UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

KOMBINIRANA METODA ENODIMENZIONALNEGA RAZREZA MATERIALA

Ljubljana, september 2002 PETER TRKMAN

IZJAVA

Študent Peter Trkman izjavljam, da sem avtor tega magistrskega dela, ki sem ga napisal pod mentorstvom prof. dr. Mira Gradišarja in skladno s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah dovolim objavo magistrskega dela na fakultetnih spletnih straneh.

V Ljubljani, dne 16. 9. 2002

Podpis: _____

Uvod

Enodimenzionalni razrez materiala se v praksi pojavlja v mnogih industrijskih procesih. Običajno je potrebno iz materiala, ki je trenutno na zalogi (na zalogi imamo torej različno število palic različnih dolžin), narezati zahtevano število krajših palic vnaprej določenih dolžin.

Različne oblike tega problema v zadnjih letih vzbujajo čedalje večjo pozornost raziskovalcev po vsem svetu. Pri tem imajo podjetja različne cilje, eden najpogostejših je poskus kar najbolj zmanjšati izgubo materiala, do katere pride pri razrezu. S tem se namreč zmanjšajo stroški materiala in posledično izboljša poslovni izid podjetja.

V magistrskem delu se ukvarjam z enodimenzionalnim razrezom, kar pomeni, da nas preostali dve dimenziji ne bosta zanimali. Metode za rešitev problema enodimenzionalnega razreza delimo na dve veliki skupini (razdelitev povzemam po Dyckoff, 1990, str. 156), in sicer:

- metoda vzorcev (angl. pattern oriented): z različnimi metodami določimo vzorce rezanja in frekvenco vsakega vzorca. Večino tovrstnih metod temelji na algoritmu, ki sta ga prva predstavila Gilmore in Gomory (Gilmore 1961, 1963). Tovrstne metode so uporabne le v primeru, če so vse palice v zalogi enakih dolžin (oziroma če imamo nekaj različnih standardnih dolžin),
- posamično obravnavanje (angl. item oriented): pri teh metodah ne določimo vzorcev, ampak vsako palico v zalogi individualno obravnavamo. Metode na tej podlagi so splošno uporabne, saj jih lahko uporabimo tako pri standardnih kot pri nestandardnih dolžinah palic v zalogi. Seveda so ob standardnih dolžinah palic metode vzorcev običajno bolj fleksibilne in primernejše za uporabo pri večjih problemih. Po drugi strani pri nestandardnih dolžinah pridejo v poštev samo metode iz druge skupine.

Pri posamičnem obravnavanju vsako palico, ki jo je potrebno razrezati, obravnavamo ločeno in za vsako palico posebej določimo načrt razreza. V kolikor so vse palice različnih dolžin, kot je primer tudi v tem magistrskem delu, je posamično obravnavanje tudi edini realistično možen pristop k reševanju (teoretično bi sicer lahko uporabili tudi metodo vzorcev, vendar bi bili v tem primeru edini možni frekvenci 0 in 1, tako da z uporabo metode vzorcev ne bi ničesar pridobili). Uporaba posamičnega obravnavanja je splošna in možna tudi v primeru, če so vse palice različnih dolžin, vendar je v takih primerih za večje primere običajno ustreznejša uporaba vzorcev.

Pri problemu, ki ga z različnih vidikov obravnavam v tem magistrskem delu, zaradi dejstva, da so vse palice na zalogi različnih dolžin, uporaba metod, ki temeljijo na vzorcih, v praksi ni možna, zato razvijam metodo, ki vsebinsko spada v drugo skupino. Obstoječe metode se namreč ukvarjajo predvsem s standardnim problemom razreza (angl. Standard One-Dimensional Cutting Stock Problem – S1D-CSP) in ne rešujejo problemov, kjer so vse palice na zalogi lahko različnih dolžin.

Primeri, kjer so vse palice lahko različnih dolžin (angl. General One-Dimensional Cutting Stock Problem – G1D-CSP), se lahko rešujejo eksaktno (na primer Trkman, 2002) ali hevristično (npr. Gradišar, 1997, 1999a). Problem eksaktne metode je, da je zaradi NP-

polnosti tega problema neprimerna za večje probleme. Problem hevrističnih metod je, da lahko privedejo do večjih izgub, saj tako pridobljena rešitev ni optimalna. Zato v tem magistrskem delu razvijem metodo, ki kombinira oba pristopa na tak način, da dobimo boljšo rešitev kot s hevrističnimi metodami in hkrati ohranimo nizko časovno kompleksnost pri reševanju.

Cilj teh metod je pridobiti ustrezen načrt razreza, ki bo minimiziral izgubo materiala ob upoštevanju morebitnih drugih omejitev in pogojev (na primer stroškov samega razreza). Pri tem upoštevam, da lahko največ eno palico, ki je vključena v načrt razreza, vrnemo nazaj v skladišče. To lahko naredimo le pod pogojem, da je ta palica daljša od neke vnaprej določene meje. Na ta način preprečimo nabiranje večje količine kratkih palic v skladišču, kar bi otežilo pripravo načrtov razreza v prihodnje in na ta način povečevalo bodoče izgube materiala.

Cilji magistrskega dela so predvsem naslednji:

- razviti eksaktno metodo za rešitev standardnega enodimenzionalnega problema razreza, ki bo omogočala poiskati optimalno rešitev posameznega problema manjšega obsega – torej tako rešitev, pri kateri je izguba materiala najmanjša,
- uporabiti to eksaktno metodo v kombinaciji z obstoječimi hevrističnimi metodami za razvoj kombinirane metode, ki vodi v skoraj optimalne rešitve tudi za večje probleme. Taka kombinacija mora nuditi boljše rešitve od hevrističnih metod in hkrati (za razliko od eksaktne metode) ohraniti ustrezno nizko časovno zahtevnost tudi za velike probleme enodimenzionalnega razreza,
- s pomočjo uporabe odločitvenih dreves razviti kriterije, ki bodo omogočali izbor ustrezne metode za posamezen problem,
- nakazati možne razširitve klasičnega problema razreza z vključevanjem stroškov skladiščenja in drugih kriterijev pri določanju načrta razreza ter prikazati možne rešitve tako razširjenega problema.

Metoda dela magistrskega dela je najprej predstavitev obstoječe literature na tem področju, pri čemer posebno pozornost namenjam obema obstoječima hevrističnima metodama ter novi eksaktni metodi za rešitev omenjene verzije problema. Nato obe metodi kombiniram tako, da pridobim boljšo metodo za reševanje problema. Razvito metodo tudi empirično testiram s pomočjo podatkov pridobljenih z generatorjem naključnih števil, razvitim v ta namen (generator je bil prvič predstavljen v Gau, 1995 in razširjen ter prilagojen za ta namen v Gradišar, 2002). Tako pridobljene primere rešujem z uporabo različnih reševalcev (solverjev) na osebnem računalniku.

S pomočjo odločitvenih dreves (predstavljenih v Quinlan, 1993) razvijem pristop za odločanje o izboru metode za posamezen primer glede na velikost problema in druge spremenljivke. Tak pristop omogoča izbiro prave metode za reševanje v veliki večini primerov.

Uporabil sem naslednjo programsko opremo: MPL/CPLEX solver, Visual Basic for Applications in program C5/See5.

Struktura poglavij v tem magistrskem delu je naslednja: v prvem poglavju predstavim literaturo s tega področja in sicer najprej splošno o problemu razreza, nakar se posvetim

enodimenzionalnemu problemu razreza. Prikažem tudi težave, ki jih pri reševanju povzroča NP-polnost tega problema.

V drugem poglavju podrobneje opišem različico enodimenzionalnega problema razreza, s katerim se ukvarjam v tem magistrskem delu. Nato predstavim eksaktno rešitev tega problema, ki temelji na metodi branch & bound. Eksaktno rešitev primerjam z dvema obstoječimi hevrističnima metodama za rešitev in prikažem pristop na podlagi odločitvenih dreves, ki v veliki večini primerov omogoča izbor prave metode za vsak posamezen primer.

Eksaktno in boljšo od obeh hevrističnih metod nato kombiniram v eno metodo, ki daje boljše ali vsaj enakovredne rešitve od vseh do sedaj obstoječih metod. Na koncu prikažem še možne razširitve problema, denimo ob upoštevanju oportunitetnih stroškov in stroškov skladiščenja. Uporabo vseh omenjenih metod za reševanje prikažem tudi na numeričnih primerih.

Pregled literature

Razrez splošno

Problem razreza (angl. cutting stock problem) je bil v taki obliki prvič formuliran pred več kot 50 leti (Paull, 1956) in je v vsem tem obdobju, predvsem v zadnjih letih, pritegnil veliko pozornost raziskovalcev, ki so se ukvarjali z različnimi oblikami tega problema in z različnimi možnimi pristopi k reševanju. Nekatere od teh oblik in pristopov predstavljam v nadaljevanju tega poglavja.

