KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Vilimantas Milevičius

Dėžių pakavimo su papildomu apribojimu optimizavimo algoritmo sudarymas ir tyrimas

Magistro darbas

Darbo vadovas:
prof. R. Butleris

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Vilimantas Milevičius

Dėžių pakavimo su papildomu apribojimu optimizavimo algoritmo sudarymas ir tyrimas

Magistro darbas

Recenzentas:
dr. D. Makackas
2007-05-28

Darbo vadovas:
prof. R. Butleris
2007-05-28

Atliko:
IFM–1/1 gr. stud.
Vilimantas Milevičius
2007-05-28

Kaunas, 2007

TURINYS

LENTELĖS

PAVEIKSLAI

SUMMARY

Creation and research of the 3D bin packing optimization algorithm with additional restriction

Presented work covers one of the most complex areas of combinatorial optimization – three dimensional bin packing problem. Solution methods of this problem are applied in the real world from logistics, packing optimization to VLSI circuit and automobile engineering. Several heuristic packing algorithms suggested by other authors are analyzed. Approach based on tree-search and wall building strategy is chosen to create a 3D packing optimization algorithm.

A bin orientation in space restriction is added to classical 3D bin packing problem to make it more complex and more suited for real world applications.

A prototype of created algorithm is created and tested with randomly generated data collections. Each data sample is processed with and without bin orientation in space restriction. Influence of restriction and maximal tree width on packing efficiency and computational time is statistically analyzed. Visualization tool based on Microsoft Direct X technology is created to view results of packing optimization.

1. ĮVADAS

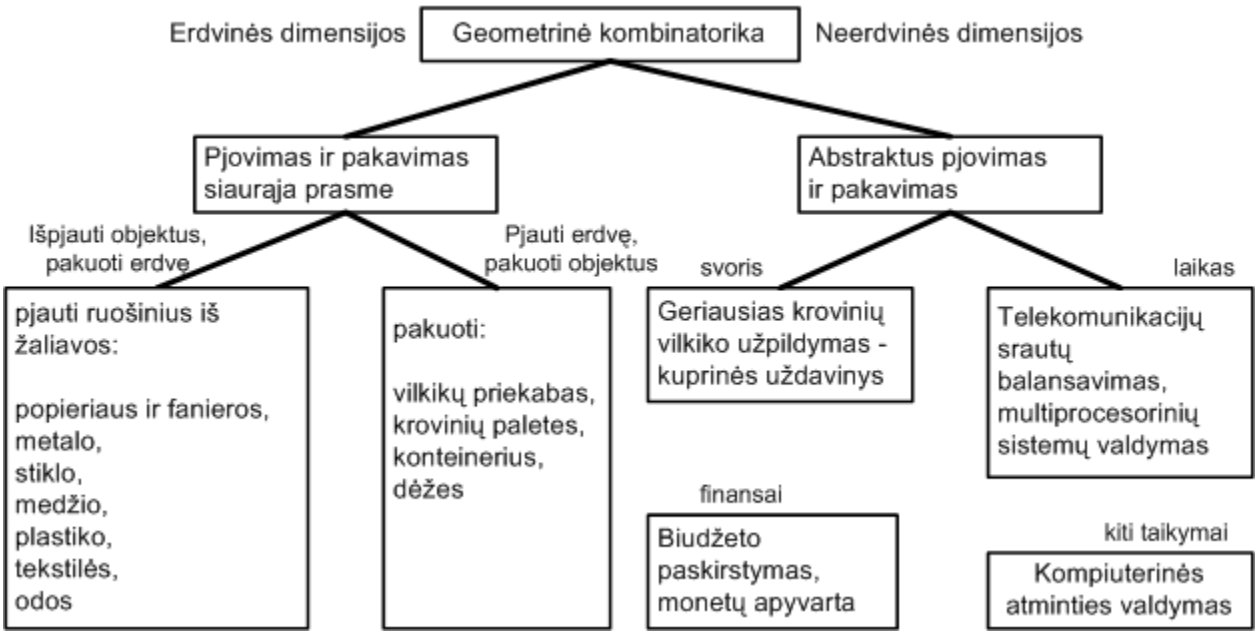
Informacinės technologijos – viena iš sparčiausiai pastaruosius kelis dešimtmečius besivystančių mokslo ir pramonės šakų. Nors jos intensyviai plėtojamos, tačiau dėžių pakavimas (*bin-packing problem*), pjovimo (*cutting stock problem*), trimačių dėžių pakavimas (*3D bin packing problem*) lieka sudėtingais klasikiniais uždaviniais. Informatikos ir diskrečiosios matematikos mokslininkai analizavo ir sprendė šiuos uždavinius jau pora dešimtmečių, tačiau dar nėra sukurtas algoritmas, kuris rastų visiškai optimalų sprendinį per priimtiną skaičiavimo laiką. Tačiau niekas dar nepaneigė tokio sprendinio egzistavimo galimybės. Teoretikai dažniausiai priskiria šį uždavinį prie potencialiai „neišsprendžiamų“ uždavinių, kurie priklauso kombinatorinių NP – sudėtingų matematinių problemų aibei. Šiuo metu negalima gauti optimalaus sprendinio, beveik neperrenkant visų įmanomų pakavimo uždavinio sprendinių. Kitaip tariant, netrivialų dėžių pakavimo uždavinį patys galingiausi kompiuteriai išspręstų tik per kelis mėnesius ar netgi metus.

Kol yra tyrinėjama optimizavimo teorija, pramonėje ir logistikoje jau diegiami šio uždavinio sprendimo būdai, kurie kartais būna visiškai neoptimalūs. Informatikos ir kombinatorinės matematikos mokslininkai sukūrė euristinių algoritmų dėžių pakavimo optimizavimo uždaviniui. Tad daugeliu atvejų kompiuteris gali rasti pakankamai gerą sprendinį per priimtiną laiko tarpą ir toks sprendinys gali būti vertingas pritaikymui. Dėžių pakavimo optimizavimo uždavinys yra pritaikomas įvairiose gyvenimiškose srityse, pradedant televizijos tinklelio sukūrimu ir baigiant automobilių ar lėktuvų konstravimu.

Šio darbo tikslas – sukurti trimačių dėžių su papildomu orientacijos erdvėje apribojimu pakavimo optimizavimo algoritmą, o realizavus jį programinėmis priemonėmis – ištirti jo efektyvumą su atsitiktinai sugeneruotų kraunamų dėžių rinkiniais ir prie skirtingų algoritmo veikimo parametrų. Taip pat sukurti ir pakavimo optimizavimo sprendinio vizualizavimo trimatėje erdvėje programinę įrangą.

2. DĖŽIŲ PAKAVIMO UŽDAVINIO ANALIZĖ

Smulkesnių objektų pakavimas į stambesnius objektus sudaro vieną svarbiausių pjovimo ir pakavimo problemos (*C&P*) sudedamųjų dalių. Šis uždavinys galima sakyti priklauso geometrinės kombinatorikos uždavinių sričiai. Šių uždavinių apimamos srities struktūra pavaizduota *1 pav*. Ši sritis apima daug tarpusavyje glaudžiai susijusių uždavinių. Pavyzdžiui, dėžių pakavimo problema (*BPP*), kurio tikslas yra minimizuoti objektų kiekį, kurie turi būti supakuoti į žinomos talpos dėžes, gali būti traktuojamas kaip vienos dimensijos pjovimo optimizavimo uždavinys (*one-dimensional CSP*). Vienintelis skirtumas tarp šių dviejų problemų tipų yra pakuojamų objektų aibė: BPP paprastai turi daug objektų, turinčių skirtingus ilgius, kai tuo tarpu CSP tie ilgiai būna identiški daugumai objektų (Karelahti, 2002:5).



1 pav. Pjovimo ir pakavimo uždavinių fenomenologija (Dyckhof, 1990)

Klasikinis pakavimo optimizavimo uždavinys yra vienos dimensijos dėžių pakavimo uždavinys (*bin packing problem*): baigtinis objektų rinkinys, kurių skirtingi ilgiai yra pakuojami į vieną ar kelis konteinerius. Kiekvienas konteineris gali turėti bet kokį objektų rinkinio poaibį, tačiau negali viršyti jo talpos. Kiekvienas objektas yra dedamas į konteinerį ar konteinerius, taigi lieka vis mažiau laisvos vietos. Šis uždavinys gali būti traktuojamas kaip pasirinkimo arba optimizavimo uždavinys. Kaip pasirinkimo uždavinys, tai turi būti nustatoma: ar visi objektai telpa į dėžes? Kitais žodžiais tariant, pasirinkimo uždavinys pabrėžia ar yra pakankamai laisvos

vietos, kad būtų galima patalpinti objektą ar ne. Ir atvirkščiai, optimizavimo uždavinio sprendimu stengiamasi minimizuoti panaudojamų konteinerių kiekį, arba minimizuoti kraunamo konteinerio neišnaudotos talpos nuostolius. Šiuo uždavinio formulavimu stengiamasi supakuoti dėžes į konteinerius efektyviausiai. Jeigu pakavimo uždavinį traktuoti kaip pasirinkimo, tai atsakymas yra iš Būlio algebros, taigi žymiai paprastesnis. Ir atvirkščiai, optimizavimo uždavinio sprendimo atsakymas yra skaitinis. Matematikos ir informatikos mokslininkai laiko pasirinkimo uždavinį NP – baigtiniu, o optimizavimo uždavinį – NP – sudėtingu (Sweep, 2004: 2, 3).

Vienmačio dėžių pakavimo (*1D-BPP*) optimizavimo **[3]** uždavinio formuluotė yra tokia pati kaip ir vienmačio pjovimo optimizavimo (*CSP*) uždavinio:

- Neapibrėžtas vienos dimensijos konteinerių (žaliavos ruošinių), kurių ilgis W, kiekis.
- Supakuoti (išpjauti) m dėžių (detalių), kurių tipas i, kiekviena kurių turi plotį w_i ir kiekį (poreikį) b_i.
- Supakuoti visas dėžės (išpjauti detales), panaudojant kuo mažesnį kiekį konteinerių (žaliavos ruošinių).

Matematinė vienmačio dėžių pakavimo formuluotė **[3]**:

$$\begin{cases} \sum_{i=1}^{n} y_i \to \min \\ \sum_{j=1}^{n} w_i x_{ij} \le W_{y_i}, \quad i \in N \\ \sum_{i=1}^{n} x_{ij} = 1, \qquad j \in N \\ x_{ij}, y_j \in \{0,1\}, \quad i, j \in N. \end{cases}$$

kur:

y_i – požymis, ar konteineris i panaudotas sprendinyje;

x_{ij} – požymis, ar dėžė j patalpinta konteineryje i;

w_i – dėžės i plotis

2.1. Dėžių pakavimo uždavinys dvimatėje erdvėje

Išplėtus klasikinį dėžių pakavimo uždavinį iš vienmatės į dvimatę erdvę, jis tampa žymiai sudėtingesnis. Kiekvienas objektas vienmatėje dėžių pakavimo erdvėje turėjo fiksuotą plotį ir kintamą ilgį arba atvirkščiai. Dvimatėje uždavinio versijoje, abu parametrai yra kintamieji. Dvimatis dėžių pakavimo uždavinys gali būtį įsivaizduojamas kaip stačiakampių dėliojimas ant plokščių grindų. Stačiakampių pakavimas tolygus įvairaus dydžio grindų plokščių efektyviam išdėstymui didesnėje stačiakampėje grindų erdvėje. Klasikinis dvimatis pakavimas reikalauja,

kad kiekvienas objektas būtų ortogonalus, taigi objektai negali būti sukinėjami erdvėje (Sweep, 2004: 3). Šio uždavinio formuluotė **[3, 8]**:

- Aibė *R*, susidedanti iš *n* stačiakampių, kurių pločiai w_j ir ilgiai h_j.
- Konteineris, kurio plotis *H* ir ilgis *W*.
- Sudėlioti stačiakampius ant konteinerio taip, kad būtų maksimizuojama vertės funkcija, šiuo atveju minimizuojant nepanaudoto ploto nuostolius.

Tikrinimas ar dėžės telpa į konteinerį vienmačiu atveju užima tiesinį laiko tarpą, tačiau uždavinys, ar dėžė telpa į konteinerį dvimačiu atveju yra NP – sunkus **[2]**.

Fraunhorferio Algoritmų ir Skaičiavimų Institute (SCAI) yra sukurta sudėtinių dvimačio neortogonalaus pakavimo optimizavimo algoritmų siuvyklos žaliavos sukirpimo optimizavimui („AutoNester – T“), taip pat odos pramonei („AutoNester – L“). Dvimačio pakavimo ortogonalaus pakavimo algoritmai panaudoti labai didelės integracijos schemų projektavime (VLSI routing) **[10]**.

2.2. Dėžių pakavimo uždavinys trimatėje erdvėje

Trimatis dėžių pakavimo uždavinys yra dar sudėtingesnis negu mažiau dimensijų turinčių dėžių uždaviniai, tačiau turi unikalių ir svarbių pritaikymų. Kiekvienas objektas ir dėžė turi tris dimensijas. Kiekviena objektą ar dėžę nusako tripletai iš aukščio, pločio ir ilgio. Kaip ir dvimačiame dėžių pakavime, kiekviena dėžė turi būti ortogonali, arba išlaikyti savo orientaciją pakavimo metu. Jeigu dvimatis dėžių pakavimas atitinka stačiakampių ant grindų pakavimą, tai trimatis – dėžių kambaryje pakavimą (Sweep, 2004: 3).

Šio pakavimo uždavinio (3D-BPP) tikslas – ortogonaliai supakuoti visus objektus į minimalų konteinerių kiekį. 3D-BPP yra labai NP- sunkus uždavinys. Bet, tai yra ir vienos dimensijos dėžių pakavimo uždavinio (1D-BPP) išplėtimas. 1D-BPP yra 3D-BPP atskiras atvejis – tiesiog dėžių pločiai ir aukščiai yra lygūs konteinerio pločiui ir aukščiui **[6]**.

Fraunhorferio Algoritmų ir Skaičiavimų Institutas (SCAI) taiko trimačio pakavimo optimizavimo algoritmus tiriant molekulinę biologiją ir chemiją, taip pat bendradarbiaujant su automobilių gamintojais „Mercedes-Benz“, „Audi“ sukūrė trimačių pakavimo aproksimavimo algoritmų, išplėtotų iš dvimačių VLSI schemų projektavimo algoritmų, kurie panaudoti konstruojant automobilius efektyviam vidinių komponentų išdėliojimui **[2, 10]**.

2.3. 3D pakavimo uždavinio sprendimo algoritmai

Trimatis pakavimo uždavinys **[2]** turi panašumų su dvimačiu pakavimo uždaviniu, tik yra sudėtingesnis. Taigi, dauguma sukurtų algoritmų, skirtų trimačiam pakavimo uždaviniui spręsti, yra paremti euristiniais ar metaeuristiniais algoritmais, kurie buvo skirti vienmačiams ar dvimačiams pjovimo ir pakavimo uždaviniams spręsti.

Yra išskiriamos trys skirtingos pakavimo optimizavimo uždavinių sprendimo metodų klasės **[4]**:

- *Algoritminiai* metodai garantuoja, kad bus surastas optimalusis sprendinys. Tačiau didelis šių metodų trūkumas yra skaičiavimų sudėtingumas, todėl yra labai ilga sprendinio ieškojimo trukmė, ypač didelės apimties uždaviniams. Praktikoje jie nėra plačiai taikomi.
- *Euristiniai (Heuristic)* metodai retai randa optimalųjį sprendinį, bet paprastai gana greitai randa priimtiną rezultatą, kuris yra pakankamai arti optimaliojo sprendinio. Trūkumas yra tas, kad *euristiniai* metodai yra labai siauros specializacijos – tinka spręsti tik tiems uždaviniams, kuriems jie yra sukurti ir juos labai sunku pritaikyti šiek tiek besiskiriantiems uždaviniams. Taip pat nežinoma kokia yra sprendinio kokybė.
- *Aproksimavimo* metodai yra tokie, kurių grąžinamas sprendinys yra gaunamas aproksimuojant optimalųjį sprendinį. Tad skiriasi tam tikru, nedideliu skaičiumi.

Nors euristiniai metodai ir negarantuoja jokio pakankamai optimalaus sprendinio, tačiau yra žymiai greitesni negu algoritminiai, taip pat jų naudojamas euristikas yra lengviau modifikuoti atsiradus papildomiems apribojimams.

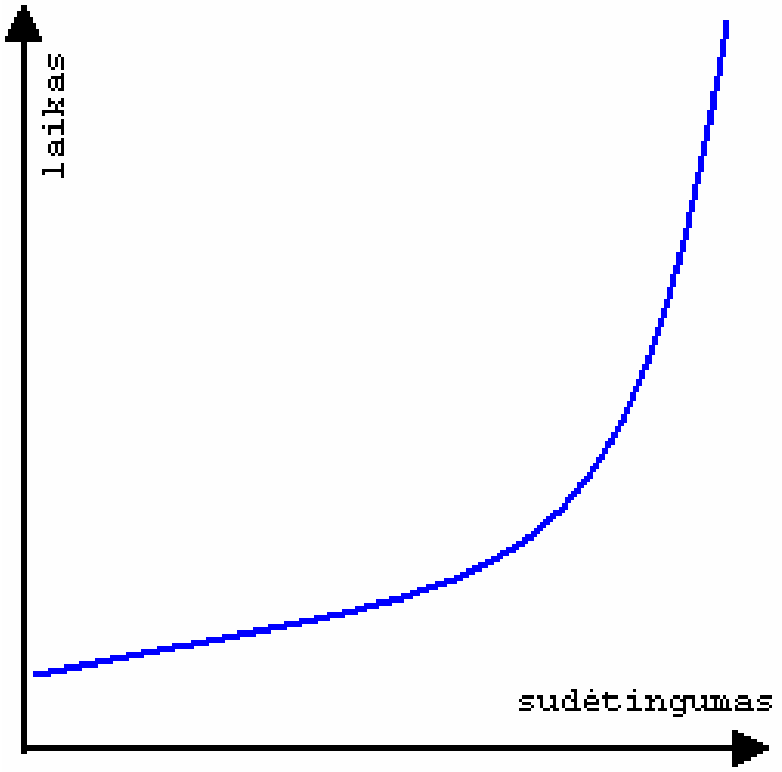
Pačios euristikos, ir apskritai NP – sudėtingiems uždaviniams spręsti skirtos bendrosios euristikos, yra skirstomos į dvi stambias grupes **[4]**:

- *Konstravimo euristikos (constructive heuristics)* metodai gauną sprendinį, pridedant po vieną papildomą objektą.
- *Lokalios paieškos euristikos (local search heuristics)* metodai pakeisdami mažas jau sudaryto neoptimalaus sprendinio dalis gauną pakankamai optimalų sprendimą.

Matematinėje literatūroje **[2]** dažniausiai sutinkami vienmačio dėžių pakavimo algoritmai. Buvo nustatyta, kad šis uždavinys yra NP – pilnas. Perfrazuojant uždavinį į trimatę erdvę, jis taps žymiai sudėtingesnis negu vienmatis, bet vis dėlto jį bus įmanoma išspręsti per polinominį laiko

tarpą nedeterminuotomis skaičiavimo mašinomis. Iš čia seka, kad trimatis uždavinys irgi yra NP – pilnas.

Deja, visi žinomi algoritmai, kurie pilnai išsprendžia NP-pilnumo uždavinius, turi eksponentinę laiko priklausomybę, kuri yra ekvivalentiška pilno perrinkimo laikui **[8]** (*2 pav.*). Tai reiškia, kad pagal bet kurį tokį algoritmą, netgi mažo krovinio (apie 50 dežių) pakavimo optimizavimas, atliekant skaičiavimus su superkompiuteriu, užtruktų ne vienerius metus.



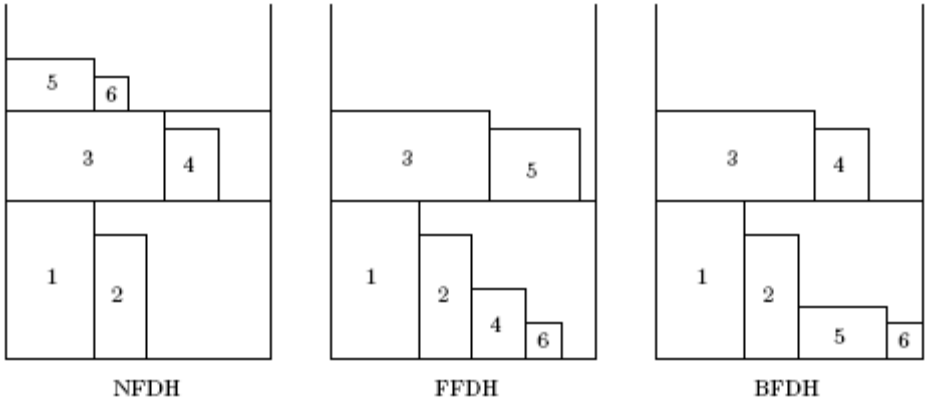
2 pav. NP sudėtingo uždavinio sprendimo laiko ir sudėtingumo priklausomybė

Vienas iš būdų išspręsti šią problemą galėtų būti euristinio algoritmo panaudojimas, kuris duotų rezultatą, artimą optimaliam sprendimui, per priimtiną laiką. Iš tokio algoritmo būtų galima tikėtis per keletą minučių pasiekti vidutinį 80-90% pakrovimo tankį **[1]**.

3. EURISTINIAI DĖŽIŲ PAKAVIMO ALGORITMAI

Trys klasikinės dvimačių dėžių pakavimo sluoksniais euristikos yra išplėtotos nuo vienmačių dėžių pakavimo algoritmų. Kiekvienu atveju (*3 pav.*) pakuojami objektai yra surikiuojami nemažėjančio aukščio tvarka ir yra supakuojamos atitinkama tvarka. Tegul j – pakuojamas objektas ir s – paskutinis sukurtas sluoksnis **[5]**:

- „Kitas telpantis, mažėjančio aukščio“ (*Next – Fit Decreasing Height* (NFDH)) strategija: objektas *j* yra pakuojamas sluoksnyje kuo arčiau kairės sienos, jei jis ten telpa. Kitu atveju sukuriamas naujas sluoksnis (*s = s + 1*) ir objektas *j* supakuojamas į kairįjį kampą.
- „Pirmas telpantis, mažėjančio aukščio“ (*First-Fit Decreasing Height* (FFDH)) strategija: objektas *j* pakuojamas į pirmąjį sluoksnį kuo arčiau kairės sienos, kuriame dar dėžė telpa. Jei nėra nė vieno tokio sluoksnio – kuriamas naujas sluoksnis kaip ir NFDH.
- „Geriausiai telpantis, mažėjančio aukščio“ (*Best-Fit Decreasing Height* (BFDH)) stretegija: objektas *j* yra pakuojamas tokiame sluoksnyje kuo arčiau kairės sienos, kuriame nepanaudotas horizontalus plotis – minimalus. Jei į nė vienas sluoksnį netelpa j, tai sukuriamas naujas kaip ir NFDH.



