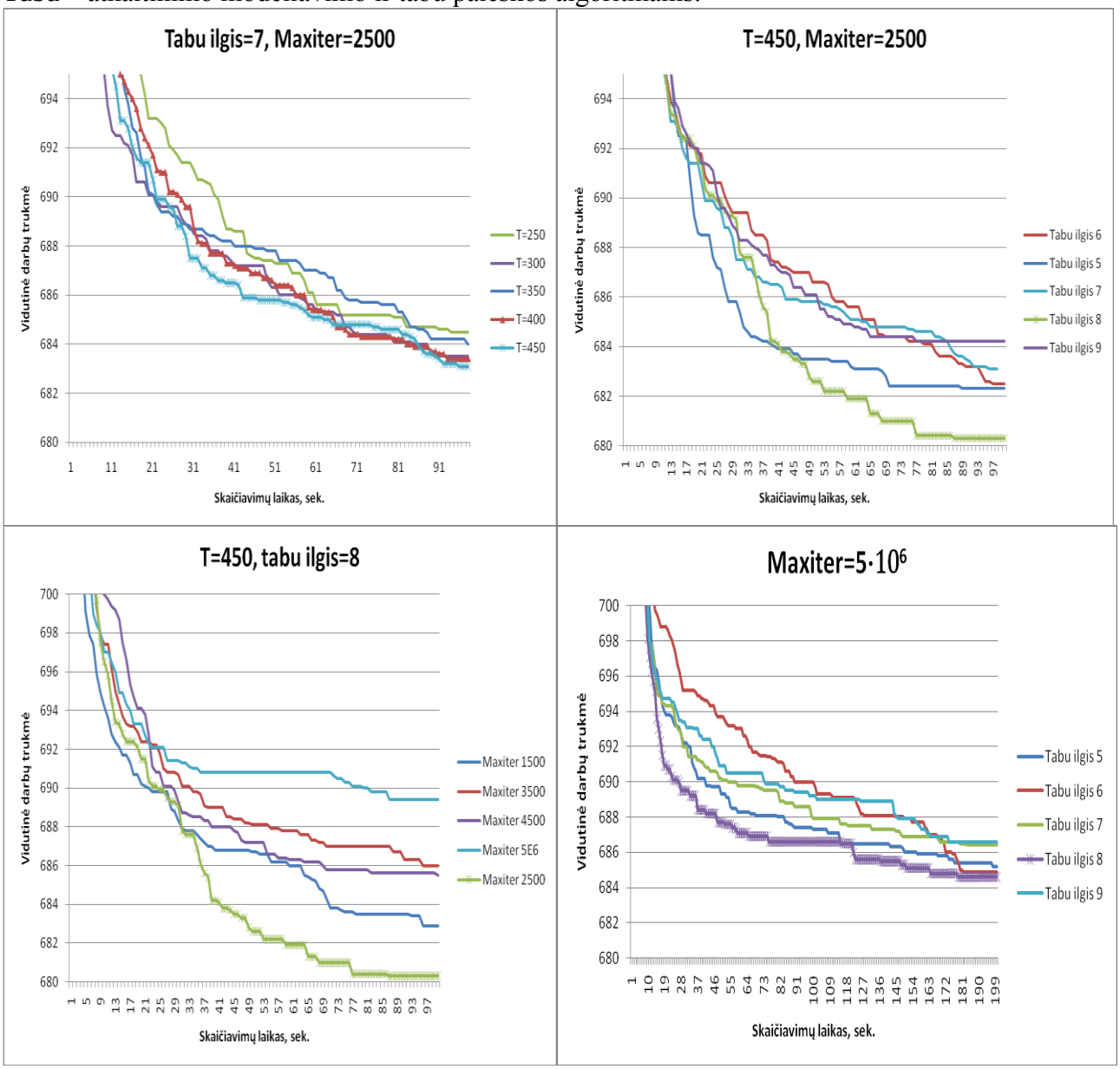






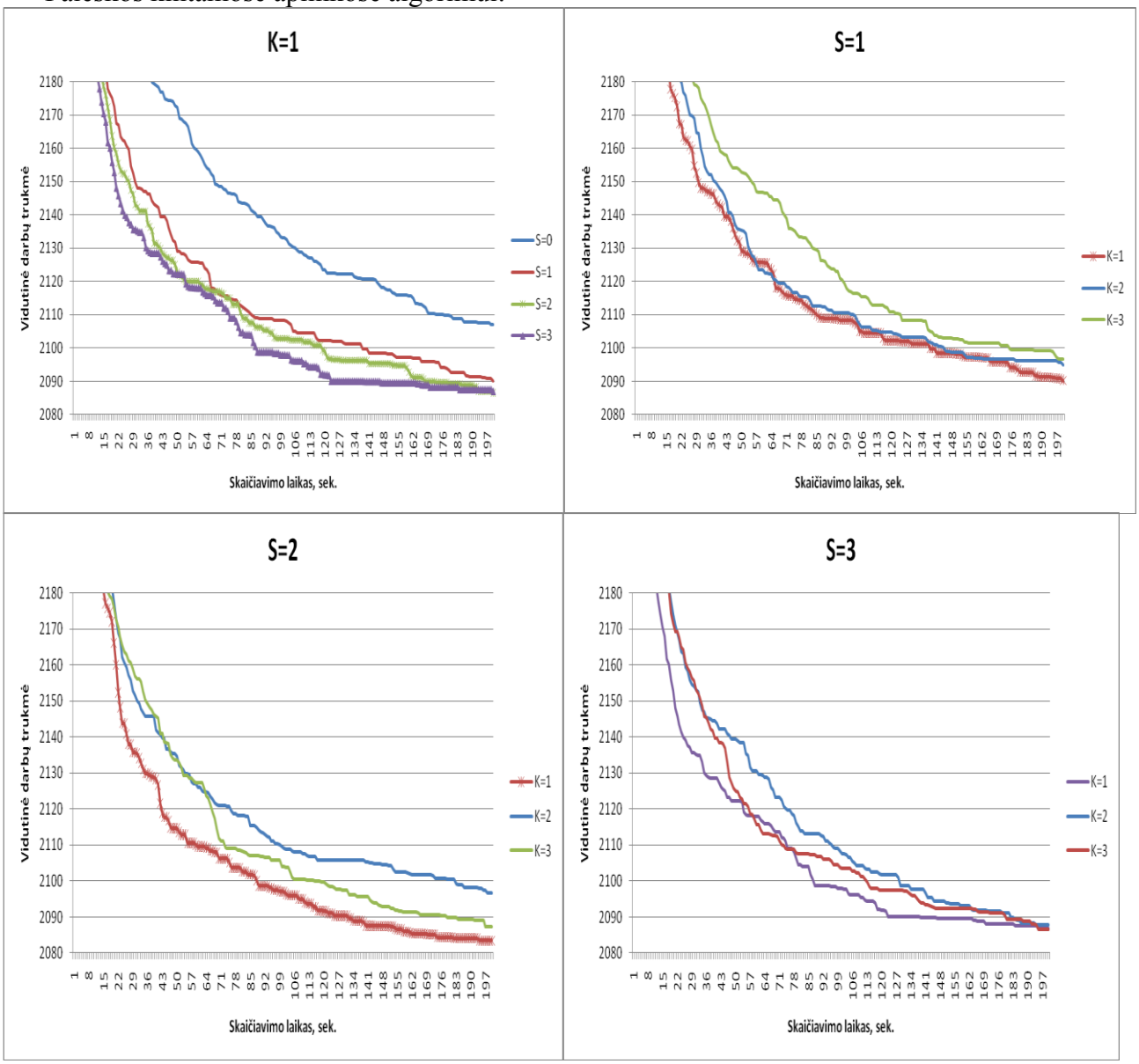


Tabu – atkaitinimo modeliavimo ir tabu paieškos algoritmams:

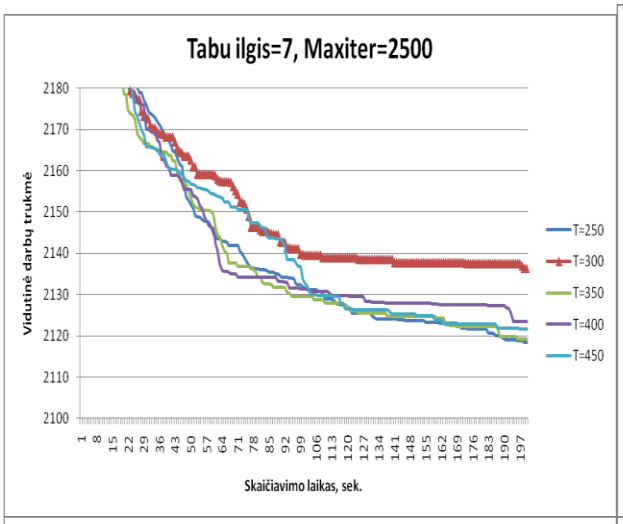


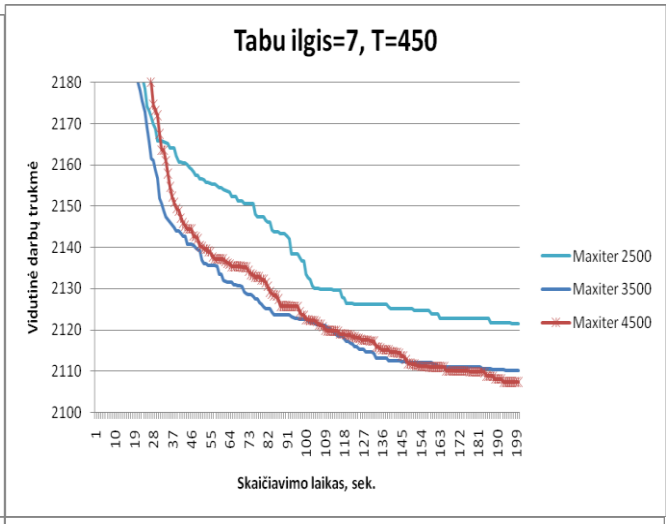
8. Uždavinys TAI41 (30×20).