Osnovni problem je razrez materiala na zalogi v želeno količino in dolžino naročil, pri čemer je potrebno minimizirati izgubo, ki nastane pri tem razrezu. Tak problem se pojavlja v različnih panogah, kot so na primer papirna, tekstilna, kovinska itd. Na razpolago imamo torej material določene dolžine in širine. Razrez lahko opravimo na več različnih strojih. Nože na vsakem stroju je možno nastaviti na poljubno kombinacijo dolžin naročil, tako da skupna dolžina naročil ne presega dolžine palice. Naročila so določena kot število kosov posamezne dolžine ali kot skupna dolžina, ki jo rabimo za posamezno naročilo. Običajno se izgubi ne da povsem izogniti. Osnovni problem je torej v tem, kako naročila razporediti na posamezne kose materiala in stroje tako, da bo izguba materiala minimalna (Gass, 1985, str. 390-391). V praksi je seveda veliko različnih verzij tega problema.

V tem magistrskem delu se sicer ukvarjam s problemom enodimenzionalnega razreza, vendar je potrebno poudariti, da ima poleg problema razreza tudi veliko problemov v matematiki, operacijskih raziskavah, logistiki, proizvodnji in ekonomiji podobno logično strukturo. Dyckhoff (Dyckhoff, 1990) tako navaja naslednje probleme s podobno strukturo, ki se tako lahko rešujejo z istimi ali vsaj zelo podobnimi metodami:

- problemi razreza in problemi zmanjševanja odpadka,
- problemi pakiranja (kako določene predmete tako razporediti v določen prostor, da bo kar najmanjši del prostora ostal neizkoriščen),
- problemi nakladanja npr. na palete, v kontejnerje ali na vozila,
- problemi razporeditve, sortiranja, razvrščanja,

- določanje urnika pri večprocesorskih računalnikih, razporejanje kapitala, dodeljevanje spomina.

Kot že omenjeno, so ti problemi logično podobni in se lahko rešujejo s podobnimi metodami. Med različnimi tipi so nekatere vsebinske razlike – tako denimo (Vahrenkamp, 1996, str. 196) navaja razlike med problemi razreza in problemi pakiranja, kot je, da pri problemih razreza običajno potrebujemo več kosov posamezne dolžine naročila, medtem ko je to število pri problemih pakiranja manjše (običajno kar 1). Ravno tako je običajno cilj problemov razreza minimiziranje izgube materiala, pri problemih pakiranja pa običajno minimiziramo število uporabljenih posod. Kljub tem razlikam so metode za različne probleme medsebojno dovolj podobne, da jih navajam skupaj, pri čemer se posebej osredotočim na probleme razreza.

Za razporeditev posameznih problemov v kategorije je zelo pomembna naslednja klasifikacija (avtor te klasifikacije je Dyckhoff (Dyckhoff, 1990), kasneje jo je razširil Gradišar (Gradišar, 2002)). Ta klasifikacija vsak problem razdeli po 4 kriterijih (v oklepaju je oznaka za posamezno kategorijo):

1. Število dimenzij (gre za verjetno najpomembnejši kriterij za delitev problemov na različne kategorije) :
    a. ena dimenzija (1),
    b. dve dimenziji (2),
    c. tri dimenzije (3),
    d. n dimenzij, kjer je n>3 (N): 4-dimenzionalni problem pride v poštev, če vključimo še čas – denimo, da morajo biti 3-dimenzionalne škatle spravljene v zabojniku za določeno neprekinjeno časovno obdobje.
2. Tip problema:
    a. vsi veliki predmeti in izbor manjših predmetov: gre za t. i. probleme nahrbtnika, kjer moramo v enega ali več večjih predmetov spraviti izbor manjših predmetov – izbor izvedemo glede na relativno pomembnost posameznega predmeta (B),
    b. izbor velikih predmetov in vsi manjši: običajno pri problemih razreza in pakiranja – vse manjše predmete moramo razporediti v izbor večjih predmetov, tako da dosežemo čim manjšo izgubo ali realiziramo kakšen drug cilj (V).
3. Vrsta velikih predmetov:
    a. en predmet (O): imamo samo en velik predmet,
    b. več identičnih predmetov (I): imamo več velikih predmetov, ki so vsi enaki,
    c. nekaj skupin identičnih predmetov (G) – to je omenjena razširitev v (Gradišar, 2002) – imamo več velikih predmetov, ki so v nekaj standardnih velikostih,
    d. različni predmeti (D) – vsi veliki predmeti so medsebojno različni.
4. Vrsta majhnih predmetov:

a. nekaj predmetov (različnih oblik) (F) – imamo razmeroma majhno število medsebojno različnih predmetov,
b. veliko predmetov različnih oblik (M) – imamo večje število medsebojno različnih predmetov,
c. veliko predmetov, ki niso enake oblike (R) – npr. pakiranje 1000 predmetov velikosti med 0 in 1 v zabojnike velikosti 1,
d. predmeti ene oblike (C): npr. nakladanje palet.

Poleg te klasifikacije se ponekod (npr. Hinxman, 1980; Abraham, 1976) omenja še t. i. 1 ½ dimenzionalne probleme, kjer lahko pri vsakem razrezu uporabimo samo določeno manjše število osnovnih načrtov.

Velika večina problemov razreza je NP polna (angl. NP complete), kar pomeni, da večjih problemov običajno ni možno rešiti eksaktno. Probleme v operacijskih raziskavah lahko namreč v splošnem razvrstimo v dve vrsti:

- problemi, za katere obstajajo algoritmi, ki jih rešijo v polinomskem času, torej v $O(n^r)$ zahtevnosti, kjer je r konstanten (Kozak, 1986, str. 24). Gre torej za probleme, ki jih je možno rešiti v polinomskem času na deterministočnem Turingovem stroju (NP Complete, 2002). Ta razred problemov označimo s črko P.
- problemi, za katere smo uspeli najti le rešitve eksponenčne zahtevnosti. Ti problemi sodijo v razred NP. Ta oznaka označuje probleme, ki jih lahko rešimo z nedeterminističnim algoritmom v polinomskem času (Kozak, 1986, str. 24).

Problem je NP poln, če je v razredu NP in se da vsak drug problem v NP prevesti v ta problem s polinomskim algoritmom. Če bi torej našli učinkovit (polinomski) algoritem za rešitev enega NP polnega problema, bi to pomenilo, da lahko s polinomskim algoritmom rešimo tudi vse ostale NP polne probleme (Schrijver, 1986, str. 21). Do sedaj tak algoritem še ni bil najden, splošno (sicer nedokazano) mnenje je, da ne obstaja (NP Complete, 2002)

Razlika med polinomskimi in eksponentnimi algoritmi je razvidna iz tabele 1, kjer je prikazana sprememba v velikosti problema, ki ga lahko rešimo ob uporabi hitrejšega računalnika. Za algoritme s polinomsko časovno zahtevnostjo (prve štiri vrstice v tabeli) se ta velikost hitro povečuje, za oba eksponentna algoritma pa bistveno počasneje.

Iz te tabele je razvidno, da za večino NP-polnih problemov razreza tudi v prihodnosti ne moremo pričakovati eksaktnih rešitev za velike probleme.

funkcija časovne zahtevnosti	velikost problema s sedanjim računalnikom	100-krat hitrejši računalnik	1000-krat hitrejši računalnik
n	N_1	$100\ N_1$	$1000\ N_1$
n^2	N_2	$10\ N_2$	$31{,}6\ N_2$
n^3	N_3	$4{,}64\ N_3$	$10\ N_3$
n^5	N_4	$2{,}5\ N_4$	$3{,}98\ N_4$
2^n	N_5	$N_5 + 6{,}64$	$N_5 + 9{,}97$
3^n	N_6	$N_6 + 4{,}19$	$N_6 + 6{,}29$

Tabela 1: Vpliv hitrejšega računalnika na polinomske in eksponentne algoritme (Garey, 1979, str. 8).

V tem poglavju na kratko predstavim še ostale tipe razreza (predvsem dvodimenzionalni razrez in dvodimenzionalno pakiranje) in metode, ki so se v zadnjih letih uveljavile za njihovo reševanje. V nadaljevanju se posvetim izključno enodimenzionalnemu razrezu.

