KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Alvydas Muliuolis

TRIMAČIO PAKAVIMO UŽDAVINIO ALGORITMAI IR ANALIZĖ

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 05

TRIMAČIO PAKAVIMO UŽDAVINIO ALGORITMAI IR ANALIZĖ

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
doc. dr. N. Listopadskis
2010 06 03

Recenzentas

2010 06 01

Atliko
FMMM 8 gr. stud.
A. Muliuolis
2010 05 25

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“)

Muliuolis A. Analyze and algorithms of three dimensional packing problem: 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. – 43 p.

SUMMARY

Packing problems are scientifically challenging problems with a wide spectrum of applications. They are very interesting NP-hard combinatorial optimization problems; that is, no procedure is able to exactly solve them in deterministic polynomial time. They are encountered in a variety of real-world applications including production and packing for the textile, apparel, naval, automobile, aerospace, and food industries. They are bottleneck problems in computer aided design where design plans are to be generated for industrial plants, electronic modules, nuclear and thermal plants, and so forth. Packing problems consist of packing a set of geometric objects/items of fixed dimensions and shape into a region of predetermined shape while accounting for the design and technological considerations of the problem. However, the search for exact local extreme is time consuming without any guarantee of a sufficiently good convergence to optimum .

We use several algorithms to solve the bin packing to obtain optimal solution. We also presented programming in C++ implementation of these meta-heuristic algorithms and showed their results. At the end of this paper we tell the decision experimentation result.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Pakavimo problemos yra sudėtingos ir reikalaujančios daug pastangų, jų taikymo sfera labai plati. Tai kombinatorinio optimizavimo problemos keliančios mokslinius iššūkius. Skaičiavimo sudėtingumo teorijoje dėžių pakavimo uždavinys priskiriamas prie potencialiai „neišsprendžiamų“ uždavinių, kurie priklauso kombinatorinio NP – sudėtingo matematinių problemų aibei. Nėra tokios procedūros kuri galėtų gauti tikslų sprendinį per priimtiną polinominį laiką.

Egzistuoja labai įvairių pakavimo uždavinio modifikacijų. Realiame pasaulyje dažniausiai ši problema taikoma logistikoje, tekstilės, drabužių, jūrų, automobilių, kosminės aviacijos ir maisto pramonėse. Pakavimo ir pjaustymo problemos yra glaudžiai susijusios ir vienos ar dviejų dimensijų atveju identiškos. Norint, kad minėtose srityse būtų efektyviai sumažinti atliekų kiekiai, transportavimo kaštai ir efektyviau panaudoti turimi resursai taikomi pakavimo optimizavimo uždaviniai. Pakavimo problemos identifikuojamos pagal geometrinio matavimo dimensijas, pakuojamų objektų aibės savybes ir kitų papildomų sąlygų. Šiame darbe bus nagrinėjama dėžių pakavimo problema. Kuri apibrėžiama taip: duota stačiakampio gretasienio formos mažų objektų (dėžių) aibė, tikslas jas taip supakuoti į vienodo dydžio didelius objektus (konteinerius), kad jų kiekis būtų minimalus. Tokią problemą vadinsime vienodo dydžio trimačių konteinerių pakavimo problema. Pateiksime autorių Gerhard Wäscher, Heike Haußner, Holger Schumann [2] siūlomą pakavimo problemų tipologiją ir apžvelgsim pagrindinius aspektus. Remiantis minėta tipologija darbe nagrinėjamą problemą sutrumpintai žymėsime 3D-SBSBPP.

Konteinerių pakavimo problemos buvo pagrinde nagrinėjamos dviejų matavimų atveju. Ir tik gana neseniai buvo susitelkta į trijų matavimų problemas. Yra svarbūs šių autorių darbai A. Lodi, S. Martello, D. Vigo [6], G. Perboli [7] ir O. Faroe, D. Pisinger,M. Zachariasen [9]. Verta pažymėti, kad dauguma atvejų trimatis uždavinio sprendimas išplečiamas į trimatį.

Kadangi tai NP – sudėtingumo problema, todėl dažniausiai jai spręsti naudojami euristiniai algoritmai. Tad daugeliu atveju kompiuteris gali rasti pakankamai gerą sprendinį per priimtiną laiko tarpą ir toks sprendinys gali būti taikomas praktikoje. Algoritmų sprendinio optimalumui nustatyti naudosime kriterijus (apatines ribas) pasiūlytus Marco A. Boschetti. [5]. Detaliai ištirsime du algoritmus:

- Dviejų žingsnių šakos ir ribos (S.Martello, D. Prisinger, D. Vigo [4]);
- Tabu paieškos (A. Lodi, S. Martello [6]).

Minėti algoritmai realizuoti C++ kalba. Pateiksime algoritmų analizės rezultatus ir išvadas.

1. ANALITINĖ DALIS

1.1. PAKAVIMO PROBLEMŲ TIPOLOGIJA

Tipologija yra metodinis objektų suskirstymas į homogenines grupes remiantis duota charakterizavimo savybių aibe. Tai praktiškai orientuotas metodas, kurio prioritetas skirtas nagrinėti svarbiausias tyrimo objektų savybes ir tik tada mažiau svarbesnes hipotetines detales. Tokiu būdu sudaromas sudėtinis problemos vaizdas, tai padeda praktiniuose tyrimuose. Be to tipologija padeda suvienyti apibrėžimus ir pažymėjimus, taip palengvina nustatyti ryšį tarp mokslinių tyrimų. Pakavimo problemos topologija suteikia pagrindus: struktūrinei problemos tipų analizei, identifikacijai ir pagrindinių problemų formulavimui, modelių ir algoritmų vystymui, problemos generavimui ir t.t. Pakavimo problemos struktūra gali būti apibendrinama tokiu būdu:

duota dvi elementų aibės, būtent

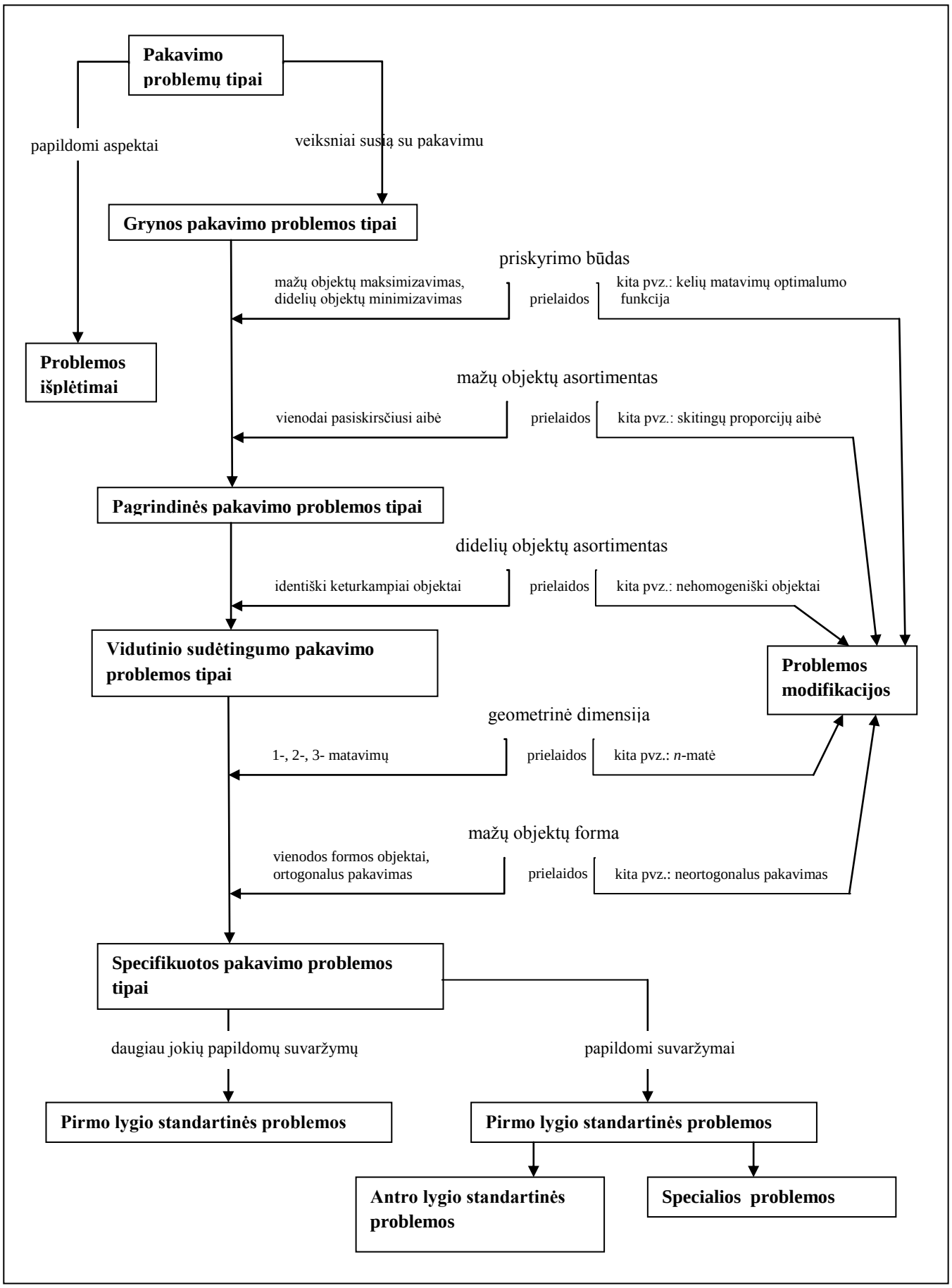
- didelių objektų (įvestis, pasiūla) ir
- mažų objektų (išvestis, paklausa),

kurios yra išsamiai apibūdinamos vienmatėje 1D, dvimatėje 2D, trimatėje 3D ar net didesnio skaičiaus *(n)* geometrinio matavimo dimensijoje. Paimkime keletą ar visus mažus objektus, sugrupuokime juos į vieną ar keletą poaibių ir priskirkim kiekvieną iš gautų poaibių vienam iš didelių objektų taip, kad geometrinės sąlygos būtų tenkinamos ir duota (vienos dimensijos ar daugelio dimensijų) tikslo funkcija būtų optimizuota. Sprendinys gali naudoti kelis ar visus didelius objektus, ir kelis ar visus mažus objektus. Formaliai sprendžiamos tokios mažesnės problemos:

- didelių objektų parinkimo problema
- mažų objektų parinkimo problema
- mažų objektų grupavimo problema
- mažų objektų poaibių paskirstymo tarp didelių objektų
- mažų objektų išdėstymo kiekviename iš pasirinktų didelių objektų remiantis geometrinėmis sąlygomis

Specifiniai pakavimo problemos tipai charakterizuojami į vedant papildomų sąlygų. Vienas iš pirmųjų autorių nagrinėjas pakavimo uždavinių tipologija buvo Dyckhoff [1].

Kadangi pakavimo problemų nagrinėjimas sparčiai plėtojosi tai Dyckhoff pasiūlyta klasifikacija neatitiko keliamų problemų, todėl autoriai Gerhard Wäscher, Heike Haußner, Holger Schumann [2] išplėtojo pakavimo uždavinių tipologiją. Jų pasiūlytas bendras pakavimo problemų tipų supratimas pateiktas 1.1 paveiksle.



1.1 pav. Pakavimo problemos tipų apžvalga.

Geometrinė dimensija

Yra išskiriama trys geometriniai matavai: vienmatis, dvimatis ir trimatis. Literatūroje [3] galima rasti problemų kurios nagrinėjamos $n > 3$ dimensijoje.

Priskyrimo būdas

- Išvesties maksimizavimas

  Mažų objektų aibė yra priskiriama duotai didelių objektų aibei. Didelių objektų aibė yra nepakankama, kad sutalpintų visus mažus objektus. Taigi nėra didelių objektų pasirinkimo problemos, bet sprendžiama mažų objektų poaibių maksimizavimo problema.
- Įvesties minimizavimas

  Vėl reikia priskirti mažų objektų aibe dideliems objektams, bet kitaip nei išvesties maksimizavimo atveju dideli objektai talpina visus mažus. Visi maži objektai turi būti priskirti dideliems, sprendžiama didėlių objektų minimizavimo uždavinys.

Čia įvesties maksimizavimas ir išvesties minimizavimas naudojamas pagrindines prasme (objektų kiekybės), o ne kokia nors specialiąja. Kai norime traktuoti kokiu nors išskirtiniu atveju tada objekto vertė turi būti apibrėžta labiau specifikuotai ir gali būti apibūdinama kaip kaina, medžiagos kiekis ar kaip nors kitaip. Dažnai tiek mažų tiek didelių objektų vertė gali būti traktuojama tiesiogiai proporcingai jų matmenims. Tada tikslo funkcija minimizuoja ar maksimizuoja ilgį(viena dimensija), plotą(dvi dimensijos), tūrį(trys dimensijos).

Mažų objektų asortimentas

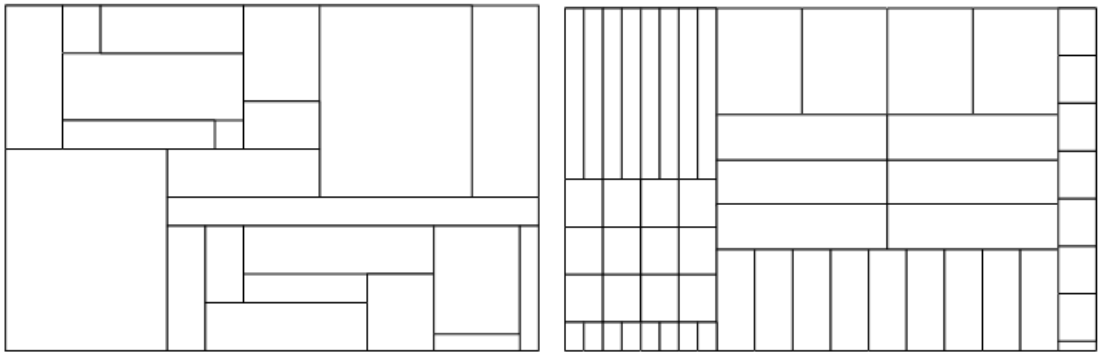
- Identiški maži objektai

  Visi objektai yra identiški tiek forma tiek matmenimis, gali skirtis tik jų vertė. Išvesties maksimizavimo atveju jų kiekis gali būti begalinis.
- Silpnai heterogeninis asortimentas

  Maži objektai gali būti santykinai suskirstyti į keletą klasių, kuriose jie yra identiški dydžiu ir forma. Apbrėžiant mažus objektus identiškos formos ir dydžio, kurie reikalauja skirtingos orientacijos, ji laikomi skirtingos rūšies. Kiekvienų objektų kiekis yra santykinai didelis ir gali begalinis.
- Stipriai heterogeninis asortimentas

  Mažų objektų aibė charakterizuojama taip kad tik labai mažas kiekis elementų yra identiško dydžio ir formos. Taigi elementai traktuojami kaip individualūs objektai. Kiekvieno iš lentų pasiūla lygi vienam.

