KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

Evelina Stanevičienė

GILJOTININIO PJAUSTYMO METODAS IR JO TYRIMAS

Magistro darbas

Vadovas
prof. dr. D. Rubliauskas

KAUNAS, 2012

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

TVIRTINU
Katedros vedėjas
prof. dr. D. Rubliauskas
2012-05-25

GILJOTININIO PJAUSTYMO METODAS IR JO TYRIMAS

Informatikos inžinerijos magistro baigiamasis darbas

Vadovas
prof. dr. D. Rubliauskas
2012-05-24

Recenzentas
prof. K. Motiejūnas
2012-05-25

Atliko
IFN 0/1 gr. stud.
Evelina Stanevičienė
2012-05-22

KAUNAS, 2012

TURINYS

Summary

The theme of this master Thesis is “Guillotine cutting method and its analysis”. The main aim of this thesis is to design and to realize two-dimensional guillotine cutting program, which will pack given rectangles in slab, leaving the minimum of waste.

The analytical part of the Thesis overviews the guillotine cutting methods and algorithms descripted by other authors, short introduces the tasks of combinatorial optimization, submits the main advantages and disadvantages of commercial packaging programs available on the market.

The design part of this Thesis consists of projecting system’s requirements specification, in which are described the main functions and requirements of developing software, system models, algorithm descriptions and testing results of created software.

The comprehensive system manual is submitted in the consumers’ documentation part, which describes in detail the menu bar and calculation functions of created program.

In the experimental part is submitted the comparison of the results of in slab rectangular packing program with in analytical part described results of commercial programs.

Paveikslų sąrašas

Lentelių sąrašas

1. ĮVADAS

Su pjovimo ir pakavimo problemomis susiduriama daugelyje pramonės šakų. Medžio, stiklo ir popieriaus pramonės daugiausiai susijusios su taisyklingų formų ruošinių pakavimu arba pjovimu, tuo tarpu tekstilės ir odos pramonės labiau susijusios su netaisyklingų ruošinių pakavimo problema.

Pjaustymas yra priskiriamas optimizavimo uždaviniams, kurių tikslas yra rasti gerą keleto objektų (ruošinių) išdėstymą dideliame stačiakampiame objekte. Paprastai išdėstymo proceso tikslas yra minimizuoti žaliavos panaudojimą kartu minimizuojant nepanaudotą (atliekų) plotą. Šios problemos sprendimas yra aktualus masinės gamybos pramonės šakoms, kadangi nedidelis išdėstymo pagerinimas gali leisti sutaupyti nemažai žaliavų bei sumažinti gaminio savikainą [4].

Noint pasiekti, kad atliekų kiekiai būtų efektyviai sumažinti, **aktualu** nagrinėti pjaustymo uždavinio metodus, tirti jų ypatumus ir pritaikymo galimybes.

Pjaustymo ir pakavimo uždavinys priskiriamas kombinatorinio optimizavimo užduočių klasei. Šiems uždaviniams yra būdinga tai, kad didėjant jų apimčiai, visais atvejais tiksliai išspręsti juos per trumpą sprendimo laiko tarpą pasidaro nebeįmanoma.

Šiame darbe bus nagrinėjamas stačiakampių pakavimo plokštėje uždavinys.

Darbo tikslas. Suprojektuoti ir realizuoti dvimačio giljotininio pjaustymo programą, kuri duotus stačiakampius supakuotų plokštėje, paliekant kuo mažiau atraižų.

Darbo uždaviniai:

- Išanalizuoti kitų autorių sukurtus ir aprašytus pakavimo algoritmus ir metodus.
- Išanalizuoti komercinę programinę įrangą, skirtą pjaustymo uždaviniams realizuoti.
- Sukurti programinės įrangos produktą, kuris per nedidelį laiką pateiktų optimalų arba artimą jam sprendinį.
- Suprojektuoti aiškią ir lengvai įsisavinamą vartotojo sąsają.
- Atlikti eksperimentinį tyrimą, lyginant sukurto produkto pateikiamus rezultatus su darbe apžvelgtomis komercinėmis programomis.

Metodai. Mokslinės litratūros analizė, programavimas.

2. ANALITINĖ DALIS

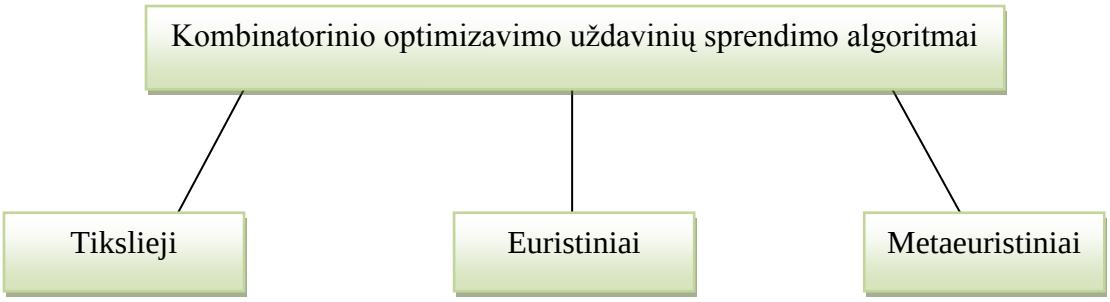
2.1 Kombinatorinio optimizavimo uždavinio samprata

Pramonėje, komercinėje veikloje ir daugelyje kitų veiklos sričių dažnai keliami uždaviniai, susiję su veiklos efektyvumu. Pavyzdžiui, architektas nori žinoti kaip suprojektuoti gamyklos patalpas, kad atstumai tarp įrenginių būtų kuo mažesni, laivų planuotojas nori suprojektuoti laivo patalpas taip, kad jos būtų kuo efektyviau panaudojamos gabenant krovinius. Telekomunikacijų specialistai nori žinoti kaip siusti signalą kanalais, kad klaidos tikimybė būtų kuo mažesnė. Visuotinio transporto strategai nori žinoti kaip sudaryti autobusų, traukinių tvarkaraščius, kad keleivių pervežimas būtų kuo efektyvesnis. Taip pat gamyklose labai svarbu paskirstyti operacijas laike taip, kad bendras gamybos laikas būtų kuo mažesnis, nes papildomas gamybos laikas reiškia bereikalingas papildomas sąnaudas [6].

Formaliai kombinatorinio optimizavimo uždavinį galima aprašyti pora (*S*, *f*). *S* yra sprendinių aibė (angl. *set of feasible solutions*). Antrasis poros (*S*, *f*) narys yra funkcija, kurios apibrėžimo sritis — aibė *S*, o reikšmių aibė — realieji skaičiai. Ši funkcija vadinama tikslo funkcija (TF) (angl. *objective function*). TF pobūdis ir išraiška priklauso nuo konkretaus sprendžiamo uždavinio. Labai dažnai KO uždavinių tikslo funkcijos yra netiesinės, nediferencijuojamos, daugiaekstremės. Neprarasdami universalumo tarsime, kad tikslo funkcija f turi būti minimizuojama. Tokiu būdu, išspręsti uždavinį (*S*, *f*) reiškia surasti sprendinį $s^* \in S$ ir tokį, jog $s^* \in S^* = \left\{ s^\nabla \mid s^\nabla = \arg\min_{s \in S} f(s) \right\}$. Sprendinys s^* vadinamas uždavinio (*S*, *f*) (globaliai) optimaliu sprendiniu, o aibė $S^* \subseteq S$ — optimalių sprendinių aibe [8].

Kombinatorinis optimizavimas – palyginus su kitomis, jauna matematikos atšaka, išsivysčiusi XX a. antroje pusėje kartu su skaičiuojamosios technikos raida ir sparčiai besivystanti iki šių dienų.

Visus kombinatorinio optimizavimo uždavinių sprendimo metodus - algoritmus galima būtų suskirstyti į tris pagrindines grupes: tiksliuosius, euristinius ir metaeuristinius (1 pav.).



1 pav. Kombinatorinio optimizavimo uždavinių sprendimo algoritmų rūšys (6)

Tikslieji algoritmai garantuoja uždavinio optimalaus sprendinio radimą. Tačiau bendru atveju beveik visų kombinatorinio optimizavimo uždavinių, tiksliųjų sprendimo algoritmų vykdymo laikas auga eksponentiškai, didėjant uždavinio apimčiai.

Euristiniu laikomas toks optimizavimo uždavinio sprendimo metodas, kuriuo siekiama rasti aukštos kokybės, bet nebūtinai optimalų sprendinį per priimtiną skaičiavimų laiką. Tai yra pagrindinis euristinių algoritmų skirtumas nuo anksčiau minėtų tiksliųjų metodų. Šios rūšies algoritmai remiasi tam tikrų taisyklių, kurios vadinamos euristikomis, pritaikymu sprendžiant konkretų kombinatorinio optimizavimo uždavinį.

Metaeuristinius metodus apibrėžiame kaip tam tikro aukšto abstrakcijos lygio nurodymų rinkinius. Priešingai nei euristinių algoritmų, tokių nurodymų paskirtis yra formaliai aprašyti kurios nors klasės uždavinių sprendimo idėją, principą. Taigi, sąvoka „metaeuristinis algoritmas“ yra talpesnė nei „euristinis metodas“ .

Įvairių optimizavimo uždavinių sprendimo algoritmuose galima pritaikyti praktiškai visų minėtų metaeuristinių nurodymų idėjas. Tai būtų pagrindinis metaeuristinių metodų pranašumas prieš kokiam nors vienam konkrečiam uždaviniui spręsti tinkamus euristinius algoritmus. Norint rasti geresnės kokybės sprendinius, dėl metaeuristinių metodų universalumo, dažnai galime apjungti kelių skirtingų rūšių metaeuristinių algoritmų idėjas. Todėl šiuo metu metaeuristiniai algoritmai ir įvairios jų ir kelių metaeuristinių metodų idėjų kombinacijos, pavyzdžiui genetiniai algortimai ir lokalioji paieška, tabu paieška ir atkaitinimo modeliavimas ir kt., yra pagrindiniai tyrinėjimų objektai, kuriais siekiama sukurti algoritmus, kuo efektyviau sprendžiančius kombinatorinio optimizavimo uždavinius, t.y. per tam tikrą skaičiavimų laiką gauti optimalius ar jiems artimus sprendinius [6].

2.2 Problemų sudėtingumo klasės

Sudėtingumo klasė yra aibė problemų susietų su panašiu sudėtingumu. Skiriamos dvi pagrindinės klasės P ir NP.

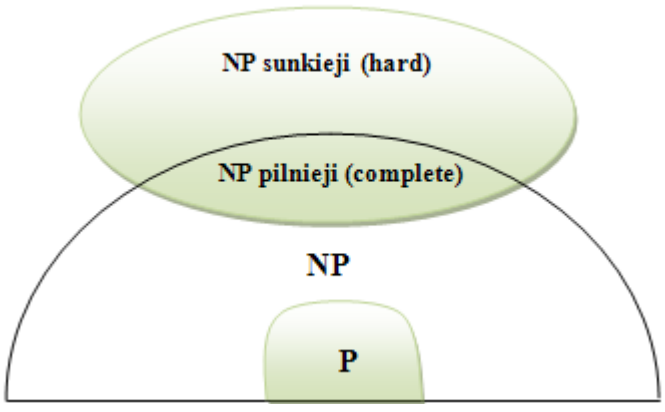
P – tai tokia sudėtingumo klasė, apjungianti problemas, kurias galima išspręsti deterministine mašina per polinominį laiką, t.y. $m(n) = O(n^k)$, kur k – konstanta, priklausanti nuo problemos, n – problemos dydis, $\frac{m(n)}{O(n^k)} = 1,\ n \to \infty$.

NP – tai tokia sudėtingumo klasė, apjungianti problemas, išsprendžiamas nedeterministine Tiuringo mašina per polinominį laiką – *m(n)* [11].

Uždavinys A priklauso **NP-hard** uždavinių klasei (NP sunkių), jei egzistuoja toks uždavinys B ∈NP-complete, kurį polinominio sudėtingumo algoritmu galime transformuoti į A.

Taigi NP-hard uždaviniai yra ne mažiau sudėtingi už NP-complete uždavinius, nes, jei pavyktų greitai išspręsti. NP-hard uždavinį, tai mokėtume greitai išspęsti ir visus NP-complete uždavinius. Pastebėkime, kad NP-hard uždaviniams nėra reikalavimo, kad jie būtų iš NP. Todėl NP-hard uždavinys nebūtinai turi būti sprendimo priėmimo uždaviniu (pvz., optimizavimo). NP-hard uždaviniui gali būti ir nežinomas polinominis sprendinio patikrinimo algoritmas [13].

Akivaizdu, kad $(P \subseteq NP)$, tačiau ar P = NP ? Daugelis mano, kad atsakymas yra neigiamas. Tačiau dar ši problema nėra įrodyta ir tam, kuriam pavyks, tai padaryti, yra numatyta 1 milijono dolerių premija [11]. Sudėtingumo klasės pavaizduotos 2 paveiksle.



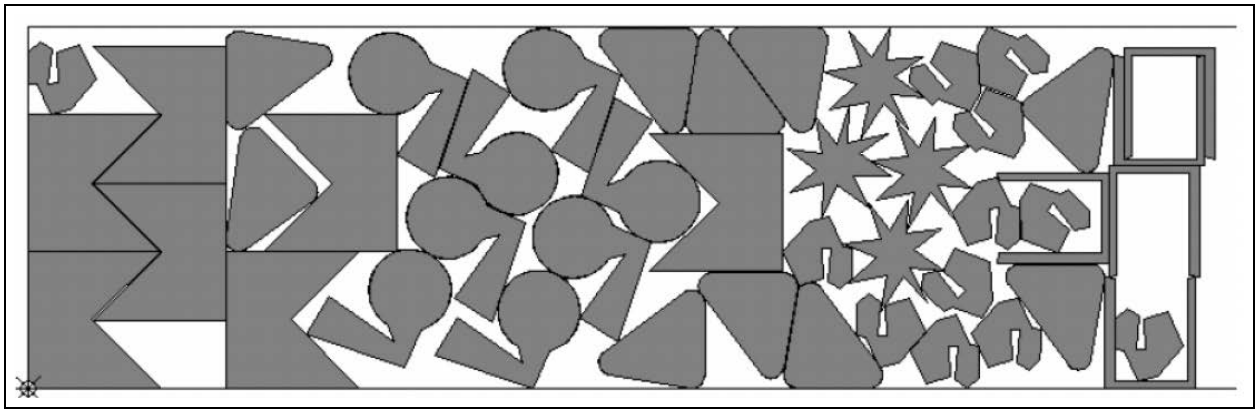
2 pav. Sudėtingumo klasių diagrama (16)

NP – Complete tai pačios sudėtingiausios problemos NP klasėje, kurios greičiausiai nepriklauso klasei P. Dar niekam nepavyko sukurti algoritmo, kuris per polinominį laiką

visada tiksliai (visais atvejais) išspręstų NP – Complete problemą. Verta paminėti, kad egzistuoja polinominio laiko algoritmai, sprendžiantys tokio tipo problemas, esant prielaidai, kad $P = NP$ [11].

2.3 Pjautymo uždavinio kilmė

Pjaustymas – tai fizinio objekto arba jo dalies padalijimas į dvi ar keletą naujų dalių, panaudojant pjaustymo įrankį. Akivaizdu, kad pjaustymo įrankiai gali būti labai įvairūs, taip pat ir pjaustymo objektai bei išpjovos gali būti visokiausių formų (3 pav.):



3 pav. Įvairių formų detalių pjaustymas (11)

Kai smulkios detalės (išpjovos) pjaustomos iš didelių objektų (lakštų), iškyla atliekų minimizavimo problema, t.y. kokiu būdu pjaustyti pasirinktą objektą, kad atliekų kiekis būtų mažiausias. Literatūroje tokia situacija dažnai sutinkama pavadinimu žaliavų pjaustymo problema (Cutting Stock Problem, CSP).

2.3.1 Pjaustymo problemos klasifikacija

Pjaustymo problema gali būti klasifikuojama pagal Dyckhoff pasiūlytą schemą. Pradžioje išskiriamos dvi pagrindinės grupės:

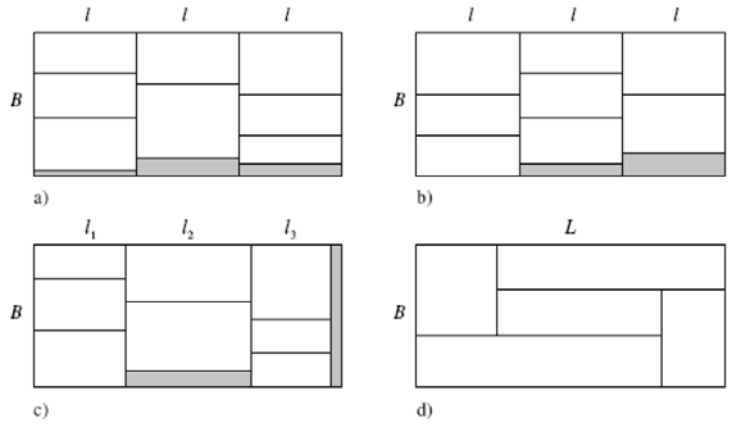
- Orientuotas į išpjovą būdas (pjaustymo metu kiekviena išpjova išpjaunama individualiai);
- Orientuotas į šabloną būdas (pirmiausiai iš išpjovų yra sudaromi šablonai, kuriems nustatomi intensyvumai, užsakymui patenkinti) [11].

Be šių grupių apibrėžiamos keturios pagrindinės charakteristikos:

1. ***Dimensijų skaičius*** - – tai minimalus skaičius dimensijų, kurios yra reikšmingos sprendimo nustatymui. Paprasčiausias yra vienos dimensijos atvejis, tipinis pavyzdys būtų plieno strypų pjaustymas, kurių ilgis yra fiksuotas. Dviejų dimensijų atveju detalės pozicionuojamos objekte dviejų kintančių dimensijų atžvilgiu ir t.t.

   Dviejų dimensijų atveju, pjaustymo procesas gali būti sudarytas iš atskirų žingsnių, kuriuose objektas pjaustomas į paobjekčius atitinkamais kampais, atsižvelgiant į ankstesnius pjūvius – tai *fazinis dvidimensinis pjaustymas*. Jei pjūviai daromi lygiagrečiai lakšto šonui, tuomet problemą vadinsime *ortogonaliąja dvidimensine*. Pagaliau, pjūvį, kuris užima visą lakšto arba sublakšto, kuris ankstesnio pjūvio rezultatas, plotį – *giljotininiu*.

   Skirtingos *CSP* pjaustymo procedūros pavaizduotos 2.4 paveiksle: a) ir b) *vienadimensinis CSP* atvejis, pjaustomo objekto B ilgis fiksuotas. *Dvidimensiniu CSP* atveju c) išpjovų ilgiai skiriasi. Be to a) – c) atvejais turime *ortogonalųjį dvifazį giljotininį pjūvį,* o d) – pjūviai nėra giljotininiai.

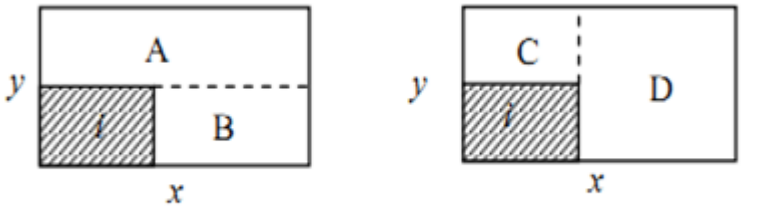


4 pav. Pjūvių pavyzdžiai (5)

2. ***Žaliavų kiekis.*** Konkrečiai *CSP* klasifikacijai mažiausiai reikia dviejų kategorijų. Pirmoji, visos žaliavos bus sunaudotos, bet ne visi užsakymai bus įvykdyti. Antroji – visi užsakymai bus įvykdyti.
3. ***Lakštų asortimentas.*** Ši charakteristika gali būti suskirstyta į tris tipus. Pirmas, yra tiktai vienas didelis lakštas, antras – daug didelių lakštų su panašiomis dimensijomis, trečias – daug lakštų su skirtingomis dimensijomis.
4. ***Išpjovų asortimentas.*** Pirmu atveju nedaug išpjovų su skirtingomis dimensijomis, antruoju – daug išpjovų su daug skirtingų dimensijų, trečiuoju – daug išpjovų su panašiomis dimensijomis, ketvirtuoju – daug identiškų išpjovų [11].

