Liet. matem. rink, **46***, spec. nr.*, 2006, 384–389

Pjaustymo uždavinio algoritmo analizė

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

Matematinė užduoties formuluotė

Žinomas užsakymas figūrų: $\{(w_j, l_j, q_j) \mid j, w_j, l_j, q_j \in N\}$, čia w_j – plotis, l_j – ilgis, q_j – kiekis, $j = 1, \ldots, n$, ir lakštų (W, L), čia W – plotis, L – ilgis, iš kurių bus išpjautos figūros. Organizuoti lakštų išpjaustymą taip, kad atliekų kiekis būtų mažiausias. Lakštai ir figūros yra stačiakampiai.

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.

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

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

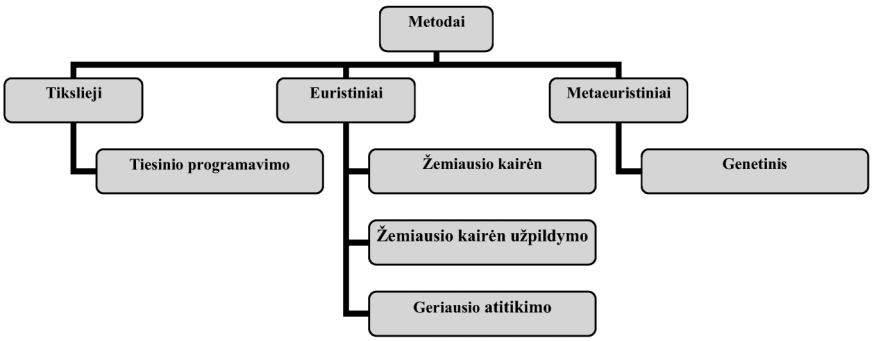
$$w_i \leqslant W, \quad l_i \leqslant L, \quad i = 1, \ldots, n, \tag{3}$$

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

(1.1) reikšmė nurodo minimalų atliekų kiekį, įvykdžius pjaustymą. Apribojimu (1.2) reikalaujama, kad išpjautų stačiakampių kiekis atitiktų užsakymo kiekius q_j. (1.3) apribojimo prasmė yra tokia, kad pjaustomos detalės neišeitų iš lakšto ribų. Iš (1.4) išraiškos išplaukia, kad tai yra sveikaskaičio programavimo uždavinys. Be to akivaizdus papildomas reikalavimas, kad išdėstytos lakšte (išpjautos) detalės tarpusavyje neturėtų bendro ploto, nesikirstų.

Metodų apžvalga

Literatūroje randamos trys pagrindinės pjaustymo uždavinio metodų sprendimo klasės. Tikslieji metodai dažniausiai garantuoja optimalų problemos sprendimą, tačiau sudėtingiems uždaviniams reikalauja pakankamai ilgo sprendimo laiko. Euristiniai metodai retai randa optimalų, bet dažnai greičiau generuoja pakankamai gerą bei priimtiną sprendimą. Euristiniai metodai yra labai priklausomi nuo problematikos srities, tai reiškia, kad jie naudoja informaciją apie konkrečias užduotis, kurioms jie



1 pav. Metodų klasifikacija.

yra sukurti spręsti. Metaeuristiniai metodai turi savybę nesuklysti lokaliai optimalaus sprendinio prasme, kaip tai gali atsitikti su tradicinėmis euristikomis. Metaeuristiniuose metoduose sprendinio paieškos procesas dažnai reguliuojamas žemesnio lygio euristikos.

Pjaustymo uždavinys yra priskiriamas NP–Complete sudėtingumo klasei, tai reiškia, kad neegzistuoja polinominio laiko algoritmas, kuris visais atvejais tiksliai išspręstų minėtą uždavinį, t.y. visą laiką rastų globaliai optimalų sprendinį. Atsižvelgiant į šią aplinkybę, tiksliųjų metodų panaudojimas yra gana ribotas, jie yra dažniausiai aptinkami sprendžiant nedidelės apimties uždavinius. Čia atsiveria galimybė taikyti kombinatorinio optimizavimo instrumentus, tokius kaip euristiniai ir metaeuristiniai metodai, vieną iš jų pristatysime šiame straipsnyje.