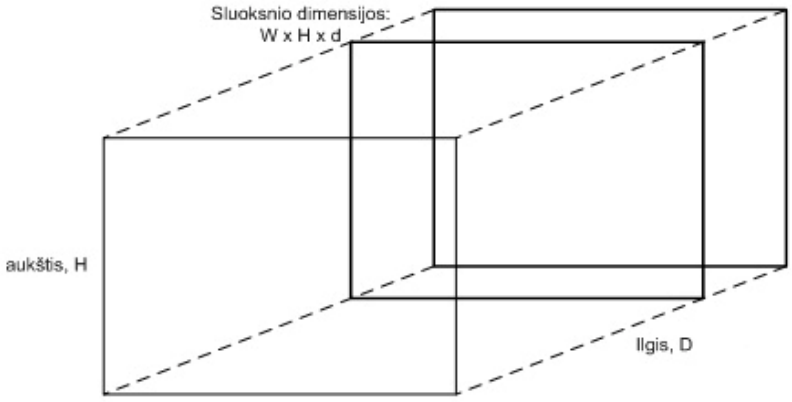
3 pav. Trys dvimačių dėžių pakavimo strategijos.

Šias euristines dvimačių dėžių pakavimo strategijas galima pritaikyti ir trimačiam atvejui, jeigu vienu metu yra sprendžiama tik dvimatis uždavinys, trečią koordinatę užfiksavus kaip nekintamą.

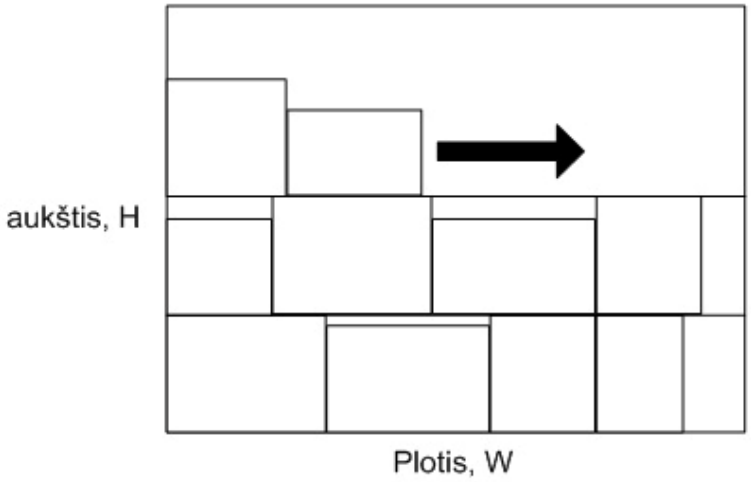
3.1. George ir Robinson sluoksniavimo algoritmas

1980 metais *J. A. George* ir *D. F. Robinson* **[3, 4, 7]** pasiūlė trimačių dėžių pakavimo algoritmą sluoksniuojant (*4 pav.*). Visoms pakuojamoms dėžėms yra surandama minimali dimensija, ir pagal ją visi dėžių tipai yra surikiuojami mažėjimo tvarka. Pradedama dėlioti nuo tų dėžių, kurių minimali dimensija – didžiausia, nes tokias dėžes yra sunkiausia patalpinti. Uždavinys tampa dvimatis – dėžės reikia dėlioti dvimatėje erdvėje (*5 pav.*), nes trečioji

dimensija yra dėžės minimali dimensija. Dėžės pakuojamos formuojant lygių sluoksnius dvimatėje erdvėje minimizuojant godžios funkcijos (*greedy strategy*) reikšmę. Pagal *David Pisinger* **[3]** šis algoritmas pasižymi dideliu greičiu, užpildymo efektyvumas – virš 80%. Tačiau galima naudoti ir kitas sluoksnio pločio parinkimo strategijas – 1990 metais *E. E. Bischoff* ir *M. D. Marriott* tyrė George ir Robinson sluoksniavimo algoritmą, taikydami 14 skirtingų sluoksnio storio parinkimo strategijų ir nė viena jų nebuvo geresnė už kitas **[4]**.



4 pav. Sluoksnio sudarymas kraunamajame konteineryje.



5 pav. Sluoksnio pildymas dėžėmis – dvimačio dėžių pakavimo uždavinys.

3.1.1. Sluoksniavimo algoritmo realizacija

Dėžių pakavimas į konteinerį sluoksniuojant beveik atitinka realiai naudojamą prekių krovimą – prekės kraunamos iki viršaus siena, ant konteinerio grindų naujajame sluoksnyje dėžės imamos dėlioti tik tada, kai prieš tai buvusiame sluoksnyje viršuje nebelieka vietos.

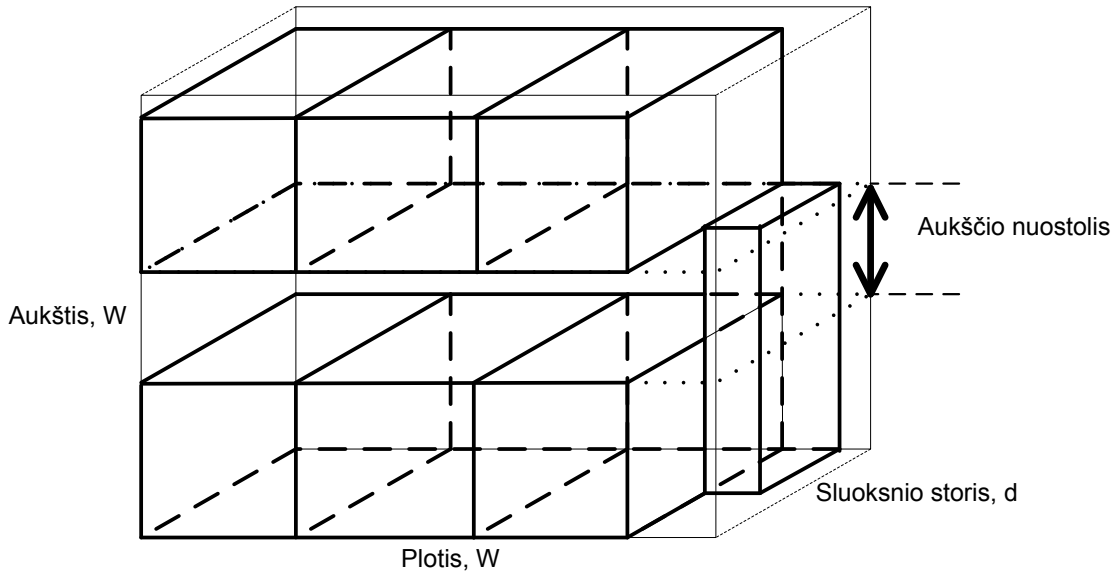
1 lentelė Eksperimentiniai *George* ir *Robinson* algoritmo rezultatai

Nr.	Dėžių rūšių kiekis	Dėžių užimamas tūris	Sluoksnių užimamas tūris	Efektyvumas
1	3	77.064	114.180	67.494 %
2	6	48.672	62.760	77.553 %
3	12	60.684	69.144	87.765 %
4	12	90.921	106.908	85.045 %
5	12	187.514	204.000	91.919 %
6	12	184.977	203.796	90.766 %

Iš eksperimentinio algoritmo realizacijos rezultatų pastebime – kuo didesnis dėžių rūšių skaičius, yra tendencija, kad jos bus erdvėje efektyviau supakuotos. Tai savaime suprantama – kuo didesnė dėžių dimensijų įvairovė, tuo daugiau galimybių parinkti dėžę taip, kad erdvės nuostoliai būtų mažesni. Algoritmo skaičiavimo laikas nebuvo matuojamas, nes algoritmo veikimo greitis yra labai didelis, todėl užfiksuotas skaičiavimo laikas praktiškai nesiskyrė visiems duomenų rinkiniams.

3.1.2. Sluoksniavimo algoritmo trūkumas

Dėžės pradedamos pakuoti nuo tolimiausio kairiojo sluoksnio krašto. Jei lieka tarpas tarp sluoksnio sienelės ir dėžių, tačiau į jį einamasis dėžės netelpa, bandoma ten patalpinti kito tipo dėžę. Kai daugiau į einamąjį dėžių lygį nebetelpa, yra kuriamas naujas dėžių lygis.



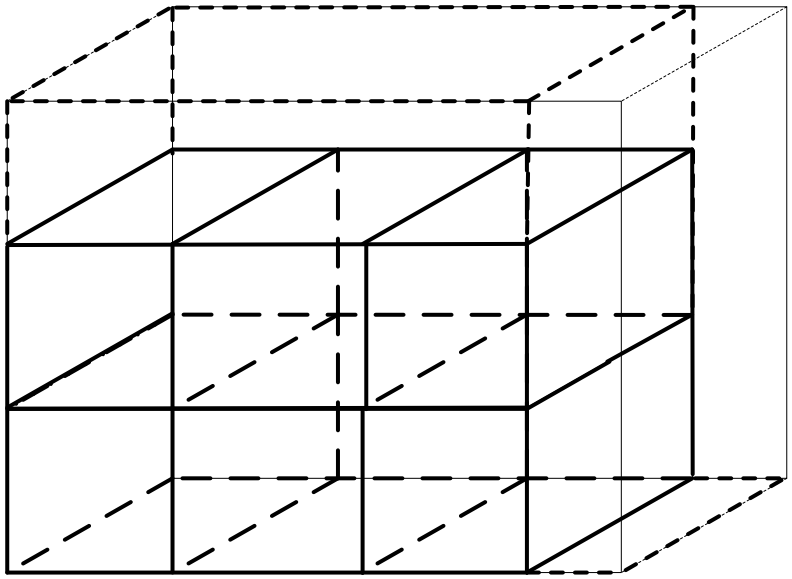
6 pav. George ir *Robinson* algoritmo trūkumas.

Vizualizavus eksperimentinius *George* ir *Robinson* algoritmo rezultatus (*1 lentelė*) buvo pastebėta, kad jei į plyšį telpa aukštesnė nei pakuojamas dėžių lygis objektas – ten jis ir

talpinamas (*6 pav.*). Naujas sluoksnis jau yra aukštesnio objekto lygyje, todėl atsiranda tarpas tarp dėžių sluoksnių. Taip krauti dėžių negalima, nes dėžės negali būti „pakibusios ore“ – visos turi būti padėtos ant tvirto pagrindo. Be to taip yra prarandama nemaža formuojamo sluoksnio erdvės dalis.

3.1.3. Galimas sluoksniavimo algoritmo realizacijos patobulinimas

Kai dėžė nebetelpa į plyšį tarp sluoksnio sienelės ir ką tik padėtos dėžės, nebūtina iškart stengtis jį užpildyti kitų tipų dėžėmis. Šį plyšį galima būtų pratęsti iki pat sluoksnio viršaus ir palikti vėlesniam užpildymui (*7 pav.*, punktyrų ribojama erdvė). Nekeičiant dėžės tipo, kurti naujus dėžių sluoksnius, iki jos nebetilps sluoksnyje. Kai dėžės nebetelpa vertikaliai, likusiai neužpildytai erdvei (*7 pav.*, punktyrų ribojama erdvė) galima taip pat būtų taikyti algoritmą, kaip ir plyšiui aukščiau.



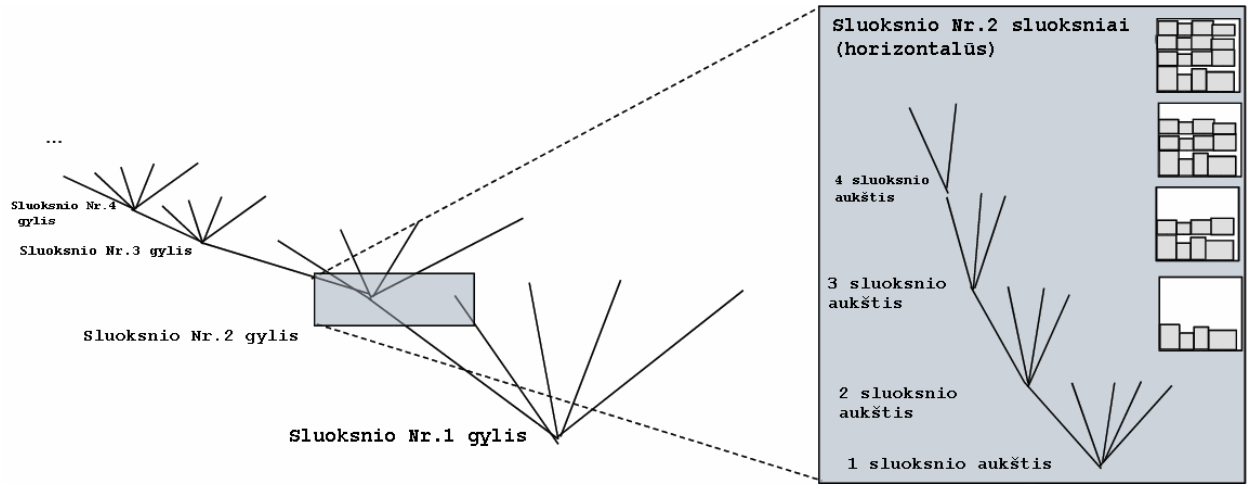
7 pav. Kitoks sluoksnio užpildymas.

Tokiu būdų galima būtų sutaupyti nemažai erdvės, nes aukštesnės dėžės nesukuria aukštesnio nei einamasis sluoksnis, dėžės dedamos viena ant kitos, todėl sprendinys turėtų būti korektiškas ir efektyvesnis. Šios euristikos žingsniai pateikti **[7]**.

3.2. David Pisinger pasiūlytas patobulintas algoritmas

David Pisinger 2002 metais pasiūlė patobulintą sluoksniavimu pagrįstą algoritmą **[4]**. Norint grįžti pakavimo procese atgal, t. y. nuimti jau supakuotą dėžę – reikia registruoti patį pakavimo procesą. Tai yra reikalinga, kad būtų galima sugeneruoti daugiau algoritmo žingsnio šakų. Absoliučiai visų šakų perrinkti yra neįmanoma, nes ribos yra per silpnos šakų ir ribų algoritmui. Tačiau medžio paieškos algoritmas naudojamas atrinkti m (m_1 – geriausių sluoksnių pločių ir m_2 – geriausių dėžių juostų pločių) geriausią pakavimo rezultatą (mažiausius nuostolius) žadančių šakų, kurios bus panaudojamos tolimesniame algoritmo skaičiavime – likusios sugeneruotos šakos atmetamos kaip neperspektyvios.

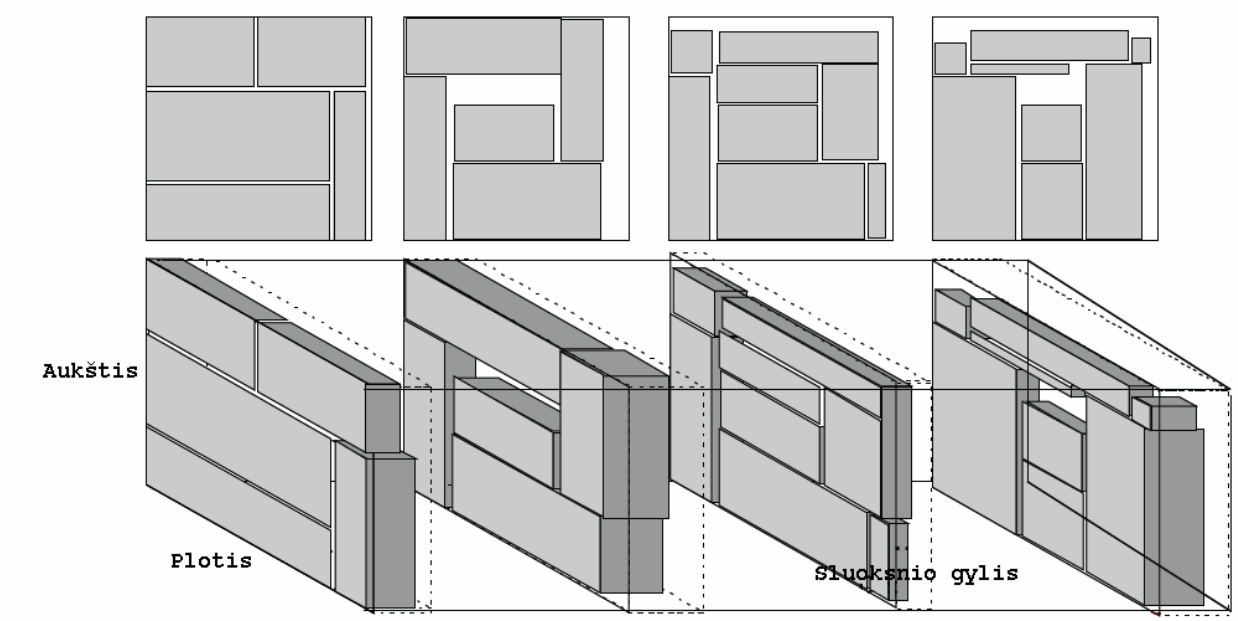
Sluoksnio storis parenkamas jau nebe vien pagal didžiausią iš visų dėžių mažiausių dimensijų. Parenkami 4 kandidatai pagal nesupakuotų dėžių statistinę informaciją – įvertinamas ir dėžės tipo pasikartojimo dažnis tarp dar nesudėliotų dėžių.



8 pav. *David Pisinger* pasiūlyto algoritmo medis.

Sluoksnis pildomas dėžėmis vertikaliais ir horizontaliais rėžiais (*9 pav.*), sprendžiant kuprinės turinio maksimizavimo (*knapsack packing*) uždavinį, kuriame kuprinės turinio nauda yra atvirkščiai proporcinga nepanaudotam konteinerio tūriui. Rėžiai gali būti kaitaliojami. Taip pat dėžes galima grupuoti ir interpretuoti kaip vieną objektą. Kiekviename žingsnyje yra pasirenkami 8 kandidatai naujo vertikalaus rėžio pločiui pagal dimensijos pasikartojimo rėžį.

Pagal *David Pisinger* šio algoritmo efektyvumas – vidutiniškai apie 91%.



9 pav. Konteinerių pildymas sluoksniais – vertikaliais ir horizontaliais rėžiais.

3.3. Sluoksniavimo algoritmų apibendrinimas

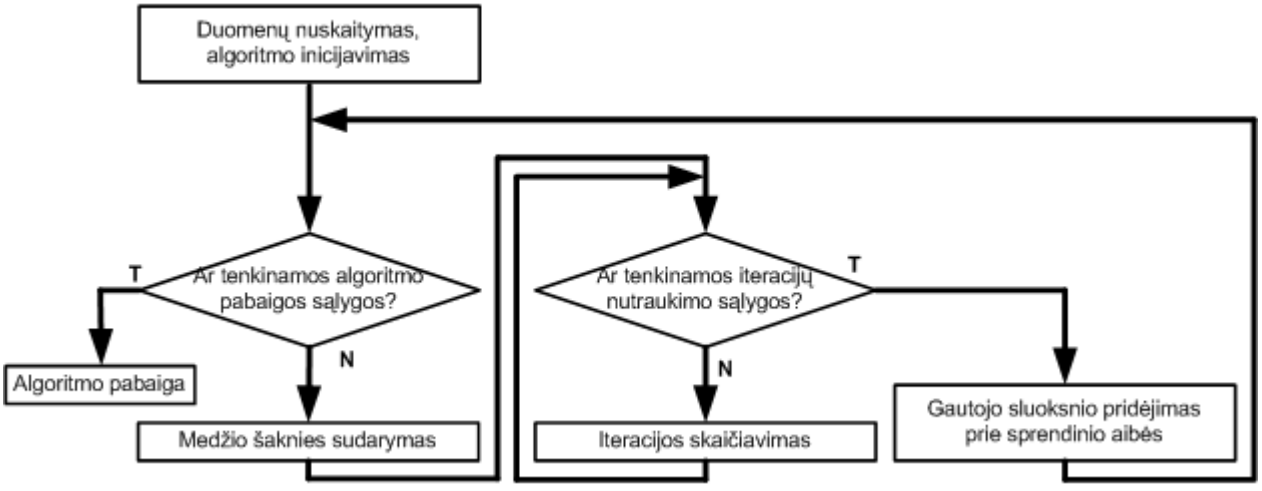
George ir *Robinson* pasiūlytas algoritme stengiamasi sumažinti uždavinio sudėtingumo laipsnį sumažinant dimensijų kiekį – fiksavus sluoksnio plotį, sprendžiamas jau nebe trimatis, o dvimatis optimizavimo uždavinys. Tačiau pati sluoksnio pildymo dėžėmis euristika turi trūkumą – kai kurios dėžės gali nesiremti į kitas (*6 pav*.). Formuojant kiekvieną dėžių sluoksnį yra parenkamas tik vienas sluoksnio plotis, remiantis tuo, kad dėžę su didžiausia iš visų dėžių mažiausia dimensija yra sunkiausia patalpinti jau formuojamame sluoksnyje. Pats euristinis algoritmas yra labai greitas, prototipo testavimo metu nepavyko užfiksuoti skaičiavimo trukmės skirtumų.

David Pissinger pasiūlytas algoritmas remiasi *George* ir *Robinson* sluoksniavimo strategija, bet pritaikius m – šakų medžio paiešką patikrinama daugiau sluoksnių pločių. Ta pati medžio paieška naudojama ir pildant patį sluoksnį dėžėmis (8 pav.). Pildymas vyksta ir vertikaliais rėžiais, todėl išvengiama „kabančių“ dėžių trūkumo. Šio algoritmo trūkumas – toks pat kaip ir *George* ir *Robinson* – sluoksnio pildymas dėžėmis vyksta tik dviejose dimensijose, tad jei yra tuštuma tarp dėžės ir sluoksnio galinės sienelės, ji nėra nagrinėjama. Patobulintas algoritmas yra žymiai lėtesnis dėl didesnės medžio šakų tikrinimo skaičiavimų apimties, tačiau pakavimo vidutinis efektyvumas yra didesnis apie 13% **[3]**.

