VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INŽINERINĖS GRAFIKOS KATEDRA

Aurimas Jokymaitis

ĮVAIRIOS FORMOS 2D KONTŪRŲ OPTIMALAUS DĖSTYMO STAČIAKAMPIAME LAKŠTE ALGORITMAS
ALGORITHM FOR OPTIMAL POSITIONING OF 2D VARIOUS CONTOURS ON RECTANGULAR PIECES

Baigiamasis magistro darbas

Informacinių technologijų studijų programa, valstybinis kodas 6211BX016

Skaitmeninės inžinerinės grafikos technologijų specializacija

Informatikos inžinerijos studijų kryptis

Vilnius, 2022

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Inžinerinės grafikos katedra

ISBN ISSN
Egz. sk. .....
Data .....-.....-.....

Antrosios pakopos studijų **Informacinių technologijų** programos magistro baigiamasis darbas 4

Pavadinimas **Įvairios formos 2D kontūrų optimalaus dėstymo stačiakampiame lakšte algoritmas**

Autorius **Aurimas Jokymaitis**

Vadovas **Saulius Dereškevičius**

Kalba: lietuvių

Anotacija

Magistro baigiamajame darbe nagrinėjama įvairios formos 2D kontūrų dėstymo stačiakampiame lakšte algoritmo tema. Darbo tikslas – išanalizuoti įvairios formos 2D kontūrų optimalaus dėstymo stačiakampiame lakšte algoritmus. Atlikta įvairios formos 2D kontūrų dėstymo algoritmų ir jų galimybių analitinę literatūros apžvalga. Ištirtas įvairios formos 2D kontūrų dėstymo algoritmų veikimo CAN sistemoje efektyvumas. Eksperimentiniu būdu ištirtas euristinių dėstymo algoritmų efektyvumas sprendžiant skirtingų formų 2D kontūrų dėstymo uždavinius. Darbo apimtis: 62 psl. teksto be priedų, 32 paveikslėliai, 13 lentelių, 23 bibliografiniai šaltiniai, 1 priedas.

Prasminiai žodžiai: Įvairios formos 2D kontūrų dėstymo algoritmai, stačiakampis lakštas, euristiniai dėstymo algoritmai, aptvaro algoritmai, dinaminiai paskirstymo algoritmai, dėstymo apačioje kairėje algoritmai, grupavimo algoritmai, atsitiktinio dėstymo algoritmai, laisvos vietos ieškojimo algoritmai, įterpimo funkcija, NFP, IFP, CAN sistema, vidiniai ir išoriniai kontūrai, grafiniai regionai.

Vilnius Gediminas Technical University
Faculty of Fundamental Sciences
Department of Engineering Graphics

ISBN ISSN
Copies No. .....
Date .....-.....-.....

Master Degree Studies **Information Technologies** study programme Master Graduation Thesis 4

Title	**Algorithm for Optimal Positioning of 2D Various Contours on Rectangular Sheet**
Author	**Aurimas Jokymaitis**
Academic supervisor	**Saulius Dereškevičius**

Thesis language: Lithuanian

Annotation

In master's thesis two-dimensional nesting problem with items of irregular shape and rectangular sheets is analyzed. The aim of master's thesis is to analyze algorithms for optimal positioning of 2D irregular shape contours on rectangular sheet. Analytical review of algorithms for optimal positioning of 2D irregular shape contours on rectangular sheet is performed. The effectiveness of algorithms implemented in CAN system is analyzed. The effectiveness of heuristic nesting algorithms with implemented fitting functions and NFP, IFP technologies is investigated. Thesis consist of: 62 p. text without appendixes, 32 figures, 13 tables, 23 bibliographical entries, 1 appendix included.

Keywords: Two-dimensional nesting problem with items of irregular shape, algorithms for optimal positioning of 2D irregular shape contours on rectangular sheet, enclosure algorithm, dynamic allocation algorithm, bottom-left nesting algorithm, heuristic nesting algorithms, random nesting algorithms, clustering algorithms, space searching algorithms, fitting function, NFP, IFP, intelligent CAN system, graphical regions.

Turinys

Paveikslų sąrašas

Lentelių sąrašas

1. ĮVADAS

Magistro baigiamajame darbe nagrinėjama įvairios formos 2D kontūrų dėstymo (angl. *nesting*) stačiakampiame lakšte algoritmo tema. Algoritmo pagrindinis uždavinys yra automatizuotai išdėstyti pateikiamų figūrų kontūrus stačiakampės formos perimetro viduje taip, kad būtų optimaliai išnaudojamas paviršiaus plotas – būtų sunaudojamas mažiausias kiekis medžiagos (figūros turi būti išdėstytos taip, kad tarp jų būtų maži tarpeliai). Kelias, reikalingas apvesti figūrų kontūrus, turi būti apskaičiuojamas ir parenkamas trumpiausias iš galimų, nes tai aktualu, kai algoritmas pritaikomas praktikoje – dažniausiai tokie algoritmai veikia pjovimo įrenginiuose, kuriuose pjovimą atlieka robotizuota pjovimo galva su lazeriu, stiklo rėžtuku ar kitu mechanizmu. Todėl aktualu, jog pjovimo kelias būtų kuo mažesnis – būtų sutaupomi energijos ir laiko resursai. Algoritmas turi veikti greitai ir tiksliai, todėl šie parametrai yra ypač svarbūs analizuojant algoritmus.

Šiame darbe analizuojami įvairios formos 2D kontūrų dėstymo stačiakampiame lakšte algoritmai, jų veikimo greičio ir taškų suradimo bei skaičiavimo parametrai, savybės.

1.1. Darbo aktualumas

Įvairios formos 2D kontūrų dėstymo stačiakampiame lakšte algoritmai plačiai taikomi gamyboje. Dažniausiai jie naudojami įrenginiuose (automatinės pjaustymo mašinos, angl. *2D cutting machines*), skirtuose valdyti robotizuotas pjovimo galvas, kurios pagal įvestus ir iš valdymo įrenginio gaunamus parametrus išpjauna nustatytos formos dalis. Toks įrenginys valdomas kompiuterio, kuriame yra realizuota galimybė įkelti 2D figūrų kontūrus bei juos išdėstyti plokštumoje. Kompiuteryje turi būti įdiegta specializuota programinė įranga skirta valdyti pjovimo galvą ir įdiegtas programinis kodas (algoritmas) leidžiantis įkeltus 2D figūrų kontūrus automatizuotai išdėstyti stačiakampiame plote.

Siekiant darbo su minėtais įrenginiais kokybės bei laiko ir materialių resursų sutaupymo, reikalingas įvairios formos 2D kontūrų optimalaus dėstymo stačiakampiame lakšte algoritmas. Svarbiausios tokių algoritmų savybės: veikimo greitis, kontūrų padėčių suradimas bei atstumų suskaičiavimo tarp jų tikslumas. Taip pat kontūrų išdėstymo kokybė paviršiaus ploto sutaupymo atžvilgiu bei optimalus kontūrų padėties parinkimas.

Darbo tikslas – išanalizuoti įvairios formos 2D kontūrų optimalaus dėstymo stačiakampiame lakšte algoritmus.

1.2. Darbo uždaviniai

Siekiant užsibrėžto tikslo, darbe formuluojami šie uždaviniai:

1. Atlikti įvairios formos 2D kontūrų dėstymo algoritmų ir jų galimybių analitinę literatūros apžvalgą.
2. Ištirti įvairios formos 2D kontūrų dėstymo algoritmų veikimo CAN sistemoje efektyvumą.
3. Ištirti euristinių dėstymo algoritmų efektyvumą sprendžiant skirtingų formų 2D kontūrų dėstymo uždavinius.

1.3. Naudoti tyrimo ir analizės metodai

Darbe naudojama analitinė literatūros apžvalga, kuris skirta analizuoti egzistuojančius įvairios formos 2D kontūrų optimalaus dėstymo algoritmus, jų savybes ir dėstymo uždavinių sprendimo galimybes. Taip pat darbe aprašomi atlikti eksperimentiniai tyrimai su euristiniais dėstymo algoritmais bei pateikiami tyrimų rezultatai ir išvados.

1.4. Darbo naujumas ir praktinė nauda

Darbe analizuojant įvairios formos 2D kontūrų dėstymo algoritmų literatūrą, tiriant kombinuotų dėstymo algoritmų veikimo savybes CAN sistemoje bei atliekant eksperimentinius euristinių dėstymo algoritmų tyrimus ieškoma šiuolaikinių sprendimų įvairios formos 2D kontūrų optimalaus dėstymo užduotims spręsti. Darbas yra aktualus, nes šių užduočių sprendimas tiesiogiai įtakoja energijos bei medžiagos resursų sąnaudų panaudojimą, kai algoritmai yra taikomi praktiškai.

1.5. Darbo struktūra

Magistro baigiamojo darbo aiškinamąjį raštą sudaro 5 skyriai. 1 aiškinamojo rašto skyrius – įvadas. 2-ame skyriuje pateikiama analitinė literatūros apžvalga. Nagrinėjami įvairios formos 2D kontūrų optimalaus dėstymo stačiakampiame lakšte algoritmai, jų taikymo sritys, parametrai, savybės. Apžvelgiamos algoritmų įgyvendinimo galimybės įrenginiuose. Pasirenkami keli algoritmai tolesniam tyrimui. 3 skyrius skirtas algoritmų realizavimo metodikoms bei uždavinių sprendimo technologijų tyrimui skirtingose programinėse aplinkose. 5-ame skyriuje darbas apibendrinamas ir pateikiamos išvados apie įvairios formos 2D kontūrų optimalaus dėstymo stačiakampiame lakšte algoritmų analizės ir tyrimo rezultatus.

2. ANALITINĖ LITERATŪROS APŽVALGA

Šio skyriaus tikslas yra apžvelgti įvairios formos 2D kontūrų optimalaus dėstymo stačiakampiame lakšte algoritmus, kurie gali būti realizuojami automatinėse pjaustymo, kirpimo mašinose.

Sprendžiant 2D kontūrų dėstymo plokštumoje problemas, reikalinga analizuoti kontūrų ir lakštų tipus ir dydžius, skirtingas kontūrų dėstymo užduočių sąlygas. Šios problemos kompleksiškumas yra aukšto lygio, todėl norint spręsti sudėtingus įvairių kontūrų dėstymo uždavinius dažniausiai yra pasitelkiamas kompleksas priemonių (algoritmai ir jų kombinacijos, juos apjungiančios sistemos). Šiame skyriuje sistemingai apžvelgiami dėstymo algoritmai, kurie pritaikyti pasiekti skirtingus 2D kontūrų dėstymo užduočių rezultatus. Analitinėje apžvalgoje išskiriami keli algoritmų tipai:

- aptvaro (angl. *enclosure algorithms*);
- dinaminis paskirstymo (angl. *dynamic allocation algorithm*);
- dėstymo apačioje kairėje (angl. *bottom-left nesting algorithms*);
- euristiniai dėstymo (angl. *heuristic nesting algorithms*);
- atsitiktinio dėstymo (angl. *random nesting algorithms*);
- grupavimo (angl. *clustering algorithms*);
- laisvos vietos ieškojimo (angl. *space searching algorithms*);
- hibridiniai (angl. *hybrid algorithms*).

Išanalizavus įvairios formos 2D kontūrų dėstymo algoritmų literatūros šaltinius, pastebėta jog šių algoritmų problematika priklauso nuo pačių 2D kontūrų tipo. Todėl optimalaus dėstymo užduotims spręsti kontūrai suskirstomi į 3 grupes:

1. Identiški kontūrai (angl. *identical parts*) – kontūrai turintys tokias pat formas, dydžius. Kiekvienas toks kontūras yra kopija.
2. Kolekcionuojami kontūrai (angl. *collectable parts*) – kontūrai, kurių formos yra skirtingos, tačiau jie lengvai sugrupuojami tarpusavyje ir patalpinami į aptvarus (dažniausiai stačiakampius).
3. Skirtingi kontūrai (angl. *multiple parts*) – turintys skirtingas formas, dydžius, negali būti lengvai sugrupuojami ir patalpinami į aptvarus.

Neegzistuoja toks dėstymo algoritmas, kuris pats vienas sugebėtų išspręsti visus skirtingų kontūrų grupių dėstymo uždavinius. Todėl sprendžiant dėstymo uždavinius algoritmai yra kombinuojami tarpusavyje. 1.1 lentelėje pateikiamas sąrašas algoritmų, kurie suskirstyti pagal tinkamumą spręsti atitinkamų kontūrų grupių dėstymo uždavinius (International Journal of Computer Integrated Manufacturing, 2007).

2-1 lentelė. Dėstymo algoritmų parinkimas pagal kontūrų grupes

Eil. Nr.	Algoritmai	Identiški kontūrai	Kolekcionuojami kontūrai	Skirtingi kontūrai
1	Aptvaro	Taip	Taip	Taip
2	Dinaminiai paskirstymo	Taip	Taip	Taip
3	Dėstymo apačioje kairėje	Taip	Taip	Taip
4	Grupavimo		Taip	Taip
5	Atsitiktinio dėstymo			Taip
6	Euristinis paieškos			Taip
7	Euristinis užpildymo			Taip

Pagal pateiktą sąrašą galima išvada, jog aptvaro, dėstymo apačioje kairėje ir dinaminiai paskirstymo algoritmai yra dažniausiai naudojami sprendžiant įvairių 2D kontūrų optimalaus dėstymo problemas. Euristiniai algoritmai labiau tinkami atvejuose kai kontūrai tarpusavyje skiriasi ir jie yra sunkiai sugrupuojami. Tačiau beveik visose kontūrų dėstymo užduotyse, kuriose dalyvauja euristiniai algoritmai, taip pat yra panaudojami ir aptvaro, dėstymo apačioje kairėje algoritmai atlikti tam tikriems proceso žingsniams.

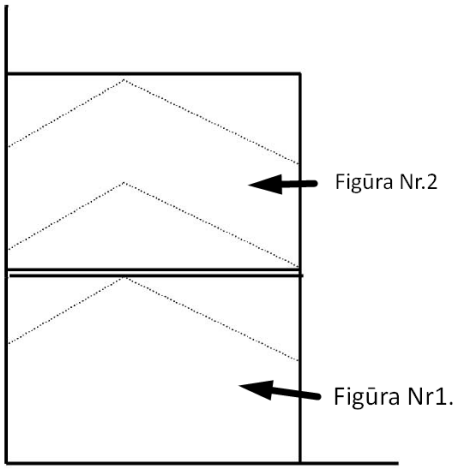
Kiekvienam darbe analizuojamam algoritmų tipui yra skiriamas atskiras skyriaus poskyris. Išvardinti visų aptartų algoritmų veikimo principai, privalumai, trūkumai bei galimos jų taikymo sritys.

2.1. Aptvaro algoritmai

Aptvaro algoritmai apima tris proceso žingsnius: figūros kontūro įterpimas į aptvarą, kontūrų dėstymas ir atnaujinimas. Netaisyklingos formos daugiakampių kontūrams dėstyti plačiai naudojami dviejų tipų aptvaro algoritmai: stačiakampis ir šešiakampis. Šių algoritmų pagrindinis privalumas – didelis skaičiavimo greitis, tačiau kontūrų išdėstymo efektyvumas yra žemas (Cheng ir Rao, 2000).

Stačiakampio ir šešiakampio aptvaro algoritmų veikimo principas: kiekvienos figūros kontūras įkeliamas į stačiakampį arba šešiakampį aptvarą, po to šie aptvarai su kontūrais išdėstomi lakšte. Jeigu figūra yra daugiakampis, algoritmas pirmiausia suranda visas padėtis

kuomet figūros kraštinė yra lygiagreti su horizontalia arba vertikalia ašimi. Kiekvienai padėčiai apskaičiuojamas stačiakampis arba šešiakampis aptvaras (Roth et al., 1985).



2.1 pav. Kontūrai „Figūra Nr.1“ ir „Figūra Nr.2“ patalpinti į stačiakampius aptvarus

Jeigu figūra yra išgaubtas daugiakampis, tokiu atveju algoritmas apskaičiuoja mažiausio ploto (angl. *minimum-area rectangular enclosure*) stačiakampį aptvarą. Tokio aptvaro, apgaubiančio išgaubtą daugiakampį, bent viena kraštinė yra lygiagreti su vienu iš daugiakampio kraštinių (Grinde ir Cavalier, 1995).

Aptvaro algoritmai yra plačiai naudojami dėl to, jog figūrų patalpinimas į aptvarus leidžia šiuos duomenis ir rezultatus vėliau panaudoti įvairiems kitiems skaičiavimams. Dažnai šie algoritmai kombinuojami su kitais 2D kontūrų dėstymo algoritmais.

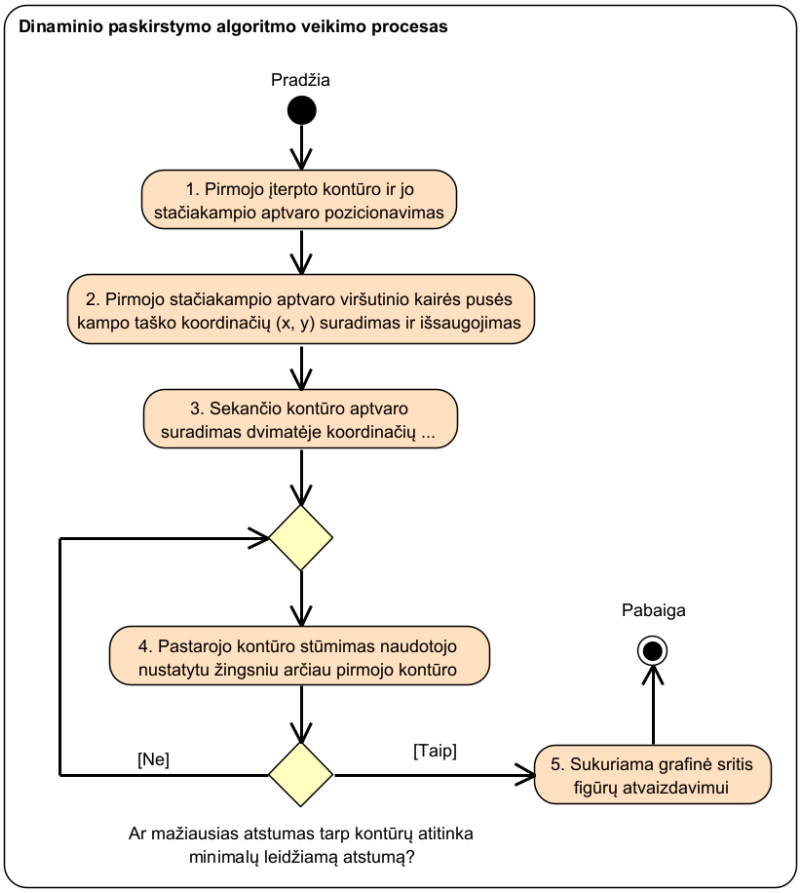
Stačiakampio aptvaro algoritmas plačiai taikomas kai reikia išdėstyti vienodas figūras lakšte. Taip yra dėl to, jog apskaičiavus kontūrui aptvarą nebereikia jo skaičiuoti kitoms figūroms. Toliau algoritmas naudojamas tik kontūrų dėstymo lakšte proceso įgyvendinimui. Taip pat, jeigu aptvaro algoritmas kombinuojamas kartu su dinaminiu paskirstymo algoritmu (žr. 2.2 poskyrį), galima išgauti labai efektyviai veikiančią aplikaciją, kuri itin greitai apskaičiuos aptvarus ir optimaliai išdėstys kontūrus lakšte leisdama maksimaliai sutaupyti medžiagos resursų. Tokia aplikacija vartotojui aiškiai suprantamu būdu leis pasirinkti dėstymo žingsnius bei tarpelių tarp kontūrų minimalius ir maksimalius atstumus.

Aptvaro algoritmų pagrindinis trūkumas – apgaubiant kontūrus stačiakampiu arba daugiakampiu lieka dideli tušti tarpai tarp aptvaro ir kontūro kraštinių. Dėl šios priežasties aptvaro algoritmas turi būti modifikuojamas, tačiau dažniausiai jis panaudojamas kaip dalis viso sprendimų komplekso. Tuščių tarpų tarp kontūrų problemą geriausiai išsprendžia euristiniai paieškos bei užpildymo algoritmai (žr. 2.4 poskyrį).

2.2. Dinaminiai paskirstymo algoritmai

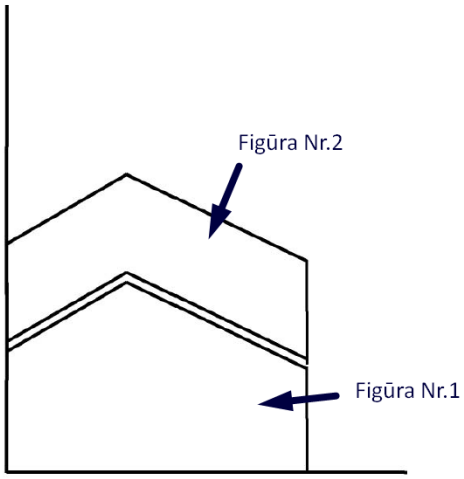
Išstudijavus tradicinių aptvaro algoritmų problematiką (tuščių tarpų atsiradimas po kontūrų apgaubimo aptvaru) buvo sukurtas dinaminis paskirstymo algoritmas. Pritaikant šį algoritmą pagrindinis tikslas yra sumažinti tarpus, atsiradusius po kontūrų patalpinimo aptvare ir jų išdėstymo lakšte. Tai galima pasiekti keičiant kontūrų padėtis ir ieškant, kurioje tarpai būtų kiek įmanoma mažesni. Dinaminio paskirstymo algoritmo pagrindiniai proceso žingsniai:

1. Pirmojo kontūro ir jo stačiakampio aptvaro pozicionavimas.
2. Pirmojo stačiakampio aptvaro viršutinio kairės pusės kampo taško koordinačių (x, y) suradimas ir išsaugojimas.
3. Sekančio kontūro aptvaro suradimas dvimatėje koordinačių sistemoje (x, y). Svarbi sąlyga – šis aptvaras neturi persidengti su pirmojo kontūro aptvaru.
4. Pastarasis kontūras stumiamas žingsniu (žingsnio parametras nustatomas naudotojo) arčiau pirmojo kontūro taip, kad figūros nesusiliestų.
5. Sukuriama grafinė sritis kurioje atvaizduojamos abi figūros.