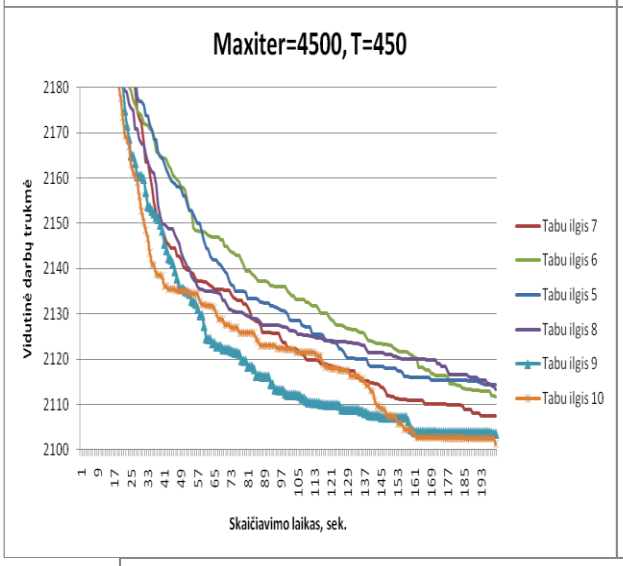
Paieškos kintamose aplinkose algorimui:

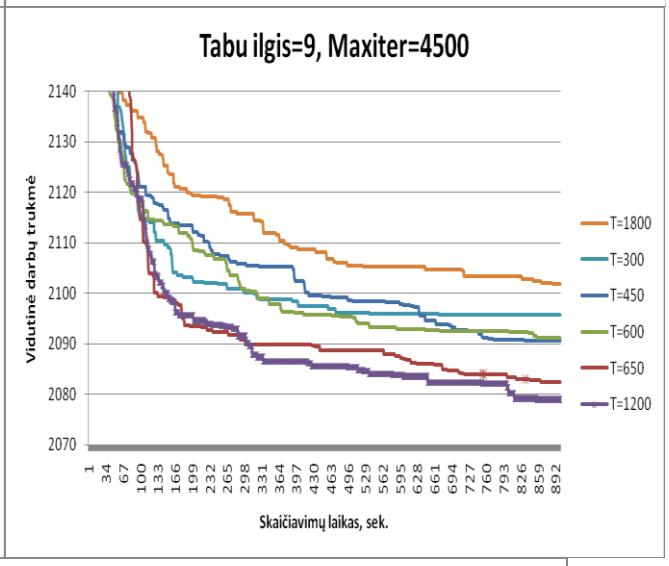


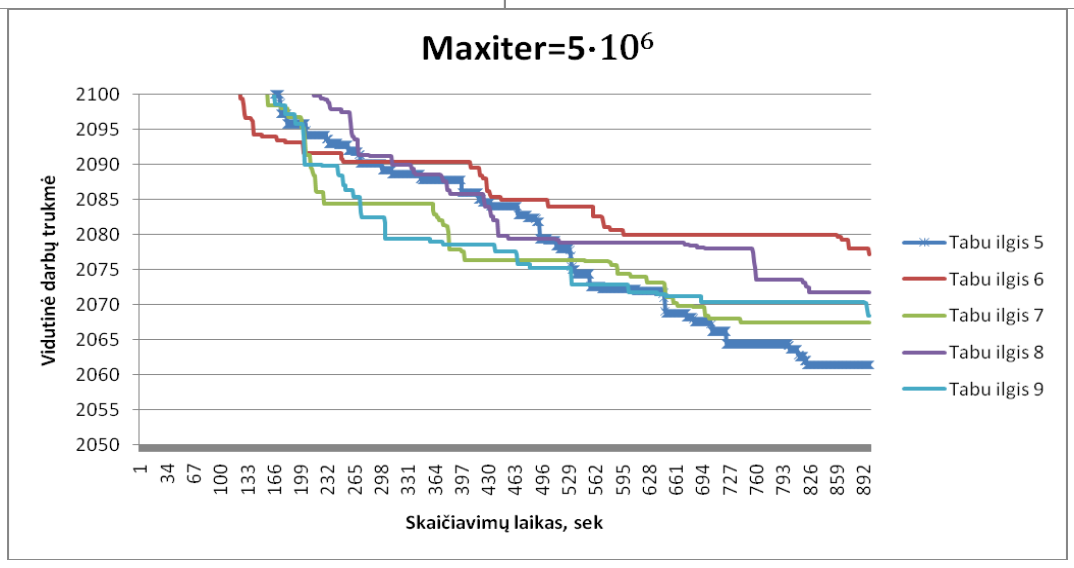
Tabu – atkaitinimo modeliavimo ir tabu paieškos algoritmams:

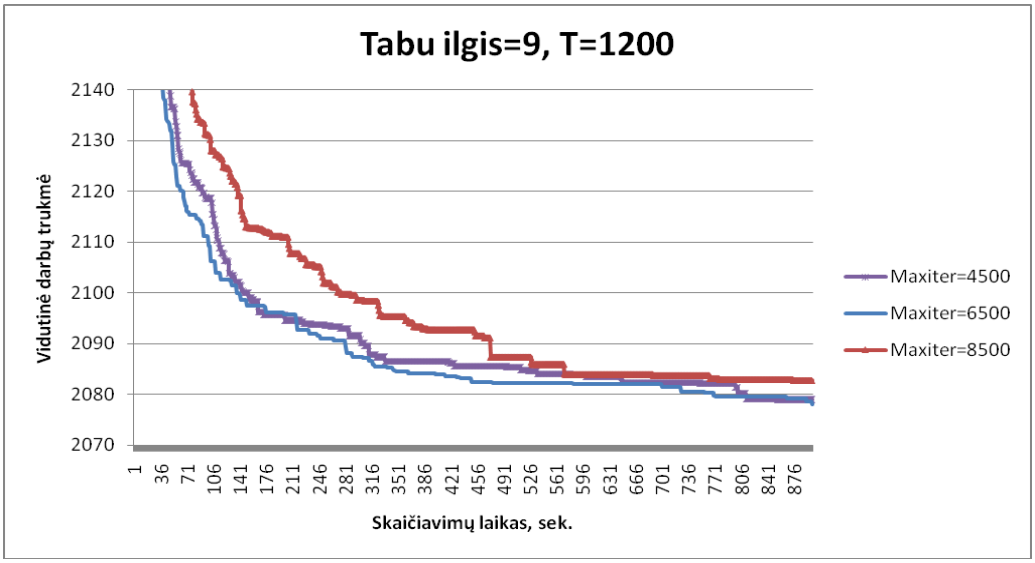

Tabu ilgis=7, Maxiter=2500
Vidutinė darbų trukmė
Skaičiavimo laikas, sek.
T=250
T=300
T=350
T=400
T=450


Tabu ilgis=7, T=450
Vidutinė darbų trukmė
Skaičiavimo laikas, sek.
Maxiter 2500
Maxiter 3500
Maxiter 4500


Maxiter=4500, T=450
Vidutinė darbų trukmė
Skaičiavimo laikas, sek.
Tabu ilgis 7
Tabu ilgis 6
Tabu ilgis 5
Tabu ilgis 8
Tabu ilgis 9
Tabu ilgis 10