2.4 Dvimatis giljotininis pjaustymas

Giljotininis pjūvis – tai pjūvis kuis prasideda nuo vienos stačiakampio briaunos iki priešingos briaunos, tiesiniu pjūviu. Dviejų dimensijų giljotininis pjūvis pavaizduotas 5 paveiksle.

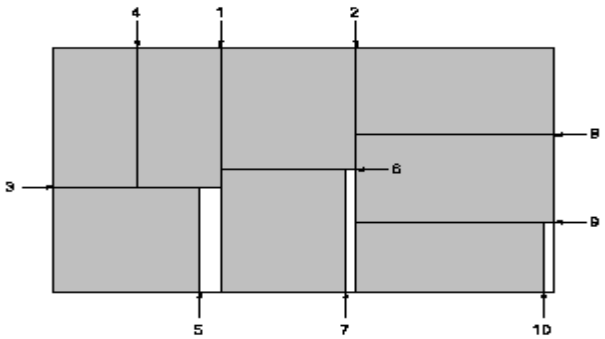


a) horizontalus giljotininis pjūvis; b) vertikalus giljotininis pjūvis

5 pav. Vertikalus ir horizontalus giljotininiai pjūviai (12)

Pjaustymo uždaviniuose ruošiniai dažniausiai yra stačiakampio formos, sprendimo radimui naudojami taip vadinami dvimačio supjaustymo algoritmai. Tokiose pramonės šakose kaip stiklo ar baldų ruošinių pjaustymui yra naudojami pjaustymo įrenginiai galintys atlikti tik giljotininius pjūvius, todėl šiose veiklos srityse naudojami taip vadinami giljotininio pjaustymo algoritmai. Giljotina pasižymi tuo, kad pjūvis gali būti daromas tik nuo vieno plokštės krašto iki kito, be to pjūvio linija turi būti statmena pjaunamai kraštinei [3].

Galima būtų taip suformuluoti dvimačio giljotininio pjaustymo uždavinį: Duotas baigtinis ruošinių kiekis n , kurių ilgiai h_i ir pločiai w_i yra žinomi. Taip pat turime teoriškai begalinę aibę pjaustomų plokščių $\{R_1, R_2,....R_n\}$, kurių ilgiai H_j ir pločiai W_j taip pat žinomi. Reikia supjaustyti ruošinius taip, kad būtų sunaudotas minimalus pjaustomų plokščių kiekis ir atliekų kiekis būtų minimalus. Galimi tik giljotininiai pjūviai. Siekiant gauti mažą atliekų plotą ruošinius galima sukioti 90º kampu. Giljotininio supjaustymo pavyzdys pateiktas 6 paveiksle.



6 pav. Galimi giljotininiai pjūviai (2)

2.5 Uždavinio sprendimo būdų analizė

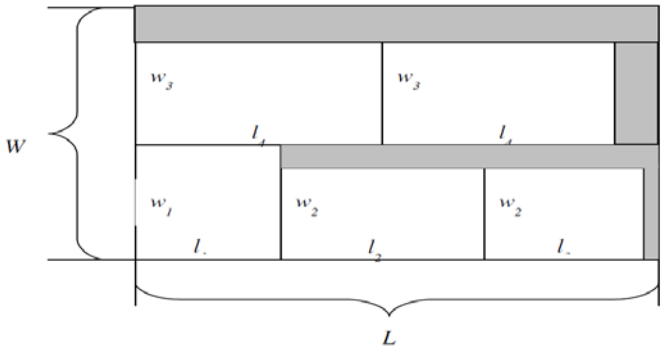
Dauguma kombinatorinio optimizavimo uždavinių priklauso *NP-hard* klasei ir negali būti išspręsti per polinominį laiką. Taigi naudojami euristiniai algoritmai, arba taip vadinamos euristikos, tam kad rasti sprendinį artimą optimaliam. Eurisitiniai algoritmai suranda labai gerus sprendinius per labai trumpą laiką, tačiau negali garantuoti optimalaus sprendinio. Dažnai net neįmanoma pasakyti kaip rastasis sprendinys yra arti optimalaus sprendinio. Euristiniai algoritmai taip pat gali būti suprantami kaip intelektualūs metodai, kurie remiasi žmogaus patirtimi, kuri savo ruožtu remiasi procesais vykstančiais aplinkiniame pasaulyje (fizika, gamta, biologija ir t.t.) [7].

Darbe nagrinėsime tokį pjaustymo uždavinį, kai lakštai, iš kurių bus pjaustomos figūros ir pačios figūros yra stačiakampiai (kvadratai). Atliekami pjūviai bus giljotininiai.

Tarkime, kad turime stačiakampę plokštę LxW (L – ilgis, W plotis), kurią reikia supjaustyti į m stačiakampių dalių l_i x w_i, $i=1,..,m$. Kiekviena dalis turi teigiamą dydį v_i (dalies vertė) ir skaičių b_i, kuris nurodo maksimalų dalių skaičių bet kuriame pjaustymo plane. Tegu a_i būna dalių l_i x w_i esančių pjaustymo plane skaičius. Tada 2-fazių dviejų dimensijų pjaustymas su apribojimais gali būti išreikštas formule:

Maksimizuoti $f(a_1,...,a_m)=\sum_{i=1}^{m} v_i a_i$ atsižvelgiant į tai, kad (a_1, a_2, …, a_m) atitinka 2-fazių giljotininio pjovimo planą LxW ir $0 \le a_i \le b_i$, $i=1,...,m$.

Reikia pastebėti, kad jei $v_i = \frac{l_i w_i}{LW},...$ $i=1,...,m$, tai užduotis tampa ekvivalenti užduočiai minimizuoti nuostolius. Tolesnį užduoties formulavimą atliksime ta pačia eiga kaip ir pjaustymo užduoties be apribojimų t.y. darysime prielaidą, kad primas pjūvis yra horizontalus ir sudarysime vienos dimensijos pjovimo planus juostoms Lxw_j, $j=1,…,m$. Tada nustatysime kiek kartų šie pjovimo planai yra atkartojami išilgai juostos pločio.



7 pav. Dviejų dimensijų pjovimo planas (11)

2.5.1 Užduoties formuluotė

Vienos dimensijos w_j – juostos pjovimo planas gali būti aprašomas tokiu vektoriumi:

$$(\alpha_{kj}^1 \ \alpha_{kj}^2 \ \ldots \ \alpha_{kj}^m), \quad j = 1,\ldots,r$$

atsižvelgiant į:

$$l_1\alpha_{kj}^1 + l_2\alpha_{kj}^2 + \ldots + l_m\alpha_{kj}^m \le L$$

$$\alpha_{kj}^i \ge 0 \ \textit{ir sveikieji skaičiai}$$

$$\alpha_{kj}^i = 0 \quad \textit{jei} \quad w_i > w_k, \quad i = 1,\ldots,m$$

čia r – reiškia juostų, turinčių skirtingą plotį, skaičių;

α_{kj}^i – dalių kurių tipas yra i ir *k-asis* pjovimo planas juostos Lxw_j ir $k = 1,\ldots,K_j = \left[\frac{W}{w_j}\right]$.

Pavadinkime *k-ajį* juostos Lxw_j pjovimo planą – *(k,j)-planu*. Apsibrėžkime aibę $W_j = \{i \mid w_i \le w_j\}$. Tada *(k,j)-planas* apibrėžiamas taip:

$$\sum_{i \in W_j} l_i \alpha_{kj}^i \le L,$$

$\alpha_{kj}^i \ge 0$ *ir sveikieji skaičiai* $k = 1,\ldots,K, \ j = 1,\ldots,r$.

Dabar pažymėkime *(k,j)-plano* panaudojimo skaičių visame dvimačiame pjaustyme – β_{kj}. Atkreipkime dėmesį į tai, kad visame dvimačiame pjaustyme panaudotų Lxw_j – juostų skaičius lygus

$$\sum_{k=1}^{K_j} \beta_{kj}$$

Taigi

$$\sum_{j=1}^{r} w_j \sum_{k=1}^{K_j} \beta_{kj} \le W$$

Taip pat suminis panaudotų i dalių skaičius yra lygus:

$$a_i = \sum_{j=1}^{r} \sum_{k=1}^{K_j} \alpha_{kj}^i \beta_{kj}$$

ir šis skaičius negali būti didesnis nei b_i

$$a_i = \sum_{j=1}^{r}\sum_{k=1}^{K_j}\alpha_{kj}^{i}\beta_{kj} \le b_i, \quad i = 1,...,m$$

Taigi matematinis modelis aprašantis dvimatį giljotininį pjaustymą:

$$\max_{f(\alpha,\beta)} \sum_{j=1}^{r}\sum_{i=1}^{m}\sum_{k=1}^{K_j} V_i \alpha_{kj}^{i}\beta_{kj}$$

Atsižvelgiant į:

$$\sum_{i\in W_j} l_i \alpha_{kj}^{i} \le L \quad j = 1,...,r \quad k = 1,...,K_j \tag{1}$$

$$\sum_{j=1}^{r} w_j \sum_{k=1}^{K_j}\beta_{kj} \le W \tag{2}$$

$$\sum_{j=1}^{r}\sum_{k=1}^{K_j}\alpha_{kj}^{i}\beta_{kj} \le b_i, \quad i = 1,...,m \tag{3}$$

$$\alpha_{kj}^{i} \ge 0,\ \beta_{kj} \ge 0 \ \textit{ir sveikieji skaičiai}\ i = 1,...,m,\ k = 1,...,K,\ j = 1,...,r \tag{4}$$

Užrašyta užduotis yra sveikaskaitė netiesinė optimizavimo užduotis. Jei pašalintume (3) apribojimą tada optimizavimo uždavinys taptų dviejų dimensijų dviejų fazių giljotininio pjaustymo uždaviniu be apribojimų. Gilmore ir Gomory (1965) pasiūlė dekompozicijos metodą spręsti dviejų fazių dviejų dimensijų giljotininio pjaustymo uždavinį. Šis metodas gali būti išvestas iš anksčiau pateikto modelio (vėliau šis metodas bus praplėstas iki giljotininio pjaustymo užduoties su apribojimais):

(k,j)-plano vertę pažymėkime $V_{kj} = \left(\sum_{i=1}^{m}\alpha_{kj}^{i} v_i\right)$ ir perrašykime sudarytą modelį tik be (3) apribojimo:

$$\max_{j,k} \sum_{j=1}^{r}\sum_{k=1}^{K_j} V_{kj}\beta_{kj} \tag{5}$$

Atsižvelgiant į:

$$\sum_{i=1}^{m} l_i \alpha_{kj}^{i} \le L \quad j = 1,...,r \quad k = 1,...,K_j \tag{6}$$

$$V_{kj} = \sum_{i=1}^{m} v_i \alpha_{kj}^{i} \quad j = 1,...,r \quad k = 1,...,K_j \tag{7}$$

$$\sum_{j=1}^{r} w_j \sum_{k=1}^{K_j}\beta_{kj} \le W \tag{8}$$

$$\alpha_{kj}^{i} \ge 0,\ \beta_{kj} \ge 0 \ \textit{ir sveikieji skaičiai}\ i = 1,...,m,\ j = 1,...,K_j \tag{9}$$

Kadangi pjaustymo užduotis yra be apribojimų t.y. pašalintas apribojimas maksimaliam dalių skaičiui ((3) sąlyga), tai mums reikalingas tik vienas, geriausias, pjovimo

planas w_j-juostai. Todėl galima praleisti indeksą *k* iš modelio formuluotės ir pirmoje metodo stadijoje imti tik geriausią w_j-juostos pjovimo planą:

$$V_j = \max \sum_{i \in W_i} V_i \alpha_j^i \quad j = 1,...,r \tag{10}$$

Atsižvelgiant į:

$$\sum_{i \in W_i} v_i \alpha_j^i \le L \quad \alpha_j^i \ge 0 \ \textit{ir sveikieji skaičiai} \ i \in W_j \tag{11}$$

Antroje metodo stadijoje yra sprendžiamas kuprinės uždavinys tam, kad nustatyti kiek kartų Lxw_j-juosta (kartu su geriausiu jos pjovimo planu) turi atsikartoti visame dvimačiame plane, t.y. $\textit{maksimizuoti} \sum_{j=1}^{r} V_i \beta_j$. Atsižvelgiant į: $\sum_{j=1}^{r} w_j \beta_j \le W \quad \beta_j \ge 0$ ir sveikieji skaičiai j = 1, 2,….γ [11].

2.5.2 Euristinis dekompozicijos metodas

Naudojant euristinį dekompozicijos metodą, visų prima reikia rasti geriausią venos dimensijos pjovimo planą visoms w_j – juostoms atsižvelgiant į maksimalų leistiną dalių į skaičių b_i.

Pirmas žingsnis: kiekvienam *j=1,…,r* (t.y. kiekvienai juostai), išspręsti kuprinės su apribojimais užduotį:

$V_j = \max \sum V_i \alpha_j^i$, atsižvelgiant į: $\sum_{i \in W_i} v_i \alpha_j^i \le L \quad 0 \le \alpha_j^i \ge b_i$ *ir sveikieji skaičiai* $i \in W_j = \{i \mid w_i \le w_j\}$.

Dabar turėdami pjovimo planus kiekvienai juostai, galime juos išdėstyti išilgai pjaunamo lakšto pločio.

Antras žingsnis: Išspręsti sveikaskaičio tiesinio optimizavimo užduotį: $\textit{maksimizuoti} \ \sum V_j \beta$.

Atsižvelgiant į: $\sum_{j=1}^{r} w_j \beta_j \le W$ (12)

$\sum_{j=1}^{r} \alpha_j^i \beta_j \le b_i, \ \ i = 1,...,m. \ \beta_j \ge 0 \ \textit{ir sveikieji skaičiai} \ j = 1,2,...,r.$ (13)

Reikia pastebėti, kad antrojo žingsnio užduotis jau nebėra kuprinės užduotis dėl maksimalaus dalių i skaičiaus apribojimo. Taip pat ši procedūra yra euristinė, nes imamas tik vienas (geriausias) pjovimas kiekvienai juostai, o optimalus sprendimas gali būti sudarytas iš

skirtingų pjovimų tai pačiai juostai. Be to antrojo žingsnio modelis nėra lengvai sprendžiamas ir reikalauja labai sudėtingo kodo palyginus su kuprinės užduotimi, todėl vietoj antrojo žingsnio spręsime kitą užduotį gautą pasinaudojus relaksacija (t.y. pakeisime apribojimų (12 ir 13) sąlygą), tam kad identifikuotume labiausiai vertingą juostą iš visų juostų sugeneruotų pirmame žingsnyje.

Antras žingsnis: sumuodami (12) padaugintą iš γ ir (13) padaugintą iš (1-γ), ir relaksavę sveikaskaitį apribojimą, turime tolydžiąją kuprinės užduotį:

maksimizuoti $\sum V_j \beta$.

$$Atsižvelgiant\ į\ \sum_{j=1}^{r}\left(\gamma w_j + (1-\gamma)\sum_{i=1}^{m}\alpha_j^i\right)\beta_i \le \gamma W + (1-\gamma)\sum_{i=1}^{m} b_i \quad \beta_j \ge 0 \quad j = 1,2,...,r \qquad (14)$$

kas yra tiesinio optimizavimo užduotis su vienu apribojimu, ir vienintelis indeksas *k* pagrindinio optimalaus kintamojo yra apskaičiuojamas pagal:

$$\frac{V_k}{\gamma w_k + (1-\gamma)\sum_{i=1}^{m}\alpha_k^i} = Max\left\{\frac{V_j}{\gamma w_j + (1-\gamma)\sum_{i=1}^{m}\alpha_k^i},\quad j = 1,...,r\right\} \qquad (15)$$

Tada naudojame w_k – juostą, kartu su geriausiu jos pjovimo planu gautu pirmame žingsnyje, kuri gali būti panaudota dviem būdais:

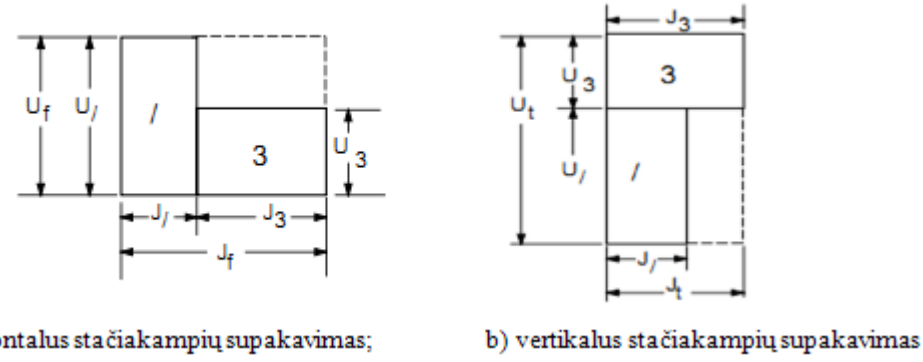
- Tiktai vieną kartą (t.y. βk=1, βj=0, j≠k);
- Keletą kartų iš eilės (t.y. $\beta_k = \text{minimum}\left\{\left[\frac{W}{w_k}\right], \left[\frac{b_i}{\alpha_k^i}\right],\ i = 1,...,m\right\},\ \beta_j = 0,\ j \ne k$).

  Tada atnaujinami W←W-$w_k\beta_k$ $W \leftarrow W - w_k\beta_k$ ir $b_i \leftarrow b_i - \alpha_k^i\beta_k$ ir kartojamas pirmasis žingsnis [11].

2.5.3 Genetinis algoritmas

Giljotininio pjaustymo problemos buvo nagrinėtos dauglio mokslininkų, taikant skirtingus metodus, tokius kaip euristinis metodas (heuristic approach (Ghandforoush, 1992)), AND/OR grafo metodas (Morabito, 1996), medžio sudarymo algoritmas (tree – seach algorithms (Christofides, 1995) ir kitus [3]. Tačiau kai kurie mokslininkai labiau linkę naudoti genetinių algoritmų galimybes, bandydami pagreitinti skaičiavimus bei mažindami atliekų kiekį. Dažnai šie sprendimo būdai apjungia kelias technologijas. Toshihiko Ono pateiktame sprendimo variante yra naudojami genetinis algoritmas (GAs) ir giljotininio nustatymo algoritmas (GCLAs). Paanalizuosime plačiau šį algoritmą [14].

Algoritmas susideda iš dviejų dalių, horizontalios ir vertikalios funkcijos pakavimas, ir nuoseklaus dalies numerio priskyrimas visoms dalims. Išlyginimas vyks taip, visos stačiakampio dalys jungsis rekursyviai tol, kol pabaigoje plokštėje susidarys vienas didelis stačiakampis. Šis susijungimas vyks pagal sudarytus algoritmus: supakavimas ir išlyginimas bus perduodamas genetiniam algoritmui (GAs), ir genetinis algoritmas perduos reikalingą plokštės ilgį. Sekantis veiksmas: genetiniame algoritme tikslus supakavimas modifikuojamas genetinės operacijos, kuri sukuria mažiausiai tikėtiną reikalingo ilgio meta stačiakampį. Susijungus šioms dviems operacijoms, pavaizduojamas optimaliausias dalių supakavimas [14].

