KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Mindaugas Kvederas

Giljotininio pjaustymo uždavinio euristinių algoritmų tyrimas

Tiriamasis darbas

Vadovas

Prof. J. Mockus

KAUNAS, 2007

TURINYS

Įvadas

Visais laikais žaliavų taupymo problema buvo opiu klausimu. Optimaliai suplanavus gamybą mažinamos išlaidos žaliavų pirkimui, gerėja įmonės rodikliai ir taupomi gamtos ištekliai.

Išdėstymo uždaviniai neapsiriboja tik vienmatės ar dvimatės erdvės uždaviniais. Jiems taip pat priklauso ir talpaus daiktų išdėstymo būdai, pavyzdžiui: krovininio laivo triumo optimalus panaudojimas arba taros projektavimas optimaliam prekių išdėstymui.

Racionalaus supjaustymo uždaviniai aprašomi panašiais matematiniais modeliais kaip ir išdėstymo uždaviniai. Pagrindinis supjaustymo uždavinių tikslas — racionalus žaliavos panaudojimas, t.y.

Galimi sprendimo metodai gali būti skirstomi į tris pagrindines klases: *tikslieji, euristiniai ir metaeuristiniai*. *Tikslieji algoritmai* garantuoja rasti optimalų sprendinį nesvarbu kokio pobūdžio kombinatorinio optimizavimo problema būtų. Tačiau egzistuoja tokių uždavinių, kurių sprendimo laikas gali būti labai ilgas. Šiuo atveju reikia priimti kompromisą tarp sprendimo tikslumo ir laiko jam rasti. *Euristiniai metodai* būtent tam ir buvo sukurti. Jie 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 problemas, kurioms yra sukurti. *Metaeuristiniai metodai* turi savybę nesuklysti lokalaus optimalaus sprendinio prasme, kaip tai gali atsitikti su tradicinėmis euristikomis. Metaeuristiniuose metoduose sprendinio paieškos procesas dažnai reguliuojamas žemesnio lygio euristikos.

Bendroji dalis

Kombinatorinio optimizavimo samprata

Daugumos praktinių ir teorinių optimizavimo problemų svarba susideda iš geriausios kintamųjų konfiguracijos paieškos tam, kad būtų pasiekti kažkokie tikslai. Spręsdami kombinatorinio optimizavimo problemą, ieškosime optimalaus sprendinio baigtinėje arba skaičiojoje, t.y. diskrečiojoje aibėje, todėl vietoj kombinatorinio dažnai vartojama diskretaus optimizavimo sąvoka.

Apibrėžimas. Kombinatorinė optimizavimo problema *KOP* = (*X*, *P*, *Y*, *f*, *extr*) formaliai apibrėžiama tokiu būdu, kur

- *X* diskrečioji sprendinių aibė, kurioje apibrėžtos *f* ir *P*
- *P* sprendinių, tenkinančių apribojimus, predikatas $P : X \to \{0,1\}$, čia *P* = *0*, jei apribojimai netenkinami ir *P* = *1* atvirkščiai.
- Y tekinančių apribojimus sprendinių aibė.
- Tikslo funkcija *f*, kur $f : Y \to R$
- *extr* – tikslo funkcijos ekstremumas (minimumas ar maksimumas)

Y dažnai vadinama paieškos arba sprendinių, tenkinančių apribojimus, erdve. Tam, kad išspręsti kombinatorinio optimizavimo problemą, reikia rasti tokį sprendinį $y^* \in Y$, prie kurio tikslo funkcijos reikšmė butų minimali (arba maksimali), t.y.

$f(y^*) \le f(y), \forall y \in Y$. Tuomet y^* vadinamas problemos *KOP* globaliai optimaliuoju sprendiniu ir aibė $Y^* \subseteq Y$ - globaliai optimalių sprendinių aibe.

Problemų sudėtingumo klasės

Tiuringo mašina

Tiuringo mašina - abstraktus kompiuterio vykdymo modelis, kurį 1936 metais sukūrė Alanas Tiuringas, norėdamas matematiškai apibrėžti algoritmus.Tiuringo mašina - tai automatas, vykdantis begalinę rūšiuotų instrukcijų seką, bei įsimenantis būseną. Skirtingų instrukcijų bei būsenų kiekiai - baigtiniai.

Tiuringo mašiną sudaro:

- *Juosta*, padalinta į langelius, kuriuose gali būti vienas iš naudojamos abėcėlės simbolių. Abėcėlę sudaro *tuščias* simbolis ('0') ir vienas ar daugiau kitų simbolių. Į neužpildytus langelius žiūrima kaip užpildytus tuščiu simboliu.
- *Galvutė*, kuri skaito ir rašo į langelį, taip pat gali judėti į abi puses.
- *Būsenų registras*, saugantis automato būseną. Būsenų skaičius baigtinis, pradinė būsena visada apibrėžta.
- *Veiksmų lentelė*, nusakanti kokį simbolį rašyti, į kurią pusę per vieną langelį pajudėti ('K' į kairę, 'D' į dešinę), taip pat kokia bus nauja būsena priklausomai nuo esamos būsenos ir perskaitytos langelio reikšmės. Jei veiksmų lentelėje nėra aprašyto veiksmo dabartinei būsenai ir langelio reikšmei, mašina baigia darbą.

Formalus aprašymas. Vienos juostos Tiuringo mašina

Vienos uostos Tiuringo mašiną galima aprašyti kaip $M = (Q,\Gamma,\Sigma,s,b,F,\delta)$, kur

- Q yra baigtinė būsenų aibė
- Γ yra baigtinė juostos abėcėlės aibė
- Σ baigtinė pradinė abėcėlė $(\Sigma \subseteq \Gamma)$
- $s \in Q$ yra pradinė būsena
- b yra tuščias simbolis $(b \in \Gamma \setminus \Sigma)$
- $F \subseteq Q$ yra aibė galutinių arba priimamų būsenų
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{K, D\}$ yra dalinė funkcija, nusakanti perėjimą; K yra postūmis kairėn, D - dešinėn.

k-juostų Tiuringo mašina

Naudojant k juostų, Tiuringo mašiną taip pat galima aprašyti kaip $M = (Q,\Gamma,\Sigma,s,b,F,\delta)$, tik δ funkcija skirsis:

$\delta : Q \times \Gamma^k \to Q \times (\Gamma \times \{K, D, S\})^k$; S - reiškia kad juosta paliekama toje pačioje pozicijoje.

Jei kiekvienai simbolio ir būsenos porai yra daugiausiai viena reikšmė veiksmų lentelėje, Tiuringo mašina vadinama *deterministine*, priešingu atveju - *nedeterministine*.

P, NP ir NP - Complete

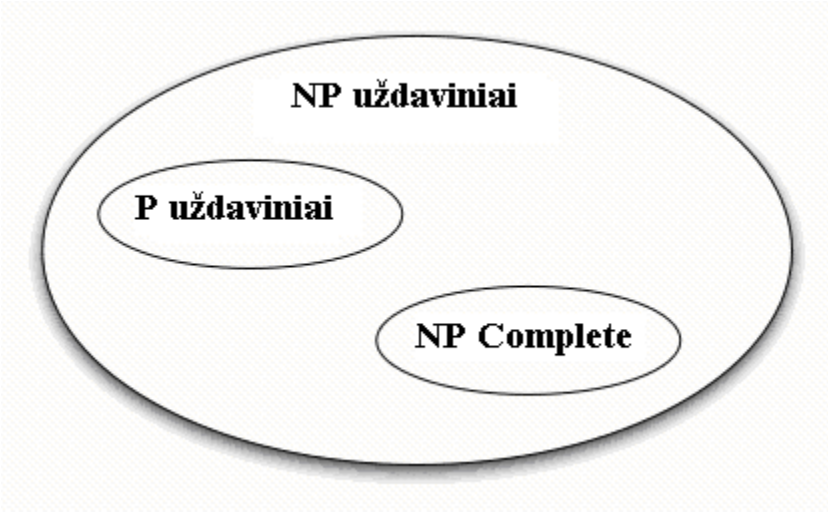
Sudėtingumo klasė yra aibė problemų susietų su panašiu sudėtingumu. Skiriamos dvi pagrindinės klasės *P* ir *NP*.

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

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

Akivaizdu, kad $P \subseteq NP$, tačiau ar $P = NP$? Daugelis mano, kad atsakymas yra neigiamas. Tačiau dar ši problema nėra įrodyta ir tam, kuriam pavyks, tai padaryti, yra numatyta 1 milijono dolerių premija.