Dvodimenzionalno pakiranje (kot že omenjeno, gre pri razrezu za analogen problem) pomeni, da imamo n pravokotnih predmetov z določeno dolžino in širino ter neomejeno količino zabojnikov ravno tako z neko dolžino in širino. Naš cilj je razporeditev teh predmetov v zabojnike brez prekrivanja, tako da so robovi predmetov vzporedni z robovi zabojnikov – predmetov torej ne moremo obračati. Če imajo vsi predmeti enako širino, gre za enodimenzionalen problem pakiranja (1BP). Tako enodimenzionalno kot dvodimenzionalno pakiranje sta NP-težka (Lodi, 2002, str. 379). Različni pristopi za reševanje tega problema so predstavljeni v (Lodi, 2002), druge variante problema, kjer lahko denimo predmete tudi rotiramo in podobno, pa v (Lodi, 1999).

Ena od predstavljenih rešitev je denimo eksaktna rešitev, ki s kombinacijo uporabe metode razveji&omeji (angl. branch&bound) in dinamičnega programiranja eksaktno reši problem, kjer je potrebno serijo pravokotnikov razporediti v pas fiksne širine in neomejene dolžine (Hifi, 1998). V (Carnieri, 1993) je predstavljen hevristični algoritem za razrez stavbnega lesa z napako (en del lesa je neprimeren za uporabo in se ne sme uporabiti v načrtu razreza). Tudi tu je kot glavni problem omenjeno zelo veliko število možnih vzorcev. V (Pisinger, 2002) je več hevrističnih algoritmov za reševanje problema pakiranja pravokotnih škatel v zabojnik. Najboljši od teh algoritmov (kateri je najboljši, je bilo ugotovljeno na podlagi statistične analize) pri večjih problemih doseže do 95% izkoriščenost zabojnika.

Za razliko od teh metod, kjer je glavni cilj minimiziranje izgube, se Bhadury (Bhadury, 1996) ukvarja z minimiziranjem dolžine rezov, ki so potrebni, da izrežemo določen vzorec iz razpoložljivega materiala (s tem namreč dosežemo krajši čas razreza, hkrati pa tudi manjšo obrabo rezila).

Enodimenzionalni razrez

V splošnem za enodimenzionalni razrez lahko uporabljamo metode, temelječe na dveh principih:

- metode vzorcev,
- posamično obravnavanje.

Pri metodah vzorcev je pri iskanju optimalne rešitve največji problem veliko število možnih vzorcev, ki jih lahko uporabimo za razrez posamezne palice. Za srednje velik problem, kjer je denimo 50 različnih dolžin naročil in posamezno palico razrežemo na 5

delov (iz vsake palice torej pridobimo 5 različnih ali enakih dolžin naročil), lahko število možnih vzorcev izračunamo po formuli za kombinacije s ponavljanjem[1]:

$$C^{r}_{n++r-1} = \binom{n+r-1}{r}$$

V našem primeru je torej n=50, r=5, saj iz množice s 50 elementi (število različnih dolžin naročil) izbiramo po 5 elementov (število dolžin naročil, ki se uporabijo za razrez posamezne palice). Tako dobimo rezultat, da lahko za posamezno palico uporabimo 3.162.510 različnih vzorcev. Ker moramo pri razrezu običajno uporabiti več palic, je število vseh možnih kombinacij vzorcev še bistveno večje. Iz tega je razvidno, da je vsaka metoda, ki bi se poskušala do rešitve prebiti z generiranjem in preizkusom vseh možnih vzorcev, vnaprej obsojena na neuspeh. Zato metode, ki temeljijo na tem pristopu, za večje probleme ne preizkusijo vseh vzorcev, ampak samo nekatere možne. Tako se na različne načine najprej določijo najboljši vzorci in nato frekvenca za vsak posamezen vzorec. Od delovanja metode je odvisno, ali zagotovo privede do optimalne rešitve ali ne.

Tako pri metodah vzorcev kot pri posamičnem obravnavanju lahko uporabljamo eksaktne in hevristične metode za reševanje problemov.

Eksaktne metode temeljijo na algoritmih, ki zagotovo v končnem številu korakov privedejo do optimalne rešitve problema. Poglavitna prednost teh metod je prav v tej optimalnosti rešitve. Glavna pomanjkljivost je hitro naraščanje potrebnega časa za določitev te rešitve ob večanju problema zaradi omenjene NP polnosti večine problemov razreza. Eksaktne metode običajno uporabljajo različne tehnike za pridobitev rešitve kot so na primer:

- linearno programiranje (obsežen pregled metod, ki temeljijo na linearnem programiranju je v (Carvalho, 2002)),
- metoda razveji & omeji – to metodo nekoliko podrobneje predstavljamo še v nadaljevanju tega magistrskega dela,
- dinamično programiranje.

Hevristične metode so v splošnem primernejše za večje probleme. V poštev pridejo predvsem v primerih, kjer algoritem za optimalno rešitev še ni bil najden ali je tak algoritem preveč računsko zahteven za večje primere. Tovrstne metode se štejejo kot ustrezne, če so tako pridobljene rešitve dovolj dobre. To pomeni, da dokazano ali verjetno odstopajo od optimalne rešitve za manj od neke, po različnih kriterijih določene, še sprejemljive vrednosti. Dokaz, da rešitev odstopa manj kot za to vrednost, je lahko matematičen ali statističen na podlagi večjega števila generiranih in rešenih problemov (Hinxman, 1980).

Problem pri hevrističnih metodah je tudi ta, da so običajno precej specifične in primerne le za posamezen primer. Hevristična metoda, ki dobro deluje na enem primeru ob določenih ciljih in predpostavkah, namreč zelo verjetno ne bo ustrezna za kakšen drug problem, čeprav sta oba problema medsebojno navidez morda zelo podobna.

[1] Formulo za kombinacije s ponavljanjem moramo izbrati zato, ker lahko posamezen element v izbrani podmnožici nastopa večkrat. Povedano drugače - iz vsake palice lahko torej izrežemo več enakih dolžin naročila

Hevristične metode lahko glede na način, kako poskušajo pridobiti dobro rešitev problema, razdelimo na več skupin (Hinxman, 1980; Nilsson, 1971):

- preiskava grafa (angl. state-space search): delne rešitve problema vzamemo kot vozle v grafu in poiščemo najboljšo pot v grafu od začetnega (v celoti nerešenega problema) do končnega stanja (končna rešitev problema, ki je optimalna ali vsaj dovolj dobra),
- razdelitev problema (angl. problem reduction): osnovni problem razdelimo na več manjših podproblemov in vsak podproblem rešimo ločeno (ker so te podproblemi manjši, lahko uporabimo tudi eksaktne metode). Končna rešitev je unija posameznih rešitev,
- prekinitev iteracije (angl. cut-off): v kolikor eksaktna metoda temelji na pridobitvi optimalne rešitve z iteracijami, jo lahko pretvorimo v hevristično tako, da določimo pogoj, kdaj naj se iteracije prekinejo. Iteracije lahko prekinemo, ko se rešitev razlikuje od optimalne za manj kot neko določeno vrednost ali ko bi stroški in čas računanja presegli neko še sprejemljivo mejo,
- iskanje sprejemljive rešitve (angl. aspiration level): v kolikor hevristična metoda temelji na pridobivanju različnih možnih rešitev za problem, je možen pristop, da sprejmemo prvo rešitev, ki izpolnjuje nek vnaprej določen kriterij,
- ponavljanje vzorca (angl. repeated exhaustion reduction): tovrstne metode najprej pridobijo nek ustrezen vzorec z majhno izgubo. Ta vzorec se nato ponovi čim večkrat, dokler ne pride do prevelike proizvodnje posamezne dolžine naročila. Nato popravimo podatke in določimo nov vzorec, ki ga spet uporabimo čim večkrat. Tako nadaljujemo do popolne izpolnitve vseh dolžin naročila,
- vzorčenje (angl. sampling): gre za podoben pristop kot pri iskanju sprejemljive rešitve: identificiramo več možnih rešitev in za vsako izračunamo vrednost kriterijske funkcije. Nato izberemo točko z najboljšo (najnižjo pri minimizacijskih, najvišjo pri maksimizacijskih problemih) vrednostjo kriterijske funkcije.

Prva kvalitetna in najbolj znana metoda za pridobitev dobre hevristične rešitve na podlagi vzorcev je metoda Gilmorja in Gomoryja (Gilmore, 1961; Gilmore 1963). Verzija problema, s katerim se ukvarjata v prvem članku, je naslednja: na zalogi imamo palice standardnih dolžin (na voljo je neomejeno število kosov vsake dolžine). Zagotoviti moramo določeno število kosov vnaprej znanih dolžin (zaradi neomejene količine zaloge bo naročilo možno izpolniti v vsakem primeru, razen v kolikor je najdaljša dolžina naročila večja od najdaljše dolžine palice). Vsaka palica na zalogi ima določeno ceno, skupni stroški razreza so vsota stroškov vseh v razrezu uporabljenih palic (Gilmore, 1961, str. 850). Naš cilj je poiskati načrt razreza z minimalnimi stroški.