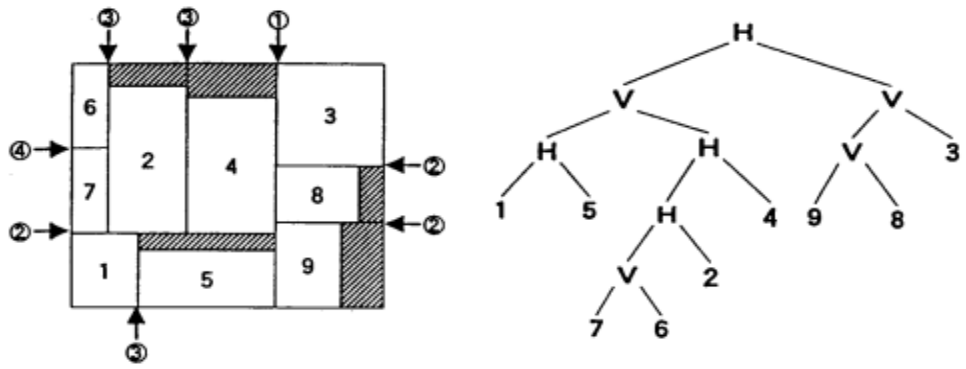


a) horizontalus stačiakampių supakavimas; b) vertikalus stačiakampių supakavimas

8 pav. Horizontalus ir vertikalus stačiakampių supakavimas plokštėje (14)

Sprendžiant optimalų stačiakampių pakavimą plokštėje galimi du būdai: iš viršaus žemyn ir iš apačios aukštyn.

Norint supakuoti visus stačiakampius plokštėje, naudojamas toks metodas: keletas stačiakampių jungiami į mažiausią stačiakampį, verčiant ir jungiant šiuos du stačiakampius horizontaliai ar vertikaliai vieną šalia kito. Pritaikant šitą metodą visiems stačiakampiams ir stačiakampių susijungimui, pabaigoje gaunamas vienas didelis stačiakampis.



a) stačiakampių išdėstymas; b) medžio pavaizdavimas.

9 pav. Stačiakampių supakavimas naudojant giljotininį pjaustymą (15)

Atliekant genetinį stačiakampių dalių pavaizdavimą ir pakavimą plokštėje, šiuo ateju yra naudojami dvinariai operatoriai: H ir V. H - operatorius, kaip argumentus paima du stačiakampius ir sukuria mažiausią stačiakampį, pakuojant juos horizontaliai. Pavyzdžiui, formulė „1 5 H“ reiškia, kad stačiakampis Nr.5 yra iš dešnės pusės stačiakampiui Nr.1, ir gražina identifikacinį šio integruoto stačiakampio numerį.

Taigi meta stačiakampio ilgis L_h ir plotis W_h apskaičiuojamas:

$$L_h = L_1 + L_5 \text{ , } W_h = \max(W_1, W_5), \quad (16)$$

kur, L_1 ir L_5 yra šių dviejų mažų stačiakampių ilgis, W_1 ir W_5 yra pločiai. Funkcija max (*, *) suskaičiuoja dviejų reikšmių maksimumą.

Operatorius V yra labai panašus į H operatorių, išskyrus tai, kad du stačiakampiai yra supakuoti vertikaliai. Taigi formulė „1 5 V“ parodo, kad stačiakampis Nr.5 yra virš stačiakampio Nr.1 ir meta stačiakampio išmatavimai yra nustatomi taip:

$$L_v = \max(L_1, L_5) \text{ , } W_v = W_1 + W_5 \text{ .} \quad (17)$$

Panagrinėkime šių stačiakampių supakavimą, kurį galima užrašyti tokia formule:

1 5 H 7 6 V 2 H 4 H V 9 8 V 3 V H.

Ši formulė rodo ruošinių supakavimą plokštėje, ir kaip matyti iš 1.9 paveikslo, rodyklės rodo giljotininio pjūvio vietas. Iš 1.9 paveikslo taip pat galima nesunkiai suprasti, kad nubraižytas medis rodo, kaip bus atliekamas giljotininis pjūvis. Formuojant medį, einantį žemyn nuo medžio šaknų iki lapų, galima gauti giljotininių pjūvių eilę, išpjaunant reikiamas stačiakampio dalis iš plokštės.

Išsiaiškinsime keletą šios lygties charakteristikų. Pirma, čia yra keletas apribojimų, kuriuos operatoriai naudoja plokštės pakavimui ir šių operatorių pozicija formulėse. Tegu stačiakampio dalių skaičius būna N_p, ir atitinkamai operatorių skaičius būna N_o, tada dvinarė lygtis yra:

$$N_o = N_p - 1 \text{ .} \quad (18)$$

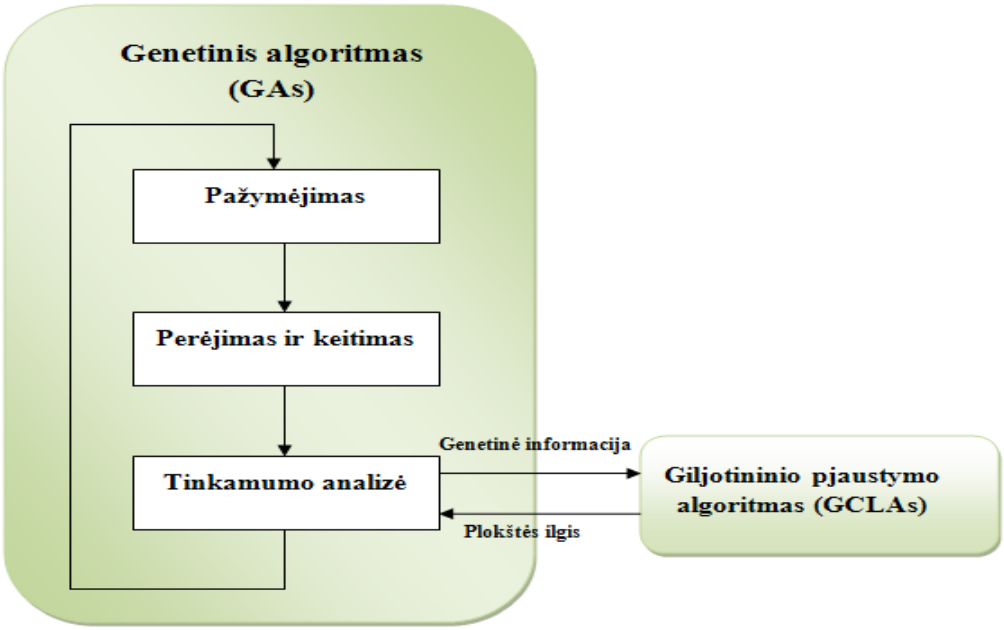
Antra, rašant formulę geriausia, kad nebūtų jokių skliaustelių, tada nekils jokių neaiškumų, ir formulės ilgio rezultatas bus aiškus, taigi N_g ilgis bus laikomas konstanta.

$$N_g = N_p + N_o = 2N_p - 1. \quad (19)$$

Kitas svarstomas klausimas yra apribotas stačiakampio dalių ir operacijų skaičius. Palyginkime bet kurį vieną operatorių; tegul kairėje šio operatoriaus pusėje dalies numeris bus n_p ir operacijos numeris bus n_o, iš to seka ryšys [3]:

$$1 \le n_o \le n_p - 1. \quad (20)$$

Sistema sudaryta iš dviejų dalių, genetinio algoritmo ir giljotininio pjaustymo algoritmo (2.10 pav.). Giljotininio pakavimo planavimo algoritmas pakuoja plokštėje stačiakampius, taip kaip jam nurodo genetinis algoritmas (GAs), apskaičiuoja reikalingą ilgį plokštėje ir jį gražina GAs. Genetinis algoritmas (GAs) modifikuoja stačiakampių pakavimą plokštėje taip, kad reikiamos plokštės ilgis būtų minimalus [14].



10 pav. Sistemos konfigūracija (14)

Šis metodas gali būti aprašomas panaudojant steko mašinos idėją.

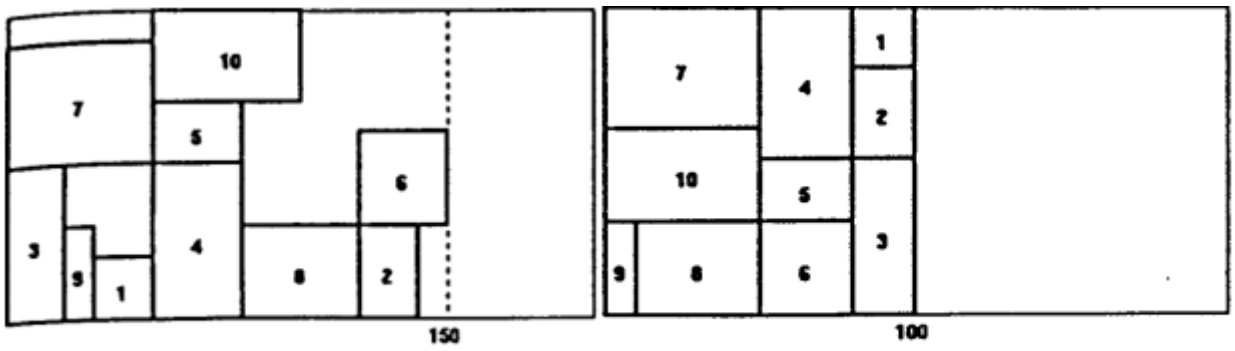
1. Jeigu tai yra dalies numeris, tai jis įrašomas į sąrašą (steką).
2. Jeigu jis yra bet kuri H ar V operacija, tada iš keleto dalių imamos dvi dalys iš kurių gaunamas bendras stačiakampis ir šio naujai registruoto stačiakampio dalies numeris taip pat įrašomas į steką.
3. Kada lygties elementai interpretuoti, visi likučiai sudedami į vieną dalį ir dedami į galą, kai iš stačiakampių susidaro sujungtas stačiakampis, pakavimo operacija baigėsi. Horizontalus šio stačiakampio ilgis ir yra reikalingas plokštės ilgis.
4. Grįžtame prie 1 žingsnio.

Vykdant šias procedūras, jungiamo stačiakampio vertikalus ilgis gali būti didesnis negu plokštės ilgis. Kad tai neįvyktų, reikia ištaisyti įdiegtą algoritmą. Kai tai atsitinka, dabartinis lygties operatorius pakeičiamas į kitą tipą, tai reiškia kad V operatorius pakeičiamas į H operatorių ir šis gražinamas į genetinį algoritmą(GAs). Jeigu nepastebėta pasikeitimų, kada procedūrą vykdo H operatorius, paprasčiausiai tuo metu jokių sutrikimų nebuvo.

Genetinis algoritmas (GAs) vartojamas išreikšti ir sujungti stačiakampius ir operatorius, genetinis operatorius, skiriasi nuo paprasto operatoriaus. Stačiakampių dalių numeriai nagrinėjami kitokia tvarka, kol operatoriai nagrinėja kuris iš dviejų atliks veiksmą: H ar V operatorius, šis operatorius sprendžia, ar pasirinkti sumaišytas genetines operacijas. Genetinis algoritmas suskirstytas į du veiksmus, tai: stačiakampio dalių veiksmas ir operacijų veiksmas, naudojant kaukę. Po privalomos genetinės operacijos kiekvienas veiksmas yra atliekamas taip: du veiksmai sujungiami į vieną veiksmą, ir vėl iš naujo pradeda vykti genetinės operacijos tol, kol visos operacijos sudėlioja stačiakampių dalis.

Crossover (perėjimas). Stačiakampio dalių grupės paverčiamos į kito tipo genetinį algoritmą (GAs) kol pritaikomos.

Optimalaus uždavinio sprendimo suradimas yra sumodeliuotas ir parodytas 11 paveiksle:



a) generacijos pradžioje; b) po 36 generacijų.

11 pav. Rezultatų pavaizdavimas (15)

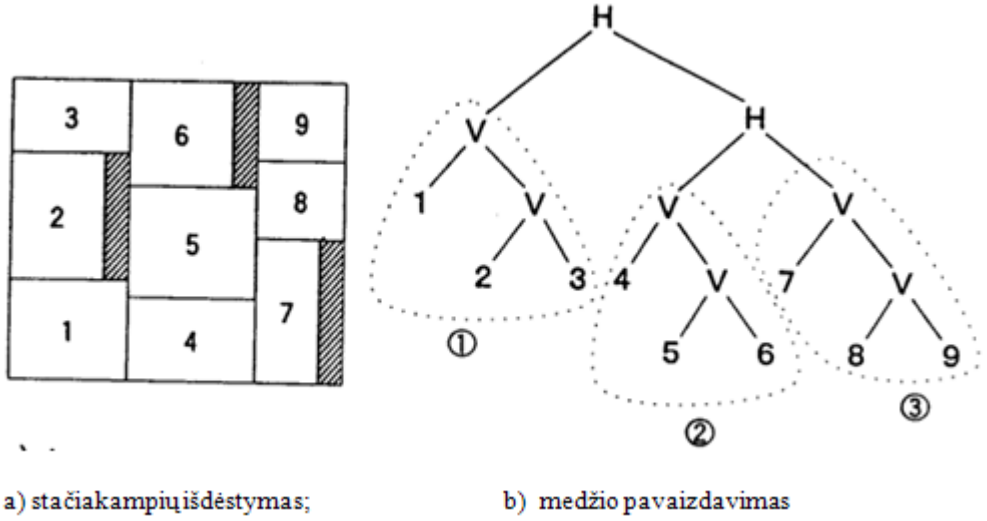
Mutation (keitimasis). Keitimasis prasideda kai susidaro ir apsikeičia du elementai, įskaitant abu operatorius ir staičakampio dalis kai jos apsikeičia skirtingo tipo operatoriais. Pasirinkime du atsitiktinius elementus ir pavadinkime juos atitinkamai p_1 ir p_2, kur p_1 randasi p_2 kairoje pusėje. Pagal tai kiek yra elementų rūšių, atitinkamai išrenkamos apsikeitimo tipo sekos. Kai abi p_1 ir p_2 yra stačiakampio dalyse, tai apsikeitimas tarp jų yra leidžiamas. Jeigu p_1 yra stačiakampio dalis ir p_2 operacija, apsikeitimas pasiseks tik tada, kai operacijos reikalavimo padėtis atitiks veiksmą. Po operacijos p_2 yra apsikeitusi su stačiakampio dalimi p_1, tai nustatyti nelygybės apribojimai turi pasilikti, sekantis ryšys turi būti padengtas visoms operacijoms esamoms tarp p_1 ir p_2 .

$$n_0 \leq n_p - 3, \qquad (21)$$

Kitame pavyzdyje, kur p_1 yra operacija, nekyla jokių klausimų darant apsikeitimą nepaisant p_2 rūšies.

Kitokia procedūra, kai keičiasi H ir V operatoriai, tai reiškia, kad išsijungimas tarp šių operatorių yra galimas [15].

Giljotininio pjūvio ištęsimo stadija. Giljotininiame pjūvyje, kiekvieną kartą pjūvio kryptis keičiasi, pavyzdžiui iš horizontalaus į vertikalų ir atvirkščiai, pjūvis gali būti pasuktas 90 laipsnių. Tai yra papildomas darbas, norint sumažinti pasisukimų skaičių kiek galima mažiau. Mažiausias apsisukimų skaičius yra 2, kiekvienam horizontaliam ir vertikaliam pjūviui (12 pav.).



12 pav. Stačiakampių išdėstymas giljotininiam pjūviui (15)

12 paveiksle pavaizduotas stačiakampių išdėstymas išreikštas tokia formule:

1 2 3 V V 4 5 6 V V 7 8 9 V V H H.

Viena iš lygties padėčių išsidėstymo, po kurio galimi du giljotininiai pjūviai, kai visi H operatoriai nustatyti iš dešnės pusės visiems V operatoriams. Pridėjus neribotą genetinę operaciją, antra stadija būtų realizuoti giljotininį pjūvį.

Taigi šio algoritmo esmė yra ta, kad GAs ir GCLAs dirba paeiliui bendraudami tarpusavyje. GAs nustato šablonų eilę, kuri turi būti išdėliota plokštėje ir siunčia ją GCLAs. GCLAs fiksuoja kiekvieno šablono padėtį taip, kad šie šablonai tarpusavyje nepersidengtų bei nesudarytų laisvų plotų. Kuomet visų šablonų padėtis tampa fiksuota GCLAs siunčia reikalingą plokštės ilgį GAs. tam, kad pastarasis atitinkamai išdėliotų šablonus. GAs turėdamas gautą ilgį nustato šablonų eiliškumo išdėstymą taip, kad reikiamos plokštės ilgis būtų minimalus [14].

2.6 Pjaustymo programų analizė

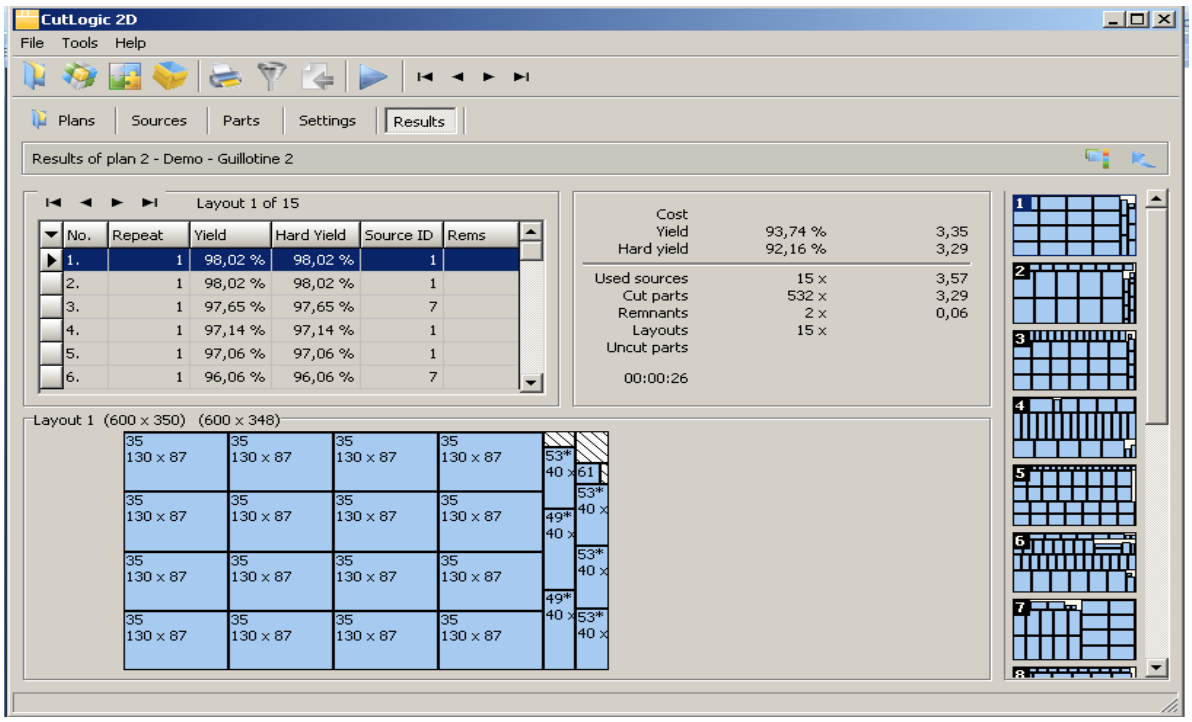
Rinkoje esančios stačiakampių pakavimo plokštėje programiniai paketai skiriasi tiek savo sudėtingumu, tiek skaičiavimo laiku, tiek rezultatų tikslumu. Apžvelgsime keletą komercinių pjaustymo programinių paketų, kurių buvo galimybė parsisiųsti bandomąją versiją.