4. EURISTINIS PAKAVIMO ALGORITMAS

Į konteinerius pakuojamų vienodų dėžių gali būti daugiau negu viena, tad pakuojant galima efektyviau išnaudoti erdvę jas sublokuojant, ir laikant kelias tokias dėžės kaip vieną objektą – dėžių bloką. Užpildant konteinerį dėžių sluoksniais, visas pakavimo uždavinys yra geriausių sluoksnių paieška duotajai kraunamų dėžių aibei ir laisvam konteinerio tūriui. Norint išvengti *George* ir *Robinson* pakavimo algoritmo trūkumo, visos laisvos formuojamo dėžių sluoksnio erdvės bus registruojamos ir traktuojamos kaip atskiri dėžių pakavimo uždaviniai, kurių kiekvienas bus sprendžiamas atskirai. Realiame gyvenime kraunant krovinius į konteinerius atsižvelgiama į daugybę faktorių ir apribojimų – ne tik į dėžės dimensijas, bet ir į galimas jos orientacijos erdvėje konteinerio dugno plokštumos atžvilgiu, jos svorį, maksimalų pakuotės apkrovimą, iškrovimo eiliškumą, pakrovimo prioritetą ir t. t. Sudaromame pakavimo optimizavimo algoritme bus atsižvelgta į dėžės orientacijos erdvėje apribojimą.

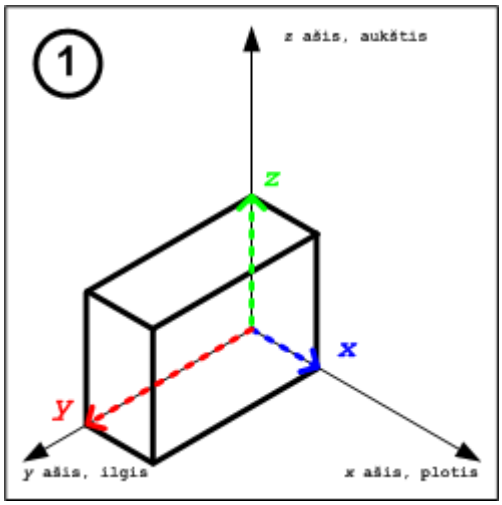
Konteineris bus pildomas sluoksniais, tad visas dėžių pakavimo uždavinys yra sudėtas iš sluoksnių pildymo uždavinių. *10 pav.* pavaizduota abstrakti algoritmo schema, susidedanti iš dviejų lygių iteracijų – išorinių iteracijų, kurių kiekviena grąžina jau užpildytą dėžėmis sluoksnį ir vidinių – kurios užpildo patį sluoksnį. Vidinės iteracijos vykdomos tol, kol į neišnagrinėjami visi sluoksnio užpildymo variantai m-šakų medyje ir atrenkamas efektyviausiai supakuotas sluoksnis. Išorinės iteracijos baigiamos, kai nebelieka nesupakuotų dėžių, arba konteineryje nebelieka vietos suformuoti bent vienam dėžių sluoksniui.



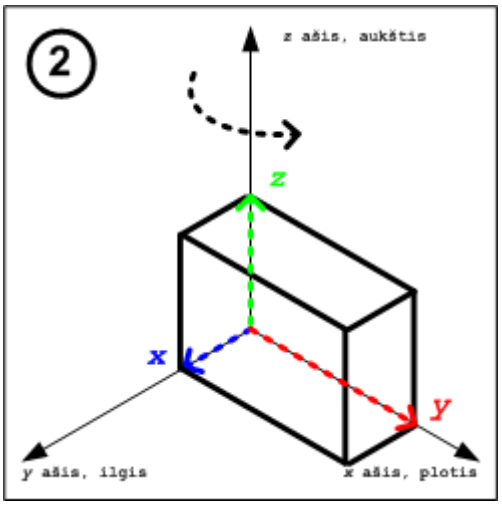
10 pav. Abstrakti pakavimo algoritmo schema.

4.1. Dėžės orientacijos erdvėje apribojimas

Trijų matavimų dėžę konteineryje galima pastatyti šešiais būdais. Bet dedant dėžę į konteinerį lieka svarbu kokia plokštuma ji padedama konteinerio dugno plokštumos atžvilgiu (*z* ašis), nes tarp kitų dviejų plokštumų (*x* ir *y* ašys) nėra jokio principinio skirtumo, ir jos gali būti sukeičiamos be apribojimų (*11 pav.* ir *12 pav.*).



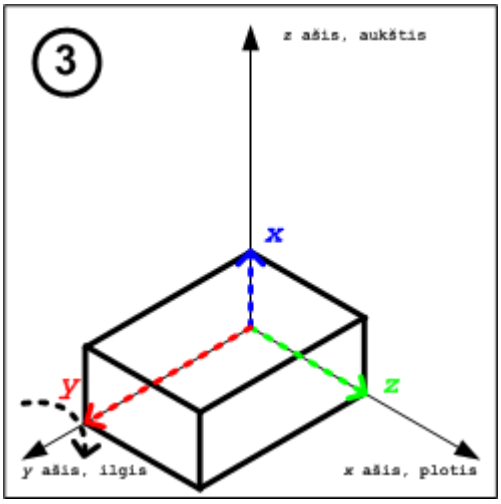
11 pav. Pirmasis dėžės padėjimo variantas – originalus dėžės koordinačių išdėstymas



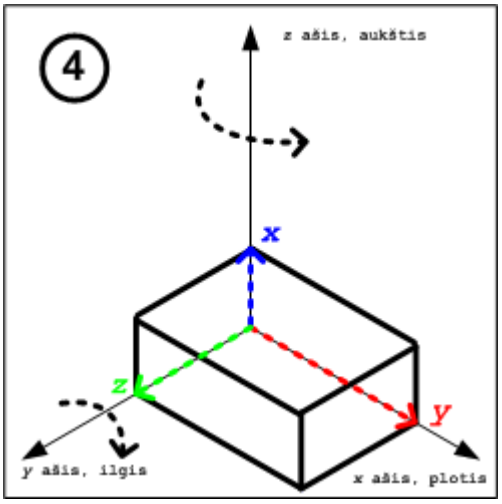
12 pav. Antrasis dėžės padėjimo variantas – *z* ašis ta pati, sukeistos *x* ir *y* ašys: dėžė pasukta 90° kampu aplink konteinerio *z* ašį.

Dėžės *z* ašis yra laikoma susieta su dėžės aukščio dimensija, *x* ašis – su dėžės pločiu, o *y* ašis – su dėžės ilgiu. Pirmasis dėžės orientacijos erdvėje variantas yra, kai visos dėžės ašys sutampa su konteinerio ašimis (*11 pav.*). Pasukus 90° kampu dėžę aplink konteinerio z ašį, gauname naują dėžės orientacijos erdvėje variantą – dėžės *z* ašis sutampa su konteinerio *z* ašimi, bet *x* ir *y* ašys yra sukeistos vietos (*12 pav.*). Abi šios dėžės orientacijos priklauso vienai apribojimų klasei, kuri tinka visoms kraunamoms dėžėms, nes aukščio dimensiją turi kiekviena trimatė dėžė.

Pasukus 90° kampu dėžę aplink *y* ašį, gauname vėl naują dėžės orientacijos erdvėje variantą – dėžės *y* ašis sutampa su konteinerio *y* ašimi, bet *x* ir *z* ašys yra sukeistos vietomis (*13 pav.*). Dar kartą pasukus dėžę aplink konteinerio y ašį, gauname ketvirtą dėžės orientacijos erdvėje variantą (*14 pav.*). Pastarieji du variantai priklauso kitai apribojimų klasei, kurioje galima dėžę dėti taip, kad konteinerio *z* ir dėžės *x* ašys sutaptų.

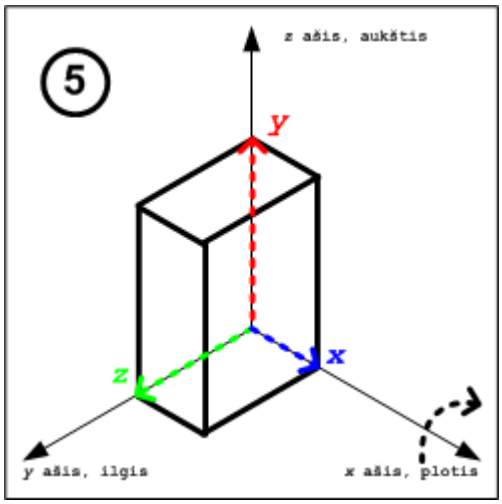


13 pav. Trečiasis dėžės padėjimo variantas – *y* ašis ta pati, sukeistos *x* ir *z* ašys: dėžė pasukta 90° kampu aplink konteinerio *y* ašį.

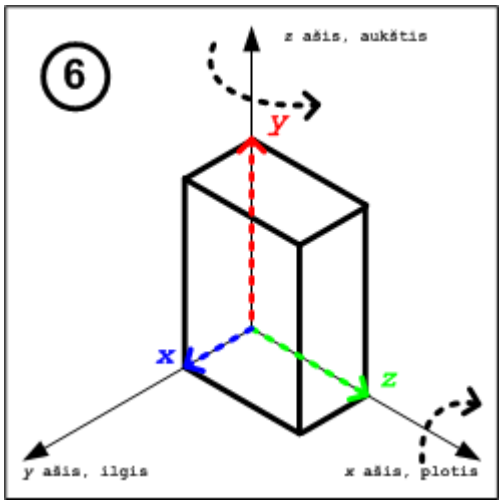


14 pav. Ketvirtasis dėžės padėjimo variantas – dėžės *x, y, z* ašys sutampa su konteinerio *z, x, y* ašimis: dėžė pasukta 90° kampu aplink konteinerio *y* ašį ir aplink konteinerio *z* ašį.

Paskutinei dėžės orientacijos konteineryje apribojimų klasei priklauso dėžės, kurias galima dėti taip, kad dėžės *y* ir konteinerio *z* ašys sutaptų. Šias orientacijas gauname dėžę pasukę 90° kampu aplink konteinerio *x* ašį (*15 pav.*) ir dar kartą pasukę 90° kampu dėžę aplink konteinerio *z* ašį (*16 pav.*).



15 pav. Penktasis dėžės padėjimo variantas – *x* ašis ta pati, sukeistos *y* ir *z* ašys – dėžė pasukta 90° kampu aplink konteinerio *x* ašį.



16 pav. Penktasis dėžės padėjimo variantas – dėžės *x, y, z* ašys sutampa su konteinerio *y, z, z* ašimis: dėžė pasukta 90° kampu aplink konteinerio *x* ašį ir aplink konteinerio *z* ašį.

Iš viso gali būti 4 dėžės išdėstymo erdvėje apribojimai – jokio apribojimo (visi 6 variantai *11 pav. – 16 pav.*), dėžės *z* ašis sutampa su konteinerio *z* ašimi (tinka bet kokiai dėžei, dėl

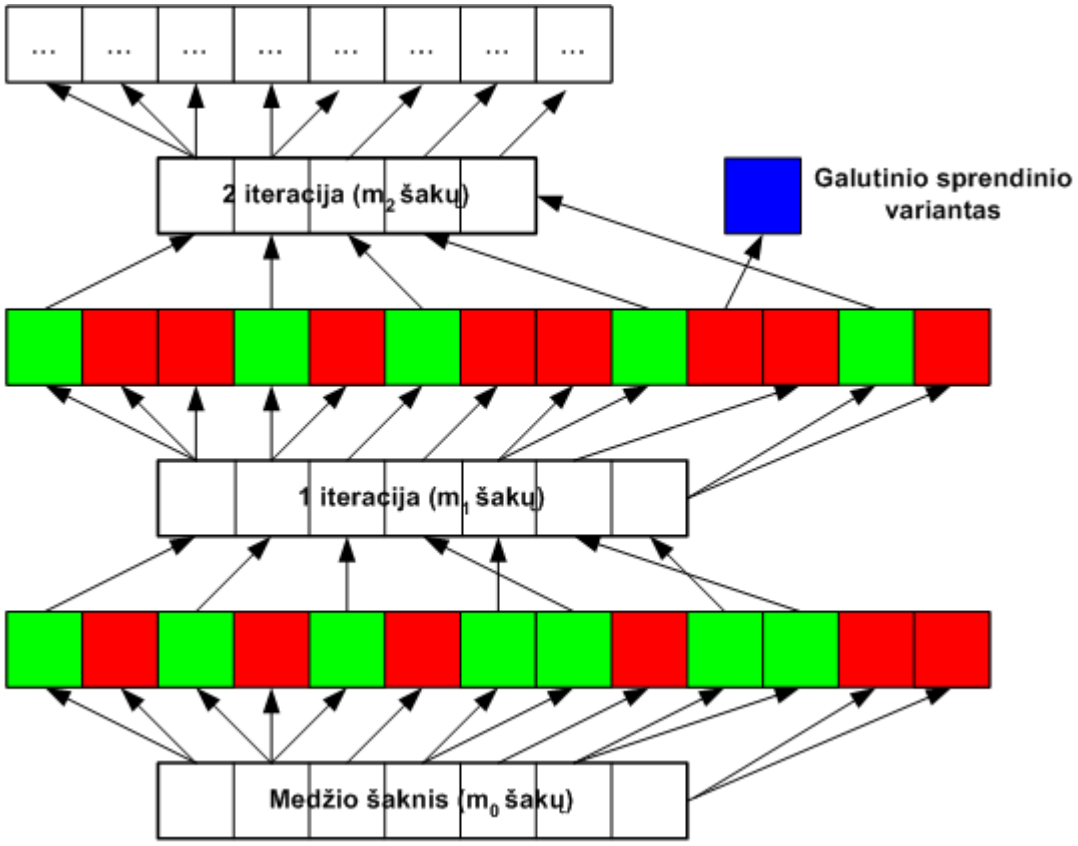
aukščio dimensijos susiejimu su *z* ašimi pagal apibrėžimą. 2 variantai: *11 pav.*, *12 pav.*), dėžės *x* ašis gali sutapti su konteinerio *z* ašimi. (4 variantai: *13 pav.*, *14 pav.*, taip pat ir *11 pav.*, *12 pav.*) ir dėžės *y* ašis gali sutapti su konteinerio *z* ašimi. (4 variantai: *15 pav.*, *16 pav.*, taip pat ir *11 pav.*, *12 pav.*).

4.2. Medžio struktūros panaudojimas

David Pisinger pasiūlytame algoritme medis yra panaudojamas norint peržiūrėti kuo daugiau dėžių išdėstymų galimybių. – tiek pagal skirtingų dėžių sluoksnių pločių, tiek ir pagal sluoksnyje dedamų dėžių blokų aukščių. Formuojant dėžių sluoksnį, galima būtų panaudoti medžio struktūrą norint turėti kelis pakankamai efektyviai dėžėmis užpildytus sluoksnių variantus. Pati medžio struktūra nebūtinai turi būti sudėtinga, nes euristiniame konstravimo algoritme nėra reikalavimo saugoti prieš tai buvusių iteracijų reikšmes – plėtojamos tik tos medžio šakos, kuriose yra mažiausi erdvės nuostoliai, o tai savo ruožtu žada efektyvesni dėžių supakavimą, tad pakavimo optimizavimo algoritmo programinio prototipo duomenų struktūromis tai būtų keli paprasti masyvai, kurių turiniai – sluoksnio užpildymo variantai, perskaičiuojami kiekvienoje tuščių sričių pildymo dėžėmis iteracijoje. Atlikus visas sluoksnio pildymo iteracijas gaunamas vienintelis sluoksnis, kuriame nebėra tuščių erdvių, į kurias galėtų tilpti dar bent viena nesupakuota dėžė.

Galutinio sluoksnio užpildymo sprendinio variantų (*17 pav.*, mėlyna spalva) gali atsirasti ir nebaigus skaičiuoti kitų dėžių tuščių erdvių užpildymo iteracijų, todėl reikėtų atskirai išsaugoti tokius duomenis, nes pats optimizavimo procesas vyksta pagal prarasto tūrio, į kuri nebetelpa nė viena dėžė, minimizavimą ir šio sprendinio varianto efektyvumas gali būti geresnis, negu gautojo jau baigus visas pildymo dėžėmis iteracijas.

17 pav. pavaizduota principinė šakų ir *m* ribų schema, ieškant geriausio dėžių sluoksnio sprendinio. Kiekvienos iteracijos metu tiek ir pradinės medžio šaknies atveju maksimalus medžio šakų kiekis yra *m*. Sugeneruoti užpildymo variantai yra surikiuojami pagal nebepanaudojamo tūrio nuostolius didėjimo tvarka ir yra paimami *m* arba mažiau geriausių variantų (žalia spalva), likę variantai yra atmetami, kaip mažiau perspektyvūs arba jau užpildyti (raudona spalva). Tarp atmetamų gali pasitaikyti ir tokių užpildymo variantų, kuriuose jau nebegalima patalpinti naujų dėžių – toks sprendinys palyginamas su turimu sprendiniu (jei toks yra) ir jei pakavimo nuostoliai yra mažesni – užregistruojamas (mėlyna spalva).



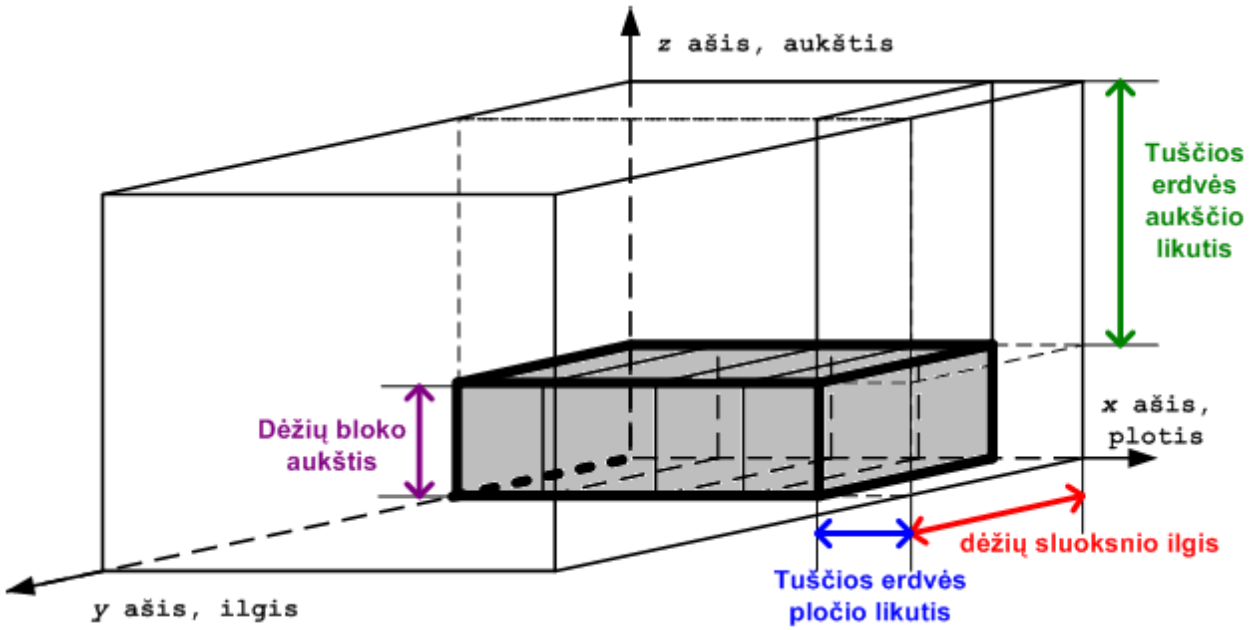
17 pav. Medžio struktūra formuojant dėžių sluoksnį

Kadangi nėra poreikio grįžti iteracijomis atgal, šis medis gali būti realizuotas dviem masyvais, kurių vienas skirtas iteracijoms, kitas – galimų variantų generavimui ir rikiavimui pagal optimizuojamą funkciją. *17 pav.* pirmasis masyvas yra nespalvotas (*Medžio šaknis, 1 iteracija, 2 iteracija...*), jo narių skaičius neviršija maksimalaus nagrinėjamo medžio šakų kiekio *m*. Antrasis masyvas *17 pav.* pavaizduotas žalia ir raudona spalvomis, šiam masyvui narių kiekio apribojimo nėra.

4.3. Medžio šaknies sudarymas formuojant dėžių sluoksnį

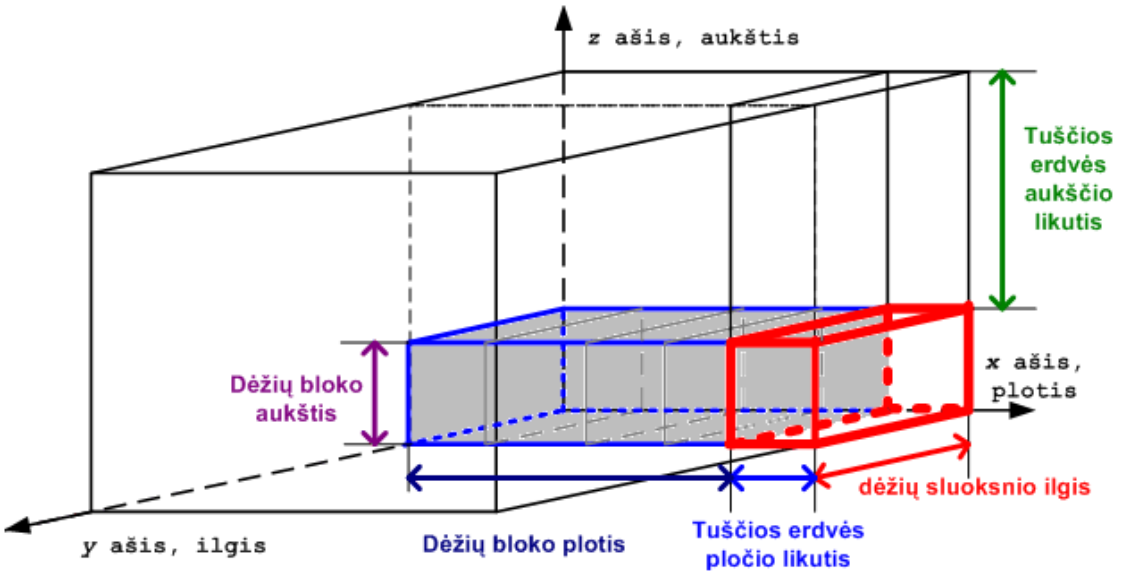
Pradedant formuoti naują dėžių sluoksnį yra peržiūrimos visos likusios nesukrautos dėžių rūšys. Kiekvienai rūšiai ieškoma geriausios leistinos orientacijos erdvėje ir išdėstomų dėžių bloko parametrų, kad tuščios erdvės pločio likutis būtų kuo mažesnis (*18 pav.*, mėlyna spalva). Be šio skaitmens kiekvienai dėžei yra priskiriamas ir tuščios erdvės aukščio likutis (*18 pav.*, žalia spalva). Abu šie parametrai yra euristikos, pagal kurias suformuojama medžio šaknis.

