Liet. matem. rink., **47**, *spec. nr.*, 2007, 467–472

Pjaustymo uždavinio metaeuristinių algoritmų analizė

Jonas POKŠTAS, Narimantas LISTOPADSKIS (KTU)
el. paštas: jonas_pokstas@yahoo.com, narlis@ktu.lt

Reziumė. Darbe nagrinėjamas negiljotinio pjautymo uždavinio sprendimas euristiniais ir metaeuristiniais algoritmais. Realizuoti šie euristiniai algoritmai – „Žemiausio kairėn užpildymo“ (ŽKU), „Geriausiai tinkamo“ (GT), „Žemiausio tarpo“ (ŽT), kuris yra originali GT metodo modifikacija. Šie algoritmai įvesti į genetinio algoritmo schemą. Apžvelgiant euristinių algoritmų (ŽT, GT, ŽKU) rezultatus paaiškėjo, kad sudėtingėjant uždaviniui pateikiamas vis geresnis sprendinys.

Pjaustymo uždavinys. Pjaustymo uždavinio klasifikacija suformuluota bei pateikta [1]. Tokių užduočių sprendimo vykdymo metu, atliekamas optimizavimas. Kadangi paprastai susiduriama su diskrečiais dydžiais ir optimizavimo procesas grindžiamas elementariomis kombinatorinėmis (perstatymo ir pan.) operacijomis, tai pjaustymo uždavinys priskiriamas *kombinatorinio optimizavimo* užduočių klasei [1].

Šiems pjaustymo uždaviniams yra būdinga tai, kad didėjant jų apimčiai, visais atvejais tiksliai išspręsti juos per trumpą sprendimo laiko tarpą pasidaro nebeįmanoma. Todėl algoritmų sudėtingumo teorijoje tokios užduotys priskiriamos *NP sunkių* problemų sudėtingumo klasei [2]. Be to, kai kurių pjaustymo užduočių sprendimo variantuose reikalaujama operuoti sveikaisiais (arba natūriniais) skaičiais, taigi susidurima su *sveikaskaičio programavimo* uždaviniu. Dėl šių priežasčių yra ribotas tikslių metodų panaudojimas, t.y. šie metodai taikomi, tačiau nedidelės apimties užduotims spręsti.

Sparčiai besivystant kompiuterinėms technologijoms, pradėtas simuliuoti pjaustymo procesas kompiuterio pagalba. Todėl atsivėrė plačios galimybės taikyti *euristinius* ir *metaeuristinius* sprendimo metodus, kurie sėkmingai panaudoti apytiksliam pjaustymo uždavinių sprendimui.

Uždoties matematinė formuluotė. Nagrinėsime tokį pjaustymo uždavinį, kai lakštai iš kurių bus pjaustomos figūros ir pačios figūros yra stačiakampiai (kvadratai). Atliekami pjūviai bus negiljotininiai, t.y. vykdant pjūvį, galimas jo krypties keitimas. Taip pat pjūvio vykdymo kryptys bus lygiagrečios vienai iš pjaustomo lakšto kraštinių. Papildomai atsiribosime ir nuo pjaustymo siūlės storio, tarsime, kad jis yra nykstamai mažas ir musų nagrinėjamo uždavinio sprendimui įtakos neturi.

Žinomas stačiakampių užsakymas, kuriuos reikės išpjauti:

$$\{(w_j, l_j, q_j)|j, w_j, l_j, q_j \in N\},$$

čia w_j – plotis, l_j – ilgis, q_j – kiekis, $j = 1, \ldots, n$ ir lakštų *(W,L)*, čia W – plotis, L – ilgis (pagrindinio stačiakampio, iš kurio bus išpjautos figūros). Organizuoti lakštų išpjaustymą taip, kad atliekų kiekis būtų mažiausias (arba kiek galima mažesnis). Pjaustymo metu galimas stačiakampio pasukimas 90 laipsnių kampu.