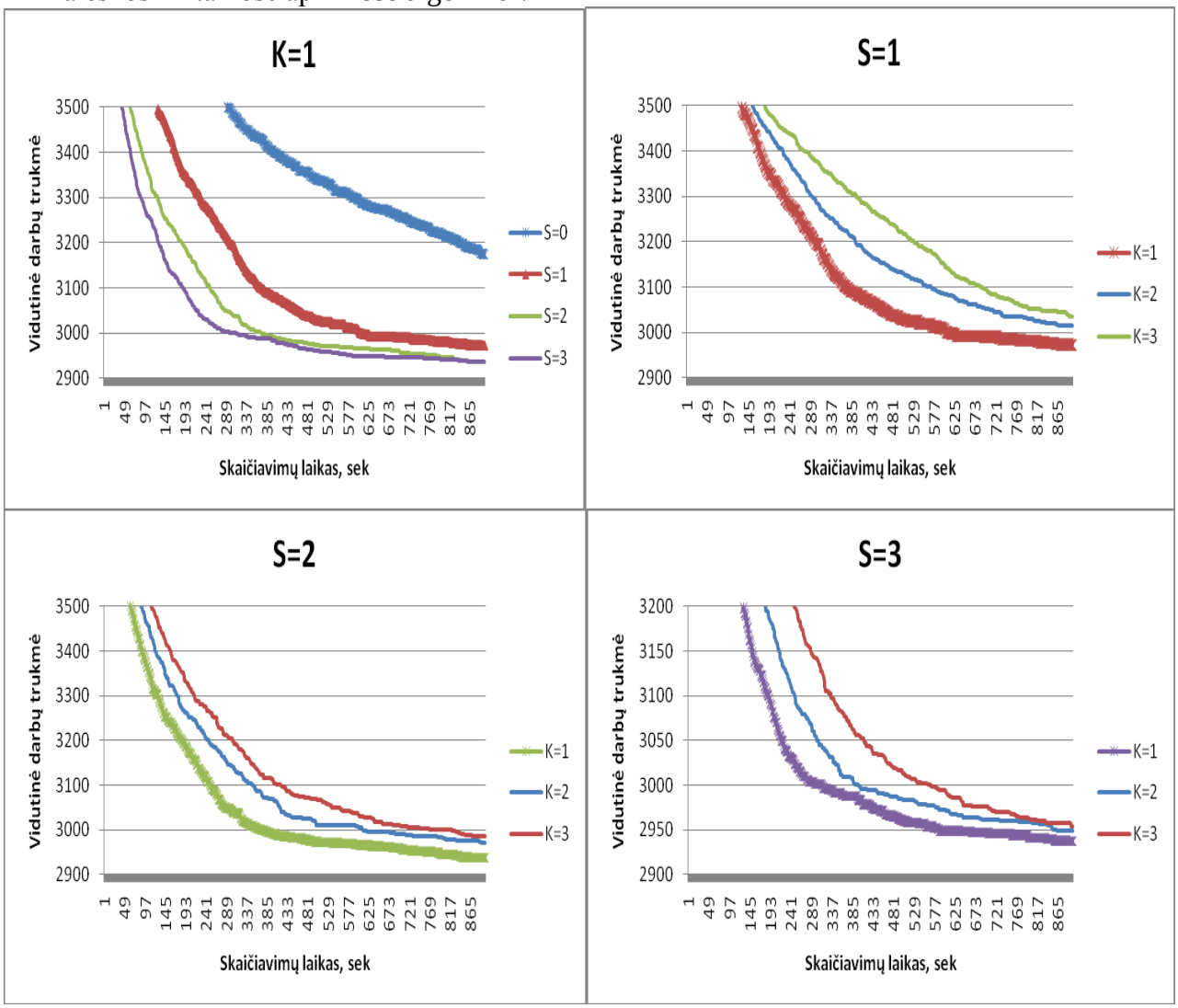

Tabu ilgis=9, Maxiter=4500
Vidutinė darbų trukmė
Skaičiavimų laikas, sek.
T=1800
T=300
T=450
T=600
T=650
T=1200


Maxiter=5·10^6
Vidutinė darbų trukmė
Skaičiavimų laikas, sek
Tabu ilgis 5
Tabu ilgis 6
Tabu ilgis 7
Tabu ilgis 8
Tabu ilgis 9