Optimizuoto pjaustymo programa „CutLogic 2D“.

Ši programa skirta stačiakampių pjovimui, tinkanti įvairioms pramonė šakoms, pavyzdžiui: medienos, baldų, metalo lakštų pjovimui ir t.t. Pagrindinės programos funkcijos ir privalumai:

- Giljotininis (nuo krašto iki krašto) pjūvis, negiljotininis pjūvis, kirpimas.
- Keletas dviejų dimensijų ir trijų dimensijų pjaustymo tipų.
- Galimybė pasirinkti pirmo pjūvio horizontalią arba vertikalią orientaciją.
- Pilnas medžiagų ir likučių atsargų valdymas.
- Grupės optimizavimas – randamas optimalus pjaustymo planas nustatytiems pogrupiams.
- Likučių stebėjimo galimybė.
- Platus pjovimo parametrų pasirinkimas – pjovimo peilio pločio pasirinkimas, apibrėžti likučiai, minimalus stačiakmpių išdėstymo pasikartojimas ir t.t.
- Įvairiapusiškumas ir lankstumas bet kokiem pjaustymo atvejams, palaikymas iki 1 milijono šaltinių ir iki 1 milijono dalių vienam pjaustymo planui.
- Platus iš anksto nustatytų standartinių ataskaitų ir šablonų pasirinkimas.
- Ataskaitų spausdinimo galimybė [1].

„CutLogic 2D” programos langas pateikiamas 13 paveiksle.



13 pav. „CutLogic 2D" programos langas (1)

Pagrindiniai šios programos trūkumai: ilgas skaičiavimo laikas ir sudėtingas naudojimas. Norint dirbti su šia programa, reikėtų papildomai ją paanalizuoti. Taip pat prie trūkumų galima priskirti ir gana didelę programos kainą – standartinė versija kainuoja 499 JAV doleriai (apytiksliai 1322 litai), o profesionali versija nuo 999 JAV dolerių (apytiksliai 2647 litai).

Plokščių (skersinių) pjaustymo optimizavimo programa „Juva1d2d".

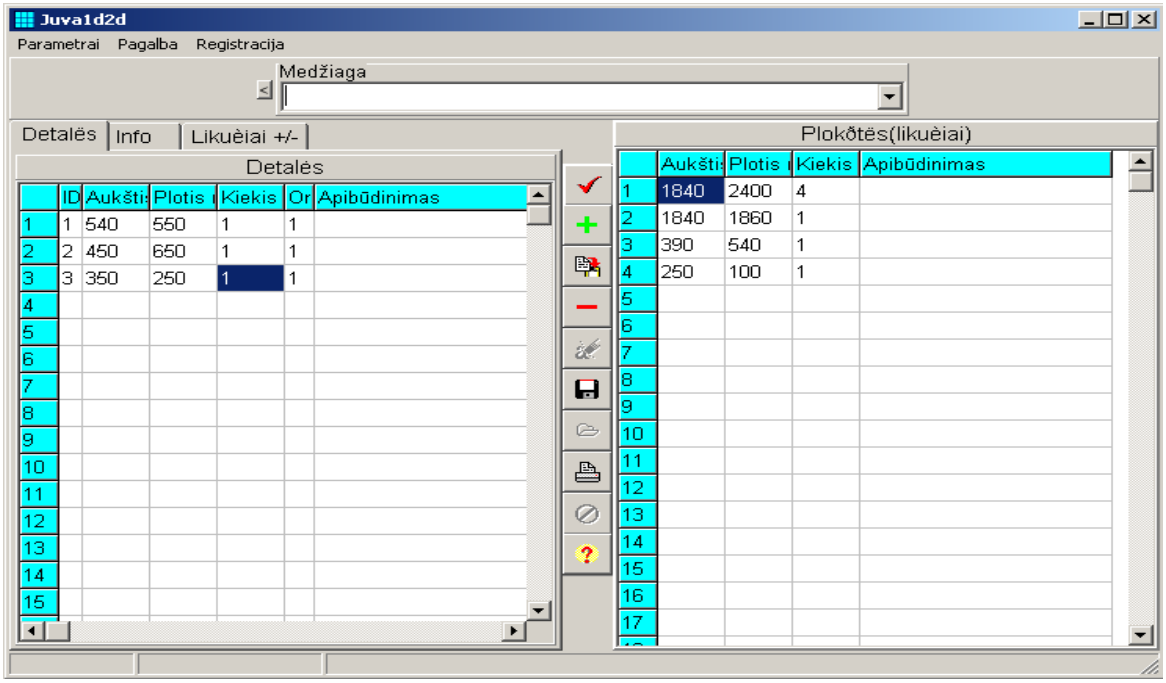
Ši programa pritaikyta baldinei plokštei, stiklui ir metalui pjauti. Optimaliai išdėlioja stačiakampių detales pagal pasirinktus parametrus.

Pagrindiniai programos ypatumai ir privalumai:

- Galimybė pasirinkti 1D ir 2D pjovimo būdą.
- Detalios ataskaitos.
- Likučių apskaita.
- Kraštų, detalių, atraižų, atliekų spalvos ir rašto nustatymas.
- Duomenų eksportas į txt, xls formato bylas.
- Galimybė atspausdinti išpjaustymo ir likučių ataskaitas.
- Nesudėtinga vartotojo sąsaja.

- Palyginti nedidelė produkto kaina – standartinio paketo kaina 350 litų, pasirinkus papildomas funcijas, maksimali kaina 600 litų [10].

„Juva1d2d" programos langas pateikiamas 14 paveikle.



14 pav. „Juva1d2d" programos langas (10)

Taip pat prie programos privalumų būtų galima priskirti ir tai, kad programinis paketas yra lietuvių kalba.

Pagrindinis trūkumas būtų - gana sudėtinga sistema.

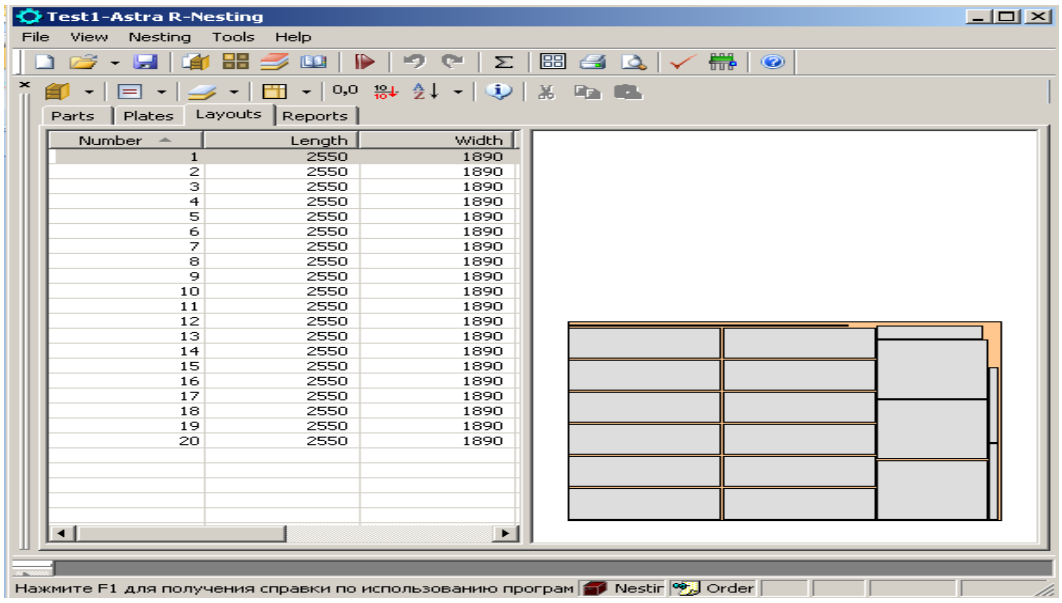
<u>Stačiakampių pjaustymo programa „Astra R – Nesting".</u>

Tai lakštų pjaustymo programa, skirta drožlių plokščių, metalo, stiklo ir plastiko optimizuotam pjaustymui. Programa leidžia vartotojui lengvai ir paprastai sukurti geriausios kokybės pjovimo maketus. Pagrindinės programos galimybės ir privalumai:

- Paprastas lentelinis duomenų įvedimas, taip pat galimybė duomenis nukopijuoti iš MS Word, Excel ir kitų programų.
- Geras pjaustymo algoritmas, leidžiantis minimizuoti atliekų kiekį.
- Galimybė peržiūrėti ir spausdinti įvairias ataskaitas, tokias kaip pjovimo maketai arba specifikacija ir t.t.
- Automatinis atraižų skaičiavimas, skaičiavimas nustatytoje zonoje, rankinis redagavimas, naudojant pelę.

- Rankinis ir automatinis patekimas į atraižų sandėlio duomenų bazę, atraižų sąrašo redagavimas, visų išteklių ir atraižų filtravimas, rūšiavimas, statistinnių duomenų pateikimas.
- Didelis vartotojo sąsajos kalbų pasirinkimas (anglų, rusų, vokiečių, bulgarų, turkų) [9].

„Astra R – Nesting“ programos langas pavaizduotas 15 paveiksle.



15 pav. „Astra R – Nesting“ programos langas (9)

Šios programos trūkumas taip pat būtų sudėtinga sistema ir įsisavinimas. Taip pat prie trūkumų galime priskirti ir gana didelę programos kainą – nuo 349 JAV dolerių (apytiksliai 925 litų) už standartinę versiją iki 1299 JAV dolerių (apytiksliai 3442 litų).

Visų apžvelgtų programų apibendrinimas pateiktas 1 lentelėje.

1 lentelė. Komercinių pjaustymo programų pagrininių savybių apžvalga

	CutLogic 2D	**Juva1d2d**	**Astra R – Nesting**
Privalumai:			
Pakankamai optimalus sprendinys	✓	✓	✓
Lengvas naudojimas	-	✓	✓
Greitas skaičiavimo laikas	-	✓	✓
Lietuvių kalba	-	✓	-
Ataskaitų spausdinimo galimybė	✓	✓	✓
Trūkumai:			
Ilgas skaičiavimo laikas	✓	✓	✓

	CutLogic 2D	Juva1d2d	Astra R – Nesting
Sudėtinga sistema	✓	✓	✓
Sudėtinga vartotojo sąsaja	✓	-	✓
Didelė produkto kaina	✓	-	✓

(Šaltinis: sudaryta autoriaus remiantis 1.7 skyriaus duomenimis)

Matome, kad iš nagrinėtų pjaustymo programų geriausia šiuo atveju būtų Juva1d2d". Tačiau ši programa, kai ir likusios dvi turi gana sudėtingą sistemą. Vartotojas norėdamas naudotis šiomis aprašytomis programomis, turėtų papildomai pastudijuoti jų naudojimą.

Todėl šiame darbe tikslinga sukurti programinį paketą, kuris skaičiavimus pateiktų greitai, būtų paprastas, nesunkiai įsisavinamas ir nebrangus, nors ir neturės tokių didelių parametrų pasirinkimo galimybių. Kuriama programa vartotojas galės naudotis be papildomo apmokymo, o tai aktualu tarkim mažai įmonei, kuri neturi galimybių ir lėšų papildomai apmokinti darbuotojus ar įsigyti brangią programinę įrangą.

2.7 Analitinės dalies išvados

Apibendrinant analitinę darbo dalį, pateikiamos išvados:

- Pjaustymo ir pakavimo uždavinys priskiriamas kombinatorinio optimizavimo užduočių klasei. Šiems uždaviniams yra būdinga tai, kad didėjant jų apimčiai, visais atvejais tksliai išspręsti juos per trumpą sprendimo laiko tarpą pasidaro nebeįmanoma.
- Sprendžiant pjaustymo uždavinius yra naudojama eilė įvairių algoritmų, metodikų ar jų modifikacijų. Tai susiję su tuo, kad pasireiškiant vienai ar kitai uždavinio specifikai tenka naudoti būtent tam atvejui skirtą metodiką ar jos modifikaciją. Tai taip pat susiję ir su kai kurių algoritmų trūkumais bei jų specifika.
- Sprendžiant dviejų dimensijų pakavimo uždavinį, dažniausiai yra naudojami euristiniai metodai, kai per trumpą laiką bandoma rasti optimalų arba artimą jam sprendinį.
- Gilmore ir Gomory pasiūlytas euristnis dekompozicijos metodas naudojamas pjaustymo uždaviniui spręsti. Šis metodas gali būti išvestas iš darbe aprašytos dvimačio giljotininio pjaustymo uždoties formuluotės.

- Daugelis mokslininkų giljotininio pjaustymo problemai spręsti naudoja genetinius algoritmus. Darbe apžvelgtas Toshihiko Ono pateiktas sprendimo variantas, kuriame yra naudojami genetinis algoritmas (GAs) ir giljotininio nustatymo algoritmas (GCLAs).
- Aprašytų algoritmų privalumas yra mažas atliekų procentas, pagrindinis trūkumas – didelis skaičiavimo laikas.
- Sukurta nemažai programinės įrangos, naudojamos pjaustymo uždaviniui spręsti, tačiau norint jomis naudotis, reikia papildomų įgūdžių. Todėl tikslinga kurti greitai įsisavinamą stačiakampių pakavimo plokštėje sistemą tam, kad vartotojai greitai perprastų ir pritaikytų jos funkcijas savo darbuose.

3. PROJEKTINĖ DALIS

3.1 Reikalavimų projektuojamai sistemai specifikacija

Šios dalies tikslas aiškiai apibrėžti kuriamam programinės įrangos produktui keliamus reikalavimus. Detali reikalavimų projektuojamam produktui specifikacija padeda išvengti nesusipratimų su užsakovu, o taip pat garantuoja, kad kūrimo proceso eigoje keliami reikalavimai bus suprasti teisingai ir kuriamas produktas maksimaliai atitiks kliento pageidavimus.

3.1.1 Bendri reikalavimai

Sistema turi būti tokia, kad kiekvienas vartotojas be specialaus pasiruošimo greitai perprastų sistemą ir sugebėtų greitai bei patogiai gauti visą jam reikalingą informaciją, atlikti visas būtinas funkcijas.

- Kuriama programinė įranga turėtų būti nesunkiai įsisavinama, nebrangi ir pateikianti optimalų arba jam artimą rezultatą per trumpą laiką.
- Vartotojų tikslai ir poreikiai: produkto galimybė apskaičiuoti ir išvesti į ekraną optimaliausią arba artimą jam stačiakampių išsidėstymą plokštėje giljotininiam pjaustymui.
- Bendri apribojimai: produktas turi veikti greitai, turi būti nesudėtingas naudojimas.

3.1.2 Funciniai reikalavimai

Programinė įranga turi atlikti užklausos priėmimo, užklausos atlikimo uždavinį.

Funciniai produkto reikalavimai:

- Užklausos priėmimo uždavinys. Priimti užklausą skaičiavimams atlikti. Vartotojas gali įvesti duomenis pats arba pasirinkti duomenis iš failo.
- Pagal duotus stačiakampių ir plokštės išmatavimus (aukštį ir plotį) pateikti optimalų stačiakampių pakavimo plokštėje sprendimo būdą.
- Pateikti vartotojui ir išvesti į ekraną stačiakampių pakavimo plokštėje sprendinį. Pateikti rezultatai turi būti tikslūs.

- Duomenų įvedimo ir išvedimo procedūros turi būti aiškios ir paprastai naudojamos.

Programinė įranga yra skirta jos vartotojui, kuris tiesiogiai sąveikauja su ja. Vartotojas įveda stačiakampių ir plokštės matmenis arba jie yra nuskaitomi iš vartotojo apibrėžto failo.

Funkciniai vartotojo reikalavimai:

- Programa turi leisti pasirinkti bet kokius plokštės matmenis.
- Programa turi leisti pasirinkti skirtingus stačiakampių matmenis.
- Vartotojas turi galėti pats, nepriklausomai nuo programinės įrangos, sukurti įvesties failus ir juos koreguoti (keisti stačiakampių matmenis).
- Programinė įranga turi greitai pateikti rezultatus, išvesdama juos į ekraną grafiniu pavidalu.

3.1.3 Nefunkciniai reikalavimai

- Sukurta programinė įranga turi realiai funkcionuoti ir išvesti teisingus rezultatus.
- Programinė įranga turi veikti taip kaip tikisi vartotojas ir atitikti reikalavimų dokumentą.
- IBM PC tipo personalinis kompiuteris.
- Operacinė sisema: Microsoft® Windows® 7/2000/XP/2003 Server.

3.1.4 Reikalavimai vartotojo sąsajai

- Grafinė vartotojo sąsaja turi būti paprasta ir nesunkiai įsisavinama. Laukai skirti duomenims įvesti išdėstyti aiškiai ir patogiai.
- Klaidų pranešimų aprašymas – jeigu funkcija vykdoma neteisingai, turi įsiterpti aiškus pranešimas apie tai, jog funkcija atliekama neteisingai.
- Sugeneravus rezultatą, vartotojo grafinėje sąsajoje turi būti parodomas aiškus supakuotų stačiakampių vaizdas.

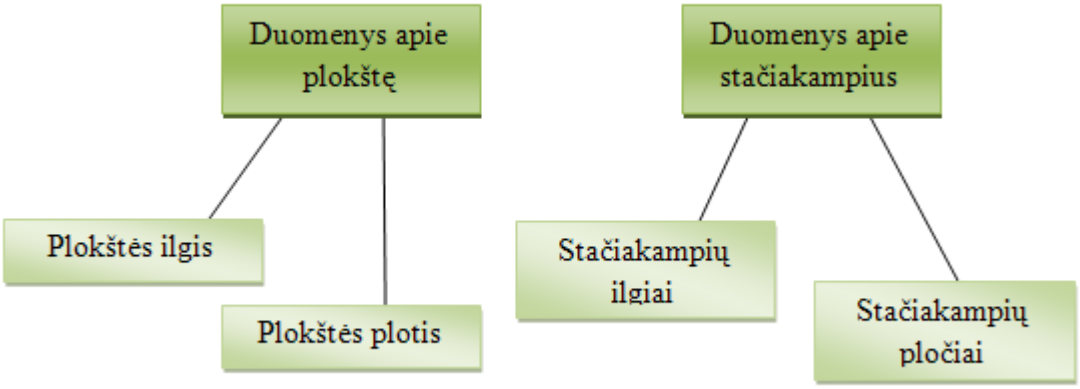
3.1.5 Projekto apribojimai

- Prie sistemos gali jungtis tik vienas vartotojas. Jei užsakovas nori, kad programa galėtų naudotis du ir daugiau darbuotojų, sistemą reikia įdiegti kiekvieno darbuotojo, pageidaujančio dirbti su šia sistema, kompiuteryje.
- Vartotojo kompiuteryje turi būti įdiegta „.NET Framework ” platforma.
- Sistemą galima instaliuoti Windows XP, Windows Server 2003, Windows 2000 ir Windows Vista operacinėse sistemose. Jei kompiuteryje įdiegta Windows Server 2003 arba Windows Vista operacinė sistema, “.NET Framework” platformos įdiegti nereikia, kadangi šios operacinės sistemos turi integruotą “.NET Framework” komponentą. Naudojant kitą operacinę sistemą, reikia šią platformą įdiegti į kompiuterį.
- Apribojimai techninei įrangai: privalomi visi reikalingi techniniai įrenginiai išskyrus kolonėles.