Osnovni problem smo že omenili – gre za veliko število možnih vzorcev. Ta problem je v tej metodi rešen tako, da se v vsaki fazi simpleks metode, ko potrebujemo nov stolpec oziroma novo aktivnost[2], ne pregleda celotna zbirka možnih stolpcev, ampak se ustrezen

[2] Z izrazom aktivnost je mišljen načrt razreza za posamezno dolžino palice v zalogi – v bistvu gre torej za vzorec.

stolpec najde z rešitvijo pomožnega problema. V tem primeru gre pri tem problemu za tako imenovani problem nahrbtnika (angl. knapsack), ki ga je možno rešiti z različnimi metodami (pregled eksaktnih metod za rešitev problema nahrbtnika je v (Dudzinski, 1987), pregled približnih pa v (Fisher, 1980)).

V poznejšem članku (Gilmore, 1963) je ta metoda prilagojena za uporabo pri razrezu papirja. Tako so narejene nekatere spremembe v formulaciji problema (omejitve pri številu nožev za rezanje, možnost približne izpolnitve naročila kupca) in v algoritmu za rešitev, ki predvsem vpliva na hitrejšo pridobitev rešitve (hitrejša metoda za rešitev problema nahrbtnika, način za zmanjšanje velikosti tega problema…). Osnovna ideja metode ostaja nespremenjena – pri pridobivanju rešitve v vsakem koraku uvedemo samo en nov stolpec (vzorec) in ne upoštevamo vseh možnih vzorcev.

V kasnejših letih se je razvilo veliko število metod, ki rešujejo take ali podobne probleme – nekatere metode gradijo na osnovnem pristopu Gilmorja in Gomoryja in ga prilagodijo za konkreten primer, druge iščejo drugačne poti do končne rešitve, ki je bodisi optimalna bodisi neka dobra hevristična rešitev.

Možno rešitev za enodimenzionalni razrez z eno dolžino palice je predlagana v (Vahrenkamp, 1996), kjer je uporabljen genetski algoritem, pri katerem se v vsaki fazi algoritma upošteva samo tako imenovane učinkovite vzorce – to so taki vzorci, kjer je izguba manjša od neke vnaprej določene meje. Tak način sicer ne zagotavlja optimalne rešitve, pač pa neko rešitev, ki je tej blizu, dostikrat pa dobimo celo optimalno rešitev.

V (Antonio,1999) sta predstavljeni dve metodi na podlagi dinamičnega programiranja, ki temeljita na tem, da v vsaki fazi obdržimo samo tiste rešitve, za katere je najverjetneje, da bodo privedle do dobre rešitve. Manjše število rešitev, ki jih obdržimo v posamezni fazi, pomeni krajši čas reševanja, hkrati pa tudi večjo verjetnost, da bo tako dobljena rešitev slaba. Pri eni od predstavljenih metod je tako poudarek na času reševanja (ta se uporablja, kadar je potreben takojšen odgovor na povpraševanje kupcev), pri drugi na kvaliteti rešitve (ta se uporablja za pridobitev dobrih rešitev v nočnem času).

Gradišar (Gradišar, 1999a) predstavlja podoben problem, kjer je večina zalog v obliki ene ali nekaj standardnih dolžin palic, del zaloge pa predstavljajo nestandardne dolžine palic, ki so večinoma še uporabni ostanki iz prejšnjih razrezov. Predstavljena rešitev je dvofazna: najprej generiramo vzorce, nato v drugi fazi odstranimo tiste vzorce, ki vsebujejo več naročil, kot jih potrebujemo (s tem preprečimo preveliko proizvodnjo posameznih naročil). Seveda s tem za nekatera naročila ne dobimo dovolj kosov. Ta del nato rešimo z ustrezno zaporedno hevristično proceduro, končno rešitev pa dobimo tako, da združimo rezultate obeh faz.

Pomanjkljivost metode Gilmorja in Gomoryja, ki generira le majhen del vzorcev delno popravlja Degraeve (Degraeve, 1999), ki kombinira to metodo z metodo razveji&omeji. S pomočjo te metode pridobi tiste stolpce, ki jih Gilmorjeva metoda ne upošteva, lahko pa privedejo do optimalne rešitve. Predstavljena metoda vodi do optimalnih rešitev in je nekoliko boljša od prejšnje metode (Wäscher, 1994), ki je do zagotovo optimalne rešitve privedla v 3984 od 4000 primerov.

Prej omenjen problem prevelikega števila možnih vzorcev, ki vodijo do računalniško nerešljivih problemov, je možno rešiti tako, da vnaprej generiramo možne vzorce s pomočjo iskalnega drevesa (angl. search tree) in nato pri izdelavi načrta razreza upoštevamo samo te vzorce (Suliman, 2001).

Belov (Belov, 2002) predstavlja metodo za rešitev problema razreza s palicami različnih dolžin (na zalogi imamo več primerkov palice posameznega tipa). Pri tej metodi, ki je primernejša za probleme z več različnimi tipi palic kot za tiste z manj, kombinira tako imenovani cutting plane algorithm z generacijo stolpcev. Na ta način doseže optimalno rešitev pri približno 90% testnih primerov ob časovni omejitvi nekaj minut.

Do sedaj predstavljene rešitve so predpostavljale, da je edini cilj doseči čim manjšo možno izgubo materiala. Dostikrat pri kreiranju načrta razreza zasledujemo tudi druge cilje kot so na primer čim manjše število vzorcev, čim krajši čas razreza in podobno.

V (Umetani, 2002) je tako predstavljena hevristična metoda, pri kateri je glavni cilj minimiziranje števila vzorcev, ki jih uporabimo v načrtu razreza. To je izbrano kot glavni cilj na podlagi predpostavke, da največjega stroška pri razrezu materiala ne predstavlja morebitna izguba materiala ampak premestitev rezil, ki je potrebna ob vsaki spremembi vzorca. Tudi tu izguba materiala ni nepomembna – postavljena je namreč omejitev, koliko največ je lahko izguba posameznega vzorca. Vzorci, ki to omejitev presegajo, ne pridejo v poštev za razrez. Podobna je metoda, ki ravno tako kreira vzorce in omogoča vključitev drugih faktorjev v izdelavo načrta razreza, kot so denimo stroški menjave vzorcev in prihodek od morebitnega presežka narezane količine (Schiling, 2002). Pregled nekaterih novejših metod za probleme razreza in pakiranja je tudi v (Wang, 2002), pregled nekaterih metod odkritih pred letom 1990 v (Sweeney, 1992), pregled metod za pakiranje pa v (Coffman, 1984).

Westerlund (Westerlund, 1998) predstavlja dvofazno metodo za minimiziranje stroškov pri razrezu v papirni industriji. Stroške predstavljajo izguba materiala, skladiščenje delno porabljenih navitkov papirja ter spremembe v nastavitvi stroja in nožev. Prvi korak metode je nelinearna procedura za generiranje vseh možnih vzorcev razreza. V drugem koraku se končna rešitev pridobi z rešitvijo mešano celoštevilskega linearnega problema.

Vasko (Vasko, 1999) opisuje algoritem DYNACUT_CS, ki se uporablja za razrez jekla. Ta algoritem na podlagi hierarhičnega pristopa zasleduje naslednje 3 cilje (cilj so navedeni po pomembnosti v padajočem vrstnem redu):

- kar najbolje izpolniti naročilo kupca (torej imeti čim manj nenarezanih naročil) – to je glavni cilj tega algoritma,
- čim bolj zmanjšati razrez iz predobrega materiala (angl. overgrading): dostikrat se zgodi, da kupec naroči material slabše kakovosti. Če podjetje tega naročila ne more izpolniti s to kakovostjo, ga lahko izpolni z boljšim materialom, kar seveda pomeni ustrezno višje stroške, ki jih mora kriti podjetje, saj stroškov višje kakovosti običajno ni možno prevaliti na kupca. Zato algoritem poskuša tak razrez iz predobrega materiala čim bolj zmanjšati,

- čim manjša izguba materiala – kot vidimo, je ta cilj šele na tretjem mestu, saj je za doseganje poslovnih ciljev podjetja pomembnejša izpolnitev drugega in predvsem prvega cilja.

Podoben kot zadnji primer je predstavljen v (Rönnqvist, 1995), kjer imamo pri razrezu lesa v isti deski les različne kvalitete. Poglavitna omejitev pri tem problemu je zelo kratek čas za odločitev, saj se mora načrt razreza izdelati v 2 sekundah, nato pa gre deska na žago. Zato je izvajanje algoritma omejeno na 10 iteracij, ki običajno najdejo dobro rešitev, ki ni nujno tudi optimalna.