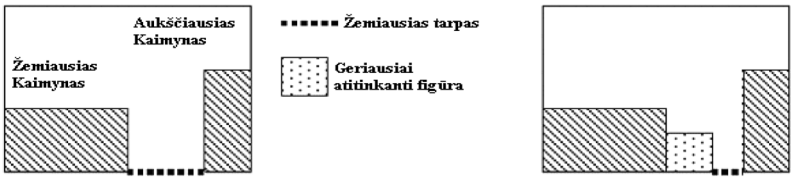
„Geriausio atitikimo“ metodas

Šis euristinis algoritmas taikomas negiljotininio pjaustymo uždaviniams (kai yra leistinas pjūvio krypties keitimas) spręsti. Metodo esmė – rasti žemiausią tarpą (figūrai įdėti) ir jį geriausiai atitinkantį stačiakampį. Jeigu stačiakampis idealiai neatitinka tarpo, vadinasi jį galima patalpinti trimis būdais:

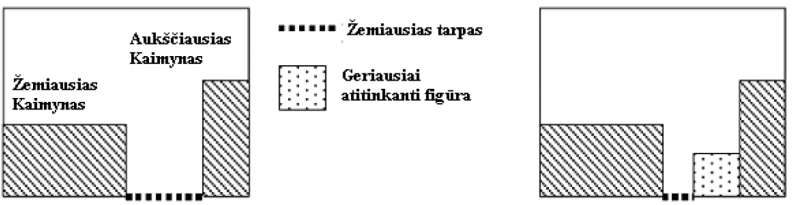
1) Kairiausio kaimyno idėja – figūra visada dedama tarpo kairėje pusėje.
2) „Žemiausio kaimyno“ – figūra visada dedama toje tarpo pusėje, kur yra žemesnis kaimynas (2 pav.).
3) „Aukščiausio kaimyno“ – figūra visada dedama toje tarpo pusėje, kur yra aukštesnis kaimynas (3 pav.).

Pjaustomas lakštas reprezentuojamas vienmačiu lygių masyvu, kurio indeksas lygus koordinatei, o paties elemento reikšmė – jau išdėstytui detalių aukščiui (4 pav.). Jeigu nustatytam tarpui nerandame atitinkamo stačiakampio, tuomet esantis plotas interpretuojamas kaip atliekos, o tarpą atitinkantys lygių masyvo elementai padidinami iki žemiausio kaimyno reikšmės (5 pav.).

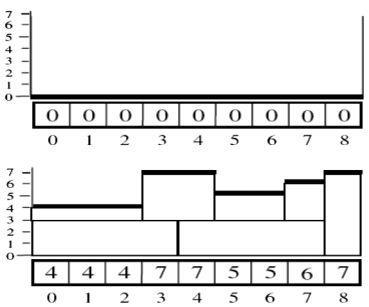
Taigi optimizavimo kriterijus yra atliekų kiekis vienam lakštui: $WL - \sum_{i=1}^{n_0} w_i l_i$, čia $n_0 \leqslant n$ išpjautų detalių kiekis lakšte.



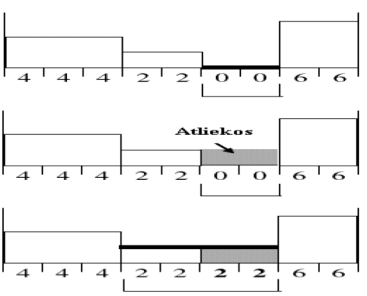
2 pav. „Žemiausio kaimyno“ užpildymo idėja.



3 pav. „Aukščiausio kaimyno“ užpildymo idėja.