3.2 Stačiakampių pakavimo sistemos modeliai

3.2.1 Duomenų struktūrų modeliai (DSD)

Duomenų struktūrų modeliai realizuoja struktūros supratimą. DSD sudaro galimybę išskaidyti sistemą į priimtino dydžio nepriklausomus vienetus taip, kad jie, o kartu ir sistema, būtų lengviau ir geriau suprantami. Be to, informacija pateikiama grafiškai ir glaustai. Grafinis pateikimo būdas reiškia, kad informacija gali būti naudojama dvejopai - kaip statinė dokumentacijos dalis, ir kaip visų lygių bendravimo priemonė: tarp analitikų ir vartotojų, tarp analitikų ir kūrėjų, tarp analitikų ir analitikų. Tai, kad DSD suprantama vartotojams, leidžia lengviau gauti teisingą modelį, todėl didėja programinės įrangos efektyvumas.



16 pav. Duomenų struktūrų diagrama (DSD)

Kuriamoje programinėje įrangoje bus naudojami dviejų rūšių duomenys: duomenys apie plokštę iš kurios bus pjaunami stačiakampius ir duomenys apie stačiakampius (16 pav.).

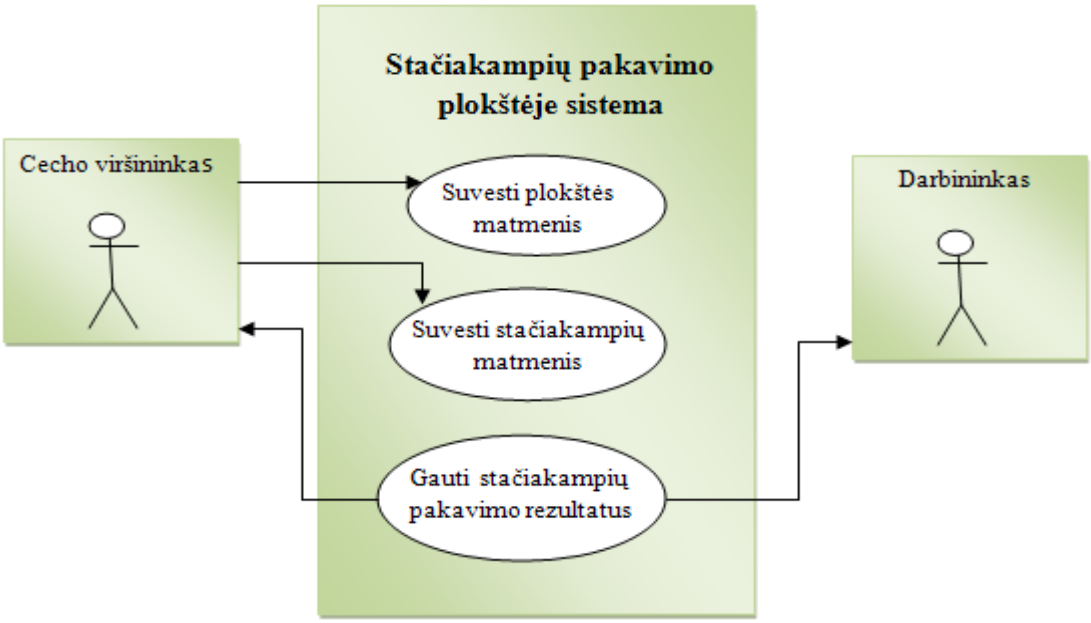
3.2.2 Veiklos uždavinių modelis (UCM notacija)

Daugelis veiklos analizės ir projektavimo metodų vartotojo informaciniams reikalavimams aprašyti naudoja grafinį modelį Use Case Model (UCM), kurio autorius yra Ivar Jacobson. Ivar Jacobson apibūdina Use Case Model taip: “Sistemos vykdomų transakcijų visuma, kurios paskirtis yra pateikti veiklos “dalyviui” (actor) pageidaujamą konkretų rezultatą”.

Transakcijos (sąveikos) apima informacijos srautų ir (arba) materialių srautų perdavimą veiklos procesui (funkcijai, uždaviniui, padaliniui) arba gavimą iš veiklos proceso (funkcijos, uždavinio).

Praktikoje UCM gali būti taikomas dviems tikslams, kurie siejasi tarpusavyje:

a) analizuojamos **veiklos srities modeliui** aprašyti - nurodyti svarbiausias veiklos dalyvių sąveikas (materialias ir informacines) su veiklos uždaviniais (biznio, gamybiniais, informacijos apdorojimo);
b) kompiuterizuojamos **veiklos srities informaciniams poreikiams specifikuoti** – modeliuoti tik informacines sąveikas tarp veiklos dalyvių ir kompiuterizuojamų procesų bei funkcijų (t.y. taikomųjų uždavinių).



17 pav. Veiklos uždavinio modelis (UCM)

Cecho viršininkas suveda duomenis apie plokštę ir stačiakampius, kuriuos reikia išpjauti ir gauna stačiakampių pakavimo rezultatus. Darbininkas iš sistemos gauna stačiakampių pakavimo rezultatus ir pagal juos išpjauna ruošinius (17 pav.).

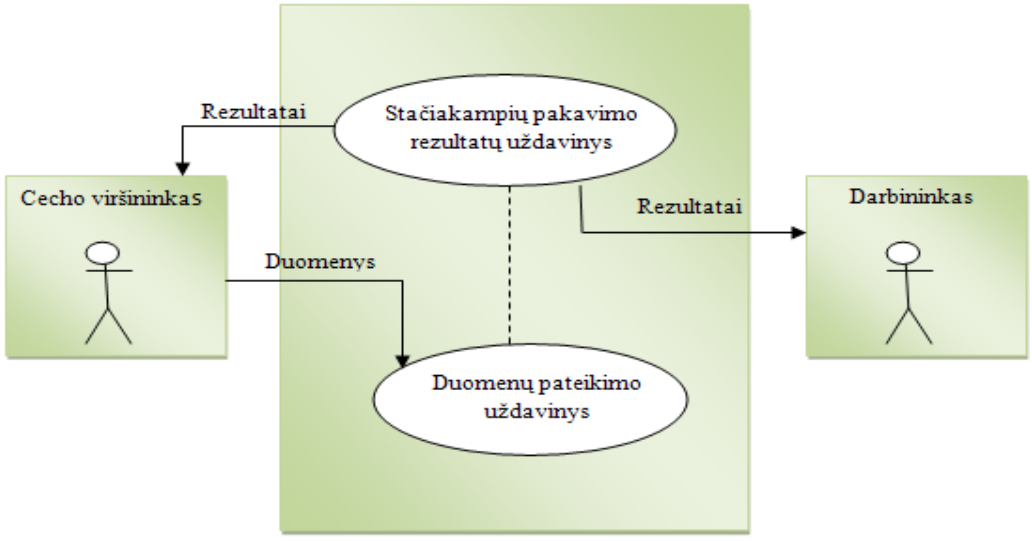
2 lentelė. Veiklos uždavinių detalus aprašymas

1. VEIKLOS UŽDAVINYS: Suvesti plokštės matmenis	
Vartototojas/Aktorius:	Cecho viršininkas.
Aprašas:	Tiesiogiai dirbama su sistema.
Prieš sąlyga	Cecho viršininkas gauna užsakymą baldų gamybai.
Sužadinimo sąlyga:	Suvedami plokštės iš kurios bus pjaunami stačiakampiai baldų gamybai, matmenys – aukštis ir plotis.
Rezultatas:	Įkeliami duomenys apie plokštę į duomenų lauką.
2. VEIKLOS UŽDAVINYS: Suvesti stačiakampių matmenis	
Vartototojas/Aktorius:	Cecho viršininkas.
Aprašas:	Tiesiogiai dirbama su sistema.
Prieš sąlyga	Įkelti duomenys apie plokštę.

Sužadinimo sąlyga:	Suvedami reikalingų išpjauti stačiakampių matmenys – aukštis ir plotis
Rezultatas:	Įkeliami duomenys apie stačiakampių išmatavimus į duomenų lauką.
3. VEIKLOS UŽDAVINYS: Gauti stačiakampių pakavimo rezultatus	
Vartototojas/Aktorius:	Cecho viršininkas, darbininkas
Aprašas:	Apima procesą, kurio metu peržiūrimi pateikti rezultatai.
Prieš sąlyga	Suvesti duomenys.
Sužadinimo sąlyga:	Cecho viršninkas aktyvuoja skaičiavimo funkciją.
Rezultatas:	Ekrane pateikiamas grafinis stačiakampių pakavimo vaizdas. Darbininkas išpjauna ruošinius.

3.2.3 Vartotojo informacinių poreikių modelis (UCM)

Šiuo atveju UCM vadinamas vartotojo (informacinių) poreikių modeliu, nes visi srautai - informaciniai ir identifikuoti (įvardinti).



18 pav. Vartotojo informacinių poreikių modelis (UCM)

3 lentelė. Duomenų srautų aprašymas

Duomenų srautas	Aprašymas
Duomenys	Cecho viršininkas pateikia duomenis apie plokštes ir reikiamus išpjauti stačiakampius. Plokštės ir stačiakampių išmatavimus (ilgį ir plotį).
Rezultatai	Suformuojami rezultatai, kurie parodo optimalų stačiakampių supakavimą plokštėje.

(Šaltinis: sudaryta autoriaus, remiantis 18 paveikslo duomenimis)

3.2.4 Sistemos struktūros modelis

Kuriamos sistemos struktūra galima pavaizduoti tokiu sistemos struktūros modeliu:

19 pav. Sistemos struktūros modelis

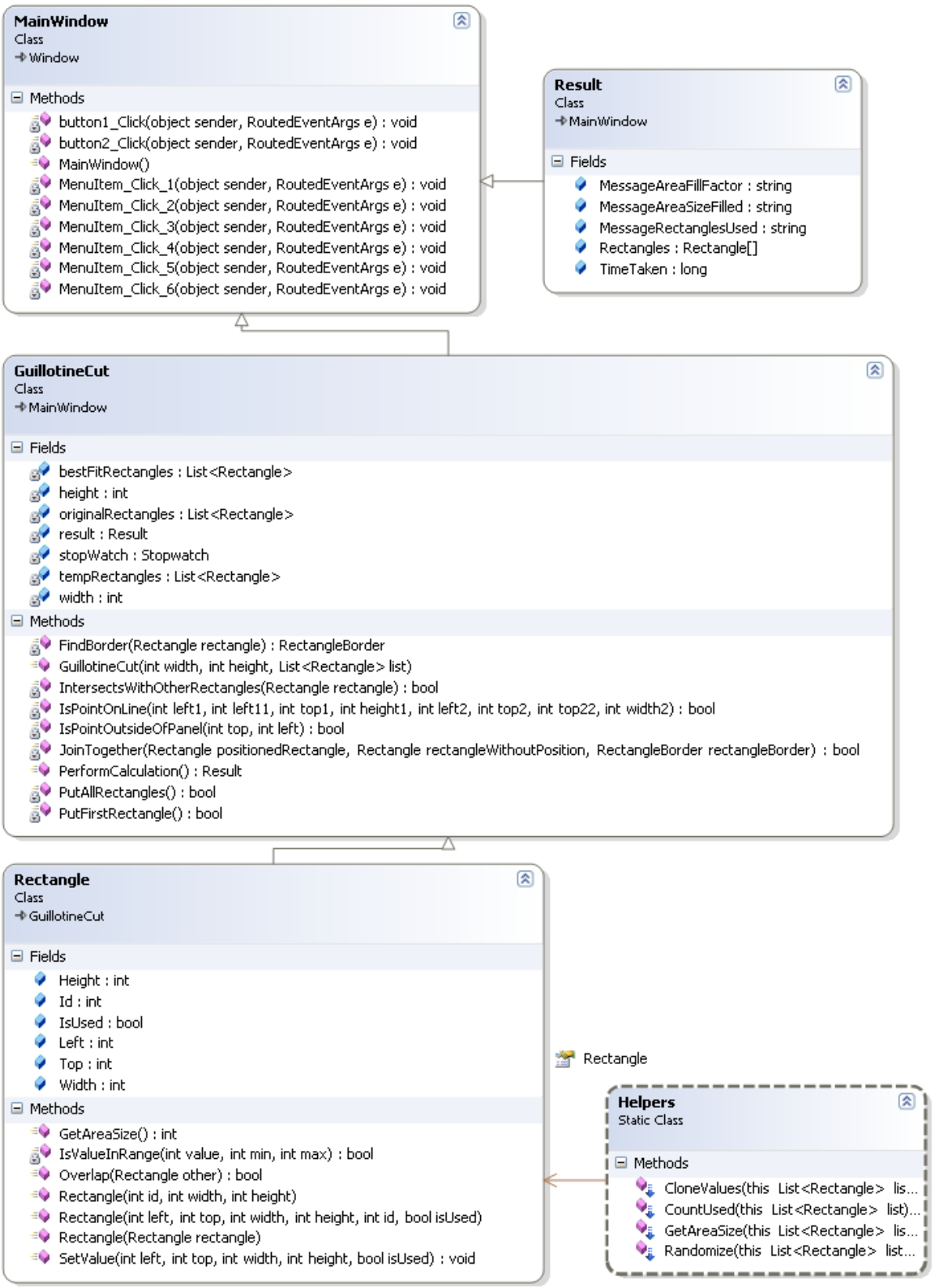
Duomenų įvedimas – tai vartotojo suvedami duomenys į ekraną arba į failą. Duomenų tipas – planuojamų pakuoti stačiakampių ir plokštės, kurioje juos pakuosime, aukštis ir plotis.

Skaičiavimų vykdymas – tai vartotojo įvestų duomenų generavimas ir optimalaus sprendinio suradimas.

Rezultatų grafinis vaizdavimas – tai rezultatų išvedimas į ekraną grafiniame pavidale.

3.3 Programinių modulių specifikacija

Duomenų struktūrą taip pat atvaizduoja klasių diagrama (20 pav.).



20 pav. Klasių diagrama

4 lentelė. Klasių funkcinis aprašymas

Modulio vardas	Funkcinis aprašymas
MainWindow.cs	Pagrindinė klasė, kuri naudojama pilnam sistemos funkcionalumui užtikrinti. Šis modulis iškviečiamas aktyvavus sistemos paleidimo failą. Iškviečiama pagrindinė vartotojo sąsaja. Per modulį valdomi meniu punktai, iškviečiami pagalbiniai dialogai, vykdomi skaičiavimai.
GuillotineCut.cs	Klasė skirta pagrindiniams skaičiavimams (stačiakampio kraštinių radimui, tikrinimui ar stačiakampis telpa į plokštę, stačiakampių pakavimui ir pjovimui).
Rectangle.cs	Klasė yra stačiakampių duomenų klasė. Ji naudojama GuillotineCut klasėje.
Result.cs	Pagalbinė klasė skirta duomenis išvedamiems būsenos juostoje aprašyti.
Helpers.cs	Klasė skirta pradinių duomenų generavimui ir jų sąrašo sudarymui

(Šaltinis: sudaryta autoriaus)

Detalus klasių metodų aprašymas pateikiamas 5 lentelėje.

5 lentelė. Klasių metodai ir jų aprašymai

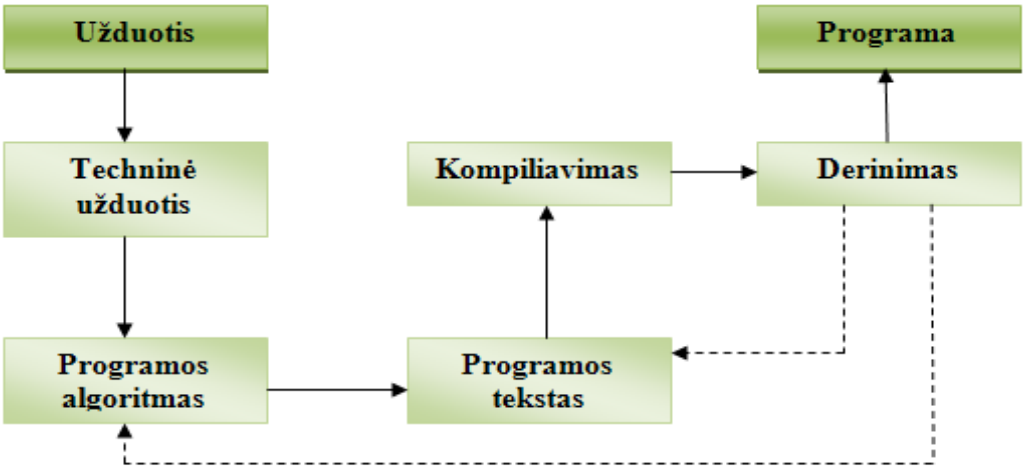
Metodas	Paskirtis
***MainWindow.cs* klasė**	
Button1_Cick(object sender, RoutedEventArgs e)	Valdymo mygtukas skirtas pradėti skaičiavimams
Button2_Cick(object sender, RoutedEventArgs e)	Išvalo duomenų laiką (mygtukas „valyti“)
MeniuItem_Click_1(object sender, RoutedEventArgs e)	Atidaro meniu juostos lauką „failas“
MeniuItem_Click_2(object sender, RoutedEventArgs e)	Meniu juostos mygtukas skirtas atidaryti failui
MeniuItem_Click_3(object sender, RoutedEventArgs e)	Meniu juostos mygtukas, skirtas išsaugoti failą
MeniuItem_Click_4(object sender, RoutedEventArgs e)	Meniu juostos mygtukas, skirtas baigti darbą
MeniuItem_Click_5(object sender, RoutedEventArgs e)	Meniu juostos komanda „Apie“ – programos aprašymas

Metodas	Paskirtis
***GuillotineCut.cs* klasė**	
FindBorder(Rectangle rectangle)	Nustato stačiakampių kraštines
GuillotineCut(int width, int height, List<Rectangle> list)	Išveda supakuotų stačiakampių grafinį vaizdą
IntersectsWithOtherRectangles(Rectangle rectangle)	Tikrina ar stačiakampiai nesikerta
IsPointOnLine(int left1, int left11, int top1, int height1, int left2, int top2, int top22, int width2)	Randa vietą plokštėje, kur bus dedamas stačiakampis
IsPointOutsideOfPanel(int top, int left)	Nustato ar yra vietos sekančiam stačiakampiui
JoinTogether(Rectangle positionedRectangle, Rectangle rectangleWithoutPosition, RectangleBorder rectangleBorder)	Nustato prie kurio taško bus glaudžiamas kitas stačiakampis
PerformCalculation()	Atlieka skaičiavimo funkciją
PutAllRectangles()	Jeigu įmanoma (visi stačiakampiai telpa) stačiakampiai talpinami plokštėje
PutFirstRectangle()	Talpina pirmą stačiakampį
***Rectangle.cs* klasė**	
GetAreaSize()	Nustato plokštės plotą
IsValueInRange(int value, int min, int max)	Nustato ar išmatavimas telpa plokštėje
Rectangle(int left, int top, int width, int height, int id, bool isUsed)	Aprašo kintamuosius
SetValue(int left, int top, int width, int height, bool isUsed)	Priskiria reikšmes kintamiesiems
***Helpers.cs* klasė**	
Randomize(this List<Rectangle> list)	Rikiuoja ruošinių sąrašą
GetAreaSize(this List<Rectangle> list)	Iškviečia plokštės plotą
CountUsed(this List<Rectangle> list)	Grąžina panaudotų ruošinių sąrašą

3.4 Programinės įrangos projektavimas

Programos sukūrimas yra gan ilgas ir sudėtingas procesas, kuris schematiškai pavaizduotas 21 paveiksle. Kaip ir bet kuris darbas jis prasideda nuo užduoties, kurioje turi būti išaiškinta ką reikia padaryti. Išsiaiškinus ką reikia padaryti, smulkiai aprašomi visi įeinantys duomenys ar signalai, valdymas ir išeinantys duomenys ar signalai.

Programinės įrangos projektavomo pradžioje kuriamas algoritmas. Sudarius algoritmą yra sprendžiama kokia programavimo kalba rašyti programą. Pasirinkus programavimo kalbą, pagal algoritmą yra parašomas programos tekstas.