Mokslininkai tiki, kad egzistuoja sąryšis tarp šių klasių būtent toks:



Pav. 1 Sudėtigumo klasių diagrama

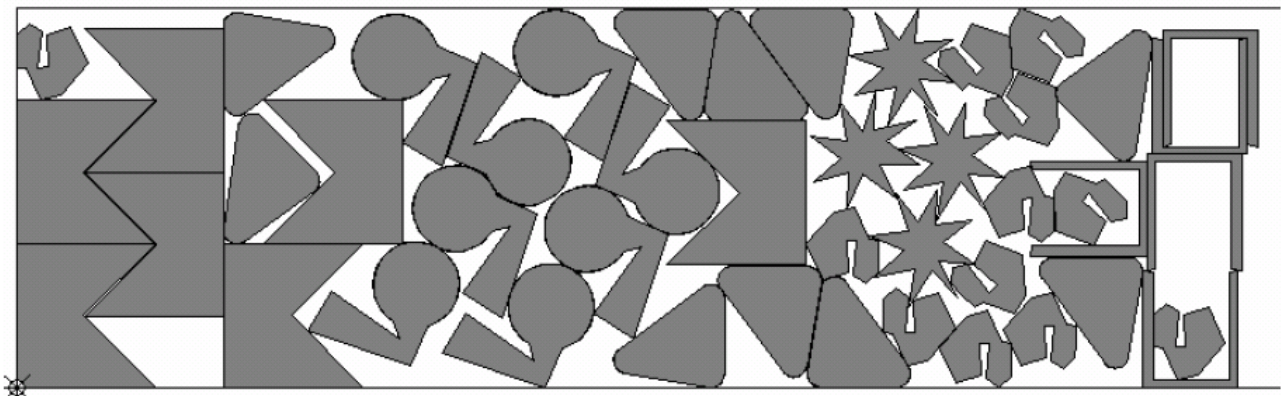
NP – Complete tai pačios sudėtingiausios problemos *NP* klasėje, kurios greičiausiai nepriklauso klasei *P*.

Dar niekam nepavyko sukurti algoritmo, kuris per polinominį laiką visada tiksliai (visais atvejais) išspręstų *NP – Complete* problemą. Verta paminėti, kad egzistuoja polinominio laiko algoritmai, sprendžiantys tokio tipo problemas, esant prielaidai, kad $P = NP$.

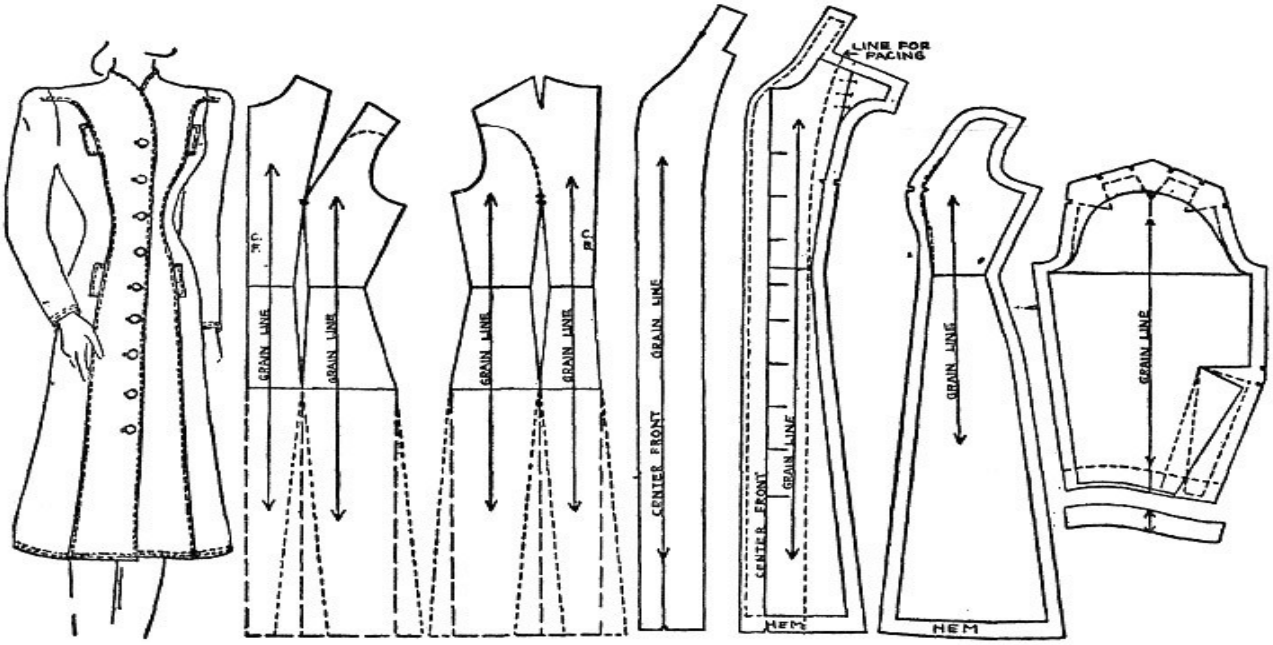
Pjaustymo uždavinio kilmė

Pjaustymas – tai fizinio objekto arba jo dalies padalijimas į dvi ar keletą naujų dalių, panaudojant pjaustymo įrankį.

Akivaizdu, kad pjautymo įrankiai gali būti labai įvairūs, taip pat ir pjaustymo objektai bei išpjovos gali būti visokiausių formų:



Pav. 2 Įvairių formų detalių pjaustymas



Pav. 3 Suknelės pjaustymo šablonų parengimo pavyzdys

Susiaurinsim pjausymo problemos klasę ir nagrinėsim atvejį, kai pjaustomi objektai ir iš jų gaunamos detalės yra stačiakampiai, medžiagos prigimties nepaisysim, nes tai esminio skirtumo nesuteikia mūsų nagrinėjamai problemai.

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

Klasifikacija

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

- Orientuotas į išpjovą būdas (pjaustymo metu kiekviena išpjova išpjaunama individualiai)

- Orientuotas į šabloną būdas (pirmiausiai iš išpjovų yra sudaromi šablonai, kuriems nustatomi intensyvumai, užsakymui patenkinti)

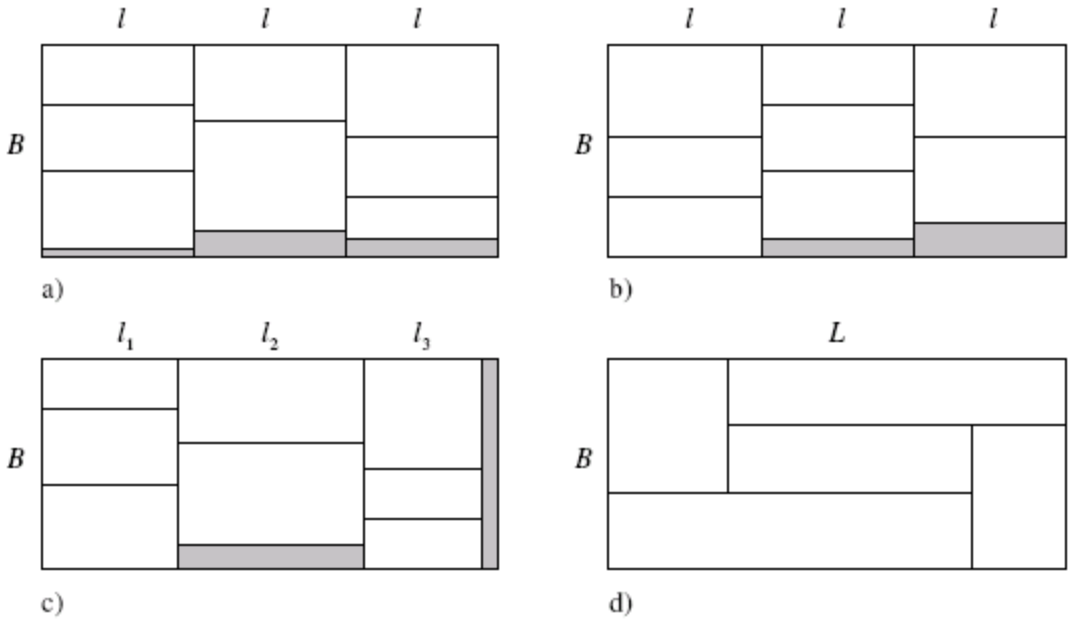
Be šių grupių apibrėžiamos keturios pagrindinės charakteristikos:

1. Dimensijų skaičius
    - 1 dimensija
    - 2 dimensijos
    - 3 dimensijos
    - N dimensijų (N > 3)
2. Žaliavų kiekis
    - Žaliavų deficitas
    - Žaliavų pakankamumas
3. Lakštų asortimentas
    - Vienas lakštas
    - Daug identiškų lakštų
    - Skirtingi lakštai
4. Išpjovų asortimentas
    - Nedaug išpjovų su skirtingomis dimensijomis
    - Daug išpjovų su skirtingomis dimensijomis
    - Daug išpjovų su panašiomis dimensijomis
    - Daug identiškų išpjovų

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

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

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



Pav. 4 Pjūvių pavyzdžiai

Antroji charakteristika yra *žaliavų kiekis.* Konkrečiai CSP klasifikacijai mažiausiai reikia dviejų kategorijų. Pirmoji, visos žaliavos bus sunaudotos, bet ne visi užsakymai bus įvykdyti. Antroji – visi užsakymai bus įvykdyti.