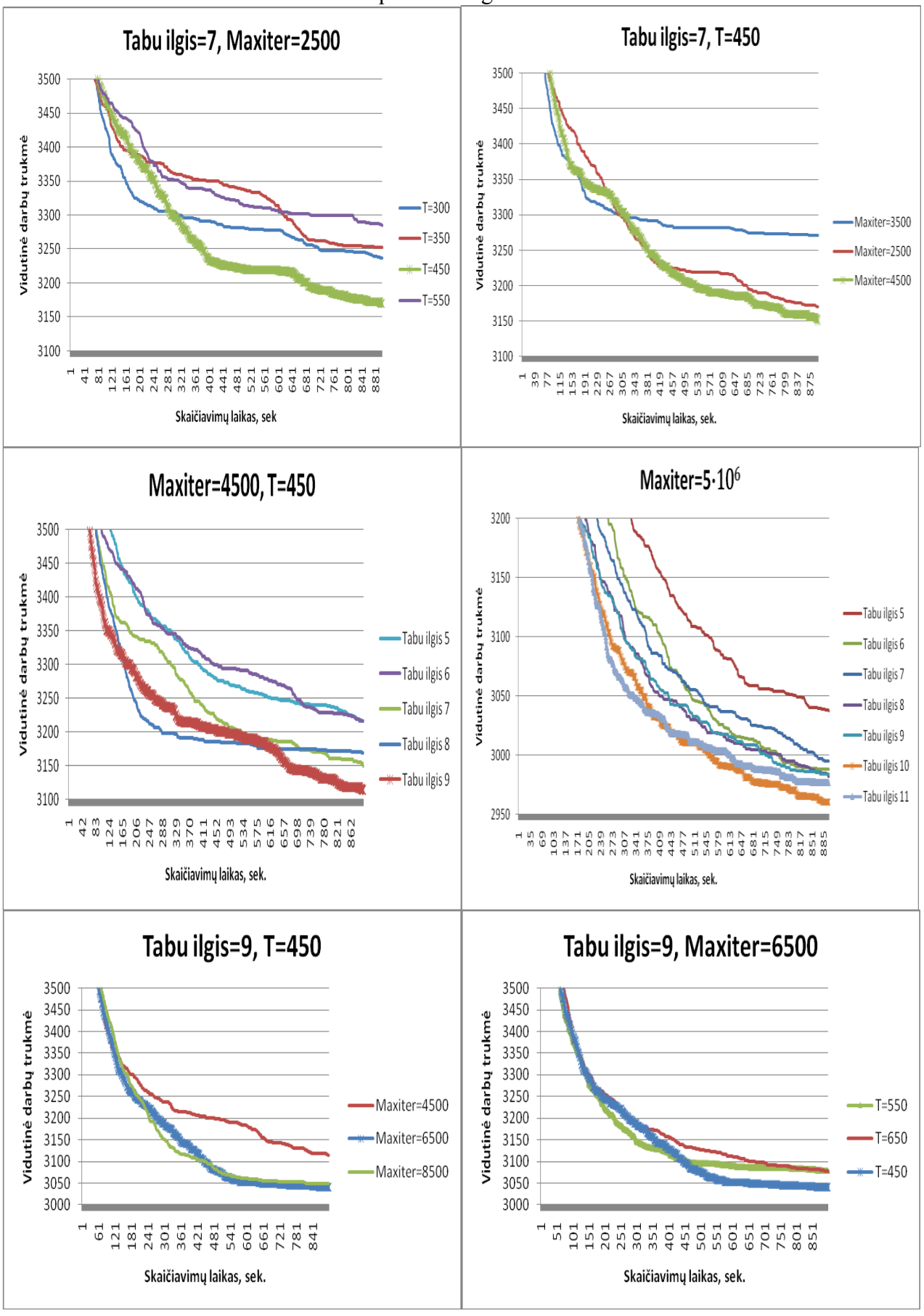


9. Uždavinys SWV15 (50×10).

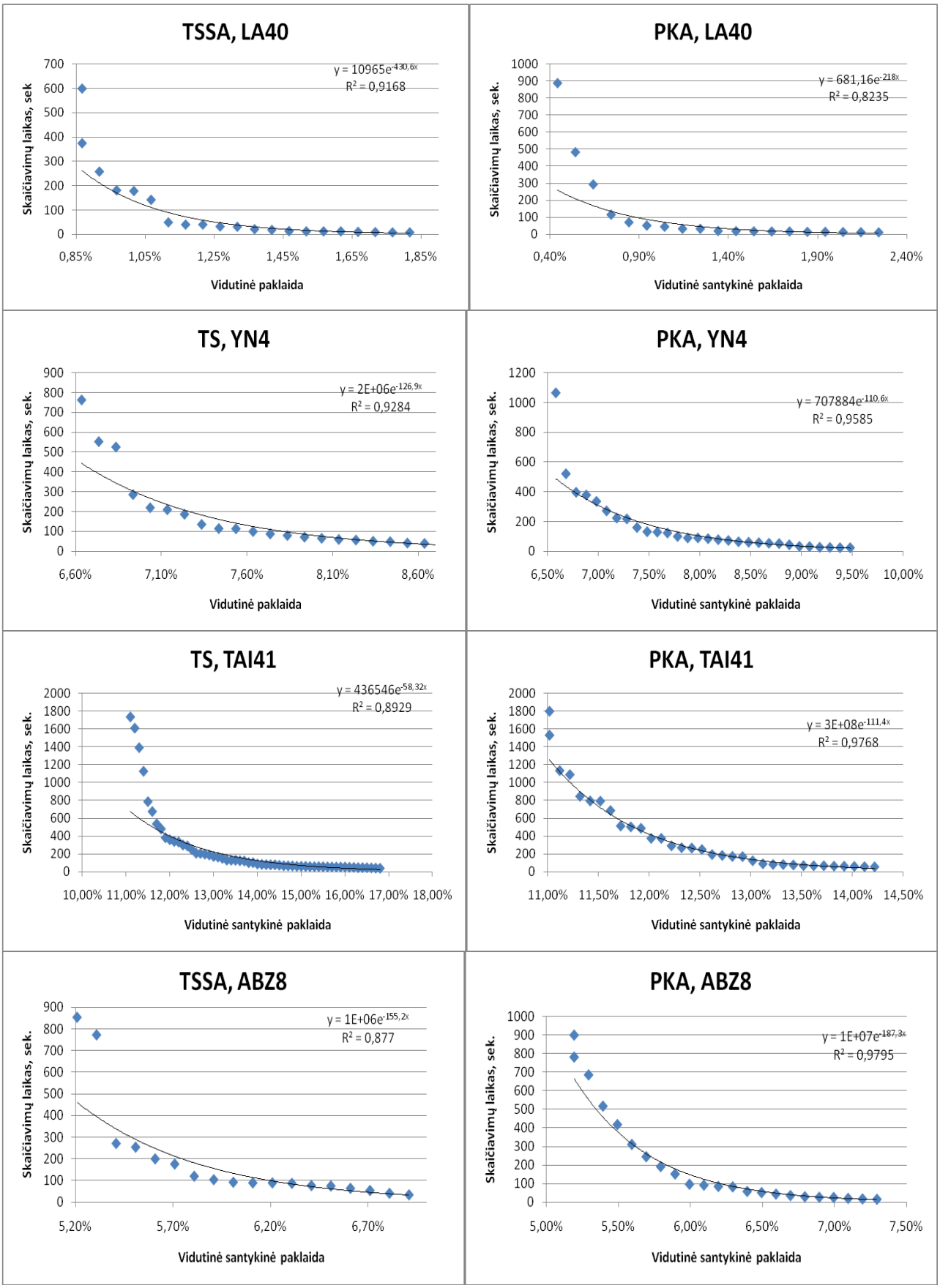
Paieškos kintamose aplinkose algorimui:



Tabu – atkaitinimo modeliavimo ir tabu paieškos algoritmams:

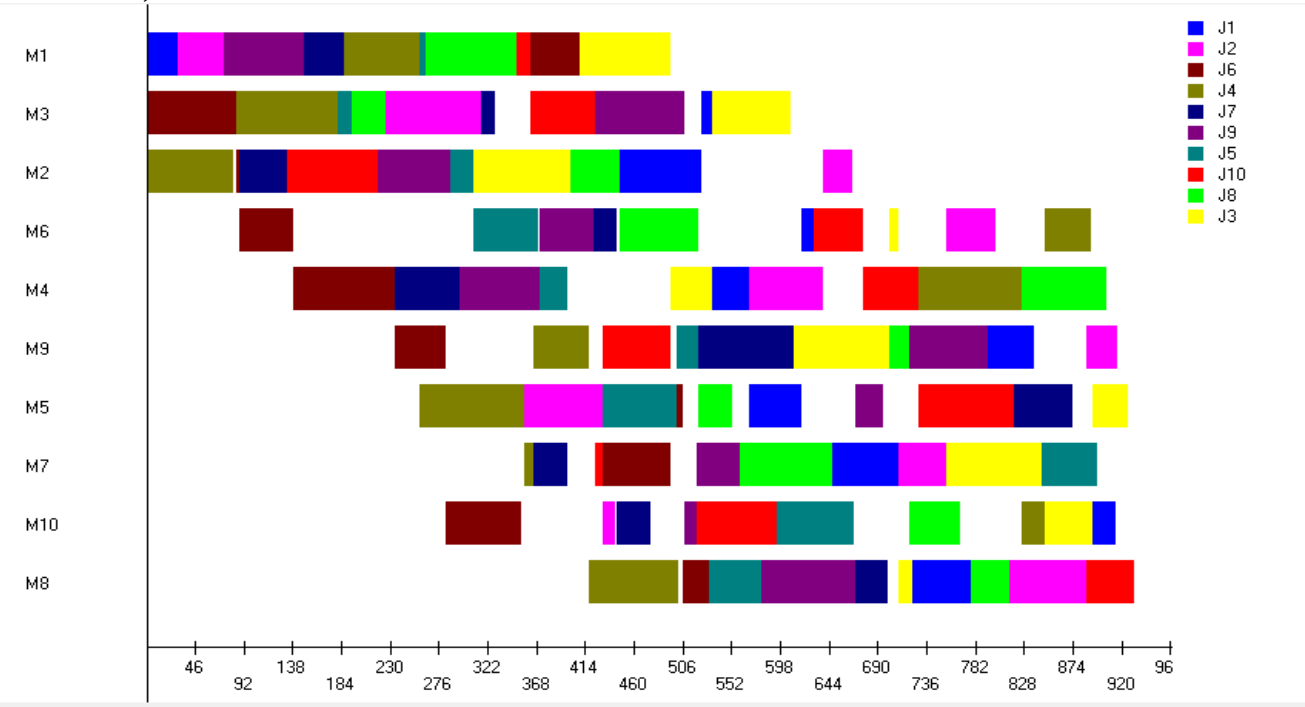


3. PRIEDAS: PAKLAIDŲ TAŠKINĖS DIAGRAMOS

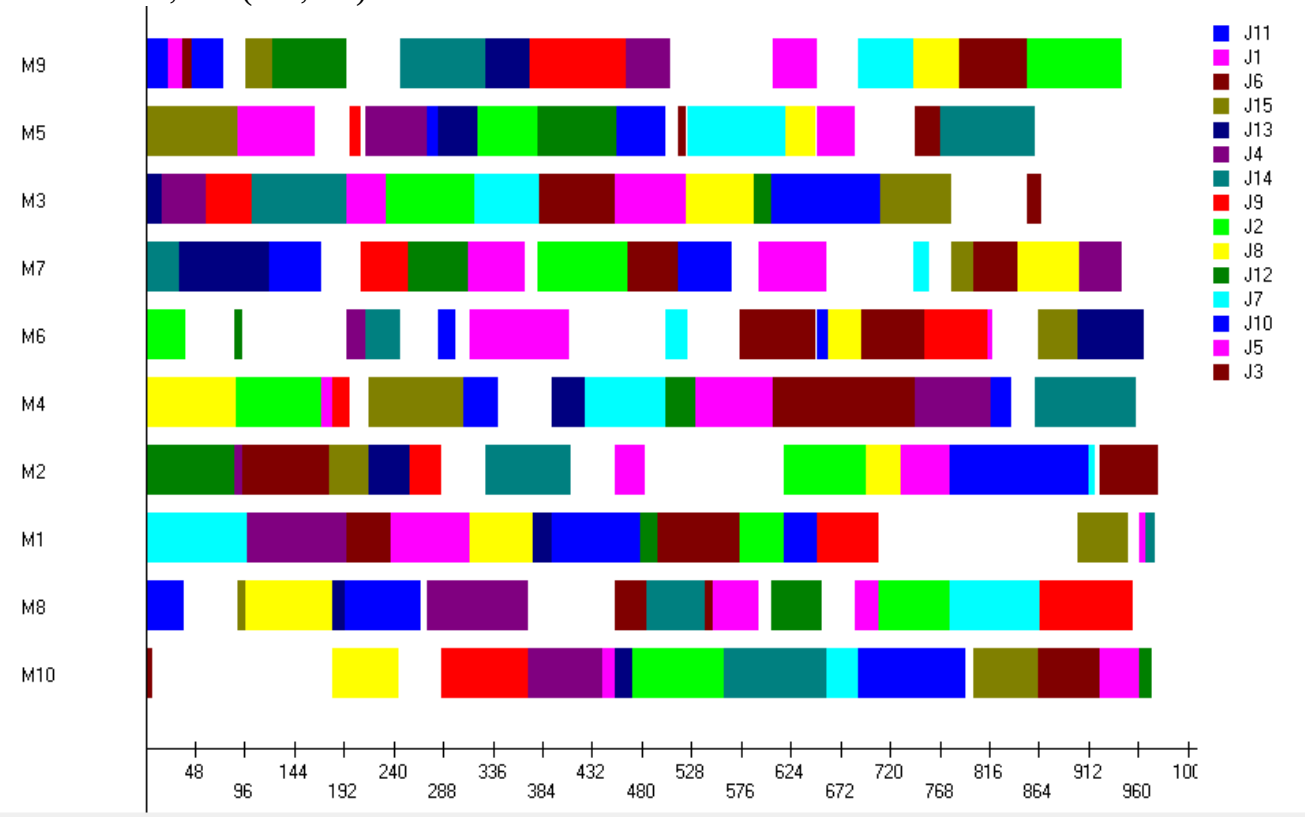


4. PRIEDAS: GERIAUSIŲ RASTŲ SPRENDINIŲ GANTO DIAGRAMOS

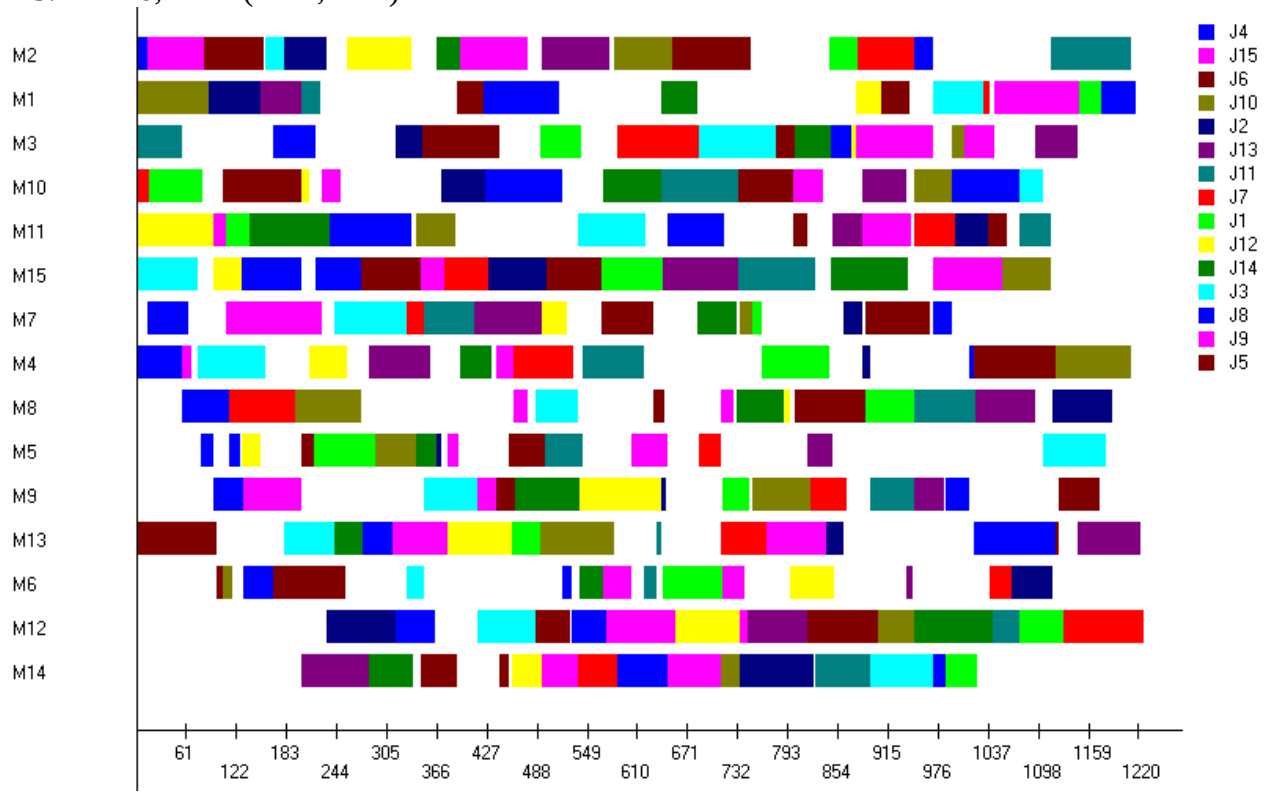
1. FT10, 930 (optimalios darbų trukmės įvertis 930, geriausio žinomo sprendinių darbų trukmė 930)



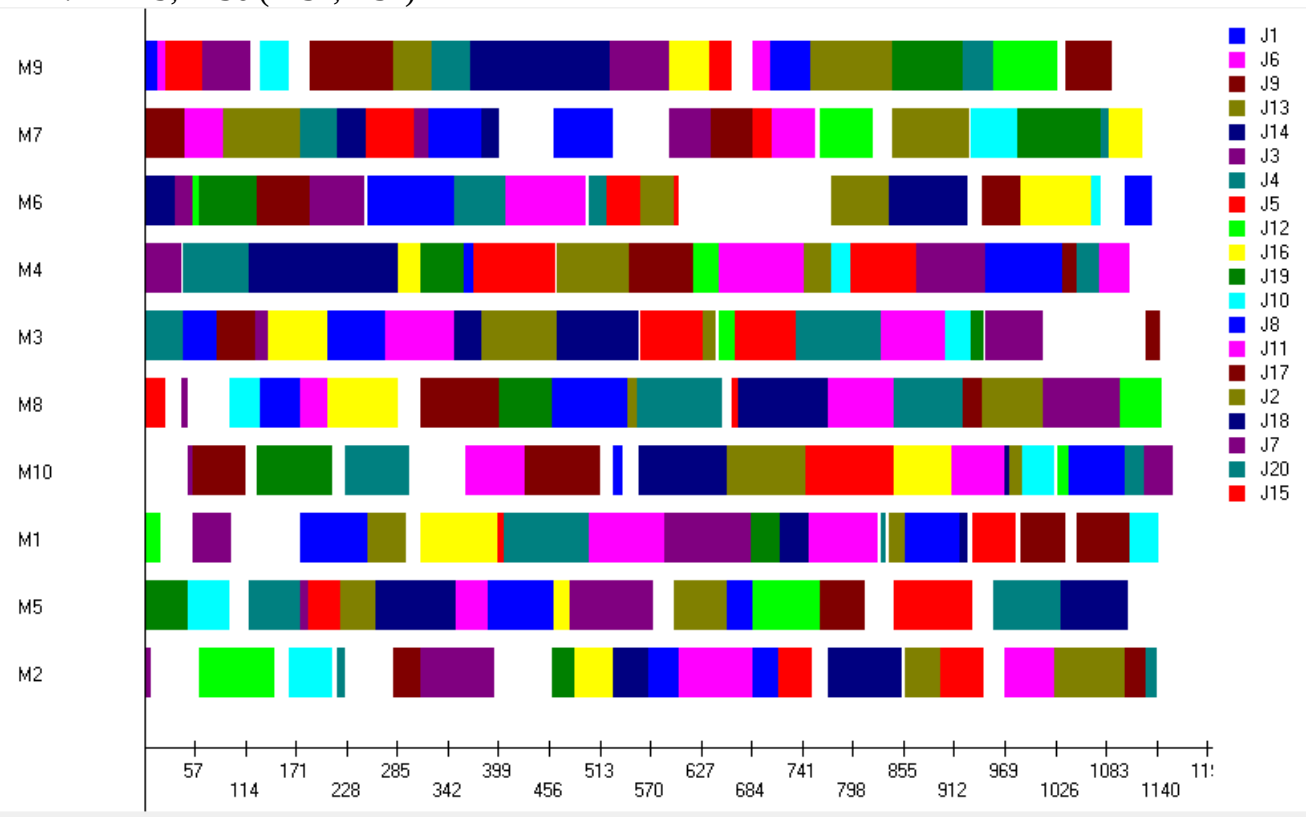
2. LA25, 977 (977,977)



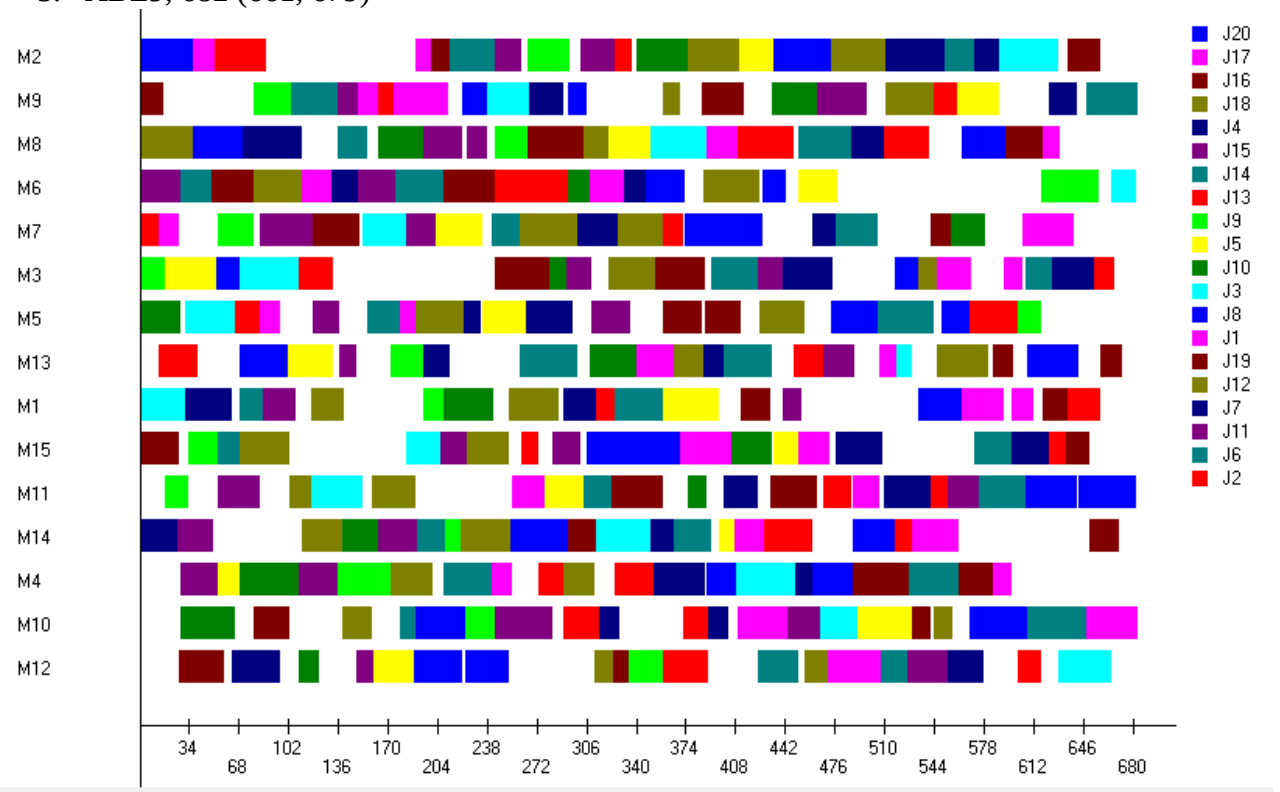
3. LA40, 1224 (1222,1222)