Apibrėžiant standartines problemas laikomės prielaidos, kad visi maži objektai yra vienodai pasiskirstę. Tai yra nėra taip , kad vienos rūšies objektų yra didelė pasiūla, o kitų maža.



1.2 pav. Stipriai ir silpnai heterogeninis asortimentas

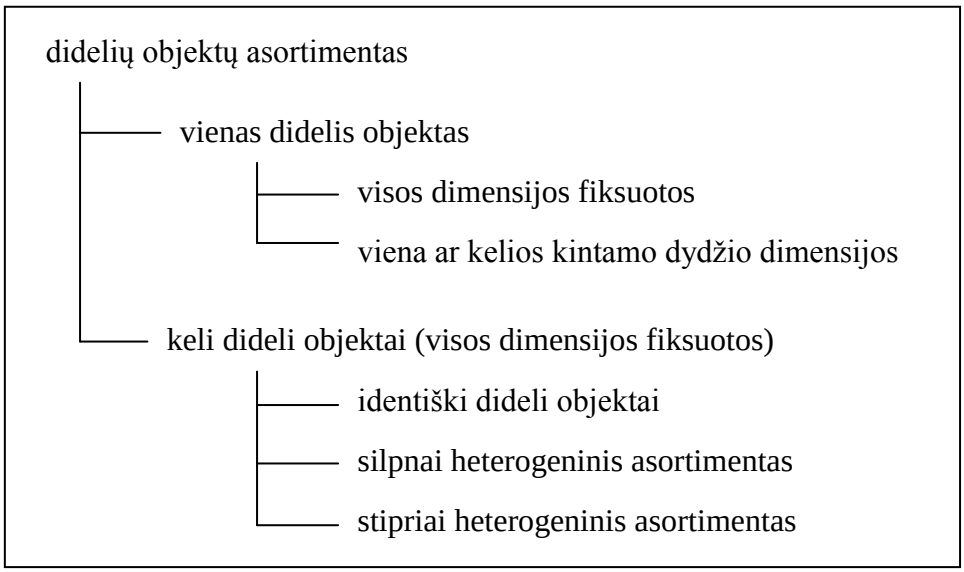
Didelių objektų asortimentas

- Vienas didelis objektas

  Pakavime naudojamas tik vienas didelis objektas į kurį talpiname kitus mažesnius. Konteinerio matmenys gali būti arba fiksuoti arba gali prasitęsti kuria nors kryptimi priklausomai nuo dimensijos ir problemos formulavimo.

- Keli dideli objektai

  Kai didelių objektų kiekis yra didesnis nei vienas tada visi objektai turi fiksuoto dydžio matmenis. Jų matmenys gali būti identiški, silpnai ar stipriai heterogeniniški.



1.3 pav. Galimas didelių objektų asortimentas.

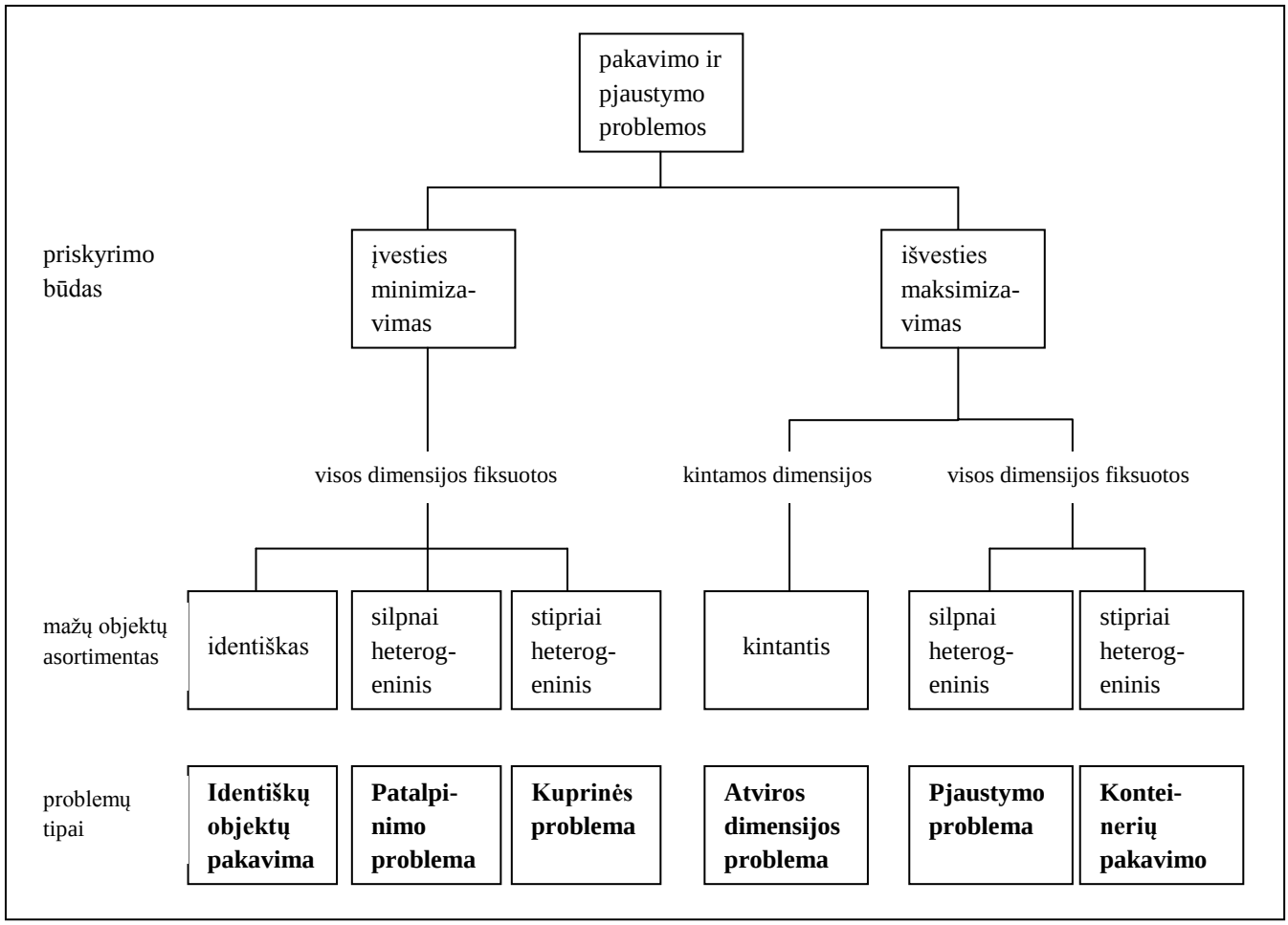
Apibrėžiant pagrindinius pakavimo problemos tipus yra laikoma, kad vienos, dviejų ir trijų dimensijų atveju – visi objektai yra keturkampės formos.

Mažų objektų forma

Pakavimo problemos tipai nagrinėjami kai maži objektai yra standartiniai(keturkampiai, apskritimai, dėžės, cilindrai, sferos ir t.t.) ir nestandartiniai. Literatūroje dažniausiai nagrinėjami atvejai kai keturkampiai objektai yra pakuojami ortogonaliai. Be to nagrinėjama tik toki atvejai kai objektu yra tik reguliarūs arba tik nereguliarūs. Problemos kai leidžiamas neortogonalus pakavimas ir maišyto tipo pakuojami objektai kol kas nebuvo nenagrinėjamos.

1.2. PAGRINDINĖS, VIDUTINIO SUDĖTINGUMO IR SPECIFIKUOTOS PAKAVIMO PROBLEMOS.

Pagrindinai pakavimo problemos tipai yra vystomi kombinuojant du kriterijus tai *priskyrimo tipą* ir *mažų objektų asortimentą*. 1.3 paveiksle pateikiam problemos tipų schema. Apžvelgsime visus tipus detaliau.



1.4 pav. Pagrindiniai pakavimo problemos tipai

Išvesties maksimizavimas

Išvesties maksimizavimo problema seka iš to, kad yra duotas ribotas kiekis didelių objektų ir į juos negalima patalpinti visų pakuojamų objektų. Reikalingas pakavimas, kad mažų objektų tikslo funkcija butų maksimizuota ir panaudojami visi dideli objektai. Kitaip tariant sprendžiama mažų objektų parinkimo problema.

- Identiškų objektų pakavimo problema

Ši problema susideda iš to, kad reikia priskirti didžiausią įmanomą kiekį mažų objektų, duotam ribotam didelių objektų kiekiu. Kadangi visi pakuojami objektai yra identiški tai jokios realios parinkimo ar netgi grupavimo problemos nėra. Pakavimo problema suvedama į mažų objektų išdėstymą remiantis tam tikra nustatyta tvarka remiantis geometrinėmis pakavimo sąlygomis.

- Patalpinimo problema

Literatūroje ši problema žinoma daugybe įvairių vardų. Norint išvengti papildomų neaiškumų terminą *patalpinimo problemą* apibrėšime kaip pakavimo problemos kategoriją kuri nagrinėja silpnai heterogeninių objektų priskyrimą duotai ribotai didelių objektų aibei. Bendra vertė supakuotų objektų turi būti maksimizuota arba alternatyviai nepanaudotos pakavimo vietos minimizavimas.

- Kuprinės problema

Remiantis autorių Gerhard Wäscher, Heike Haußner, Holger Schumann [2] interpretacija, kuprinės uždavinio problema pristato pakavimo problemų kategoriją kurios pagrindinė charakteristika yra stipriai heterogeniškų mažų objektų pradinė aibė. Kadangi yra ribota didelių objektų aibė tai nevisi pakuojami objektai atrenkami. Reikia maksimizuoti patalpintų objektų vertę, čia objektų vertė suprantama ne trivialiu tūrio mato, o labiau specifikuotu matu.

Įvesties minimizavimas

Įvesties minimizacija charakterizuojama tuo, kad yra toks kiekis didelių objektų jog gali sutalpinti visus pakuojamus objektus. Taigi nėra jokios mažų objektų pasirinkimo problemos. Dideli objektai turi talpinti visus mažus objektus taip, kad didelių objektų vertė būtų minimali.

Atviros dimensijos problema

Atviros dimensijos problema apibrėžia kategoriją kurioje vis maži objektai yra pakuojami į viena didelį talpinantį objektą. Duoto objekto nevisi matmenys yra fiksuoti, galimas jų begalinis išplėtimas bet vienai geometrinei dimensijai. Reikia minimizuoti didelio objekto vertę, priklausomai nuo

dimensijos tai gali būti ilgis, plotas, tūris. Apibrėžiant pagrindines pakavimo problemas apsiribojama tik fiksuotų matmenų dideliais objektais.

Konteinerių pakavimo uždavinys

Yra duota stipriai heterogeninė aibė mažų objektų. Ją reikia patalpinti į identiško didelių objektų aibę, silpnai heterogonišką arba stipriai heterogonišką didelių objektų aibę. Vertė, kiekis ar bendras dydis didelių konteinerių reikalingų sutalpinti visus pakuojamus objektus turi būti minimizuojamas.

Vidutinio sudėtingumo pakavimo problemos tipai

Norint apibrėžti labiau homogeniškų problemų tipus pagrindiniai problemų tipai išplėtojami pridedant didelių objektų pasirinkimo asortimentą kaip papildomą kriterijų. 1.1 lentelėje charakterizuojamos vidutinio sudėtingumo problemos skirtos išvesties maksimizavimui, o 1.2 lentelėje charakterizuojamos vidutinio sudėtingumo problemos susijusio su įvesties minimizavimu. Tai pat šiuose paveikslėliuose ir pateikiami autorių Gerhard Wäscher, Heike Haußner, Holger Schumann [2] siūlomi pavadinimai.

1.1 lentelė

Vidutinio sudėtingumo problemos: išvesties maksimizavimas

<table>
<tr><td colspan="2">mažų objektų asortimentas
didelių objektų charakteristikos</td><td>identiški</td><td>silpnai heterogeniniai</td><td>silpnai heterogeniniai</td></tr>
<tr><td rowspan="3">visos dimensijos fiksuotos</td><td>vienas didelis objektas</td><td>identiškų objektų pakavimo problema

IIPP</td><td>vieno didelio objekto užpildymo problema

SLOPP</td><td>vienos kurpinės problema

SKP</td></tr>
<tr><td>identiški</td><td rowspan="2"></td><td>kelių identiškų didelio objektų užpildymo problema

MILOPP</td><td>kelių identiškų kurpinių problema

MIKP</td></tr>
<tr><td>heterogeniniai</td><td>kelių heterogeniniu didelio objektų užpildymo problema

MHLOPP</td><td>kelių heterogeniniu kurpinių problema

MIKP</td></tr>
</table>

1.2 lentelė

Vidutinio sudėtingumo problemos: įvesties minimizavimas

mažų objektų asortimentas / didelių objektų charakteristikos		stipriai heterogeniniai
visos dimensijos fiksuotos	identiški	vienodo dydžio konteinerių pakavimo problema SBSBPP
	silpnai heterogeniniai	kelių dydžio konteinerių pakavimo problema MBSBPP
	stipriai heterogeniniai	nepanaudotos laisvos vietos konteinerį problema RBPP
vienas didelis objektas, kintančios dimensijos		atviros dimensijos problema ODP

1.3. TRIMAČIO KONTEINERIO PAKAVIMO PROBLEMOS FORMULAVIMAS

Yra duota aibė n stačiakampės formos dėžių, kiekviena iš jų charakterizuojama pločiu w_j, aukščiu h_j ir ilgio d_j ($j \in J = \{1, \dots, n\}$), ir neribotas skaičius identiškų trimačių konteinerių turinčių plotį W, aukštį H ir ilgį D. Trimačio konteinerio pakavimo problema apibūdinama taip: reik patalpinti visą dėžių aibę į minimalų konteinerių kiekį. Kol kas įveskime vienintelį ribojimą, kad dėžės turi būti pakuojamos į konteinerius ortogonaliai. Tai pat laikysime neprarasdami bendrumo, kad visi įvedami duomenys t.y. matmenys yra teigiami sveikieji skaičiai.