Trečioji charakteristika – lakštų asortimentas, gali būti suskirstyta į tris tipus. Pirmas, yra tiktai vienas didelis lakštas, antras – daug didelių lakštų su panašiomis dimensijomis, trečias – daug lakštų su skirtingomis dimensijomis.

Analogiškai suprantama ketvirtoji charakteristika - *išpjovų asortimentas.* Pirmu atveju - nedaug išpjovų su skirtingomis dimensijomis, antruoju – daug išpjovų su daug skirtingų dimensijų, trečiuoju – daug išpjovų su panašiomis dimensijomis, ketvirtuoju – daug identiškų išpjovų.

Verta paminėti, kad Dyckhoff pasiūlyta tipologija nėra bendra ir turi keletą trūkumų. Pirmiausia naudojant šią klasifikaciją ne kiekviena pjaustymo problema gali būti unikaliai apibrėžta. Tarkim ta pati strypų pjaustymo problema gali būti klasifikuojama dviem būdais:

i) 1 dimensija | Žaliavų pakankamumas| Daug identiškų strypų | *Nedaug išpjovų su skirtingomis dimensijomis*

ii) 1 dimensija | Žaliavų pakankamumas| Daug identiškų strypų | *Daug išpjovų su skirtingomis dimensijomis*

Be to tipologijos reprezentacija yra dalinai nesuderinta, nes pakankamai skirtingos problemos gali buti vienodai klasifikuojamos, pvz: metalo strypų ir lentų pjaustymo problema.

Net ir paprasčiausia CSP yra *NP – complete* sudėtingumo, tai reiškia, kad labai mažai tikėtina, kad egzistuoja polinominio laiko algoritmas, kuris galėtų pateikti sprendimą. Dėl problemos kombinatorinio sudėtingumo prigimties, jos sprendimas reikalauja *optimizavimu* pagrįstų metodų.

CSP sprendimo metodai

Pagrinde egzistuoja trys pagrindinės CSP sprendimo klasės. *Tikslieji metodai* 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 problemas, kurioms jie yra sukurti. *Metaeuristiniai metodai* turi savybę nesuklysti lokalaus optimalaus sprendinio prasme, kaip tai gali atsitikti su tradicinėmis euristikomis. Metaeuristiniuose metoduose sprendinio paieškos procesas dažnai reguliuojamas žemesnio lygio euristikos.

Dvimačio giljotininio pjaustymo užduoties sprendimo euristiniai metodai

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

(AN OVERVIEW OF SOME HEURISTIC ALGORITHMS FOR COMBINATORIAL OPTIMIZATION PROBLEMS Alfonsas Misevičius[1], Tomas Blažauskas[2], Jonas Blonskis[1], Jonas Smolinskas[1])

Užduoties formuluotė

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

$$\text{Maksimizuoti}\quad f(a_1,..., a_m) = \sum_{i=1}^{m} v_i a_i$$

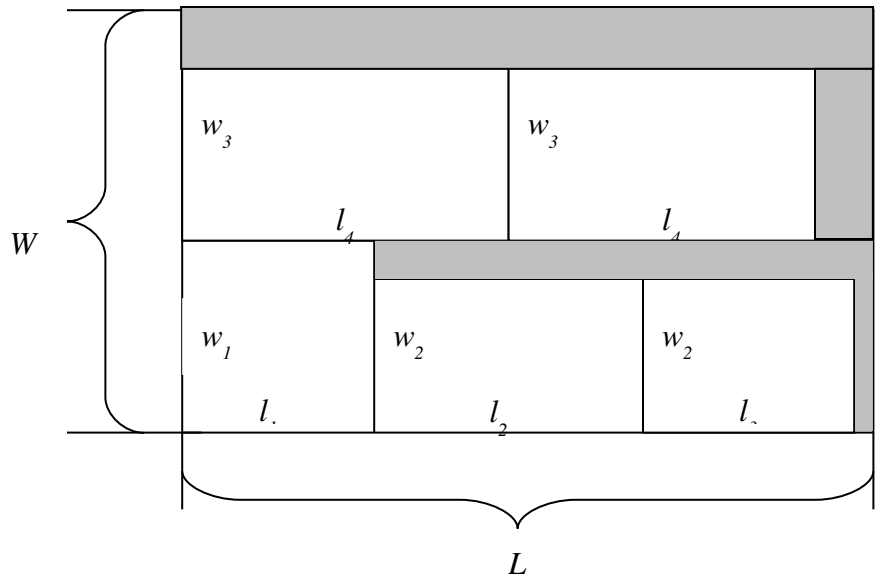
Atsižvelgiant į tai, kad $(a_1, a_2, \ldots, a_m)$ *atitinka 2-fazių giljotininio pjovimo planą* LxW *ir* $0 \le a_i \le b_i$, $i=1,\ldots, m.$

Reikia pastebėti, kad jei

$$v_i = \frac{l_i w_i}{LW},\ldots \quad i = 1,...,m$$

tai užduotis tampa ekvivalenti užduočiai minimizuoti nuostolius.

Tolesnį užduoties formulavimą atliksime ta pačia eiga kaip ir pjaustymo užduoties be apribojimų t.y. darysime prielaidą, kad primas pjūvis yra horizontalus ir sudarysime vienos dimensijos pjovimo planus juostoms $Lxw_j,\ j=1,\ldots,m.$ Tada nustatysime kiek kartų šie pjovimo planai yra atkartojami išilgai juostos pločio (Gilmore ir Gomory, 1965 ir Morabito ir Arenales, 1995).



Pav. 5 Dviejų dimensijų pjovimo planas

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

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

atsižvelgiant į:

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

$$\alpha_{kj}^i \ge 0 \quad ir \quad sveikasis \quad skaičkai$$

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

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

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

Pavadinkime *k-ąjį* juostos *Lxw$_j$* pjovimo planą – *(k,j)-planu.* Apsibrėžkime aibę $W_j = \left\{ i \,\middle|\, w_i \le w_j \right\}$. Tada *(k,j)-planas* apibrėžiamas taip:

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

$$\alpha_{kj}^i \ge 0 \quad ir \quad sveikasis \quad skaičkai, \quad k = 1,...,K_j, \quad j = 1,...,r$$

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

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

Taigi

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

Taip pat suminis panaudotų *i* dalių skaičius yra lygus

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

ir šis skaičius negali būti didesnis nei b_i

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

Taigi matematinis modelis aprašantis dvimatį giljotininį pjaustymą:

$$Maksimizuoti \quad f(\alpha, \beta) = \sum_{j=1}^{r} \sum_{i=1}^{m} \sum_{k=1}^{K_j} v_i \alpha_{kj}^i \beta_{kj} \qquad (4)$$

Atsižvelgiant į:

$$\sum_{i \in W_j} l_i \alpha_{kj}^i \le L \quad j = 1,...,r, \quad k = 1,...,K_j \qquad (5)$$

$$\sum_{j=1}^{r} w_j \sum_{k=1}^{K_j} \beta_{kj} \le W \qquad (6)$$

$$\sum_{j=1}^{r} \sum_{k=1}^{K_j} \alpha_{kj}^i \beta_{kj} \le b_i \quad i = 1,...,m \qquad (7)$$

$$\alpha_{kj}^i \ge 0, \quad \beta_{kj} \ge 0 \quad ir \quad sveikieji \quad skaičkai, \quad i = 1,...,m, \quad j = 1,...,r, \quad k = 1,...,K_j \qquad (8)$$

Užrašyta užduotis yra sveikaskaitė netiesinė optimizavimo užduotis. Jei pašalintume (7) apribojimą tada optimizavimo uždavinys taptų dviejų dimensijų dviejų fazių giljotininio pjaustymo uždavinių be apribojimų.