2.2 pav. Dinaminio paskirstymo algoritmo veikimo proceso diagrama

Žemiau esančiame 2.3 paveikslėlyje matomas mažas tarpelis tarp kontūrų. Jis buvo gautas žingsniškai stumiant figūrą Nr.2 arčiau figūros Nr.1 iki nustatyto minimalaus leidžiamo atstumo tarp kontūrų.



2.3 pav. Kontūrų padėtys po dinaminio paskirstymo algoritmo pritaikymo

Naudojant dinaminį paskirstymo algoritmą realizuojama galimybė naudotojui keisti tarpelio dydį. Palyginus su 2.1. pav., kuriame kontūrams dėstyti buvo taikomas tik stačiakampio aptvaro algoritmas, aiškiai matoma jog dinaminis paskirstymo algoritmas naudodamas kontūrų perstūmimą optimaliai parinko kontūrų padėtis. Tokiu būdu sutaupoma nemaža dalis lakšto medžiagos. Taip pat šis algoritmas tinkamas naudoti su visais kontūrų tipais (identiškais, kolekcionuojamais, skirtingais).

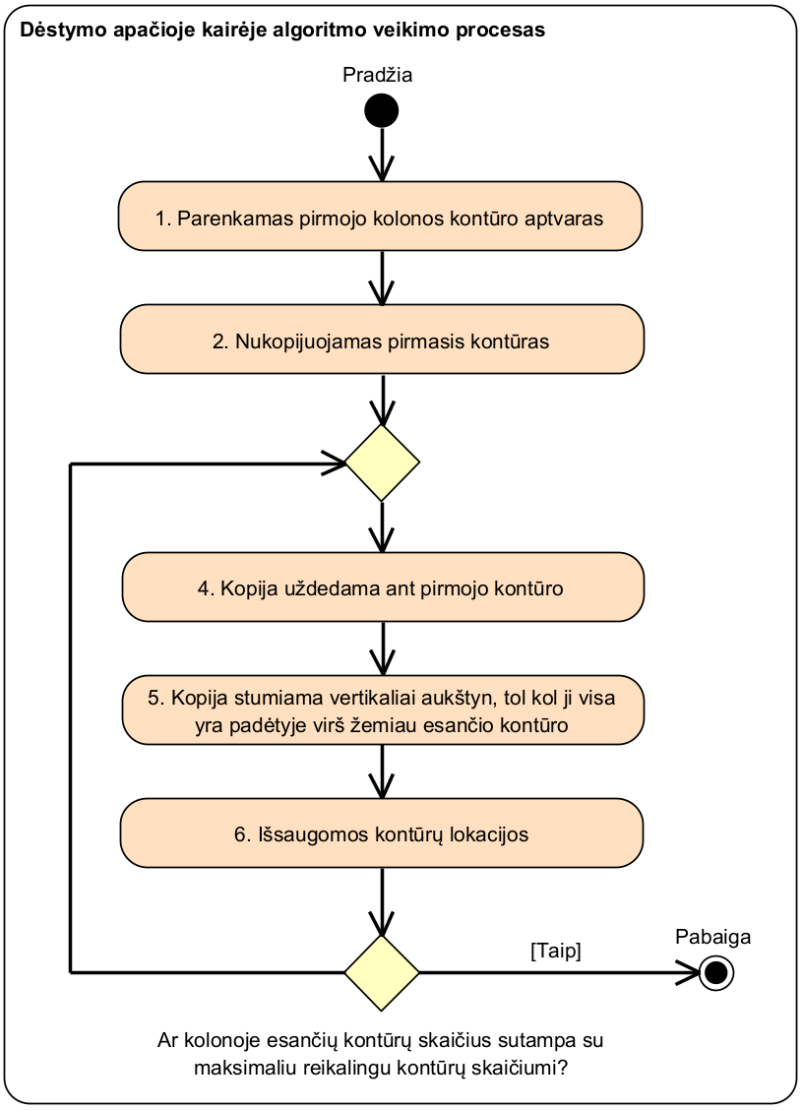
2.3. Dėstymo apačioje kairėje algoritmai

Dėstymo apačioje kairėje algoritmo veikimo principas: figūrų kontūrai dėstomi kolonomis pradedant nuo lakšto apatinio kairiojo kampo. Kai viena kolona užpildoma kontūrais tiek, kad į ją daugiau kontūrų nebetelpa – pradedama pildyti sekanti kolona nuo apačios. Jeigu komplekte yra keli figūrų kontūrai, kurių pločiai panašūs, algoritmas dėstymo procese jiems suteikia prioritetą. Vėliau pagal figūrų kontūrų matmenis arba pagal pasirinktą šabloną parenkamas optimaliausias sprendimas. Šio algoritmo versijos apima vieną ar kelis skirtingus dėstymų variantus. (Dowsland et al. 2002, Nandy ir Bhattacharya 2003, Roy ir Dasari 1998).

Dėstymo apačioje kairėje algoritmo svarbiausi pranašumai (lyginant su kitais įvairios formos 2D kontūrų optimalaus dėstymo algoritmais) – veikimo greitis ir paprastumas. Lyginant su kitais šios paskirties algoritmais jie yra labiau ištobulinti ir gali generuoti geresnius sprendimų rezultatus, tačiau dėstymo apačioje kairėje algoritmas labiau naudingas ten, kur

reikalingas didesnis veikimo greitis ir ne taip svarbu sutaupyti kiekvieną pjaustomos medžiagos gabalėlį. Dėl šių priežasčių šis algoritmas yra įdiegtas dideliame kiekyje komerciškai naudojamų įrenginių.

Tradicinis dėstymo apačioje kairėje algoritmas gali efektyviai ir greitai apskaičiuoti ir išdėstyti pasikartojančius kontūrus, todėl jis tinkamas identiškų kontūrų grupės dėstymo uždaviniams spręsti. Dėstant identiškus kontūrus dėstymo apačioje kairėje algoritmo veikimo proceso žingsnių diagrama pateikta žemiau esančiame 2.4 paveikslėlyje.



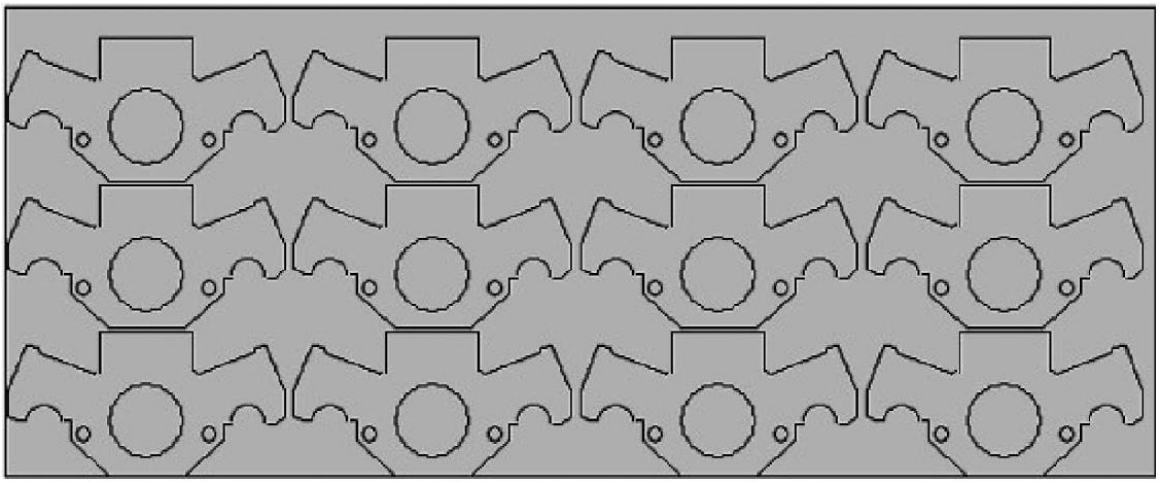
2.4 pav. Įprasto dėstymo apačioje kairėje algoritmo vienos kolonos dėstymo proceso diagrama

Beveik visada, po kontūrų išdėstymo naudojant dėstymo apačioje kairėje algoritmą, pasitaikanti problema – lieka laisva erdvė tarp viršutinio lakšto kraštinės ir arčiausiai esančių kontūrų. Į šią erdvę nebetelpa sekantis kontūras, todėl algoritmas pradeda likusius kontūrus dėstyti jau kitoje kolonoje. Šios problemos rezultatas: žemas 2D kontūrų išdėstymo

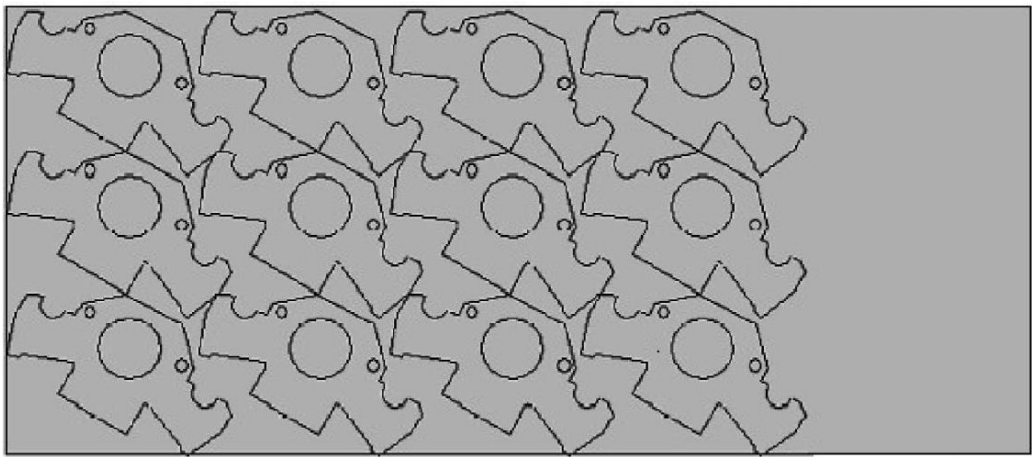
naudingumo koeficientas. Spręsti šiai problemai dėstymo apačioje kairėje algoritmas modifikuojamas. Modifikuoto algoritmo veikimo procesas aprašomas šiais žingsniais:

1. Kontūrai išdėstomi kolonomis (kaip ir įprastai).
2. Po pirmos kolonos užpildymo kontūrais, kontūrai pasukami (angl. *rotate*) nuo 0° iki 360°.
3. Po kiekvieno pasukimo algoritmas apskaičiuoja atstumą nuo viršutinės stačiakampio lakšto kraštinės iki kolonos viršutinio dešinio kampo, kuris sutampa su aukščiausiai kolonoje esančio kontūro aukščiausiu tašku.
4. Algoritmas išsaugo kiekvieno pasukimo atveju, apskaičiuotą algoritmo dėstymo naudingumo koeficientą.
5. Lyginami visų pasukimo padėčių apskaičiuoti naudingumo koeficientai ir išrenkamas pats efektyviausias.
6. Efektyviausią naudingumo koeficientą turinti kontūrų kolona kopijuojama iki kol pasiekiamas reikalingas kontūrų kiekis lakšte.

Žemiau esančiuose 2.5 ir 2.6 paveikslėliuose pavaizduoti identiškų kontūrų dėstymo uždavinių rezultatai gauti naudojant paprasčiausią dėstymo apačioje kairėje algoritmą ir modifikuotą jo versiją. Pirmame paveikslėlyje matyti, jog yra tarpas tarp lakšto viršutinės kraštinės ir aukščiausiai esančių kontūrų. Antrasis paveikslėlis vaizduoja kontūrus po pasukimo veiksmo. Matoma, jog tarp viršutinės stačiakampio lakšto kraštinės ir aukščiausiai esančio kontūro, tarpelis minimalus. Po modifikavimo naudingumo koeficientas pakilo nuo 66% iki 81% (International Journal of Computer Integrated Manufacturing, 2007).

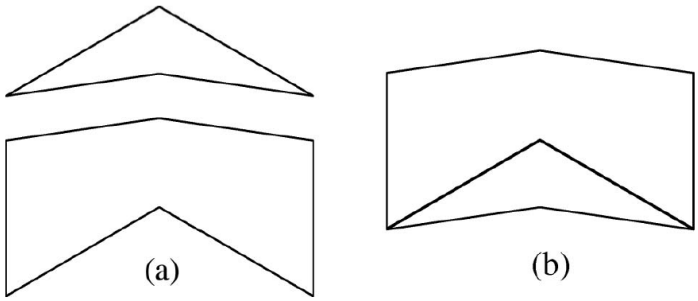


2.5 pav. Kontūrų dėstymas naudojant įprastą dėstymo apačioje kairėje algoritmą



2.6 pav. Kontūrų dėstymas naudojant modifikuotą dėstymo apačioje kairėje algoritmą

Dėstymo uždaviniuose, kuriuose dominuoja identiški kontūrai arba kontūrai kurių pločiai yra panašūs, ypač efektyviai sprendimus generuoja dėstymo apačioje kairėje ir euristinio algoritmo junginys.



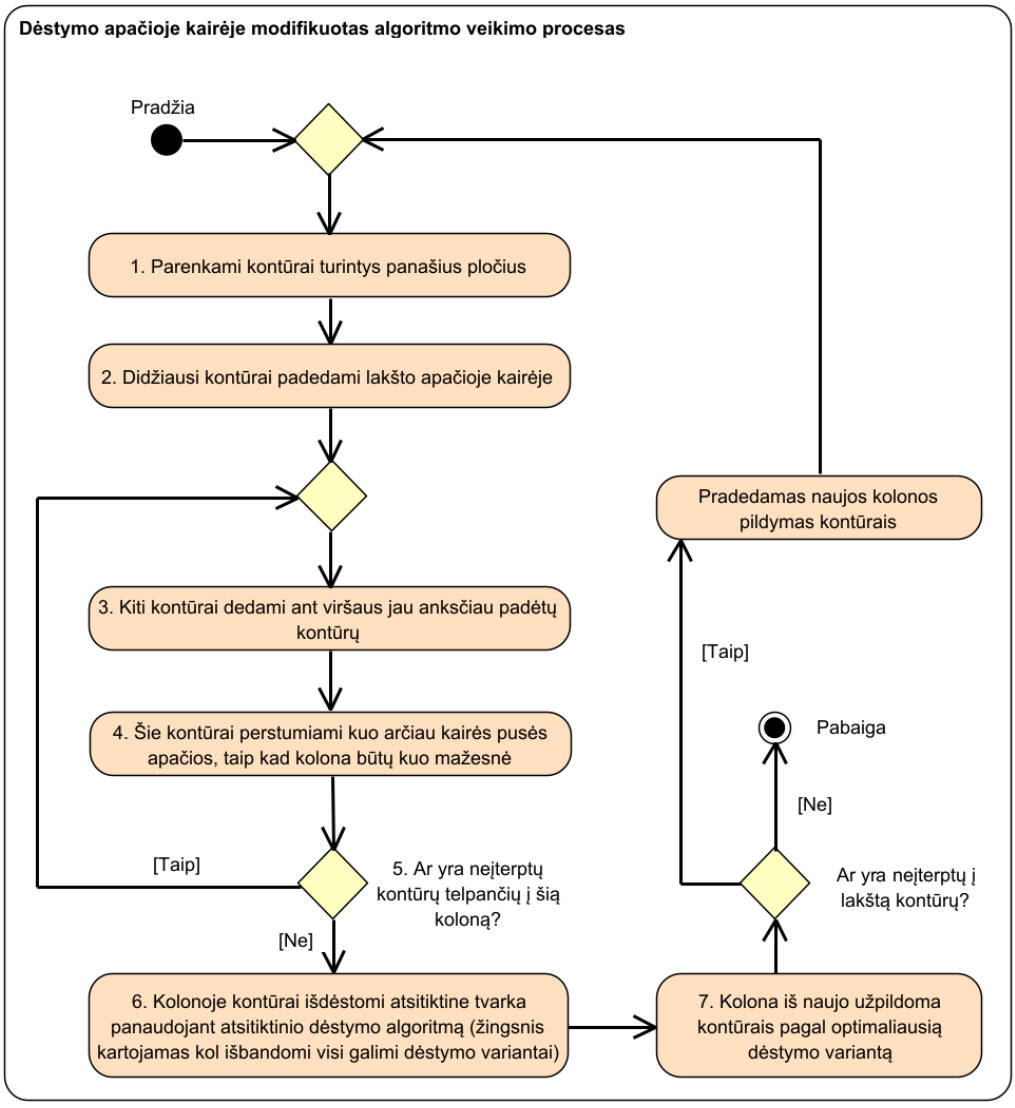
2.7 pav. (a) tradicinio apačioje kairėje dėstymo algoritmo sprendimo rezultatas; (b) optimalus kontūrų dėstymo sprendimas

Aukščiau esančiame 2.7 paveikslėlyje pavaizduota kaip 2 kontūrus lakšte išdėsto tradicinis dėstymo apačioje kairėje algoritmas (a dalis). Dėstymo efektyvumų skirtumas tarp a dalies ir b dalies yra 76% ir 91% (pagal International Journal of Computer Integrated Manufacturing, 2007). Tačiau optimalus tokio kontūrų uždavinio dėstymo variantas turėtų atitikti b dalyje pavaizduotą sprendimą. Naudojant dėstymo apačioje kairėje algoritmą galimi 2 sprendimai šiam uždaviniui:

1. Kontūrus įdėti į aptvarus panaudojant aptvaro algoritmą o likusias tuščias erdves tarp kontūrų užpildyti kontūrais su euristinio paieškos algoritmo pagalba (žr. 2.4. poskyrį) juos įterpiant ir perstumiant. Nors euristinio algoritmo naudojimas įterpiant kontūrus yra efektingas, tačiau algoritmo sudėtingumas žymiai prailgina uždavinio sprendimo laiką. Šiuo būdu optimaliai išspręsti uždavinio nepavyks, nes apačioje kairėje

pirmiausia bus pozicionuojamas didžiausias kontūras ir virš jo įterpiamas panašiausią plotį turintis kitas kontūras.

2. Modifikuoti dėstymo apačioje kairėje algoritmą taip, kad jis būtų orientuotas į lakšto ploto optimizavimą kolonų atžvilgiu. Šiai modifikacijai panaudojamas atsitiktinio dėstymo algoritmas (žr. 2.5. poskyrį). Atsitiktinio dėstymo algoritmas pritaikomas 6-ame žingsnyje, tam kad būtų sumažinami tarpai tarp lakšto kraštinių ir kontūrų. Dėstymo apačioje kairėje ir atsitiktinio dėstymo algoritmų junginio proceso žingsniai pateikti žemiau esančioje diagramoje (2.8 pav.).



2.8 pav. Modifikuoto dėstymo apačioje kairėje proceso žingsnių diagrama

Dažniausiai dėstymo apačioje kairėje algoritmai ir jų modifikacijos yra įdiegiami ir naudojami su kitais įvairios formos 2D kontūrų optimalaus dėstymo algoritmais. Tokių sprendimų rezultatas priklauso nuo konkrečios dėstymo užduoties ir atliktų modifikacijų.

2.4. Euristiniai dėstymo algoritmai

Euristinis dėstymo algoritmas tuo pačiu yra ir euristinis optimizavimo algoritmas (angl. *heuristic optimization algorithm*). Todėl pirmiausia pateikiamas euristinių optimizavimo algoritmų aprašymas.

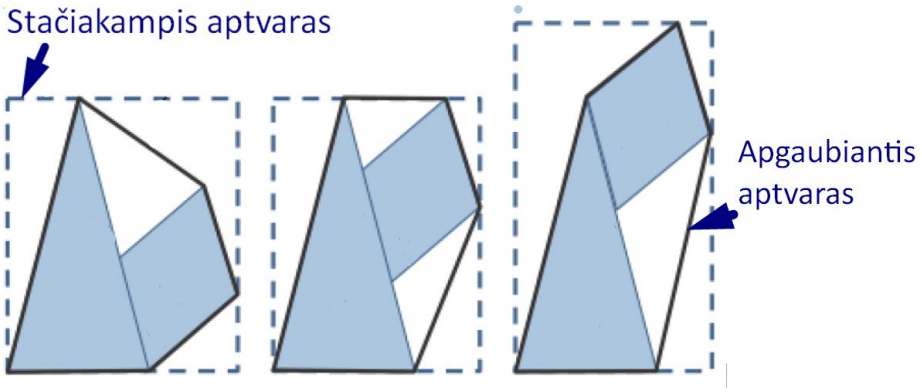
Naudojant euristinius optimizavimo algoritmus siekiama rasti geriausius įmanomus sprendimus sprendžiant optimizavimo uždavinius kai problemos sudėtingumas arba riboti laiko resursai neleidžia tiksliai ir pilnai išspręsti pačios problemos. Skirtingai nuo kitų algoritmų, kuriuos naudojant svarbiausias parametras yra laiko sutaupymas, yra dvi aktualios euristikos vertinimo problemos: kaip greitai sprendimas gali būti pateikiamas ir kiek jis priartėja prie optimalumo. Euristinis algoritmas yra toks, kuris yra skirtas išspręsti problemą greičiau ir efektyviau nei tradiciniai metodai, paaukojant optimalumą, rezultatų tikslumą. Euristinis sprendimo modelis gali būti naudojamas kaip pagrindinis, tačiau dažniausiai šie algoritmai yra papildomi optimizavimo algoritmais (pats euristinio algoritmo sprendimas tik padaro pagrindą tolesniam optimalaus sprendimo ieškojimui) (Ronal L. Rardin ir Reha Uzsoy 2001).

Nors euristiniai algoritmai taikomi įvairiose srityse ir jų veikimas gali būti skirtingai interpretuojamas priklausomai nuo taikymo srities, šiame skyriuje toliau nagrinėjami tik įvairios formos 2D kontūrų dėstymui taikomi euristiniai algoritmai ir jų modifikacijos.

Įvairios formos 2D kontūrų dėstymui euristinių algoritmų veikimo principas apibrėžiamas taip: euristiniai algoritmai atsitiktinai generuodami skirtingas kontūrų padėtis ieško optimaliausio rezultato. Šio tipo algoritmai dažniausiai kontūrus selektyviai suporuoja arba sugrupuoja naudojant pavertimus, įterpimą ir pasukimus. Šis procesas vadinamas suporavimu arba sugrupavimu (angl. *pairwising or clustering*) (Dumitrescu, 1998). Tačiau daugiau nei dviejų kontūrų grupavimas yra sudėtingesnis, nes grupavimo tvarka turi esminę įtaką sprendimo kokybei (galutiniam rezultatui). Todėl šiam atvejui yra taikomas supaprastintos euristikos algoritmas – sutelkiamas dėmesys tik į apribotas kontūrų grupių dėstymo galimybes (Cheng ir Rao 2000).