Pakavimo problemos tikslo funkcija

$$f = \frac{\sum_{j=1}^{n} w_j d_j h_j}{n_{konteineri\ ų} \times WDH}$$

Pakavimas yra vadinamas galimu jai tenkinamos tokios sąlygos:

- kiekviena dėžė patalpinta vidurį konteinerio;

$$0 \le x_j \le D - d, \qquad j = 1, \dots, n$$

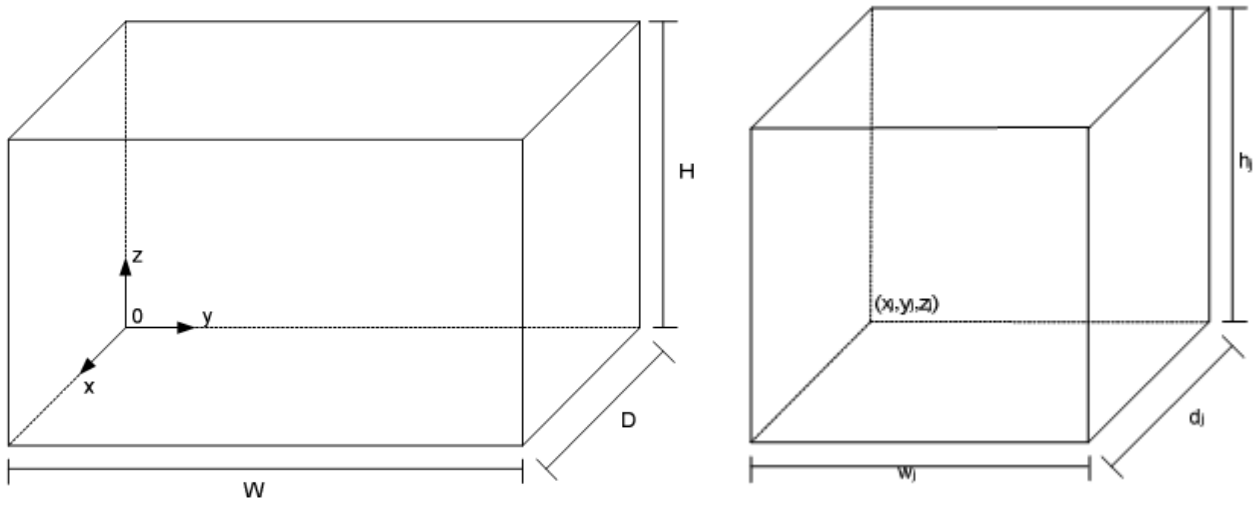
$$0 \le y_j \le W - w_j, \qquad j = 1, \dots, n$$

$$0 \le z_j \le H - h_j, \qquad j = 1, \dots, n$$

- dėžė *i* nepersidengia su dėže *j*;

$$x_i \geq x_j + d_j \ \vee \ y_i \geq y_j + w_j \ \vee \ z_i \geq z_j + h_j \ \vee \ x_j \geq x_i + d_i \ \wedge$$
$$y_j \geq y_i + w_i \ \vee \ z_j \geq z_i + h_i, \quad i, j = 1, \dots, n$$

- kiekviena dėžė patalpinta lygiagrečiai konteinerio sienom.



1.5 pav. Konteineris ir dėžė.

Pakavimo uždavinio modifikacijos:

Nedeterminuotas(On-line) 3D-BPP – iš anksto nieko nežinoma apie pakuojamos aibės *S* objektus. Imama pirma dėžė iš *S* ir dedama į konteinerį, nežinoma koki bus kitos dėžės matmenis. Patalpinti objektai negali būti iš naujo perpakuojami viename konteineryje ar per renkama visų pildomų konteinerių aibė.

Determinuotas(Off-line) 3D-BPP – visa informacija žinoma apie pakuojamus objektus $a_1 \in S$, leistinas naujo objekto patalpinimo metu bet koks konteinerių parinkimas.

Ortogonalsu 3D-BPP – tai apribojimas kuris leidžia dėžes patalpinti tik lygiagrečiai konteinerio sienoms.

3D-BPP su posūkiais – leista pakuojamų objektų pasukimas arba apvertimas. Klasikiniu atveju talpinami objektai sukami tik ortogonaliai, nors galimas ir bet kokio kampo posūkis.

Yra išskiriama ir kitokių ribojimų:

- ribojamas objektų, dedamų ant kitų viršaus, svoris;
- reikalaujama, kad būtų pakuojamo konteinerio (autofurgono) ratų ašių svorio balansas;
- gali reikėti į konteinerį patalpinti tam tikrą kai kurios rūšies objektų kiekį;
- kartais supakuotus objektus reikalaujama iškrauti skirtingose vietose, todėl jie pakuojami tam tikra eilės tvarka, kad būtų galima juos išimti iš konteinerio reikiamu laiku.

Atsižvelgiant į trimačio pakavimo uždavinio galimas modifikacijas ir įvairius ribojimus galima sukonstruoti pakavimo uždavinį atspindintį realaus pasaulio pakavimo problemas. Sprendinio optimalumui patikrinti naudosime taip vadinamą apatinę ribą, kurią apibrėšime metodologinėje dalyje.

1.4. ALGORITMAI SKIRTI 3D-SBSBPP

Sprendžiant pakavimo problemas buvo atlikta daugybė tyrimų ypač vienos arba dviejų dimensijų atvejams. Šiuo metu didelis dėmesys yra skiriamas 3D objektų pakavimo problemas sprendžiančių efektyvių modelių kūrimui ir vystymui. Yra taikomi toki pakavimo problemos sprendimo metodai:

- *Algoritminiai* metodai garantuoja, kad bus surastas optimalusis sprendinys. Tačiau didelis šių metodu trūkumas yra skaičiavimų sudėtingumas, todėl yra labai ilga sprendinio ieškojimo trukme, ypač dideles apimties uždaviniams. Praktikoje jie nėra plačiai taikomi.
- *Euristiniai* metodai retai randa optimalųjį sprendinį, bet paprastai gana greitai randa priimtiną rezultatą, kuris yra pakankamai arti optimaliojo sprendinio. Trukumas 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 kokybe.
- *Aproksimavimo* metodai yra tokie, kuriu grąžinamas sprendinys yra gaunamas aproksimuojant optimalųjį sprendinį. Tad skiriasi tam tikru, nedideliu skaičiumi. Nors aproksimaciniai metodai ir negarantuoja jokio pakankamai optimalaus sprendinio, tačiau yra žymiai greitesni negu algoritminiai, taip pat jų naudojamas idėjas yra lengviau modifikuoti atsiradus papildomiems apribojimams.

George ir Robinson (1980) yra pirmieji anglų mokslininkai pradėję tyrinėti objektų pakavimo problemas. Autoriai tiksliai įvardijo konteinerių pakrovimo problemą, smulkiai aprašydami įvairaus kiekio ir skirtingų dydžių dėžėmis konteinerio pakrovimo algoritmą. Jie panaudojo pakankamai sudėtingą sienų rentimo būdą, kuriame konteinerio dalys yra pilnai užpildomos atsižvelgiant į pločio ir aukščio matus. *Algoritmo idėja* – padalinti konteinerį į kelis lygiagrečius sienoms sluoksnius ir juos užpildyti atskirai. Toks būdas daugeliu atvejų garantuoja, kad taip supakuoto krovinio didesnė dalis bus stabili. George ir Robinson sukurtas algoritmas pakuoja iš 20 dėžių susidedantį krovinį. Pakuojant skirtingų dydžių dėžes ir naudojant skirtingas jų padėtis, gali būti gaunami skirtingi pakavimo variantai. Kai naudojant to pačio tipo dėžę nepavyksta užpildyti sieną, sienos viršuje ir dešinėje generuojami tarpai, juos užpildant tarpų pildymo procedūra. Metodas visada stengiasi plokštumą išlaikyti pakuojamo paviršiaus viršuje. Procedūra stengiasi neišskirti panašaus tipo dėžių, apibrėžiant

dėžių tipus: 'atidaryta' ir 'uždaryta'. Taip pat procedūra skaičiuoja dėžių padėties apribojimus. Atidaryto tipo dėžės – tai būtini dėžės tipai, kuriais visada pradedamas pakavimas, tuo tarpu uždaryto tipo dėžės – nebūtini pakavimui dėžių tipai.

George ir Robinson sukurto algoritmo privalumai:

- gali supakuoti įvairaus kiekio ir skirtingų dydžių dėžes;
- naudojama tarpų pildymo procedūra;
- procedūra skaičiuoja dėžių padėties apribojimus;
- procedūra garantuoja supakuoto krovinio didesnės dalies stabilumą;
- nurodo dvi alternatyvias atvirų dėžių tipo pasirinkimo taisykles.

Trūkumai:

- sukurtas algoritmas efektyviai (per trumpą laiką ir pan.) gali supakuoti ne daugiau kaip iš 20 dėžių susidedantį krovinį;
- nepateikia tikslių detalių, kurią pasirinkus atvirų dėžių tipo taisyklę bus gaunami geresni rezultatai.

Han, Knott ir Egbelu (1989) teigia, jog sienų idėja neturi būti apribota vertikaliomis konteinerio kraštinėmis. Autoriai apibūdina algoritmą, kuriame konteineris (didžioji prizmė) pakuojamas identiškomis dėžėmis (mažiausiomis prizmėmis). Apibūdintas algoritmas yra projektuotas atskiram dėžės tipui, kuris yra pastovus dydžio ir formos atžvilgiu, neįvertinant praktinių apribojimų. Han, Knott ir Egbelu siūlo pateikti L-formos modulių pakavimus su pradiniais modulių vertinimais, apimančiais visą konteinerio bazę ir vieną iš konteinerio sienų. Išdėstymas „L“ viduje yra determinuojamas dinaminio programavimo, kuris maksimizuoja kraštų panaudojimą. Vienas iš įdomesnių sienų užpildymo būdų yra pagal vieną iš šešių konteinerio plokštumų. Autorių aprašomame pavyzdyje talpinama viena dėže mažiau, negu gaunama kraunant dėžes dviejų skirtingų sienų išdėstymo būdais ant konteinerio grindinio. Han, Knott ir Egbelu pateikto metodo trūkumas yra L modulio perimetro utilizacijos maksimizavimas. Autoriai nepateikia įrodymų, kodėl turi būti pasirinktas būtent L-formos modelio metodas.

Han, Knott ir Egbelu sukurto algoritmo privalumai:

- maksimizuojamas pakuojamo konteinerio kraštų panaudojimas.

Trūkumai:

- neįvertinami pakuojamos dėžės praktiniai apribojimai;
- naudojant L modulį maksimaliai išnaudojamas perimetras;
- nepateikiami įrodymai, kodėl naudoti L-formos modulį.

Mohanty, Mathur ir Ivancic (1994) pasiūlė pritaikyti daugiamatį ,,kuprinės“ problemos metodą trijų dimensijų pakavimo problemoms spręsti, pripildant dėžėmis skirtingus konteinerius. Jų tikslas buvo konteineriuose maksimaliai išnaudoti erdvę arba konteinerių turinių dydį. Autoriai pateikė stulpelių generavimo procedūrą, kuri euristiškai naudoja ,,godų“ metodą. Šis metodas vienu metu generuoja stulpelius, neatsižvelgdamas į apribojimus, išskyrus persidengimą ir konteinerių dimensijas. Autorių siūlomas metodas nėra stabilus ir jame problematiška atrinkti skirtingus pakavimo elementus.

Mohanty, Mathur ir Ivancic sukurto algoritmo privalumai:

- siekiama maksimaliai išnaudoti konteinerių erdvę.

Trūkumai:

- naudojant ,,godų“ metodą neatsižvelgiama į apribojimus išskyrus persidengimą ir konteinerių dimensijas;
- metodas nėra stabilus;
- problematiška atrinkti skirtingus pakavimo elementus.

Chen, Lee ir Shen (1995) pristatė sveikų skaičių 0-1 linijinio programavimo modelį (angl. *zero-one mixed integer linear programming model*), skirtą konteinerio trijų dimensijų pakrovimo problemai spręsti. Uždavinio tikslas – turint daug įvairaus dydžio konteinerių, supakuoti duotas dėžes į juos taip, kad transportavimo kaštai būtų mažiausi. Modelis nagrinėja dėžės vietos nustatymo, sudėtinio dėžės dydžio, sudėtinio konteinerio dydžio, dėžių laisvos vietos perdengimo ir erdvės utilizavimo problemas. Buvo atkreipiamas dėmesys į tokias konteinerio pakrovimo specifines problemas, kaip vieno konteinerio išrinkimas iš keleto variantų, svorio balanso ir skirtingų konteinerių ilgių. Šioms situacijoms buvo numatyta bendro modulio modifikacija. Modulio pagrindimą aiškina remiantis siauro pavyzdžio problemomis (tik su 6 dėžėmis). Tolimesniuose tyrimuose gali būti pasiūlyta į modelius įtraukti papildomus apribojimus konteinerio pakrovimo problemai spręsti: pakavimo šablono stabilumas, kiekvienos dėžės tipo integralumas, svorio apribojimai. Naudojant autorių pateiktą analitinį modelį neįmanoma realiai išspręsti problemos, kol kintamųjų ir apribojimų skaičius išauga dvigubai kaip ir dėžių skaičius. (Jis išauga $2n^2$, kur n yra dėžių skaičius). Išvadose Chen, Lee ir Shen teigia, jog didesnės apimties konteinerio pakrovimo problemoms spręsti reikia efektyvesnio būdo.

Chen, Lee ir Shen sukurto algoritmo privalumai:

- dėžės pakuojamos parenkant vieną konteinerį iš keleto galimų variantų;
- atsižvelgiama į svorio balansą, skirtingus konteinerių ilgius;
- yra galimybė į modelį įtraukti papildomus apribojimus.

Trūkumai:

- modulio pagrindimas aiškinamas remiantis siauro pavyzdžio problemomis
- (tik su 6 dėžėmis);
- netinka didesnės apimties konteinerio pakrovimo problemoms spręsti.

Bischoff, Janetz ir Ratcliff (1995) vystydami trijų dimensijų euristinį požiūrį, išbandė kelias savo algoritmo modifikacijas. Viename algoritmo variante jie leido tą patį sluoksnį formuoti iš ne daugiau kaip dviejų tipų dėžių, kitame – iš ne daugiau kaip dviejų tipų dėžių su vienodais aukščiais, trečiame – iš vieno tipo dėžių. Jie išbandė visus savo algoritmo variantus pakuodami 1400 skirtingų problemų. Efektyviausias pasirodė pirmasis variantas, tačiau didėjant skirtingų dėžių kiekiui sprendinio kokybė mažėja.

Bischoff, Janetz ir Ratcliff sukurto algoritmo privalumai:

- algoritmas modifikuojamas, surandant efektyviausią spendimo variantą;
- pasižymi dideliu pakavimo stabilumu.