Gilmore ir Gomory (1965) pasiūlė dekompozicijos metodą spręsti dviejų fazių dviejų dimensijų giljotininio pjaustymo uždavinį. Šis metodas gali būti išvestas iš anksčiau pateikto modelio (vėliau mes šį metodą praplėsime iki giljotininio pjaustymo užduoties su apribojimais):

(k,j)-plano vertę pažymėkime $V_{kj} = \left(\sum_{i=1}^{m} \alpha_{kj}^i v_i \right)$ ir perrašykime sudarytą modelį tik be (7) apribojimo:

$$Maksimizuoti \qquad \sum_{j=1}^{r} \sum_{k=1}^{K_j} V_{kj} \beta_{kj} \qquad (9)$$

Atsižvelgiant į:

$$\sum_{i=1}^{m} \alpha_{kj}^i l_i \le L \qquad j = 1,...,r, \qquad k = 1,...,K_j \qquad (10)$$

$$V_{kj} = \sum_{i=1}^{m} \alpha_{kj}^i v_i \qquad j = 1,...,r, \qquad k = 1,...,K_j \qquad (11)$$

$$\sum_{j=1}^{r} w_j \sum_{k=1}^{K_j} \beta_{kj} \le W \qquad (12)$$

$$\alpha_{kj}^{i} \geq 0, \qquad \beta_{kj} \geq 0 \quad ir \quad sveikasis \quad skaičkai, \quad i = 1,...,m, \qquad j = 1,...,K_j \qquad (13)$$

Kadangi pjaustymo užduotis yra be apribojimų t.y. pašalintas apribojimas maksimaliam dalių skaičiui ((7) sąlyga), tai mums reikalingas tik vienas, geriausias, pjovimo planas w_j-juostai. Todėl galima praleisti indeksą *k* iš modelio formuluotės ir pirmoje metodo stadijoje imti tik geriausią *w_j-juostos* pjovimo planą:

$$V_j = Max \sum_{i \in W_j} v_i \alpha_j^i, \qquad j = 1,...,r$$

Atsižvelgiant į: $\sum_{i \in W_j} l_i \alpha_j^i \leq L$

$\alpha_j^i \geq 0$ *ir sveikasis skaičkai,* $i \in W_j$.

Antroje metodo stadijoje yra sprendžiamas kuprinės uždavinys tam, kad nustatyti kiek kartų *L*xw_j*-juosta* (kartu su geriausiu jos pjovimo planu) turi atsikartoti visame dvimačiame plane, t.y.

$$Maksimizuoti \quad \sum_{j=1}^{r} V_j \beta_j$$

$$Atsižvelgiant \text{ į:} \quad \sum_{j=1}^{r} w_j \beta_j \leq W$$

$$\beta_j \geq 0 \quad ir \quad sveikasis \quad skaičkai, \quad j = 1,2,...,r.$$

Euristinis dekompozicijos metodas

Visų prima rasime geriausią venos dimensijos pjovimo planą visoms w_j-*juostoms*, tačiau dar atsižvelgsime į maksimalų leistiną dalių *i* skaičių b_i.

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

$$V_j = Max \sum v_i \alpha_j^i$$

$$\textit{Atsižvelgiant į:} \sum_{i \in W_j} l_i \alpha_j^i \le L$$

$$0 \le \alpha_j^i \le b_i \quad \textit{ir} \quad \textit{sveikasis} \quad \textit{skaičkai}, \quad i \in W_j = \{ i \mid \ w_i \le w_j \}.$$

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

Antras žingsnis: Išspręsti sveikaskaičio tiesinio optimizavimo užduotį:

$$\textit{Maksimizuoti} \quad \sum V_j \beta_j \quad (23)$$

Atsižvelgiant į:

$$\sum_{j=1}^{r} w_j \beta_j \le W \quad (24)$$

$$\sum_{j=1}^{r} \alpha_j^i \beta_j \le b_i, \quad i = 1,...,m \quad (25)$$

$$\beta_j \ge 0 \quad \textit{ir} \quad \textit{sveikasis} \quad \textit{skaičkai}, \quad j = 1,2,...,r. \quad (26)$$

Reikia pastebėti, kad antrojo žingsnio užduotis jau nebėra kuprinės užduotis dėl maksimalaus dalių i skaičiaus apribojimo.Taip pat ši procedūra yra euristinė, nes imamas tik vienas (geriausias) pjovimas kiekvienai juostai, o optimalus sprendimas gali būti sudarytas iš skirtingų pjovimų tai pačiai juostai. Be to antrojo žingsnio modelis nėra lengvai sprendžiamas ir reikalauja labai sudėtingo kodo palyginus su kuprinės užduotimi, todėl vietoj antrojo žingsnio spręsime kitą užduotį gautą pasinaudojus relaksacija (t.y. pakeisime apribojimų (24) ir (25) sąlygą), tam kad identifikuotume labiausiai vertingą juostą iš visų juostų sugeneruotų pirmame žingsnyje.

Antras žingsnis': sumuodami (24) padaugintą iš γ ir (25) padaugintą iš (1-γ), ir relaksavę sveikaskaitį apribojimą, turime tolydžiąją kuprinės užduotį:

$$\textit{Maksimizuoti} \quad \sum_{j=1}^{r} V_j \beta_j$$

$$\textit{Atsižvelgiant į:} \quad \sum_{j=1}^{r} \left(\gamma w_i + (1-\gamma) \sum_{i=1}^{m} \alpha_j^i \right) \beta_i \le \gamma W + (1-\gamma) \sum_{i=1}^{m} b_i$$

$$\beta_j \ge 0, \qquad j = 1,...,r.$$

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

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

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

- Tiktai vieną kartą (t.y. $\beta_k=1,\ \beta_j=0,\ j\neq k$)
- Keletą kartų iš eilės (t.y. $\beta_k = \min imum\left\{\left[\frac{W}{w_k}\right], \left[\frac{b_i}{\alpha_k^i}\right], i=1,...,m\right\}, \beta_j = 0,\ j \neq k$)

  Tada atnaujinami $W \leftarrow W - w_k\beta_k$ ir $b_i \leftarrow b_i - \alpha_k^i\beta_k$ ir kartojamas pirmasis žingsnį.

Sutrumpintas AND/OR-medžio metodas

Tarkime, kad turime lakštą *LxW*, kurį reikia supjaustyti ir pirmame pjovimo etape yra atliktas horizontalus giljotininis pjūvis taške *p* (0<*p*<*L*), gauname du stačiakampius: *pxW* ir *(L-p)xW*. Jei prieš tai buvo atliktas vertikalus pjūvis tai horizontalus pjūvis turi būti atliekamas tik viename iš dviejų vertikalaus pjovimo metu susidariusių stačiakampių. Tokia pjovimų seka gali būti formalizuota kaip AND/OR – grafas (detaliau aptarsime vėliau). Tačiau čia vietoj to, kad pilnai ieškotume AND/OR – grafe, mes sustosime iš kart po pirmo pjūvio, nesvarbu ar tai būtų vertikalus, ar horizontalus pjūvis, ir pritaikysime susidariusiems stačiakampiams dekompozicijos euristiką. Panašios idėjos buvo panaudotos Fayard (1998). Euristika aprašoma taip:

1. Visiems *p* tokiems, kad

   $$p = \sum_{i=1}^{m} \alpha_i w_i < W, \quad kur \quad \alpha_i \geq 0 \quad ir \quad sveikasis$$

1.1 Pritaikyti dekompozicijos euristiką stačiakampiui *pxW*, gaunant α_i^1 skaičių *i-tipo* dalių.

1.2 Pritaikyti dekompozicijos euristiką stačiakampiui *(L-p)xW*, bet su viršutine riba b_i pataisyta į $b_i - a_i^1$.

2. Visiems *q* tokiems, kad

$$q = \sum_{i=1}^{m} \beta_i w_i < W, \quad kur \quad \beta_i \geq 0 \quad ir \quad sveikasis$$

2.1 Pritaikyti dekompozicijos euristiką stačiakampiui *Lxq*, gaunant α_i^1 skaičių *i-tipo* dalių.

2.2 Pritaikyti dekompozicijos euristiką stačiakampiui *Lx(W-q)*, bet su viršutine riba b_i pataisyta į $b_i - a_i^1$.

3. Paimti geriausia gautą sprendinį iš 1 ir 2 žingsnių.

Pastaba: diskretizavimo aibė: $\left\{ p = \sum_{i=1}^{m} \alpha_i w_i < W, \quad kur \quad \alpha_i \geq 0 \quad ir \quad sveikasis \right\}$, gali būti iš anksto nusakyta Christofides ir Whitlock (1977) formule ir visi pjūviai atlikti su šia aibe neatmeta nei vieno optimalaus sprendinio.

Lagranžo relaksacija ir subgradiento optimizavimas

Dualiai Lagranžo problemai, gautai dualizuojant netiesiniu apribojimus (7), pritaikysime subgradiento algoritmą. Nėra jokios garantijos, kad šiuo metodu bus gautas optimalus sprendimas, nes gali susidaryti dualumo tarpas. Bet šiuo metodu yra nustatomos viršutinės ir apatinės tikslo funkcijos ribos, kas leidžia užtikrinti ε-optimalumą gautam sprendiniui.