V (Chengbin, 1999) je predstavljen hevristični algoritem za minimiziranje izgube materiala in časa razreza pri rezanju kovinskih cevi. Kriterijska funkcija je tu ponderirana vsota izgube in časa, potrebnega za izvedbo razreza (čas se lahko skrajša s hkratnim razrezom več palic). Podobno kot v (Antonio, 1999) sta predstavljeni dve verziji algoritma – ena hitro privede do solidne rešitve, druga v nekaj daljšem času do rešitve, ki je blizu optimalne. Ta algoritem pripelje do rešitev, ki so v povprečju za 8% boljše od tistih, ki so jih prej ročno izdelali izkušeni zaposlenci.

Standardnega problema razreza se lahko lotimo tudi tako, da poiščemo rešitev LP relaksacije, nato pa realne vrednosti zaokrožimo na celoštevilčne. Dve hevristični metodi za zaokroževanje, ki vodita do dobrih (dostikrat tudi optimalnih) rešitev sta predstavljeni v (Scheitauer, 1995).

Faggioli (Faggioli, 1998) predstavlja trifazno metodo za problem določanja vrstnega reda razreza. Najprej se s požrešnim algoritmom (angl. greedy algorithm) pridobi dobra začetna rešitev. Ta rešitev se nato izboljša z generaliziranim lokalnim iskanjem. Tako dobljena rešitev se nato uporabi kot zgornja meja pri iskanju optimalne rešitve z implicitnim oštevilčenjem (angl. implicit enumeration). Kot pri vseh metodah za optimalno reševanje je problem hitro naraščanje časa z naraščanjem velikosti problema – za problem s 40 vzorci se tako porabi skoraj cel dan za optimalno rešitev. Seveda lahko v tem primeru obdržimo rešitev 2. faze kot končno (hevristično) rešitev problema. Ta rešitev potem seveda ni optimalna, je pa večinoma blizu optimalne in predvsem zahteva bistveno manj časa reševanja.

Z določanjem vrstnega reda razreza se ukvarja tudi Yuen (Yuen, 1995), ki ravno tako predstavlja dve metodi. Enostavnejša se lahko uporablja le v nekaterih specifičnih primerih, medtem ko je druga splošneje uporabna. Pri drugi metodi, ki preišče vsa možna zaporedja razreza, je spet problem čas računanja, ki narašča več kot proporcionalno z naraščanjem števila vzorcev. Določanje vrstnega reda na različnih strojih, na katerih se izvaja razrez, je bistveno tudi pri primeru razreza jekla v gradbeništvu, ki je predstavljen v (Armbuster, 2002).

Degraeve (Degraeve, 1998) je avtor mešano celoštevilskega modela za razrez v tekstilni industriji, kjer minimizira število potrebnih nastavitev ob ohranitvi nizke izgube. Model omogoča upoštevanje različnih omejitev, kot so na primer dolžina nožev, debelina tkanine in dolžina mize, na kateri izvajamo razrez.

Podobno Liang (Liang, 2002) v svoji metodi na podlagi evolucijskega programiranja poleg čim manjše izgube zasleduje tudi cilj, da bi bilo čim manjše število palic z izgubo. Pri tem želi minimizirati tako število uporabljenih kot predvsem število delno uporabljenih palic.

Nekoliko drugačen od prej omenjenih problemov enodimenzionalnega razreza je problem večfaznega razreza (Zak, 2002). Tu najprej material na prvem stroju razrežemo v polizdelke, nato pa razrez dokončamo na drugem stroju. Za pridobitev dobre rešitve za ta problem je bila uporabljena posplošitev Gilmorjeve metode generiranja stolpcev.

Več ciljev ima tudi metoda, ki uporablja linearno programiranje za maksimiranje celotnih stroškov, povezanih z razrezom, kot so denimo stroški prevelike količine narezanih naročil, stroški materiala, stroški skladiščenja in seveda stroški izgube materiala. Pri tem je možen problem tudi pridobitev natančnih in popolnih podatkov, denimo o stroških zalog (Wäscher, 1990).

Ekskatna rešitev za problem, pri katerem je cilj minimiziranje potrebnega števila vzorcev za izpolnitev naročila (s tem dosežemo tudi minimiziranje potrebnih sprememb nastavitev rezila in tako praviloma krajši čas razreza), je predstavljena v (Vanderbeck, 2000). S to metodo je avtorju v dveh urah uspelo optimalno rešiti 12 od 16 testnih problemov.

Carvalho (Carvalho, 1995) predstavlja dvofazni problem razreza, kjer v prvi fazi najprej dobimo vmesne navitke, potem pa te v drugi fazi razrežemo na končne izdelke. Ta metoda poleg čim manjše izgube zasleduje tudi druge cilje, kot so na primer čim manjše število vzorcev, čim manjše število vmesnih navitkov in čim manjše število različnih vmesnih navitkov (zaradi lažjega načrtovanja razreza v drugi fazi). Po ugotovitvah avtorja je ta metoda primerna za manjše in srednje velike probleme, za večje pa ne (oziroma vsaj ni bila primerna v času nastanka metode).

Dvofazni je tudi algoritem za razrez v jeklarstvu (Ferreira, 1990), ki temelji na modificiranem Gilmorjevem algoritmu. Hevristična procedura generira dobre vzorce za obe fazi in vsak vzorec uporabi čim večkrat. Ta procedura se ponavlja, dokler niso izpolnjene vse potrebe. Najprej poiščemo rešitev za drugo fazo, na podlagi pridobljenih rezultatov še za prvo, pri čemer se najprej poišče rešitev za dolga naročila. Pri iskanju rešitve se upoštevajo stroški materiala in stroški spremembe vzorca.

Wagner (Wagner, 1999) se ukvarja z razrezom stavbnega lesa. Ker ta metoda prvotno dobi necelоštevilske rezultate, je poseben poudarek na zaokrožitvi teh števil – metoda za zaokroževanje lahko namreč pomembno vpliva na končno izgubo. Ugotovljeno je tudi, da iz razumljivih razlogov metode na podlagi linearnega programiranja običajno najdejo rešitev z več zamenjavami vzorcev kot druge metode. V to metodo je možno vključiti tudi druge cilje, kot so na primer čim manjše število zamenjav vzorcev ali minimalno število porabljenih kosov materiala.

Po tej kratki predstavitvi metod za druge primere se v nadaljevanju ukvarjamo le še z našo verzijo problema, ki jo podrobneje predstavljamo v naslednjem poglavju.

Opis problema

Problem, s katerim se ukvarjamo v tem magistrskem delu, je ena od številnih različic enodimenzionalnega problema razreza. Kot že omenjeno, se ta problem od preostalih do sedaj v magistrskem delu predstavljenih problemov razlikuje toliko, da ni mogoče uporabiti znanih hevrističnih metod. Poglavitni razliki sta, da so vse palice različnih dolžin in da največ ena delno narezana palica ne šteje kot ostanek.

Problem je definiran tako: za izpolnitev naročila (vsako naročilo predstavlja točno določeno število kosov posamezne dolžine) imamo na razpolago določeno število palic, dolžina palic je v splošnem medsebojno različna. Predpostavljamo, da so vsi podatki celoštevilčni, v kolikor niso celoštevilčni, jih je možno pomnožiti z določenim faktorjem in tako dobiti celoštevilčne podatke.

Cilj je pridobiti tak načrt razreza (pod tem terminom smatramo natančen načrt, katere palice bomo uporabili pri razrezu in katere dolžine naročil bomo izrezali iz vsake posamezne palice), ki bo prinesel čim manjšo izgubo materiala – gre torej za enokriterijski problem, ki ga le v zadnjem delu magistrske razširimo še z nekaj dodatnimi kriteriji.

V splošnem ločimo primere, kjer je dovolj materiala, in primere, kjer materiala primanjkuje. V obeh primerih je cilj minimizirati izgubo materiala, pri čemer v kolikor materiala primanjkuje, del naročila ne more biti izpolnjen.