Trūkumai:

- didėjant skirtingų dėžių kiekiui sprendinio kokybė mažėja.

Martello, Pisinger ir Vigo (2000) vystė „šakos ir ribos“ algoritmą, padedantį išspręsti trijų-dimensijų dėžių pakavimo problemas. Uždavinio tikslas – turint daug vienodo dydžio konteinerių, supakuoti duotas dėžes į kiek galima mažesnį jų skaičių. Jų sprendimas nėra tiksliai trijų dimensijų. Pagal autorių sukurtą algoritmą, pirmiausia sukuriami dėžių sluoksniai, turintys plotį W, aukštį H ir skirtingus gylius. Tuomet sluoksniai sujungiami į trijų-dimensijų dėžes. Į pakavimo algoritmą nebuvo įtraukti skirtingi svarbūs dėžių pakavimui apribojimai ir nebuvo leistinas daiktų kaitaliojimas. Nepaisant tokių sąlygų, autorių pateiktas 3D objektų pakavimo būdas pasiekė gerų rezultatų.

Martello, Pisinger ir Vigo sukurto algoritmo privalumai:

- dėžės pakuojamos į kuo mažesnį vienodų konteinerių skaičių.

Trūkumai:

- pakuojant dėžes, neleistinas jų kaitaliojimas;
- neatsižvelgiama į svarbius skirtingus dėžių pakavimo apribojimus.

Faina (2000) nagrinėdamas 3D objektų pakavimo problemas nenaudoja konteinerio dalijimo į sluoksnius euristikos ir pateikia modelį, suvedantį trimačio pakavimo sprendinius į skaičiąją aibę. Modelis leidžia nagrinėti problemas, kai leistini visi galimi dėžių vartymai, o vienintelis taikomas apribojimas – dėžės turi tilpti į konteinerį. Pakavimo uždaviniui spręsti autorius naudoja atkaitinimo modeliavimo metodą. Pakuojant 32 dėžes, autorius sugebėjo parinkti aušinimo tvarkaraščius, duodančius geriausius rezultatus, tačiau, kai dėžių skaičius viršydavo 64, skaičiavimo laikas tapdavo nebepriimtinas.

Faina sukurto algoritmo privalumai:

- galimi visi dėžių vartymai, taikomas tik vienas apribojimas.

Trūkumai:

- didėjant dėžių skaičiui rezultatas blogėja;
- didėjant dėžių skaičiui skaičiavimo laikas ilgėja ir tampa nebepriimtinas.

2. METODOLOGINĖ DALIS

2.1. PAKAVIMO PROBLEMOS OPTIMALUMO KRITERIJAI

Kadangi 3D-SBSPP yra NP-pilnoji problema, tai jos išspręsti per polinominį laiką neįmanoma. Todėl reikia rasti būdą kaip įvertinti trimačio pakavimo algoritmų sprendinio kokybę. Toliau pristatysime autorių S.Martello, D. Prisinger ir D. Vigo [4] algoritmų sprendinio įverčius.

Akivaizdi apatinė riba skirta apibūdinti 3D-BPP optimalumui išplaukia iš konteinerių užpildymo dėžėmis:

$$L_0 = \left\lceil \frac{\sum_{i=1}^{n} v_j}{B} \right\rceil \quad (1)$$

L_0 apskaičiuojama per O(n) laiko.

Dabar ištyrinėsime blogiausią atvejį apatinės ribos L_0. Tegul $Z(I)$ yra optimalaus sprendinio reikšmė, $L(I)$ reikšmė gauta apskaičiavus apatinę ribą L_0 ir $\rho(I) = L(I)/Z(I)$. Absoliučiai blogiausio atvejo charakteristikos L_0 koeficientas yra apibrėžiamas kaip mažiausias realus skaičius ρ toks, kad $\rho(I) \geq \rho$ kiekvienam *I.* Asimptotiškai blogiausio atvejo charakteristikos L_0 koeficientas yra realus skaičius ρ^{∞} toks, kad egzistuoja $N \in Z^{+}$ kuriam $\rho(I) \geq \rho^{\infty}$ visiems *I* tenkinamiems $Z(I) > N$.

Teorema 1. Asimptotiškai blogiausio atvejo charakteristikos L_0 koeficientas lygus $\frac{1}{8}$.

Apatinė riba L_0 tinka geriausiai skaičiuoti kai dėžių matmenys lyginant su konteinerio matmenims yra maži. Todėl reikia apibrėžti geresnį įvertį.

$$J^{WH} = \left\{ j \in J : w_j > \frac{W}{2} \wedge h_j > \frac{H}{2} \right\} \tag{2}$$

$$L_1^{WH} = \left| \left\{ j \in J^{WH} : d_j > \frac{D}{2} \right\} \right| +$$

$$+ \max_{1 \le p \le \frac{D}{2}} \left\{ \left\lceil \frac{\sum_{j \in J_s(p)} d_j - \left(|J_l(p)| D - \sum_{j \in J_s(p)} d_j \right)}{D} \right\rceil, \left\lceil \frac{|J_s(p)| - \sum_{j \in J_l(p)} \left\lfloor \frac{D - d_j}{p} \right\rfloor}{\left\lfloor \frac{D}{p} \right\rfloor} \right\rceil \right\} \tag{3}$$

$$J_l(p) = \left\{ j \in J^{WH} : D - p \ge d_j \ge \frac{D}{2} \right\} \tag{4}$$

$$J_s(p) = \left\{ j \in J^{WH} : \frac{D}{2} \ge d_j \ge p \right\} \tag{5}$$

Analogiškai apatines ribos L_1^{WD}, L_1^{HD} gali būti gaunamos apibrėžiant:

$$J^{WD} = \left\{ j \in J : w_j > \frac{W}{2} \wedge d_j > \frac{D}{2} \right\}$$

$$J^{HD} = \left\{ j \in J : h_j > \frac{H}{2} \wedge d_j > \frac{D}{2} \right\}$$

ir lygiagrečiai pertvarkant (3)-(5).

Teorema 2. Efektyvi apatinė riba trimačio konteinerio pakavimo problemai yra

$$L_1 = max\{L_1^{WH}, L_1^{WD}, L_1^{HD}\}$$

Teorema 3. Duota sveikųjų skaičių pora (p,q) tokių, kad $1 \le p \le \frac{W}{2}$ ir $1 \le q \le \frac{H}{2}$ apibrėžkime

$$K_v(p,q) = \{ j \in J : w_j > W - p \wedge h_j > H - q \}$$

$$K_l(p,q) = \left\{ j \in J \backslash K_v(p,q) : w_j > \frac{W}{2} \wedge h_j > \frac{H}{2} \right\}$$

$$K_s(p,q) = \{ j \in J \backslash \left(K_v(p,q) \cup K_l(p,q) \right) : w_j > p \wedge h_j > q \}$$

$$L_2^{WH}(p,q) = L_1^{WH} + max\left\{ 0, \left\lceil \frac{\sum_{j \in K_l(p,q) \cup K_s(p,q)} v_j - \left(D \cdot L_1^{WH} - \sum_{j \in K_v(p,q)} d_j \right) WH}{B} \right\rceil \right\}$$

Tada efektyvi apatinė riba trimačio konteinerio pakavimo problemai yra

$$L_2 = max\{L_2^{WH}, L_2^{WD}, L_2^{HD}\}$$

kur

$$L_2^{WH} = \max_{1 \le p \le \frac{W}{2}; 1 \le q \le \frac{H}{2}} \{L_2^{WH}(p,q)\}$$

Tiriant algoritmų sprendinio kokybės efektyvumą ir taikysime L_2

2.2. BREANCH AND BOUND

2.2.1. VIENO KONTEINERIO UŽPILDYMAS

Skyrelyje 2.2.3. aprašysime algoritmą skirtą spręsti 3D-BPP. Šis algoritmas pakartotinai sprendžia problemas, kuriose visi objektai iš duotos aibės $\bar{J}$ turi būti supakuoti, jai yra įmanoma į vieną konteinerį. Taigi turim poaibį $\vec{J} \subseteq \bar{J}$ elementų, reikia kiekvienam $j \in \vec{J}$ priskirti tokias koordinates (x_j, y_j, z_j), taip kad pakuojami objektai nepersidengtų, nepatektų už konteinerio ribų ir bendras aibės $\bar{J}$ elementų tūris būtų maksimalus. Laikysime, kad koordinačių pradžios taškas yra kairysis apatinis galinis konteinerio kampas. Kiekviename medžio mazge, esamą dalinį sprendinį, kuris pakuoja objektus iš tam tikro poaibio $I \subset \bar{J}$, padidinam imdami nelemtus $j \in \bar{J} \backslash I$ iš eilės, ir generuojame naujus mazgus padedant j įvisus leistinus taškus. Tegul (x_p, y_p, z_p) taško p koordinatės: akivaizdu kad objektas j negali būti padėtas taške p jai $x_p + w_j > W$ ar $y_p + h_j > H$ ar $z_p + d_j > D$. Laikysimės tokių prielaidų:

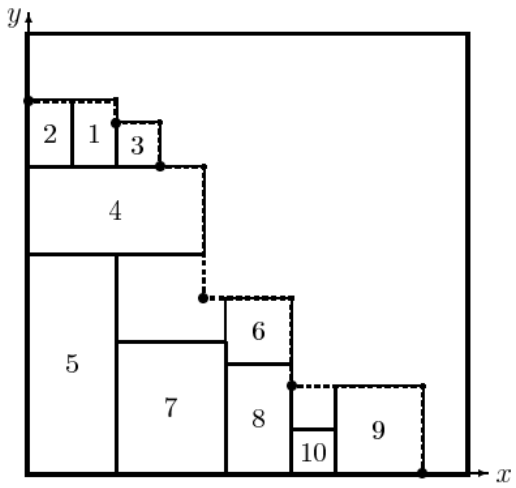
- Bet kuris konteinerio užpildymas dėžėm gali būti pakeistas ekvivalenčiu pakavimu, kai nė viena dėžė negali būti patraukta x,y ar z ašies kryptimi.
- Optimalaus sprendinio elementų tvarka tokia kad jai $i < j$, tai $x_i + w_i \le x_j$ arba $y_i + h_i \le y_j$ arba $z_i + d_i \le z_j$.

Iš antros sąlygos gauname svarbią išvadą, kad kai objektai iš I yra jau supakuoti ir imamas vienas $j \in \bar{J} \backslash I$ iš likusių objektų, jis gali būti padėtas į tokį tašką p, jog nei vieno elemento iš I neegzistuotų kokia dalis dešinėja taško p, ar viršuje – p, ar priekyje. Kitaip tariant jau patalpintų dėžių aibė I padalina konteinerį į du regionus kur nauji objektai gali būti pakuojami. Formaliai nauji pakuojami objektai gali būti padėti tik tokioje taškų aibėje:

$$S(I) = \{(x,y,z): \forall i \in I, x \ge x_i + w_i \; arba \; y \ge y_i + h_i \; arba \; z \ge z_i + d_i\}$$

2.3.2. GALIMOS DĖŽĖS PADĖJIMO VIETOS RADIMAS

Galimos pakavimo vietos radimo trijų matavimų geometrinėje erdvėje problemą spręsime pakartotinai taikant dviejų matavimų problemą atitinkamai keičiant koordinates. Taigi pirmiausiai ir aprašysime dviejų dimensijų algoritmą skirta spręsti galimo pakavimo vietos radimui.



2.1 pav. Galimos pakavimo pozicijos

Algoritmas nustatantis kampo taškus sudeda iš tokių žingsnių:

- Remiantis antra prielaida iš rikiuojam dėžes pagal jų *pabaigos taškus* $(x_j + w_j, y_j + h_j)$, t.y. didėjimo pagal $x_j + w_j$, o po to reikšmės $y_j + h_j$ ne didėjimo tvarka.
- Randam ekstremalius pakuojamus objektus, t.y. tokius kurių kampai sutampa su kreivės(riboja jau su pakuotus objektus nuo laisvos vietos) taškais kur ji keičia savo kryptį iš vertikalios pozicijos į horizontalę.
- Randame kampo taškus
- Pašaliname netinkamus kampo taškus

Algoritmo psiaudo kodas pateiktas žemiau.

algorithm 2D-CORNERS:
begin
if $\hat{I} = \emptyset$ **then** $\hat{C}(\hat{I}) \coloneqq \{(0,0)\}$ **return**;
$\bar{x} \coloneqq 0$;
$m \coloneqq 0$;
for $j \coloneqq 1$ **to** $|\hat{I}|$ **do**
if $(x_j + w_j > \bar{x})$ **then**
begin
$m \coloneqq m + 1$;
$e_m \coloneqq j$;
$\bar{x} \coloneqq x_j + w_j$;
end;
end for;
$\hat{C}(\hat{I}) \coloneqq \{(0, y_{e_1} + h_{e_1})\}$;
for $j \coloneqq 2$ **to** m **do** $\hat{C}(\hat{I}) \coloneqq \hat{C}(\hat{I}) \cup \{(x_{e_{j-1}} + w_{e_{j-1}}, y_{e_1} + h_{e_1})\}$;

$\hat{C}(\hat{I}) \coloneqq \hat{C}(\hat{I}) \cup \{(x_{e_m} + w_{e_m}, 0)\}$
for each $(x_j^{'}, y_j^{'}) \in \hat{C}(\hat{I})$ **do**
if $x_j^{'} + \min_{i \in \bar{J} \setminus I}\{w_i\} > W$ **or** $y_j^{'} + \min_{i \in \bar{J} \setminus I}\{h_i\} > H$ **then** $\hat{C}(\hat{I}) \coloneqq \hat{C}(\hat{I}) \setminus \{(x_j^{'}, y_j^{'})\}$
end.