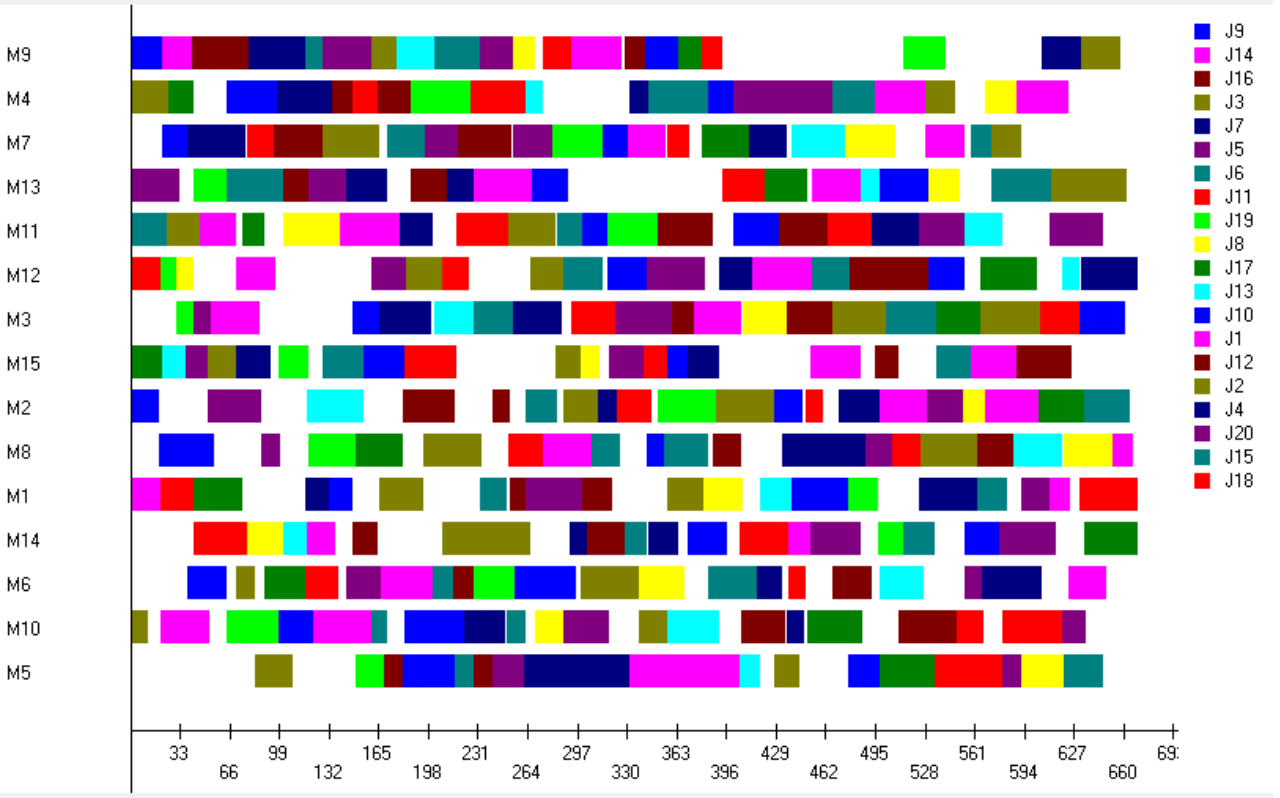
4. LA29, 1156 (1152,1152)



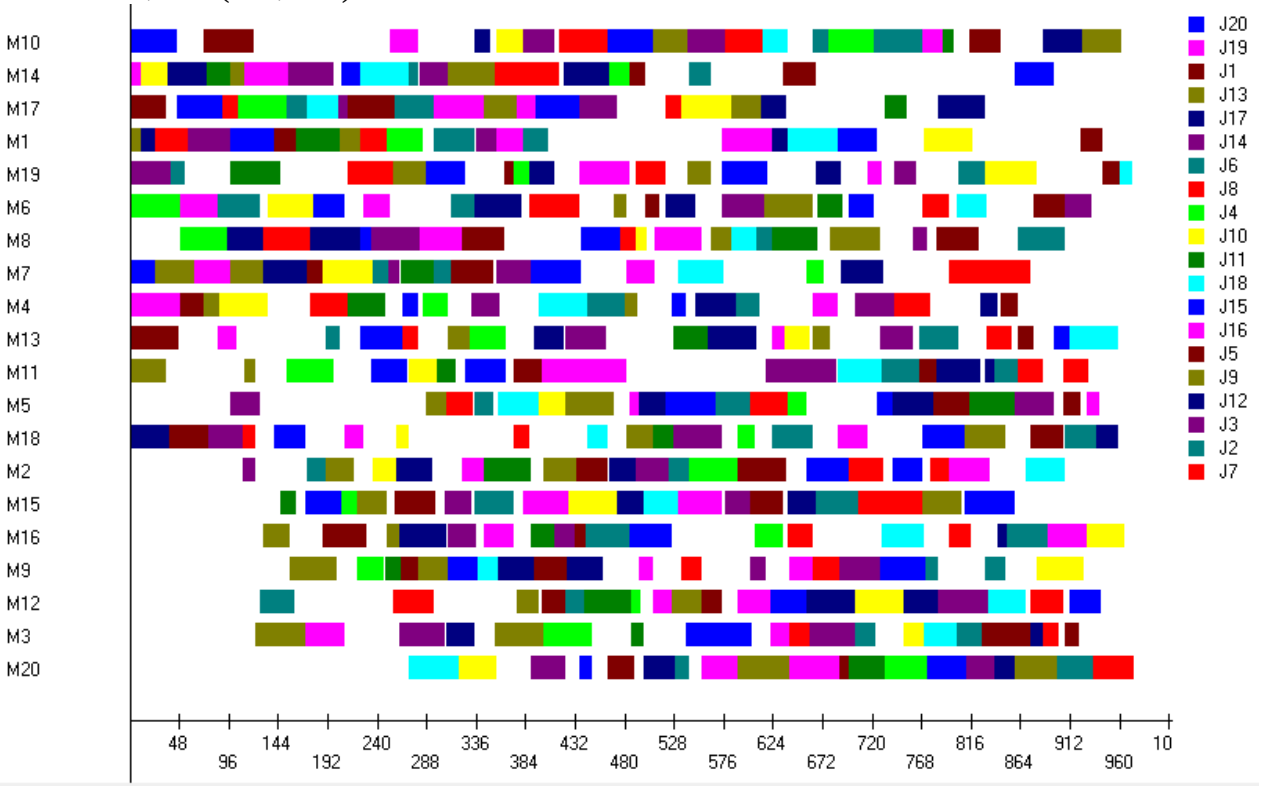
5. ABZ9, 682 (661, 679)