Tarkime, $s_{i,j}$ $(i = 1, \ldots, m,\ j = 1, \ldots, n)$ reiškia kiekį išpjautų detalių (w_j, l_j), įvykdžius šabloną i, kurį suprasime kaip tam tikrą vieno lakšto išpjovimo būdą. Vienu šablonu galima išpjauti keletą lakštų, kurių skaičių žymėsime u_i.

$$\min \sum_{i=1}^{m} \Big(WL - \sum_{j=1}^{n} s_{i,j} w_j l_j \Big) u_i, \tag{1}$$

$$\sum_{i=1}^{m} s_{i,j} u_i = q_j, \quad j = 1, \ldots, n, \tag{2}$$

$$w_i \leqslant W, \quad l_i \leqslant L, \quad i = 1, \ldots, n, \tag{3}$$

$$u_i \in N, \quad i = 1, \ldots, m. \tag{4}$$

1 tikslo funkcijos reikšmė nurodo minimalų atliekų kiekį, įvykdžius pjaustymą. Apribojimu 2 reikalaujama, kad išpjautų stačiakampių kiekis atitiktų užsakymo kiekius q_j. 3 apribojimo prasmė yra tokia, kad pjaustomos detalės neišeitų iš lakšto ribų. Iš 4 išraiškos išplaukia, kad tai yra sveikaskaičio programavimo uždavinys. Taip pat akivaizdus papildomas reikalavimas, kad išdėstytos lakšte (išpjautos) detalės nesikirstų.

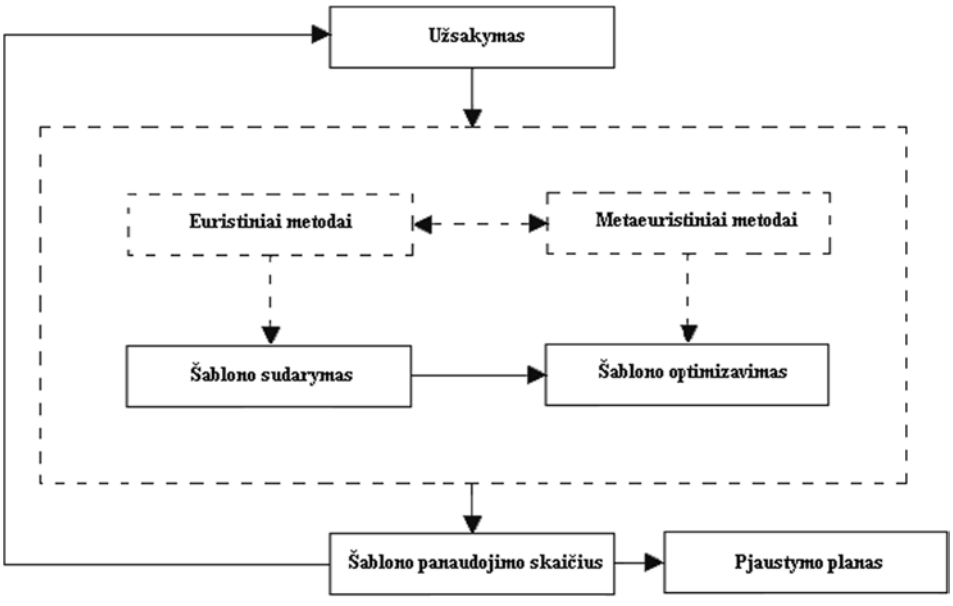
Iš uždavinio apibrėžimo išplaukia, kad lakštų (pradinių žaliavų) kiekis yra be galo didelis. Nagrinėsime taip pat situaciją, kai pradinių žaliavų kiekis yra ribotas, todėl įvesime dar vieną apribojimą:

$$\sum_{i=1}^{m} u_i \leqslant k, \quad k \in N. \tag{5}$$

Apribojimu 5 reikalaujama, kad sudarytų šablonų bendras skaičius neviršytų tam tikrų iš anksto turimų lakštų atsargų.

Užduoties sprendimo schema. Duotam figūrų užsakymui išpjauti, sudarysime pjaustymo šablonus (1 pav.). Pjaustymo šablonas suprantamas kaip tam tikras medžiagos lakšto išpjovimo būdas. Pjaustymo planas yra visų sudarytų šablonų visuma, kuri reprezentuoja užduoties sprendimą duotajam užsakymui. Šablono sudarymo užduotį vykdys euristiniai metodai. Svarbu suvokti tai, kad euristinių pjaustymo metodų veikimas pagrįstas tam tikromis iš anksto nustatytomis taisyklėmis, kurias vykdant nuosekliai arba su tam tikru ciklišku gaunamas rezultatas. Dauguma euristinių pjaustymo metodų nedaro jokio sprendinių perrinkimo, nenagrinėja jų paieškos erdvės ir nelygina jų tarpusavyje, todėl šių metodų negalime vadinti optimizavimo algoritmais, todėl papildomai verta nagrinėti ir metaeuristinius algoritmus, kurie turi šias savybes. Nepaisant to, sprendinio optimalumui euristiniai algoritmai daro didelę įtaką, nes jų pagalba konstruojamas pats sprendinys ir pateikiamas jo kokybės įvertinimas.