Esant situacijai, kai figūrų kontūrai yra išgaubti daugiakampiai, euristinio algoritmo dėstymo uždavinys – surasti minimalaus (mažiausio) ploto išgaubtą aptvarą, kuriame yra abu figūrų kontūrai. Išgaubtų daugiakampių atveju egzistuoja ribotas skaičius padėčių, todėl naudojantis šia aplinkybe algoritmas nuosekliai skaičiuoja ir tikrina kiekvieną galimą padėtį. Vienas figūros kontūras yra apvedamas aplink kitos figūros kontūro kiekvieną iš kraštinių, išbandomos visos galimos padėtys. Vykstant šiam procesui, kiekvienai padėčiai yra suformuojamas daugiakampis kuris apgaubia figūrų kontūrus. Optimali pozicija nustatoma kai

šio daugiakampio plotas yra mažiausias. Tačiau šis būdas veikia tik esant išgaubto daugiakampio formos figūrų kontūrams.

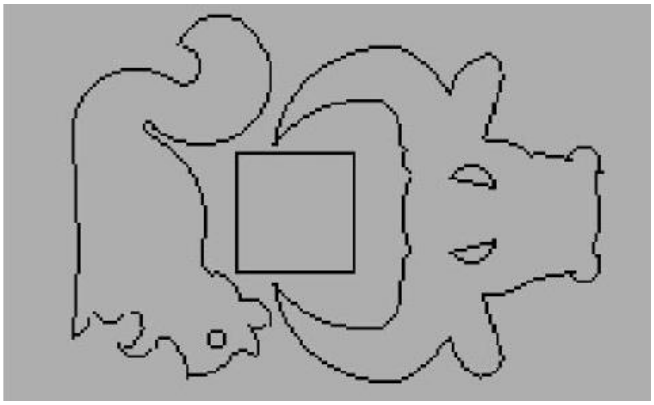


2.9 pav. Kontūro slinkimas apie kito kontūro kraštinę

Dėstant pasikartojančius kontūrus galima juos sugrupuoti poromis – bus gaunami atskiri moduliai (kiekvienai porai). Toliau šiems moduliams dėstyti vėl galima taikyti euristinį arba stačiakampį aptvaro algoritmą (jeigu modulis yra stačiakampio formos).

2.4.1. Euristiniai paieškos algoritmai

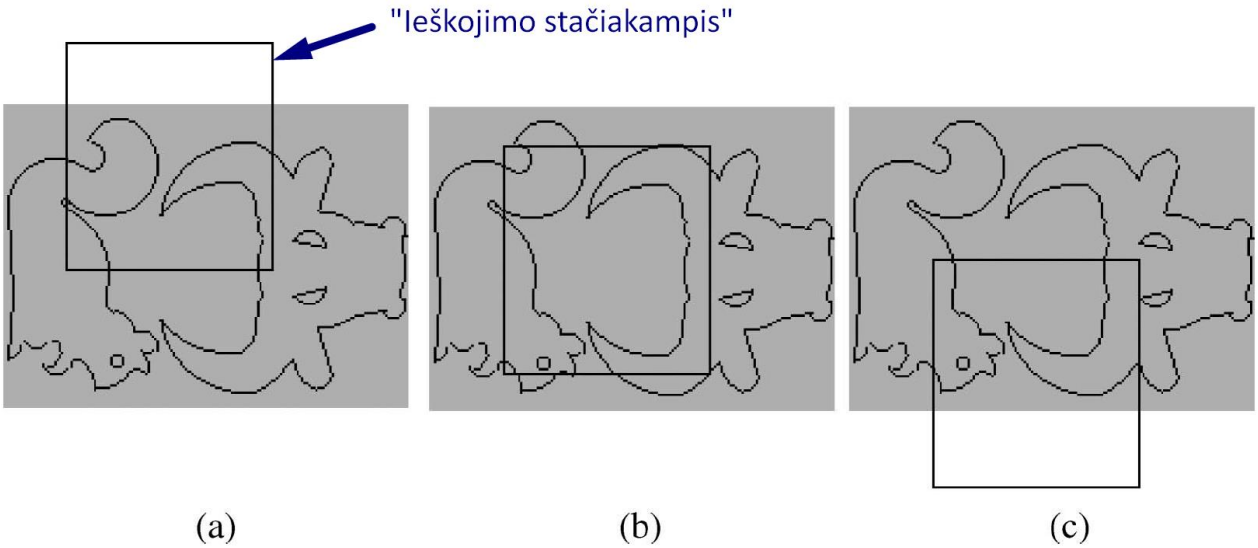
Euristinis paieškos algoritmas skirtas ieškoti laisvų erdvių tarp lakšte esančių kontūrų. Tai yra svarbu norint sutaupyti lakšto medžiagos resursų. Surastos laisvos vietos vėliau gali būti užpildomos kontūrais, kurių matmenys tilptų į jas. Žemiau esančiame 2.10 paveikslėlyje pavaizduotas stačiakampis, kuris yra įterptas tarp dviejų figūrų. Šio stačiakampio patalpinimas ir yra euristinio paieškos algoritmo veikimo rezultatas.



2.10 pav. Euristinio paieškos algoritmo surasta laisva erdvė tarp figūrų

Euristinės paieškos tikslas yra surasti ir išsaugoti stačiakampio formos aptvarą taip, kad laisvos erdvės tarp kontūrų būtų “aptvertos”.

Tam, kad būtų panaudotas euristinis paieškos algoritmas, pirmiausia turi būti atliktas šis veiksmas: surandamas ir apskaičiuojamas “ieškojimo stačiakampis”, kuris bus skirtas laisvos erdvės tarp figūrų ieškojimui. Jo dydis yra lygus didžiausio ploto figūros kontūro stačiakampiam aptvarui.



2.11 pav. „Ieškojimo stačiakampio“ skirtingos padėtys

Toliau algoritmas atlieka tokius proceso žingsnius:

1. Pirmiausia atsitiktiniu būdu parenkamas pirminis taškas, nuo kurio bus pradedama laisvos erdvės tarp kontūrų paieška. “Ieškančiojo stačiakampio” kampas sulyginamas su šiuo tašku.
2. Ieškoma ar padėtame stačiakampyje yra laisva erdvė tarp kontūrų. Laisva erdvė tarp figūrų randama pagal šią išraišką (angl. *boolean calculation*):

$$SpaceRgn = RectRgn - UsedRgn \quad (2.1)$$

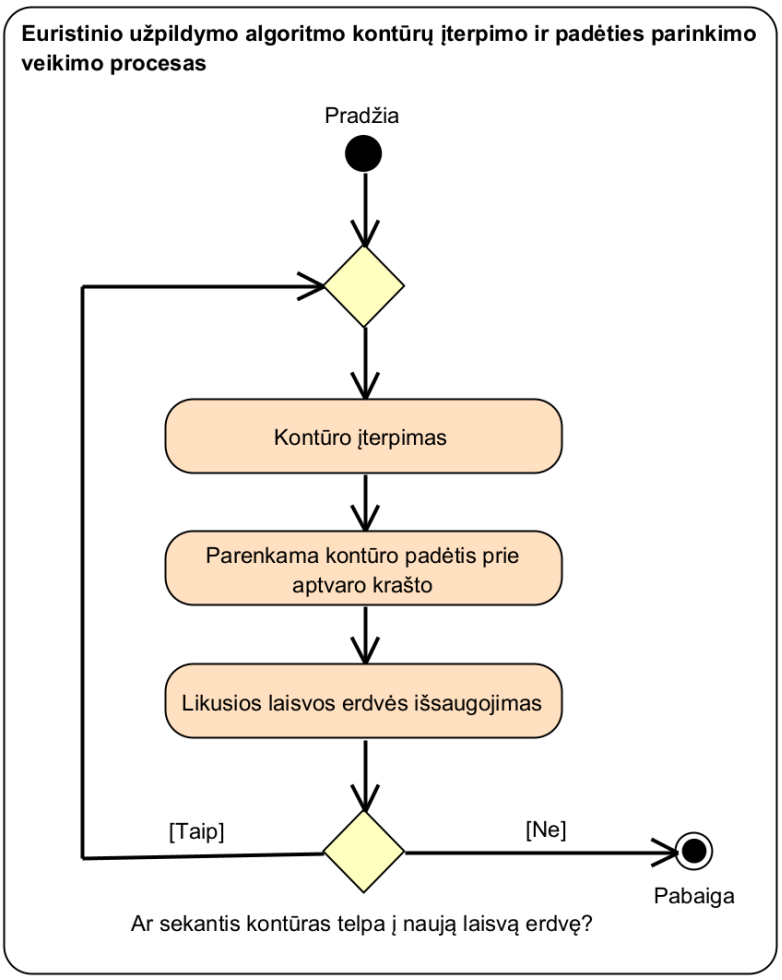
   čia: $UsedRgn$ – patalpintų kontūrų sulietas sluoksnis, kurį sudaro tik kontūrų tūris (jame nėra jokios laisvos erdvės),
   $RectRgn$ – „ieškančiojo stačiakampio“ erdvė,
   $SpaceRgn$ – laisva erdvė, joje nėra kontūrų.
3. Algoritmas perkelia aptvarą į kitą vietą ir atlieka laisvos erdvės paiešką.
4. Jeigu algoritmas suranda laisvų erdvių, į kurias gali būti patalpinti kontūrai, sumažinamas “ieškantysis stačiakampis” iki tokio dydžio, kada jis visas telpa į surastą laisvą erdvę tarp kontūrų. Į šį naują aptvarą įterpiamas naujas kontūras ir sukant (angl. *rotating*) parenkama jo padėtis.

2.4.2. Euristiniai užpildymo algoritmai

Euristiniai paieškos algoritmai leidžia į surastas laisvas erdves įterpti daugiau negu vieną kontūrą viename erdvės plote. Tam, kad kontūrų įterpimas būtų kiek įmanoma efektyvesnis – leistų sutaupyti kuo daugiau šios erdvės, panaudojami euristiniai užpildymo algoritmai. Šie algoritmai turi būti realizuojami kartu su dėstymo apačioje kairėje algoritmais (žr. 2.3. poskyrį).

Euristinių paieškos ir užpildymo ir dėstymo apačioje kairėje algoritmų kombinacijos veikimo proceso žingsniai:

1. Išsaugomas laiko momentas, kai tik laisvos erdvės tarp kontūrų plotas susilygina su pageidaujamų įterpti kontūrų plotu.
2. Sustabdomas dėstymo apačioje kairėje algoritmo veikimas ir į laisvas erdves (stačiakampius, kurie gauti kaip euristinio paieškos algoritmo rezultatas) atsitiktiniu būdu įterpiami kontūrai.
3. Algoritmas kiekvienam iš įterptų kontūrų parenka padėtį. Padėtis parenkama iš eilės imant po vieną kontūrą. Pirmojo kontūro padėtis ir tinkamas pasukimo kampas parenkami prie pat aptvaro krašto. Sekantys įterpiami kontūrai atitinkamai pozicionuojami prie aptvaro kraštų ir/ar kitų kontūrų.
4. Algoritmas išsaugo likusios laisvos erdvės ribas (pirminis aptvaro plotas minus įterptas kontūro plotas).
5. Panaudojant euristiką naujas kontūras įterpiamas į šią laisvą erdvę ir atitinkamai pasukamas.
6. Kartojamas 4 žingsnis – išsaugoma likusi laisva erdvė.
7. Kartojamas 5 žingsnis – į likusią laisvą erdvę vėl įterpiamas naujas kontūras ir atitinkamai pasukamas (3, 4 ir 5 proceso žingsniai pateikti žemiau esančioje proceso diagramoje).
8. Generuojami ir išsaugomi skirtingi dėstymo padėčių maketai tam, kad būtų galima palyginti laisvos erdvės panaudojimą užpildant ją kontūrais.



2.12 pav. Kontūro įterpimo į laisvą erdvę ir padėties parinkimo naudojant euristinį užpildymo algoritmą proceso diagrama

Euristinio užpildymo algoritmo veikimas išsiskiria iš kitų kontūrų dėstymo algoritmų savo veikimo greičiu, kuris yra labai didelis. Šis algoritmas gali būti pritaikomas įvairiuose (kontūrų dėstymo stačiakampiuose ar netgi neapibrėžtos formos lakštuose) uždaviniuose (International Journal of Computer Integrated Manufacturing, 2007).

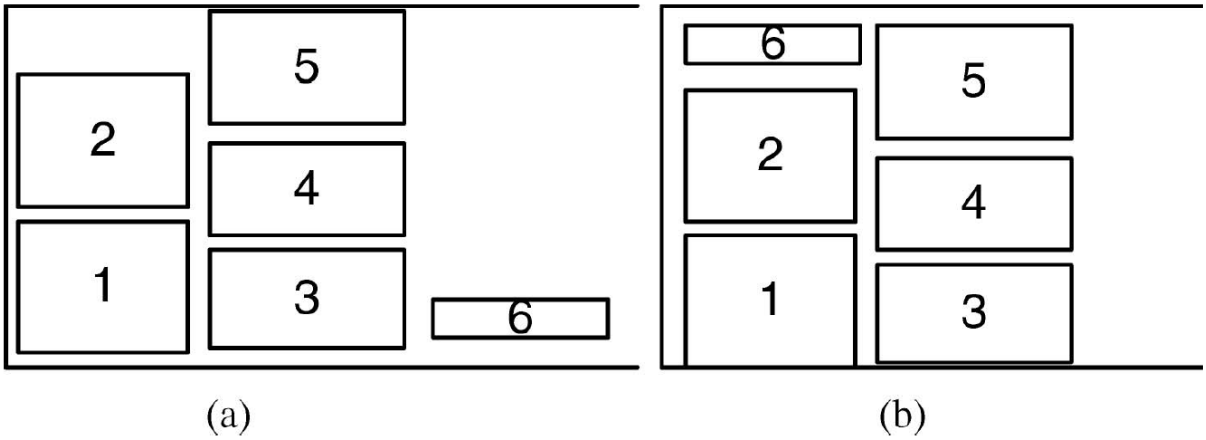
2.5. Atsitiktinio dėstymo algoritmai

Atsitiktinio dėstymo algoritmo veikimo principas: atsitiktine tvarka suteikiami numeriai kontūrams, pagal kuriuos kontūrai išdėstomi lakšte (arba aptvare) ir užregistruojamas (išsaugomas) kiekvienas dėstymo variantas (angl. *layout*) su paskaičiuotu naudingumo koeficientu (International Journal of Computer Integrated Manufacturing, 2007).

Kontūrai lakšte (arba aptvare) dėstomi eilės tvarka atsižvelgiant ar įterpiamas kontūras neuždengia prieš tai padėto. Jeigu kontūrai persidengia – algoritmas įterpiamą kontūrą

perstumia tiek kartų, kol kontūras visas padedamas į laisvą vietą. Perstumiant kontūrą algoritmas pirmenybę jo naujai pozicijai parenka kuo arčiau kontūrų masės centro. Šis perstūmimo būdas užtikrina, kad kontūrai nepersidengs, tačiau gali susiliesti kraštais (jeigu to nedraudžia pirminė užduotis). Visi lakšte (arba aptvare) esantys kontūrai po aprašyto dėstymo proceso ir sudaro aktualų dėstymo maketą – variantą. Tuomet kiekvienam dėstymo variantui paskaičiuojamas kontūrų dėstymo naudingumo koeficientas. Po šio veiksmo algoritmas palygina paskutinio dėstymo varianto naudingumo koeficientą su prieš tai paskaičiuotais ir eliminuoja mažiausią vertę turinčius. Toliau kartojamas visas procesas kol atrenkamas optimaliausias dėstymo variantas pagal didžiausią naudingumo koeficientą.

Jeigu kontūrų yra daug, dėstymo variantų gali būti labai didelis skaičius, todėl skaičiavimo procesas užtruktų nemažą kiekį laiko kol būtų išbandyti visi. Tokiu atveju, gali būti nustatomos naudingumo koeficento ribos, kurias atitikus algoritmo procesas sustotų suradęs į ribas patenkantį dėstymo maketą.



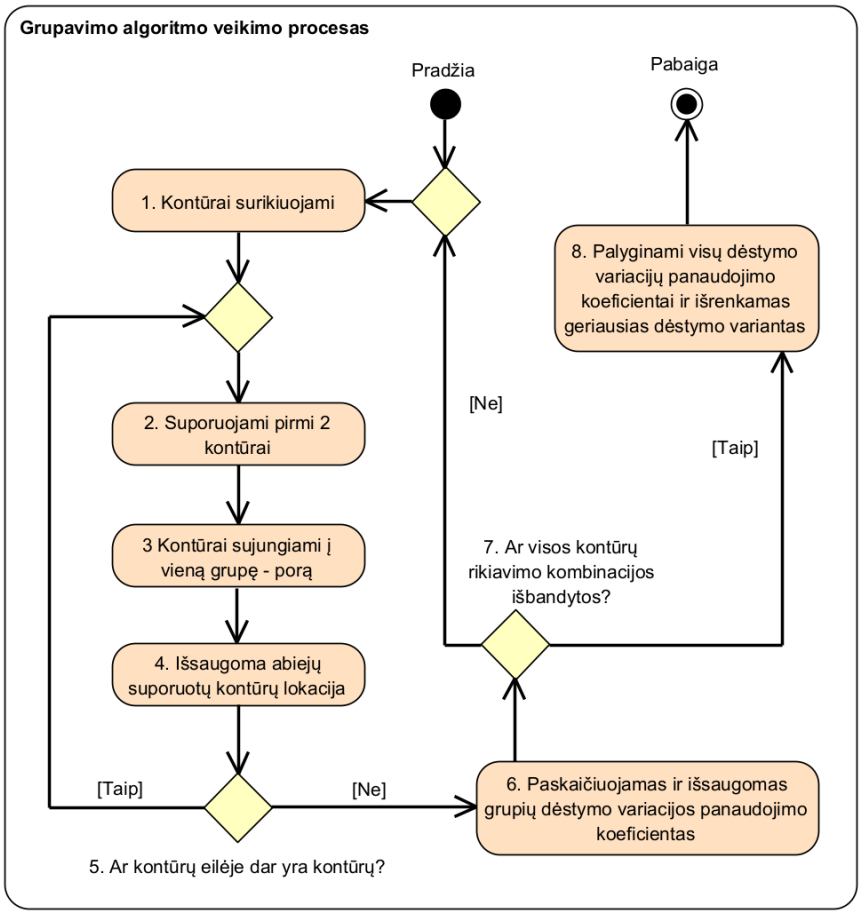
2.13 Dėstymo apačioje kairėje algoritmo ir jo kombinacijos su atsitiktinio dėstymo algoritmu sprendimo rezultatai

Atsitiktinio dėstymo algoritmas dažnai yra kombinuojamas kartu su dėstymo apačioje kairėje algoritmu (žr. 2.3. poskyrį). Aukščiau esančiame 2.13 paveikslėlyje pavaizduoti 6 kontūrai, kurie pirmiausia buvo išdėstyti pagal dėstymo apačioje kairėje algoritmą. (a) dalyje pirmi 5 kontūrai išdėstyti kolonoje, o 6 kontūras yra įterpiamas, todėl tradicinis dėstymo apačioje kairėje algoritmas jį pozicionuoja į naują koloną. (b) dalyje pavaizduotas dėstymo variantas, kuris buvo parinktas panaudojant atsitiktinio dėstymo algoritmą – surastas optimalus dėstymo variantas pagal naudingumo koeficientą iš galimų. Todėl 6 kontūras yra įterpiamas į laisvą tarpą tarp lakšto aptvaro ir pirmos kolonos kontūrų.

2.6. Grupavimo algoritmas

Gamybos procesuose labai dažnai kontūrai turi skirtingas formas. Tam, kad optimaliai dėstyti tokius kontūrus, skaičiavimo procesuose yra pritaikomas grupavimo algoritmas. Tačiau šis algoritmas efektyvus tik tuomet kai kontūrai turi nedaug skirtingų formų variantų ir jų dydžiai yra panašūs. Apibendrinus – grupavimo (arba suporavimo) algoritmas sprendžia kolekcionuojamų kontūrų dėstymo problemas (Cheng and Rao 2000).

Grupavimo algoritmo veikimo principas: siekiant sumažinti skirtingų formų kontūrų dėstymo užduoties kompleksiškumą, dalis kontūrų yra apjungiami į grupes (apjungiant kontūrus po 2 tai vadinama suporavimu), o šios grupės laikomos kaip identiški kontūrai palyginus su kitomis grupėmis ir atitinkamai toliau dalyvauja dėstymo procese. Šį procesą galima padaryti efektyvesniu jeigu algoritmas bus patobulintas taip, kad kontūrus pasuktų apie savo ašį. Grupavimo algoritmas visais atvejais yra kombinuojamas su euristiniu algoritmu (žr. 2.4. poskyrį), kuris ir atlieka kontūrų sujungimo į grupes veiksmus. Tačiau euristinis algoritmas produktyvesnis kai reikia sujungti daugiau negu 2 kontūrus į grupę. Euristinis algoritmas nesunkiai suranda geriausią kontūrų dėstymo grupėje variantą.

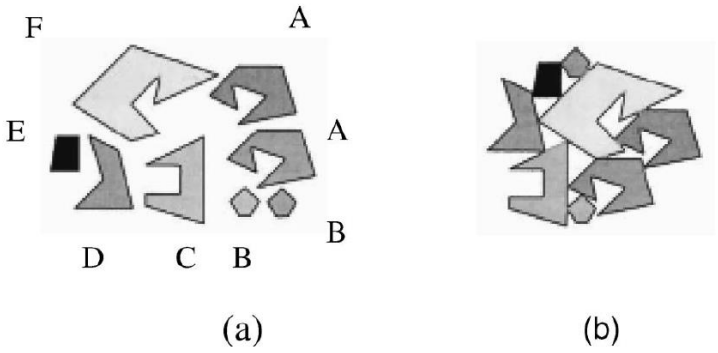


2.14 pav. Grupavimo ir euristinio algoritmų kombinacijų veikimo proceso diagrama

Tam, kad būtų surastas optimaliausias kontūrų grupės dėstymo variantas, 7 ir 8 proceso žingsniai turi būti vykdomi atsižvelgiant į šias 2 sąlygas:

1. Kontūrų grupės forma turi būti kuo panašesnė į stačiakampį.
2. Kontūrų grupę apgaubiantis aptvaro plotas turi būti kuo mažesnis.