Lagranžo problemą apsibrėžiame relaksuodami (7) apribojimą ir pridėdami jį prie tikslo funkcijos naudodami dualius daugiklius λ_i, i=1,…,m. Kadangi (7) apribojimas riboja dalių skaičių pjaustymo plane, Lagranžo uždavinys yra 2-matis 2D 2-fazių giljotininio pjaustymo uždavinys be apribojimų, kuris gali būti sprendžiamas Gilmoro

ir Gomory dekompozicijos metodu. Lagranžo problema yra aprašoma (h – duali Lagranžo funkcija):

$$h(\lambda) = Max \quad \sum_{i=1}^{m}\sum_{j=1}^{r}\sum_{k=1}^{K_j}(v_i - \lambda_i)\alpha_{kj}^{i}\beta_{kj} + \sum_{i=1}^{m}\lambda_i b_i \qquad (31)$$

Atsižvelgiant į:

$$\sum_{i=1}^{m}\alpha_{kj}^{i} l_i \le L \qquad j = 1,...,r, \qquad k = 1,..,K_j \qquad (32)$$

$$\sum_{j=1}^{r} w_j \sum_{k=1}^{K_j}\beta_{kj} \le W \qquad (33)$$

$$\alpha_{kj}^{i} \ge 0, \qquad \beta_{kj} \ge 0 \qquad ir \quad sveikieji, \quad i = 1,...,m, \qquad j = 1,...,r, \qquad k = 1,...,K_j. \quad (34)$$

Užrašysime užduoties kintamuosius matricinėje formoje: $\overline{\alpha} = \left(\alpha_{kj}^{i}\right)_{j=1,...,r,\quad k=1,...,K_j}^{i=1,...,m}$

$\overline{\beta} = (\beta_{kj})_{j=1,...,r \quad k=1,...,K_j}$. Kiekvienam tinkančiam sprendiniui $(\overline{\alpha}, \overline{\beta})$ (t.y. tokiam sprendiniui, kuris tenkina (5) – (8) sąlygas) ir $\lambda \ge 0$, teisinga nelygybė:

$$h(\lambda) \ge f(\overline{\alpha}, \overline{\beta})$$

Tai reiškia, kad $h(\lambda)$ yra viršutinė funkcijos $f(\overline{\alpha}, \overline{\beta})$ riba. Tada dualus Lagranžo uždavinys suvedamas į tai, kad reikia nustatyti minimalią viršutinę ribą:

$$Minimizuoti_{\ \lambda \ge 0}\ h(\lambda)$$

Šis uždavinys yra sprendžiamas subgradiento algoritmu.

Pastabos:

1. Pakankamos sąlygos optimalumui:
    a) Lagranžo užduoties sprendinys tenkina relaksuotus apribojimus ir
    b) $h(\lambda) = f(\overline{\alpha}, \overline{\beta})$

    Tačiau šios dvi sąlygos nebus visada tenkinamos nes neiškilioms problemoms gali susidaryti dualus tarpas.
2. Tinkamas sprendinys $(\overline{\alpha}, \overline{\beta})$, kuris tenkina sąlygą: $h(\lambda) \le (1+\varepsilon) f(\overline{\alpha}, \overline{\beta})$ yra vadinamas ε-optimaliu
3. Plačiausiai naudojamos subgradiento algoritmo stabdymo sąlygos:
    a) maksimalus iteracijų skaičius
    b) labai mažas žingsnio dydis sukelia nedidelius sprendinio modifikavimus
    c) apatinė riba yra pakankamai artima viršutinei ribai t.y. $h(\lambda) - f(\overline{\alpha}, \overline{\beta})$ yra pakankamai mažas.

Subgradiento metodas

Geriausias tinkantis sprendinys bus žymimas: $(\alpha^{best}, \beta^{best})$. Apatinė tikslo funkcijos (4) riba $Z_{LB}^{best} = f(\alpha^{best}, \beta^{best})$. Pradinis tinkantis sprendinys yra gaunamas naudojant dekompozicijos euristiką. Geriausia sugeneruota viršutinė riba - Z_{UB}^{best}. Pradinė reikšmė $Z_{UB}^{best} = \sum_{i=1}^{m} v_i b_i$.

Laisvai pasirenkamas pradinis Lagrandžo daugiklis $\lambda^0 \ge 0$ ir koks nors nedidelis $\varepsilon > 0$. Maksimalus iteracijų skaičius – *it_max*.

t := 0

STOP := FALSE

While STOP=FALSE and t <= it_max do

{1. viršutinė riba}

Sprendžiamas Lagranžo uždavinys (31) – (34) pradiniam sprendiniui (α^t, β^t) ir viršutinei ribai $Z_{UB} = h(\lambda^t)$ nustatyti. Jei $Z_{UB} < Z_{UB}^{best}$ tada atnaujinti $Z_{UB} := Z_{UB}^{best}$

{2. subgradientas}

Apskaičiuoti t iteracijos subgradientą a^t, kuris yra m-matis vektorius, kur kiekviena *i*'oji koordinatė apskaičiuojama:

$$a_i^t := b_i - \sum_{j=1}^{m} \sum_{k=1}^{K_j} \alpha_{kj}^i \beta_{kj}, \qquad i = 1,...,m.$$

{3. Optimalumo patikrinimas}

if $a^t \le 0$ then

STOP = TRUE

Sprendinys yra optimalus (α^t, β^t).

else

if $a^t > 0$ then

sprendinys yra tinkamas .

$Z_{LB} := f(\alpha^t, \beta^t)$

else

($a^t < 0$) tada sprendinys yra netinkamas. Jei per paskutines iteracijas sprendinys nebuvo pagerintas, tada naudoti *Lagranžo euristiką* (aprašyta žemiau) ir apskaičiuoti apatinę ribą Z_{LB}.

end if

end if

if $Z_{LB} > Z_{LB}^{best}$ then

$Z_{LB}^{best} := Z_{LB}$ ir išsaugoti sprendinį

end if

if $Z_{UB}^{best} - Z_{LB}^{best} < \varepsilon \cdot Z_{LB}^{best}$ then

STOP := TRUE

Sprendinys yra ε-optimalus

else

$\lambda_i^{t+1} := \max\{0, \lambda_i^t + \theta_t a_i^t\}, \quad i = 1,...,m, \quad kur \quad \theta_t \quad yra \quad žingsnio \quad dydis$

t := t + 1

end if

end while

Lagranžo euristika

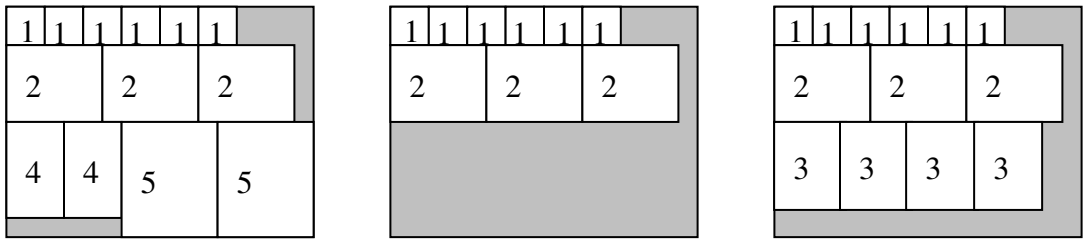
Po Lagranžo uždavinio sprendimo gautas sprendinys gali būti netinkamas pradinei sąlygai, todėl reikia atlikti tam tikras sprendinio pertvarkymo procedūras, kurios sprendinį pertvarkytų į tinkamą. Tokios procedūros labai priklauso nuo uždavinio pobūdžio ir jų gali būti labai įvairių. Čia bus naudojama tokia procedūra:

1. Reikia nustatyti kuri dalis viršija viršutinę ribą. Išmesti iš sprendinio juostą, kuri turi daugiausiai tokių dalių. Kartoti šį žingsnį kol nebeliks dalių viršijančių šią ribą. a)
2. Perkelti likusias juostas b)
3. Pritaikyti dekompozicijos euristiką likusiam stačiakampiui. c)

4.

Lentelė 1 Duomenų pavyzdys

dalis	1	2	3	4	5
l_i x w	15x15	33x36	22x38	22x43	27x49
b_i	8	3	4	7	1
v_i	0,0225	0,1188	0,0946	0,1323	0,1323



a) b) c)