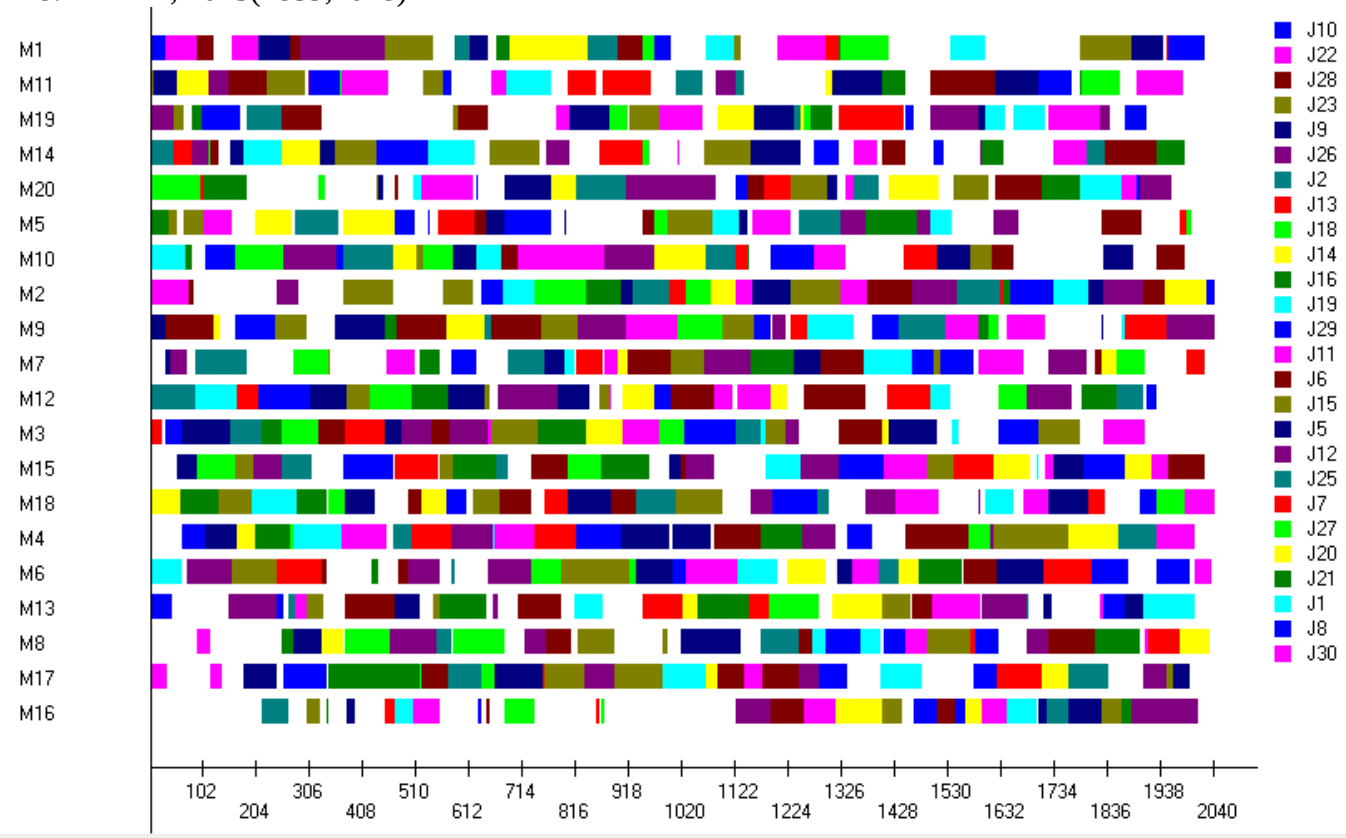
6. ABZ8, 669 (645, 665)



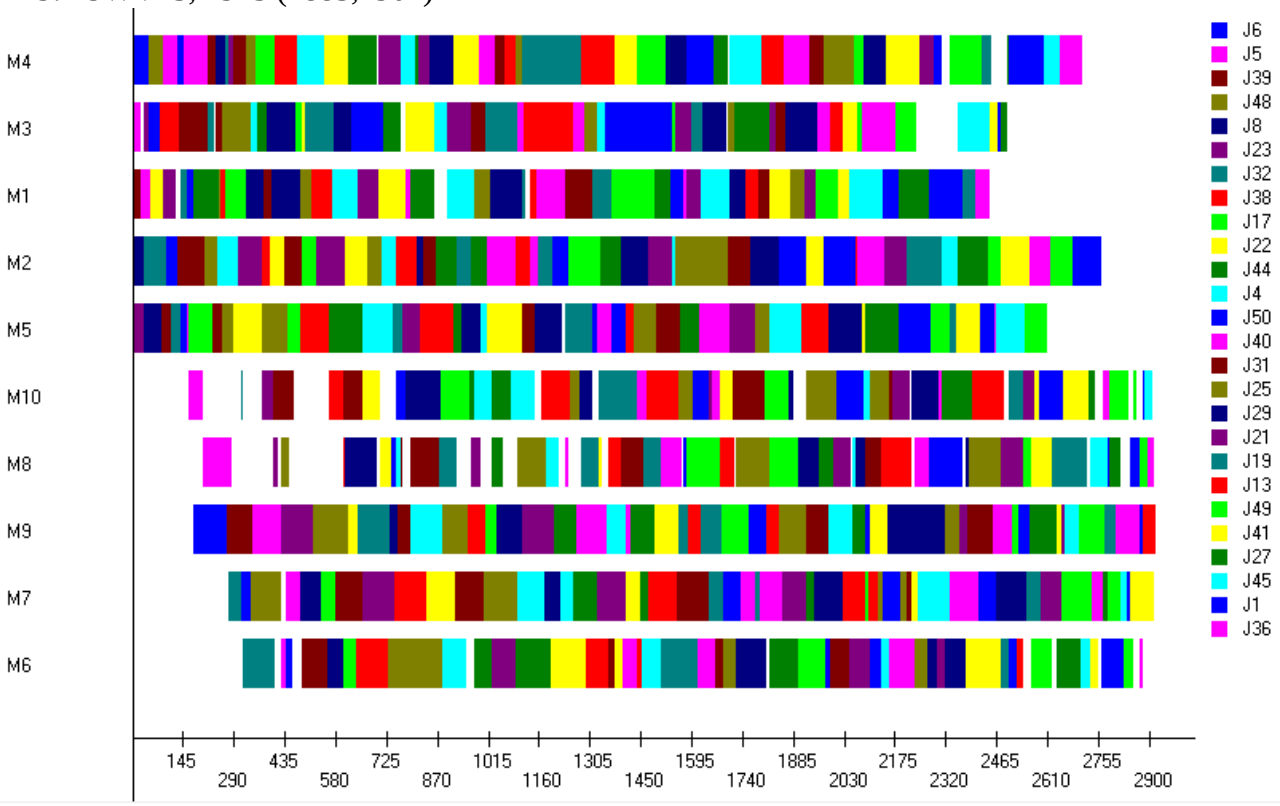
7. YN4, 972 (918, 968)



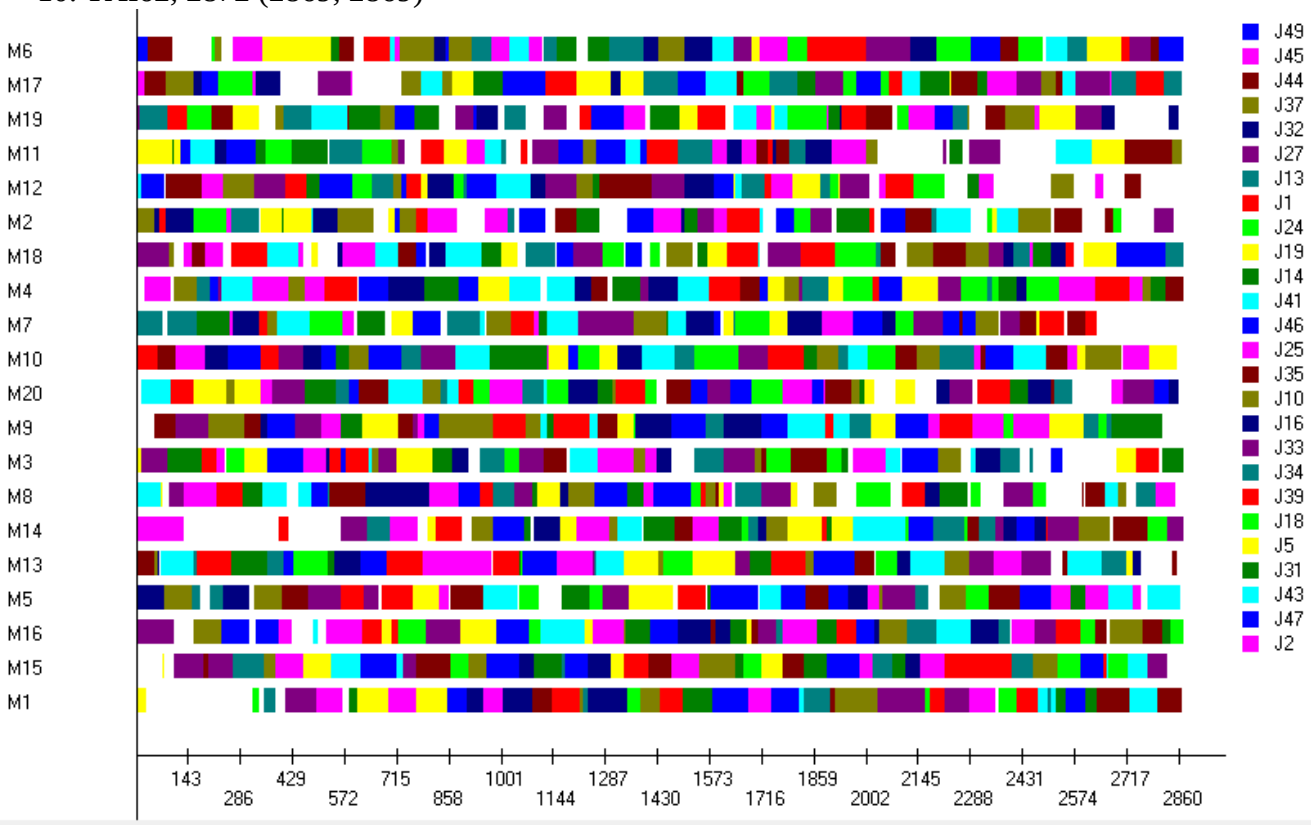
8. TAI41, 2043(1859,2018)