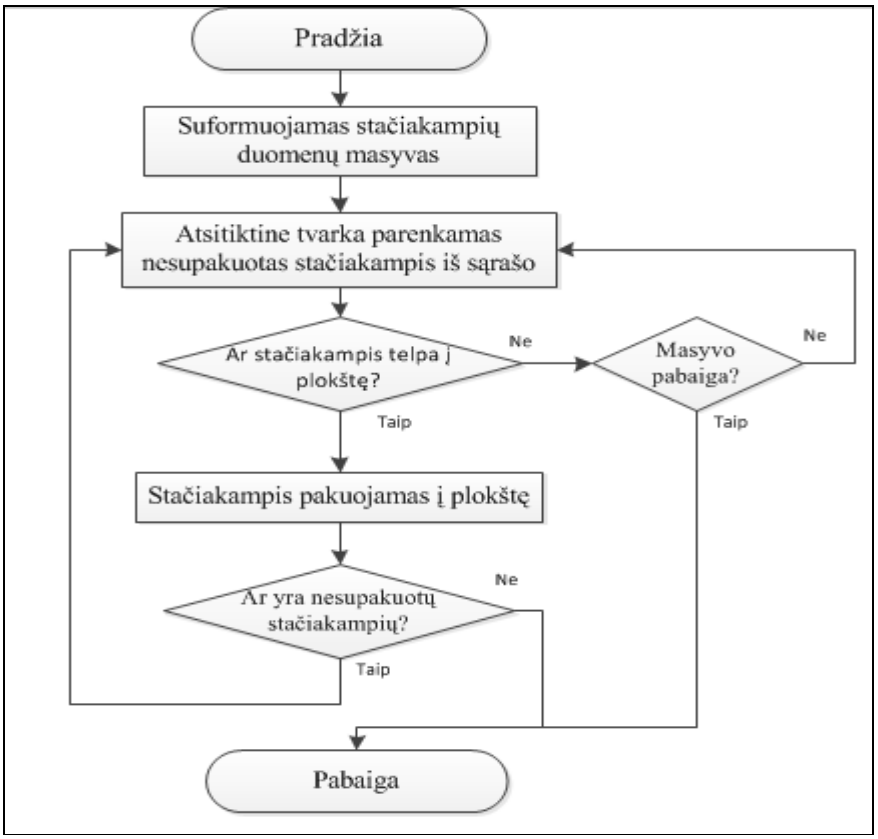


21 pav. Programos kūrimo etapai (17)

Šiame darbe stačiakampių pakavimui plokštėje buvo sukurtas modelis realizuotas CSharp (C#) programavimo kalba.

3.4.1 Uždavinio sprendimo algoritmas

Supaprastintą programinės įrangos algoritmo schemą galima pavaizduoti taip:

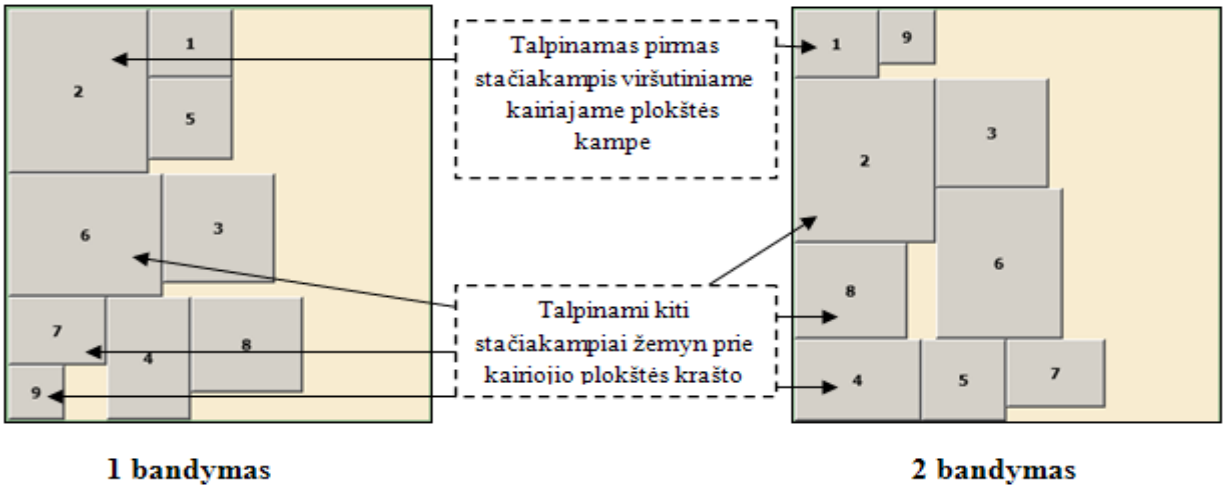


22 pav. Algoritmo scema

Detalus algoritmo aprašymas:

1 žingsnis. Pasibaigus duomenų įvesies procesui, programa nuskaito duomenis ir suformuoja stačiakampių sąrašą.

2 žingsnis. Imamas pirmas stačiakampis ir tikrinama ar jis telpa į plokštę. Jeigu stačiakampis telpa, jis pakuojamas plokštės viršutiniame kairiajame kampe. Stačiakampis pažymimas kaip panaudotas.



23 pav. Pirmo stačiakampio pakavimas

3 žingsnis. Tikrinama ar liko stačiakampių, kuriuos būtų galima supakuoti. Jeigu randamas kitas stačiakampis, telpantis plokštėje, jis talpinamas žemyn nuo pirmojo stačiakampio, prie plokštės kairiojio krašto. Taip stačiakampiai pakuojami žemyn tol kol yra nors vienas telpantis stačiakampis (23 pav.).

4 žingsnis. Kuomet nelieka stačiakampio, kuris tilptų žemiau supakuotų stačiakampių, sekantis stačiakampis talpinamas vėl pradedant nuo viršaus ir pritraukiant jo kairįjį viršutinį kraštą prie pirmoje juostoje esančio aukščiausio stačiakampio dešnio viršutinio kampo. Jeigu stačiakampis tinka šioje vietoje jis talpinamas, priešingu aveju tikrinama ar jis tinka žemiau esančio stačiakampio.

5 žingsnis. Stačiakampiai pakuojami tol kol yra nepanaudotų stačiakampių arba daugiau nei vienas stačiakampis netelpa į plokštę.

6 žingsnis. Ekrane išvedamas grafinis stačiakampių supakavimo plokštėje vaizdas.

7 žingsnis. Įvedami nauji duomenys ir pradedama nuo 1 žingsnio arba sistema baigia darbą.

3.5 Produkto testavimas

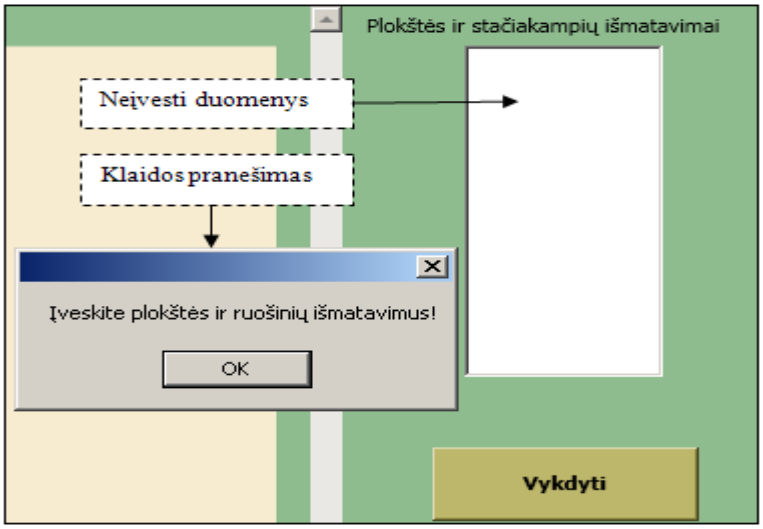
Testavimas – tai programinės įrangos vykdymo (execution) procesas, kurio metu reikia nustatyti, ar jos pateikiami rezultatai yra teisingi.

Programinė įranga vertinama pagal atitikimą reikalavimams ir kokybę, kuri dažnai siejama su testavimo veikla. Testavimas leidžia patikrinti kokybę ir atrasti defektus, o pati kokybė ir kitos programinės įrangos savybės, tokios kaip greitis, našumas, tiesiogiai priklauso nuo projektavimo sprendimų.

Programos našumas vertinamas labai gerai, jeigu vertiname, kad darbo tikslas buvo sukurti nesudėtingą ir paprastą sistemą, ir gerai, lyginant kitų panašios paskirties programomų kotekste.

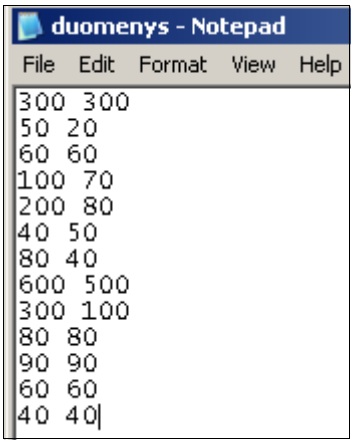
Buvo atliktas nuodugnus testavimas kiekvienai funkcijai atskirai, vėliau viską sujungus į bendrą sistemą, funkcijos buvo testuojamos kaip visuma. Jokių rimtų sutrikimų nebuvo pastebta, neskaitant keleto smulkių netikslumų, kurie buvo pataisyti. Atlikus pirminį vartotojo sąsajos testavimą, sutrikimų nebuvo pastebėta. Testavimo eiga:

1. **Meniu testavimas.**
    - Pasirinkus komandą „Failas → Atidaryti“, atsidaro failo pasirinkimo iš katalogo langas, pasirenkamas failas turi būti *.txt formato. Failo pasirinkimo lange kito formato failai nesimato ir jų pasirinkimas negalimas.
2. **Pagrindinio lango testavimas.**
    - Duomenų įvedimo lauke neįrašius jokių duomenų (arba nepasirinkus iš failo) ir įvykdžius komandą „Vykdyti“, bus išvestas sisteminis klaidos pranešimas (24 pav.)



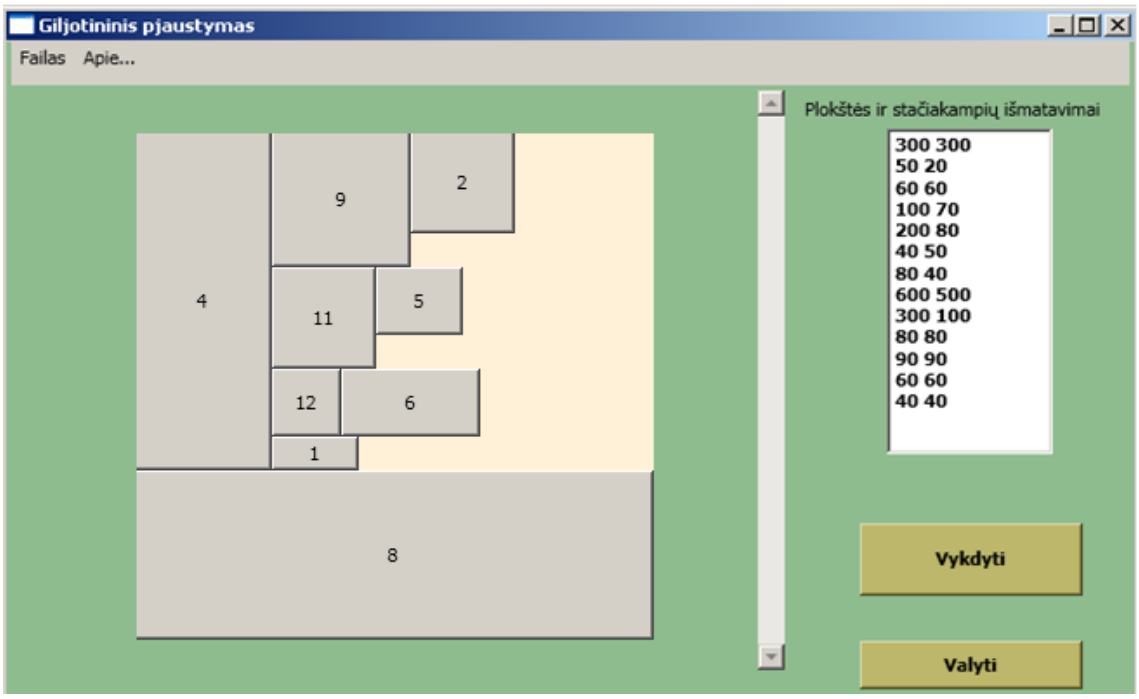
24 pav. Klaidos apie neįvestus duomenis pranešimas

- Sukurta programinė įranga buvo testuojama paruošiant klaidingus įvedimo duomenis. Įvesties faile buvo suvesti stačiakampių išmatavimai, didesni nei plokštės išmatavimai. Tokiu atveju, programa stačiakampių, kurie netilpo, nepakavo. Paruoštas klaidingas duomenų failas pateikiamas 25 pav.



25 pav. Klaidingų duomenų failas

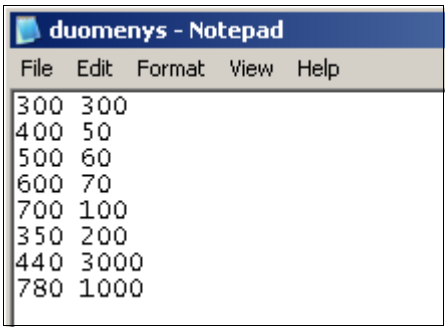
Pagal duomenys.txt failą, bandoma supakuoti daugiau stačiakampių negu telpa į plokštę.



26 pav. Sistemos reakcija į klaidingų duomenų failą

Programa supakavo tik tuos stačiakampius kurie tilpo. Matome, kad nesupakuoti liko 3 stačiakampiai iš 12 – trečias, septintas ir dešimtas.

Paruoštas antras testavimo duomenų rinkinys, kuris sudarytas iš stačiakampių, kurių kiekvieno matmenys viršija plokštės matmenis. Šiuo atveju sistema nepakavo nei vieno stačiakampio.



27 pav. Antras klaidingų duomenų rinkinys

3.6 Projektinės dalies išvados

- Aprašyti reikalavimai projektuojamai sistemai, kurie tiksliai nusako kokia programinė įranga kuriama ir ko iš jos tikimasi.
- Sistemos modeliai apibrėžia kuriamos sistemos funkcijas ir struktūrą.
- Pateiktas detalus kuriamos sistemos algoritmas.
- Testavimo mdžiagoje pateikiama informacija apie galimą vartotojo darbą su sistema. Aprašyta sistemos reakcija.

4. VARTOTOJO DOKUMENTACIJA

4.1 Sistemos funkcinis aprašymas

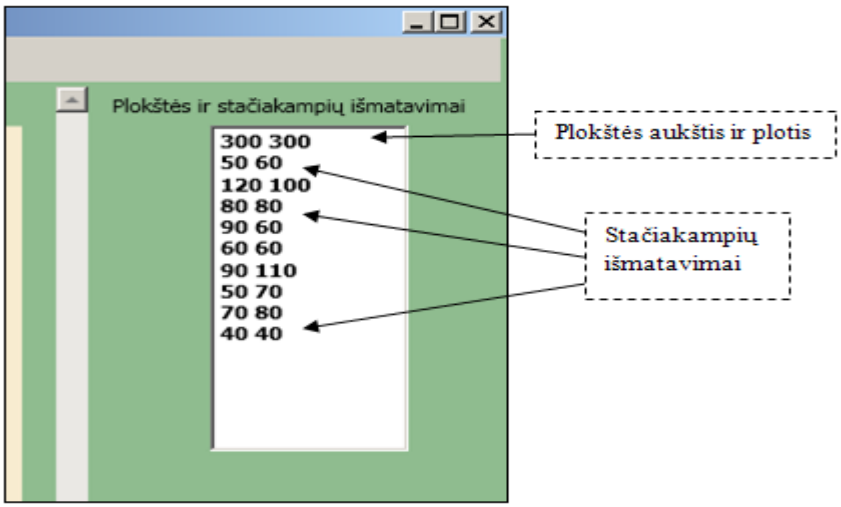
Programinė įranga yra skirta stačiakampių giljotininio pjaustymo uždaviniui realizuoti. Pagal įvestus norimų išpjauti stačiakampių duomenis ir plokštės iš kurios juos pjausime išmatavimus, sistema per trumpą laiką pateikia kiek galima optimalesnį stačiakampių pakavimo plokštėje rezultatą. Rezultatas pateikiamas grafiniame pavidale, ekrane. Programinė įranga kurta nesudėtinga, patogi vartoti ir rezultatą pateikianti per kuo trumpesnį laiką.

Programinis paketas užima nedaug vietos kompiuteryje, jo nereikia instaliuoti, tačiau vartotojo kompiuteryje būtinai turi būti įdiegta „.NET Framework ” platforma, kitu atveju programa neveiks.

4.2 Sistemos vadovas

Programos langas susideda iš srities, kurioje pateikiamas 2D stačiakampių supakavimo grafinis vaizdas, valdymo mygtukų ir meniu juostos.

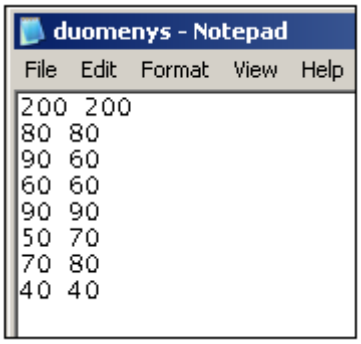
Pradėjus naudotis programa, vartotojas turi įvesti ruošiamų pjaustyti stačiakampių išmatavimus (aukštį ir plotį) ir plokštės išmatavimus. Duomenis vartotojas gali suvesti tiesiai programos pagrindiniame lange arba pasirinkti skaityti duomenis iš anksčiau sukurto failo.



28 pav. Duomenų įvedimo laukas

Pirmoje duomenų įvedimo lauko „Plokštės ir stačiakampių išmatavimai“ eilutėje įvedama plokštės aukštis ir plotis), sekančiose eilutėse vedami norimų pakuoti stačiakampių aukščiai ir pločiai. Duomenys atskiriami tarpo simboliu, kiekvieno stačiakampio išmatavimai vedami naujoje eilutėje (28 pav.).

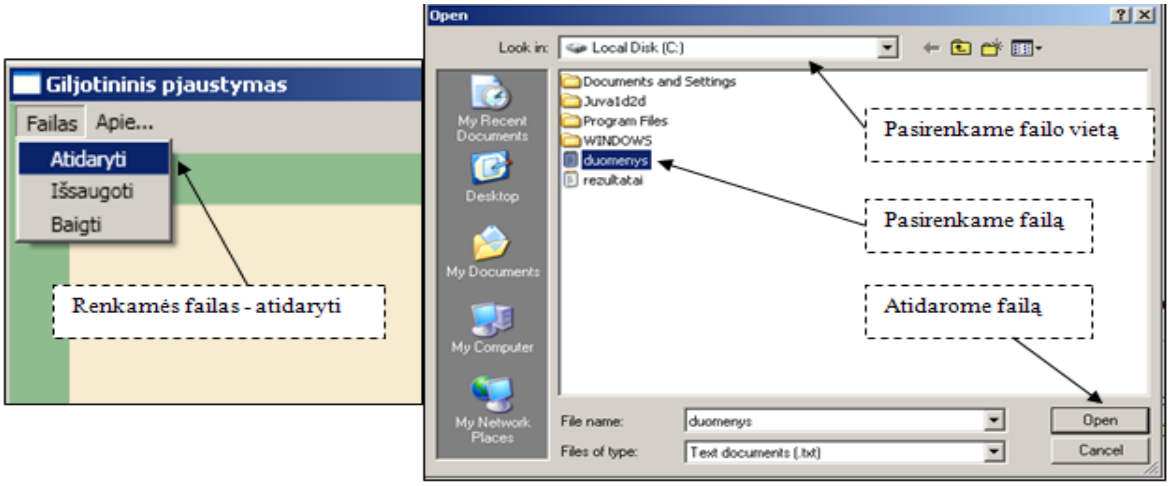
Renkantis skaityti duomenis iš failo, prieš tai turime susikurti duomenų failą. Informaciją vartotojas turi suvesti į *.txt formato failą. Tokio formato failą galima sukurti naudojantis Microsoft Windows ir Microsoft Office standartinio paketo programomis ir daugeliu kitų sukuriančių *.txt formato failus. Šiuo atveju tekstinis failas buvo sukurtas Notepad programa ir pavadintas duomenys.txt.