Aukščiau esančiame 2.14 paveikslėlyje aprašyto algoritmo veikimo proceso skaičiavimo rezultatas yra pateiktas žemiau esančiame 2.15 paveikslėlyje. (a) dalyje matomi skirtingi kontūrai, o (b) dalyje kontūrai sujungti į grupę, kurios dėstymo naudingumo koeficientas yra didžiausias (International Journal of Computer Integrated Manufacturing, 2007).



2.15 pav. Grupavimo ir euristinio algoritmų kombinacijų sprendimas – kontūrai sujungti į kontūrų grupę

Kolekcionuojamų kontūrų dėstymo naudingumo koeficientas skaičiuojamas realiu laiku (6 proceso žingsnis). Taip daroma dėl to, kad būtų iškart (8-ame žingsnyje) surandamas didžiausią naudingumo koeficientą turintis dėstymo variantas.

2.7. Laisvos vietos ieškojimo algoritmai

Talpinant figūros kontūrą į laisvą erdvę tarp kitų kontūrų arba į kontūro viduje esančią laisvą erdvę svarbiausi procesiniai žingsniai yra šie: laisvos erdvės suradimas ir jos užpildymas figūros kontūru. Tai ir yra laisvos vietos ieškojimo algoritmo pagrindiniai uždaviniai. Kai surandama laisva erdvė, sekančiame žingsnyje darbą atlieka kitų tipų algoritmai, jų modeliai. Tačiau šio tipo algoritmai rečiau naudojami įvairios formos 2D kontūrų dėstymo praktikoje, nes esant netaisyklingų formų kontūrams yra pakankamai sunku tiksliai surasti laisvas erdves.

Laisvos vietos ieškojimo algoritmo pranašumas – jo tikslus laisvos vietos suradimas esant daugiakampėms figūroms.

2.8. Hibridiniai algoritmai

Jau aprašyti įvairios formos 2D kontūrų dėstymo algoritmai dažniausiai sprendžia vieną ar kelias kontūrų dėstymo problemas, tačiau negali pateikti rezultato, kuris apimtų visą įvairios formos 2D kontūrų optimalaus išdėstymo užduotį. Šiam uždaviniui išsamiai išspręsti reikalinga sujungti kiekvieno iš aprašytų algoritmų proceso dalis ir konkrečiam atvejui pritaikyti šias kombinacijas sujungiant jas į hibridinį algoritmą. Hibridinį dėstymo algoritmą sudaro kelių skirtingų tipų 2D kontūrų dėstymo algoritmų junginys (Alves et al. 1999 ir Lutfiyya et al. 1992, Dufour et al. 1997).

2.9. Trumpas skyriaus apibendrinimas

Įvairios formos 2D kontūrų optimalaus dėstymo uždavinys yra komplikuotas, nes šio uždavinio sprendimas apima kelis skirtingus etapus (kontūrų įterpimas, laisvos vietos ieškojimas, kontūrų grupavimas, dėstymas, optimalaus varianto suradimas). Dėl šios priežasties nei vienas iš aprašytų algoritmo tipų pats savaime neišsprendžia viso uždavinio. Visam uždaviniui išspręsti reikalinga sujungti kelis algoritmus ar jų dalis.

Beveik visi algoritmai, skirti spręsti įvairios formos 2D kontūrų optimalaus dėstymo uždavinį, turi euristinius sprendimo elementus: generuojamos kontūrų padėtys siekiant per kuo trumpesnį laiką gauti pradinius dėstymo rezultatus. Vėliau šie gauti rezultatai gali būti analizuojami ir iš jų išrenkamas optimaliausias variantas. Visa tai atliekama pritaikius matematinius skaičiavimus. Tačiau toks sprendimo būdas užima nemažai laiko resursų ir yra netinkamas sąlygose kai reikalingas skubus rezultatų pateikimas.

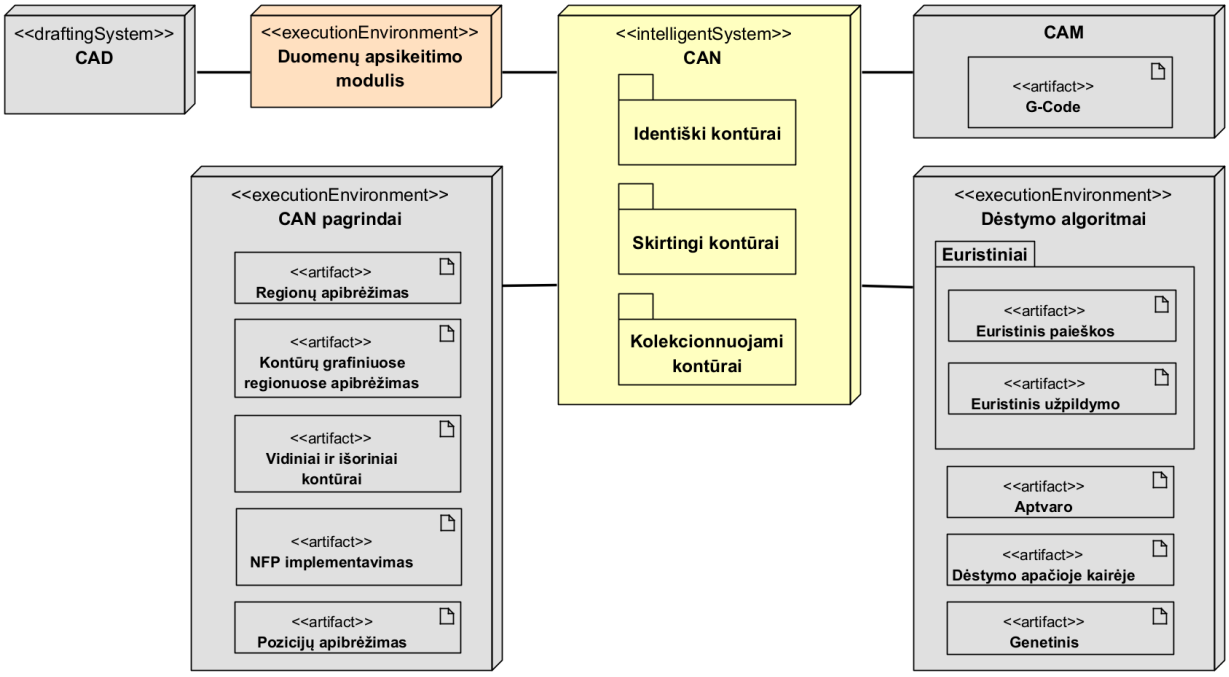
Kontūrų padėties parinkimo uždaviniams spręsti didžiausią pranašumą turi euristiniai algoritmai, nes jų veikimo principas pritaikytas kontūrų dėstymui, pasukimui ir perkėlimui. Taip pat naudojant šiuolaikinių kompiuterių spartos resursus išvystomas didelis greitis atliekant skaičiavimus. Dėl šios priežasties euristiniai algoritmai turi galimybes pasiekti optimalius praktinius rezultatus.

3. ĮVAIRIOS FORMOS 2D KONTŪRŲ DĖSTYMO ALGORITMŲ TYRIMAS

Šiame skyriuje pateikiami įvairios formos 2D kontūrų optimalaus dėstymo stačiakampiame lakšte algoritmų realizavimo skirtingose programinėse aplinkose metodai, užduočių sprendimo būdai ir rezultatai. Realizuojami ir tiriami aptvaro, dėstymo apačioje kairėje, euristiniai ir atsitiktinio dėstymo algoritmai bei jų kombinacijos sprendžiant sudėtingas įvairios formos kontūrų dėstymo užduotis.

3.1. Algoritmų integravimas CAN sistemoje

CAN sistema (angl. *the intelligent CAN system*) skirta skirtingų įvairios formos 2D kontūrų dėstymo algoritmų apjungimui ir dėstymo uždavinių sprendimui. Šios sistemos veikimo principas: sukuriama aplinka CAD (angl. *computer-aided design*) brėžinių grafinės informacijos transformavimui į CAN, kurioje yra pritaikomi dėstymo algoritmai ir išsprendžiami įvairios formos 2D kontūrų uždaviniai. Toliau sprendimo rezultatai paverčiami į G-Codes[1] pavidalą ir perduodami CAM (angl. *computer-aided manufacturing*) sistemoms. Tokios sistemos struktūrinė diagrama pateikta žemiau esančiame 3.1 paveikslėlyje.



3.1 pav. CAN sistemos struktūrinė diagrama

[1] G-Code – plačiausiai naudojama kompiuterių skaitmeninio valdymo CNC (angl. computer numeric control) programavimo kalba. G-Codes daugiausia naudojami kompiuteriuose skirtuose automatinėms staklėms valdyti.

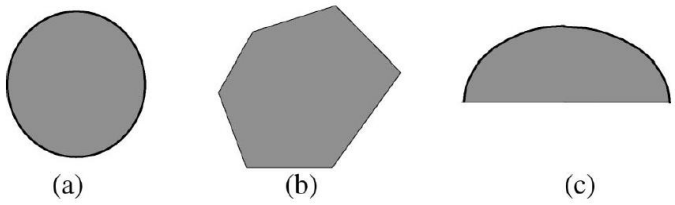
CAN sistema skirta efektyviai integruoti dėstymo algoritmus tam, kad būtų optimaliai naudojami lakšto medžiagos resursai. Apibendrinus – šią sistemą sudaro 3 pagrindinės dalys:

1. CAN pagrindai – skirti palaikyti algoritmų integravimo ir veikimo procesus.
2. Dėstymo algoritmai – pagrindas spręsti įvairios formos kontūrų dėstymo uždaviniams.
3. CAN integravimas su CAD ir CAM sistemomis. Tai vartotojo sąsajos ir duomenų apsikeitimo modulių tarp paminėtų sistemų realizacija.

Sprendžiant kontūrų dėstymo uždavinius, geometrinio dėstymo problematika yra pati svarbiausia, nes šiuose uždaviniuose dalyvauja skirtingų formų kontūrai. Kontūrų formų neįmanoma iš anksto tiksliai apibrėžti. Dėstymo problemoms spręsti nepakanka turėti tik patį dėstymo algoritmą. Reikalingi papildomi sistemos technologiniai moduliai, kurie palaiko ir geometrinių įgyvendinimo užduočių bei dėstymo algoritmų integravimą tarpusavyje (International Journal of Computer Integrated Manufacturing, 2007). Šie moduliai yra CAN sistemos pagrindas (žr. aukščiau esantį paveikslėlį). Toliau šiame skyriuje tiriami CAN sistemos pagrindai, geometrinių skaičiavimų technologiniai moduliai.

3.1.1. Grafiniai regionai

Geometriniuose skaičiavimuose ir modeliavime kontūrai pagal savo formą apibrėžiami grafiniais regionais: (a) apskritiminis, (b) daugiakampis, (c) lankas. Šie pagrindiniai regionai modeliuojami naudojant objektiškai orientuotą programavimo metodiką. Žemiau esančiame 3.2 paveikslėlyje pavaizduoti skirtingų formų kontūrų grafinis skirstymas pagal regionus.



3.2 pav. Kontūrų pagrindiniai grafiniai regionai

Grafinių regionų apibrėžimas ir modeliavimas reikalingas tam, kad dėstymo algoritmai turėtų medžiagą (parametrus, reikšmes, duomenis), su kuria bus atliekami skaičiavimai reikalingi išspręsti konkrečius kontūrų dėstymo uždavinius. Todėl grafinių regionų modulis yra tarsi pagrindas 2D kontūrų dėstymo uždaviniams spręsti.

Grafinių regionų moduliai yra naudojami ne tik CAN sistemose, bet ir kitose kontūrų dėstymo sprendimo aplinkose. CAN sistema yra pasirinkta šiame darbe kaip skirtingus įvairių formų kontūrų dėstymo algoritmus apjungianti aplinka, kuri turi sąsajas su CAD ir CAM sistemomis.

Sudėtingų grafinių regionų apibrėžimui kompiuteriai naudoja Boolean[2] operacijas. Žemiau pateikti grafinių regionų apibrėžimai (A ir B yra apibrėžti grafiniai regionai, P yra nežinomas grafinis regionas):

- Regionų sąjunga (sąjungos operacija) (RGN_OR yra žymima – ∪):

$$P = A \cup B = \{P: P \in A \; OR \; P \in B\} \text{ [paveikslėlyje (a)]} \quad (3.1)$$

- Regionų sankirta (RGN_AND yra žymima ∩):

$$P = A \cap B = \{P: P \in A \; AND \; P \in B\} \text{ [paveikslėlyje (b)]} \quad (3.2)$$

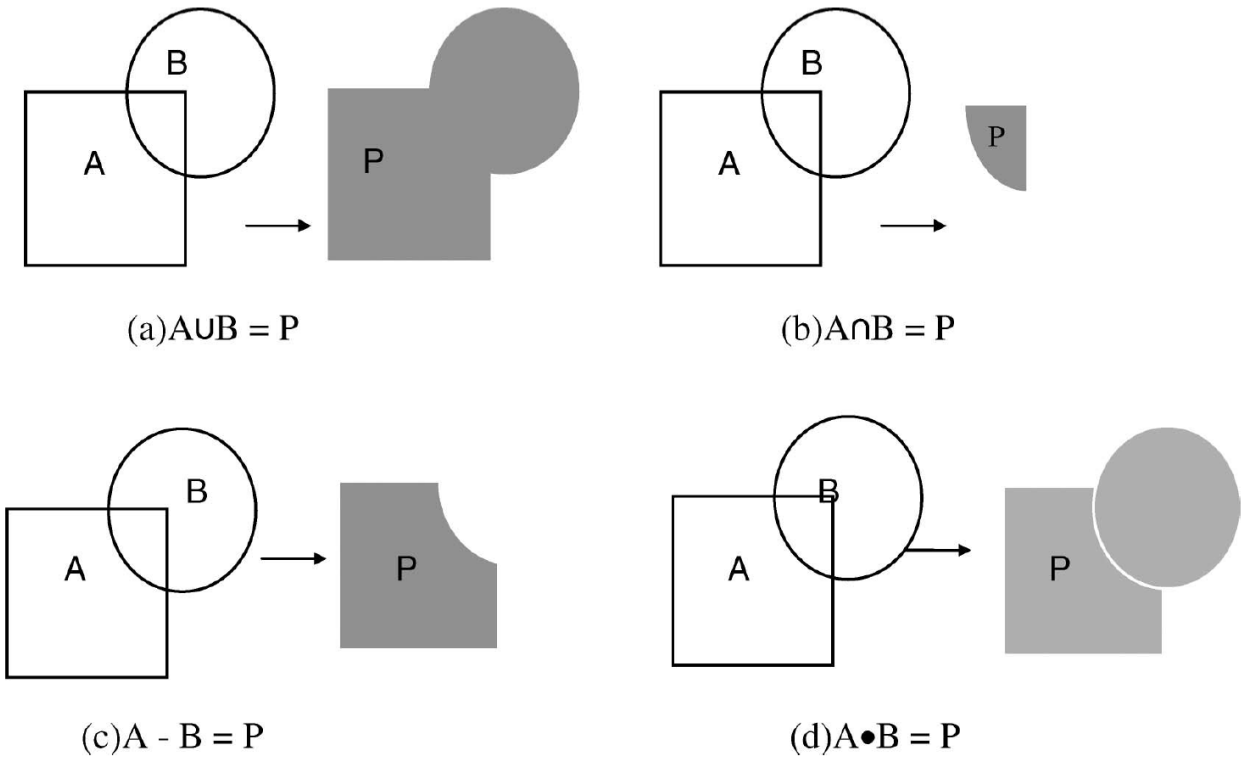
- Regionų skirtumas (RGN_DiFF yra žymima −)

$$P = A - B = \{P: P \in A \; AND \; P \in B\} \quad (3.3)$$

$$P = B - A = \{P: P \in B \; AND \; P \in A\} \text{ [paveikslėlyje (c)]} \quad (3.4)$$

- Loginis ARBA (angl. *OR*) (RGN_XOR yra žymima ●)

$$P = B \cdot A = A \cup B) - (A \cap B) \text{ [paveikslėlyje (d)]} \quad (3.5)$$

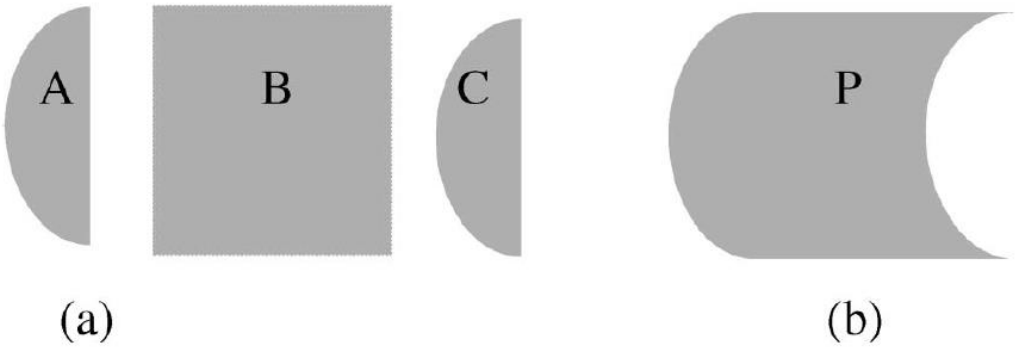


3.3 pav. Grafinių regionų Boolean operacijos

Aukščiau esančiame 3.3 paveikslėlyje pavaizduotas Boolean geometrinių operacijų realizavimas grafinėje aplinkoje (International Journal of Computer Integrated Manufacturing, 2007).

[2] Boolean operations – pagrindinės geometrinės operacijos: sąjunga, sankirta, skirtumas (angl. union, intersection, substract).

Dažniausiai grafinis regionas apima ir išgaubtus ir įgaubtus daugiakampius. Skirtumo geometrinė operacija atliekama įgaubto lanko regione, o sajungos operacija atliekama išgaubto lanko regione. Žemiau pateiktame 3.4 paveikslėlyje parodyta, kaip sudaroma figūra (kontūras) naudojant pagrindines Boolean geometrines operacijas grafiniuose regionuose.



3.4 pav. Skirtingi grafiniai regionai (a); iš A, B ir C grafinių regionių suformuotas vienas grafinis regionas P, panaudojant Boolean geometrines operacijas

Kaip matoma 3.4 paveikslėlyje A regionas yra išgaubtas lankas, todėl jam turi būti naudojama sąjungos geometrinė operacija, o C regionas yra įgaubtas, todėl jam taikoma skirtumo operacija (International Journal of Computer Integrated Manufacturing, 2007).

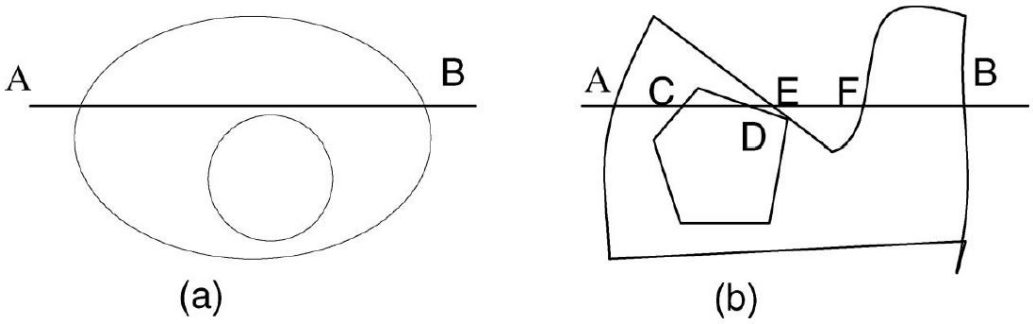
3.1.2. Vidiniai ir išoriniai kontūrai

Kaip jau aprašyta šio darbo 2.4 ir 2.7 poskyriuose – laisvos vietos ieškojimas yra ypatingai svarbus procesas įvarios formos 2D kontūrų dėstymo uždaviniuose. Laisva vieta gali egzistuoti tarp kontūrų arba pačiuose kontūruose (jų viduje). Todėl šiai problematikai tirti yra naudojami vidinių ir išorinių kontūrų apibrėžimai. Sprendžiant kontūrų dėstymo uždavinius yra svarbu identifikuoti laisvas vietas.

Specialus paieškos algoritmas yra realizuojamas surasti laisvas erdves tarp kontūrų ar jų viduje. Jo veikimo procesą galima aprašyti šiais žingsniais:

1. Nubrėžiama linija AB kertanti kontūrą (kontūrus). Laikoma kad linijos pradžia yra taškas A, o pabaiga taškas B.
2. Jeigu ši linija kerta kontūro linijas 2 kartus (3.5 paveikslėlyje (a) dalis), fiksuojama jog laisvos vietos kontūre nėra.
3. Jeigu linija AB kerta kontūro linijas daugiau negu du kartus ir kirtimų skaičius yra nelyginis (3.5 paveikslėlyje (b) dalis) – laikoma, kad AC, DE ir FB tiesės yra kontūro viduje.

4. Jeigu linija AB kerta kontūro linijas daugiau negu du kartus ir kirtimų skaičius yra lyginis – laikoma, kad linijos CD ir EF yra ne šio kontūro regionas.



3.5 pav. Vidinių ir išorinių kontūrų identifikavimas.

Pagal aukščiau aprašyto proceso rezultatus galima nesunkiai surasti, kurie kontūrai yra vidiniai ar išoriniai (International Journal of Computer Integrated Manufacturing, 2007).