Visi gautieji duomenys yra surikiuojami pagal pločio likutį didėjimo tvarka, o jei tarp dviejų dėžių blokų variantų jie vienodi, tai ir pagal aukščio likutį. Iš gautojo sąrašo atrenkami pirmieji *m* dėžių blokų variantai, o jei variantų mažiau negu *m*, tai atrenkami visi – suformuojama nauji dėžių sluoksniai, į kuriuos sudedama gautieji *m* arba mažiau dėžių blokų atskirai į savo sluoksnį.

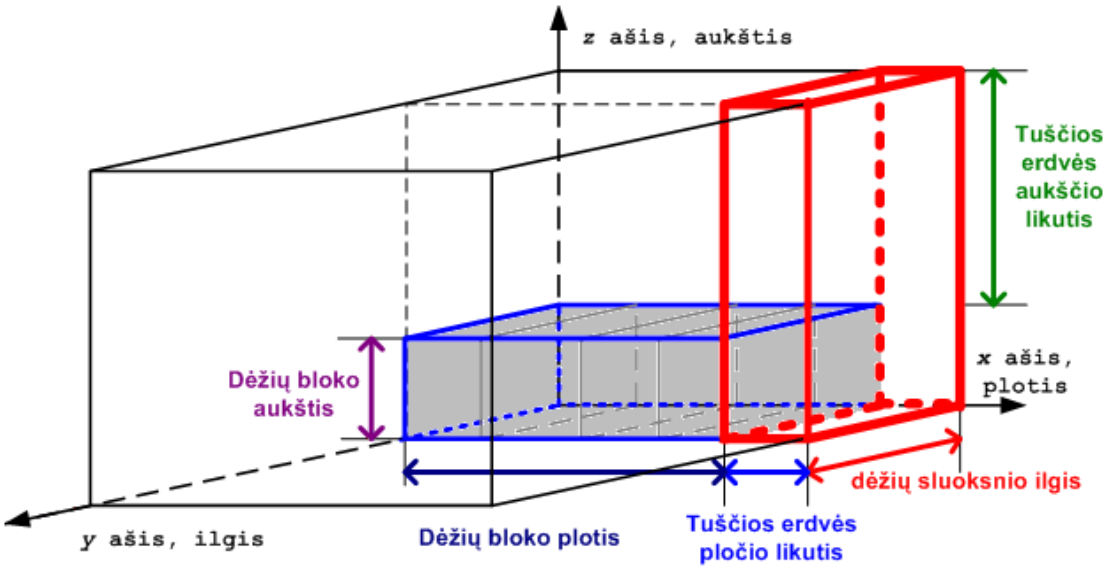


18 pav. Dėžių sluoksnis sudarant medžio šaknį.

Sukūrus naują dėžių sluoksnį, suformuojami tuščių ir nebepanaudojamų erdvių masyvai. Pirmiausia yra tikrinama tuščia erdvė, kuri yra dėžių bloko aukščio, dėžių sluoksnio ilgio ir tuščios erdvės likučio pločio (*18 pav.,* raudona spalva). Jei į šią erdvę telpa bent viena iš likusiųjų dėžių, ji kartu su kita tuščia erdve (prieš tai patikrinus ar ir į ją telpa bent viena dėžė), kuri yra formuojamo sluoksnio dimensijų, išskyrus mažesnį aukštį (*18 pav.,* raudona spalva) yra priskiriama prie tuščių ir krovimui tinkamų erdvių sąrašo. Tačiau jeigu nė viena dėžė netelpa (*19 pav.*, raudona spalva), tada nagrinėjama didesnė erdvė, kurios aukštis – lygus sluoksnio aukščiui *(20 pav.*, raudona spalva).

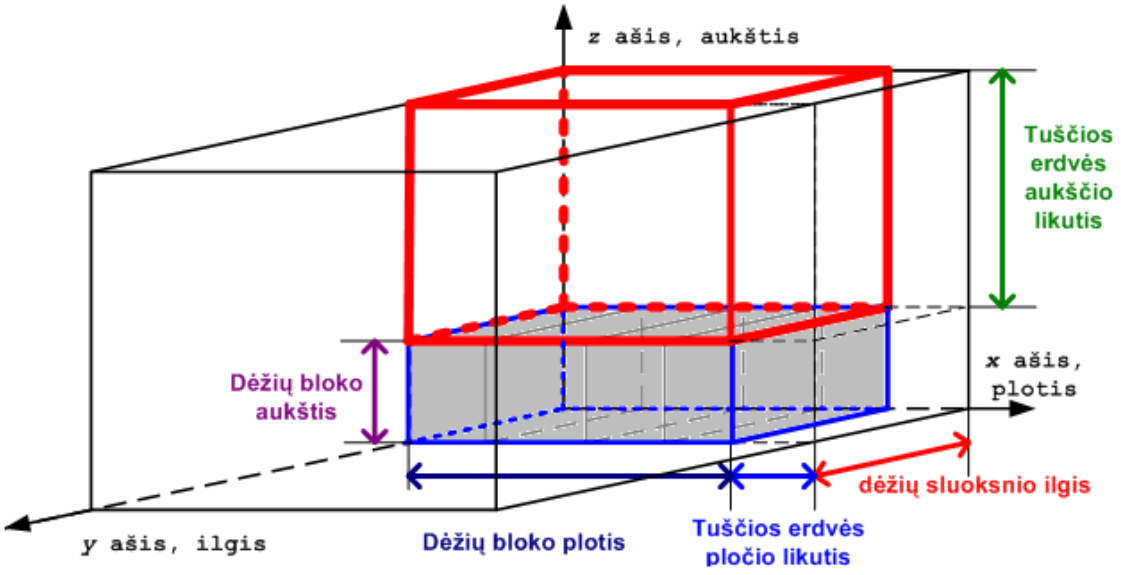


19 pav. Pirmiausiai tikrinama tuščios erdvės sritis.

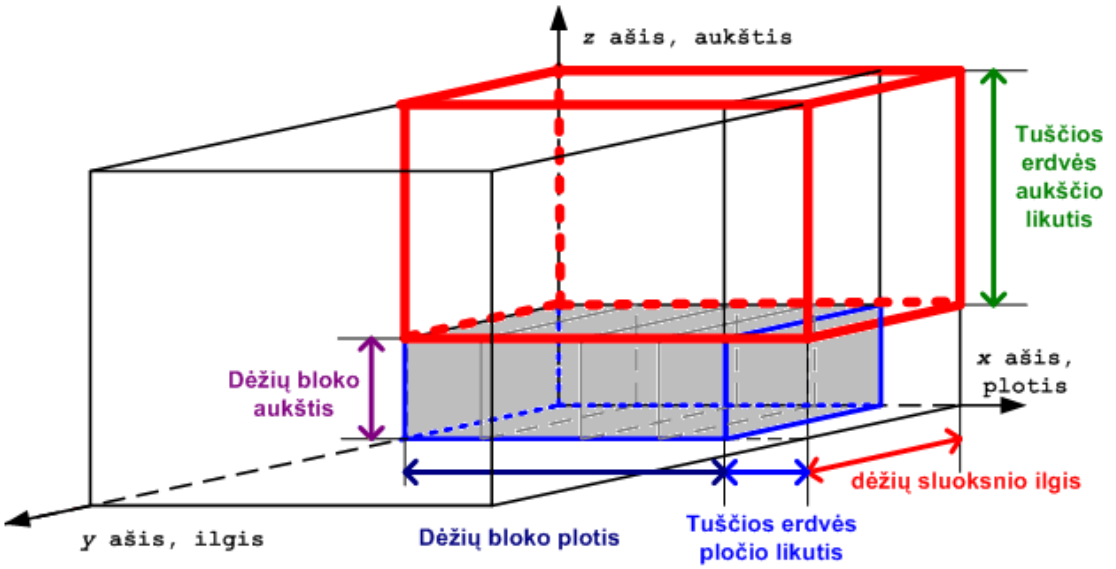


20 pav. Kita tikrinama tuščios erdvės sritis.

Jei ši erdvė yra panaudojama – tada ji kartu su likusia tuščia erdve (*22 pav.* raudona spalva), kuri savo ruožtu atskirai patikrinama, įtraukiamos į tuščių ir krovimui tinkamų erdvių sąrašą. Tačiau jeigu šioje erdvėje netelpa nė viena dėžė, tai anksčiau nagrinėta mažesnė erdvė (*19 pav.* raudona spalva) yra įtraukiama į nebepanaudojamų (erdvės nuostolių) sąrašą, o jos tūris įsimenamas kaip optimizavimo funkcijos objektas. Ištiriama likusi neužpildytoji erdvės dalis (*22 pav.*, raudona spalva) ir atitinkamai priskiriama pagal požymi telpa ar ne bent viena dėžė į atatinkamą sąrašą.



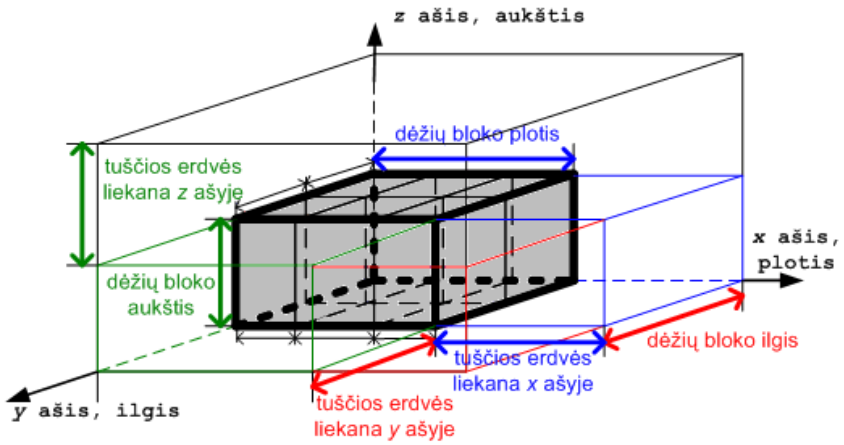
21 pav. Trečioji tikrinama tuščios erdvės sritis.



22 pav. Paskutinė tikrinama tuščios erdvės sritis.

4.4. Sluoksnio pildymas dėžėmis

Sukūrus medžio šaknį yra pradedamos jau turimų tuščių erdvių pildymo dėžėmis iteracijos. Jų metu į kiekviename sluoksnyje (*17 pav.* nespalvotas masyvas) esančią kiekvieną tuščios erdvės sritį yra supakuojama geriausia rasta dėžių bloko kombinacija (*23 pav.*). Kiekvienai laisvai sričiai yra sukuriama po naują sluoksnį, kuris įtraukiamas į aibę, iš kurio bus atrinkta geriausia *m* sluoksnių variantų kitai iteracijai (*17 pav.* spalvotas masyvas). Vienos iteracijos metu kiekvienam sluoksniui dėžėmis bandoma užpildyti tik vieną sluoksnyje esančią tuščią sritį,



23 pav. Dėžių bloko pakavimas į tuščią erdvę

Kiekvienai nesupakuotų dėžių rūšiai – „kandidatui“ į tuščios srities užpildymą yra skaičiuojamas erdvės nuostolio įvertis (*24 pav.*, pilka spalva). Jis lygus plotui, kuris lieka neužpildytas tuščios srities apačioje. Tai yra dėžės rūšies parinkimo tuščios srities užpildymui euristika. Pagal ją yra surikiuojami visi galimi variantai didėjimo tvarka ir yra pasirenkama pirmoji sąraše dėžės rūšis, kuria bus užpildomas dėžių blokas. Pagal įvertį, ji tinkamiausia užpildyti tuščiąją sritį.

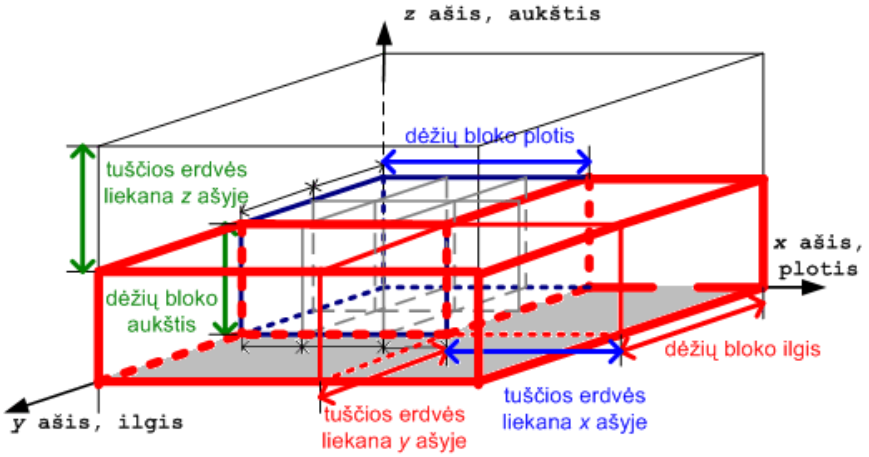
$$Įvertis_1 = \Delta x \cdot y + \Delta y \cdot x - \Delta x \cdot \Delta y,$$ kur

x – tuščiosios srities plotis, dydis konteinerio x ašyje;

y – tuščiosios srities ilgis, dydis konteinerio y ašyje;

Δx – atstumas tarp dėžių bloko ir srities sienelės x ašyje;

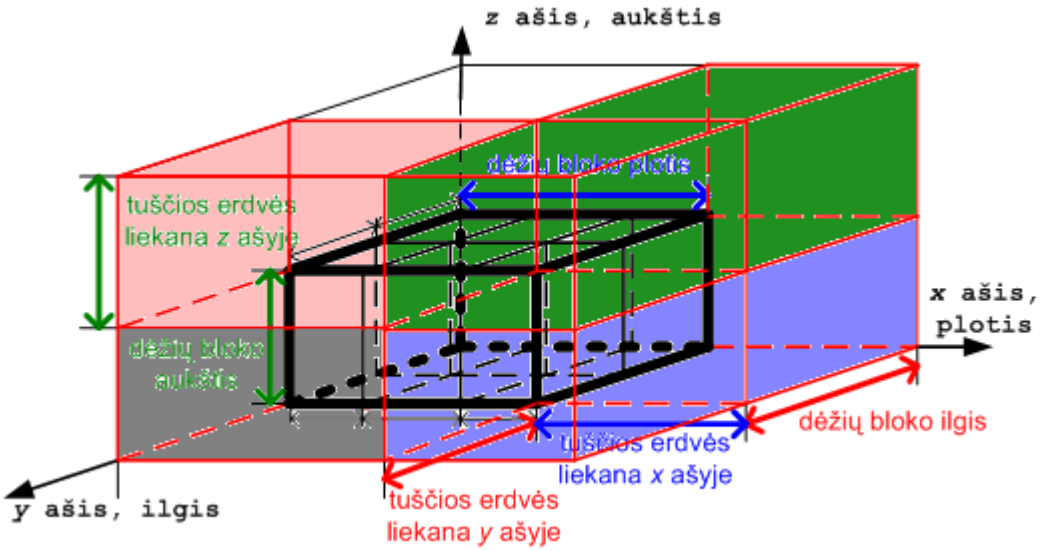
Δy – atstumas tarp dėžių bloko ir srities sienelės y ašyje.



24 pav. Dėžių sluoksnio įvertis

Išrinkus dėžės rūšį, kuri bus pakuojama į tuščią sritį, yra perskaičiuojami tuščios srities matmenys, taip pat jei atsiranda, yra užregistruojamos prarastasis pakavimui tūris bei naujos, mažesnės tuščios erdvės. Analizuojamos keturios erdvės (*25 pav.*):

- I sritis – violetinė spalva (*srities ilgis, liekana x ašyje, dėžių bloko aukštis*).
- II sritis – violetinė ir žalia spalva (*srities ilgis, liekana x ašyje, srities aukštis*).
- III sritis – pilka spalva (*liekana y ašyje, dėžių bloko plotis, dėžių bloko aukštis*).
- IV sritis – pilka ir rožinė spalva (*liekana y ašyje, dėžių bloko plotis, srities aukštis*).



25 pav. Tuščių erdvių tikrinimas.

Pirmiausia yra tikrinama *I sritis* – jeigu joje telpa bent viena dėžė iš likusių nesupakuotų dėžių aibės, ji užregistruojama kaip laisva. Jei joje iš nesupakuotų dėžių aibės netelpa nė viena dėžė, tada tikrinama *II* sritis, jei joje telpa bent viena dėžė – užregistruojama kaip laisva, jei netelpa, tada *I sritis* užregistruojama kaip prarasta, o jos tūris yra pridedamas prie formuojamo sluoksnio prarastų tūrių aibės. Vėliau tai bus naudojama kaip pildomų sluoksnių atrankos euristika. Lygiagrečiai yra analizuojamos ir *III* bei *IV sritys* atitinkamai.

Pasibaigus ankstesnėje iteracijoje medyje buvusių sluoksnių tuščių sluoksnių užpildymo dėžėmis variantų skaičiavimui, jie yra surikiuojami pagal euristiką:

$$Į\ vertis_2(i) = \sum_{j-1}^{n_i} w_j \cdot l_j \cdot h_j,$$

kur

i – sluosknio indeksas masyve;

n_i – sluoksnyje i esančių prarastų pakavimui sričių kiekis ;

w,l,h – j-tosios srities matmenys.

Ši euristika yra visų srityje esančių prarastų tūrių suma. Tai nėra tikslus įvertis, koks bus sluoksnio užpildymo efektyvumas, bet parodantis jau negrįžtamus tūrio nuostolius.

Sluoksniai su mažesniais nuostoliais yra laikomi perspektyvesniais ir ne daugiau *m* pirmųjų sąraše po surikiavimo patenka į sekančią iteraciją. Prieš tai visi sluoksniai peržiūrimi ar juose dar yra vietos, jei yra tokių, kuriuose daugiau nebetilps nė viena dėžė, tai tikrinamas efektyviausiai užpildytas sluoksnis, ir jei jis geresnis už turimą (17 pav. mėlyna spalva) ir atatinkamai įsimenamas. Likusieji sluoksniai atmetami.

Sluoksnio pildymo dėžėmis iteracijų pabaigoje nebelieka sluoksnių, į kuriuos tilptų bent viena dėžė, tad geriausias užpildytas sluoksnis tampa atskirai saugotas efektyviausiai supakuotas sluoksnis, kuris pridedamas į galutinį sprendinį.

4.5. Pakavimo optimizavimo algoritmo skaičiavimo pabaiga

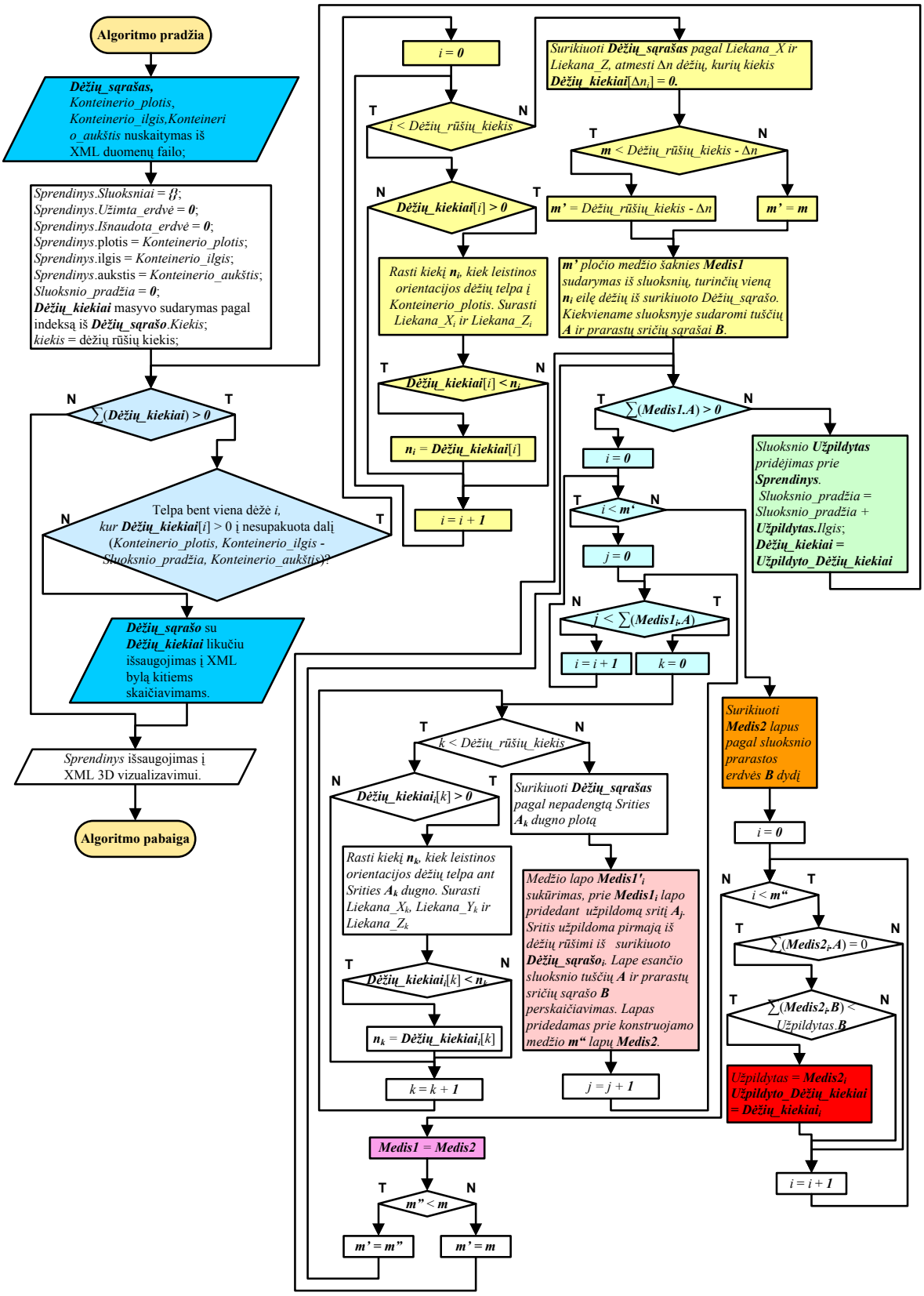
Algoritmas nutraukiamas, kai baigiasi pakuojamos dėžės, t. y. gaunamas pakavimo uždavinio sprendinys arba kai į likusią neužpildytą konteinerio erdvę nebetelpa nė viena iš likusių dėžių. Likusias nesupakuotas dėžės būtų galima atskirai išsaugoti ir pakuoti jas į naują konteinerį. Uždavinys tada būtų sprendžiamas taip pat, tik duomenų būtų jau mažiau.