4 pav. Lakšto reprezentacija.

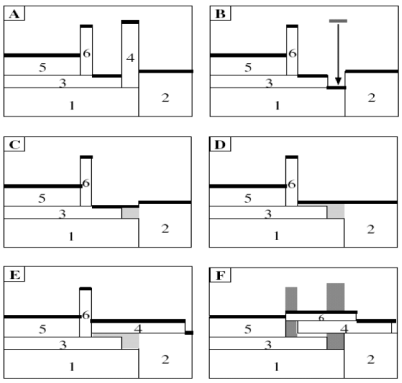


5 pav. Tarpo pavertimas atliekomis.

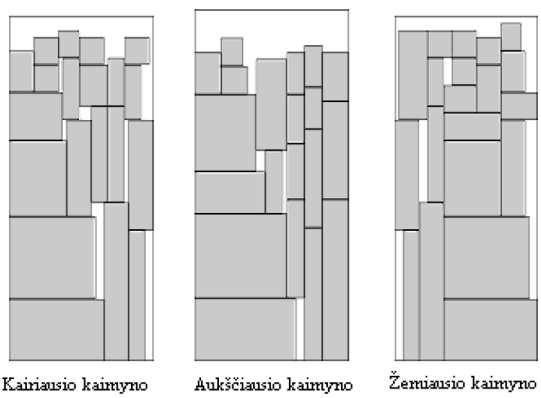
Metodo modifikacijos

Skirtingai nuo pačio pirmojo autorių pasiūlyto varianto, šis algoritmas buvo modifikuotas, papildomai buvo nagrinėta situacija, kai išpjaustome visus stačiakampius, t.y. gauname jų dėstinį, būtų verta patikrinti, ar pačiame aukščiausiame lygyje negauname smarkiai išsikišusių detalių, kurios gadina dėstinio kokybę. Surandame aukščiausią stačiakampį, tuomet patikriname ar jo plotis yra didesnis už jo ilgį, jeigu taip, tuomet jis yra išiimamas iš bendro dėstinio (iš lakšto). Tuomet stačiakampis yra pasukamas taip, kad jo ilgis įgautų pločio reikšmę ir bandoma jį idėti anksčiau aprašytais būdais. Procesas tęsiamas tol, kol įmanoma pagerinti dėstinio kokybę. Kitu atveju, jeigu rasto stačiakampio ilgis didesnis arba lygus jo pločiui, dėstinio kokybės pagerinti negalėsime, todėl procesas stabdomas (6 pav.).

Metodo autoriai nagrinėjo situaciją tik vieno lakšto atveju. Kadangi mes sprendžiam uždavinį, kur dažnai susidaro situacija, kad vieno medžiagos lakšto neužtenka užsaky-



6 pav. Pjaustymo šablono modifikavimo schema.

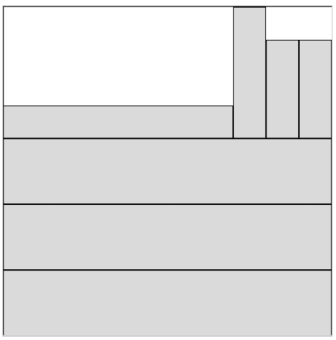


7 pav. Skirtingos tarpo užpildymo idėjos.

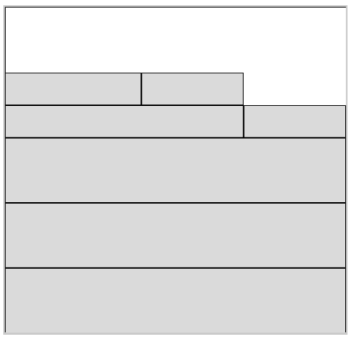
mui įvykdyti, todėl tenka organizuoti daugelio lakštų pjaustymą. Kada kurti naują šabloną? Kiek kartų pritaikyti sukurtąjį?