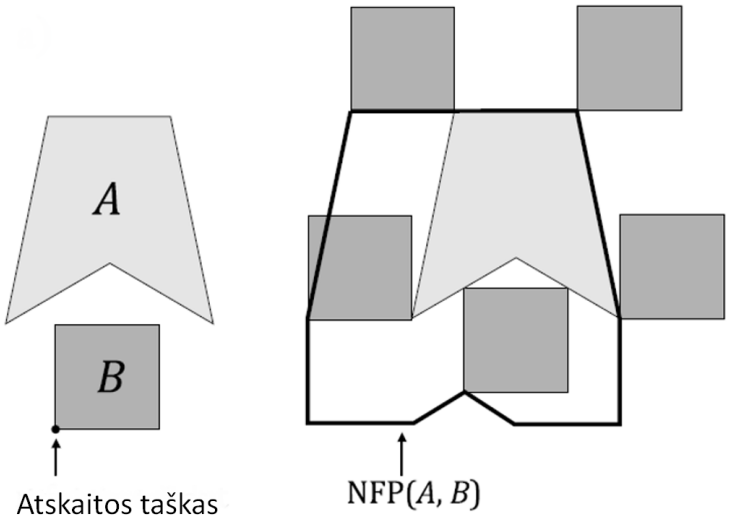
3.1.3. NFP

Kaip jau aprašyta anksčiau šiame darbe, įvairios formos 2D kontūrų optimalaus dėstymo užduotis yra kompleksiška. Ji tokia yra ir dėl to, kad dažniausiai šios užduoties sprendimo rezultatai yra pritaikomi automatinėse pjovimo mašinose tam, kad būtų išpjaunamos detalės. Tokiais atvejais labai svarbu atsižvelgti į galimus pjovimo modelius ir ypač į tai, jog detalės (kontūrai) negali persidengti. Tačiau reikalinga kontūrus pozicionuoti kuo arčiau vienas kito. Šioms sąlygoms tenkinti yra taikoma NFP (angl. *no-fit polygon*) technologija (Kierkosz ir Luczak, 2019).

3.6 paveikslėlyje grafiškai pavaizduotas NFP algoritmo veikimas. Jame A ir B yra daugiakampiai kontūrai. A kontūras įtvirtintas – jis nejuda. B kontūras stumiamas aplink A kontūrą taip, kad jis visada liečia A kontūro kraštinę, tačiau niekada nepersidengia. Parenkamas taškas ant B kontūro – atskaitos taškas. Kai B kontūras yra stumiamas aplink A kontūrą, šis taškas nubrėžia uždarą kontūrą – NFP(A,B). NFP(A,B) kontūras turi šias savybes:

- Jeigu atskaitos taškas yra NFP(A,B) kontūro viduje, kontūrai A ir B persidengia.
- Jeigu atskaitos taškas yra ant NFP(A,B) kontūro, kontūrai A ir B liečiasi.
- Jeigu atskaitos taškas yra NFP(A,B) kontūro išorėje, tarp kontūrų A ir B yra tarpas – jie atskirti.

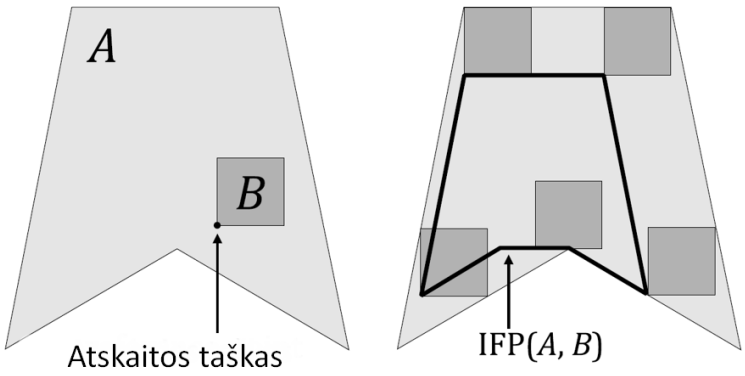
Pagal šias savybes yra paprasta nustatyti ar kontūrai persidengia, liečiasi ar jie yra atskirti.



3.6 pav. NFP pavyzdys

Praktikoje dažniausiai reikalingas rezultatas – kontūrai turi būti kuo arčiau vienas kito arba liestis kraštais. Taip pat yra galimybė modifikavus aprašytą technologiją nustatyti kontūro viduje esančių kontūrų pozicijas – ar viduje esantys kontūrai liečiasi kraštinėmis, ar jie yra už pagrindinio kontūro ribų. Ši modifikacija vadinama IFP (angl. *inner fit polygon*). Žemiau esančiame paveikslėlyje pateikta jos realizacija. Joje yra daugiakampiai kontūrai A ir B. A kontūro viduje yra kontūras B. Norint nustatyti ar šis kontūras liečiasi su A kontūro kraštinėmis pasirenkamas atskaitos taškas ant kontūro B kraštinės ir atitinkamai kaip ir NFP atveju, B kontūras stumiamas A kontūro viduje apie A kontūro kraštinę. Pagal atskaitos taško nubrėžtą liniją IFP(A,B) galima nustatyti tokias savybes:

- Jeigu atskaitos taškas yra IFP(A,B) viduje arba liečiasi su jo kraštine – vidinis kontūras B priklauso išoriniam kontūrui A.
- Jeigu atskaitos taškas yra už IFP(A,B) ribų – kontūrai A ir B persidengia.



3.7 pav. IFP pavyzdys

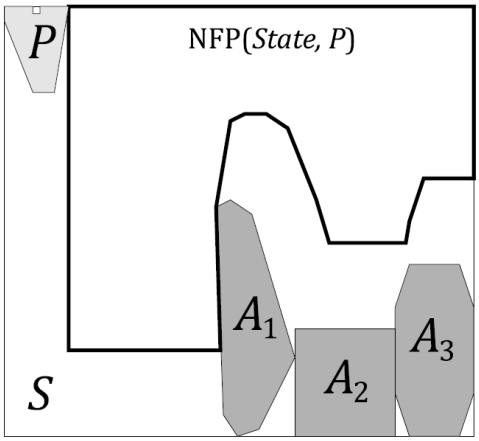
Lakšto su įterptais kontūrais variacijos NFP randamas pagal šią formulę:

$$NFP(State, P) = \frac{IFP(S,P)}{\cup NFP(A,P)} \tag{3.6}$$

čia: $State$ – lakšto su įterptais kontūrais variacija,

S – lakštas,

P, A – kontūrai.



3.8 pav. Lakšto su įterptais kontūrais variacijos NFP kontūras

Naudojant NFP technologiją galima nesunkiai realizuoti pageidaujamus tarpelius tarp kontūrų arba kitas pageidaujamas sąlygas. Svarbu paminėti, jog NFP technologija plačiai naudojama kontūrų dėstymo uždaviniuose – ji nėra CAN sistemos dalis.

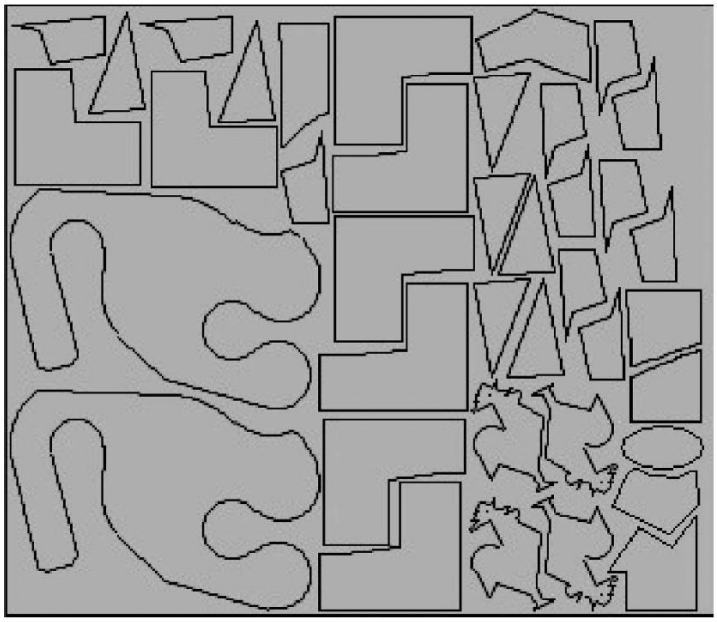
3.1.4. Testavimas CAN sistemoje

CAN sistema tai programinė įranga, kuri buvo specialiai kuriama testuoti ir išbandyti skirtingiems kontūrų dėstymo algoritmams, juos apjungiant. CAN sistemos naudotojo sąsajos vaizdas yra pateiktas 1-ame šio darbo priede. Standartinė sąsaja sukurta taip, kad į CAN sistemą galima integruoti CAD ir CAM sistemų programinius paketus. CAD sistemos tiekia informaciją į CAN sistemą apie brėžinius, o CAM sistema importuoja iš CAN sistemos kontūrų dėstymo rezultatus, kurie po to naudojami automatinėse pjovimo mašinose. CAM sistemoje dėstymo rezultatai automatizuotu būdu papildomi pjovimo keliais, pagal kuriuos mašina vykdo detalių pjovimo procesus (Xie et al. 2001).

Žemiau pateikiami 2 testavimų rezultatai atlikti su CAN sistema.

1. Pirmas pavyzdys.
    - Lakšto plotas 300 x 320 *mm*;
    - Kontūrų kiekis – 38 *vnt*;

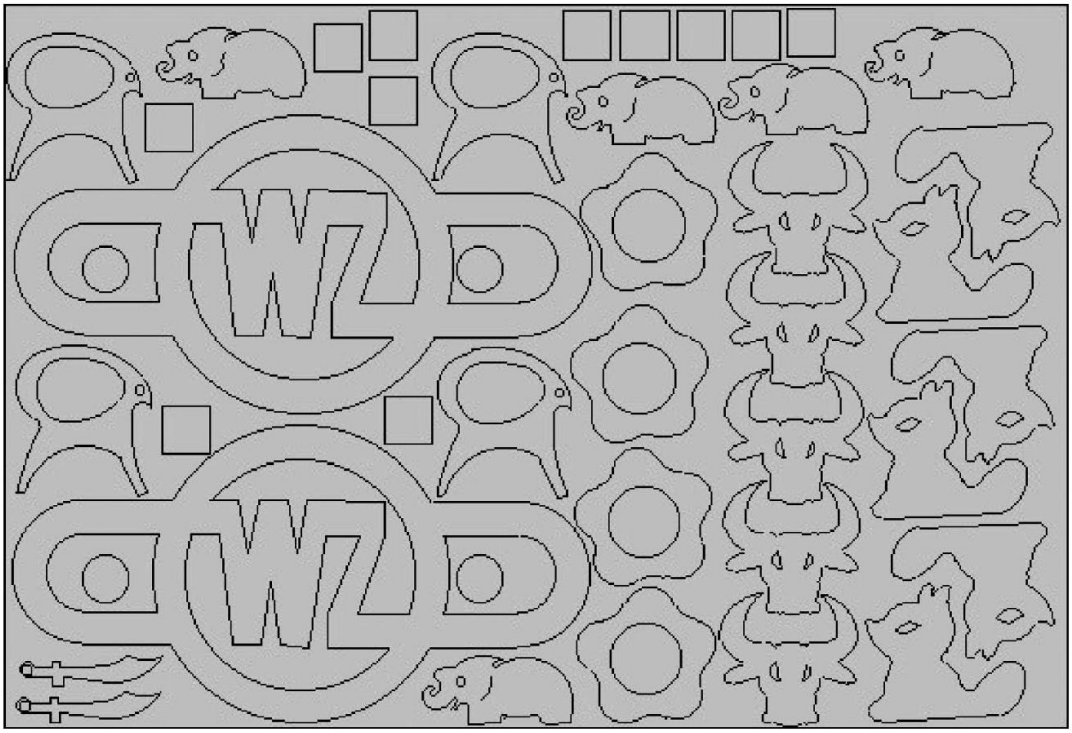
- Tarpelis tarp kontūrų – 5 *mm*.
- Gautas kontūrų dėstymo naudingumo koeficientas – 80,12% (tačiau įmanoma pasiekti didesnį naudingumo koeficientą jeigu užduotyje būtų naudojama daugiau mažesnių kontūrų).



3.9 pav. CAN sistemos kontūrų dėstymo rezultatai (pirmas pavyzdys**)**

3.9 paveikslėlyje matyti jog kontūrus sudaro L formos daugiakampiai, trikampiai, gyvūnų figūros, įvairios formos daugiakampiai. Šie dėstymo rezultatai rodo jog CAN sistema apjungianti kelis skirtingus kontūrų dėstymo algoritmus sugeba efektyviai išspręsti įvairios formos 2D kontūrų dėstymo uždavinius ir pasiekti aukštą kontūrų dėstymo naudingumo koeficientą. Šiame pavyzdyje panaudoti dėstymo apačioje kairėje, euristiniai, grupavimo, atsitiktinio dėstymo algoritmai.

2. Antras pavyzdys.
    - Lakšto plotas 500 x 450 *mm*;
    - Kontūrų kiekis – 43 *vnt*;
    - Tarpelis tarp kontūrų – 5 *mm*.
    - Gautas kontūrų dėstymo naudingumo koeficientas – 71,26% (įmanoma pasiekti didesnį naudingumo koeficientą jeigu užduotyje būtų naudojama daugiau mažesnių kontūrų).



3.10 pav. CAN sistemos kontūrų dėstymo rezultatai (antras pavyzdys)

3.10 paveikslėlyje matyti jog zuikio formos kontūrai buvo suporuoti (naudojant grupavimo algoritmą). Taip pat šiame pavyzdyje buvo naudojami tiek išgaubtų tiek įgaubtų formų kontūrai. Panašaus pločio kontūrai dėstomi kolonomis (naudojant dėstymo apačioje kairėje algoritmą), o didžiausi kontūrai pozicionuojami arčiau apačioje kairėje esančios lakšto kraštinės. Šio pavyzdžio dėstymo rezultatai rodo, jog CAN sistemoje puikiai kombinuojamas laisvos vietos ieškojimo algoritmas.

Aukščiau pateiktų pavyzdžių rezultatai rodo, jog CAN sistema, kombinuodama skirtingus įvairios formos 2D kontūrų dėstymo algoritmus, gali išgauti puikius dėstymo rezultatus. Ši sistema gali surasti ir sugeneruoti optimalius sprendimus kontūrų dėstymo uždaviniams. CAN sistemos yra naudojamos gamybos veiklas vykdančiose organizacijose. Su CAM programinių modulių pagalba gauti dėstymo rezultatai yra konvertuojami į G-codes ir perduodami CNC būdu valdomoms pjovimo mašinoms (Xie et al. 2001).

3.2. Euristinio dėstymo algoritmo tyrimas

Euristinio dėstymo algoritmo tyrimui pasirinktas problemos sprendimas su NFP (žr. 3.1.3. poskyrį) technologijos pritaikymu. Euristinio įvairios formos 2D kontūrų optimalaus dėstymo algoritmo tyrimą sudaro 3 problemų sprendimas:

1. Kontūrų dėstymas viename stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu.
2. Kontūrų dėstymas viename stačiakampiame lakšte be vienodų kontūrų kiekio ribojimo.
3. Kontūrų dėstymas neribojant stačiakampių lakštų skaičiaus su kontūrų kiekio ribojimu.

Įvairios formos 2D kontūrų optimalaus dėstymo stačiakampiame lakšte užduoties tikslas: gauti kuo didesnį išdėstytų kontūrų viename lakšte plotą arba kontūrų dėstymui sunaudoti kuo mažesnį lakštų kiekį. Užduoties tikslas išreikštas techniniu parametru: pasiekti kuo didesnį lakšto medžiagos panaudojimo koeficientą (Kierkosz ir Luczak, 2019).

3.2.1. Algoritmo aprašymas

Testavime naudojamas euristinis algoritmas kartu su NFP technologija, tačiau modifikuojamas taip, kad NFP papildomai įterpiamas į procesą, kurio metu skaičiuojamas likęs laisvos vietos lakšte kiekis po kiekvieno kontūro įterpimo į lakštą (Kierkosz ir Luczak, 2019).

Pradedant nuo tuščio lakšto algoritmas įterpia po vieną kontūrą iš kontūrų sąrašo. Įterptų ir pozicionuotų kontūrų padėtys nekeičiamos. Procesas sustoja kai nebelieka nepadėtų lakšte kontūrų arba kai lakšte nėra vietos nei vieno kontūro sekančiam įterpimui.

Algoritmas naudoja šiuos kintamuosius:

- *State* – lakštas su įterptais kontūrais,
- *Polygon* – kontūras,
- *NFP(State, Polygon)* – *State* NFP su jo NFP *Polygon*. NFP gali būti tuščio kontūro pavidalu arba nesusijusių kontūrų suma (žr. 3.1.3. poskyrį),
- $State_{current}$ – dabartinė lakšto dėstymo variacija,
- $State_{next}$ – sekanti lakšto dėstymo variacija,
- *Area* – paviršiaus plotas,
- $Area_{convex}$ – kontūro paviršiaus plotas,
- *Elems(State)* – kontūrų grupė, kurie gali būti įterptį į *State*,
- *Verts(P)* – kontūro viršūnes,
- *P* – elementas (kontūras),
- *Fit* – įterpimo funkcija.

Kiekviename žingsnyje, priklausomai nuo įterpimo funkcijos reikšmės, algoritmas įterpia kontūrą į lakštą. Pirmiausia įterpimo funkcija įvertina kontūrą ir jo padėtį. Kontūro *P* įterpimo

vieta gali būti bet kuri *NFP(State,P)* viršūnė. Tai reiškia, jog kiekviename algoritmo žingsnyje įterpiamas kontūras, kurio paskaičiuota įterpimo funkcijos reikšmė yra didžiausia su šia kintamųjų pora: kontūru *P* ir įterpimo tašku *V*.

Įterpimo funkcijos reikšmės priklauso nuo dabartinės ir sekančios lakšto su įterptais kontūrais būsenos (*State*):

$$(P,V) \leftarrow \max_{P \in Elems(State)} \max_{V \in Verts(NFP(State,P))} Fit(P,V,State) \tag{3.7}$$

Kiekvieno žingsnio dėstymo skaičiavimo laikas priklauso nuo išdėstytų kontūrų kiekio lakšte ir nuo įterpimo funcijos reikšmės skaičiavimo laiko. Algoritmo skaičiavimo greitis yra didelis dėl to, kad šis algoritmas nekeičia jau anksčiau pozicionuotų kontūrų padėčių. Tačiau dėl tokio veikimo būdo atsiranda rizika, kad bet koks netinkamai padėtas kontūras taip ir liks toje padėtyje be galimybės ją pakeisti. Svarbiausias veiksnys yra įterpimo funkcija. Jeigu įterpimo funkcija tinkamai parenkama pagal užduoties sąlygas – algoritmo veikimas bus ne tik greitas, bet ir lakšto ploto panaudojimo koeficientas bus aukštas.

Aukščiau šiame poskyryje aprašytą algoritmą galima tiesiogiai taikyti šioms problemoms spręsti:

- Kontūrų dėstymas viename stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu.
- Kontūrų dėstymas viename stačiakampiame lakšte be vienodų kontūrų kiekio ribojimo.

Kontūrų dėstymo neribojant stačiakampių lakštų kiekio su vienodų kontūrų kiekio ribojimu problemai spręsti algoritmas taikomas kiekvienam lakštui atskirai. Pirmiausia kontūrai išdėstomi pirmame lakšte (panaudojant aukščiau aprašytą algoritmą), toliau eilės tvarka dėstomi sekančiuose lakštuose iki kol visi kontūrai bus išdėstyti lakštuose. Papildomai galima modifkuoti algoritmą taip, kad kiekviename lakšte kontūrų išdėstymas būtų vienodas (jeigu tai leidžia sąlygos – toks pat kontūrų rinkinys). Tokiu būdu algoritmo veikimo greitis ženkliai padidės.

3.2.2. Įterpimo funkcijų aprašymas

Įterpimo funkcijų reikšmės priklauso nuo NFP ploto. Jos skaičiuoja ir vertina likusį laisvą plotą lakšte (vertinamas kiekvienas kontūras, jo įterpimo vieta). Tyrime algoritmai testuojami naudojant 8 skirtingas įterpimo funkcijas. Euristinio dėstymo algoritmo įterpimo funkcijos naudoja šiuos kintamuosius:

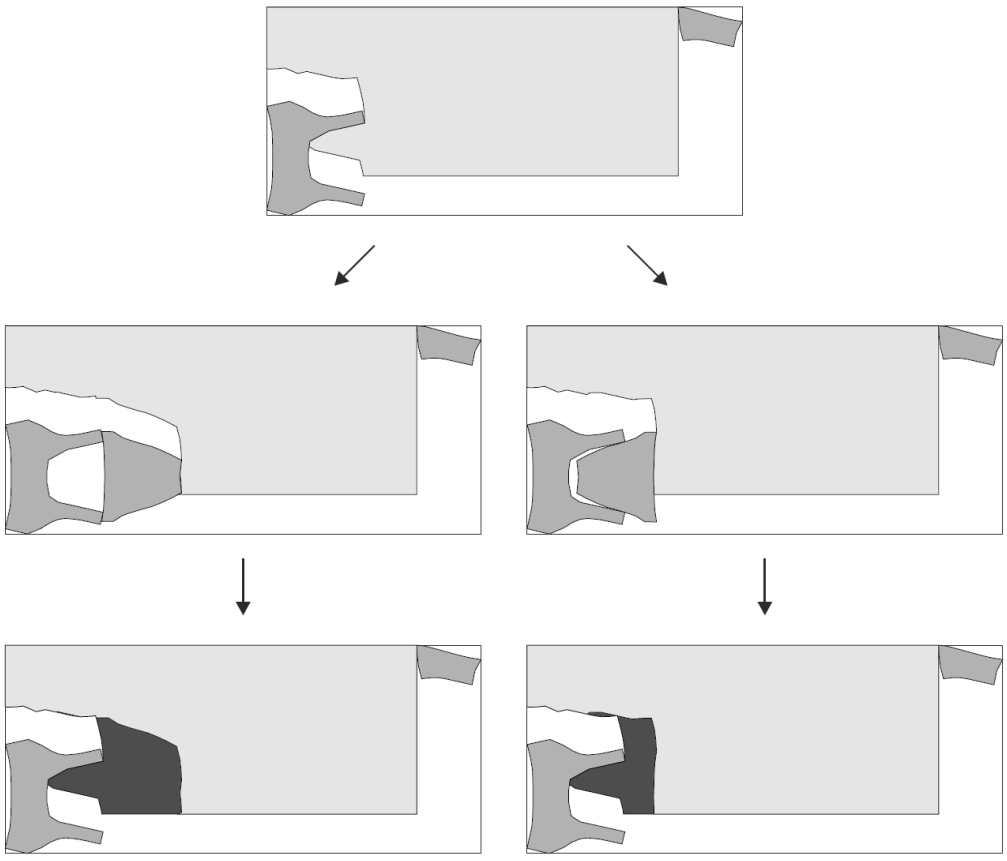
$$sumAreaNFP(State) = \sum_{P \in Elems} Area(NFP(State, P)) \quad (3.8)$$

$$maxAreaNFP(State) = \max_{P \in Elems} Area(NFP(State, P)) \quad (3.9)$$

Algoritme naudojamos įterpimo funkcijos:

1. *Opt1*. – įterpimo funkcijos reikšmė priklauso nuo kontūro ploto ir nuo NFP plotų sumos, kuri apskaičiuojama kontūro pozicionavimo konkrečioje padėtyje atveju.

$$Fit = \frac{Area(Polygon)}{sumAreaNFP(State_{current}) - sumAreaNFP(State_{next})} \quad (3.10)$$



3.11 pav. NFP ploto sumažėjimas priklausomai nuo sekančio kontūro įterpimo vietos. Tamsus plotas rodo NFP ploto sumažėjimą

3.11 paveikslėlyje matoma NFP ploto sumažėjimas. Paveikslėlio dešinėje pusėje matoma, jog kontūrą pasukant gaunamas optimalesnis NFP ploto panaudojimas – NFP sekančioje dėstymo variacijoje turės didesnį plotą.