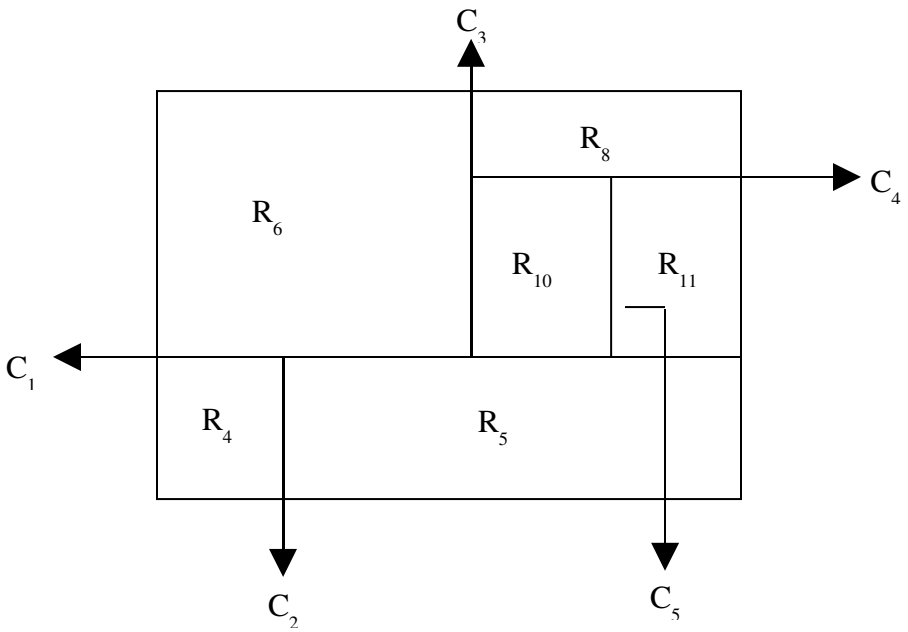
Pav. 6 Lagranžo euristikos pavyzdys

AND/OR-grafo metodas

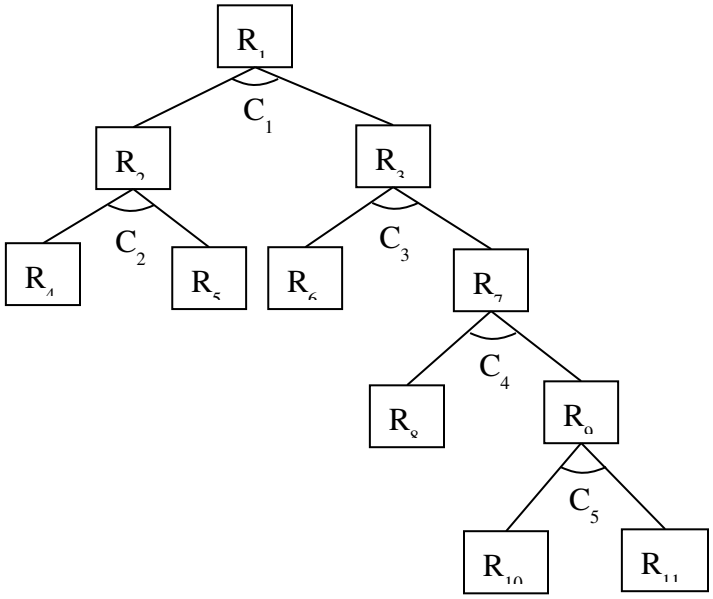
Aibę pjaustymo uždavinių galima išspręsti ieškant specialius grafus vadinamus AND/OR-grafais. Grafo mazgas vaizduoja stačiakampį – pradinį lakštą, tarpiniame pjovime gautą stačiakampį ir t.t., o AND-lankas, jungiantis N ir du kitus mazgus N_1 ir N_2 – giljotininį pjūvį einantį per stačiakampį N, kurio pjovimo pasekoje gauname N_1 ir N_2 stačiakampius. OR-lankai atvaizduoja alternatyvų stačiakampio pjovimo būdą. Taškai per kuriuos vyksta pjovimas pasirenkami iš *diskretizavimo aibės* (Morabito ir Arenales (1996)), jei mazgas, kuriam optimalus sprendinys jau yra žinomas, vadinamas išspręstu. Tokiu būdu kiekvienas pjaustymo planas yra išreiškiamas pilnu keliu AND/OR-grafe ir šio kelio paieškai gali būti pasirinktos skirtingos strategijos.

Žemiau pateiktas pjaustymo planas ir jį atitinkantis grafo kelias (pilnas grafas neatvaizduotas).

Tokia duomenų struktūra yra labai efektyvi, nesvarbu ar tai būtų vienos dimensijos, dviejų dimensijų ar trijų dimensijų, giljotininis ar negiljotininis pjaustymo uždavinys.



Pav. 7 AND/OR-grafo pjaustymo planas



Pav. 8 AND/OR-grafo kelias

k-fazės

Ieškant AND/OR-grafe fazių skaičių ir dalių skaičių. Tarkime, kad N yra AND/OR-grafo mazgas, funkcijos CUT ir STAGE mazgui N apibrėžtos taip:

$$CUT(N) = \begin{cases} 0 & jei \quad N \quad gautas \quad atliek\tan t \quad vertikalų \quad pjūjū \\ 1 & kitu \quad atveju \end{cases}$$

STAGE(N) = atliktų fazių skaičius, kad gauti N mazgą

Tarkime N_1 ir N_2 mazgai gauti pjaunant N. Akivaizdu, kad $CUT(N_1) = CUT(N_2)$, o *STAGE* funkciją apibrėšime taip:

for s = 1, 2 do

if $CUT(N_s) = CUT(N)$

then $STAGE(N_S) := STAGE(N)$

else $STAGE(N_S) := STAGE(N) + 1$

end if

end for

Pjaustymo pradžioje lakštas *LxW* atvaizduojamas mazge $N = 0$, o $STAGE(0) = 0$ ir $CUT(0) = -1$. Tarkime, kad *k* yra maksimalus fazių skaičius, tada jei $CUT(N) = 0$, o $STAGE(N) = k$, tada nuo šio momento tik vertikalūs pjūviai bus priimami pjaustant N mazgą, tol kol bus rasti paskutiniai mazgai.

Apribojimai

Tarkime kad mazgo N vaikai N_1 ir N_2. Kadangi modeliuojame giljotininį pjūvį su apribojimais, tai dalių, tipo *i*, skaičius mazge N_1 turi įtakos tokio pačio tipo dalių skaičiui mazge N_2. Tokiu atveju euristnė procedūra apibrėžiama taip: tarkime $\bar{b}_i(N)$

bus maksimalus skaičius tipo *i* dalių, kurios gali būti panaudotos mazge N. Tarkime pradinės reikšmės N = 0, o $\bar{b}_i(0) = b_i$. Iš pradžių ieškoma N_1 mazge ir $\bar{b}_i(N_1) = \bar{b}_i(N), \quad i = 1,...,m$. Tik po to kai mazgas N_1 išspręstas (visi mazgo N_1 vaikai surasti) sprendžiamas N_2 mazgas: $\bar{b}_i(N_2) = \bar{b}_i(N) - a_i(N_1), \quad i = 1,...,m$, čia $a_i(N_1)$ - tipo *i* dalių skaičius mazge N_1. Tada yra kartojama ta pati procedūra – inicijuojamas N_2 ir einama prie N_1.

Tam, kad suformuluoti šakų ir ribų metodą galima apsibrėžti viršutinę ir apatinę ribas, tokiu būdu galima išvengti neperspektyvių šakų. Paprastai apatinė riba remiasi trivialiu sprendiniu mazge N, o apatinė riba remiasi relaksacija. Tarkime U(N) – viršutinė riba, o L(N) – apatinė riba, V(N) – žinomas geriausias sprendinys šakai N, tada šakos N sprendimas bus nutrauktas jei:

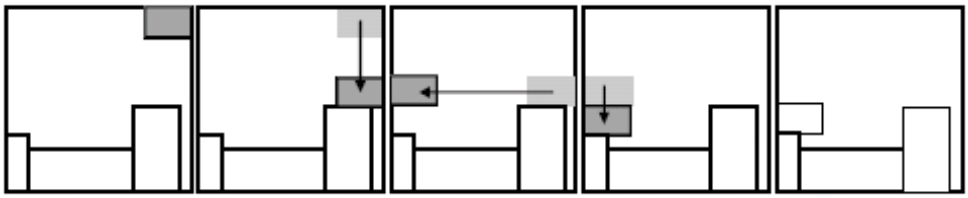
a) $(1+\lambda_1)V(N) \geq U(N_1) + U(N_2)$ (buvo panaudotas $\lambda_1 = 0.001$); arba

b) $\lambda_2 L(N) \geq L(N_1) + L(N_2)$ (buvo panaudota $\lambda_2 = 0.95$)

Euristiniai „Žemiausio kairėn“ ir „Žemiausio kairėn užpildymo“ algoritmai

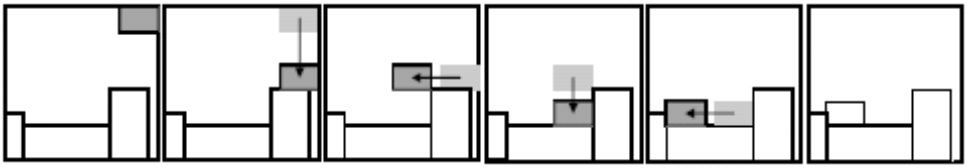
Euristiniai metodai pateikia pakankamai gerus per trumpą laiko tarpą sprendimus. Labiausiai dokumentuoti yra *„Žemiausio kairėn“ (Bottom Left, BL)* ir *„Žemiausio kairėn užpildymo“ (Bottom Left Fill, BLF)* algoritmai. 1996 metais Jakobs panaudojo *BL* metodą ortogonaliajai dvimatei negiljotininei pjaustymo problemai spręsti.

Algoritmui paduodamas stačiakampių, kuriuos reikia išpjaustyti, sąrašas atinkamai detalės yra dėliojamos ant lakšto (pjaustomo objekto) tokiu būdu: pirmiausia stačiakampis yra orientuojamas į viršutinį dešinį lakšto kampą, tada atliekami perkelimo veiksmai iš pradžių į apačią, paskui į kairę.