2.2 pav. Procedūra 2D-CORNERS

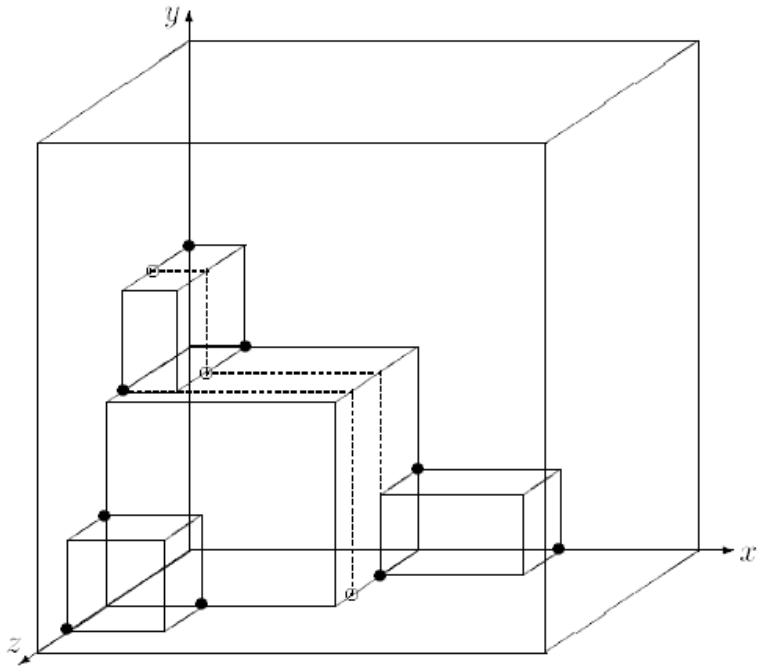
Algoritmo sudėtingumas laiko prasme yra $O(|I|)$ ir plius pradinio objektų rikiavimo $O(|I|log|I|)$. Tarkime, kad gauti kampo taškai yra $\hat{C}(I) = \{(x_1^{'}, y_1^{'}), \ldots, (x_l^{'}, y_l^{'})\} \neq \emptyset$, tada jau su pakuotų objektų apgaubtas plotas

$$A(I) = x_1^{'}H + \sum_{i=2}^{l}(x_i^{'} - x_{i-1}^{'})\, y_{i-1}^{'} + (W - x_l^{'})y_l^{'}$$

Toliau algoritmą 2D-CORNERS naudosime nustatant trijų dimensijų kampo taškų aibę $C(I)$. Pirmiausiai minėtas algoritmas skaičiuojamas kai z = 0 ir tada taikome visom kitom skirtingom *z* koordinatėms, didindami *z* reikšmę pridedant kiekvieno objekto iš aibės *I* pabaigos koordinates. Taigi algoritmas 2D-CORNERS taikomas tokiom $z^{'}$ koordinatėm kurios tenkina sąlygą

$$z_i + d_i = z^{'} \quad \forall i \in I$$

Tokiu būdu galime gauti ir klaidingų kampo taškų. Ši situacija pavaizduota žemiau 4paveiksle.



2.3 pav. 3-D konteinerio užpildymas

Klaidingos kampo taškų koordinatės pašalinamos nustatant dominuojančius taškus. Sakysime, kad taškas $(x_a^{'}, y_a^{'}, z_a^{'})$ dominuoja prieš kitą tašką $(x_b^{'}, y_b^{'}, z_b^{'})$ jai

$$x_a^{'} = x_b^{'}\ ir\ y_a^{'} = y_b^{'}\ ir\ z_a^{'} < z_b^{'}.$$

Taigi nustatyti kampo taškų aibe trimačiu atveju taikome tokį algoritmą:

algorithm 3D-CORNERS:
begin
    if $I = \emptyset$ **then** $C(I) \coloneqq \{(0,0,0)\}$ and **return**;
    $T \coloneqq \{0\} \cup \{z_i + d_i : i \in I\}$;
    rikiuoti T didėjimo tvarka;
    $C(I) \coloneqq \emptyset$;
    $\hat{C}(\hat{I}_0) \coloneqq \emptyset$;
    $k \coloneqq 1$;
    while $k \leq r$ and $z_k^{'} + \max_{i \in \bar{J} \setminus I}\{d_i\} \leq D$ **do**
    $\hat{I}_k \coloneqq \{i \in I : z_i + d_i > z_k^{'}\}$;
    $\hat{C}(\hat{I}_k)$ taikyti 2D-CORNERS;
    for each $(x_i^{'}, y_j^{'}) \notin \hat{C}(\hat{I}_k)$ **do**
        if $(x_i^{'}, y_j^{'}) \notin \hat{C}(\hat{I}_{k-1})$ **then** $C(I) \coloneqq C(I) \cup \{(x_j^{'}, y_j^{'}, z_k^{'})\}$;
    end
end.

2.4 pav. Procedūra 3D-CORNERS

Šios procedūros sudėtingumas laiko prasme yra $O(n^2)$. Laikydami, kad k^* yra indeksas paskutinės aibės I_k generuotos algoritmo 3D-CORNERS, tai jau supakuotų dėžių padengtas tūris yra

$$V(I) = \sum_{k=2}^{k^*} (z_k^{'} - z_{k-1}^{'})\, A(I_{k-1}) + (D - z_{k^*}^{'}) A(I_{k^*})$$

2.3.3. ONEBIN: TIKSLUS ALGORITMAS SKIRTAS UŽPILDYTI VIENĄ KONTEINERĮ

Dabar galime suformuluoti rekursinį algoritmą pavadintą ONEBIN, skirtą rasti geriausią vieno konteinerio užpildymą dėžėmis iš duotos aibės $\bar{J}$. Iš pradžių konteineris yra tuščias ir $C(\emptyset) = \{(0,0,0)\}$. Kiekvienos iteracijos metu, duotai pakuojamų dėžių aibei $I \subset \bar{J}$, priskiriama kapo taškų aibė $C(\mathrm{I})$ apskaičiuojant procedūra 3D-CORNERS, kartu su atitinkamu tūriu $V(\mathrm{I})$. Jai F yra bendras tūris gautas esamo geriausio užpildymo, tada algoritmą galime nutraukti bet kada kai

$$\sum_{i \in I} v_i + \big(B - V(I)\big) \leq F$$

nes jai ir užpildytume likusią nepanaudotą vietą tai nepagerinsime sprendinio vertės. Tai pat algoritmas nutraukiamas kai daugiau objektų negalų būti talpinama į konteinerį. Kitu atveju kiekvienai pozicijai $(x^{'}, y^{'}, z^{'}) \in C(I)$ ir kiekvienai dėžei $j \in \bar{J} \setminus I$ tikriname ar ji gali būti patalpinama, kad nepažeistų konteinerio ribų, jai taip tai priskiriame dėžių tam taškui ir procedūrą kartojame rekursyviai. Geriausi algoritmo rezultatai gaunami kai objektai iš pradžių išrikiuojami tūrio mažėjimo prasme. Kadangi šis algoritmas paremtas 3D-CORNERS algoritmu, kuris reikalauja visų įmanomų pakavimo vietų radimo, praktiniai eksperimentai parodė, jog jis reikalauja daug laiko. Bet kadangi

algoritmas vieno konteinerio pildymui dažniausiai sprendžiamas su mažu dėžių poaibiu tai jo naudojimas pasiteisina.

PAGRINDINS SAKOS MEDIS

Pagrindinis šakos medis priskiria objektus skirtingiems konteineriams nespecifikuodamas jų tikslios pozicijos. Objektai priešai yra išrikiuojami tūrio mažėjimo prasme ir tada pritaikoma depth-firtst strategija. Tegul Z yra dabartinio sprendinio reikšmė ir $M = \{1, \dots, m\}$ ir dabartinė konteinerių aibė naudojama patalpinti dėžes dominuojančiame sprendimo mazge. Konteineris iš aibės M vadinamas atviru jai tenkina laisvos vietos radimo sąlygas atvirkščiai pilnas. Kiekviename sprendimo mazge sekančios laisvos dėžės yra priskiriamos iš eilės kiekvienam atidarytam konteineriu, jai $|M| < Z - 1$ tada objektai priskiriami naujam konteineriu. Kai naujas objektas k yra priskiriamas konteineriu (dabartiniam konteineriu) kuris jau talpina poaibį $J_i \neq \emptyset$, tada priskyrimo galimumą patikrinamas sekančiai. Pirmiausiai, apatinė riba L_2 yra pasikačiuojama tokiai aibei $\bar{J} = J_i \cup \{k\}$: jai $L_2 \geq 2$ mazgas iškarto panaikinamas. Kitu atveju taikoma H1 euristika, jai sprendinys gaunamas iš vieno konteinerio, tai naujo objekto k priskyrimas priimtinas. Jai negaunamas sprendinys iš vieno konteinerio tada taikoma 5.2 euristika. O jai ir dabar neįmanoma supakuoti visų objektų iš $\bar{J}$ į viena konteinerį tada medžio mazgas yra ištrinamas.

Kai priskiriama nauja dėžė konteineriu, jis bandomas uždaryti. Kiekvienam laisvam objektui k' tikrinama apatinės ribos L_2 reikšmė sukonstruojant tokią aibę $\bar{J} \cup \{k'\}$. Jai gauta apatinė riba su kiekvienu k' objektu konteineris uždaromas ir mes žinome, kad daugiau jokios kitos dėžės neįmanoma patalpinti į jį. Kitu atveju tikrinama sekanti dominavimo taisyklė. Tegul K būna poaibis iš trijų elementų kuriems pirštai buvęs ribos L_2skaičiavimas gražino vieneto reikšmę. Suformuluojam tokia problemą $\bar{J} \cup K$ tada taikome euristikas H1, H2 ir jai bent viena iš jų gauna sprendinį iš vieno konteinerio tada laikome kad nėra geresnio patalpinimo būdo šioms trims dėžėms. Patalpiname jas į esamą konteinerį ir jį uždarome. Bet kada kai konteineris yra uždaromas, apatinė riba L_2 yra apskaičiuojama visiems likusiems nesupakuotiems objektams ir jai $Z \leq L_2 + c$ algoritmas kviečiamas rekursyviai.

Kai medžio mazgas yra neatmetamas nei apatinės ribos arba nepriimtas euristikos tada dėžių aibės priskyrimo galimumas testuojamas algoritmo ONEBIN. Kadangi mes tikimės rasti sprendinį tik tokį kur visi elementai iš $\bar{J}$ yra supakuojami viename konteineryje, mes apibrėžiame geriausią patalpinimą tokiu būdu $F = \sum_{l \in \bar{J}} v_l - 1$. Jai algoritmas ONEBIN gražina nepakeistą F reikšmę tada neegzistuoja pakavimo kuris patalpina visus elementus iš $\bar{J}$ į vieną konteinerį.

APROKSIMAVIMO ALGORITMAI

Tam, kad gauti gerą viršutinę ribą medžio pradiniame mazge ir apriboti algoritmo ONEBIN kvietimo skaičių dvi skirtingo euristikos yra taikomos. Du priartėjimai skaičiuojant pateikia vienas kitą papildantį bendro algoritmo elgesį: priklausomai nuo duomenų vieno iš jų turi prastas sprendimo charakteristikas tuo tarpu kitas turi geras skaičiavimo charakteristikas.

algoritmas H1:
pradžia
    $T \coloneqq \bar{J}$; išrikiuojame visus *T* gylio mažėjimo tvarka;
    kol $T \neq \emptyset$ **daryti**
        pradėti
            tegul $T = \{j_1, \dots, j_{|T|}\}$; $k := max\{r : \sum_{s=1}^{r} w_{j_s} h_{j_s} < 2WH\}$;
            komentaras: sukonstruojamas vienas gylio d_{j1} sluoksnis;
            $T' = \{j_1, \dots, j_k\}$; išrikiuojam T' elementus aukščio mažėjimo tvarka;
            kiekvienam objektui $j \in T'$ **daryti**
                jai *j* telpa į egzistuojantį sluoksnį **tada** *j* pakuojam
                kitaip jai nepakanka vertikalios vietos *j* **tada** pakuojam *j* naujame sluoksnį
            pašalinam iš *T* jau supakuotus elementus;
        baigti;
    kombinuojame gautus sluoksnius į *D* gylio konteinerius remiantis 1D-BPP algoritmu;
baigti

2.5 pav. Algoritmas H1

algoritmas H2:
pradėti
    $T := \bar{J}$; išrikiuojam dėžes tūrio mažėjimo prasme;
    kol $T \neq \emptyset$ **daryti**
        pritaikome ONEBIN aibei *T* ir pašalinam supakuotas dėžes iš *T;*
baigti

2.6 pav. Algoritmas H2

2.2. TSPACK

Pagrindinis skirtumas tarp vienos dimensijos ir kelių dimensijų pakavimo problemos yra pakavimo galimumo patikrinimas, pavyzdžiu duota objektų I_c aibė priskirta konteineriu C, nustatyti ar egzistuoja toks patalpinimo būdas konteineryje, kad objektai nepersidengtų ir pakavimas derėtų su konteinerio dydžiu. Dažnai ir tiksliose ir euristinėse procedūrose, pakavimo galimumas ir tikslo funkcijos įvertinimas yra tarpusavyje glaudžiai susiję. Atskiriant pakavimo galimumo ir optimalumo problemas naudojant dvi euristikas, leidžia kitaip suformuluoti problemos sprendimą ir taikyti skirtingus metodus spręsti dvi atskiras problemas. Be to metodai skirti patalpinimo galimumo

problemai spręsti gali būti panaudojami skirtingom pakavimo problemų klasėms ir nepaveiks uždavinio optimalumo kriterijaus. Tokiu būdu naudojamas dviejų lygių TSPACK algoritmas, kur pirmo lygio euristika sprendžia BP uždavinio optimalumo problemą, o antro lygio euristika – patalpinimo galimumo problemą. Pagrindinė algoritmo struktūra pateikta 2.7 paveiksle.

algorithm TSpack:
    $z^* \coloneqq A(\{1, \dots, n\})$ - pradinio sprendino reikšmė;
    L – optimalaus sprendinio apatinė riba;
    if $z^* = L$ **then stop**;
    sukuriami tušti tabu sąrašai;
    supakuojama kiekviena dėžė į atskirus konteinerius;
    $z \coloneqq n$ - Tabu Paieškos sprendinio reikšmė;
    $d \coloneqq 1$;
    apibrėžiame pildomą konteinerį *t*;
    while laiko ar iteracijų limitas nepasiektas **do;**
        diversify ≔ false; $k \coloneqq 1$;
        while *diversify*=false **and** $z^* > L$ **do**
            $k_{in} \coloneqq k$;
            call SEARCH(*t*,*k*,*diversify*,*z*);
            $z^* \coloneqq min\{z^*, z\}$;
            if $k \leq k_{in}$ **then** apibrėžiame naują pildomą konteinerį *t;*
        end while;
        if $z^* = L$ **then stop**
        else call DIVERSIFICATION(*d*,*z*,*t*)
    end while;
end.