2. *Opt2.* – įterpimo funkcija panaši į *Opt1.*, tačiau *Area(Polygon)* (kontūro plotas) pakeliamas kvadratu. Ši įterpimo funkcija labiau tinkama didesniems kontūrams.

$$Fit = \frac{[Area(Polygon)]^2}{sumAreaNFP(State_{current}) - sumAreaNFP(State_{next})} \tag{3.11}$$

3. *Opt1.5.* – įterpimo funkcija panaši į *Opt1.*, tačiau skaitiklyje vertinamas išgaubto daugiakampio plotas, o ne bet kokio kontūro plotas. Ši įterpimo funkcija naudojama kai reikia rasti skirtumą, kuris aiškiau nurodytų NFP ploto pokyčius esant panašiems kontūrams.

$$Fit = \frac{Area_{convex}(Polygon)}{sumAreaNFP(State_{current}) - sumAreaNFP(State_{next})} \tag{3.12}$$

4. *Opt2.5.* – įterpimo funkcija panaši į *Opt1.5.*, tačiau skaitiklyje vertinamas išgaubto daugiakampio plotas pakeliamas kvadratu. Ši įterpimo funkcija labiau tinkama didesniems išgaubto daukiakampio formos kontūrams.

$$Fit = \frac{[Area_{convex}(Polygon)]^2}{sumAreaNFP(State_{current}) - sumAreaNFP(State_{next})} \tag{3.13}$$

5. *Opt3.* – įterpimo funkcijos reikšmė priklauso nuo kontūro ploto ir nuo NFP maksimalaus ploto pokyčio, kuris bus sekančioje variacijoje.

$$Fit = \frac{Area(Polygon)}{maxAreaNFP(State_{current}) - maxAreaNFP(State_{next})} \tag{3.14}$$

6. *Opt4.* – įterpimo funkcija panaši į *Opt3.*, tačiau skaitiklyje kontūro plotas keliamas kvadratu. Ši įterpimo funkcija tinkama kai reikalinga vertinti didesnio ploto kontūrų padėtis.

$$Fit = \frac{[Area(Polygon)]^2}{maxAreaNFP(State_{current}) - maxAreaNFP(State_{next})}$$

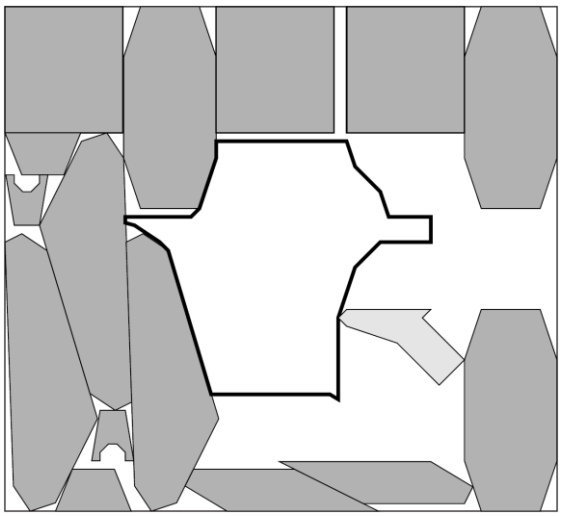
(3.15)

7. *Opt3.5.* – įterpimo funkcija panaši į *Opt3.*, tačiau skaitiklyje vertinamas išgaubtas daugiakampio kontūro plotas.

$$Fit = \frac{Area_{convex}(Polygon)}{maxAreaNFP(State_{current}) - maxAreaNFP(State_{next})} \quad (3.16)$$

8. *Opt4.5.* – įterpimo funkcija panaši į *Opt3.5.*, tačiau skaitiklyje vertinamas išgaubtas daukiamapio kontūro plotas pakeliamas kvadratu.

$$Fit = \frac{[Area_{convex}(Polygon)]^2}{maxAreaNFP(State_{current}) - maxAreaNFP(State_{next})} \quad (3.17)$$



3.12 pav. Kontūrų dėstymas pagal DAGLI algoritmo modifikaciją (Kierkosz ir Luczak, 2019)

Algoritmas parenka kontūrų padėtis priklausomai nuo to kokia įterpimo funkcija panaudota. Stačiakampio formos lakšto atveju praktiškiausia kontūrus dėstyti prie lašto kraštinių, paliekant centre laisvą erdvę. Kontūrai dėstomi apskritimu aplink lakšto centrą (Kierkosz ir Luczak, 2019).

3.2.3. Testavimo sąlygos

Šiame poskyryje aprašoma euristinio algoritmo testavime naudojama techninė ir programinė įranga ir testavimo sąlygos. Poskyryje 3.2.2. aprašytas algoritmas yra realizuotas C#.net 4.0 programavimo kalba. 3-1 lentelėje pateikiamas testavime naudojamos įrangos sąrašas.

3-1 lentelė. Euristinio dėstymo algoritmo tyrime naudojama įranga

Eil. Nr.	Įrangos tipas	Pavadinimas	Specifikacija
1	Techninė įranga	Kompiuteris	Intel Pentium Dual CPU 2.20 GHz, 2 GB RAM
2	Programinė įranga	Operacinė sistema	Microsoft Windows Vista
3	Programinė įranga	Algoritmo programavimo kalba	C#.net 4.0
4	Programinė įranga	Lines and polygons clipping tool	Clipper
5	Programinė įranga	Mathematic functions	CGAL 5.4 - 2D Minkowski Sums

Linijų, kontūrų, plotų skaičiavimams ir operacijoms naudojama pagalbinė programinė įranga – „Clipper“ (Johnson. 2014). NFP skaičiavimai atliekami vadovaujantis Minkowski suminėmis funkcijomis (angl. *Minkowski sum functions*) (Wein et al.,). Visos algoritme naudojamos funkcijos įdiegiamos ir vykdomos nuoseklia tvarka (angl. *single-threaded*).

Kontūrų dėstymo uždavinys skaidomas į 3 dalis (problemas):

1. Kontūrų dėstymas viename stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu (angl. *single knapsack problem*) – tokios pačios formos ir dydžio kontūras lakšte gali būti tik 1. Šiam uždaviniui tirti sukuriama 15 skirtingų variantų su skirtingais kontūrų ir lakšto parametrais. 3.2 lentelėje pateikti šie parametrai.
2. Kontūrų dėstymas viename stačiakampiame lakšte be vienodų kontūrų kiekio ribojimo (angl. *single large object placement problem*) – tokios pačios formos ir dydžio kontūrų kiekis lakšte neribojamas. Šiam uždaviniui tirti sukuriama 15 skirtingų variantų su skirtingais kontūrų ir lakšto parametrais, tačiau šiuo atveju neribojamas kontūrų kiekis. 3.2 lentelėje pateikti užduoties parametrai.
3. Kontūrų dėstymas neribojant stačiakampių lakštų kiekio su vienodų kontūrų kiekio ribojimu (angl. *single stock size cutting stock problem*). Šiam uždaviniui tirti sukuriama 15 skirtingų variantų su skirtingais kontūrų ir lakšto parametrais. Lakšto parametrai apribojami taip: plotis lygus aukščiui. 3.3 lentelėje pateikti užduoties parametrai.

Testavimo tikslas: ištestuoti euristinio algoritmo įvairios formos 2D kontūrų dėstymo stačiakampiame lakšte uždavinio sprendimo veikimo parametrus. Parametrai, pagal kuriuos bus vertinami gauti rezultatai:

- NFP skaičiavimo laikas.
- Įterpimo funkcijos skaičiavimo laikas.
- Bendras algoritmo skaičiavimo laikas.
- Visas kontūrų padengiamas plotas.

- Įterptų kontūrų kiekis.
- Kontūrų dėstymo maketas – kontūrų išdėstymo kokybė.
- Po kontūrų išdėstymo likęs laisvos vietos plotas.

Paminėtoms problemos tirti sukuriama 15 skirtingų užduoties sąlygų – variacijų. Kiekviena variacija skiriasi užduoties pradiniais duomenimis: lakšto parametrais (ilgis, plotis), kontūrų forma, kontūrų plotu, skirtingų kontūrų kiekiu, bendru kontūrų kiekiu naudojamu užduotyje, kontūrų pasukimo kampais. Tyrime eksperimentiniu būdu testuojama kaip kiekvienos variacijos užduotas sąlygas apskaičiuoja euristinis algoritmas su kiekviena iš 8 skirtingų įterpimo funkcijų.

Kiekvienai problemai tirti naudojami tokias pačias kontūrų formas turintys kontūrai. Taip pat, jų dydžiai nekeičiami tiriant visas 3 problemas. Tačiau, kad užduotis būtų aiškesnė, kiekvienai iš 3 aukščiau aprašytų problemų pateikiama atskira užduoties sąlygų lentelė (žr. 3.2 – 3.4 lenteles) su skirtingomis užduoties sąlygomis – pradiniais parametrais.

3-2 lentelėje pateikiami pirmąjai problemai tirti (kontūrų dėstymas viename stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu) skirti variacijų pradiniai užduoties parametrai.

3-2 lentelė. Užduoties sąlygos naudojamos kontūrų dėstymo viename stačiakampiame lakšte su vienodų kontūrų kiekio ribojimo problemai tirti

Eil. Nr.	Variacijos pavadinimas	Skirtingų kontūrų kiekis, m	Kontūrų kiekis, n	Lakšto aukštis	Lakšto plotis	Pasukimo kampas
1	FU	12	12	38,00	34,00	{0, 90, 180, 270}
2	JACKOBS1	25	25	40,00	13,00	{0, 90, 180, 270}
3	JACKOBS2	25	25	70,00	28,20	{0, 90, 180, 270}
4	SHAPES0	4	43	40,00	63,00	{0}
5	SHAPES1	4	43	40,00	59,00	{0, 180}
6	SHAPES2	7	28	15,00	27,30	{0, 180}
7	DIGHE1	16	16	100,00	138,14	{0}
8	DIGHE2	10	10	100,00	134,05	{0}
9	ALBANO	8	24	4900,00	10122,63	{0, 180}
10	DAGLI	10	30	60,00	65,60	{0, 180}
11	MAO	9	20	2550,00	2058,60	{0, 90, 180, 270}
12	MARQUES	8	24	104,00	83,60	{0, 90, 180, 270}
13	SHIRTS	8	99	40,00	63,13	{0, 180}
14	SWIM	10	48	5752,00	6568,00	{0, 180}
15	TROUSERS	17	64	79,00	245,75	{0, 180}

3-3 lentelėje pateikiami antrąjai problemai tirti (kontūrų dėstymas viename stačiakampiame lakšte be vienodų kontūrų kiekio ribojimo) skirti variacijų pradiniai užduoties parametrai.

3-3 lentelė. Užduoties sąlygos naudojamos kontūrų dėstymo viename stačiakampiame lakšte be vienodų kontūrų kiekio ribojimo problemai tirti

Eil. Nr.	Variacijos pavadinimas	Kontūrų kiekis, n	Lakšto aukštis	Lakšto plotis	Pasukimo kampas
1	FU	12	38,00	34,00	{0, 90, 180, 270}
2	JACKOBS1	25	40,00	13,00	{0, 90, 180, 270}
3	JACKOBS2	25	70,00	28,20	{0, 90, 180, 270}
4	SHAPES0	43	40,00	63,00	{0}
5	SHAPES1	43	40,00	59,00	{0, 180}
6	SHAPES2	28	15,00	27,30	{0, 180}
7	DIGHE1	16	100,00	138,14	{0}
8	DIGHE2	10	100,00	134,05	{0}
9	ALBANO	24	4900,00	10122,63	{0, 180}
10	DAGLI	30	60,00	65,60	{0, 180}
11	MAO	20	2550,00	2058,60	{0, 90, 180, 270}
12	MARQUES	24	104,00	83,60	{0, 90, 180, 270}
13	SHIRTS	99	40,00	63,13	{0, 180}
14	SWIM	48	5752,00	6568,00	{0, 180}
15	TROUSERS	64	79,00	245,75	{0, 180}

3-ajai problemai tirti sukuriama 15 skirtingų užduoties sąlygų – variacijų. Kiekviena variacija skiriasi užduoties pradiniais duomenimis: lakšto parametrais (ilgis, plotis), kontūrų pasukimo kampais. Skirtingų kontūrų kiekis yra apribojamas (100 vienetų skirtingos formos kontūrų) ir maksimalus kontūrų skaičius kiekvienoje variacijoje yra 100. Taip pat lakšto ilgis ir plotis yra suvienodinami – lakštai kvadratinės formos. Užduoties sąlygos tokios dėl to, kad šios problemos atveju yra tiriama kaip ta pati kontūrų imtis išdėstoma vienodos formos stačiakampiuose lakštuose.

Rezultatai vertinami pagal tai kiek laiko algoritmas generuoja rezultatus ir kiek lakštų yra sunaudojama kol išdėstomi visi kontūrai. Taip pat papildomai vertinama kontūrų dėstymo kokybė kiekviename lakšte. 3-4 lentelėje pateikiami 3-ajai problemai (kontūrų dėstymas neribojant stačiakampių lakštų kiekio su kontūrų kiekio ribojimu) spręsti skirti variacijų parametrai.

3-4 lentelė. Užduoties sąlygos su skirtingais lakšto parametrais ir bendrais kontūrų kiekio ribojimais

Eil. Nr.	Variacijos pavadinimas	Skirtingų kontūrų kiekis, m	Kontūrų kiekis, n	Lakšto aukštis	Lakšto plotis	Pasukimo kampas
1	FU	100	100	38,00	38,00	{0, 90, 180, 270}
2	JACKOBS1	100	100	40,00	40,00	{0, 90, 180, 270}
3	JACKOBS2	100	100	70,00	70,00	{0, 90, 180, 270}
4	SHAPES0	100	100	40,00	40,00	{0}
5	SHAPES1	100	100	40,00	40,00	{0, 180}
6	SHAPES2	100	100	15,00	15,00	{0, 180}
7	DIGHE1	100	100	100,00	100,00	{0}
8	DIGHE2	100	100	100,00	100,00	{0}
9	ALBANO	100	100	4900,00	4900,00	{0, 180}
10	DAGLI	100	100	60,00	60,00	{0, 180}
11	MAO	100	100	2550,00	2550,00	{0, 90, 180, 270}
12	MARQUES	100	100	104,00	104,00	{0, 90, 180, 270}
13	SHIRTS	100	100	40,00	40,00	{0, 180}
14	SWIM	100	100	5752,00	5752,00	{0, 180}
15	TROUSERS	100	100	79,00	79,00	{0, 180}

3.2.4. Rezultatai

Testavimo metu atliekamų eksperimentų tikslas – kiekvienai problemai surasti geriausiai tinkančią euristinio algoritmo ir įterpimo funkcijos kombinaciją, kurios pagalba būtų randamas optimaliausias įvairios formos 2D kontūrų dėstymo stačiakampiame lakšte variantas.

3-5 ir 3-6 lentelėse testavimo rezultatai pateikiami pagal įterptų kontūrų bendro ploto santykį su lakšto plotu (stulpelis „Area“) – lakšto užpildymo koeficientu ir pagal laiką, kuris sugaištas sprendžiant uždavinį. Geriausi rezultatai, pagal algoritmo įterpimo funkciją, pažymėti paryškintu šriftu.

3-5 lentelėje esantys testavimo rezultatai rodo, jog *Opt2.5.* įterpimo funkcija geriausius sprendimo rezultatus suranda esant kontūrų dėstymo stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu problemai. Algoritmas su šia įterpimo funkcija optimaliausius lakšto užpildymo rezultatus suranda 12-oje iš 15 skirtingų užduoties atvejų. Taip pat algoritmas su *Opt4.* įterpimo funkcija demonstruoja panašius rezultatus – 11 optimaliausių dėstymo variantų iš 15 skritingų užduoties atvejų.

3-5 lentelė. Dėstymo algoritmo skaičiavimo rezultatai pagal įterpimo funkcijas, sprendžiant kontūrų dėstymo stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu problemą

Name	Opt1		Opt2		Opt1.5		Opt2.5		Opt3		Opt4		Opt3.5		Opt4.5	
	Area	Time [s]	Area	Time [s]	Area	Time [s]	Area	Time [s]	Area	Time [s]	Area	Time [s]	Area	Time [s]	Area	Time [s]
FU	**0.8382**	1.00	**0.8382**	1.01	**0.8382**	0.93	**0.8382**	1.00	0.6865	1.22	**0.8382**	1.02	0.6865	1.22	**0.8382**	1.01
JACKOBS1	**0.7538**	18.84	**0.7538**	19.60	**0.7538**	22.83	**0.7538**	21.52	0.7154	15.87	**0.7538**	20.52	0.7058	37.29	**0.7538**	19.71
JACKOBS2	**0.6844**	30.52	**0.6844**	24.17	**0.6844**	31.91	**0.6844**	27.50	**0.6844**	34.00	**0.6844**	27.26	**0.6844**	34.79	**0.6844**	29.43
SHAPES0	0.5857	0.68	0.5984	0.6	0.5762	0.80	**0.6095**	0.70	0.6016	0.73	0.5587	0.44	0.5476	0.78	0.6016	0.55
SHAPES1	0.6593	2.83	0.6441	1.73	0.6458	4.73	**0.6763**	2.71	0.6763	3.30	0.6288	1.83	**0.6763**	4.26	**0.6763**	2.82
SHAPES2	0.7375	2.07	0.7643	1.81	0.7179	2.04	0.7668	1.73	0.7375	1.75	**0.7717**	1.31	0.7521	1.36	0.7375	1.09
DIGHE1	**0.7240**	0.43	**0.7240**	0.42	**0.7240**	0.45	**0.7240**	0.43	0.6561	0.40	**0.7240**	0.39	**0.7240**	0.41	0.6667	0.46
DIGHE2	**0.7460**	0.14	0.7042	0.12	**0.7460**	0.12	0.7042	0.10	**0.7460**	0.14	**0.7460**	0.12	**0.7460**	0.13	0.6285	0.08
ALBANO	0.7832	3.87	0.8297	2.52	0.6764	4.99	**0.8606**	2.76	0.7447	3.61	0.8357	2.15	0.7378	4.95	0.8287	2.74
DAGLI	**0.7731**	3.83	**0.7731**	3.58	**0.7731**	4.49	**0.7731**	3.50	0.6537	4.95	**0.7731**	3.61	**0.7731**	3.47	**0.7731**	3.80
MAO	**0.7160**	19.86	**0.7160**	16.26	**0.7160**	16.24	**0.7160**	15.21	**0.7160**	17.52	**0.7160**	18.21	**0.7160**	19.23	**0.7160**	14.27
MARQUES	0.7711	11.51	**0.8274**	5.23	0.7646	14.21	**0.8274**	7.31	0.7249	9.09	**0.8274**	9.75	**0.8274**	9.40	**0.8274**	8.31
SHIRTS	0.7958	44.42	0.8542	27.78	0.8025	39.75	0.8482	23.27	0.8025	20.70	**0.8554**	23.01	0.8025	23.94	**0.8554**	22.91
SWIM	**0.6734**	178.77	**0.6734**	159.94	**0.6734**	174.57	**0.6734**	178.85	**0.6734**	131.99	**0.6734**	131.03	0.6538	175.37	**0.6734**	111.28
TROUSERS	0.8122	105.81	**0.8863**	69.89	0.8507	71.97	**0.8863**	74.54	0.7367	62.96	0.8774	85.54	0.7866	89.01	0.8774	84.07

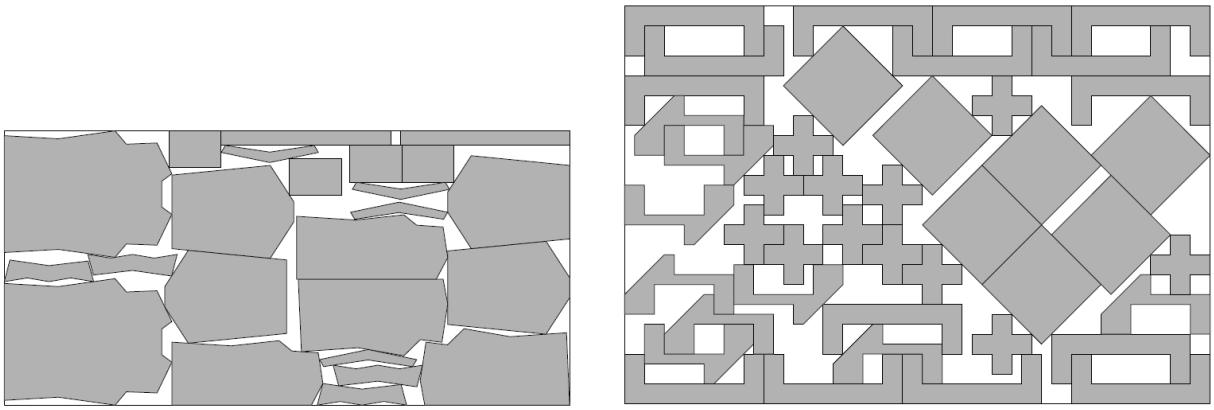
Spręsti kontūrų dėstymo stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu problemai visiškai netinkamas euristinis dėstymo algoritmas su *Opt3* įterpimo funkcija. Įvertinus visų algoritmo modifikacijų rezultatus matoma, jog šios problemos atveju, visų variacijų skaičiavimo laikas yra panašus, todėl apibendrinus užduoties rezultatus galima teigti, jog kontūrų dėstymo stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu problemai spręsti labiausiai tinkami algoritmai yra su *Opt2.5* ir *Opt4* įterpimo funkcijomis (Kierkosz ir Luczak, 2019).