Tako palice kot naročila smatramo kot enodimenzionalne – to pomeni, da sta dve dimenziji fiksni, nas pa zanima samo načrt razreza za tretjo dimenzijo – gre torej za enodimenzionalni razrez. V skladu z Dyckhoffovo klasifikacijo (Dyckhoff, 1990) lahko naš problem razreza uvrstimo v kategorijo 1/V/D/M za primere z dovolj materiala oziroma v kategorijo 1/B/D/M za primere s premalo materiala

V primerih z dovolj materiala nam ostane ena palica, ki ni do konca narezana. Ta ostanek lahko vrnemo v skladišče in je uporaben tudi v kasnejših načrtih razreza. Zato ne šteje kot izguba, vendar le v kolikor je daljši od neke vnaprej določene zgornje meje. Z eno od omejitev v modelu (omejitev 5) določimo, da lahko v skladišče vrnemo največ eno tako preostalo palico. Tako namreč preprečimo take načrte razreza, ki bi vodili do hitrega naraščanja števila palic v skladišču. Te palice bi bile namreč čedalje krajše, kar bi vodilo do večjih izgub v kasnejših razrezih.

V modelih uporabljamo naslednje spremenljivke:

s_i = dolžina naročil; i = 1,...,n,

b_i = potrebno število kosov dolžine s_i,

d_j = dolžine palic v zalogi; j = 1,...,m,

x_{ij} = število kosov dolžine s_i, ki smo jih narezali iz palice j,

UB – zgornja meja za izgubo – palica daljša od te meje se lahko uporabi še kasneje in zato ni nujno, da šteje kot izguba

Model se seveda razlikuje za primere z dovolj in primere s premalo materiala:

Primer 1: Naročilo lahko izpolnimo, saj imamo na zalogi dovolj materiala (izpolnjen je torej pogoj $\sum_{j=1}^{m} d_j \geq (\sum_{i=1}^{n} s_i * b_i)$ - celotni razpoložljivi material na zalogi presega ali je vsaj enak celotnim potrebam):

(1) $\min \sum_{j=1}^{m} t_j$ (minimiziraj vsoto ostankov, ki štejejo kot izguba)

s pogojem, da

(2) $\sum_{i=1}^{n} (s_i * x_{ij}) + \delta_j = d_j (1 - y_j) \quad \forall j$ (nahrbtnik (angl. knapsack) omejitev)

(3) $\sum_{j=1}^{m} x_{ij} = b_i \ \forall i$ (omejitve povpraševanja – število potrebnih kosov je fiksno)

(4) $UB - \delta_j + UB(u_j - 1) \leq 0 \ \forall j$ (u_j kaže, ali ostanek palice šteje kot izguba)

(5) $\sum_{j=1}^{m} u_j \leq 1$ (največje število preostalih palic, ki ne štejejo kot izguba, saj so daljše od UB)

(6) $\delta_j - t_j - (u_j + y_j) * (\max d_j) \leq 0 \ \forall j$ (t_j kaže, kakšna je izguba pri palici j, in je enak δ_i, razen pri neuporabljenih palicah in pri eni palici, ki ne šteje kot izguba)

(7) $UB \leq \max s_i$ ali $UB \leq \min s_i$

$x_{ij} \geq 0$, celoštevilčni $\forall i, j$

$t_j \geq 0$ $\forall j$

$\delta_j \geq 0$ $\forall j$

$u_j \in \{0,1\}$ $\forall j$

$y_j \in \{0,1\}$ $\forall j$

Primer 2: Naročilo zaradi pomanjkanja materiala ne more biti v celoti izpolnjeno

(1) $\min \sum_{i=1}^{n} \delta_j$ (minimiziraj vsoto izgub)

s pogojem, da

(2) $\sum_{i=1}^{n} (s_i * x_{ij}) + \delta_j = d_j \quad \forall j$ (nahrbtnik omejitev)

(3) $\sum_{j=1}^{m} x_{ij} \leq b_i \ \forall i$ (omejitve povpraševanja – posamezne dolžine naročila ne smemo narezati več, kot je povpraševanje po njej)

(4) $x_{ij} \geq 0$, celoštevilčni $\forall i, j$

$\delta_j \geq 0$ $\forall j$.

V prvem primeru minimiziramo vsoto ostankov, ki štejejo kot izguba. Kot je že omenjeno, največ ena palica ne šteje kot izguba, v kolikor je daljša od zgornje meje. Druga omejitev

določa, da vsota dolžin narezanih naročil iz ene palice ne more biti daljša od dolžine te palice, v kolikor je ta palica vključena v načrt razreza (y_j=0), oziroma mora biti enaka 0, v kolikor ta palica ni v načrtu razreza (y_j=1). Razlika med dolžino palice in vsoto dolžin narezanih naročil je enaka ostanku pri tej palici.

Tretja omejitev določa, da mora biti število narezanih kosov natančno enako potrebam – morebitno narezanje večjega števila kosov, kot je bilo zahtevano, ni dopustno. V 4. omejitvi preverjamo, ali ta palica šteje kot izguba ali ne. u_j je lahko enako 1 le, v kolikor je ostanek pri tej palici daljši od določene UB.

V 5. omejitvi omejimo število palic, ki so vključene v načrt razreza, njihov ostanek pa ne šteje kot izguba. Število takih palic je lahko največ 1. Brez te omejitve bi namreč prišlo do takih načrtov razreza, kjer bi iz vsake palice izrezali le nekaj naročil, preostanek pa vrnili v skladišče. S tem bi sicer načeloma zelo enostavno prišli do načrtov razreza z izgubo 0, po drugi strani bi to seveda povzročilo povečevanje števila čedalje krajših palic v skladišču, posledično pa višanje stroškov skladiščenja (zaradi krajših dolžin palic ob vedno večjem številu kosov) in predvsem večanje izgub pri kasnejših načrtih razreza. Zato je ta omejitev nujna.

S 6. omejitvijo določimo razmerje med δ_j (ostankom posamezne palice) in t_j (izgubo pri posamezni palici). Izguba je enaka ostanku pri vseh palicah razen pri tistih, ki sploh niso bile uporabljene v načrtu razreza (torej imajo y_j=1) in pri največ eni palici, ki ne šteje kot izguba (torej ima u_j=1).

V 7. omejitvi določimo še, kakšna naj bo UB (več o tem še v nadaljevanju). Nato določimo, da sta spremenljivki u_j in y_j binarni in da je spremenljivka x_{ij} celoštevilčna. Pogoj, da sta celoštevilčni tudi spremenljivki δ_j in t_j, ni potreben, saj to zagotavljata že omejitvi 2 oziroma 6 – ker so namreč celoštevilske vse preostale spremenljivke v tej omejitvi, bosta tako posledično celoštevilski tudi ti dve.

Pri 6. omejitvi je v neenačbi vključen tudi max d_j. Namesto tega bi lahko v neenačbo vključili tudi katero koli število N, ki bi zagotovilo, da je vrednost $\delta_j - N \leq 0$ za vsak j. S tem namreč zagotovimo, da je, v kolikor je ali u_j ali y_j enako 1, vrednost 6. omejitve negativna tudi, v kolikor je t_j enako 0. Glede na to da je δ_j lahko največ enaka vrednosti max d_j (to je maksimalna dolžina palice na zalogi), smo to vrednost vključili v neenačbo.

V kolikor imata u_j ali y_j vrednost 1, torej že ta del neenačbe zagotavlja, da bo vrednost manjša od 0. Zato lahko t_j zavzame poljubno vrednost. Ker je v tem primeru s stališča kriterijske funkcije optimalna možnost t_j=0, bo ta spremenljivka torej vedno zavzela to vrednost. Podobno lahko razložimo tudi 4. omejitev.

Pri izbiri, kateri model bomo uporabili za reševanje posameznega problema, je manjša težava, da je pogoj $\sum_{j=1}^{m} d_j \geq (\sum_{i=1}^{n} s_i * b_i)$ potreben, ne pa tudi zadostni pogoj za to, da naročilo lahko izpolnimo v celoti. V mejnih primerih, kjer celotni material le minimalno (recimo za manj kot 0,01%) presega celotne potrebe, je namreč možno, da naročila ne bo možno izpolniti, saj izguba pri optimalni rešitvi presega razliko med razpoložljivim materialom in potrebami. Tega se ne da predvideti vnaprej (ne da bi poiskali optimalno rešitev), zato moramo v takih primerih najprej poizkusiti rešiti model 1. V kolikor je ta

model nerešljiv, nato rešimo še model 2. To je zelo redek primer, saj so izgube (kot je razvidno iz naslednjih poglavij) majhne in le redko presegajo 1% porabljenega materiala. V praksi pri več tisoč rešenih problemih ni prišlo do tega problema niti enkrat, vendar je potrebno na to opozoriti.

Zelo pomembno vprašanje je, kako ustrezno postaviti zgornjo mejo. Seveda se to vprašanje pojavlja le v primeru 1 (torej ko imamo dovolj materiala), medtem ko v primeru 2 (problemi s premalo materiala) zgornja meja sploh ni postavljena, saj moramo tako ali tako porabiti celoten material. Običajno mora biti ta meja postavljena nekje med 0 in max s_i. Z višanjem te meje se nekoliko otežuje tudi problem razreza in pri optimalni rešitvi bo tako verjetno prišlo do nekaj višje izgube. V praksi se običajno uporablja omejitev UB=min s_i (Gradišar, 1997).