4.6. Pakavimo optimizavimo algoritmo realizacija

26 pav. pavaizduota sudaryto pakavimo algoritmo blokinė diagrama. Išorinių iteracijos apima medžio šaknies sudarymą vidinėms algoritmo iteracijoms (geltona spalva), algoritmo pabaigos sąlygos tikrinimą (šviesiai melsva spalva), vidinėse iteracijose gauto užpildyto dėžėmis sprendinio pridėjimą prie sprendinio aibės (žalsva spalva).

Vidinės iteracijos susideda iš iteracijų pabaigos sąlygos tikrinimo (melsva spalva), kiekvieno sluoksnio tuščios erdvės užpildymo efektyviausia dėžės rūšimi (rausva spalva), „kandidatų“ į sekančios iteracijos pildomus dėžių sluoksnius surikiavimas pagal euristiką (morkinė spalva), galimojo sprendinio (sluoksnio, kuriame nebetelpa jokia dėžė) įsiminimas (ryškiai raudona spalva), ir iteracijos m-geriausių sluoksnių perdavimas į sekančią iteraciją (violetinė spalva).

Jeigu pasibaigus iteracijoms dar lieka nesupakuotų dėžių, yra galimybė išsaugoti jų duomenis tam, kad būtų galima likutį supakuoti kito algoritmo vykdymo metu.

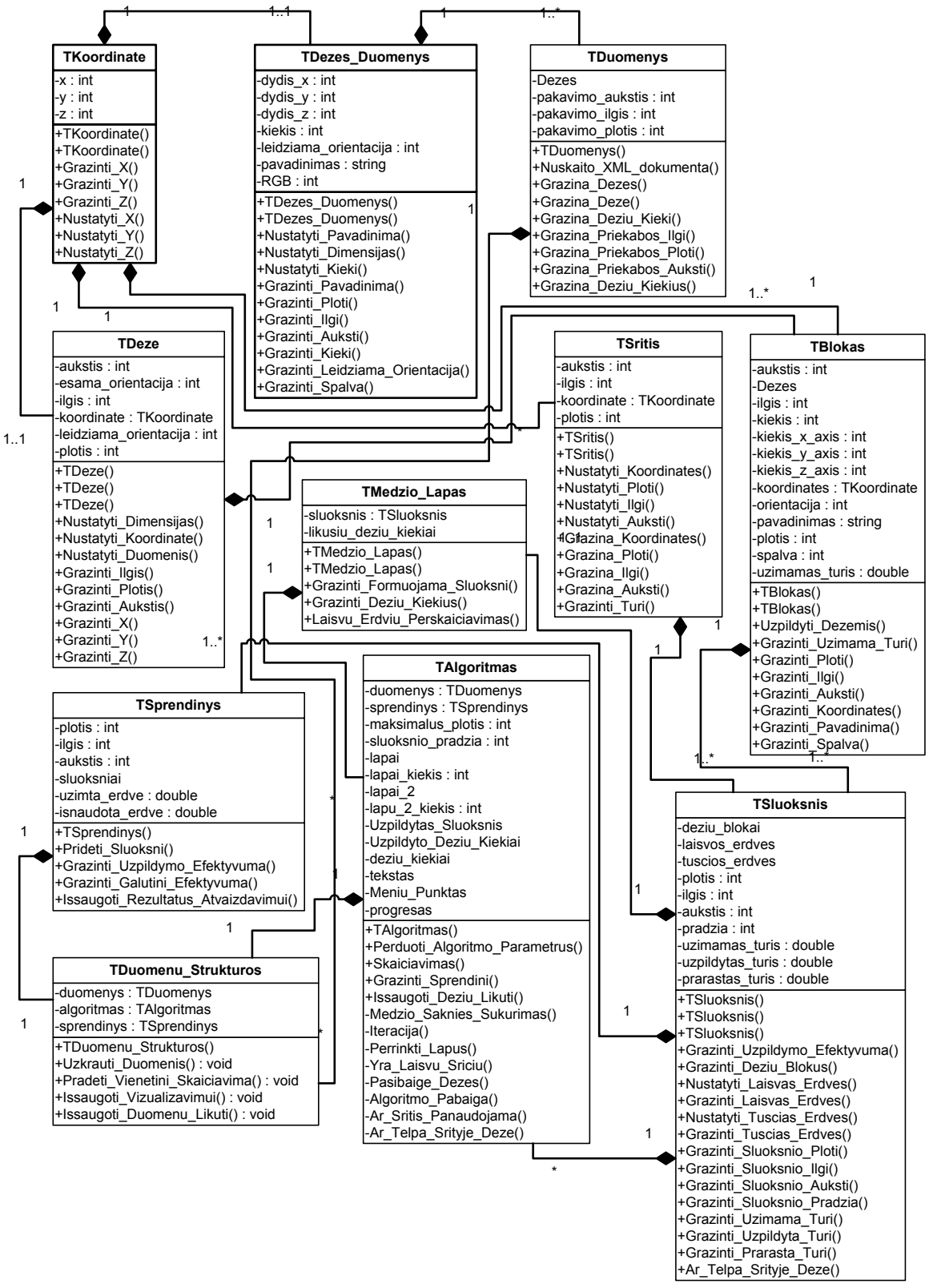


26 pav. Algoritmo blokinė diagrama.

5. ALGORITMO PROTOTIPO REALIZACIJA IR TYRIMAS

Euristinio pakavimo optimizavimo algoritmo prototipo realizacijai buvo pasirinkta „Microsoft“ „Visual Studio 2005“ programavimo terpė, C# programavimo kalba. .NET terpė buvo pasirinkta dėl platformoje integruotos atminties valdymo **[11]**. Kiekvienoje iteracijoje yra atliekama daug operacijų su atmintimi, tad sumažėja tikimybė, kad algoritme bus palikta atminties nutekėjimo spragų. Taip pat C# – objektinė programavimo kalba, tad algoritmo prototipo realizacija objektinėmis duomenų struktūromis – paprastesnė.

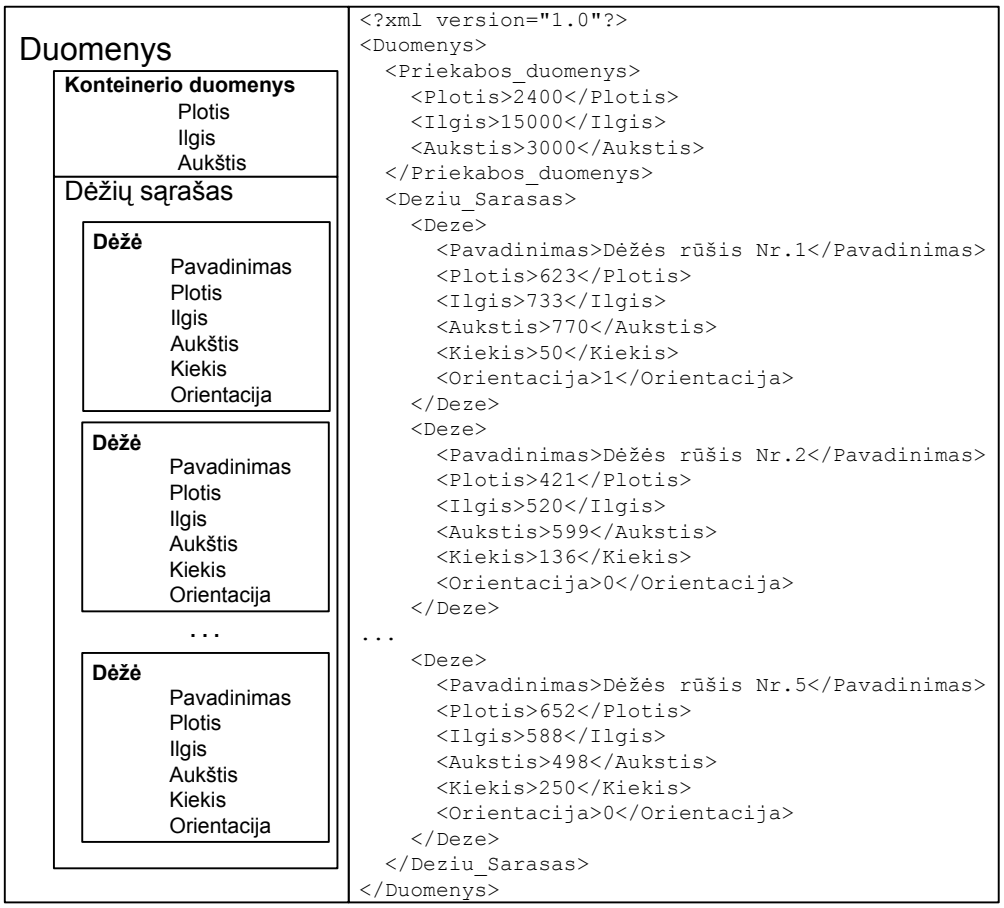
27 pav. pateikta prototipo klasių diagramą. Klasių savybės yra uždaros ir prieinamos tik per viešus metodus. Norint palengvinti programos kodo tikrinimą, klasių paveldimumas nebuvo naudojamas ir abstrakčių klasių nėra.



27 pav. Klasių diagrama

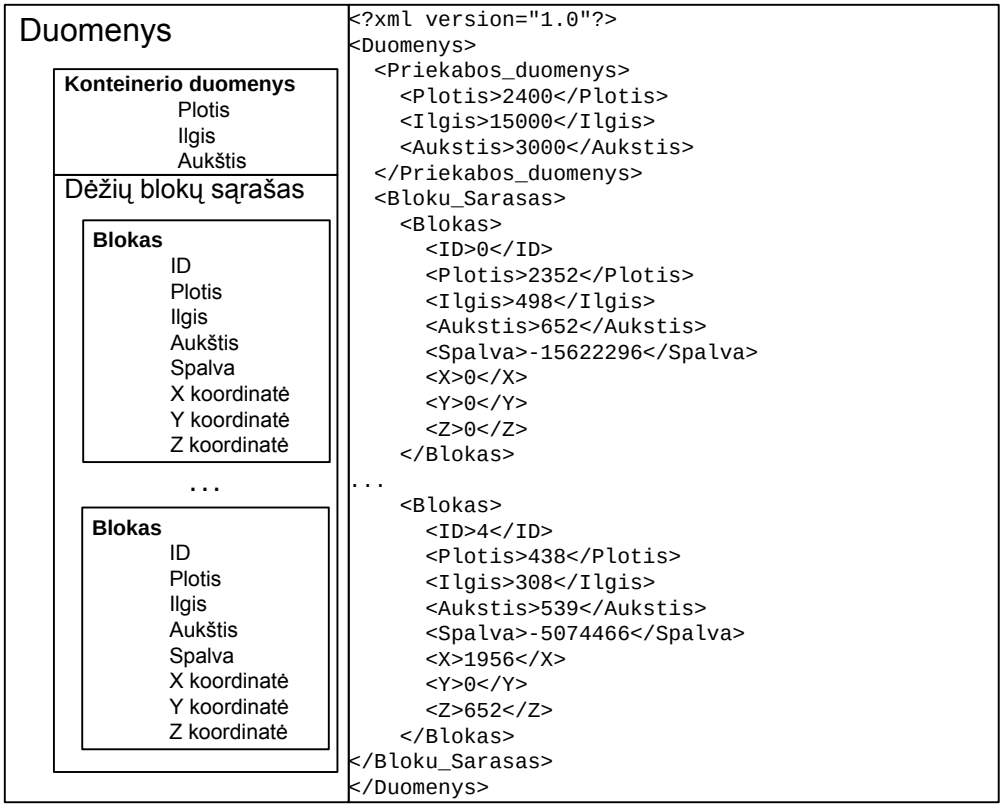
5.1. Algoritmo prototipinės realizacijos duomenų struktūros

Kadangi nebus naudojama jokia konkreti duomenų bazė, pakuojamų dėžių duomenis nutarta saugoti XML duomenų failuose. Tai suteikia šiek tiek lankstumo, nes iš kai kurių duomenų bazių nesudėtinga eksportuoti duomenis į šį formatą.



28 pav. Algoritmo prototipo naudojamų duomenų pavyzdys

28 pav. pateiktame algoritmo naudojamų duomenų pavyzdyje, matome, kad algoritmo skaičiavimui naudojami tik jam būtiniausi parametrai – konteinerio ilgis, plotis ir aukštis, bei dėžių sąrašas, kuriame yra informacija apie dėžės pavadinimą, plotį, ilgį, aukštį ,norimą supakuoti kiekį ir orientacijos erdvėje apribojimo klasę.



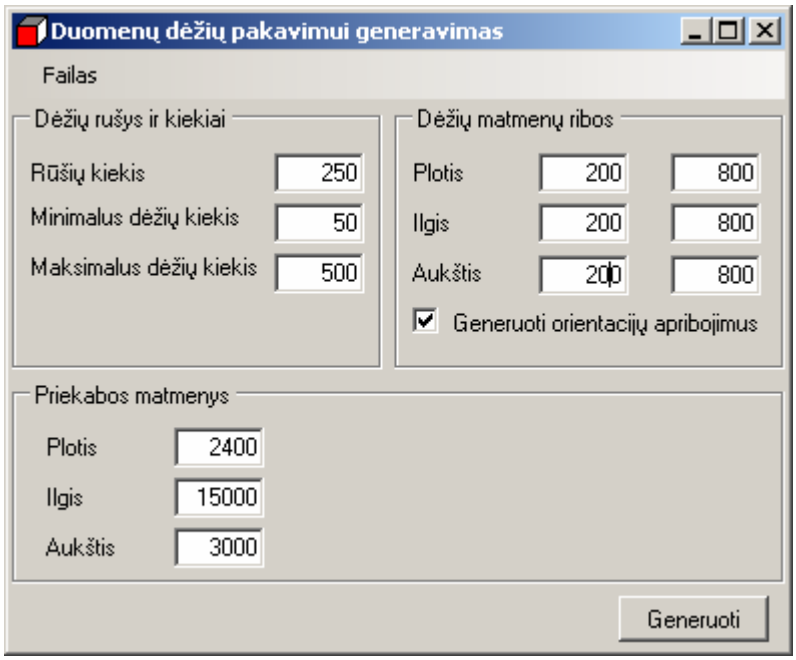
29 pav. Sprendinio vizualizacijos programos naudojamų duomenų pavyzdys

Išsaugojami tik pagrindiniai duomenys reikalingi atvaizdavimui trimatėje erdvėje – konteinerio ilgis, plotis, aukštis, bei dėžių blokų sąrašas, kuriame yra informaciją apie konkretaus bloko identifikatorių (pakavimo į konteinerį eilės numeris), bloko dimensijas ir koordinates konteinerio atžvilgiu, taip pat spalvos RGB kodą, kuri yra bendra tos pačios rūšies dėžėms. RGB kodas yra sugeneruojamas pakavimo optimizavimo metu atsitiktinai. Rezultato sprendinio duomenų struktūra atvaizduota *29 pav*.

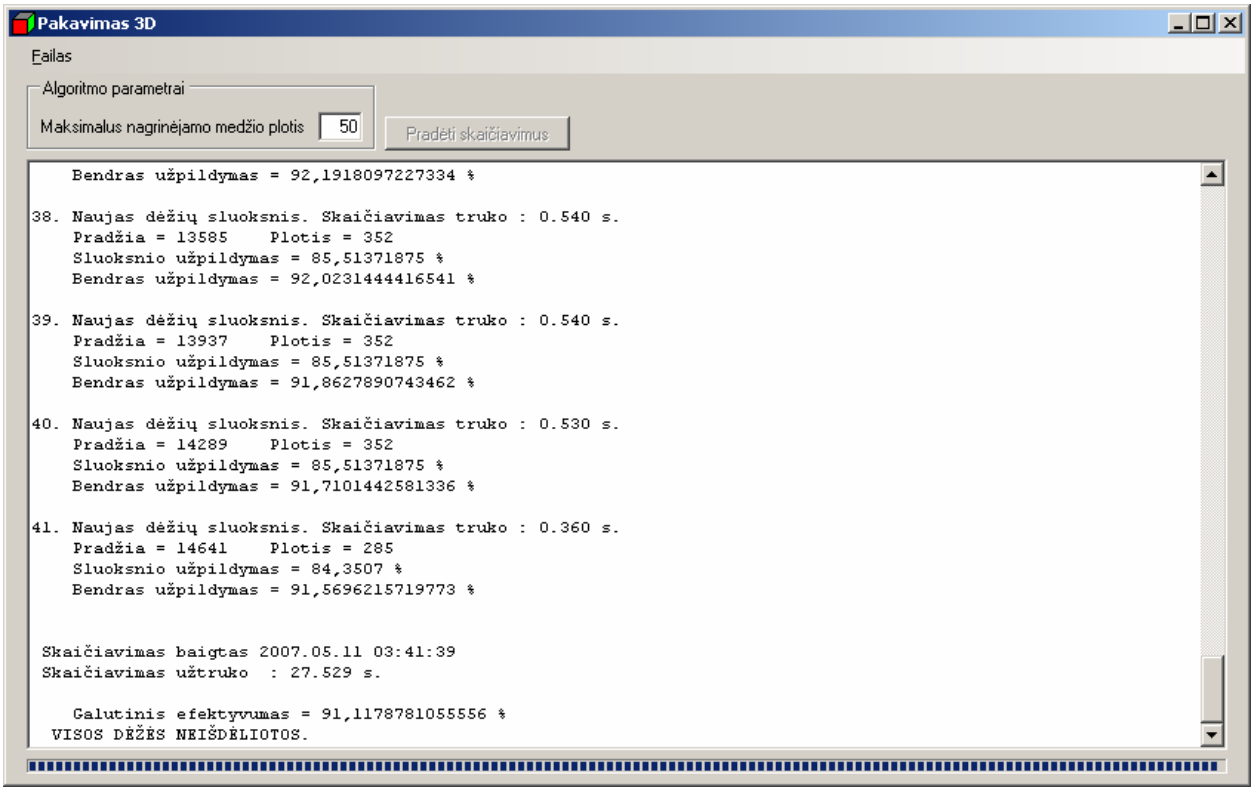
5.2. Algoritmo prototipo programinė realizacija

Pakavimo algoritmo prototipas (*31 pav.*) ir duomenų rinkinių generatorius (*30 pav.*) buvo realizuoti „Microsoft“ „Visual Studio 2005“ programavimo terpės priemonėmis objektine C# programavimo kalba. Programų sąsaja minimalistinė – vartotojas gali įvesti tik visus reikalingiausius programų veikimui duomenis. Algoritmo prototipas skaičiavimo rezultatus pateikia tekstiniame formate *TextBox* laukelyje, lango apačioje yra konteinerio užpildymo sluoksniais indikatorius *ProgressBar*. Norint įvertinti algoritmo skaičiavimo laiką, kiekvienos iteracijos pradžioje buvo įsimenamas sisteminis kompiuterio laikas, o gavus sprendinį

suskaičiuojamas laiko skirtumas. Taip išvengta skaičiavimo laiko matavimo paklaidos, kai yra perpaišomas programos langas, ir kompiuteris tuo metu dėžių pakavimo skaičiavimo nevykdo. Po kiekvienos iteracijos laiko skirtumas yra pridedamas prie bendro skaičiavimo laiko trukmės.



30 pav. Atsitiktinių duomenų rinkinių generavimo programa.



31 pav. Pakavimo algoritimo prototipo programos langas.

5.3. Algoritmo prototipo eksperimentinis tyrimas.

Algoritmo efektyvumui tirti su sukurta programine įranga (*30 pav.*) buvo atsitiktinai sugeneruota 45 duomenų rinkinių – 9 skirtingiems dėžių rūšių kiekiams po 5 variantus (2 lentelė). Konteinerio dydis buvo visiems duomenų rinkiniams vienodas – ilgis – 15000 mm, plotis – 2400 mm, aukštis – 3000 mm.

2 lentelė Sugeneruotų duomenų rinkinių charakteristikos

Eilės Nr.	Skirtingų dėžių rūšių skaičius	Kiekvienos rūšies dėžių kiekių rėžiai	Dėžių matmenų (ilgis, plotis, aukštis) rėžiai	Duomenų variantų kiekis
1	1	400	200 – 800 mm	5
2	5	50 – 300	200 – 800 mm	5
3	10	30 – 150	200 – 800 mm	5
4	15	20 – 125	200 – 800 mm	5
5	20	10 – 100	200 – 800 mm	5
6	25	5 – 80	200 – 800 mm	5
7	50	5 – 50	200 – 800 mm	5
8	100	5 – 40	200 – 800 mm	5
9	250	5 – 20	200 – 800 mm	5

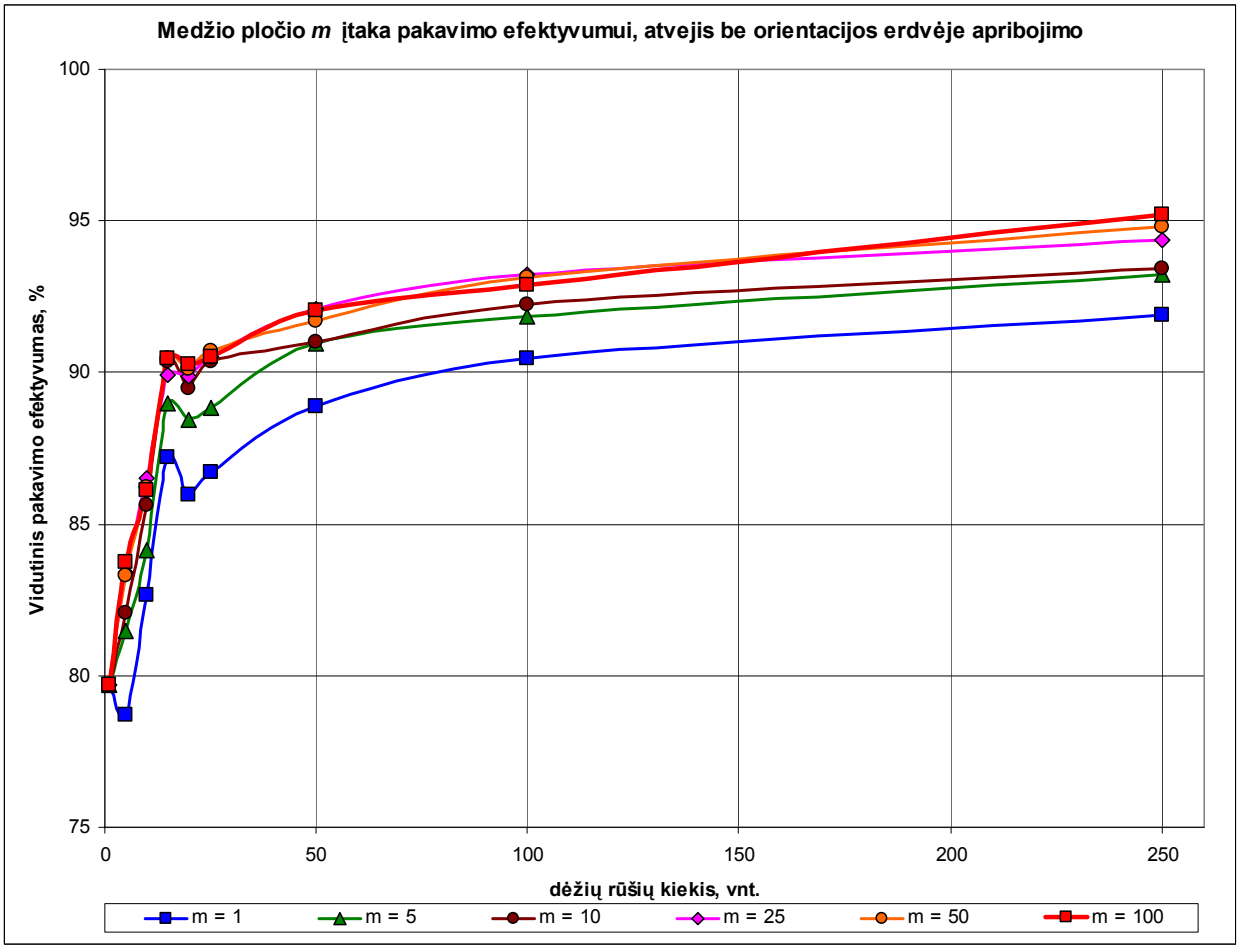
Kadangi objektyviai palyginti visų algoritmo rezultatų skirtingiems dėžių rūšių kiekiams neįmanoma dėl skirtingo sprendinio apimties kiekio, palyginimai bus atliekami pagal pirmąjį užpildytą konteinerį. Generuojant duomenų rinkinius, pakuojamų vienos rūšies dėžių kiekių rėžiai, didėjant dėžių rūšių kiekiui, buvo sąmoningai siaurinami, norint sukurti įvairesnį rezultatą. Kadangi optimizavimo algoritmas pakuoja dėžes po vieną sluoksnį, nesumažinus pačių dėžių kiekį būtų galima tikėtis visiškai vienodų sluoksnių.