Kontūrų dėstymo stačiakampiame lakšte be vienodų kontūrų kiekio ribojimo problemos sprendimo (naudojant euristinį algoritmą su kiekviena iš 8 įterpimo funkcijų) testavimo rezultatai pateikti 3-5 lentelėje. Pagal lentelės duomenis matoma, jog šiai problemai spręsti labiausiai tinkamas yra algoritmas su *Opt3* įterpimo funkcija. Šios problemos testavimo rezultatai rodo, jog skaičiavimo laikai tarp skirtingų algoritmų modifikacijų žymiai skiriasi(Kierkosz ir Luczak, 2019).

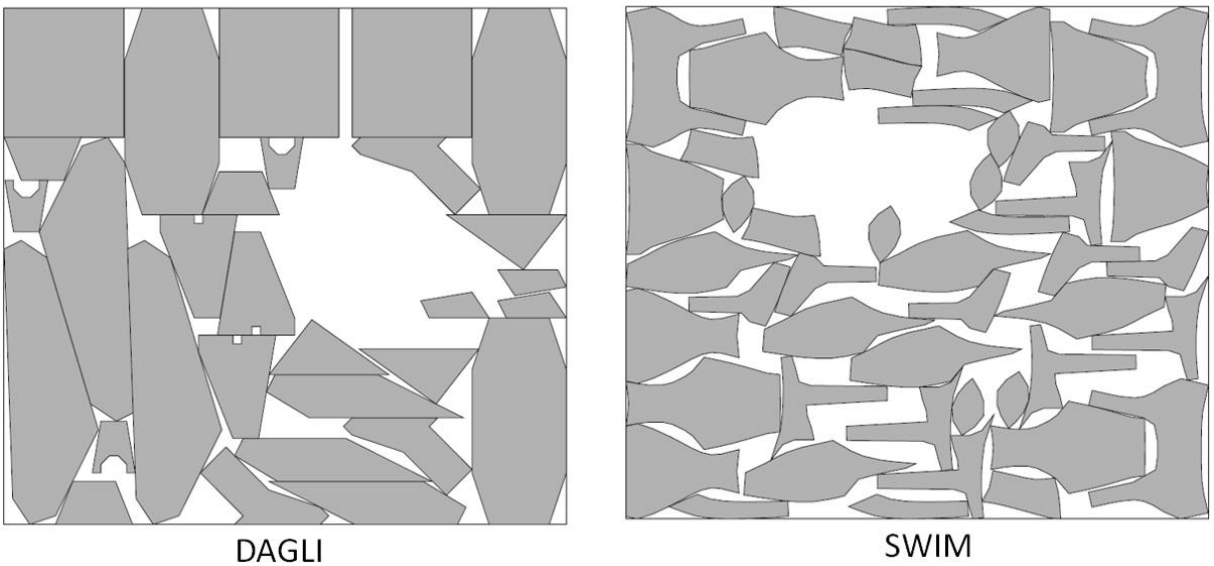
3-6 lentelė. Dėstymo algoritmo skaičiavimo rezultatai pagal įterpimo funkcijas, sprendžiant kontūrų dėstymo stačiakampiame lakšte be vienodų kontūrų kiekio ribojimo problemą

Name	Opt1		Opt2		Opt1.5		Opt2.5		Opt3		Opt4		Opt3.5		Opt4.5	
	Area	Time [s]	Area	Time [s]	Area	Time [s]	Area	Time [s]	Area	Time [s]	Area	Time [s]	Area	Time [s]	Area	Time [s]
FU	0.8390	2.58	**0.9412**	0.81	0.8390	2.57	**0.9412**	0.80	0.8189	3.86	0.9063	0.78	0.8189	3.90	0.9063	0.79
JACKOBS1	0.9058	150.36	0.8952	7.44	0.8712	168.27	0.8875	16.74	**0.9192**	193.07	0.8904	9.42	0.8808	75.95	0.8808	19.47
JACKOBS2	0.8409	87.02	0.8308	8.98	**0.8693**	192.53	0.8186	9.04	0.8354	77.29	0.8308	8.59	0.8141	111.31	0.8186	8.86
SHAPES0	0.6921	1.19	0.7397	0.32	0.6286	1.31	0.5571	0.66	**0.7810**	0.74	0.7317	0.33	0.6571	1.42	0.5540	0.61
SHAPES1	0.6932	3.59	0.7475	0.78	0.6559	6.90	0.6797	2.94	0.7373	3.37	**0.7559**	0.78	0.6593	6.16	0.6424	2.68
SHAPES2	0.8791	6.18	0.7900	1.76	0.6996	3.46	0.6093	2.08	**0.8864**	5.25	0.8132	1.59	0.6557	2.13	0.5873	1.78
DIGHE1	0.7669	1.87	0.7249	0.35	0.7669	1.85	0.7710	0.41	**0.7991**	1.52	0.7490	0.50	0.7565	1.54	0.7333	0.43
DIGHE2	0.7840	0.24	**0.8188**	0.12	0.7840	0.25	**0.8188**	0.10	0.7699	0.26	0.7292	0.08	0.7699	0.27	0.7292	0.09
ALBANO	0.8635	12.66	0.8360	2.52	0.8304	16.18	0.8474	2.65	**0.8822**	6.15	0.8331	3.14	0.7858	44.74	0.8331	3.19
DAGLI	0.8465	13.79	0.8633	2.92	0.7872	31.19	0.8569	3.50	**0.8947**	4.75	0.8615	2.29	0.8637	11.96	0.8456	2.30
MAO	**0.9197**	830.77	0.8737	19.43	0.9070	700.98	0.8647	19.10	0.9189	74.79	0.8687	10.93	0.9084	64.52	0.8746	11.13
MARQUES	**0.8821**	39.40	0.8341	4.12	0.8538	43.56	0.8357	4.72	0.8702	22.83	0.8432	7.16	0.7803	151.52	0.8432	7.61
SHIRTS	0.9512	323.30	0.8479	10.87	0.9437	319.72	0.8463	10.50	**0.9706**	37.93	0.8677	11.45	0.9069	63.86	0.8661	11.29
SWIM	0.7828	497.17	0.7643	211.32	0.7398	386.52	0.6869	130.39	**0.8081**	530.50	0.8025	181.06	0.7209	199.10	0.6563	63.64
TROUSERS	0.9302	1020.01	0.9089	23.75	0.9340	795.21	0.9089	23.41	**0.9367**	408.35	0.9122	63.63	0.9300	275.48	0.9122	64.21

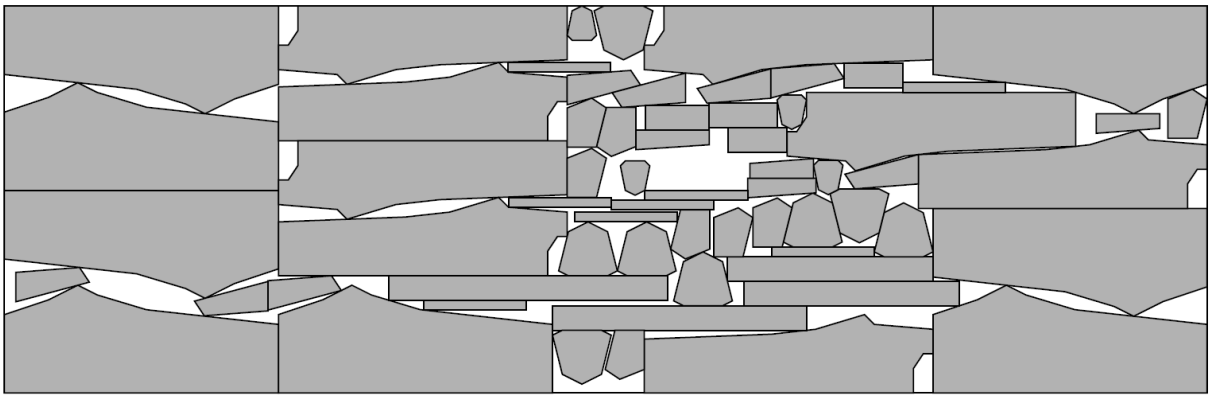
Žemiau esančiuose 3.13 – 3.15 paveikslėliuose pateikiami optimaliausi įvairios formos 2D kontūrų dėstymo rezultatai gauti sprendžiant kontūrų dėstymo viename stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu problemą. Šiems rezultatams gauti buvo naudojamas euristinis algoritmas su *Opt2.5* įterpimo funkcija.



3.13 pav. Euristinio dėstymo algoritmo su Opt2.5 įterpimo funkcija optimalūs dėstymo sprendimai ALBANO ir SHAPES1 užduoties variacijose, kai sprendžiama kontūrų dėstymo viename stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu problema



3.14 pav. Euristinio dėstymo algoritmo su Opt2.5 įterpimo funkcija optimalūs dėstymo sprendimai DAGLI ir SWIM užduoties variacijose, kai sprendžiama kontūrų dėstymo viename stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu problema

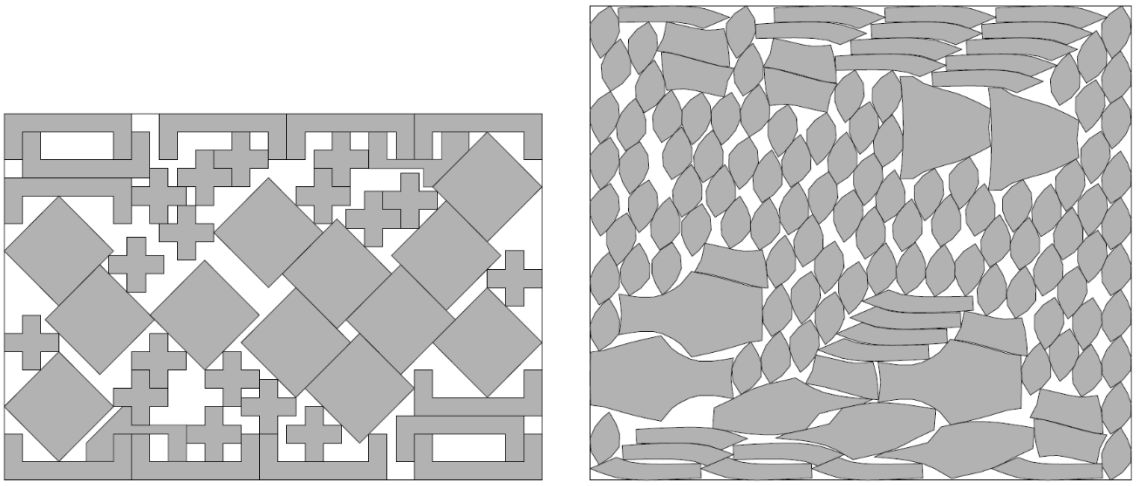


TROUSERS

3.15 pav. Euristinio dėstymo algoritmo su Opt2.5 įterpimo funkcija optimalus dėstymo sprendimas TROUSERS užduoties variacijoje, kai sprendžiama kontūrų dėstymo viename stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu problema

3.14 – 3.15 paveikslėliuose matoma, jog DAGLI, SWIM ir TROUSERS užduoties variacijų atvejais, visi kontūrai buvo patalpinti lakšte ir lakšto centre dar liko laisvos vietos. Tai rodo, jog euristinis algoritmas su Opt2.5 įterpimo funkcija ne tik pasižymi maža užduoties skaičiavimo trukme bet ir savybe rasti optimalius įvairios formos 2D kontūrų dėstymo stačiakiampiuose lakštuose sprendimus.

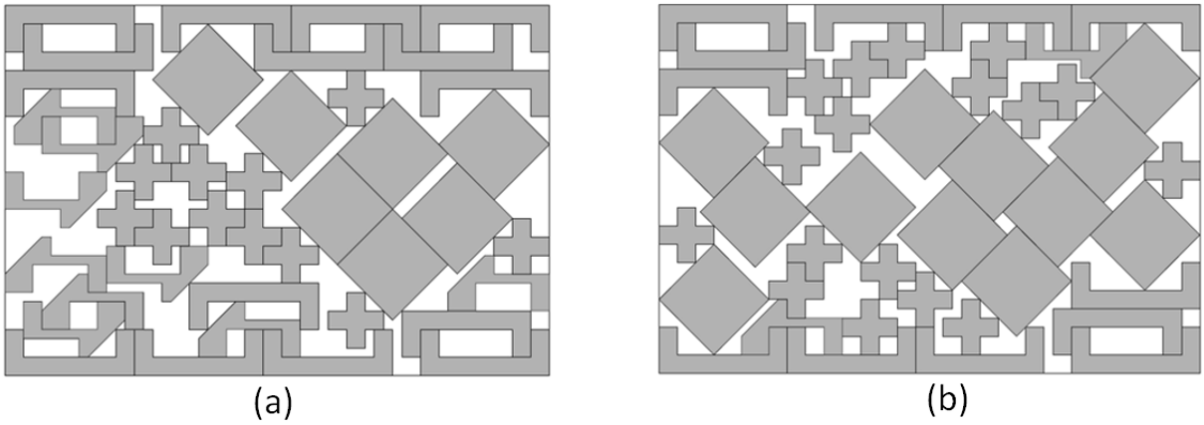
3.16 – 3.17 paveikslėliuose pateikiami optimaliausi įvairios formos 2D kontūrų dėstymo rezultatai gauti sprendžiant kontūrų dėstymo viename stačiakampiame lakšte be vienodų kontūrų kiekio ribojimo problemą. Šiems rezultatams gauti buvo naudojamas euristinis algoritmas su *Opt2.5* įterpimo funkcija.



SHAPES1 SWIM

3.16 pav. Euristinio dėstymo algoritmo su Opt2.5 įterpimo funkcija optimalus dėstymo sprendimas užduoties SHAPES1 ir SWIM variacijose, kai sprendžiama kontūrų dėstymo viename stačiakampiame lakšte be vienodų kontūrų kiekio ribojimo problema

Galima palyginti, kaip tokios pat formos kontūrus lakšte išdėsto euristiniai algoritmai su *Opt2.5* įterpimo funkcija esant kontūrų dėstymo viename stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu ir be vienodų kontūrų kiekio ribojimo problemoms. Palyginimas matomas žemiau esančiame 3.17 paveikslėlyje.



3.17 pav. Euristinio dėstymo algoritmo su Opt2.5 įterpimo funkcija optimalus dėstymo sprendimas užduoties SHAPES1 variacijoje palyginimas. (a) paveikslėlyje sprendžiama kontūrų dėstymo viename stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu problema, (b) paveikslėlyje sprendžiama kontūrų dėstymo viename stačiakampiame lakšte be vienodų kontūrų kiekio ribojimo problema

Spręsti kontūrų dėstymo stačiakampiame lakšte problemai neribojant stačiakampių lakštų kiekio, bet su vienodų kontūrų kiekio ribojimu ir maksimaliu kontūrų kiekio ribojimu, geriausius rezultatus (mažiausias sunaudotų lakštų kiekis ir didžiausias skaičiavimo greitis) pateikia algoritmas su *Opt2.5* įterpimo funkcija. Šios problemos sprendimo rezultatai matomi 3-7 lentelėje (Kierkosz ir Luczak, 2019).

3-7 lentelė. Dėstymo algoritmo skaičiavimo rezultatai pagal įterpimo funkcijas, sprendžiant kontūrų dėstymo stačiakampiame lakšte be lakšto kiekio ribojimo tačiau su vienodų kontūrų kiekio ribojimu problemą

Name	Opt1		Opt2		Opt1.5		Opt2.5		Opt3		Opt4		Opt3.5		Opt4.5	
	# Plates	Time [s]	# Plates	Time [s]	# Plates	Time [s]	# Plates	Time [s]	# Plates	Time [s]	# Plates	Time [s]	# Plates	Time [s]	# Plates	Time [s]
FU	95	23.17	88	24.99	95	23.03	88	25.20	92	25.55	**86**	18.20	92	25.41	**86**	17.78
JACKOBS1	30	7124.88	29	8174.81	30	13541.36	**28**	7379.89	30	8140.65	29	6359.71	29	7622.93	29	7637.56
JACKOBS2	34	11445.07	34	10363.43	34	14863.52	**33**	8321.25	34	5680.36	**33**	5708.01	34	5692.61	**33**	6057.03
SHAPES0	193	2.59	**168**	3.15	191	1.54	169	2.80	188	3.57	171	3.22	182	1.96	169	2.38
SHAPES1	161	5.74	**157**	8.54	166	4.34	158	6.51	158	4.83	159	7.42	168	8.26	159	6.72
SHAPES2	207	5.95	194	6.58	213	5.53	**191**	3.92	205	7.14	193	4.55	210	6.16	196	4.62
DIGHE1	156	5.81	**138**	7.28	145	5.11	140	4.20	144	5.88	144	4.34	148	6.30	142	5.39
DIGHE2	**135**	0.98	139	0.84	**135**	0.98	144	1.05	148	1.33	136	1.12	139	1.26	147	1.12
ALBANO	233	5.88	240	5.74	227	7.07	**220**	4.13	226	4.97	217	3.57	227	6.72	223	2.03
DAGLI	107	73.50	100	54.60	108	96.81	100	51.87	**85**	41.16	100	74.83	108	61.18	101	68.39
MAO	75	2748.90	71	233.10	74	2661.26	72	177.17	76	1222.06	**70**	206.29	74	437.99	**70**	181.86
MARQUES	82	324.87	**76**	315.49	86	659.26	**76**	247.24	81	245.42	77	240.03	82	601.44	77	169.96
SHIRTS	175	392.14	160	100.10	174	320.04	159	113.68	172	224.84	**158**	89.04	171	139.30	**158**	77.00
SWIM	115	1827.70	108	1375.08	111	1280.72	**107**	1357.51	114	1906.17	108	1027.46	111	1137.99	**107**	1095.78
TROUSERS	388	243.67	327	62.72	**227**	145.88	327	63.56	385	374.22	343	56.07	344	234.64	343	56.14
POLY 5B	157	15460.41	157	10632.30	157	15211.28	155	11129.09	**154**	13234.76	156	9914.31	155	12656.35	155	9563.33

Euristinio dėstymo algoritmo su įterpimo funkcija tyrimo metu gauti rezultatai papildomai palyginami su Del Valle et al. – *Heuristics for two-dimensional knapsack and cutting stock problems with items of irregular shape* (2012) leidinyje aprašyto tyrimo rezultatais. Abiejuose testavimuose yra naudojamos tos pačios užduoties variacijos ir NFP technologijos. Del Valle et al. (2012) leidinyje aprašyto tyrimo įranga pateikta 3-8 lentelėje.

3-8 lentelė. Euristinio dėstymo algoritmo tyrime pagal Del Valle et al. – *Heuristics for two-dimensional knapsack and cutting stock problems with items of irregular shape* (2012) naudojama įranga

Eil. Nr.	Įrangos tipas	Pavadinimas	Specifikacija
1	Techninė įranga	Kompiuteris	Intel Core 2 Quad CPU 2.4 GHz
2	Programinė įranga	Operacinė sistema	Microsoft Windows Vista
3	Programinė įranga	Algoritmo programavimo kalba	C++
4	Programinė įranga	Lines and polygons clipping tool	Clipper
5	Programinė įranga	Mathematic functions	CGAL 5.4 - 2D Minkowski Sums

NFP skaičiavimo trukmė visoms kontūrų dėstymo kombinacijoms nei vieno tyrimo atveju neįtraukiama į testavimo rezultatus, o vertinamas tik įterpimo funkcijos skaičiavimo laikas. NFP skaičiavimo trukmes galima palyginti tarp abiejų tyrimų.

3-9 lentelė. NFP skaičiavimui reikalingas laikas sekundėmis (Del Valle et al. 2012)

Eil. Nr.	Variacijos pavadinimas	Del Valle et. Al (2012) testavimo metu gautas NFP skaičiavimo laikas, s	Kierkosz ir Luczak (2019) testavimo metu gautas NFP skaičiavimo laikas, s
1	FU	0,26	0,50
2	JACKOBS1	59,49	1,94
3	JACKOBS2	53,80	1,91
4	SHAPES0	51,32	0,02
5	SHAPES1	202,56	0,03
6	SHAPES2	17,90	0,07
7	DIGHE1	0,12	0,10
8	DIGHE2	0,15	0,07
9	ALBANO	14,40	0,19
10	DAGLI	26,16	0,17
11	MAO	102,91	1,52
12	MARQUES	57,50	0,43
13	SHIRTS	208,49	0,11

14	SWIM	1088,21	2,86
15	TROUSERS	48,22	0,38

Taip pat palyginami kontūrų dėstymo stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu ir be vienodų kontūrų kiekio ribojimo užduočių sprendimo rezultatai su Del Valle et al. – *Heuristics for two-dimensional knapsack and cutting stock problems with items of irregular shape* (2012) rezultatais. Pagal 3.6 ir 3.7 lenteles, geriausius rezultatus pademonstravo algoritmai su *Opt2.5* ir *Opt.3* įterpimo funkcijomis. Todėl algoritmų su šiomis įterpimo funkcijomis testavimo rezultatų duomenys palyginami su Del Valle et al (2012) tyrimo rezultatų duomenimis.

Rezultatuose naudojami šie žymenys:

- Area – lakšto užpildymo koeficientas (bendras lakšte įterptų kontūrų ploto santykis su lakšto plotu).
- Time – laikas sugaištas rezultato radimui be NFP skaičiavimo laiko.
- GRASP – algoritmo pavadinimas pagal (Del Valle et al. 2012).
- Solve2KP – algoritmo pavadinimas pagal (Del Valle et al. 2012).