9. SWV15, 2913 (2885,2904)



10. TAI62, 2872 (2869, 2869)



5. PRIEDAS: PAGRINDINIAI PROGRAMOS TEKSTAI

void TForm1:: TSSA() // tabu – atkaitinimo modeliavimas

```
{
  ofstream fl;
  ofstream fr;
  ofstream fg;
  fl.open("laikai.txt");
  fr.open("reiksmes.txt", ios::app);
  fg.open("elite.txt");
  AnsiString E;
  Series1->Clear();
  chrom darbine;
  chrom geresne, geriausia;
  generuoti(0,1);
  int maxgen = StrToInt(Edit2->Text);
  int laikas = StrToInt(Edit6->Text);
  double itlaikas = 0;
  double skaitliukas = 0;
  double intervalas = 0.5;
  struct  time t;
  gettime(&t);
  pradval = t.ti_hour;
  pradmin = t.ti_min;
  pradsek = t.ti_sec;
  pradsim = t.ti_hund;
  int iter = StrToInt(Edit7->Text);
  tabusk = 0;
  aplN = 0;
  geriausia = pop[0];
  geresne = geriausia;
  abc = ivertinti(geresne);
  KritinisKelias(abc);
  aplElementai(geresne);
  double globalus = abc.ms;
  double lokalus;
  maxtabu = StrToInt(Edit1->Text);
  double mat[N];
  int pos [N];
  int aspi[N];
  int matN = 0;
  int asp = 0;
  int ats = 0;
  double cikl[N];
  int ciklN = 0;
  bool ciklas = false;
  elitN = 0;
  double temp0 = StrToFloat(Edit9->Text);
  temp = globalus/temp0;
  for(int j = 0; j <maxgen && itlaikas < laikas; j++)
  {
   double min = abc.ms;
    matN = 0;
    int mk;
    genas gen1;
    int max;
    bool ar = ArYraNeTabu(geresne, max);

    while(!ar && tabusk > 0)
    {for(int i = max; i < tabusk-2; i++)
     {tabu[i] = tabu[i+2];} tabusk -= 2;
      ar = ArYraNeTabu(geresne, max);
    }
//kaimynu sudarymas
     while(aplN > 0 && itlaikas < laikas)
     { asp = -1;
       int as = asp;
       darbine = geresne;
       darbine = sukeistiKrit(darbine, mk, asp, gen1);
       abc = ivertinti(darbine);
       variantai += 1;
        bool aspir = false;
```

```
    if(abc.ms < globalus && as - asp !=0) {aspir = true; }
    if(as - asp == 0 || aspir)
   { pos[matN] = mk; mat[matN] = abc.ms; aspi[matN] = as - asp; matN +=1;}
 }

 chrom galutine = geresne;
 aplElementai(geresne);
 int opt = aplN;
 // Geriausias ne tabu sukeitimas
 double iesk;
 int ieskV;
 for(int i = 0; i < matN-1; i++)
    for(int j = i + 1; j < matN; j++)
    { if(mat[i] > mat[j])
      {iesk = mat[i]; mat[i] = mat[j]; mat[j] = iesk;
       ieskV = pos[i]; pos[i] = pos[j]; pos[j] = ieskV;
       ieskV = aspi[i]; aspi[i] = aspi[j]; aspi[j] = ieskV;
       }
    }
 int TS = 0;
 bool ras = false;
 double mini = mat[0];
 for(int i = 1; i < matN && !ras; i++)
 {if(mat[i]== mat[0]) TS = i; else ras = true;}
  int keis = random(TS+1);
 chrom sena = geresne;
 aplElementai(geresne);
 if(!ciklas)
 {geresne = sukeistiKritTa(geresne, pos[0]);
 }
 else
 {  abc = ivertinti(geresne);
    KritinisKelias(abc);
    aplElementai(geresne);
    geresne = N1(geresne);
 }

abc = ivertinti(geresne);
KritinisKelias(abc);
aplElementai(geresne);
variantai += 1;
//elitiniai sprendiniai
double atsk = T();
double atsk2 = T();

if(abc.ms < globalus || (exp((globalus-abc.ms))/temp> atsk && elitN<maxkiekis ))
{
if(!ArYraElite(geresne))
{ for(int i = elitN-1 ; i > -1;i--)
  elite[i+1] = elite[i];

 elite[0] = geresne;
 if(elitN < 50)
 elitN +=1;

 fg << setw(5)<< elitN <<setw(5)<< abc.ms << setw(15)<< endl;

 }
}
temp = globalus/temp0;

 // ciklu nustatymas
 for(int i = ciklN - 1; i > -1;i--)
  cikl[i+1] = cikl[i];
cikl[0] = abc.ms;
if(ciklN < 400)
ciklN +=1;
ciklas = false;
if(ciklN > 10)
{for(int i = 5; i < 199 && ciklas == false; i++)
 {double kor = autokor(cikl, ciklN, i);
  if(kor == 1) ciklas = true;
 }
}

if(ciklas){fl << abc.ms <<" ciklas" << endl;}
```

```
   else{fl << abc.ms<< endl;}

   if(j-ats > iter)
   { ats = j;
     if(elitN > 0)
     {geresne = elite[0];
      for(int i = 0; i < elitN-1;i++)
       elite[i] = elite[i+1];
       elitN -=1;
             }
       else
       { geresne = geriausia;
       }
       tabusk = 0;
       temp = globalus/temp0;
       abc = ivertinti(geresne);
       KritinisKelias(abc);
       aplElementai(geresne);
       variantai += 1;
      fg << setw(5)<< elitN <<setw(5)<< abc.ms << endl;

   }
   min = abc.ms;
   if(abc.ms < globalus)
   {geriausia = geresne; globalus = abc.ms; ats = j;
    if(elitN> 3) elitN = 3;
   }
   itlaikas = trukme();
   double IKI = skaitliukas + intervalas;
   double dab = itlaikas;
   if(skaitliukas <dab && dab < IKI )
      {fr << globalus<< endl;
      //fl << itlaikas << endl;
       skaitliukas += 1;
       Series1->AddXY(itlaikas, globalus , E, clRed);
      }
  }
 fl << "Nagrineta sprendiniu: " << variantai << endl;
 glob = globalus;
 Label6->Caption = "Tvarkaraščio darbų trukmė: " + FloatToStr(globalus);
 tvarkarastis c = SpausdintiTvar(geriausia);
 Memo1->Lines->LoadFromFile("tvarkarastis.txt");
 fr.close();
 fl.close();
}
```

void TForm1:: Paieska() //paieška kintamose aplinkose

```
{ ofstream fl;
  ofstream fr;
  fl.open("laikai.txt");
  fr.open("reiksmes.txt", ios::app);
  AnsiString E;
  Series2->Clear();
  chrom darbine;
  chrom geresne, geriausia;
  generuoti(0,1);
  int maxgen = StrToInt(Edit2->Text);
  int laikas = StrToInt(Edit6->Text);
  double itlaikas = 0;
  double skaitliukas = 0;
  double intervalas = 0.5;
  struct  time t;
  gettime(&t);
  pradval = t.ti_hour;
  pradmin = t.ti_min;
  pradsek = t.ti_sec;
  pradsim = t.ti_hund;
  int KK = StrToInt(Edit3->Text); // parametras K
  int sh = StrToInt(Edit5->Text); //sužadinimų skaičius
  geriausia = pop[0];
  geresne = geriausia;
  abc = ivertinti(geresne);
  KritinisKelias(abc);

  double globalus = abc.ms;
```

```
  double lokalus;

  for(int j = 0; j <maxgen && itlaikas < laikas; j++)
  {
    double min = abc.ms;
     // lokalioji paieska
    aplinka = 0;
    lok = false;
     while(!lok && itlaikas < laikas)
     { darbine = geresne;
       darbine = sukeistiKrit2(darbine, aplinka, gen1, gen2);
       abc = ivertinti(darbine);
       lokalus = abc.ms;
     if(lokalus < min)
     {min = lokalus;
      geresne = darbine;
      KritinisKelias(abc);
      aplinka = 0;
     } else if(min == lokalus  &&  (irenginiai[kritIr] < globalus || T()<0.5) )
      { geresne = darbine;
        KritinisKelias(abc);
      }
    }
    chrom galutine = geresne;
    // sužadinimas

 if(j%KK==0)
 { for(int i = 0; i < sh; i++)
   {geresne = sukeistiKrit2(geresne, aplinka, gen1, gen2);
    abc = ivertinti(geresne);
    KritinisKelias(abc);
   }
  }
     itlaikas = trukme();
     double IKI = skaitliukas + intervalas;
     double dab = itlaikas;
     if(skaitliukas <dab && dab < IKI )
        {
        fr << globalus<< endl;
        fl << itlaikas << endl;
        skaitliukas += 1;
        Series2->AddXY(itlaikas, globalus , E, clRed);
        }

     if(min < globalus) {geriausia = galutine; globalus = min;}

  }

 Label6->Caption = "Tvarkaraščio darbų trukmė: " + FloatToStr(globalus);
 glob = globalus;
 tvarkarastis c = SpausdintiTvar(geriausia);
 Memo1->Lines->LoadFromFile("tvarkarastis.txt");
 fr.close();
 fl.close();
}
```