Skaičiavimai buvo atliekami su orientacijos erdvėje apribojimu ir be jo. Kiekvienam duomenų rinkiniui buvo fiksuojamas: algoritmo skaičiavimo laikas, sluoksnių užpildymo dėžėmis efektyvumas ir sprendinio efektyvumas. Skaičiavimo laikas pirmajam duomenų rinkiniui buvo atliekamas du kartus dėl .NET platformos specifikos – pirmojo sluoksnio pildymo dėžėmis skaičiavimas užimdavo žymiai mažiau laiko.

Skaičiuoti kiekvieno skirtingo dėžių kiekio duomenų rinkinio rezultatų vidurkiai. Taip pat perskaičiuojami ir kiekvieno gauto sprendinio efektyvumo minimumai ir maksimumai, kuriuos galima pastebėti grafikuose (žr. prieduose Nr. 1, 3, 5, 7, 9, 11).

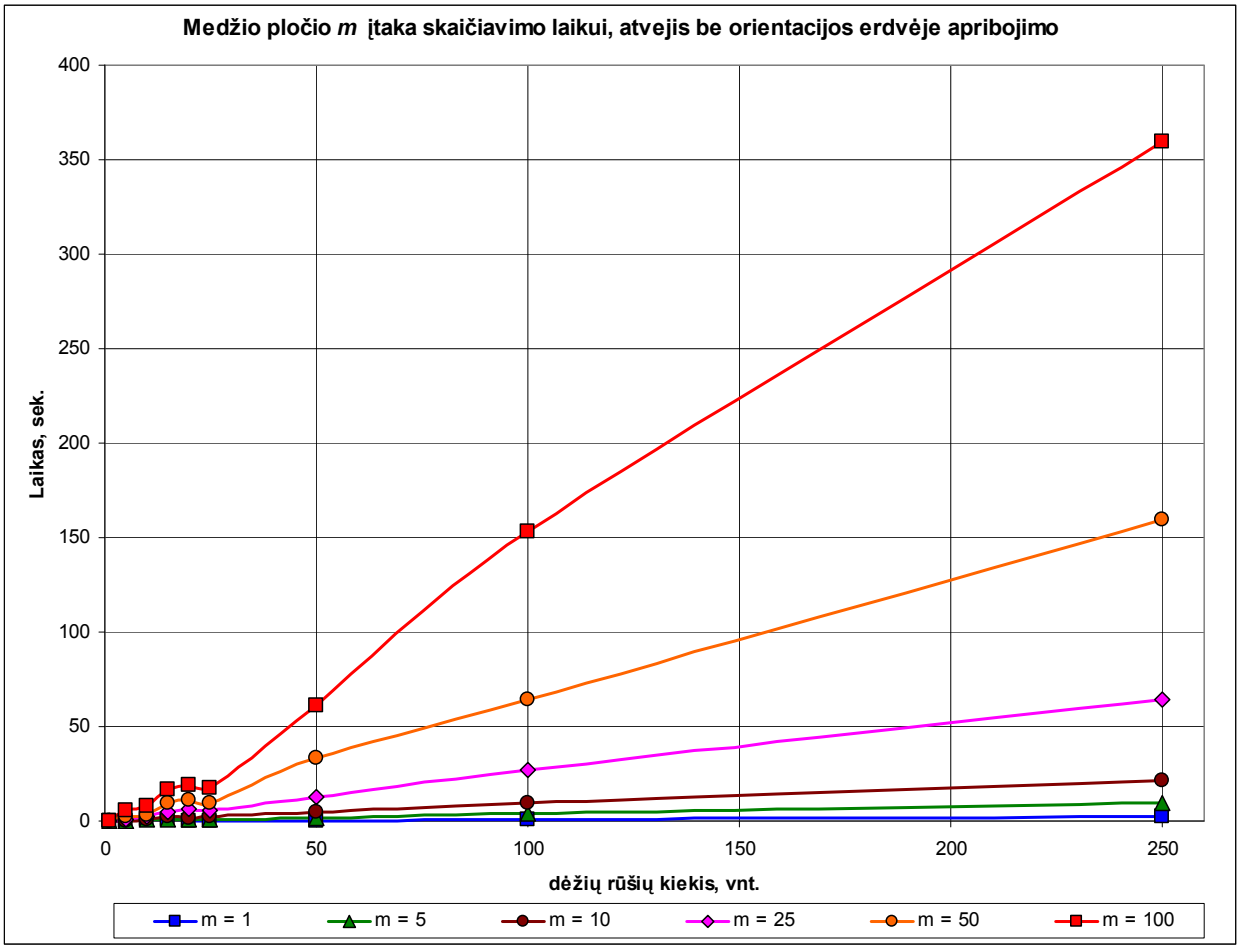
5.3.1. Maksimalaus medžio šakų kiekio *m* įtaka

Atlikti eksperimentiniai skaičiavimai su maksimalaus medžio šakų kiekiais *m* = 1, 5, 10, 25, 50 ir 100. Kadangi buvo iš viso sugeneruoti 45 duomenų rinkiniai.



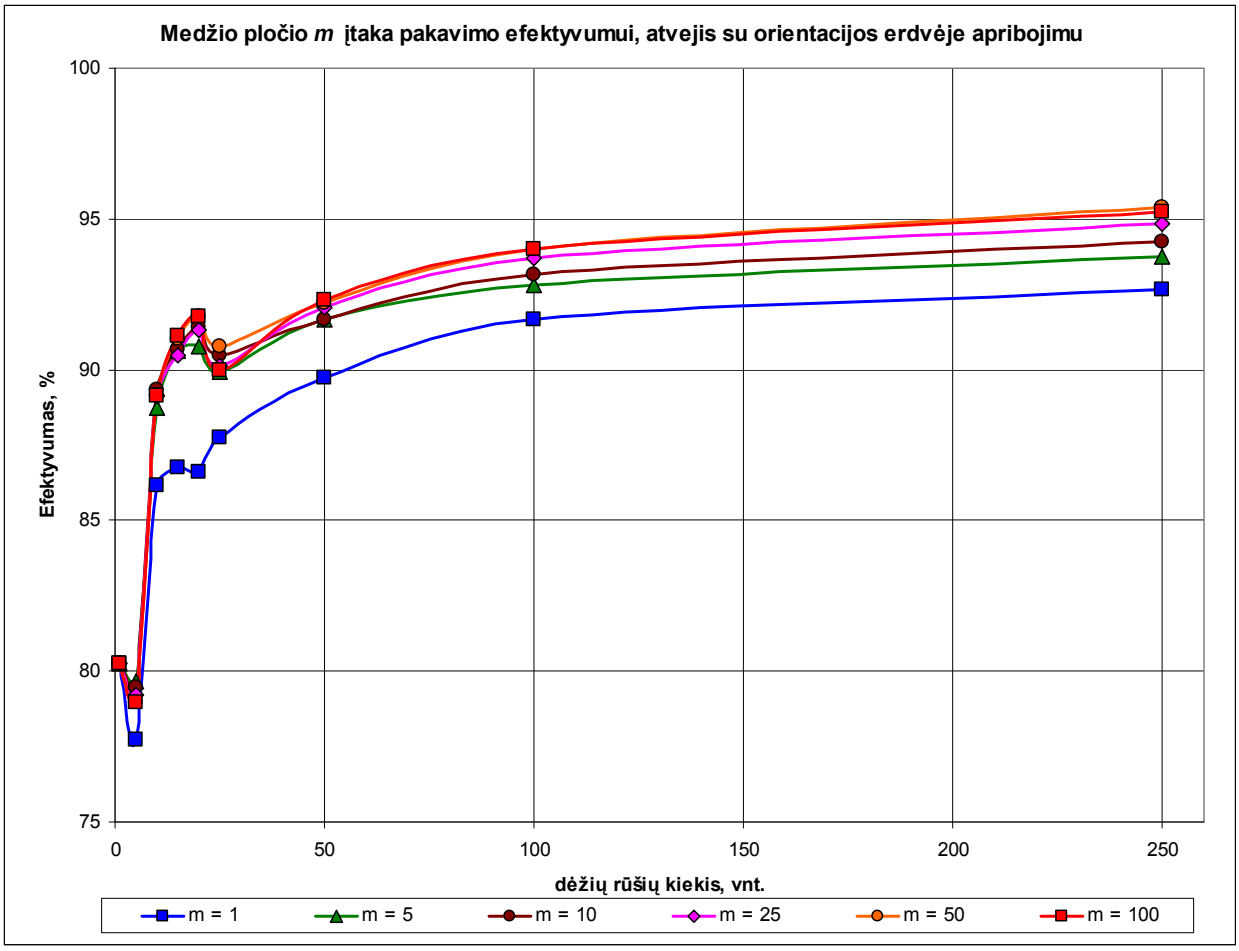
32 pav. *m* įtaka pakavimo be orientacijos erdvėje apribojimo efektyvumui.

32 pav. grafike matome, kad esant didesniam dėžių rūšių asortimentui sprendinių efektyvumas asimptotiškai didėja, tai galima būtų paaiškinti tuo, kad esant didesnei skirtingų dėžių įvairovei yra daugiau galimybių surasti efektyvesnį dėžių išdėliojimą. Didinant maksimalaus medžio plotį *m* stebime gerėjančius vidutinius sprendinio efektyvumo rodiklius. Tai ypač gerai matosi tiems duomenų rinkiniams, kuriuose dėžių rūšių skaičius viršija 20. Esant mažesniam dėžių rūšių kiekiui padidėja duomenų įtaka pakavimo efektyvumui (žr. priedus Nr.1, 3, 5, 7, 9 ,11) – t. y. kuo „tinkamesni“ dėžių duomenys, tuo efektyviau jos ir supakuojamos. Iš minėtų priedų lengva pastebėti didžiulę pakavimo sprendinių efektyvumų sklaidą, kai dėžių yra mažiau negu 20.



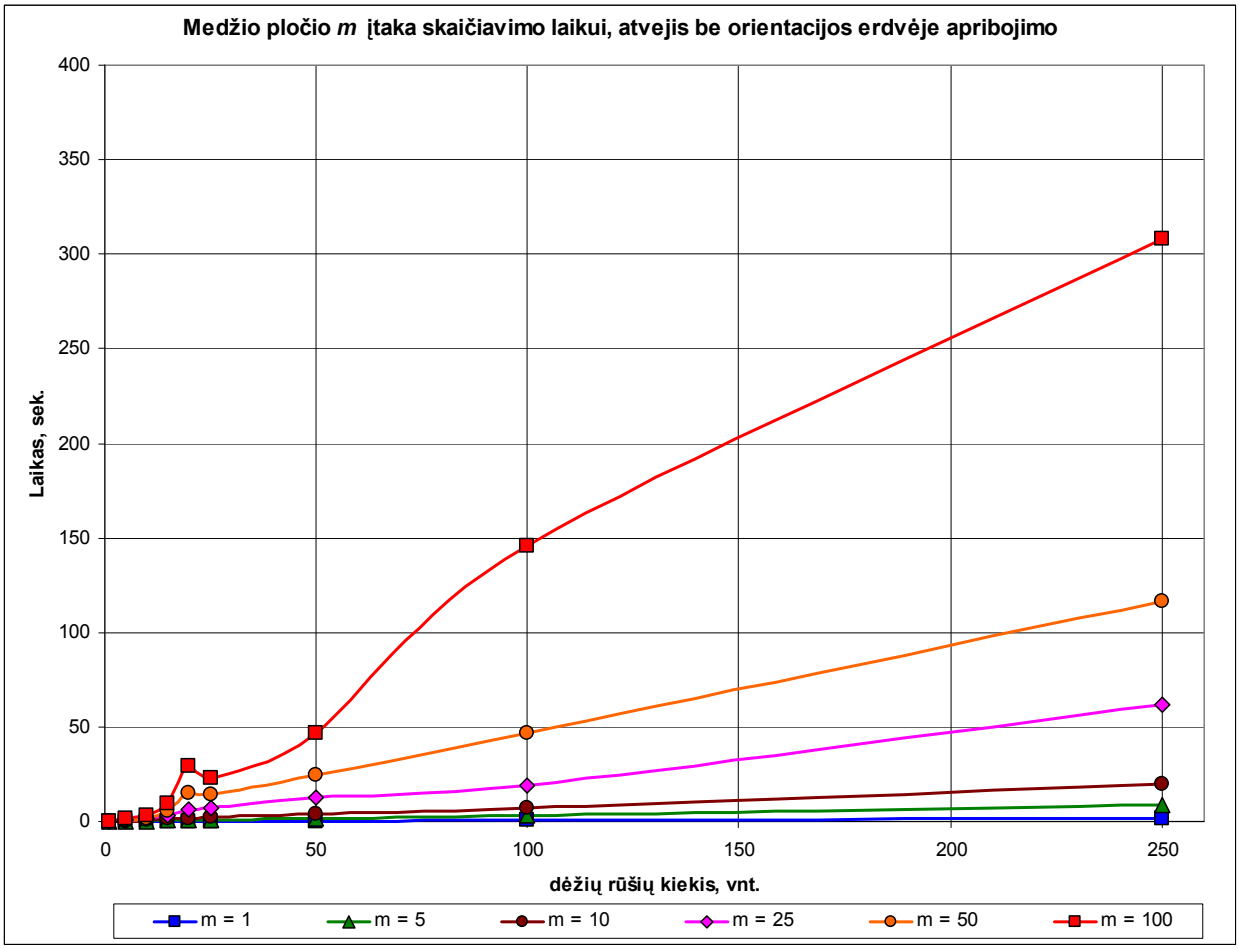
33 pav. *m* įtaka pakavimo be orientacijos erdvėje apribojimo skaičiavimo laikui.

33 pav. grafike matome beveik tiesinę priklausomybę tarp vidutinio skaičiavimo laiko ir pakuojamų dėžių rūšių kiekio. Didinant medžio plotį *m* vidutiniai skaičiavimo laikai didėja irgi tiesiškai.



34 pav. *m* įtaka pakavimo su orientacijos erdvėje apribojimu efektyvumui.

34 pav. grafikas yra beveik analogiškas grafikui *32 pav*. Tačiau galima pastebėti ir skirtumų – dėžės su orientacijos erdvėje apribojimu yra pakuojamos šiek tiek efektyviau. Taip pat mažesniam dėžių rūšių kiekiui yra „ekstremalesni“ rezultatai – tai aiškiai matosi su 5 rūšių dėžių duomenų rinkiniu. Mažesnis efektyvumo vidurkis reiškia didesnę orientacijos erdvėje apribojimo įtaką, esant mažesniam dėžių rūšių kiekiui.



35 pav. *m* įtaka pakavimo su orientacijos erdvėje apribojimu skaičiavimo laikui.

35 pav. grafikas vėlgi labai panašus į *33 pav.* grafiką. Esminis skirtumas – mažesnis skaičiavimo laikas dideliems dėžių rūšių kiekiams.

5.3.2. Dėžės orientacijos erdvėje apribojimo įtaka

Kad įvertintume dėžės orientacijos erdvėje apribojimo įtaką vidutiniam pakavimo efektyvumui bei vidutiniam algoritmo skaičiavimo laikui buvo sudarytos dvi rezultatų lentelės. Iš visų apskaičiuotų vidutinių pakavimo be orientacijos erdvėje apribojimo efektyvumo rodiklių buvo atimti atitinkami pakavimo su apribojimais efektyvumo rodikliai.

3 lentelėje matome, kad neigiamų skaičių (45) yra ganėtinai daugiau negu teigiamų (9) skaičių, o tai reiškia, kad su orientacijos apribojimu pakavimas yra efektyvesnis. Tačiau skirtumai vidutiniškai mažesni negu 1%. Didinant maksimalų medžio plotį *m* vidurkių skirtumų vidurkiai atsitiktinai didėja ir mažėja. Standartinis nuokrypis po truputį didėja, tad galima

paminėti vidurkių skirtumų sklaidos didėjimą, tačiau tai vis tiek labai mažos reikšmės, kurios nerodo jokios aiškios tendencijos. O nežymiai didesnį pakavimo efektyvumą galima būtų paaiškinti pakavimo į tuščią sritį euristikos specifika – nėra nagrinėjamos tuščios sritys pagal visą srities plotį (*25 pav.*). Pirmenybė yra suteikta sričiai, kurios ilgis yra per visą srities ilgį (*25 pav.*, I ir II sritys). Tad su apribojimais ir pagal šią euristiką mažiau efektyviais sprendiniais iteracijų eigoje gaunami šiek tiek labiau efektyvūs sprendiniai negu be orientacijos apribojimų.

3 lentelė Pakavimo *e*fektyvumo skirtumai

Skirtumas, Procentai	Maksimalus medžio plotis *m*					
Dėžių rūšių kiekis	1	5	10	25	50	100
1	-0,52763	-0,52763	-0,52763	-0,52763	-0,52763	-0,52763
5	0,97315	1,81875	2,60020	4,21282	4,33286	4,76397
10	-3,50760	-4,63522	-3,71593	-2,58830	-2,89549	-2,99686
15	0,44464	-1,62797	-0,32200	-0,58076	-0,56398	-0,63063
20	-0,59916	-2,30656	-1,97043	-1,44691	-1,65190	-1,52365
25	-1,04929	-1,06692	-0,11019	0,42769	-0,03021	0,52052
50	-0,81367	-0,71427	-0,65891	0,01116	-0,52214	-0,29086
100	-1,20075	-0,97679	-0,93299	-0,47117	-0,87957	-1,11806
250	-0,73492	-0,50423	-0,80672	-0,47423	-0,58333	-0,00972
Vidurkis	-0,77947	-1,17121	-0,71607	-0,15970	-0,36904	-0,20144
Standartinis nuokrypis	1,136178	1,574419	1,50493	1,660715	1,754054	1,89608

4 lentelėje matome, kad situacija yra visiškai atvirkščia – neigiamų skaičių (11) yra mažiau negu teigimų (34), o tai reiškia, kad kompiuteris sugaišo mažiau laiko skaičiuodamas pakavimo uždavinius su orientacijos apribojimu, negu be jo. Vidurkių skirtumų vidurkiai didėjant m skaičiui tik didėja, o tai rodo tik didėjant atotrūkį. Standartinis nuokrypis irgi didėja, kas byloja apie taip pat didėjantį skirtumų sklaidą. Įsižiūrėjus į pačius skirtumus matome, kad 250 rūšių dėžių atveju skirtumas tarp skaičiavimo laiko vidurkių žymiai auga dėl didesnio m skaičiaus. Tai galima būtų paaiškinti tuo, kad pakavimo su orientacijos erdvėje apribojimais atliekama mažiau palyginimo ir tikrinimo veiksmų – vienai dėžei gali būti leistinos keturios arba net dvi orientacijos erdvėje vietoje šešių (*11 pav. – 16 pav.*). Todėl atliekama mažiau skaičiavimo operacijų.