3-10 lentelė. Euristinio algoritmo su įterpimo funkcija testavimo rezultatų palyginimas su Del Valle et al. (2012) testavimo rezultatais sprendžiant kontūrų dėstymo stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu problemą

Name	Del Valle et al.		GRASP	Opt2.5		
	Area	Quantity of items	Time [s]	Area	Quantity of items	Time [s]
FU	**0.8382**	**12**	21.79	**0.8382**	**12**	1.00
JACKOBS1	**0.7538**	**25**	8.30	**0.7538**	**25**	21.52
JACKOBS2	**0.6844**	**25**	565.71	**0.6844**	**25**	27.50
SHAPES0	0.6016	41	1552.46	**0.6095**	40	0.70
SHAPES1	0.6424	41	3891.60	**0.6763**	**43**	2.71
SHAPES2	0.7289	26	1048.12	**0.7668**	26	1.73
DIGHE1	**0.7240**	**16**	10.68	**0.7240**	**16**	0.43
DIGHE2	**0.7460**	**10**	0.07	0.7042	9	0.10
ALBANO	0.8038	23	961.59	**0.8606**	**24**	2.76
DAGLI	0.7586	29	1106.40	**0.7731**	**30**	3.50
MAO	**0.7160**	**20**	120.42	**0.7160**	**20**	15.21
MARQUES	**0.8274**	**24**	217.77	**0.8274**	**24**	7.31
SHIRTS	0.7702	96	14317.13	**0.8482**	93	23.27
SWIM	0.6427	46	39781.43	**0.6734**	**48**	178.85
TROUSERS	0.7866	62	5796.59	**0.8863**	**64**	74.54

3.10 lentelės testavimo rezultatai rodo, jog euristinis dėstymo algoritmas su *Opt2.5* įterpimo funkcija kontūrus prie 14 iš 15 užduoties variacijų išdėsto mažesniame lakšto plote.

Taip pat, skaičiavimams sugaištama žymiai mažesnis kiekis laiko. 3-10 lentelėje stulpelyje "Quantity of items" paryškintu šriftu pažymėti optimalūs sprendimai.

3-11 lentelė. Euristinio algoritmo su įterpimo funkcija testavimo rezultatų palyginimas su Del Valle et al (2012) testavimo rezultatais sprendžiant kontūrų dėstymo problemą stačiakampiame lakšte be vienodų kontūrų kiekio ribojimo

Name	Del Valle et al.	Solve2KP (av.)	Opt3	
	Area	Time [s]	Area	Time [s]
FU	**0.9892**	5.01	0.8189	3.86
JACKOBS1	**0.9870**	49.60	0.9192	193.07
JACKOBS2	**0.9851**	66.08	0.8354	77.29
SHAPES0	0.5873	134.43	**0.7810**	0.74
SHAPES1	0.6893	248.15	**0.7373**	3.37
SHAPES2	**0.9183**	49.37	0.8864	5.25
DIGHE1	0.6909	7.62	**0.7991**	1.52
DIGHE2	0.7657	3.14	**0.7699**	0.26
ALBANO	**0.9653**	37.93	0.8822	6.15
DAGLI	**0.9196**	50.99	0.8947	4.75
MAO	**0.9644**	71.53	0.9189	74.79
MARQUES	**0.9515**	43.76	0.8702	22.83
SHIRTS	**1.0000**	508.92	0.9706	37.93
SWIM	0.7272	2206.42	**0.8081**	530.50
TROUSERS	**0.9986**	193.54	0.9367	408.35

3-11 lentelės testavimo rezultatai rodo, jog euristinis dėstymo algoritmas su *Opt.3* įterpimo funkcija (šis algoritmas, pagal 3.6 lentelę, yra labiausiai tinkamas spręsti kontūrų dėstymo stačiakampiame lakšte be vienodų kontūrų kiekio ribojimo problemas) generuoja prastesnius rezultatus negu euristinis Solve2KP algoritmas. Taip yra dėl to, jog euristinis Solve2KP algoritmas yra labiau tinkamas spręsti stačiakampio formos 2D kontūrų dėstymo uždavinius, kuriuose nėra vienodų kontūrų kiekio ribojimų. Tačiau jeigu užduotyje bus nedidelis kiekis stačiakampio formos 2D kontūrų – euristinis Solve2KP neranda optimalių dėstymo sprendimų. Apibendrinus galima teigti, kad euristinis dėstymo algoritmas su *Opt.3* įterpimo funkcija yra geresnis pasirinkimas sprendžiant 2D kontūrų dėstymo uždavinius, kuriuose dominuoja įvairios formos kontūrai. Taip pat, pagal 3.11 lentelėje esančius testavimo rezultatus matoma, jog euristinis dėstymo algoritmas su *Opt.3* įterpimo funkcija žymiai greičiau suranda užduoties sprendimus negu euristinis Solve2KP algoritmas (Kierkosz ir Luczak, 2019).

Toliau lyginami kontūrų dėstymo stačiakampiame lakšte be lakšto kiekio ribojimo tačiau su vienodų kontūrų kiekio ribojimu užduoties sprendimo rezultatai su Del Valle et al. –

Heuristics for two-dimensional knapsack and cutting stock problems with items of irregular shape (2012) rezultatais.

Šiuose tyrimuose kontūrai talpinami stačiakampiuose lakštuose. Mažiausiai lakštų numatytam kontūrų kiekui patalpinti sunaudojęs algoritmo variantas laikomas pranašesniu.

Rezultatuose naudojami šie žymenys:

- # Plates – lakštų skaičius,
- Time – laikas sugaištas rezultato radimui be NFP skaičiavimo laiko.
- Solve2CS – algoritmo pavadinimas pagal Del Valle et al. (2012).

3-12 lentelė. Euristinio algoritmo su įterpimo funkcija tyrimo rezultatų palyginimas su (Del Valle et al. 2012) tyrimo rezultatais sprendžiant kontūrų dėstymo stačiakampiame lakšte be lakšto kiekio ribojimo tačiau su vienodų kontūrų kiekio ribojimu užduotį

Name	Del Valle et al. Solve2CS		Opt4	
	# Plates	Time [s]	# Plates	Time [s]
FU	74	228.93	**67**	14.48
JACKOBS1	50	6,726.73	**46**	414.79
JACKOBS2	47	6,897.00	**38**	624.44
SHAPES0	55	26,701.14	**51**	4.24
SHAPES1	56	59,601.60	**48**	13.79
SHAPES2	**63**	9,413.40	64	11.61
DIGHE1	61	252.30	**47**	8.75
DIGHE2	46	45.18	**43**	1.41
ALBANO	84	6,750.17	**78**	38.14
DAGLI	58	9,304.73	**50**	42.34
MAO	49	7,073.23	**41**	109.59
MARQUES	53	8,298.07	**49**	137.97
SHIRTS	45	1,195,677.01	**42**	198.24
SWIM	60	466,819.10	**47**	1554.47
TROUSERS	53	273,141.60	**47**	1440.29

Palyginimo duomenys rodo, jog euristinis algoritmas su *Opt4* įterpimo funkcija generuoja geriausius rezultatus prie visų skirtingų užduoties variacijų. Euristinis algoritmas su *Opt4* įterpimo funkcija pranašesnis, dėl to kad funkcijos vykdomos nuoseklia tvarka (angl. *single-threaded*). Taip pat svarbu atkreipti dėmesį, jog šio algoritmo sprendimo ieškojimo laikas yra žymiai trumpesnis, nors tyrime naudotas kompiuteris turi prastesnių techninių parametrų techninę įrangą.

Del Valle et al. (2012) tyrimo atveju euristinis Solve2CS algoritmas realizuotas C++ programavimo kalba, o euristinio algoritmo su įterpimo funkcija tyrimo atveju C# .net. Galima daryti išvadą, jog naudojant geresnių parametrų techninę įrangą euristinis algoritmas su *Opt4* įterpimo funkcija C# .net programavimo kalboje rezultatus generuotų dar greičiau.

3.2.5. Rezultatų apibendrinimas

Euristinio algoritmo tyrime buvo sprendžiamos 3 įvairios formos 2D kontūrų dėstymo stačiakampiame lakšte problemos:

1. Kontūrų dėstymas viename stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu (angl. *single knapsack problem*) – tokios pačios formos ir dydžio kontūras lakšte gali būti tik 1.
2. Kontūrų dėstymas viename stačiakampiame lakšte be vienodų kontūrų kiekio ribojimo (angl. *single large object placement problem*) – tokios pačios formos ir dydžio kontūrų kiekis lakšte neribojamas.
3. Kontūrų dėstymas neribojant stačiakampių lakštų kiekio su vienodų kontūrų kiekio ribojimu (angl. *single stock size cutting stock problem*) tokios pačios formos ir dydžio kontūras lakšte gali būti tik 1.

Pirmų dviejų problemų sprendimo tikslas – sutalpinti kaip galima daugiau kontūrų viename lakšte (siekiama, jog bendras kontūrų plotas lakšte būtų kuo didesnis). Trečios problemos sprendimo tikslas – siekiama kontūrus sutalpinti į kuo mažesnį kiekį lakštų.

Šioms problemoms spręsti realizuotas euristinis dėstymo algoritmas, kurio funkcijos vykdomos nuoseklia tvarka (angl. *single-threaded*). Algoritme įterpimo funkcija nusprendžia kuris kontūras sekančiame žingsnyje bus patalpintas lakšte. Įterpimo funkcija naudoja NFP technologiją, kuri padeda surasti laisvą plotą, kuris liks po kontūro įterpimo. Iš viso išbandomos 8 skirtingos įterpimo funkcijos. Testavimo metu buvo nustatytas kiekvienai problemai spręsti geriausiai tinkantis euristinis algoritmas su įterpimo funkcija:

1. Kontūrų dėstymo viename stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu problemą efektyviausiai išsprendžia euristinis algoritmas su *Opt2.5* įterpimo funkcija.
2. Kontūrų dėstymo viename stačiakampiame lakšte be vienodų kontūrų kiekio ribojimo problemą efektyviausiai išsprendžia euristinis algoritmas su Opt3 įterpimo funkcija.
3. Kontūrų dėstymo neribojant stačiakampių lakštų kiekio su vienodų kontūrų kiekio ribojimu problemą efektyviausiai išsprendžia euristinis algoritmas su *Opt2.5* įterpimo funkcija.

Testavimo rezultatai papildomai palyginti su Del Valle et al. – *Heuristics for two-dimensional knapsack and cutting stock problems with items of irregular shape* (2012) leidinyje publikuotais testavimo rezultatais. Paminėto leidinio autorių testavime buvo naudojama pranašesnė techninė įranga bei algoritmas realizuotas naudojant kitą programavimo kalbą. Tačiau palyginimas rodo, jog euristinis algoritmas su įterpimo funkcija, kurio funkcijos

vykdomos nuoseklia tvarka (angl. *single-threaded*) ir yra realizuotas C#.net 4.0 programavimo kalba, yra žymiai pranašesnis sprendžiant visas aukščiau išvardintas problemas. Tačiau atvejais, kai dėstymo užduotis prašo išdėstyti kontūrus, kurių didžioji dalis yra stačiakampio formos – labiau tinkamas yra euristinis Solve2KP algoritmas.

Apibendrinus testavimo rezultatus galima išvada: euristiniai dėstymo algoritmai su įdiegtomis įterpimo funkcijomis ir NFP tehcnologijomis, generuoja geresnius įvairios formos 2D kontūrų dėstymo uždavinių rezultatus. Taip pat jų veikimas yra greitas.

3.3. Trumpas skyriaus apibendrinimas

Įvairios formos 2D kontūrų optimalaus dėstymo tyrimas atliktas analizuojant kombinuotų dėstymo algoritmų veikimą CAN sistemoje bei tiriant euristinius dėstymo algoritmus, kurių funkcijos vykdomos nuosekliai ir kurie turi įdiegtas įterpimo funkcijas. CAN sistema ir euristiniai dėstymo algoritmai, kurie turi įdiegtas įterpimo funkcijas, naudoja NFP ir grafinių regionų nustatymo technologijas padedančias nustatyti laisvą erdvę liekiančią po kontūrų perstumdymo ar įterpimo bei rasti kontūruose esančias laisvas erdves.

CAN sistema leidžia iš CAD aplinkų įterptus brėžinių duomenis naudoti dėstant įvarios formos 2D kontūrus lakšte. Dėstymo uždaviniai CAN sistemoje sprendžiami naudojant įvairios formos 2D kontūrų dėstymo algoritmus, o sprendimo rezultatai transformuojami į koduotę, kuri gali būti tiesiogiai perduodama CNC įrenginių kompiuteriams.

Euristinio algoritmo tyrimo metu naudojami euristiniai dėstymo algoritmai, kuriuose veiksmai vykdomi nuosekliai. Algoritmas naudoja įterpimo funkciją, kuri ir nusprendžia kurioje lakšto vietoje bus patalpintas sekantis kontūras. Tyrimo tikslas: išspręsti 3 skirtingas įvairios formos 2D kontūrų dėstymo problemas. Iš viso tyrime panaudota 8 skirtingos įterpimo funkcijos ir po 15 skirtingų užduoties variacijų prie kiekvienos problemos. Eksperimento rezultatų pagalba nustatyta, koks algoritmas geriausiai išsprendžia kiekvieną problemą. Taip pat rezultatai papildomai palyginami su kitų autorių atliktu tyrimu, kuriame buvo naudojami algoritmai be įterpimo funkcijų.

IŠVADOS

Baigiamajame magistro darbe iškeltas tikslas pasiektas atlikus analitinę įvairios formos 2D kontūrų dėstymo algoritmų literatūros analizę, ištyrus ir išanalizavus kombinuotų dėstymo algoritmų veikimo savybes juos jungiančioje CAN sistemoje bei atlikus euristinių algoritmų su įterpimo funkcijomis eksperimentinį tyrimą.

Atlikus analitinę literatūros apžvalgą nustatyta:

1. Įvairios formos 2D kontūrų optimalaus dėstymo stačiakampiame lakšte užduotis apima kelis skirtingus etapus (kontūrų įterpimas, laisvos vietos ieškojimas, kontūrų grupavimas, dėstymas, optimalaus varianto suradimas). Todėl norint spręsti sudėtingus įvairios formos kontūrų dėstymo uždavinius dažniausiai yra pasitelkiamas kompleksas priemonių (algoritmai ir jų kombinacijos, juos apjungiančios sistemos, NFP ir IFP grafinės technologijos).
2. Pagrindinės įvairios formos 2D kontūrų dėstymo algoritmų savybės:
   - Aptvaro algoritmai pasižymi dideliu skaičiavimo greičiu, tačiau kontūrų išdėstymo efektyvumas yra žemas dėl to, kad lieka dideli tušti tarpai tarp kontūro ir aptvaro kraštinių. Efektyviausi kai kontūrai yra pasikartojantys. Dažniausiai naudojami kartu su kitais dėstymo algoritmais.
   - Dinaminiai paskirstymo algoritmai naudojami kai reikalinga sumažinti tarpelius tarp kontūro ir aptvaro kraštinių. Pasižymi aukštu kontūrų išdėstymo efektyvumu, tačiau atlieka tik siaurią specifinę užduotį, todėl turi būti naudojami kartu su kitais dėstymo algoritmais.
   - Dėstymo apačioje kairėje algoritmai pasižymi dideliu veikimo greičiu ir paprastumu bei vidutiniu kontūrų dėstymo efektyvumu. Efektyviausi kai kontūrai yra pasikartojantys. Dažniausiai naudojami kartu su kitais dėstymo algoritmais.
   - Euristiniai algoritmai tinkamiausi atvejuose, kai kontūrai tarpusavyje skiriasi ir jie yra sunkiai sugrupuojami. Kontūrų dėstymo efektyvumas vidutinis. Ypač efektyvūs kai reikia patalpinti kontūrus į laisvas erdves tarp kontūrų ar jų viduje. Dažniausiai naudojami kartu su kitais dėstymo algoritmais.
   - Atsitiktinio dėstymo algoritmai pasižymi labai aukštu kontūrų dėstymo efektyvumu, tačiau skaičiavimo laikas yra didelis. Dažniausiai naudojami kartu su dėstymo apačioje kairėje algoritmais.

- Grupavimo algoritmai efektyvus tuomet, kai kontūrų imtyje yra nedaug skirtingų formų variantų ir jų dydžiai yra panašūs. Efektyviai sprendžia kolekcionuojamų kontūrų dėstymo problemas. Visais atvejais yra kombinuojami su euristiniais algoritmais.
- Laisvos vietos ieškojimo algoritmai pasižymi tiksliu laisvos vietos suradimu esant daugiakampiams kontūrams. Visada yra naudojami su kitais kontūrų dėstymo algoritmais.

3. Norint efektyviai spręsti įvairios formos 2D kontūrų dėstymo užduotis ir rasti kontūrų išdėstymo sprendimus, reikalinga kombinuoti kelis skirtingus kontūrų dėstymo algoritmus. Tai galima padaryti įdiegiant vieno algoritmo funkcinius elementus kartu su kitų algoritmų funkciniais elementais. Taip pat praktikoje dažnai naudojamos algoritmus apjungiančios sistemos.
4. Beveik visi algoritmai, skirti spręsti įvairios formos 2D kontūrų optimalaus dėstymo uždavinius, turi euristinius sprendimo elementus: generuojamos skirtingos kontūrų padėtys siekiant per kuo trumpesnį laiką gauti pradinius dėstymo rezultatus, kurie vėliau naudojami optimaliausio dėstymo varianto parinkimui.

Atlikus įvairios formos 2D kontūrų optimalaus dėstymo algoritmų eksperimentinį tyrimą nustatyta:

1. Euristiniai įvairios formos 2D kontūrų dėstymo algoritmai su įdiegtomis įterpimo funkcijomis ir NFP bei grafinių regionų technologijomis sugeba efektyviai spręsti įvairios formos 2D kontūrų dėstymo uždavinius.
2. Kontūrų dėstymo viename stačiakampiame lakšte su vienodų kontūrų kiekio ribojimu problemą efektyviausiai išsprendžia euristinis algoritmas su *Opt2.5* įterpimo funkcija.
3. Kontūrų dėstymo viename stačiakampiame lakšte be vienodų kontūrų kiekio ribojimo problemą efektyviausiai išsprendžia euristinis algoritmas su Opt3 įterpimo funkcija.
4. Kontūrų dėstymo neribojant stačiakampių lakštų kiekio su vienodų kontūrų kiekio ribojimu problemą efektyviausiai išsprendžia euristinis algoritmas su *Opt2.5* įterpimo funkcija.

LITERATŪRA

- Cheng, S.K. and Rao, K.P. *Large-scale nesting of irregular patterns using compact neighborhood algorithm.* J. Mater. Process. Technol. (2000). 135–140
- Dumitrescu, A. *An approximation algorithm for cutting out convex polygons*. Configurable Computing: Technology and Applications. (1998). 823–827
- Grinde, R.B. and Cavalier, T.M. *New algorithm for the minimal-area convex enclosure problem.* Eur. J. Opl Res. (1995). 522–538
- *Matheuristics for the irregular bin packing problem with free rotations*. European Journal of Operational Research. (2016). 440 – 455
- *Nesting of two-dimensional irregular parts: an integrated approach.* International Journal of Computer Integrated Manufacturing. (2007). 741 – 756
- Roth, J., Hashimshony, R. and Wachman, A. *Generating layouts with non-convex envelopes.* Build. Environ. (1985). 211–219
- Dowsland, K.A., Vaid, S. and Dowsland, W.B. *An algorithm for polygon placement using a bottom-left strategy*. Eur. J. Opl Res. (2002). 371–381
- Nandy, S.C. and Bhattacharya, B.B. *On finding an empty staircase polygon of largest area (width) in a planar point-set*. Comput. Geom.: Theory and Applications. (2003). 143–171
- Roy, U. and Dasari, V.R. *Implementation of a polygonal algorithm for surface-surface intersections*. Comput. Ind. Engng. (1998). 399–412
- Kai Zhou, Lorenz T. Biegler. *Computer Aided Chemical Engineering*. (2013)
- Ronal, L. Rardin and Reha, Uzsoy. *Experimental Evaluation of Heuristic Optimization Algorithms: A Tutorial.* Journal of Heuristics. (2001). 261–304
- Alves, J.C., Ferreira, J.C., Oliveira, J.F., Ferreira, J.S., Matos, J.S. and Albuquerque, C. *Fafner – accelerating nesting problems with FPGAs. Proceedings of the 7th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCMM 1999).* (1999). 168–177
- Lutfiyya, H., McMillin, B., Poshyanonda, P. and Dagli, C. *Composite stock cutting through simulated annealing*. Math. Comput. Modelling. (1992). 57–74
- Ramesh Babu, A., Ramesh Babu, N. *A generic approach for nesting of 2-D parts in 2-D sheets using genetic and heuristic algorithms.* Indian Institute of Technology, India (2001). 879–891

- Kierkosz, I., Luczak, M. *A one-pass heuristic for nesting problems.* Koszalin University of Technology. (2019). 75–453
- Xie, S.Q. and Tu, Y.L., Liu, J.Q. and Zhou, Z.D. *An integrated and concurrent approach for compound sheet metal cutting and punching*. Int. J. Prod. Res. (2001). 1095–1112
- Bennel, J.A., Song, X. *A comprehensive and robust procedure for obtaining the nofit polygon using Minkowski sums*. Comput. Oper. Res. (2008). 267–281
- Burke E.K., Hellier R.S.R., Kendall G., Whitwell G. *Complete and robust no-fit polygon generation for the irregular stock cutting problem*. Eur. J. Oper. Res. (2007). 27–49
- Gomes A.M., Oliveira J.F., *A 2-exchange heuristic for nesting problems*, Eur. J. Oper. Res., (2002). 359–370
- Oliveira J.F., Gomes A.M., Ferreira J.S., Topos. *A new constructive algorithm for nesting problems, OR Spektrum*, Vol. 22, Springer-Verlag, (2000). 263–284
- Johnson, A. (2014) *Clipper - an open source freeware library for clipping and offsetting lines and polygons.* Prieiga per internetą: http://www.angusj.com/delphi/clipper.php
- Wein, R., Baram, A., Fogel, E., Flato, E., Hemmer, M., Morr, S. *CGAL 5.4 - 2D Minkowski Sums.* Prieiga per internetą: https://doc.cgal.org/latest/Minkowski_sum_2/index.html
- Valle, A.M., Queiroz, T.A., Miyazawa, F.K., Xavier, E.C. *Heuristics for two-dimensional knapsack and cutting stock problems with items of irregular shape*. Expert Syst. Appl., (2012). 12589–12598.

PRIEDAI

1 priedas. CAN sistemos vartotojo sąsaja (International Journal of Computer Integrated Manufacturing, 2007).