2.7 pav. Algoritmas TSpack

Trimačio dėžių pakavimo uždavinio problema specifikuojama pagal tai koks pasirenkamas vidinis euristinis algoritmas *A.* Algoritmas *A* naudojamas apibrėžti dėžių perskirstymui tarp gretimų konteineriu (neighborhood search). Tegul *A*(*S*) yra sprendimo reikšmė gauta algoritmo *A* apdorojant pakuojamų objektų aibę *S.* Yra bandoma išimti dėžę *j* iš pasirinkto konteinerio *t* perkeliant į gretimą, taip neatsirastų papildomų konteinerių. Konteinerį iš kurio mėginama pašalinti dėžė pasirenkam panaudojant užpildymo funkciją. Tegul S_i žymi aibę dėžių kurios yra konteinerį i ir $\propto$ vartotojo apibrėžta teigiama konstanta. Taigi konteineris t pasirenkamas iš visų tų kurių funkcijos $\varphi(S_i)$ reikšmė maziausia. Užpildymo funcija yra apibrėžiama taip:

$$\varphi(S_i) = \alpha \frac{\sum_{j \in S_i} v_j}{V} - \frac{|S_i|}{n}$$

Algoritmo TSpack procedūros SEARCH(*t*,*k*,*diversify*,*z*) psiaudo kodas pateikiamas antrame paveiksle.

procedure SEARCH(*t,k,diversify,z*):
penalty* $\coloneqq +\infty$;
for each $j \in S_t$ **do**
for each *k*-tuple *K* konteinerių neturinčių *t* **do**
$S \coloneqq \{j\} \cup (\bigcup_{i \in K} S_i)$;
$penalty \coloneqq +\infty$;
case
$A(S) < k$:
vykdyti perkėlimą ir atnaujinti sprendinio reikšmę z;
$k \coloneqq max\{1, k-1\}$;
return;
$A(S) = k$:
if perkėlimas nėra tabu **or** $S_t \equiv \{j\}$ **then**
vykdyti perkėlimą ir atnaujinti sprendinio reikšmę z;
if $S_t \equiv \{j\}$ **then** $k \coloneqq max\{1, k-1\}$;
return;
end if;
$A(S) = k+1$ **and** $k > 1$:
tegul *I* yra aibė iš $k+1$ konteinerio naudoto A;
$\bar{t} \coloneqq \arg\min_{i \in I}\{\varphi(S_i)\}$, $T \coloneqq (S_t \backslash \{j\}) \cup S_t$;
if $A(T) = 1$ **and** perkėlimas nėra tabu **then**
$penalty \coloneqq \min\{\varphi(T), \min_{i \in I \backslash \{\bar{t}\}}\{\varphi(S_i)\}\}$
case end;
$penalty^* \coloneqq \min\{penalty^*, penalty\}$;
end for;
end for;
if $penalty^* \neq \infty$ **then** vykdyti perkėlimą į atitinkamą $penalty^*$
else if $k \neq k_{max}$ **then** $diversify \coloneqq true$ **else** $k \coloneqq k+1$
return.

2.8 pav. Procedūra SEARCH

procedure DIVERSIFICATION(*d,z,t*):
if $d \ll z$ **and** $d < d_{max}$ **then**
$d \coloneqq d+1$;
tegul *t* yra konteineris kurio mažiausia funkcijos reikšmė $\varphi(\cdot)$;
else
pasalinti iš sprendimo $\lfloor z/2 \rfloor$ konteinerį su maziausia funcijos reikšmė $\varphi(\cdot)$;
supakuoti dėžes iš pasalinto konteinerio į atskirus konteinerius;
išvalyti visus tabu sąrašus;
$d \coloneqq 1$;
return.

2.9 pav. Procedūra DIVERSIFICATION

3. TIRIAMOJI DALIS IR REZULTATAI

3.1. EKSPERMENTINIŲ DUOMENŲ KLASĖS

Norint visapusiškai ištirti pakavimo algoritmus labai svarbu pradiniai eksperimento duomenys, nes nuo jų priklauso algoritmo sprendinio kokybė. Konteinerio pakavimo problema tampa sudėtingesne kai pakuojamų objektų matmenys tarpusavį ryškiai išsiskiria. Atvejis kada dėžės yra labai artimu arba vienodu matmenų nenagrinėsime nes jis visai neįdomus kombinatorinio optimizavimo prasme. Taigi Tyrimui naudojama skirtingos pradinių duomenų klasės. Pirmos keturios klasės buvo pasiūlytos Martello ir Vigo, ir yra pagrįstos atsitiktiniu keturių tipų objektų generavimu:

tipas 1: h_j yra atsitiktinis dydis ir įgyja reikšmes iš intervalo $\left[1, \frac{1}{2}H\right]$

w_j yra atsitiktinis dydis ir įgyja reikšmes iš intervalo $\left[\frac{2}{3}W, W\right]$

d_j yra atsitiktinis dydis ir įgyja reikšmes iš intervalo $\left[\frac{2}{3}D, D\right]$

tipas 2: h_j yra atsitiktinis dydis ir įgyja reikšmes iš intervalo $\left[\frac{2}{3}H, H\right]$

w_j yra atsitiktinis dydis ir įgyja reikšmes iš intervalo $\left[1, \frac{1}{2}W\right]$

d_j yra atsitiktinis dydis ir įgyja reikšmes iš intervalo $\left[\frac{2}{3}D, D\right]$

tipas 3: h_j yra atsitiktinis dydis ir įgyja reikšmes iš intervalo $\left[\frac{2}{3}H, H\right]$

w_j yra atsitiktinis dydis ir įgyja reikšmes iš intervalo $\left[\frac{2}{3}W, W\right]$

d_j yra atsitiktinis dydis ir įgyja reikšmes iš intervalo $\left[1, \frac{1}{2}D\right]$

tipas 4: w_j yra atsitiktinis dydis ir įgyja reikšmes iš intervalo $\left[1, \frac{1}{2}W\right]$

h_j yra atsitiktinis dydis ir įgyja reikšmes iš intervalo $\left[1, \frac{1}{2}H\right]$

d_j yra atsitiktinis dydis ir įgyja reikšmes iš intervalo $\left[\frac{1}{2}D, D\right]$

tipas 5: w_j yra atsitiktinis dydis ir įgyja reikšmes iš intervalo $\left[1, \frac{1}{2}W\right]$

h_j yra atsitiktinis dydis ir įgyja reikšmes iš intervalo $\left[1, \frac{1}{2}H\right]$

d_j yra atsitiktinis dydis ir įgyja reikšmes iš intervalo $\left[1, \frac{1}{2}D\right]$

Klasė k $(k \in \{1,2,3,4,5\})$ gaunama generuojant objektus k-tipo su tikimybe 0,6 o likusių trijų klasių su tikimybėmis 0,1. Konteinerio matmenys bus $W = H = D = 100$.

Kitos trys klasės pasiūlytos Berkey ir Wang :

klasė 6 : $W = H = D$; w_j, h_j ir d_j atsitiktiniai dydžiai ir įgyja reikšmes iš intervalo [1; 10];

klasė 8 : $W = H = D$; w_j, h_j ir d_j atsitiktiniai dydžiai ir įgyja reikšmes iš intervalo [1; 35];

klasė 9: $W = H = D$; w_j, h_j ir d_j atsitiktiniai dydžiai ir įgyja reikšmes iš intervalo [1; 100];

3.2. ALGORITMŲ ANALIZĖ

ŠAKOS IR RIBOS ALGORITMO ANALIZĖ

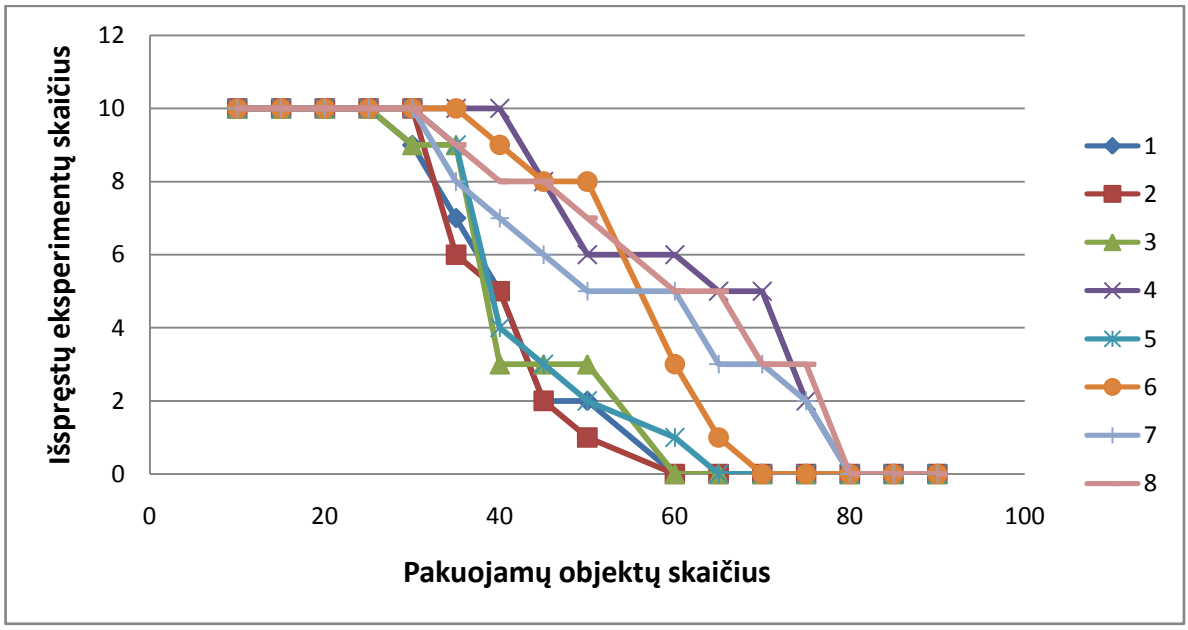
Algoritmas tiriamas generuojant kiekvienos objektų klasės duomenis. Kiekvienai klasei atliekama 10 eksperimentų. Pakuojamų objektų skaičius išlėto didinamas nuo 10 iki 90. Kadangi negalime problemos spręsti begalo daug lako, tai nusakom laiko limitą per kurį ieškosime sprendinio. Algoritmo testavimui buvo pasirinkta 100s. Tokiu būdu tikimasi ištirti kaip vidutiniškai priklauso šakos ir ribos algoritmo sprendinys, nuo pradinių duomenų aibės.

3.1. lentelė

Išspręstų eksperimentų remiantis optimalumo kriterijais skaičius

Objektų skaičius	Eksperimentinių duomenų klasė								Bendras išspręstų eksperimentų skaičius
	1	2	3	4	5	6	7	8	
10	10	10	10	10	10	10	10	10	80
15	10	10	10	10	10	10	10	10	80
20	10	10	10	10	10	10	10	10	80
25	10	10	10	10	10	10	10	10	80

30	9	10	9	10	10	10	10	10	78
35	7	6	9	10	9	10	8	9	68
40	5	5	3	10	4	9	7	8	51
45	2	2	3	8	3	8	6	8	40
50	2	1	3	6	2	8	5	7	34
60	0	0	0	6	1	3	5	5	20
65	0	0	0	5	0	1	3	5	14
70	0	0	0	5	0	0	3	3	11
75	0	0	0	2	0	0	2	3	7
80	0	0	0	0	0	0	0	0	0
85	0	0	0	0	0	0	0	0	0
90	0	0	0	0	0	0	0	0	0



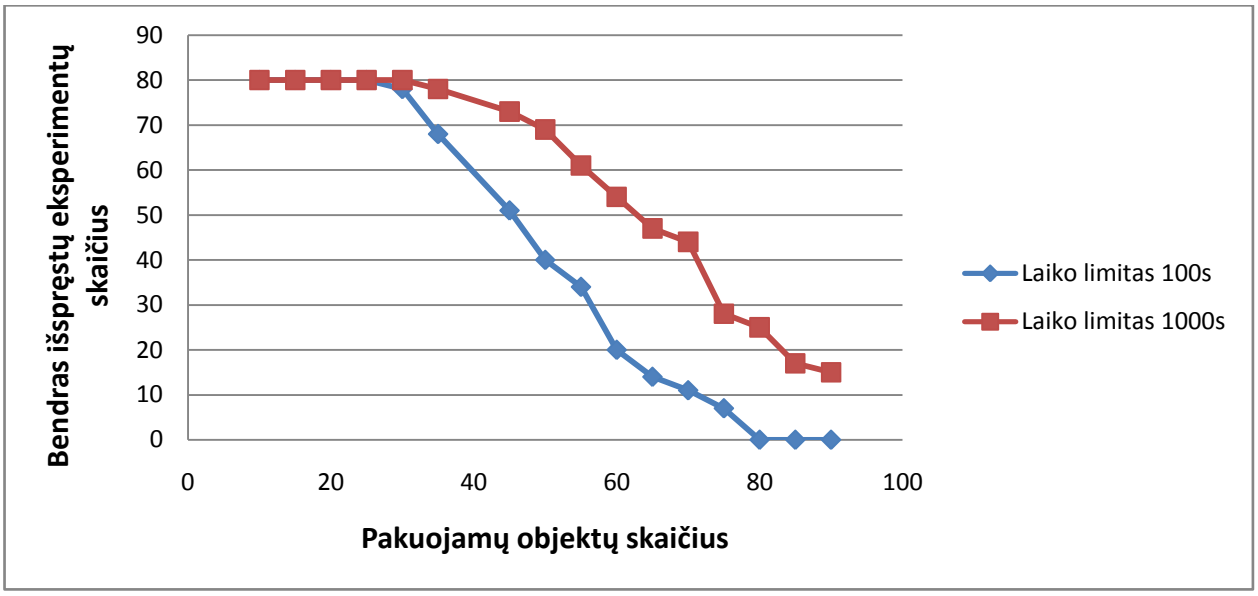
3.1 pav. Išspręstų eksperimentų remiantis optimalumo kriterijais skaičius

Kaip matome iš pateiktų rezultatų, kad optimaleliai per 100 sekundžių visada išsprendžiamas trimatis pakavimo uždavinys su 30 objektų nesvarbu kuriai klasei jie priklauso. Akivaizdžiai pastebime, kad didėjant objektų skaičių per priimtiną laiką pagal nustatytus optimalumo kriterijus išspręsti tampa neįmanoma. Suprantama, kad pasirinkus didesnį maksimalų leistiną šakų ir ribų algoritmo sprendimo laiką gauti geresnius rezultatus. Toliau apžvelgsime eksperimentų skaičiavimą kai laiko limitas 1000s.