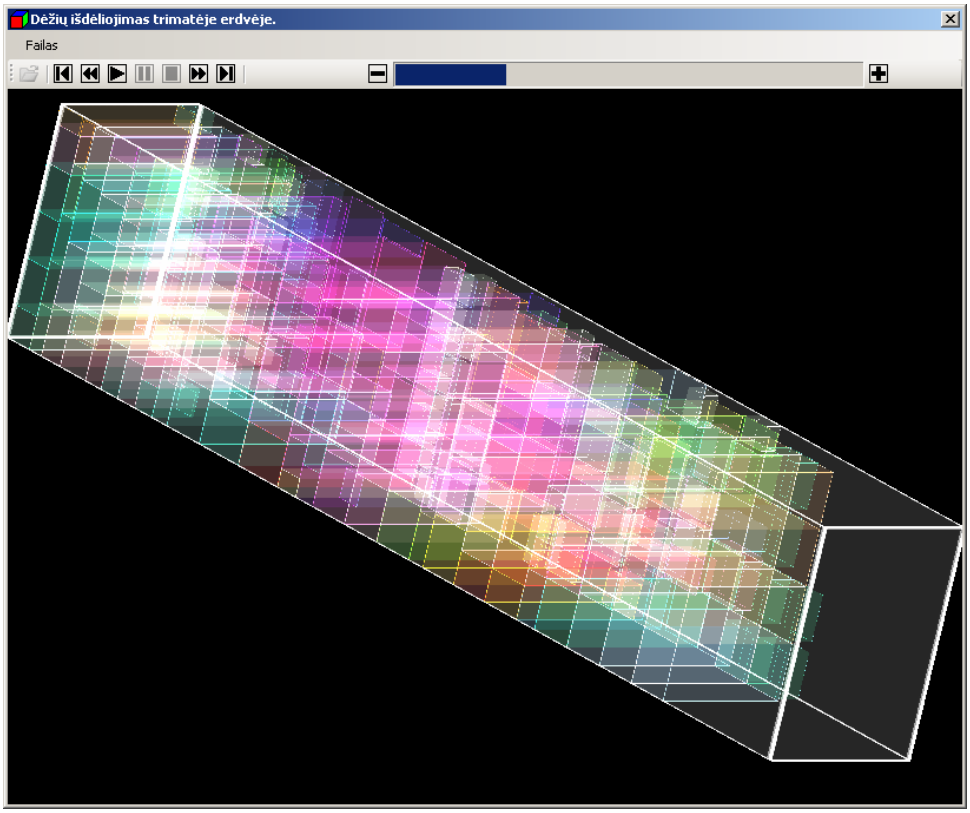
4 lentelė Pakavimo skaičiavimo laiko skirtumai

Laikas, sekundės	Maksimalus medžio plotis *m*					
Dėžių rūšių kiekis	1	5	10	25	50	100
1	0,016	0,00200	0,040	-0,018	0,080	-0,050
5	0,140	0,06200	0,108	0,627	1,178	3,245

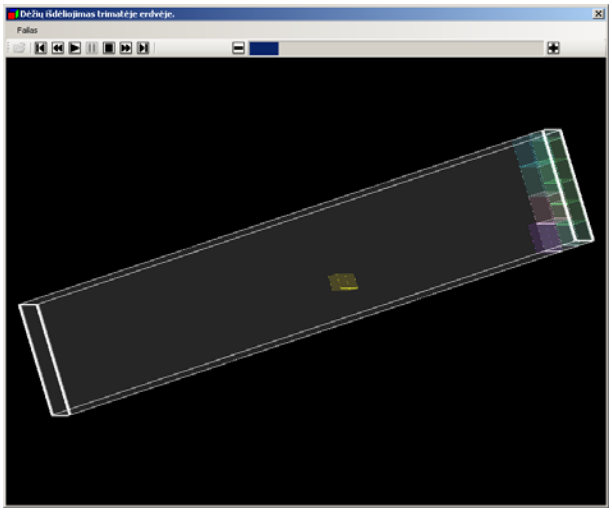
10	0,100	0,11200	0,400	0,965	1,641	4,556
15	0,240	0,11820	0,789	1,677	4,355	7,414
20	0,070	0,22440	0,254	-0,200	-3,896	-10,235
25	-0,100	-0,21000	-0,198	-1,380	-4,320	-5,385
50	0,044	0,26720	0,501	0,196	8,647	14,671
100	0,120	1,06360	2,676	8,210	17,338	7,621
250	0,254	0,66920	0,986	2,315	43,291	50,839
Vidurkis	0,098	0,257	0,617	1,377	7,590	8,075
Standartinis nuokrypis	0,102512	0,352466	0,787265	2,520472	13,56357	15,96794

5.4. Pakavimo optimizavimo sprendinio vizualizavimas

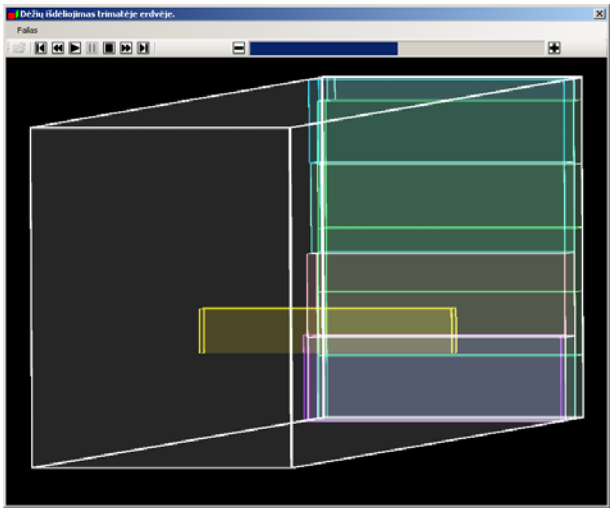
Norint vizualizuoti dėžių pakavimo sprendinį trimatėje erdvėje, su „Microsoft Visual Studio 2005“ programavimo terpe remiantis „Direct X“ technologijomis buvo sukurta programinė įranga (*36 pav.*). Kadangi dėžės į konteinerį pakuojamos blokais, atskiros dėžės nėra atvaizduojamos. Kad būtų lengviau atskirti sprendinyje vienos rūšies dėžes, kiekviena dėžės rūšis turi individualią spalvą. Sprendinį naudojantis pele galima vartyti įvairiomis projekcijomis, taip pat galima sekti patį pakavimo procesą – kiekvieno bloko judėjimą iš konteinerio galo į jam skirtą vietą (*37 pav*, *38 pav.*).



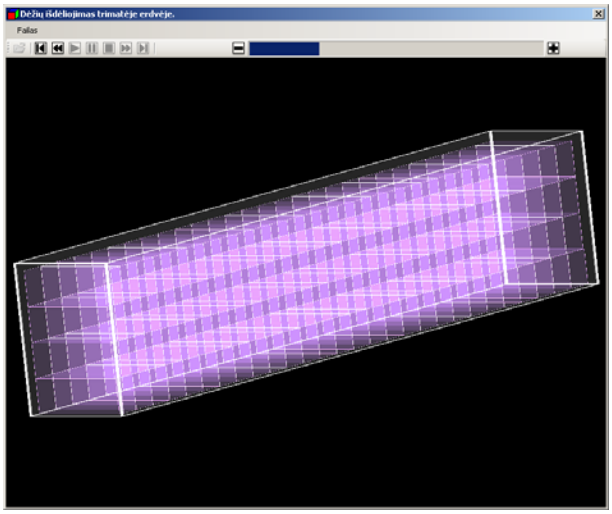
36 pav. Vizualizacijos programos langas.



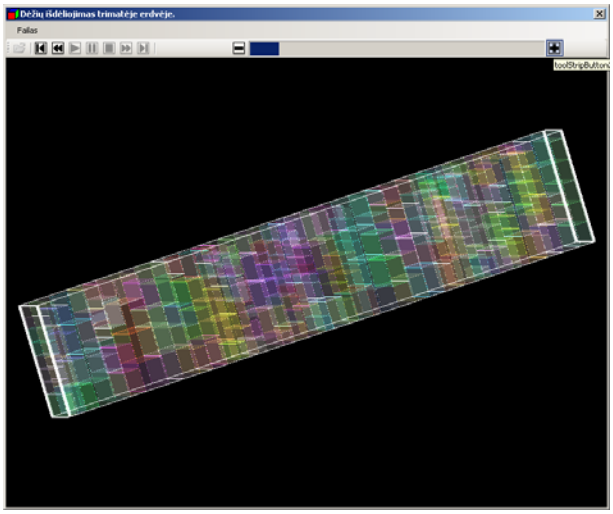
37 pav. Vienos rūšies dėžės pakavimo animacija.



38 pav. *37 pav.* vaizdas kitu kampu.



39 pav. Vienos rūšies dėžės pakavimo sprendinys.



40 pav. 250 rūšių dėžių pakavimo sprendinys.

6. IŠVADOS

1. Išanalizavus *George* ir *Robinson* bei *David Pissinger* algoritmus ir atlikus *George* ir *Robinson* algoritmo prototipinę realizaciją, buvo nustatyti jų privalumai ir trūkumai – algoritmai greitai veikiantys, pakankamai efektyvūs, tačiau gali grąžinti nesiremenčių į plokštumą dėžių sprendinius ir neištiria visų dar nepanaudotų erdvių.
2. Prie klasikinio trimačių dėžių pakavimo uždavinio buvo pridėtas papildomas apribojimas – dėžė gali būti orientuota erdvėje konteinerio dugno plokštumos atžvilgiu. Šiam uždaviniui spręsti buvo sukurtas euristinis pakavimo algoritmas, paremtas *George* ir *Robinson* sluoksniavimo strategija, *David Pissinger* medžio su *m*-šakų paieška bei atskiru kiekvienos neužkrautos erdvės vertinimu.
3. Pagal algoritmo formalizuotą aprašymą naudojant objektinio programavimo C# kalbą buvo atlikta prototipinė algoritmo realizacija, su kuria atlikti eksperimentiniai tyrimai naudojant duomenų generatoriumi sudarytus atsitiktinių duomenų rinkinius.
4. Statistiškai įvertinus rezultatus nustatyta, kad papildomas dėžės orientacijos erdvėje apribojimas turėjo labai nežymią įtaką vidutiniam pakavimo efektyvumui – pakavimo efektyvumo skirtumų vidurkiai nebuvo didesni nei 1,2%. Didėjant dėžių rūšių asortimentui, vidutinis pakavimo efektyvumas duotosioms atsitiktinių duomenų imtims, asimptotiškai artėjo prie 95%. Pastebėta, kad sprendimo laikas yra tiesiškai proporcingas duomenų rinkinio skirtingų pakuojamų dėžių rūšių kiekiui.
5. Didinant paieškos medžio maksimalų šakų skaičių m, papildomo dėžės orientacijos erdvėje apribojimo įtaka pakavimo efektyvumui nekito. Apribojimo įtaka uždavinio sprendimo laikui, didėjant pakuojamų dėžių rūšių asortimentui – didesnė. Pastebėta, kad didėja ne tik skaičiavimo laiko vidurkių skirtumai, bet ir jų sklaida.
6. Didinant pakavimo algoritmo maksimalų medžio plotį *m,* didesniems nei 15 skirtingų dėžių rūšių duomenų rinkiniams, buvo gauti geresni vidutiniai pakavimo efektyvumo rodikliai – nuo 91% iki 95%.
7. Prie algoritmo prototipo buvo sukurta pakavimo sprendinio vizualizavimo programa, kuri atvaizduoja dėžių blokus, bei pačio dėžių bloko pakavimo animaciją trimatėje erdvėje. Pelės pagalba galima keisti stebėjimo kampą ir atstumą.

7. LITERATŪRA

1. KARELAHTI J. „Solving the cutting stock problem in the steel industry“: magistro darbas. HELSINKI UNIVERSITY OF TECHNOLOGY, Department of Engineering Physics and Mathematics. 2002, p. 5, 6.
[**Žiūrėta** 2005 m. vasario 5 d.]. Prieiga per internetą:
< http://www.sal.hut.fi/Publications/pdf-files/tkar02.pdf >.

2. SWEEP. S. „Three Dimensional Bin-Packing Issues and Solutions“ Midwest Instruction and Computing Symposium: konferencijos pranešimo medžiaga. 2004.
[**Žiūrėta** 2005 m. kovo 3 d.]. Prieiga per internetą:
< http://www.micsymposium.org/mics_2004/sweep.pdf>.

3. PISSINGER D. „Cutting and packing – problem types and lower bounds“ paskaitos konspektas.
[**Žiūrėta** 2005 m. kovo 3 d.]. Prieiga per internetą:
< http://www.diku.dk/undervisning/2005e/426/packing4.pdf>.

4. EGEBLAD J. „Heuristics for packing problems“. „Topics in transportation“ paskaitos konspektas.
[**Žiūrėta** 2005 m. kovo 3 d.]. Prieiga per internetą:
< http://www.diku.dk/undervisning/2005e/426/HeuristicPackings.pdf>.

5. LODI A. „Algorithms for Two-Dimensional Bin Packing and Assignment Problems“: daktaro disertacija. UNIVERSITA DEGLI STUDI DI BOLOGNA, 2000 p.6, 7.
[**Žiūrėta** 2005 m. gegužės 8 d.]. Prieiga per internetą:
< http:/www.dei.unipd.it/~fisch/ricop/tesi/tesi_dottorato_Lodi_1999.pdf>.

6. LANGER Y., BAY M., CRAMA Y., BAIR F., CAPRACE J. D., RIGO P. „Optimization of Surface Utilization Using Heuristic Approaches“: pranešimo medžiaga. University of Liege
[**Žiūrėta** 2006 m. rugsėjo 20 d.]. Prieiga per internetą:
< http://www.anast.ulg.ac.be/files/doc/Publication010.pdf>.

7. DEAN T., BAGGALEY J.N., JAMES R.J.W. „Three Dimensional Container Packing of. Drums and Pallets“ “: pranešimo medžiaga. Department of Management. University of Canterbury
[**Žiūrėta** 2006 m. spalio 18 d.]. Prieiga per internetą:
< http://www.mngt.waikato.ac.nz/orsnz99/PDFs/James.pdf>.

8. ROBERT SEGEDWICK „Algorithms“ ADDISON-WESLEY PUBLISHING COMPANY 1983, 528 p. ISBN 0-201-06672-6

9. J. Harwig , J.W. Barnes „An Adaptive Tabu Search Approach for 2-Dimensional Orthogonal Packing Problems“, Military Operations Research, VII, No. 2, 5-26, 2006.
[**Žiūrėta** 2006 m. lapkričio 9 d.]. Prieiga per internetą:
< http://www.me.utexas.edu/~barnes/research/files/ATS-BP%20Paper%202-25-2004%20harwig.pdf>

10. Fraunhorfer SCAI: Optimization
[**Žiūrėta** 2007 m. kovo 26 d.]. Prieiga per internetą:
< http://www.scai.fraunhofer.de/opt.html?&L=1 >

11. „Microsoft® .NET Kick Start“ Sams Publishing 2003, p.416. ISBN 0-672-32574-8

8. TERMINŲ IR SANTRUMPŲ ŽODYNAS

5 lentelė Terminai

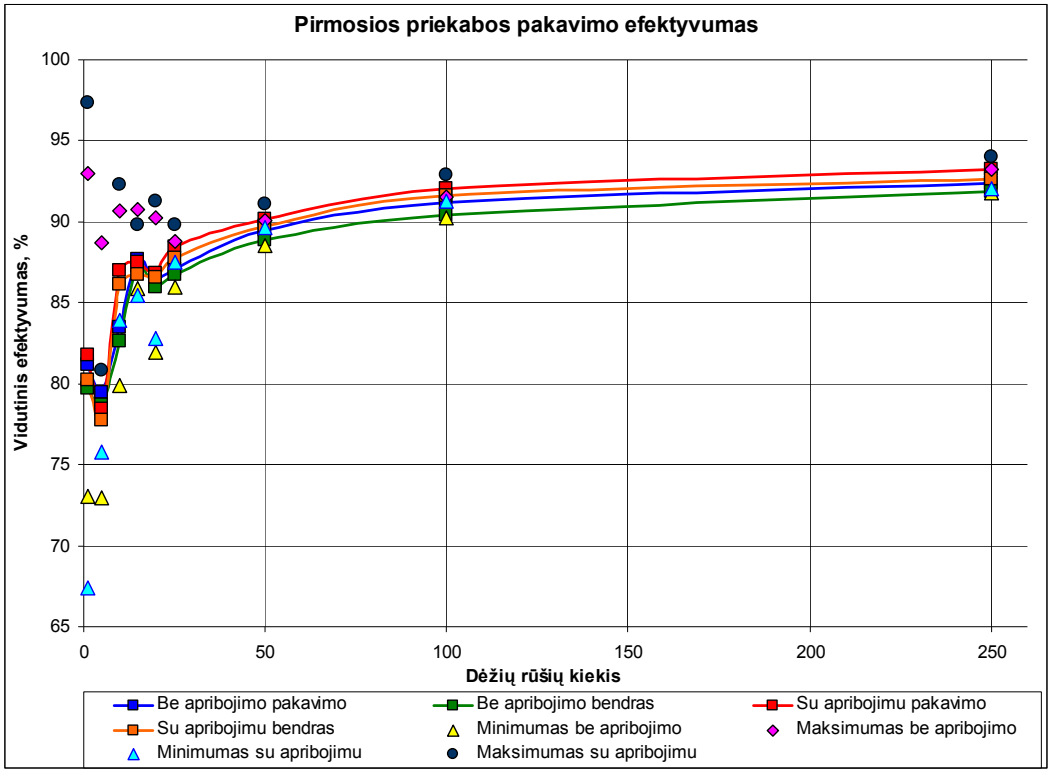
Lietuviškai	Angliškai	Paaiškinimas
Dėžių pakavimo uždavinys	*bin-packing problem*	Turimą dėžių aibę stengiamasi sukrauti į vienodo dydžio konteinerius taip, kad būtų sunaudojama kuo mažiau konteinerių, arba užpildyti vieną konteinerį efektyviausiai.
Pjovimo optimizavimo uždavinys	*cutting stock problem*	Yra aibė objektų, kuriuos reikia išpjauti iš ruošinių – reikia juos išdėlioti taip, kad ruošiniai būtų naudojami efektyviausiai.
Pelno maksimizavimo uždavinys	*knapsack problem*	Turimas riboto dydžio konteineris ir objektų aibė su pelno reitingu ir svoriu. Reikia sudėti dalį objektų taip, kad neviršytų konteinerio maksimalaus svorio ir sudėtų objektų vertė būtų didžiausia.

6 lentelė Santrumpos

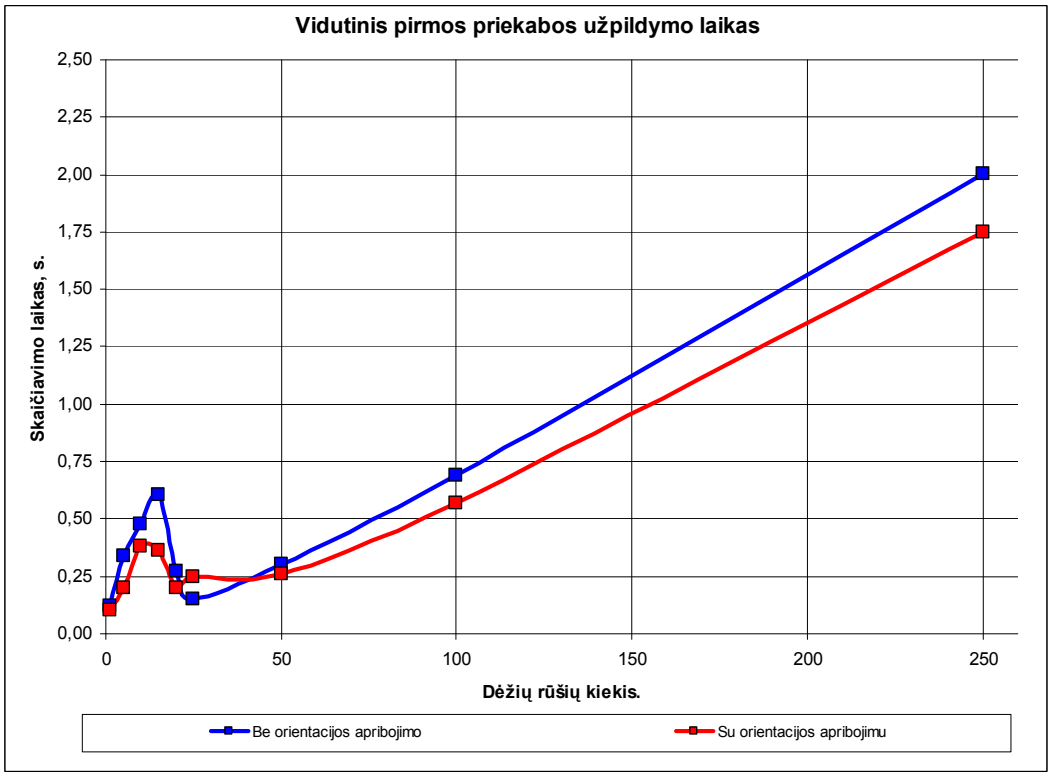
Santrumpa	Pilnas pavadinimas	Paaiškinimas
1D-BPP, BPP	*One dimension bin-packing problem*	Vienmatis dėžių pakavimo atvejis. Objektai turi vieną dimensiją kaip ir konteineriai. Objektus reikia sudėti kuo efektyviau, kad reikėtų mažiau konteinerių.
3D-BPP, BPP	*Three dimension bin-packing problem*	Trimatis dėžių pakavimo atvejis. Objektai turi tris dimensijas (ilgis, plotis, aukštis) kaip ir konteineriai. Objektus reikia sudėti taip, kad reikėtų mažiau konteinerių.
1D-CSP, CSP	*One dimension cutting stock problem*	Uždavinys analogiškas 1D-BPP
NP	*Non-deterministic Polynomial time*	„Neapibrėžtas polinominis laikas“. Tai yra uždavinių klasė, kurios nariai yra išsprendžiami per baigtini polinominį laiką su nedeterminuota Turingo mašina.

9. PRIEDAI

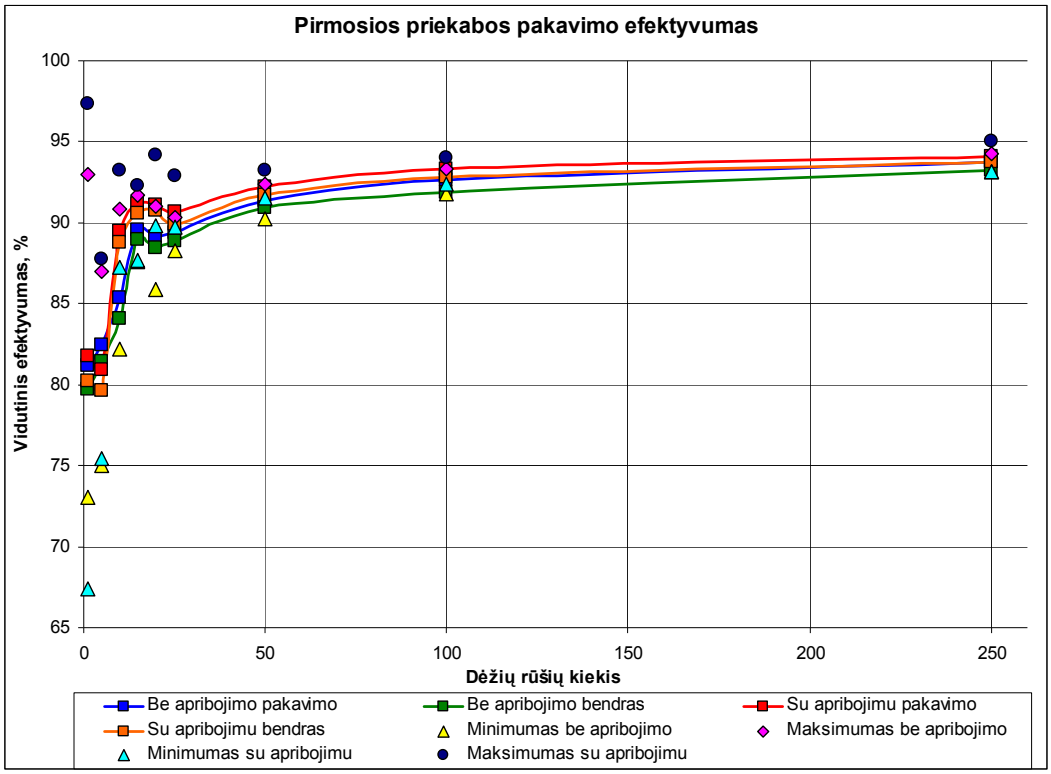
Priedas Nr.1 – pirmos priekabos pakavimo efektyvumas, kai m = 1.