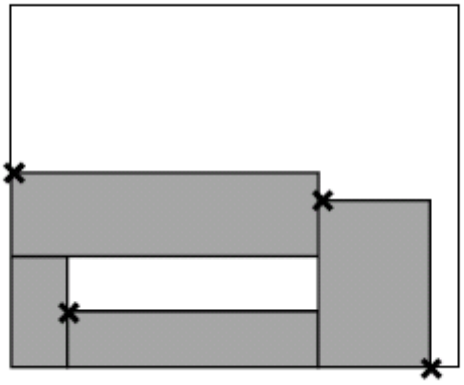
BL veikimo principas

Lui ir Teng 1999 metais sukūrė pagerintą *BL* euristiką, suteikiančią judesiui žemyn prioritetą tokį, kad figūros perstumiamos į kairę tik tada, kai negalimas slydimas žemyn. Buvo parodyta (ko negalima pasakyti apie Jakobs metodą), kad visalaik egzistuoja mažiausiai viena stačiakampių seka, galinti duoti optimalų sprendimą.



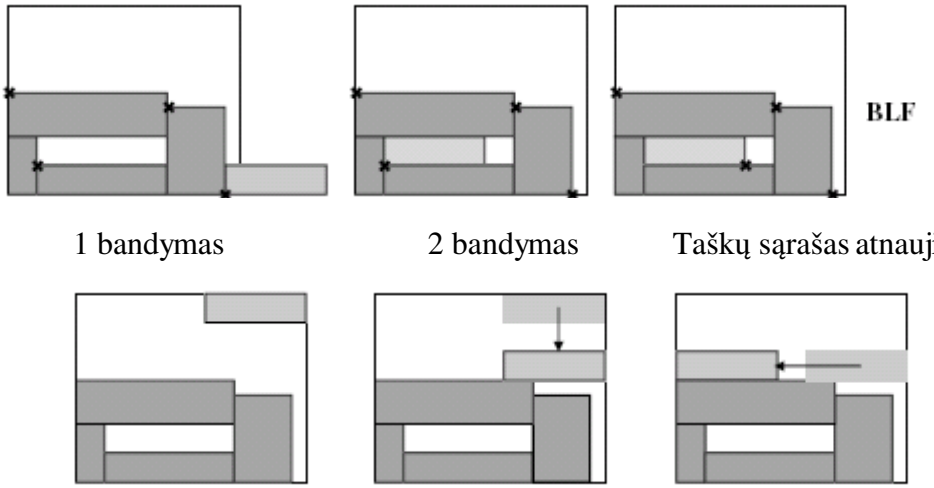
BLF veikimo principas

Antrasis *BLF* metodas yra modifikuota *BL* dėliojimo euristikos versija. Papildomai sudaromas sąrašas taškų, pradedant nuo apatinio kairiojo kampo, kurie parodo, kur būtų galima įdėti figūrą. Pjaustymo metu algoritmas pradedamas nuo žemiausio ir kairiausio taško, kuriame yra talpinama figūra. Tikrininama, ar talpinamas stačiakampis nesikerta su kokia nors kita anksčiau įdėta figūra arba lakšto kraštu. Jeigu nesikerta, figūra paliekama ir taškų sąrašas yra atnaujinamas, kitu atveju nagrinėjamas sekantis taškų sąrašo elementas (taškas), procesas tęsiamas tol, kol įmanoma įdėti figūrą taip, kad ši nesikirstų su kitomis.



Galimos figūrų įdėjimo vietos

Paveikslėlyje parodyti taškai, kuriose įmanoma talpinti sekančią figūrą (taškų sąrašas). Kaip minejome, paieškos procesas pradedamas nuo žemiausiai kairiausio taško (situacija, kai taškai yra viename lygyje, imamas kairiausias taškas). Taigi akivaizdu, kad *BLF* metodas gali užpildyti susidariusias vakancijas.



BL ir BLF veikimo principų palyginimas

Susidariusi vakancija niekada nebus užpildyta problemą sprendžiant *BL* metodu, tai yra *BLF* pranašumas. Verta paminėti, kad sprendinio kokybė priklauso nuo stačiakampių pateikimo eiliškumo. Ekspermentų metu nustatyta, kad *BLF* metodo sprendinio tikslumas buvo 25% didesnis nei *BL*, be to pradinis figūrų surikiavimas pagal mažejančius ilgius arba pločius pagerino abiejų algoritmų sprendinio tikslumą 10%.

Nustatyta, kad *BL* algoritmo sudėtingumas yra $O(N^2)$, kai *N* pjaustomų detalių skaičius, atitinkamai *BLF* - $O(N^3)$. *BLF* sudėtingumas didesnis, todėl laiko sąnaudų prasme jisai dirba ilgiau, tačiau gautas sprendinys yra tikslesnis.

Tabu paieška

Tabu paieškos metaeuristika (TS) paskelbė 1987m. Hansenas ir Jaumardas, taip pat 1989m. Gloveris. Nuo to laiko TS įsitivrtino kaip viena iš pačių galingiausių technikų sudėtingiems optimizavimo uždaviniams spręsti. Trumpai kalbant, TS pagrįsta kaimynų paieška stengiantis išvengti lokalaus minimumo. TS idėja – leisti kopti į kalną

tada kai sprendinys nėra pagerinamas. Tačiau kai kurie ėjimai turi būti uždrausti tam, kad išvengi ciklų. Taigi tabu paieška prasideda nuo pradinio sprendinio s, kuris gali būti atsitiktinai sugeneruojamas, ir judama nuo vieno sprendinio prie kaimyninio sprendinio. Kiekviename žingsnyje patikrinama aibė kaimyninių sprendinių $\phi(s)$ ir pasirenkamas žingsnis, kuris labiausiai pageriną esamą sprendinį. Jei nėra žingsniu, kurie pagerintų sprendinį, pasirenkamas tas, kuris jį pablogina mažiausiai.

Tam, kad uždrausti sugrįžimą į jau surastą lokalų minimumą jau atlikti žingsniai yra uždraudžiami. Tai atliekama saugant žingsnius tabu sąraše T . Tabu sąrašas saugo paskutiniuosius h žingsnių ($h = |T|$) šis parametras vadinamas tabu sąrašo dydžiu. Taigi žingsnis nuo s į s′ laikomas tabu jei jis saugomas tabu sąraše T. Tačiau po tam tikro laiko gali būti naudinga sugrįžti prie uždrausto žingsnio, tam tikslui sukuriamas aspiracijos kriterijus. Paprastai tabu būsena ignoruojama jei $f(s') < f(s^*)$, kur s^* kolkas pats geriausias rastas sprendinys. Supraprastinta tabu paieškos schema pateikiama žemiau.

```
procedure tabu_search
// input: s° − initial solution; output: s* − the best solution found //
s ← s°; s* ← s°;
initialize the tabu list T;
repeat // main cycle //
given neighbourhood function N, tabu list T, and aspiration criterion,
find the best possible solution s′∈N(s);
s ← s′; // replace the current solution by the new one //
insert the solution s (or its attribute, e.g. the move from s to s′)
into the tabu list T;
if f(s)<f(s*) then s* ← s; // save the best so far solution //
update the tabu list T
until termination criterion is satisfied
end // procedure //
```

Tabu paieškos algorimtai pagrinde skiriasi dėl pagrindiniu „ingredientų“: tabu sąrašo ir aspiracijos kriterijaus. Kitos taip pat svarbio tabu paieškos savybės: ilgalaikė atmintis, tikslo analizė, intensifikacijos ir diversifikacijos mechanizmai.

(AN OVERVIEW OF SOME HEURISTIC ALGORITHMS FOR COMBINATORIAL OPTIMIZATION PROBLEMS Alfonsas Misevičius[1], Tomas Blažauskas[2], Jonas Blonskis[1], Jonas Smolinskas[1])

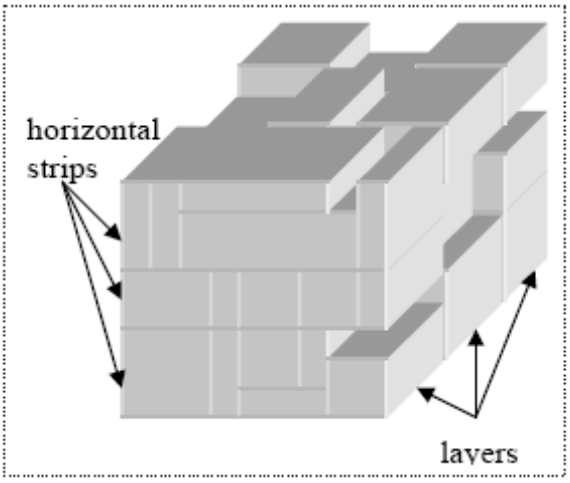
Trimačio pjaustymo užduoties sprendimo eurisitikos

The problem considered in this paper is the Knapsack Container Loading Problem (KCLP) where we assume that the profit of a box equals its volume. The objective is to fill the container most possible. The following notations will be used through the whole paper: the container has width W, height H and depth D. A set N = {1…n} of boxes is given, each box j having width w_j , height h_j and depth d_j . The volume of each box is $v_j=w_jh_jd_j$.