Kadangi euristiniai metodai labai specifiški, o metaeuristiniai pakankamai universalūs, tenka juos apjungti, nes metaeuristiniai algoritmai patys savarankiškai negalėtų išspręsti užduoties. Tuo tarpu, euristiniai gali tai padaryti. Todėl uždavinio sprendimas gali būti vykdomas ir be optimizavimo, tiesiog sudarant šabloną ir jį priskiriant pjaustymo planui, kartais netgi gaunami panašūs rezultatai lyginant su vykdymu optimizuojant.



1 pav. Bendra pjaustymo uždavinio sprendimo schema.

Šablono panaudojimo skaičiaus nustatymas. Užsakymo detalių asortimentas būna labai įvairus. Gali tekti susidurti su dideliais kiekiais stačiakampių, kurių ilgis ir plotis vienodi, tokiu atveju bus didelis kiekis tų pačių šablonų. Kyla būtinybė tokius šablonus apjungti, t.y. panaudoti keletą kartų. Todėl sukonstruotam šablonui nustatysime jo panaudojimo skaičių.

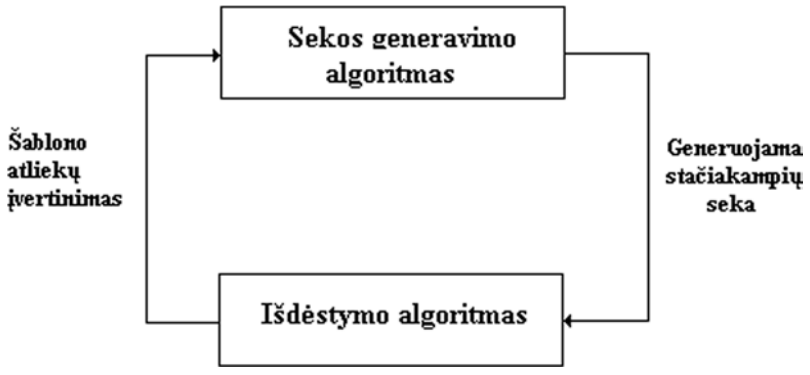
Šabloną taikysim lakštams remdamiesi tokia logika:

- tarkime, kad sudaryto šablono stačiakampių kiekiai $a_1, a_2, \ldots, a_k$. Čia $k \leqslant n, i = 1, 2, \ldots, n$, kur n – stačiakampių tipų skaičius užsakyme;
- atitinkamai parenkame užsakymo detalių kiekius pagal tai, kokie detalių tipai yra išpjauti šablone $q_{i_1}, q_{i_2}, \ldots, q_{i_k}$;
- apskaičiuojame $m = \lfloor \min\{\frac{q_{i_j}}{a_j}\} \rfloor$.Čia $j = 1, 2, \ldots, k$, $\lfloor\rfloor$ – apvalinimo žemyn operacija;
- vadinasi, sudarytą šabloną lakštams pjauti taikysim m kartų;
- atitinkamai sumažinami užsakymo kiekiai $q_{i_j} = q_{i_j} - m \cdot a_j$.

Hibridinio algoritmo principas. Pjaustymo šablono sudarymas grindžiamas dviem etapais:

- generuojama stačiakampių seka;
- stačiakampiai išdėstomi ant medžiagos lakšto.

Šiuos du etapus vykdys atskiri algoritmai, o galutinis rezultatas (pjaustymo šablonas) bus abiejų algoritmų darbo pasekmė. Nepaisant to, kad bus sprendžiamos iš pažiūros dvi skirtingos užduotys tuo pačiu metu, tačiau algoritmų veikimas prik-



2 pav. Hibridinio algoritmo schema.

lausys vienas nuo kito. Todėl iškyla būtinybė apjungti algoritmus į vieną, t.y. juos hibridizuoti.

Taigi pirmasis algoritmas generuos tam tikrą sprendinio variantą – šabloną, o antrasis jį interpretuos ir pateiks jo kokybinį įvertinimą, kuris apibrėžiamas formule:

$$a_i = W \cdot L - \sum_{j=1}^{n_i} w_j \cdot l_j \tag{6}$$

Čia a_i – atliekų plotas, gautas sukonstravus i-tąjį šabloną, n_i – i-tąjį šabloną sudarančių stačiakampių skaičius, $n_i \leqslant n$, kur n – bendras stačiakampių skaičius (viso užsakymo). w_i, l_i, W, L atitinkamai apibrėžti užduoties matematinėje formuluotėje.