3.2. lentelė

Išspręstų eksperimentų remiantis optimalumo kriterijais skaičius

Objektų skaičius	Eksperimentinių duomenų klasė								Bendras išspręstų eksperimentų skaičius
	1	2	3	4	5	6	7	8	
10	10	10	10	10	10	10	10	10	80
15	10	10	10	10	10	10	10	10	80
20	10	10	10	10	10	10	10	10	80
25	10	10	10	10	10	10	10	10	80
30	10	10	10	10	10	10	10	10	80
35	10	10	10	10	10	10	8	10	78
45	9	8	10	10	9	10	7	10	73
50	9	8	7	10	9	10	7	9	69
55	7	8	7	10	8	10	4	7	61
60	6	7	4	10	8	9	3	7	54
65	5	6	4	9	6	9	3	5	47
70	5	6	3	9	6	6	3	6	44
75	4	2	3	7	4	6	2	0	28
80	4	2	1	7	4	6	1	0	25
85	2	0	0	6	3	5	1	0	17
90	2	0	0	6	3	4	0	0	15



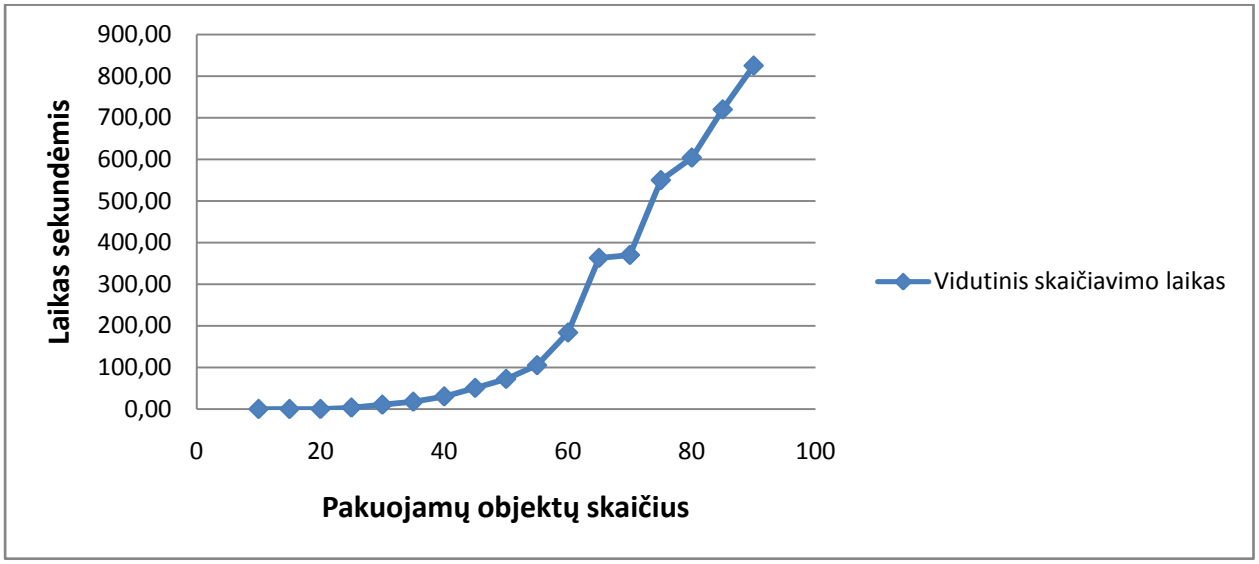
3.2 pav. Šakos ir ribos algoritmo išspręstų eksperimentų palyginimas

Kaip matome iš pateiktų rezultatų, kad optimaleliai per priimtiną laiką visada išsprendžiamas trimatis pakavimo uždavinys su 35 objektais nesvarbu kuriai klasei jie priklauso. Algoritmo išspręstų eksperimentų palyginimas laiko atžvilgiu pateiktas 3.2 paveiksle. Toliau apžvelgsime eksperimentų skaičiavimo trukmes.

3.3. lentelė

Išspręstų eksperimentų remiantis optimalumo kriterijais skaičiavimo vidutiniai laikai

Objektų skaičius	Eksperimentinių duomenų klasė								Vidutinis skaičiavimo laikas
	1	2	3	4	5	6	7	8	
10	0	0	0	0	0	0	0	0	0,00
15	0	0	0	0	0,01	0	0	0	0,00
20	0,01	0,01	0,02	0,01	0,15	0,02	0,04	0,03	0,04
25	0,07	0,03	0,08	0,01	4,32	0,15	22,65	0,16	3,43
30	4,62	0,25	0,54	0,02	11,5	0,54	53,53	12,5	10,44
35	11,53	12,185	5,13	0,51	40,52	1,25	49,34	20,15	17,58
40	17,13	20,52	47,25	0,55	50,55	3,5	72,36	30,261	30,27
45	54,62	30,48	53,28	1,55	100,53	12,165	83,11	70,26	50,75
50	64,53	55,65	90,26	5,25	199,29	20,33	83,15	60,23	72,34
55	79,12	43,15	156,55	10,25	216,54	40,55	215,45	80,26	105,23
60	90,36	50,26	60,52	20,15	426,35	156,21	452,02	214,25	183,77
65	171,38	223,26	457,56	50,61	616,35	320,23	512,25	553,5	363,14
70	226,12	252,45	475,25	80,25	403,28	144,25	765,36	614,52	370,19
75	421,28	538,35	602,25	152,55	619,25	316	861,265	889,63	550,07
80	456,56	654,36	617,25	165,25	643,25	502,15	852,51	941,4	604,09
85	546,9	799,36	901,36	264,69	823,29	464,58	959,92	998,31	719,80
90	645,48	831,65	964,34	459,58	1000,2	700,26	1000,02	1000,13	825,21



3.3 pav. Algoritmo skaičiavimo laiko priklausomybė nuo pradinių duomenų kiekio

Toliau pateiksime algoritmo apskaičiuotas ir optimalumo kriterijaus L2 reikšmes.

3.4. lentelė

Vidutinė algoritmo sprendinio reikšmė

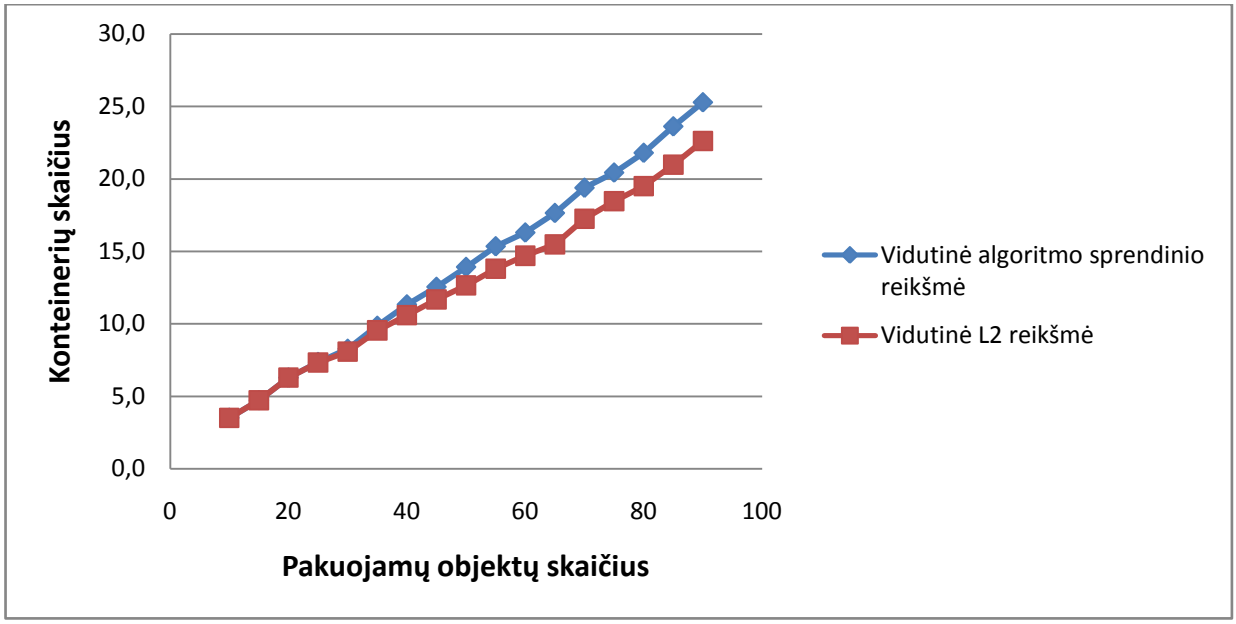
Objektų skaičius	Eksperimentinių duomenų klasė								Vidutinė algoritmo sprendinio reikšmė
	1	2	3	4	5	6	7	8	
10	3,3	3,5	3,6	6,6	2,5	2,9	2,9	2,8	3,5
15	4,7	4,5	4,8	8,7	2,9	4,4	4,1	3,7	4,7
20	6	6,6	6	12,3	3,9	5,4	5,2	4,9	6,3
25	7,4	7	7,1	15,4	4,6	5,9	5,8	5,7	7,4
30	8,6	8	8,6	17,2	5,6	6,7	5,4	6,1	8,3
35	9,4	9,6	10,3	21,1	6,4	7,9	7	7,2	9,9
40	11,4	10,9	11,6	24,3	7,5	8,9	8,4	7,7	11,3
45	12,5	12,6	12,2	27,6	8	9,9	9,1	8,5	12,6
50	13,7	14,1	13,6	29,6	9,5	10,1	10,8	10	13,9
55	15	14,9	14,9	32,7	9,7	12	11,9	11,7	15,4
60	15,7	15,5	15,4	36,4	10,1	13,2	12,1	12	16,3
65	17,2	17	17,7	37,6	12,1	13,8	13,7	12,1	17,7
70	19,2	18,7	19,7	41,7	12,7	15,4	14,2	13,5	19,4
75	20	19,6	20,1	44,9	12,3	16,4	14,7	15,5	20,4
80	21,3	21,2	21,4	48,1	14	17,1	15,8	15,5	21,8
85	23,3	23,2	23,5	51,9	15,6	17,5	18,1	15,9	23,6
90	25,3	25,4	24,9	54,9	16,2	19,6	18,8	17,2	25,3

3.5. lentelė

Vidutinė L2 reikšmė

Objektų skaičius	Eksperimentinių duomenų klasė								Vidutinė L2 reikšmė
	1	2	3	4	5	6	7	8	
10	3,3	3,5	3,6	6,6	2,5	2,9	2,9	2,8	3,5
15	4,7	4,5	4,8	8,7	2,9	4,4	4,1	3,7	4,7
20	6	6,6	6	12,3	3,9	5,4	5,2	4,9	6,3
25	7,4	7	7,1	15,4	4,6	5,9	5,8	5,4	7,3
30	8,2	7,8	8,4	17,2	5,4	6,5	5,4	5,8	8,1
35	9,2	9,1	9,7	21,1	6,2	7,5	6,9	6,7	9,6
40	10,5	9,9	10,5	24,3	6,1	8,6	7,4	7,5	10,6
45	11,3	11,5	11,3	27,6	6,6	9,2	7,9	8	11,7
50	12,5	12,7	12,3	28,9	7,5	9,4	9,4	8,5	12,7
55	13,2	13,1	13,1	32	6,8	11,4	10,8	10	13,8
60	14,1	14,1	13,9	36,1	7	11,6	10,3	10,5	14,7
65	15,3	15,2	16	36,6	9	11,9	10,5	9,4	15,5
70	17,1	16,7	17,5	40,1	9,6	13,4	12,3	11,3	17,3
75	18	17,7	18,2	43,3	9,2	15,3	13	13	18,5
80	19,4	19,2	19,7	46,9	10,4	15,3	12,8	12,3	19,5
85	21,3	21,3	21,2	50,6	12	15,5	14,4	11,5	21,0

90	22,9	23,2	22,5	53,3	12,7	17,9	15,5	12,9	22,6



3.4 pav. „Šakos ir ribos algoritmo“ ir L2 apskaičiuotų reikšmių palyginimas

TABU PAIEŠKOS ALGORITMO ANALIZĖ

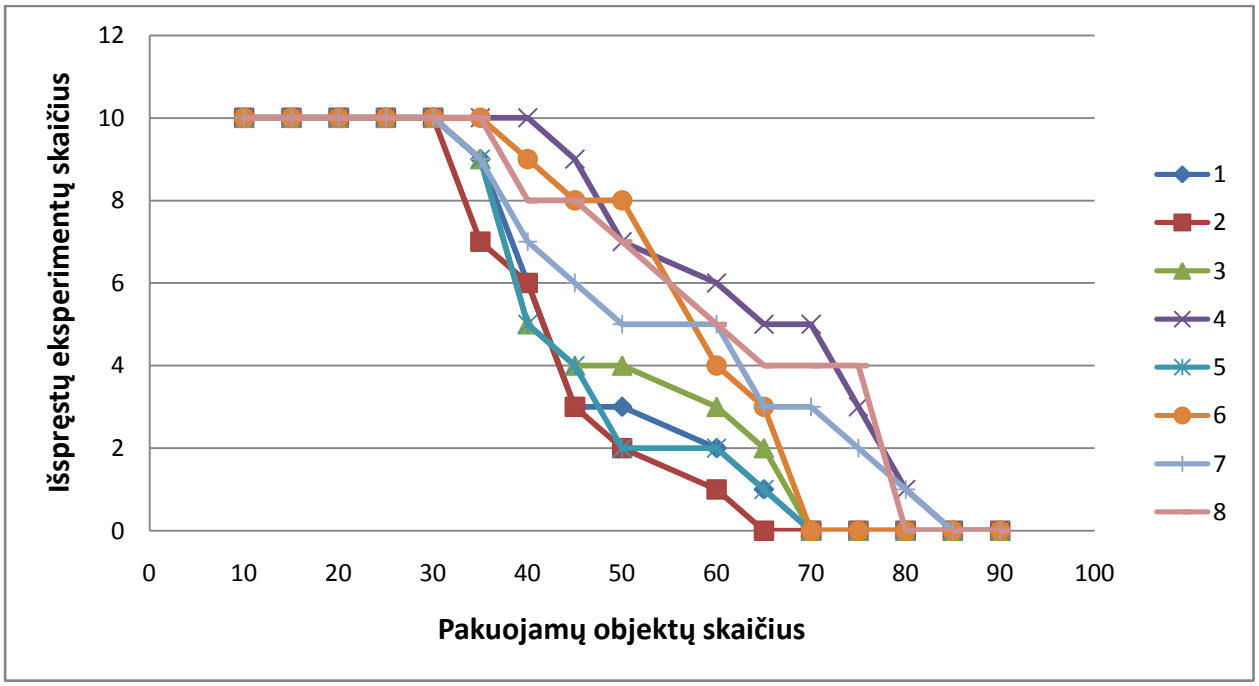
Remiantis ta pačia metodika kaip ir šakos ribos algoritmui tirsime tabu paieškos algoritmą. Taikysime 100s laiko limitą.