Nadaljnje pomembno vprašanje je, kakšno je pravzaprav število palic, ki presegajo zgornjo mejo. V (Gradišar, 1999) je postavljen pogoj, da lahko dolžino zgornje meje presega največ en ostanek, vsi ostali ostanki pa morajo biti krajši.

Model v tem magistrskem delu se v tem delu nekoliko razlikuje, saj z omejitvama 5 in 6 namreč ne zahteva, da je število palic, kjer je ostanek daljši od zgornje meje, večje ali kvečjemu enako 1. Model omejuje le to, da je število takih ostankov daljših od zgornje meje, ki ne štejejo kot izguba, največ 1. Število ostankov daljših od UB je torej lahko tudi večje kot 1, vendar v tem primeru le eden od njih (to je seveda najdaljši) ne šteje kot izguba. Ta razlika je majhna in vpliva le v izjemnih primerih, ko je spodnja meja postavljena na manj kot max s_i

V kolikor je zgornja meja namreč postavljena na max s_i, lahko enostavno dokažemo, da je pri optimalni rešitvi največje število narezanih palic, kjer je ostanek daljši od zgornje meje, enako 1.

Predpostavimo, da je A rešitev problema, pri kateri sta dva ostanka pri narezanih palicah (pri palicah z indeksom k in p) daljša od zgornje meje, eden ne šteje kot izguba (to je daljši ostanek), drugi pa. V tem primeru velja:

$y_k=0$, $u_k=0$, $t_k= \delta_k$

$y_p=0$, $u_p=1$, $t_p=0$

$\delta_k>\max s_i$

$\delta_p>\max s_i$

$\delta_p> \delta_k$

UB<max s_i

skupna izguba= δ_k

Sedaj zagotovo obstaja tak x_{ip} (i=1,….,n); $x_{ip}>0$, da pri rešitvi B velja:

$x_{ipB}=x_{ip}-1$

$x_{ikB}=x_{ik}+1$

$\delta_{pB}= \delta_p+s_i$

$\delta_{kB}= \delta_k-s_i$; $\delta_{kB}>0$

skupna izguba B = skupna izguba A – s_i

Povedano drugače – v tem primeru lahko v načrtu razreza en kos naročila zagotovo prenesemo iz palice z daljšim ostankom na palico s krajšim (to je gotovo, saj vidimo, da je tudi ta ostanek daljši od najdaljše dolžine naročila). Tako dosežemo, da je palica, ki se vrne v skladišče daljša, izguba pa ustrezno manjša.

Iz tega dokaza jasno sledi, da nobena rešitev, pri kateri sta dva ostanka narezanih palic daljša od UB, ne more biti optimalna in je slabša od optimalne za vsaj min s_i. V tem primeru sta torej pri optimalni rešitvi pogoja, da je lahko največ ena palica daljša od UB in da je največ ena palica ne šteje kot izguba, ekvivalentna.

Podobno lahko dokažemo tudi to, da je nemogoče, da bi bila dva ostanka narezanih palic daljša od max s_i ne glede na to, kako je postavljena zgornja meja.

Če je zgornja meja postavljena denimo na min s_i, števila ostankov daljših od zgornje meje ne moremo vnaprej predvideti in se lahko giblje med 0 in številom narezanih palic. V kolikor bi v takem primeru pogoj zaostrili tako, da bi zahtevali, da je največ ena narezana palica daljša od zgornje meje, bi se namreč lahko zgodilo, da pri določenih problemih model ne bi našel rešitve, čeprav je sam model povsem enostavno rešljiv.

To bi se denimo zgodilo v enostavnem primeru z naslednjimi podatki: dolžine palic: 1100, 1200, 1300, 1400, 1500, 1600; rabimo po 2 kosa naročil dolžine 810, 820, 830 in 10. Spodnja meja je torej postavljena na min s_i, kar je v tem primeru 10. Pri optimalni rešitvi je izguba 2370, prav vseh šest ostankov palic je daljših od spodnje meje (očitno je, da za te podatke tak načrt razreza, kjer bi bil vsaj en ostanek krajši od UB, niti ni možen), le najdaljši med njimi pa ne šteje kot izguba. Če bi pogoj zaostrili in zahtevali, da je le en ostanek daljši od UB, bi torej po nepotrebnem prišli do nerešljivega problema.

Iz tega sledi, da je v primerih UB=min s_i opredelitev tega pogoja v našem modelu ustreznejša kot v (Gradišar, 1997), medtem ko v primeru UB=max s_i med obema opredelitvama ni razlik.

Eksaktna rešitev problema

Eksaktna rešitev tega problema je predstavljena tudi v (Trkman, 2002) in (Gradišar, 2001a). Tu najprej predstavljamo metodo za reševanje tega modela.

Omenjeni model namreč predstavlja mešani celoštevilčni problem (angl. mixed integer programming), ki se običajno (in tudi v našem primeru) rešuje s tehniko imenovano razveji & omeji, ki je podrobneje predstavljena v (Winston, 1993; Papadimitriou, 1982 ter tudi v Kozak, 1986, od koder povzemamo predvsem prevode izrazov v slovenščino).

Glede na to da je število možnih rešitev (angl. feasible point) pri večjih problemih zelo veliko in je nemogoče preveriti vse, ta metoda temelji na pametnem izboru točk, ki jih bomo preverili.

Princip tega pristopa k reševanju celoštevilskega linearnega programa je bil prvič predstavljen v (Land, 1960), kasneje pa se je še precej dodelal. Osnovna ideja je, da najprej rešimo linearno (LP) relaksacijo (angl. LP relaxation) problema in pridobimo rešitev, pri kateri vrednosti spremenljivk običajno niso celoštevilčne. V kolikor vrednosti so celoštevilčne, je to tudi optimalna rešitev problema, saj velja, da je optimalna rešitev LP

relaksacije, pri kateri so vse vrednosti celoštevilčne, hkrati tudi optimalna rešitev celoštevilčnega problema (Winston, 1993, str. 502).

V kolikor vrednosti niso celoštevilčne, nam rešitev linearne relaksacije problema predstavlja spodnjo mejo pri minimizacijskem problemu (oz. zgornjo mejo, če gre za maksimizacijski problem) za rešitev – zagotovo vemo, da je končna rešitev slabša[3] (ali v skrajnem primeru kvečjemu enaka) od te vrednosti.

Naslednji korak je, da problem razdelimo na dva dela, tako da ga razvejimo (angl. branching) z dodajanjem nove omejitve. Izberemo eno od spremenljivk[4], ki v prvotni rešitvi nima celoštevilske vrednosti. Denimo, da je to spremenljivka y_1 in da ima v rešitvi LP relaksacije vrednost 7,5674. V enem podproblemu dodamo omejitev $y_1 \geq 8$, v drugem podproblemu $y_1 \leq 7$. S tem smo dobili dva podproblema, ki očitno nimata nobene skupne točke (saj sta dodani omejitvi medsebojno izključujoči). Skupaj podproblema vključujeta vse možne celoštevilske rešitve problema, ne pa tudi rešitve LP relaksacije, ki ni celoštevilska.

Prikaz vseh kreiranih podproblemov imenujemo drevo, vsak posamezen rešen podproblem pa vozel (angl. node).

Sedaj izberemo enega od podproblemov in poiščemo rešitev njegove relaksacije. V splošnem imamo pri rešitvi LP relaksacije podproblema tri možnosti:

1. ta problem je nerešljiv: to pomeni, da ne obstaja nobena rešitev, ki izpolnjuje vse pogoje, vključno z dodanimi omejitvami. Očitno je, da nadaljnje razvejevanje takega vozla nima smisla, saj ne more privesti do nobene rešitve in tako seveda tudi ne do optimalne. Pravimo, da je tak vozel prečrtan (angl. fathomed, killed).
2. v rešitvi podproblema so vse vrednosti celoštevilske: dobili smo možno rešitev (angl. candidate solution) našega problema. V tem trenutku še ne moremo trditi, da je ta rešitev optimalna, nam pa ta vrednost predstavlja zgornjo mejo za rešitev – zagotovo vemo, da je vrednost kriterijske funkcije pri optimalni rešitvi minimizacijskega problema manjša ali kvečjemu enaka, kot je ta zgornja meja. Ko najdemo tak vozel, seveda ne nadaljujemo z njegovim razvejevanjem.