29 pav. Tekstinis duomenų failas

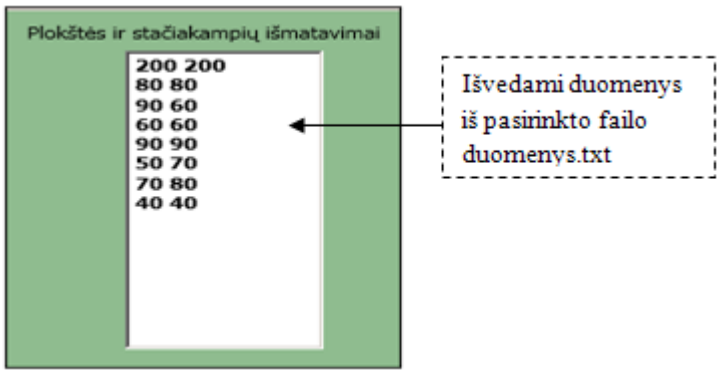
Tekstiniame duomenų faile duomenys suvedami pagal tą pačią taisyklę, kaip ir vedant duomenis tiesiai į programos langą: pirma eilutė – plokštės išmatavimai, sekančios eilutės – stačiakampių išmatavimai, duomenys atskiriami tarpo simboliu.

- įvykdžius šią komadą atsidaro failo pasirinkimo langas, kuriame pasirenkama failo vieta, susirandamas ir pasirenkamas failas (šiuo atveju, duomenys.txt).



30 pav. Duomenų skaitymas iš failo

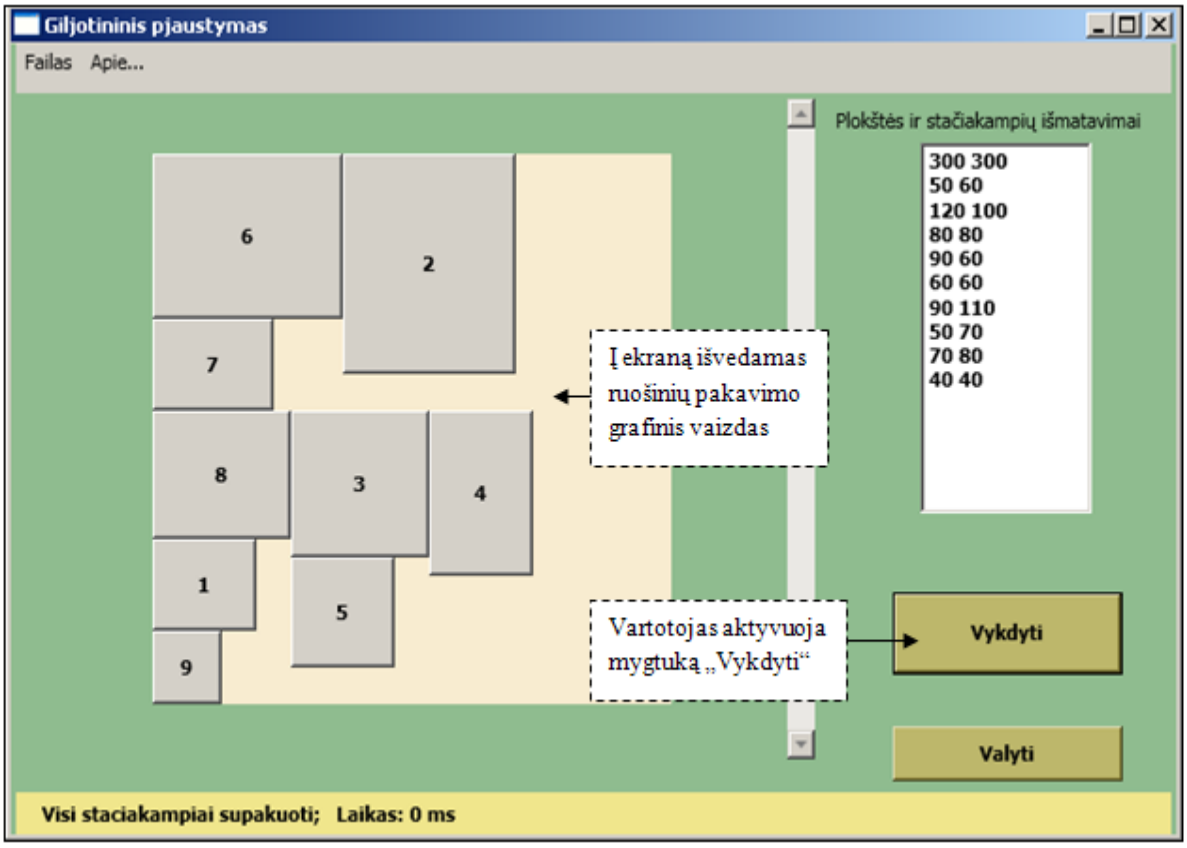
Pasirinkus atidaryti duomenų failą, duomenys išvedami duomenų įvedimo lauke „Plokštės ir stačiakampių išmatavimai“.



31 pav. Nuskaityti duomenys iš failo

- aktyvavus mygtuką programa pateikia grafinį stačiakampių pakavimo plokštėje vaizdą.

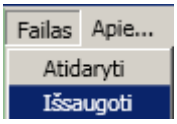
- aktyvavus mygtuką duomenų įvedimo langas „Plokštės ir stačiakampių išmatavimai“ išvalomas. Vartotojas gali suvesti naujus duomenis.



32 pav. Pagrindinis giljotininio pjaustymo programos langas

Programos lango apačioje, būsenos juostoje, pateikiamas paskaičiuotas laikas, kurį programa užtruko dėliodama ruošinius. Laikas skaičiuojamas milisekundėmis (ms – viena tūkstantoji s dalis), kadangi įvedus nedidelį skaičių ruošinių, programa greitai pateikia rezultatą ir skaičiuojant sekundėmis, jis dažniausiai būtų lygus nuliui. Sudėliojus visus stačiakampius plokštėje, būsenos juostoje matome užrašą „Visi stačiakampiai supakuoti“.

Užpildžius visą plokštės plotą, būsenos juostoje matomas pranešimas „Visa plokštė sunaudota“.



- komanda vykdoma norint išsaugoti duomenis. Įvykdžius komandą failas išsaugomas faile rezultati.txt.

- komanda vykdoma norint baigti darbą.

4.3 Sistemos instaliavimo dokumentas

Sistemą galima instaliuoti Windows XP, Windows Server 2003, Windows 2000 ir Windows Vista operacinėse sistemose. Kompiuteryje turi būti įdiegta “.NET Framework” platforma. Programa paleidžiama aktyvavus failą CuttingOptimization.Guillotine.exe.

Minimalūs reikalavimai kompiuteriniai technikai:

Operacinė sistema: Microsoft® Windows® 7/2000/XP/2003 Server.

Procesorius: 1,79 GHz AMD Athlon^{tm}.

Laisva atmintis: 10 MB.

4.4 Sistemos administratoriaus vadovas

Sistema yra nepriklausoma nuo kitų sistemų, todėl jokių pranešimų, kaip sistema bendrauja su kitomis sistemomis, nenumatyta.

4.5 Išvados

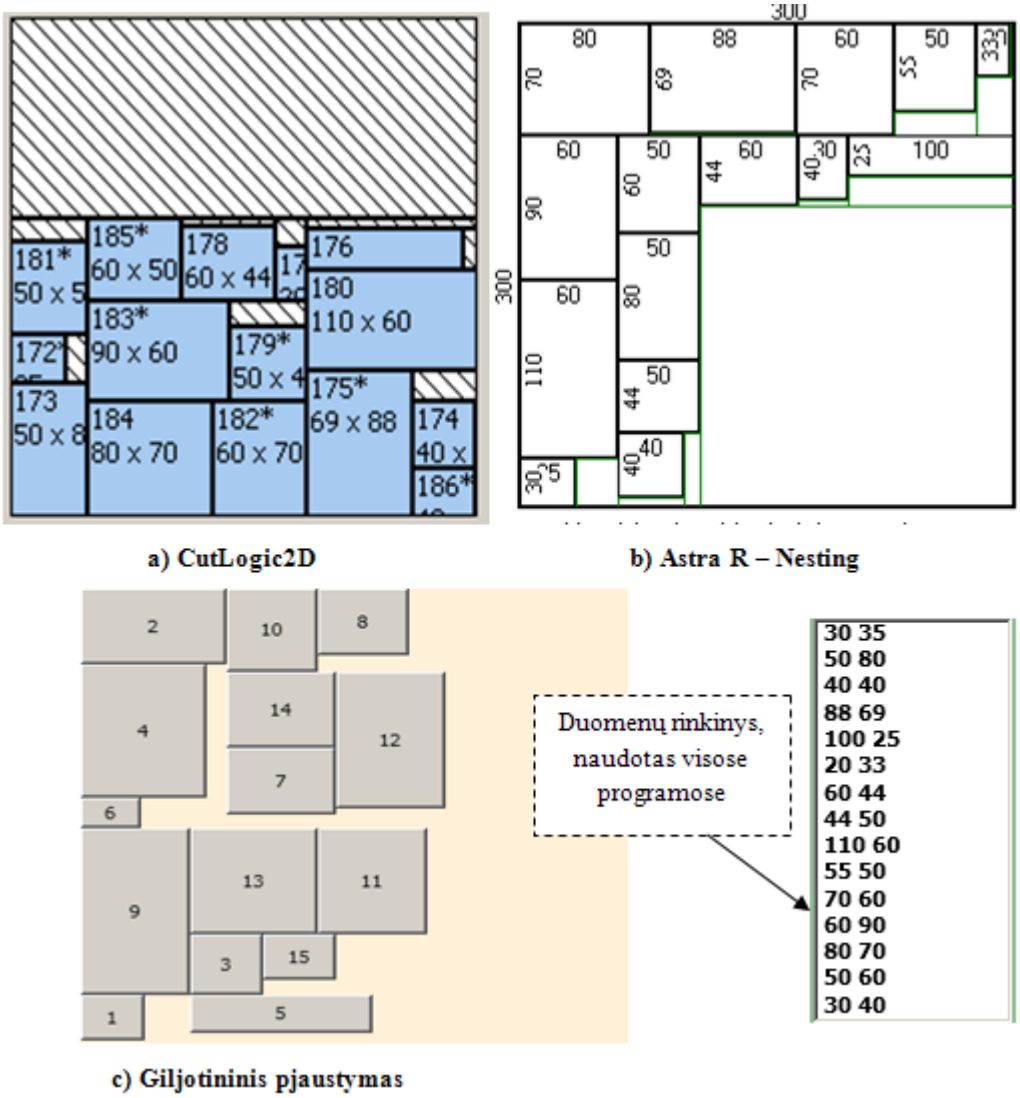
- Aprašytas detalus programos vadovas, kuris padės vartotojui greitai ir lengvai įsisavinti sukurtą programinį paketą.

5. EKSPERIMENTINIS TYRIMAS

Eksperimentiniam tyrimui buvo paruoštas duomenų rinkinys (1 priedas), kurį sudaro 15 skirtingų matmenų stačiakampių ir plokštė, kurios ilgis ir plotis yra 300. Pagal šį duomenų rinkinį buvo atliekami skaičiavimai su sukurtu programiniu paketu, kurį toliau vadinsime „Giljotininis pjaustymas“ programa ir 2.7 skyriuje aprašytomis programomis.

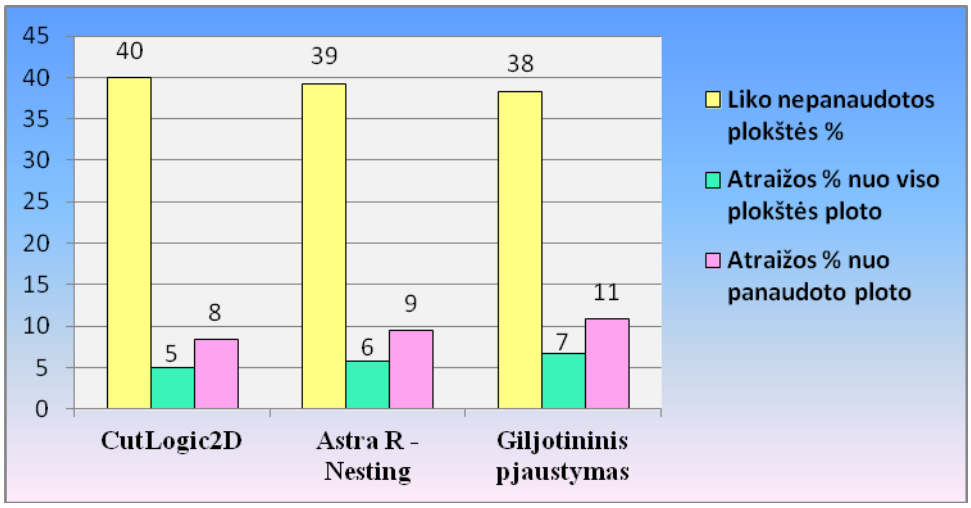
Tyrimas buvo atliekamas, siekiant palyginti stačiakampių pakavimą plokštėje skirtingomis programomis.

Naudojant paruoštą pirmą duomenų rinkinį, skaičiavimai nebuvo atlikti su „Juva1d2d” programa, kadangi šios programos bandomoji versija leidžia pakuoti tik penkis stačiakampius. Visų programų skaičiakampių išdėstymo rezultatai pateikiami 33 paveiksle.



33 pav. Stačiakampių pakavimas skirtingomis programomis

Atlikus stačiakampių pakavimą matome, kad geriausiai stačiakampiai buvo supakuoti naudojant „CutLogic2D“ programą – liko nepanaudoto ploto 40 proc. nuo bendro plokštės ploto. „Astra R – Nesting“ programa stačiakampius supakavo plokštėje palikdama 39 proc. nepanaudoto plokštės ploto, tuo tarpu „Giljotininis pjaustymas“ programa paliko 37 proc. naudingo ploto, lyginant su visos plokštės plotu.

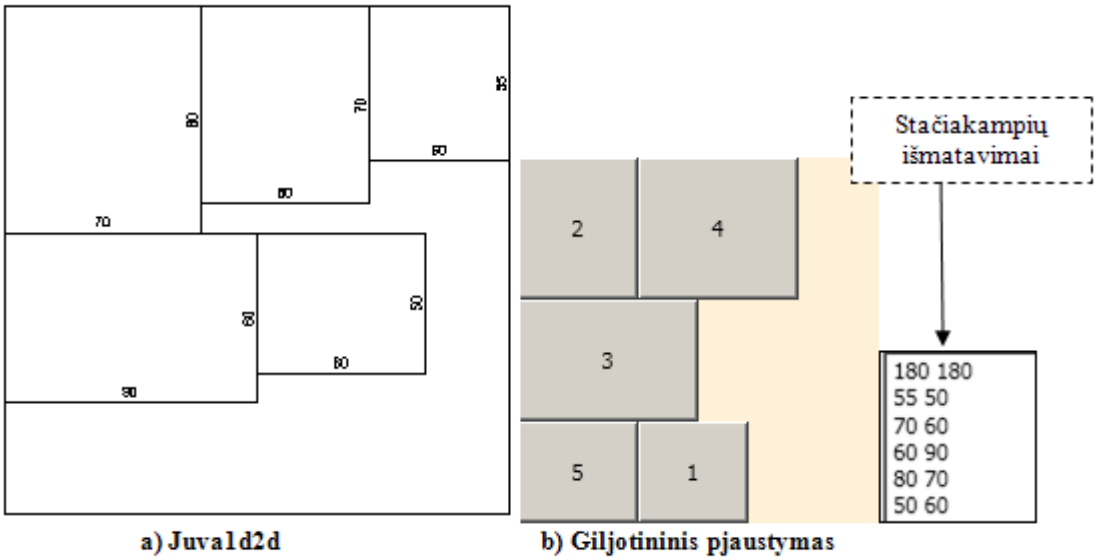


34 pav. Atraižų likučiai, stačiakampius pakavus skirtingomis programomis

„CutLogic2D“ programinis paketas paliko mažiausiai atraižų – 8 proc. nuo panaudoto plokštės ploto ir 5 proc. nuo viso plokštės ploto. „Astra R – Nesting“ paliktų atraižų procentinė dalis vienu procentiniu punktu didesnė už „CutLogic2D“ paliktas atraižas, atitinkamai 9 proc. atraižų nuo panaudoto plokštės ploto ir 6 proc. nuo viso plokštės ploto. „Giljotininis pjaustymas“ programa, lyginant su kitomis programomis, pakuodama stačiakampius plokštėje paliko daugiausiai atraižų, 11 proc. nuo panaudoto plokštės ploto ir 7 proc. nuo viso plokštės ploto.

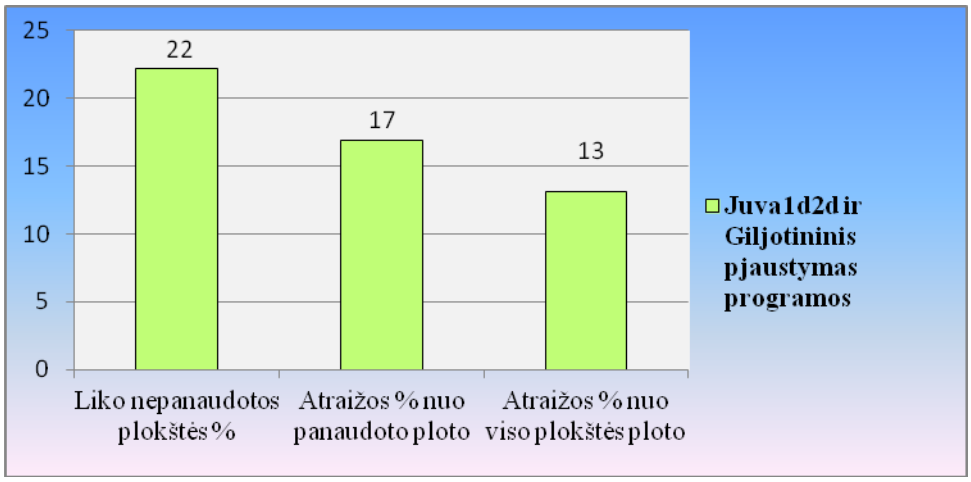
Tai rodo, kad „Giljotininis pjaustymas“ programinis paketas ir kitos lyginti programiniai paketai pateikia labai panašius rezultatus. „Giljotininis pjaustymas“ programos pranašumas tai, kad rezultatus programa pateikia labai greitai.

Norint stačiakampių pakavimą palyginti su „Juva1d2d“ programiniu paketu buvo paruoštas antras duomenų rinkinys, kurį sudaro penki skirtingų matmenų stačiakampiai (2 priedas).