Šabloną taikysim plokštėms remdamiesi tokia logika:

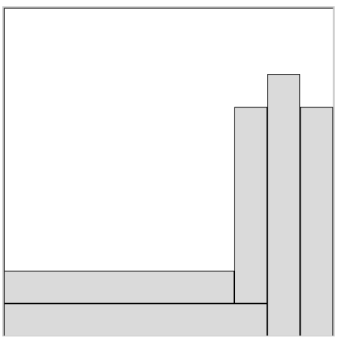
1. Sukursime šabloną vienai plokštei, remdamiesi šablono sudarymo procedūra.
2. Tarkime, kad sudaryto šablono detalių kiekiai $a_1, a_2, \ldots, a_{n_0}$.
3. Atitinkamai parenkame užsakymo detalių kiekius (pagal tai, kokie detalių tipai yra šablone) $q_1, q_2, \ldots, q_{n_0}, n_0 \leqslant n$.
4. Apskaičiuojame $k = \left\lfloor \min \left\{ \frac{q_i}{a_i} \right\} \right\rfloor$, čia $i = 1, \ldots, n_0$; $\lfloor\rfloor$ – apvalinimo žemyn operacija.
5. Vadinasi, sudarytą šabloną plokštėms taikysim k kartų.



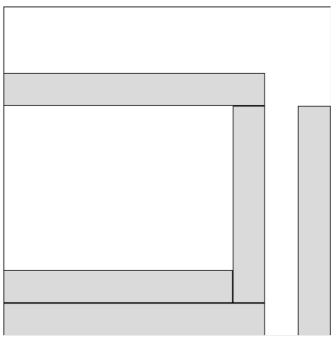
8 pav. Pradinis variantas.



9 pav. Modifikuotas variantas.



10 pav. Pradinis variantas.



11pav. Modifikuotas variantas.

Metodo analizė

„Geriausio atitikimo“ algoritmas buvo realizuotas programiškai panaudojant pradinę autorių idėją ir pritaikant ankščiau minėtas modifikacijas.

Tas pats uždavinys spręstas pagal tris skirtingas tarpo užpildymo idėjas (7 pav.). Šiuo atveju lygtais geriausias pjaustymo šablonas būtų sudarytas, remiantis aukščiausio kaimyno užpildymo idėja, tačiau negalima daryti apibendrintų išvadų iš šio vieno pavyzdžio apie tai, kuri taktika geresnė.

Pateiksime atliktą analizę: KK – kairiausio kaimyno, AK – aukščiausio kaimyno, MK – mažiausio kaimyno nišos užpildymo taktikų gautieji maksimalūs dėstinių aukščiai (1 lentelė).

Kaip matome sprendinio kokybė labai priklauso nuo norimų išpjauti stačiakampių kiekio ir formų bei pačių lakštų.

Modifuokuoto algoritmo atveju tam tikros situacijos sprendžiasi geriau, nei pradinio varianto (8 ir 9 pav.). Tačiau gali pasitaikyti ir tokių atvejų, kur pradiniu sprendimo būdu gauti rezultatai yra geresni (10 ir 11 pav.).

1 lentelė

Lakšto (ilgis; plotis)	Stačiakampių skaičius	KK	AK	MK
35; 50	18	38	38	**37**
35; 50	20	48	**44**	50
85; 85	35	**81**	**81**	82
85; 85	41	81	**79**	81
85; 85	43	**80**	82	**80**

Literatūra

1. E.K. Burke, G. Kendall, G. Whitwell, *A New Placement Heuristic for the Orthogonal Stock-Cutting Problem* (2004).
2. J. Karelahti, *Solving the Cutting Stock Problem in the Steel Industry* (2002).

SUMMARY

J. Pokštas, N. Listopadskis. The analysis of algorithm for cutting stock problem

The analysis of heuristic algorithm for solving two dimensional cutting stock problem is presented in this paper.

Keywords: combinatorial optimization, two dimensional cutting stock problem, heuristic algorithm.