Heuristic algorithms

Konteinerio užpildymo uždavinys yra NP-hard tipo užduotis – tik labai mažos apimtis uždaviniai gali būti išsprendžiami optimaliai, o didelės apmites uždaviniams reikia taikyti euristinius metodus. Populiariausi euristiniai metodai gali būti suskirstyti į sienos-statymo algoritmus (Bischoff and Marriott, Hemminki, Gehring, Menscher, Meyer), krūvos-statymos algorimtus (Gilmore and Gomory), giljotininio pjaustymo algorimtus (Morabito and Arenales), ir kūbų-rikiavimo algoritmus (Bortfeldt and Gehring).

George and Robinson pirmieji pristatė sienos-statymos algorimtą. Algorimtas užpildo konteinerį aibe sluoksnių, kurie išsidėstę išilgai konteinerio gyliui.



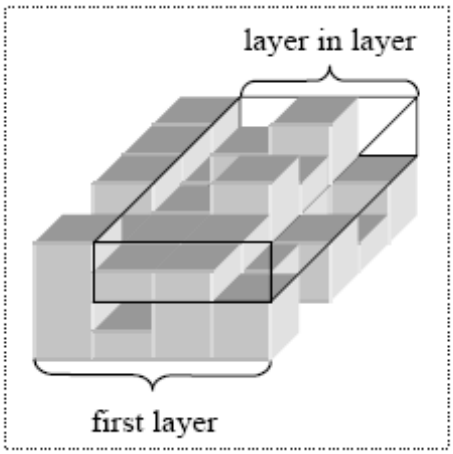
Sienos-statymo algoritmas

Ši euristika remiasi normalizuotu sluoksnio storiu: priimami tik tokie sluoksniai kurių storis sutmapa su kurios nors dėžės storiu. Tam, kad pasiekti gerus rezultatus sluoksnio storis turi būti atidžiai parenkamas. Tada siena yra užpildoma horizontaliomis juostomis. Kiekviena juosta užpildoma dėže su didžiausiu rangu. Rangai nustatomi pagal mažiausią dėžės dimensiją, tada pagal tokio paties tipo dėžių skaičių ir galiausiai pagal didžiausia dėžės dimensiją.

Žinoma, mes galime rasti daugybę euristinių algorimtų, kurie remiasi sienos-statymo principu ir kiekvienas jų gali naudoti daug skirtingų rango suteikimo funkcijų. Pagrindinis tokių algoritmų trūkumas yra tas, kad jie gali užstrigti lokaliame minimume. Tam, kad pagerinti sprendinį reikalinga kokia nors metodika leidžianti išeiti iš tokio minimumo.

Sienos-statymo euristikos praplėtimas - Medžio-paieškos euristika. Čia medžio paieškos strategija yra naudojama rasti sluoksnio pločių ir gylių aibę. Tam, kad sumažinti kompleksiškumą taikomas m-pjūvio metodas, kur naudojamas fiksuotas analizuojamų viršūnių skaičius. Kiekvienai viršūnei parenkama M_1 skirtingų sluoksnio storių. Kiekvienas sluoksnis užpildomas juostomis horizontaliai arba vertikaliai. Vėlgi yra naudojamas tik ribotas skaičius M_2 skirtingų juostų pločių. Tada kiekviena juosta yra yžpildoma sprendžiant kuprinės uždavinį.

Baltacioglu pasiūlė statyti sienas išilgai bet kurios iš šešių konteinerio sienų. Šis metodas apima ir sluoksnio pakavimo ir sienos-statymo algorimtus. Vienas iš svarbiausių šio algorimto bruožų - sluoksnio sluoksnyje principas t.y. kai sub-sluoksnis patalpinamas į paskutinį sudarytą sluoksnį.



Sluoksnio sluoksnyje pakavimas

Kitas labai svarbus šios euristikos bruožas – žmogaus intuicijos panaudojimas siekiant nuspręsti kurią dėžę pasirinkti. Žmonės renkasi tokias dėžes, kurios sumažina pakuojamo paviršiausius netaisyklingumus – vizualiai išmatuodami susidariusius tarpus ir pasirinkdami tokias dėžes, kurios išlaikytų kiek galima tolygesnę topologiją. Ši euristika veikia tokiu pačiu principu. Užpildant sluoksnius siekiama išlaikyti plokščią pakavimo paviršių. Kiekviename žingsnyje yra nustatomos susidariųsių tarpų dimensijos ir analizuojamos tinkamos dėžės bei jų orientacijos. Algoritmas atlieka sluoksnio pakavimą sumažindamas užduoties apimtį iki dvimačios pakavimo uždavinio.

Randomizuota Eurisitka

Siekiant nustatyti kuri dėžė bus naudojama sekanti naudojamos tokios funkcijos:

1. $f_1 \rightarrow \max\{v_i\}$

Dėžė, kurios plotas didžiausias yra pati vertingiausia.

2. $f_2 \rightarrow \max\{v_i / V_i\}$, kur V_i gaubiančios sferos plotas.

Tokiu būdu suprantama, kad geriausia dėžė yra ta kuri yra arčiausia kūbo formai.

3. $f_3 \to \max\{\min\{v_i, h_i, d_i\}\}$

Dėžė, kurios mažiausioji dimensija yra didžiausia yra vertingiausia.

Konteinerio dalinimas į keletą sluoksnių duoda geresnį sprendinį. Konteineris dalinamas į sluoksnius išilgai didžiausios dimensijos. Pakavimui yra pasirenkamas sluoksnio storis kurio rangas didžiausias. Čia naudojamos sekančios rangų funkcijos:

4. $f_k^1 \to \sum_{i=1}^{n} 1(w_i = k \vee h_i = k \vee d_i = k), \quad kiekvienam \quad k = \alpha,...,\beta$

5. $f_k^2 \to \sum_{i=1}^{n} 1(\max\{w_i, h_i, d_i\} = k), \quad kiekvienam \quad k = \alpha,...,\beta$

6. $f_k^3 \to \sum_{i=1}^{n} 1(\min\{w_i, h_i, d_i\} = k), \quad kiekvienam \quad k = \alpha,...,\beta$

7. $f^4 \to \max\{\min\{v_i, h_i, d_i\}\}, \quad i = 1...n$

Eksperimentai parodė, kad įmanoma pasiekti geresnių rezultatų naudojant šių euristikų mišinį. Kiekvienam sluoksniui priskiriami keturi storiai gauti naudojant šias euristikas ir pasirenkamas geriausias gautas sprendinys.

Visi sluoksniai pakuojami pagal šią euristiką:

8. Dėžė, kurios nors viena dimensija lygi sluoksnio storiui yra pati vertingiausia, jei tokios dėžės nėra tada naudojamos 1-3 euristikos.

Vėliau pakuojama naudojantis šių euristikų mišiniu (1), (2), (3), (8) ir Monte-Carlo paieška. Ši algoritmo dalis kartojama tam tikrą iteracijų skaičių kiekvienam sluoksniui.

Euristikos (1), (2), (3) ir (8) skirtos nustatyti, kuri dėžė bus naudama pakavimui. Tam, kad nustatyti kur dėžė turi būti padėta naudojamas galimų dėžės pozicijų sąrašas. Pradžioje šiame sąraše bus tik vienas taškas – konteinerio kampas (arba sluoksnio kampas, jei pakuojamas sluoksnis). Tada yra ieškoma dėžė, kuri gali tilpti į šią padėti. Jei dėžė telpa tada šis taškas pašalinamas ir įtraukiami trys papildomi taškai. Jei dėžė netelpta – bandoma ją pasukti. Jei dėžė netelpa jokioje padėtyje – ši dėžė ignoruojama.

(M. Juraitis, T. Stonys, A. Starinskas, D. Jankauskas, D. Rubliauskas)

LITERATŪRA

1. A. Žilinskas. Matematinis programavimas. Kaunas, VDU leidykla, 1999.

2. Informacines technologijos ’98 : konferencijos pranešimu medžiaga, 1998 m. sausio 28-29 d. ISBN 9986-13-597-4. Kaunas : Technologija, 1998, p. 267-269

3. MORABITO, R. and ARENALES, M. (1996), *Staged and Constrained two-dimensional guillotine cutting problems:An AND/OR-graph approach*, European Journal of Operationl Research, 94, 548-560.

4. MORABITO R., and ARENALES, M. (1995), *Performance of two heuristics for solving large scale two-dimensional cutting problems*, INFOR 33/2, p. 145-155

5. Janne Karelahti, *Solving the cutting stock problem in the steel industry*