35 pav. Stačiakampių pakavimas, naudojant antrą duomenų rinkinį

Stačiakampius pakuojant „Juva1d2d“ ir „Giljotininis pjaustymas“ programomis, rezultatai buvo pateikti tokie patys. Abi programos paliko vienodą procentą atraižų – 17 proc. atraižų lyginant su panaudotu plokštės plotu ir 13 proc. lyginant su visu plokštės plotu.



36 pav. Atraižų kiekis

Tai rodo, kad sukurtas programinis paketas yra tinkamas, pateikia per trumpą laiko tarpą optimalų sprendinį.

5.1 Išvados

- Palyginus stačiakampių pakavimą pagal pirmą duomenų rinkinį, „Giljotininis pjaustymas“ programa lyginant su „CutLogic2D“ ir „Astra R – Nesting“ programomis, pateikė labai panašius rezultatus.
- Pagal antrą duomenų rinkinį, lyginant su „Juva1d2d“ programa, stačiakampių pakavimo rezultatai buvo vienodi.
- Eksperimentinis tyrimas parodė, kad sukurtas programinis paketas optimaliai sprendžia stačiakampių pakavimo uždavinį.

6. IŠVADOS

1. Kitų autorių sukurtų metodų ir algoritmų naudojamų pjaustymo uždaviniams spręsti apžvalga parodė, kad pjaustymo problema yra aktuali ir yra tikslinga nagrinėti pjaustymo uždavinio sprendimo metodus ir būdus.

2. Atsižvelgiant į analitinėje dalyje aprašytus giljotininio bpjaustymo metodus, sukurtas stačiakampių pakavimo plokštėje, naudojant giljotinį pjūvį, algoritmas.

3. Apibrėžti rinkoje esančių programų privalumai ir trūkumai. Pagrindinis trūkumas, kad tai gana sudėtingos programos, todėl šio darbo tikslas buvo sukurti paprastą ir lengvai įsisavinamą programą.

4. Panaudojus objektiškai orientuotą C# programavimo kalbą buvo sukurta giljotininio pjaustymo programinė įranga.

5. Sukurta programinė įranga skirta giljotininio pjaustymo uždaviui realizuoti. Programinė įranga pakuoja stačiakampius plokštėje, palikdama kiek galima mažiau atraižų.
6. Atliktas eksperimentinis tyrimas parodė, kad sukurtas programinis paketas lyginant su kitais pjaustymo programiniais paketais, pateikia panašius rezultatus per labai trumpą laiką.

7. LITERATŪRA

1. CutLogic 2D - Panel Cutting Optimization Software, [interaktyvus], [žiūrėta 2012-04-12]. Prieiga per internetą: http://www.tmachines.com/cutlogic-2d.htm.

2. Fritsch, A.; Vornberger, O. Cutting Stock by Iterated Matching. [interaktyvus], [žiūrėta 2012 -02-20], p. 2 – 8. Prieiga per internetą: http://www.informatik.uni-osnabrueck.de/papers_pdf/or_94.pdf.

3. HOPPER, E.; TURTON B.C. An Empirical Study Of Meta – Heuristics Applied To 2D Rectangular Bin Packing. 2002, p. 77.

4. HOPPER E.; TURTON B. A Genetic Algorithm for a 2D Industrial Packing Problem. *Computers and Industrial Engineering*, 1999, p. 1 – 2.

5. KARELAHTI, J. Solving the cutting stock problem in the steel industry. *Master's thesis.* HELSINKI UNIVERSITY OF T EC HNOL OGY, 2002. [interaktyvus], [žiūrėta 2012-03-29]. Prieiga per internetą: http://www.notasyon.com/blog/4.pdf.

6. Listopadskis, N. Kombinatorinio optimizavimo uždaviniai ir jų sprendimo algoritmai. Iš *Technologijos.lt* [interaktyvus]. 2011 balandžio 8 [žiūrėta 2012–01–20]. Prieiga per internetą: http://www.technologijos.lt/p/spausdinti?name=S-18508.

7. MISEVIČIUS, A.; BLAŽAUSKAS, T.; BLONSKIS, J.; SMOLINSKAS, J. An owerview of some heuristic algorithms for combinatorial optimization problems. *Informacinės technologijos ir valdymas* [interaktyvus], 2004, Nr. 1(30) [žiūrėta 2012-03-26], p. 21 – 22. Prieiga per internetą: http://itc.ktu.lt/itc30/Misev30.pdf .

8. MISEVIČIUS, A.; BLONSKIS, J.; BUKŠNAITIS, V. Kombinatorinis optmizavimas ir metaeuristiniai metodai: teoriniai aspektai. *Informacijos mokslai* [interaktyvus]. 2007, [žiūrėta 2012 – 03 – 01], p. 213 – 215. Prieiga per internetą: http://www.leidykla.vu.lt/fileadmin/Informacijos_mokslai/42-43/213-219.pdf.

9. NESTING SOFTWARE FOR OPTIMUM USE OF MATERIALS, [interaktyvus], [žiūrėta 2012-04-13]. Prieiga per internetą: http://www.techno-sys.com/.

10. Plokščių skersinių pjaustymo optimizavimo programa Juva1d2d, [interaktyvus], [žiūrėta 2012-04-12]. Prieiga per internetą: http://www.juva1d2d.com/index.htm.

11. PUPEIKIENĖ, L. Diskretusis (vektorinis) optimizavimas. Paskaitų medžiaga, [interaktyvus], 2011 [žiūrėta 2012-02-05], p. 1 – 7. Prieiga per internetą: http://e-stud.vgtu.lt/users/files/dest/13962/.

12. SONG, X.; CHU, C. B.; NIE I. I. A Heuristic Dynamic - Programming Algorithm for 2D Unconstrained Guillotine Cutting. *IADIS International Conference Applied Computing [interaktyvus], 2004 [žiūrėta 2012-03-10], p.* 527 – 531. Prieiga per internetą: http://www.iadis.net/dl/final_uploads/200401L065.pdf.

13. Starikovičius, V. Algoritmų analizės specialieji skyriai. *Paskaitų kursas 6-oji dalis.* [interaktyvus]. 2011 [žiūrėta 2012–02–11], p. 15 – 20. Prieiga per interetą: http://www.techmat.vgtu.lt/~vs/Skaidres/AlgAnSpecSk_print_6.pdf

14. TOSHIHIKO, ONO. Optimization of Two - dimensional Guillotine Cutting by Genetic Algorithms. [interaktyvus], 1999, Fukuoka [žiūrėta 2012-03-06], p. 450 – 454. Prieiga per internetą: http://homepage3.nifty.com/ono-t/GA/GA-papers/TO-110.pdf.

15. TOSHIHIKO, ONO; IKEDA, T. Optimization of Two - dimensional Guillotine Cutting by Genetic Algorithms. EUFIT'98, 1998, September 7 – 10, p. 451 – 454.

16. TOVEY, C. A. Tutorial on Computational Complexity. School of Industrial and Systems Engineering, Georgia Institute of Technology, [interaktyvus], Atlanta, 2002 [žiūrėta 2012-03-05], p. 33 – 34. Prieiga per internetą: http://www2.isye.gatech.edu/~ctovey/tovey.tutorial.pdf.

17. VYŠNIAUSKAS, V. Programavimas c kalba. Mokomoji priemonė Elektronikos specialybės studentams, [interaktyvus], 2008 [žiūrėta 2012-01-20], p. 62 – 63. Prieiga per internetą: http://e-stud.vgtu.lt/users/files/dest/4162/programavimas_c_kalba.pdf.

8. TERMINŲ IR SANTRUMPŲ ŽODYNAS

TF – tikslo funkcija (angl. objective function).

CSP – žaliavų pjaustymo problema (Cutting Stock Problem).

GAs – genetinis algoritmas.

GCLAs – giljotininio nustatymo algoritmas.

9. PRIEDAI

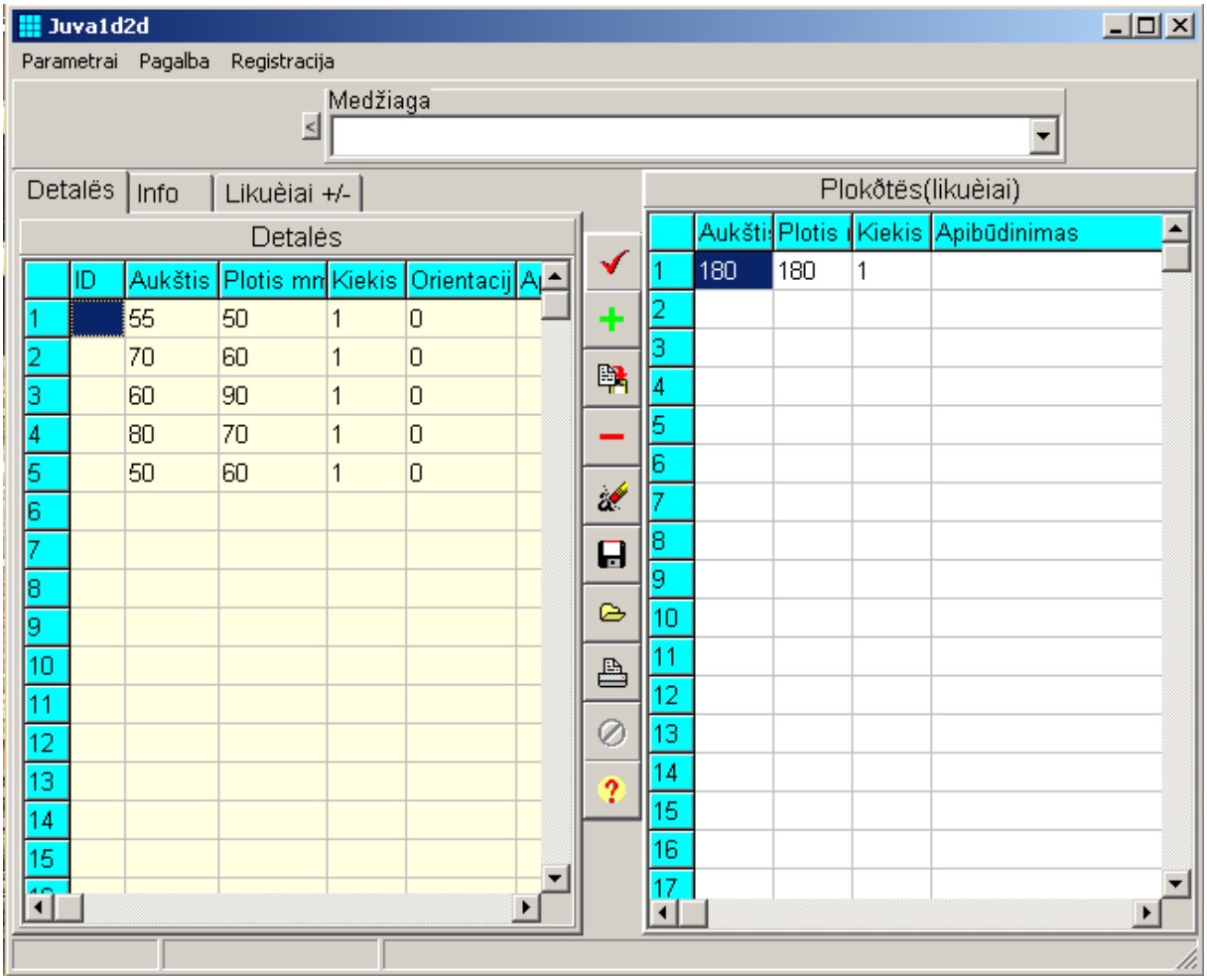
1 priedas. Pirmas duomenų rinkinys, naudotas eksperimentiniame tyrime

Stačiakampiai	Aukštis	Plotis	Plotas	CutLogic2D	Astra R - Nesting	Giljotininis pjaustymas
1	30	35	1050			
2	50	80	4000			
3	40	40	1600			
4	88	69	6072			
5	100	25	2500			
6	20	33	660			
7	60	44	2640			
8	44	50	2200			
9	110	60	6600			
10	55	50	2750			
11	70	60	4200			
12	60	90	5400			
13	80	70	5600			
14	50	60	3000			
15	30	40	1200			
Viso			49472			
Plokštė	300	300	90000			
Ploto užpildymas %			55			
Likęs naudingas plotas				36000	35340	34500
Liko nepanaudotos plokštės %				40	39	38
Sunaudota ploto				54000	54660	55500
Atraižos				4528	5188	6028
Atraižos % nuo panaudoto ploto				8	9	11
Atraižos % nuo viso plokštės ploto				5	6	7

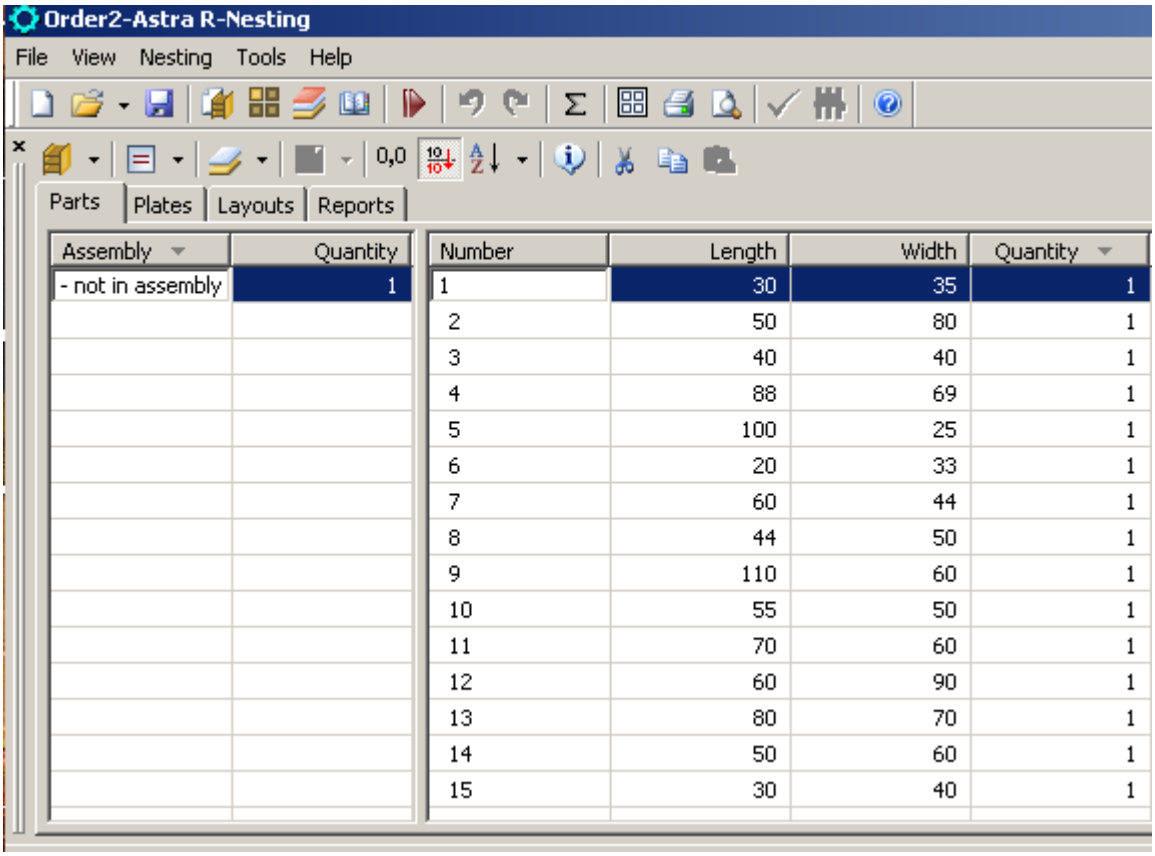
2 priedas. Antras duomenų rinkinys naudotas eksperimentiniame tyrime

Stačiakampiai	Aukštis	Plotis	Plotas	Juva1d2d	Giljotininis pjaustymas
1	55	50	2750		
2	70	60	4200		
3	60	90	5400		
4	80	70	5600		
5	50	60	3000		
Viso			20950		
Plokštė	180	180	32400		
Likęs naudingas plotas				7200	7200
Liko nepanaudotos plokštės %				22	22
Sunaudota ploto				25200	25200
Atraižos				4250	4250
Atraižos % nuo panaudoto ploto				17	17
Atraižos % nuo viso plokštės ploto				13	13

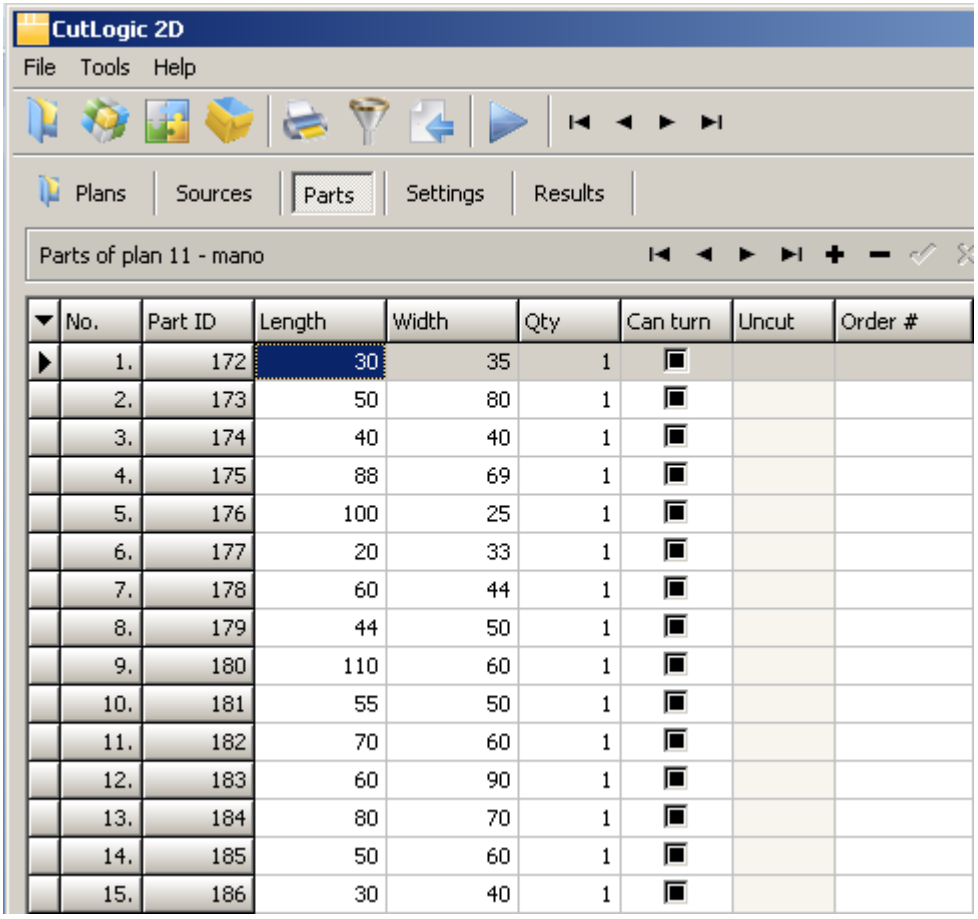
3 priedas. Juva1d2d programos langas su įvestais duomenimis



4 priedas. Astra R – Nesting programos langas su įvestais duomenimis



5 priedas CutLogic2D programos langas su įvestais duomenimis



No.	Part ID	Length	Width	Qty	Can turn	Uncut	Order #
1.	172	30	35	1	■		
2.	173	50	80	1	■		
3.	174	40	40	1	■		
4.	175	88	69	1	■		
5.	176	100	25	1	■		
6.	177	20	33	1	■		
7.	178	60	44	1	■		
8.	179	44	50	1	■		
9.	180	110	60	1	■		
10.	181	55	50	1	■		
11.	182	70	60	1	■		
12.	183	60	90	1	■		
13.	184	80	70	1	■		
14.	185	50	60	1	■		
15.	186	30	40	1	■		