3.6. lentelė

Išspręstų eksperimentų remiantis optimalumo kriterijais skaičiavimo vidutiniai laikai

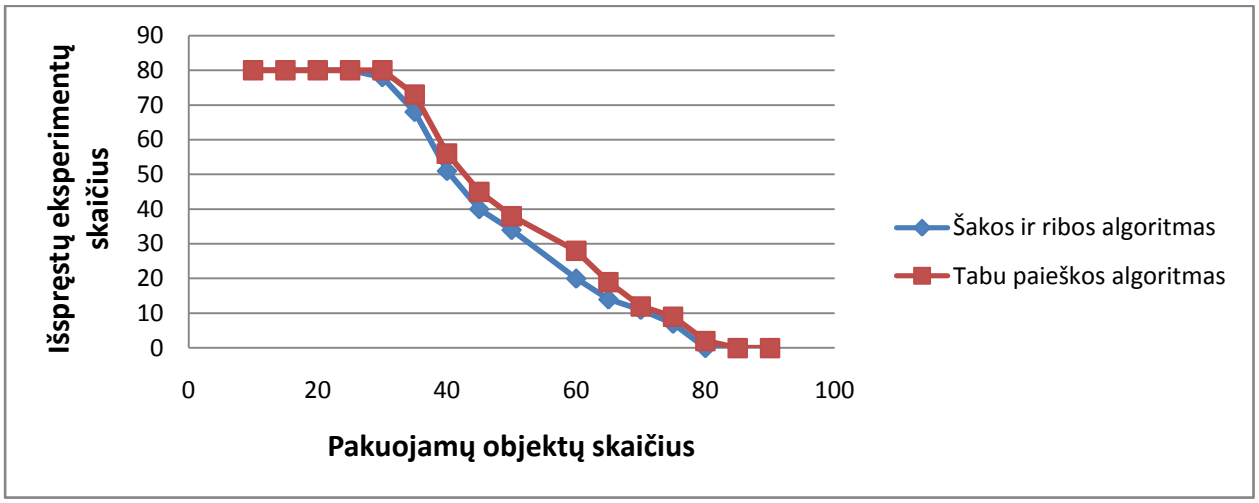
Objektų	Eksperimentinių duomenų klasė								
skaičius	1	2	3	4	5	6	7	8	
10	10	10	10	10	10	10	10	10	80
15	10	10	10	10	10	10	10	10	80
20	10	10	10	10	10	10	10	10	80
25	10	10	10	10	10	10	10	10	80
30	10	10	10	10	10	10	10	10	80
35	9	7	9	10	9	10	9	10	73
40	6	6	5	10	5	9	7	8	56
45	3	3	4	9	4	8	6	8	45
50	3	2	4	7	2	8	5	7	38
60	2	1	3	6	2	4	5	5	28
65	1	0	2	5	1	3	3	4	19
70	0	0	0	5	0	0	3	4	12
75	0	0	0	3	0	0	2	4	9
80	0	0	0	1	0	0	1	0	2
85	0	0	0	0	0	0	0	0	0

90	0	0	0	0	0	0	0	0	0



3.5 pav. Išspręstų eksperimentų remiantis optimalumo kriterijais skaičius

Atlikus eksperimentus gavome kai skaičiavimo laikas apribotas 100s, kad šakos ir ribos algoritmas gauna blogesnius išspręstų eksperimentų rezultatus. Tai pavaizduota žemiau:

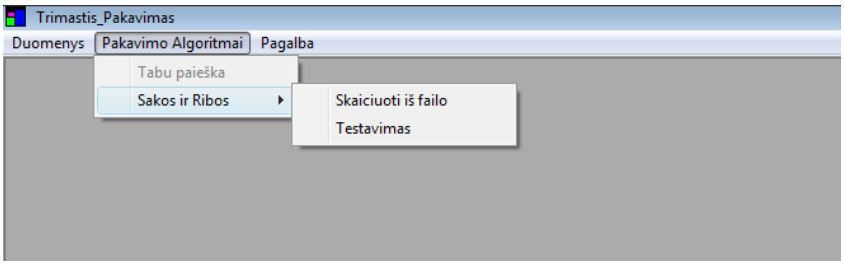


3.6 pav. Algoritmų palyginimas

4. PROGRAMINĖ REALIZACIJA

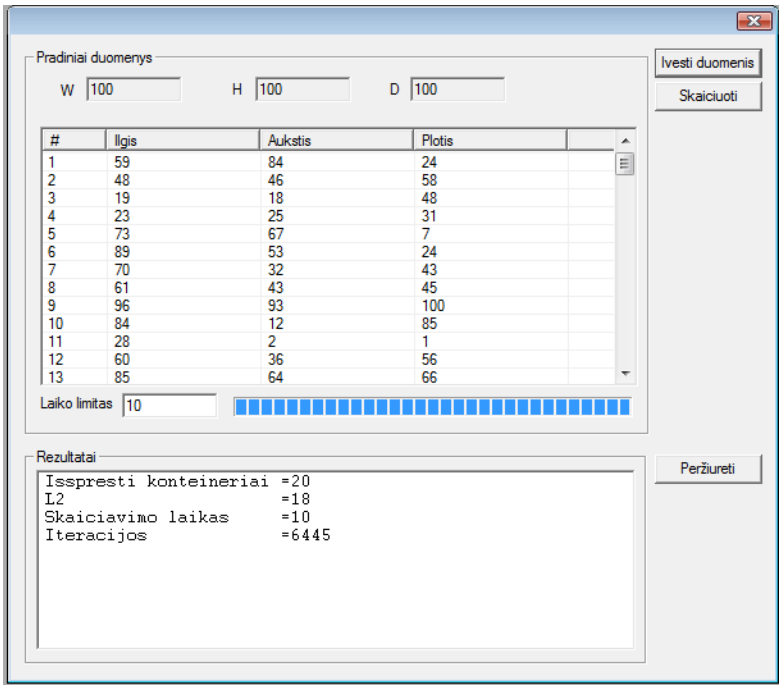
Buvo sukurta programa, kuri realizuota c++ programavimo kalba su „Microsoft Visual Studio 2008“ programiniu paketu. Programos tikslas realizuoti dviejų pakavimo algoritmų „Šakos ir Ribos“ ir „Tabu paieškos“ skaičiavimą ir pateikti grafinius rezultatus. Programoje naudojama OpenGL ir objektinio programavimo technologijos.

Programa valdoma pagrindinio meniu pagalba.



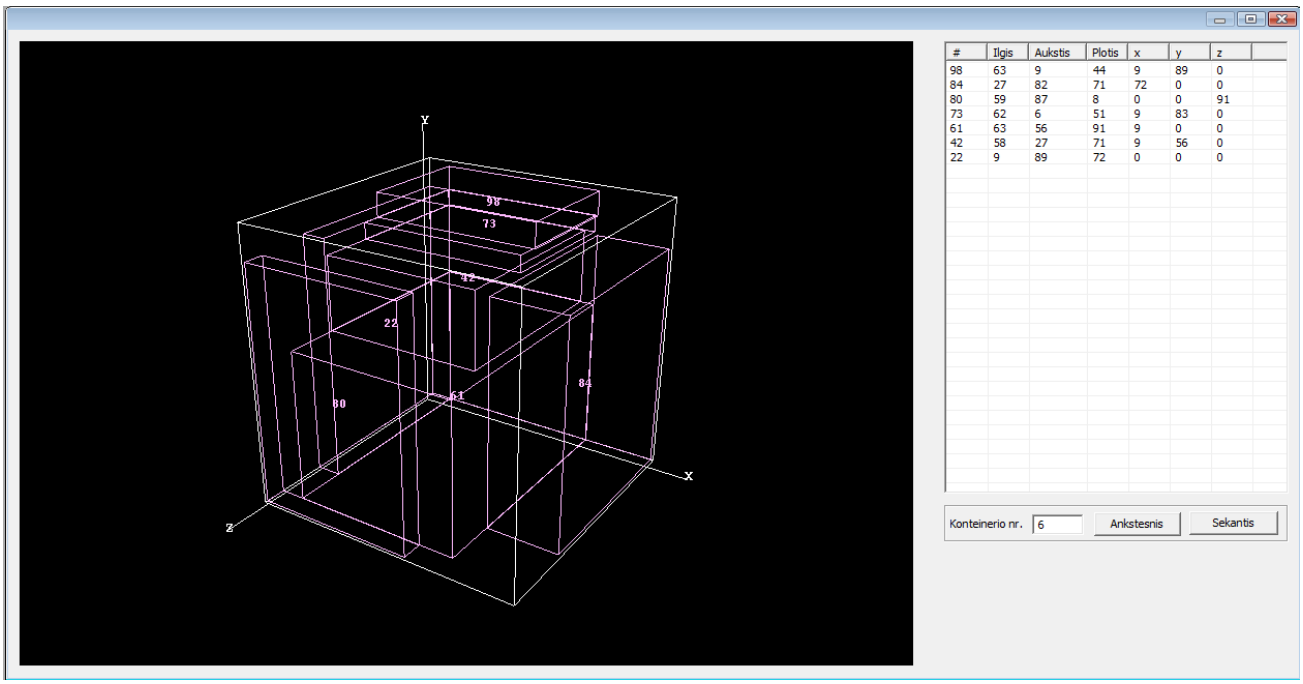
4.1 pav. Pagrindas programos meniu

Pasirinkus norimą algoritmą ir įvedam pradinius duomenis iš failo paspadus mygtuką “Ivesti duomenis”. Nustatom maksimalų algoritmo vykdymo laika sekundėm ir spaudžiam “Skaičiuoti”.



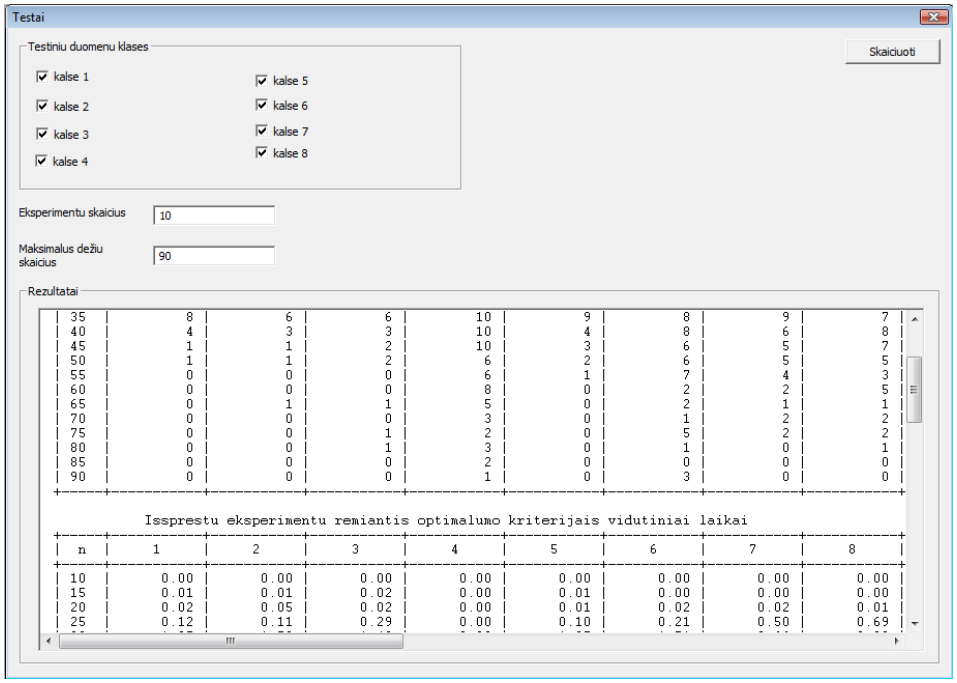
4.2 pav. Algoritmo skaičiavimo dialogas

Norint peržiūrėti supakuotus konteinerius spaudžiam mygtuką „Peržiūrėti“.



4.3 pav. Algoritmo rezultatų peržiūra

Algoritmų testavimas atliekamas pasirenkant viena arba keletą testinių pradinių duomenų klasių. Nustatomas eksperimentų skaičius ir maksimalus dėžių kiekis.



4.4 pav. Algoritmo testavimo dialogas

Pradiniai algoritmų duomenys įvedami iš tekstinio failo. Duomenų failo struktūra pateikta žemiau:

```
 59 84 24
 48 46 58
 19 18 48
 23 25 31
 73 67  7
 89 53 24
 70 32 43
 61 43 45
96 93 100
```

4.5 pav. Duomenų failas

Pirmame stulpelyje j tosios dėžės aukštis h_j, antrame stulpelyje jos plotis w_j ir ilgis d_j. Duomenys turi būti korektiški t.y. išlaikyta failo struktūra, nes kitaip programos veikimas gali gražinti klaidingus rezultatus. Skaičiai turi būti skiriami bent vienu tarpo simboliu.

IŠVADOS

- Algoritmų rezultatai labai priklauso nuo pradinės duomenų aibės. Tas pats algoritmas su vienokio tipo duomenims gauna tenkinamą rezultatą, o su kitokio tipo optimalus sprendinys nepasiekiamas.
- Algoritmų optimalumas gali būti vertinamas įvairiais aspektais nebūtinai mažiausio užimamo tūrio prasme.
- Algoritmai yra labai specifikuoti. Jie tinka tik trimačio pakavimo problemos kurios turi vieną arba kelis apribojimus. Praktiškai nėra algoritmų sprendžiančių realaus pakavimo uždavinio problemos
- Dauguma iš jų dirba su riboto kiekio duomenimis, išaugus jų kiekiu arba algoritmas nebeveikia arba gauti optimalų sprendinį mažai tikėtina. Abu nagrinėti algoritmai negali tiksliai spręsti problemos kurios apimtis didesnė už 35 objektus.

LITERATŪROS SĄRAŠAS

1. Dyckhoff, H., 1990. A typology of cutting and packing problems. European Journal of Operational Research 44, 145–159.

2. Gerhard Wäscher, Heike Haußner, Holger Schumann., 2007. An improved typology of cutting and packing problems. European Journal of Operational Research 183, 1109–1130.

3. Lins, L., Lins, S., Morabito, R., 2002. An n-tet graph approach for non-guillotine packings of n-dimensional boxes into an n-container. European Journal of Operational Research 141,421–439.

4. S.Martello, D.Pisinger, D.Vigo., 2000. The three-dimensional bin packing problem. Operations Research, 48, 256-267.

5. Marco A. Boschetti., 2004. New lower bounds for the three-dimensional finite bin packing problem. Discrete Applied Mathematics 140, 241 – 258.

6. A. Lodi, S. Martello, D. Vigo,. 2004. Tspack: A unified tabu search code for multi-dimensional bin packing problems. Annals of Operations Research 131, 203–213.

7. G. Perboli, Bounds and heuristics for the packing problems, Ph.D. thesis, Politecnico di Torino, available at http://www.orgroup.polito.it/People/perboli/phd-thesys.pdf 2002

8. O. Faroe, D. Pisinger,M. Zachariasen, Guided local search for the three-dimensional bin packing problem, INFORMS Journal on Computing 15 (3) (2003) 267–283.