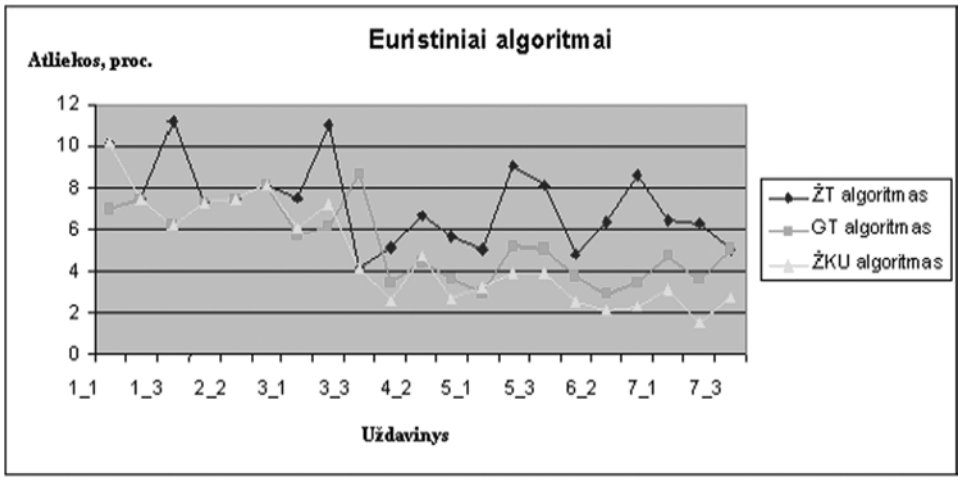
Realizuoti algoritmai. Darbo metu realizuoti šie euristiniai algoritmai – „Žemiausio kairėn užpildymo“ (ŽKU), „Geriausiai tinkamo“ (GT), „Žemiausio tarpo“ (ŽT), kuris yra originali GT metodo modifikacija [3]. Taip pat metaeuristinis hibridinis genetinis algoritmas apjungtas su ŽKU ir ŽT euristika, kai pradinės sąlygos generuojamos atsitiktinai (GA + ŽKU), (GA + ŽT) ir kombinuotai su ŽKU, (ŽKU + GA + ŽKU).

Paminėsime tik tiek, kad pavienių euristinių algoritmų veikimas pagrįstas, stačiakampių išrikiavimu atitinkamai pagal ilgį ir plotį mažėjimo tvarka. Tokiu būdu stengiamasi pradžioje patalpinti ant lakšto didesnes figūras, o paskui atsiradusį laisvą plotą užpildyti mažesnėmis.

Algoritmų palyginimas. Tyrimui pasirinktos užduotys buvo suformuluotos Hopper ir Turton [4], jų optimalūs sprendiniai žinomi „a priori“. Tai vienas iš plačiausiai naudojamų uždavinių rinkinių pjaustymo algoritmams testuoti. Euristiniai algoritmai buvo patikrinti su visais uždaviniais (21 užd.). Apskaičiuota procentinė atliekų ploto dalis:

$$S_{proc} = \frac{S}{W \cdot L} \cdot 100\%. \tag{7}$$

Čia S – gautas atliekų plotas (m^2), išpjovus medžiagos lakštą, čia W – lakšto plotis, L – lakšto ilgis. Gauti rezultatai (3 pav.).



3 pav. Euristinių algoritmų palyginimas.

Taip pat atliktas bendras algoritmų palyginimas, nagrinėta 13 pirmųjų uždavinių. Genetiniam algoritmui nustatyta vidutinė procentinė atliekų ploto dalis:

$$MS_{proc} = \frac{MS}{W \cdot L} \cdot 100\% \tag{8}$$

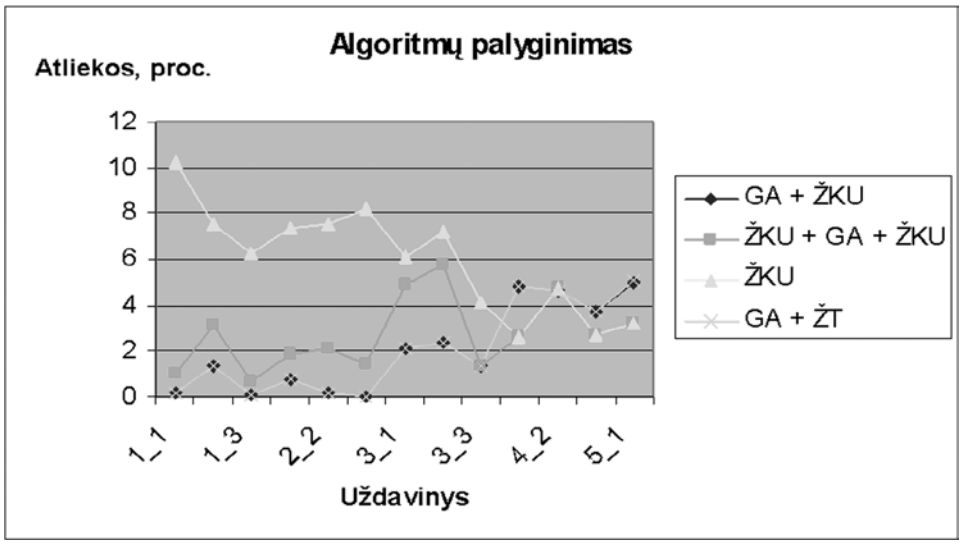
Čia $MS = \frac{\sum_{i=1}^{N} S_i}{N}$, kur N – stebėjimų skaičius, S_i – atliekų plotas (m^2), gautas stebėjimo momentu i, o W ir L atitinkamai lakšto plotis ir ilgis. Mūsų atveju $N = 10$. Gauti rezultatai (4 pav.).

Apžvelgiant euristinių algoritmų (ŽT, GT, ŽKU) rezultatus (3 pav.) paaiškėja, kad sudėtingėjant uždaviniui pateikiamas vis geresnis sprendinys. Kai užsakymo dydis buvo mažiau nei 30 figūrų, pasiekta atliekų procentinė ploto dalis 7–10 proc., tuo tarpu, kai užsakymas padidėjo iki 70 figūrų ir daugiau, gauta atliekų procentinė ploto dalis 2–5 proc.

Tuo tarpu apjungus ŽT ir ŽKU algoritmus su genetiniu metodu (GA + ŽKU, GA + ŽT, ŽKU + GA + ŽKU) gauti rezultatai byloja visiškai ką kitą. Sprendžiant mažos apimties uždavinius, šie algoritmai pademonstravo daug geresnius rezultatus, nei pavieniai euristiniai (3 ir 4 pav.). Tuo tarpu, kai uždavinio apimtis viršijo 50 stačiakampių, geresni rezultatai jau gauti sprendžiant euristiniais metodais (ŽKU).

Buvo manoma, kad parinkus pradinės sąlygas hibridiniam GA + ŽKU algoritmui, panaudojant vieną iš euristinių metodų (ŽKU) pagerės sprendinio kokybė. Tačiau eksperimentai paneigė šią prielaidą (4 pav.). Pradinė populiacija buvo konstruojama tokiais būdais: vienas sprendinys sugeneruojamas ŽKU metodu, o kiti atsitiktinai; arba visi atsitiktinai. Lyginant rezultatus, esant mažos apimties uždaviniams, lyderiavo atsitiktinis parinkimas.

Dar vienas įdomus pastebėjimas (4 pav.) GA + ŽT hibridinis algoritmas parodė tuos pačius rezultatus, kaip ir GA + ŽKU (išskyrus uždavinį 5_1), nors pavieniui ŽT



4 pav. Bendras algoritmų palyginimas.

ir ŽKU metodai veikė labai skirtingai (3 pav.).Tai paaiškinama panašia ŽT ir ŽKU veikimo logika. Nepaisant to, kad ŽT algoritmas turi trūkumų lyginant su ŽKU tie trūkumai yra kompensuojami genetinio algoritmo veikimu [3].

Literatūra

1. H. Dyckhoff, A typology of cutting and packing problems, *European Journal of Operational Research*, **44**, 145–159 (1990).
2. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco, CA (1979).
3. J. Pokštas, *Pjaustymo uždavinio algoritmų realizacija ir tyrimas*, magistro tezės, KTU (2007).
4. E. Hopper, B.C.H. Turton, A genetic algorithm for a 2D industrial packing problem, *Comput. Indust. Engrg.*, **37**, 375–378 (1999).

SUMMARY

J. Pokštas, N. Listopadskis. Analysis of cutting stock problem metaheuristic algorithms

The analysis of cutting stock problem and heuristic and metaheuristic algorithms for solving it are presented in this paper.

Keywords: combinatorial optimization, cutting stock problem.