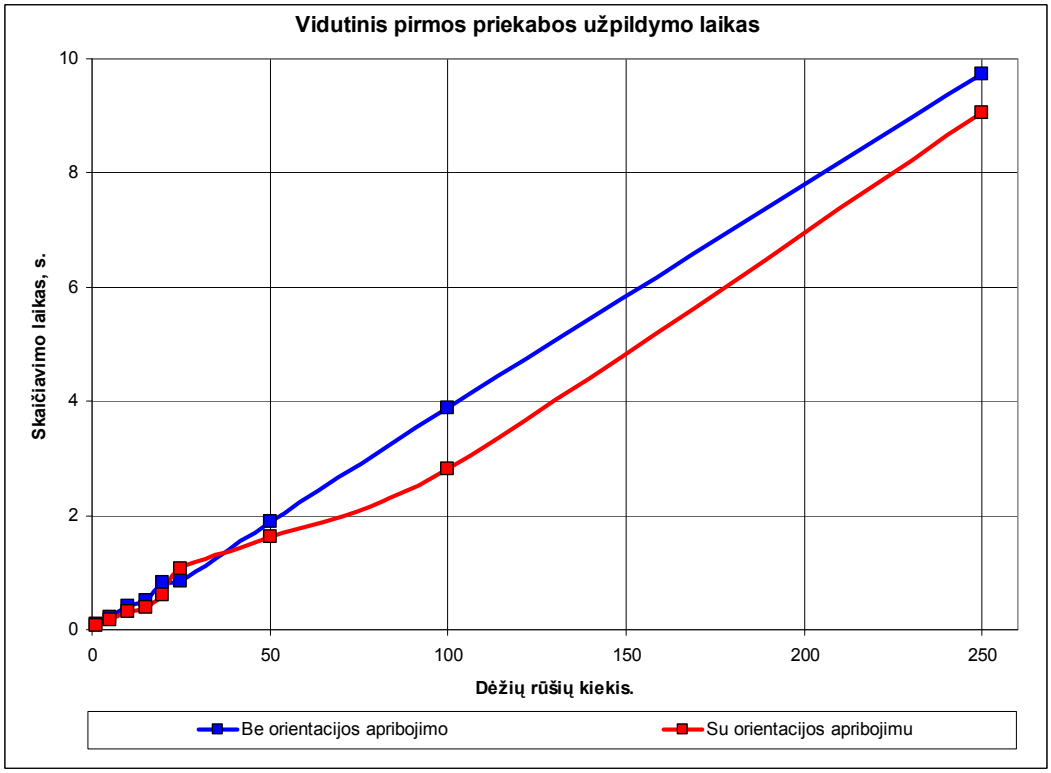
Priedas Nr.2 – pirmos priekabos pakavimo laikas, kai m = 1.



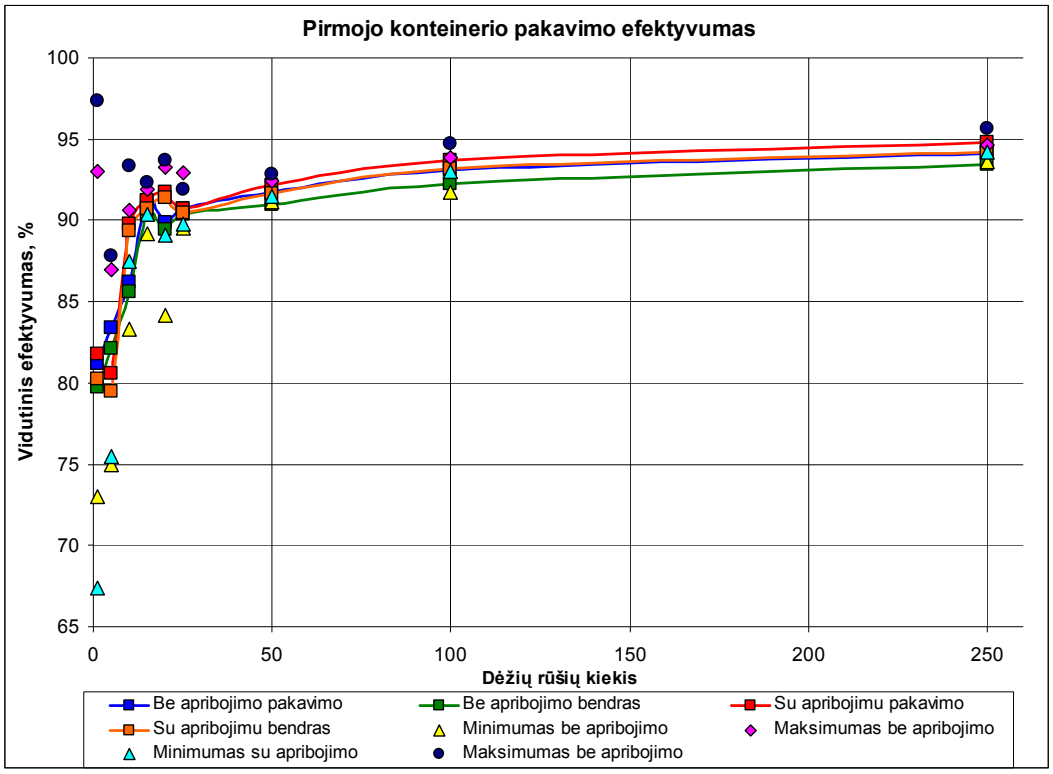
Priedas Nr.3 – pirmos priekabos pakavimo efektyvumas, kai m = 5.



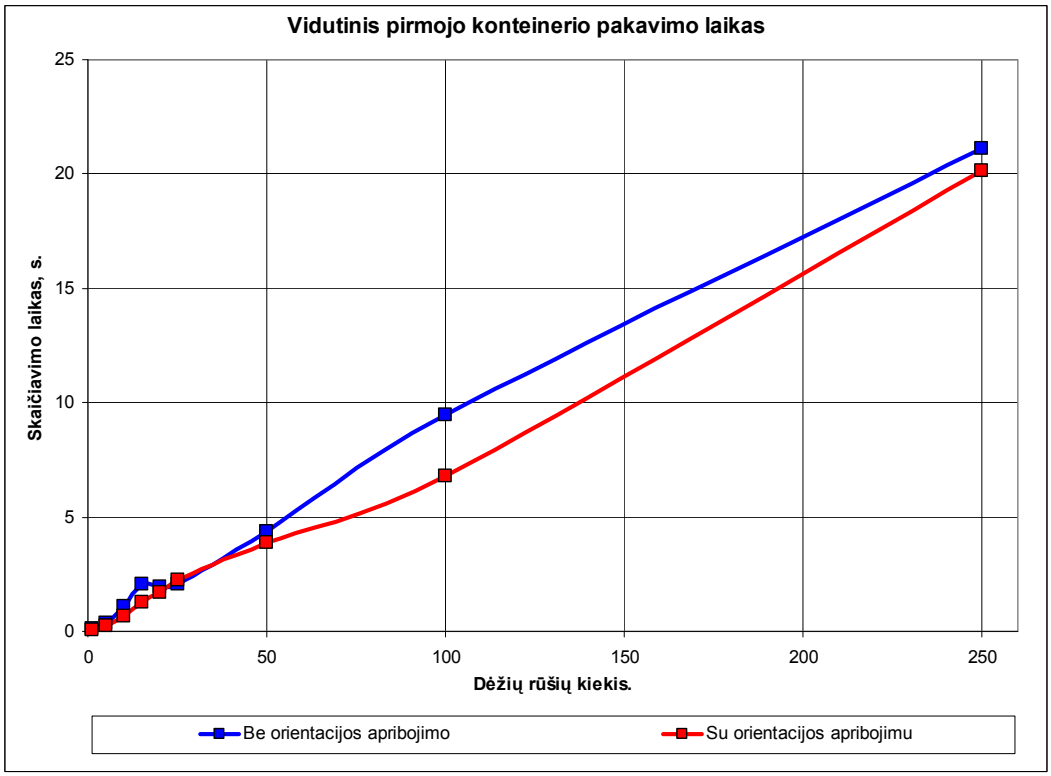
Priedas Nr.4 – pirmos priekabos pakavimo laikas, kai m = 5



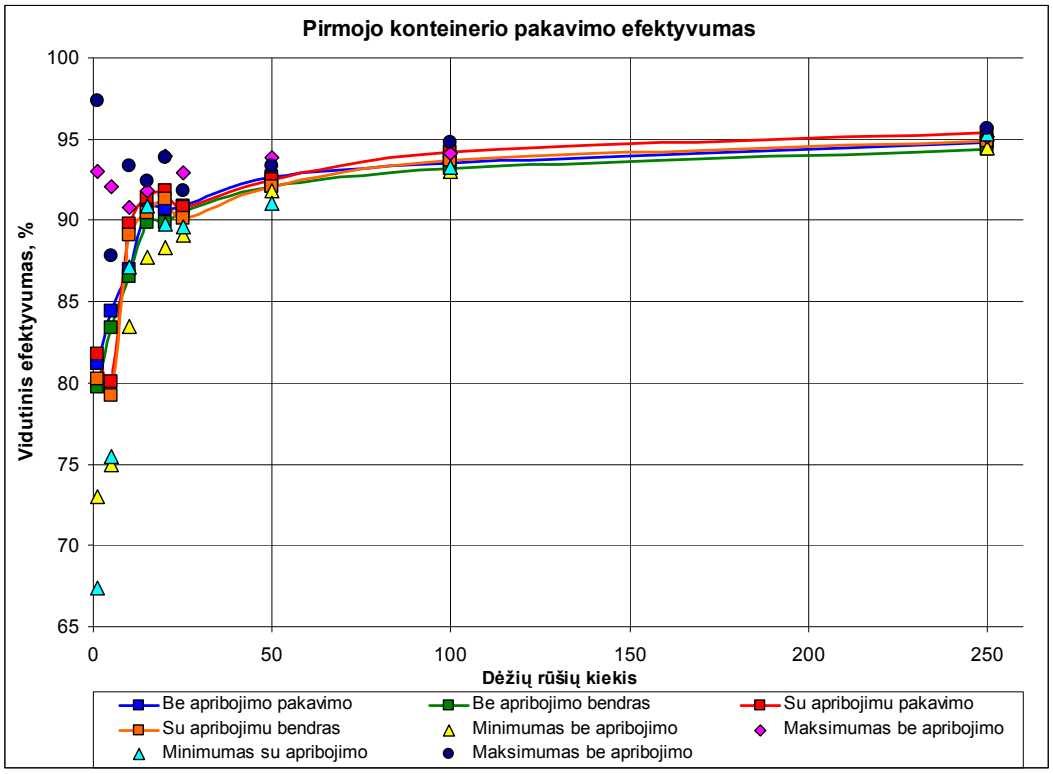
Priedas Nr.5 – pirmos priekabos pakavimo efektyvumas, kai m = 10.



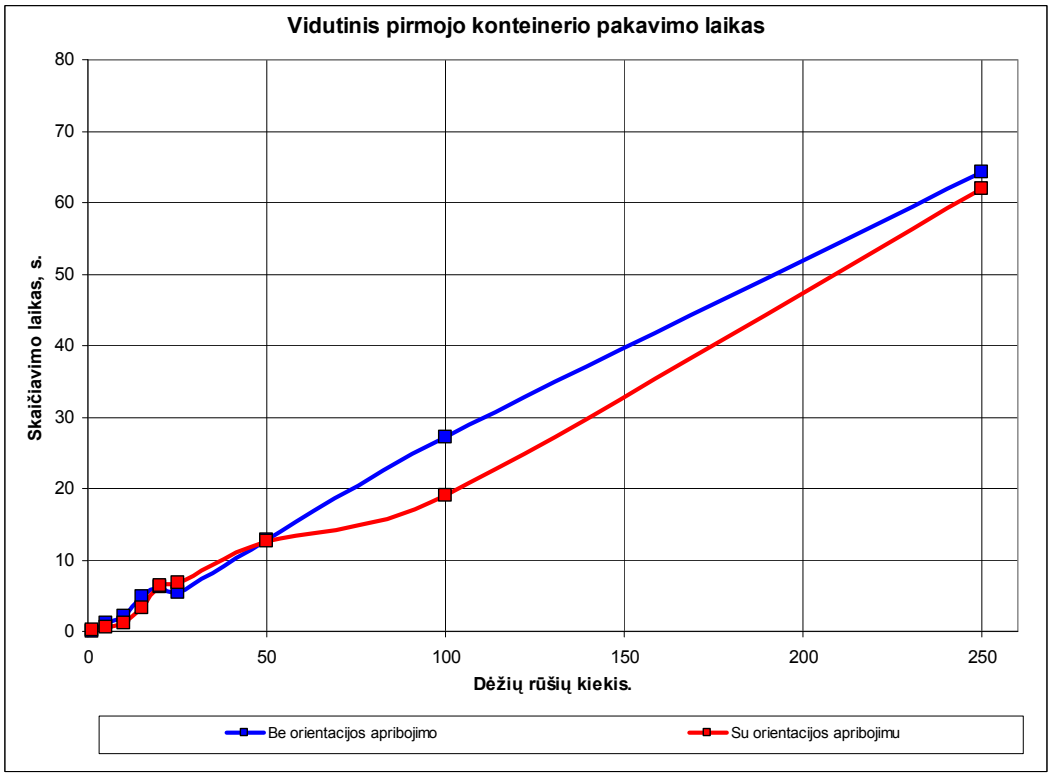
Priedas Nr.6 – pirmos priekabos pakavimo laikas, kai m = 10.



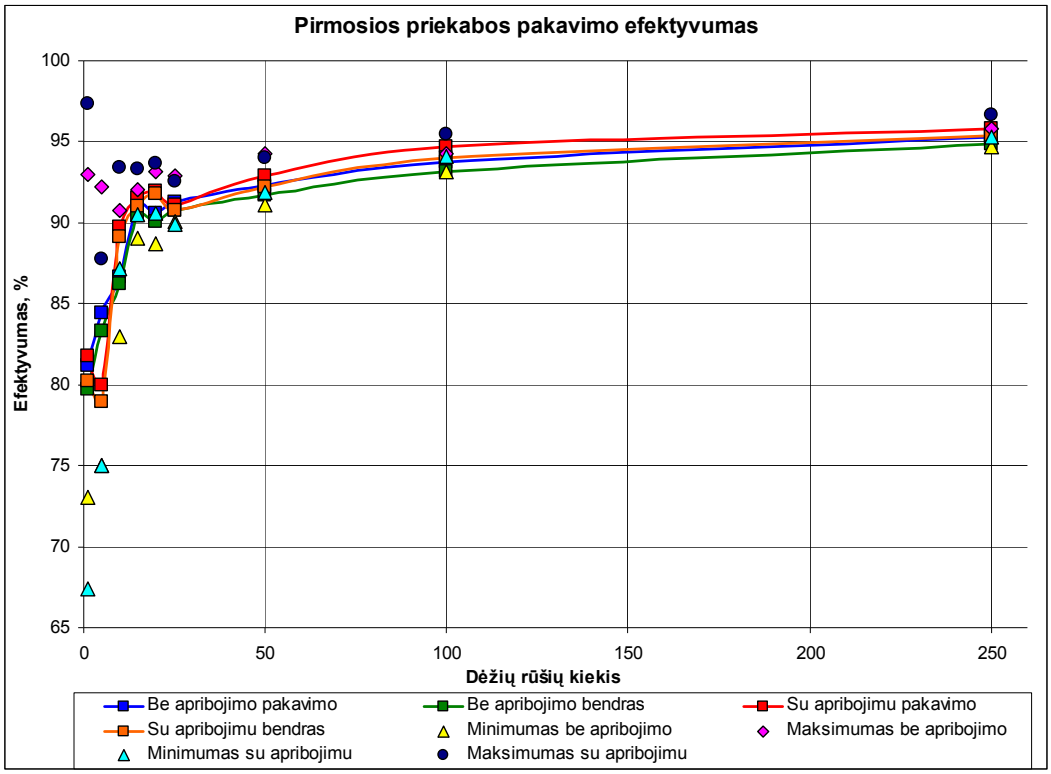
Priedas Nr.7 – pirmos priekabos pakavimo efektyvumas, kai m = 25.



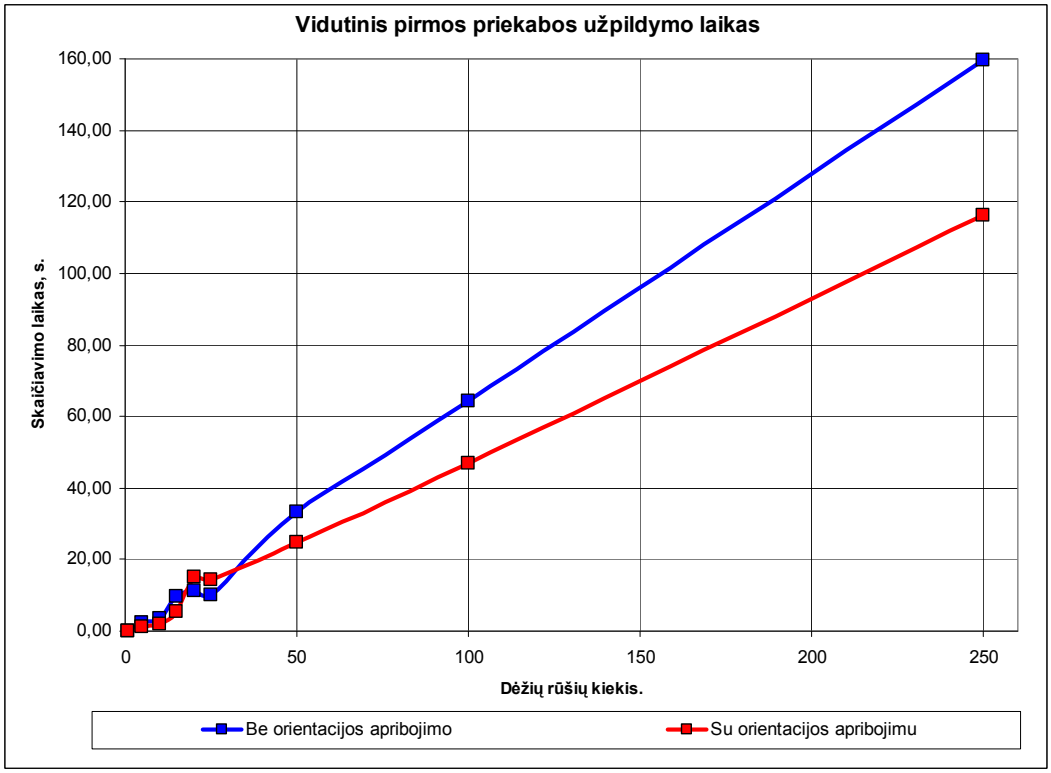
Priedas Nr.8 – pirmos priekabos pakavimo laikas, kai m = 25.



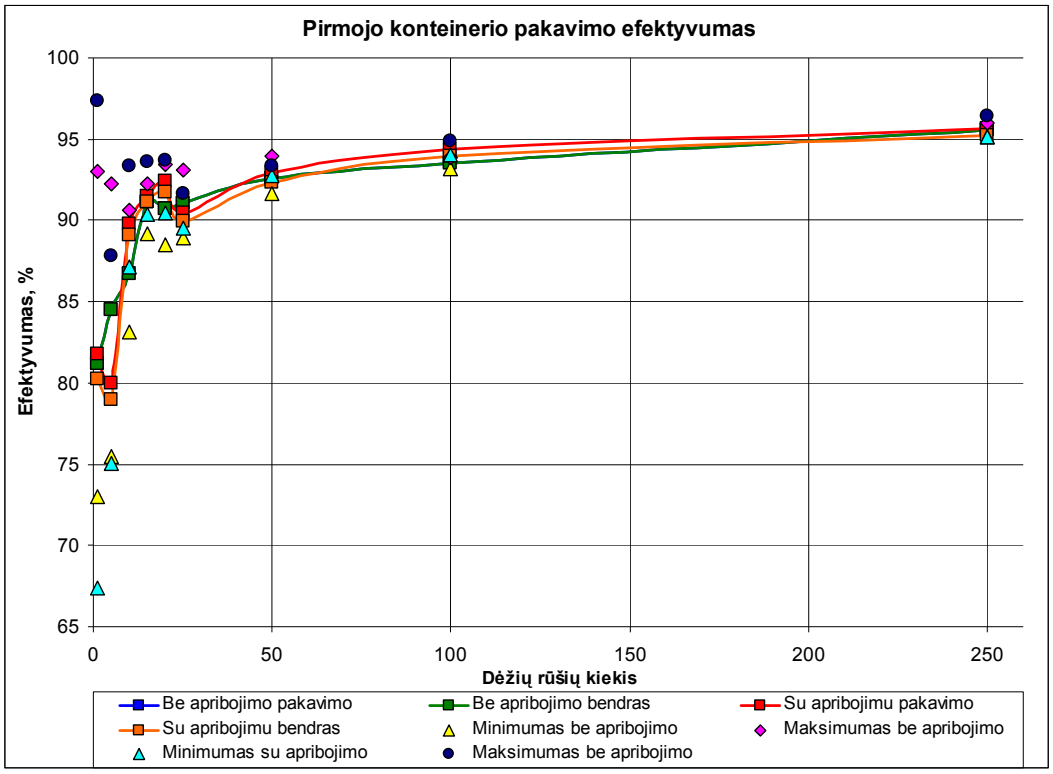
Priedas Nr.9 – pirmos priekabos pakavimo efektyvumas, kai m = 50.



Priedas Nr.10 – pirmos priekabos pakavimo laikas, kai m = 50.



Priedas Nr.11 – pirmos priekabos pakavimo efektyvumas, kai m = 100.



Priedas Nr.12 – pirmos priekabos pakavimo laikas, kai m = 100.