   Seveda velja, da če je tako pridobljena možna rešitev slabša od celoštevilske rešitve, ki smo jo že dobili prej, lahko tudi ta vozel prečrtamo.
3. rešitev problema vsebuje necelоštevilske vrednosti: v tem primeru nadaljujemo z razvejevanjem tega vozla (lahko takoj ali pa prej razvejimo še kakšnega od preostalih nerazvejenih vozlov). V kolikor optimalna vrednost rešitve LP relaksacije ne presega vrednosti do sedaj dosežene možne rešitve, lahko tudi ta vozel prečrtamo, saj bodo vse rešitve, ki bi jih lahko pridobili iz tega vozla z dodajanjem novih omejitev, zagotovo slabše od rešitve LP relaksacije, s tem pa tudi od že pridobljene možne rešitve.

[3] Če gre za minimizacijski problem je vrednost končne rešitve večja, pri maksimizacijskih problemih pa manjša od tako pridobljene vrednosti.

[4] Več o tem na kakšni podlagi izberemo, po kateri spremenljivki bomo razvejili problem, je v nadaljevanju tega magistrskega dela.

Z navedenim postopkom nadaljujemo toliko časa, dokler niso vsi vozli prečrtani. Rešitev, ki ostane takrat, je zagotovo optimalna. Osnovna prednost te ideje je, da lahko s prečrtanjem enega vozla hkrati izločimo večje število možnih točk (torej točk, ki izpolnjujejo omejitve v modelu), ki ne morejo privesti do optimalne rešitve.

Za velike probleme nam tudi uporaba metode razveji & omeji ne omogoča vedno, da najdemo optimalno rešitev. V tem primeru lahko obdržimo do sedaj pridobljeno rešitev (to je torej tisto, ki nam trenutno predstavlja zgornjo mejo) kot najboljšo najdeno rešitev problema. To smo dostikrat uporabili tudi pri izdelavi načrta razreza, ko ta metoda za večje probleme ni našla optimalne rešitve v dani časovni omejitvi.

Tako metoda deluje, če morajo biti vse spremenljivke celoštevilčne. Pri našem primeru nekatere lahko zavzamejo tudi realne vrednosti. Edina razlika je v tem, da sedaj problem razvejujemo le po tistih spremenljivkah, ki morajo nujno biti celoštevilčne, po preostalih pa ne.

Pomembno vprašanje je, po katerih spremenljivkah naj razmejujemo in katere vozle naj razmejimo najprej. Jasnega teoretičnega odgovora na to vprašanje ni, v praksi se uporablja več različnih strategij. Pri izbiri vozlov, ki jih razmejujemo najprej, se dostikrat ravnamo po LIFO (angl. last in first out) načelu. To pomeni, da najprej rešimo tisti podproblem, ki smo ga ustvarili nazadnje. Tako najprej obdelamo en del drevesa, pridobimo možno rešitev, nato pa nadaljujemo z reševanjem preostalega drevesa (angl. backtracking).

Druga možnost je, da vedno rešimo tisti podproblem, ki ima najvišjo vrednost kriterijske funkcije. To pomeni, da dostikrat skačemo z enega na drug dela drevesa (angl. jumptracking).

V našem primeru smo uporabljali naslednje nastavitve: za izbor spremenljivke, po kateri razmejujemo, smo pustili izbrano nastavitev v programu, ki omogoča programu določitev pravila za izbor spremenljivke na podlagi problema in napredka pri reševanju problema. Ravno tako je bil programu prepuščen izbor, katero od obeh vej, ki izhajajo iz posameznega vozla, bo obdelal najprej. Za razmejevanje je bil uporabljen tisti vozel, ki je imel najboljšo vrednost kriterijske funkcije pri rešitvi LP relaksacije.

Očitno je torej, da z opisano metodo lahko pridemo do optimalnih rešitev posameznega problema. Zaradi opisane NP-polnosti tega problema in s tem povezanega hitrega naraščanja časa, potrebnega za pridobitev rešitve pri večanju obsega problema, je pomembno vprašanje, koliko je metoda uporabna v praksi in kakšne so približne velikosti problemov, pri kateri jo lahko še uporabimo. Zato smo metodo obsežno eksperimentalno testirali.

Ko govorimo o rešitvah, ki smo jih pridobili z eksaktno metodo, imamo v mislih rešitve, najdene v dani časovni omejitvi. Ta rešitev je lahko optimalna (metoda razveji&omeji je torej našla optimalno rešitev v krajšem času, kot je bila postavljena časovna omejitev) ali neka do tedaj najboljša najdena rešitev, ki ni nujno optimalna.

Z eksaktno metodo smo najprej rešili problem že naveden v literaturi (Gradišar, 1999)[5]. Navedena rešitev ima izgubo 2 centimetra, medtem ko ima rešitev predstavljena v sliki 1

[5] Celotne potrebe presegajo celotni material, ki je na razpolago v zalogi, zato rešujemo model 2

za polovico manjšo izgubo (1 cm). Rešitev je bila najdena v 10,1 sekundah, potem ko je razveji & omeji metoda preiskala 59467 vozlov.

PODATKI O NAROČILIH

zap. št. naročila	dolžina	št. kosov
1	304	12
2	319	38
3	397	14
4	415	34
5	366	27

PODATKI O PALICAH V ZALOGI

zap. št. palice	dolžina
1	13535
2	11301
3	9408
4	9341

REZULTATI RAZPOREJENI PO ZAPOREDNI ŠTEVILKI NAROČIL

zap. št. naročila	zap. št. palice	št. kosov
1	1	1
1	2	1
1	3	0
1	4	10
2	1	29
2	2	4
2	3	1
2	4	2
3	1	4
3	2	1
3	3	9
3	4	0
4	1	4
4	2	11
4	3	8
4	4	11
5	1	2
5	2	13
5	3	6
5	4	3

IZGUBA

zap. št. palice	izguba
1	0
2	1
3	0
4	0

REALIZACIJA

zap. št. naročila	načrt	realizacija	razlika
1	12	12	0
2	38	36	2
3	13	13	0
4	34	34	0
5	25	22	3

Slika 1: Izboljšana rešitev problema iz (Gradišar, 1999).

Pri problemu predstavljenem v sliki 1 gre seveda za razmeroma majhen primer s štirimi dolžinami palic in petimi dolžinami naročil. Ob povečevanju števila spremenljivk zaradi NP polnosti problema kompleksnost hitro narašča, s tem pa tudi čas potreben za rešitev. Dostikrat zato v normalnem času (npr. nekaj minutah) ni možno najti optimalne rešitve. K sreči razveji&omeji, kot smo navedli pri opisu te metode, deluje tako, da se postopno bliža optimumu, medtem pa že privede do rešitve, ki je lahko blizu optimalni.

Preden se lotimo testiranja metode, potrebujemo zadostno število podatkov, ki jih lahko uporabimo kot vhodne podatke za metodo. Eden od problemov, ki se pojavljajo v literaturi pri medsebojni primerjavi različnih algoritmov in metod je prav pomanjkanje dovoljšnega števila testnih problemov, saj so tako rezultati predstavljeni le na nekaj posebej izbranih primerih (primer tega lahko vidimo denimo v (Pierce, 1964), (Johnston, 1986), (Stadtler, 1990)). Zato je tudi za učinkovito testiranje metod v tem magistrskem delu nujno uporabiti generator naključnih problemov, ki na podlagi podanih parametrov zgenerira ustrezno število problemov.

V ta namen smo uporabili generator PGEN (Gradišar, 1999b; Gradišar, 2002), ki je razširjena in izboljšana verzija generatorja CUTGEN1 (Gau, 1995).

Pri generiranju podatkov z CUTGEN1 lahko nastavimo vrednosti naslednjih parametrov, na podlagi katerih se nato zgenerirajo naključni podatki:

m – velikost problema (število različnih naročil),

L – standardna dolžina palice,

v_1 – spodnja meja za relativno dolžino naročil v primerjavi s standardno dolžino palice L : $l_i \geq v_1 * L$ (i=1,….,m),

v_2 – zgornja meja za relativno dolžino naročil v primerjavi s standardno dolžino palice L : $l_i \leq v_2 * L$ (i=1,….,m),

d – povprečno naročilo posamezne dolžine palice.

Za vsak primer s parametri (m, L, v_1, v_2, d) moramo (ob dani standardni dolžini palice, torej L) sedaj naključno generirati 2m spremenljivk (m spremenljivk za dolžine naročil in m spremenljivk za število kosov posameznega naročila).

Dolžina naročil se v tem generatorju dobi tako, da najprej izračunamo vrednost y_i, nato pa to vrednost zaokrožimo navzdol na največje celo število. Za generiranje naključnih števil na intervalu med 0 in 1 ($rand_i(0,1)$) se uporablja Lehmerjev generator naključnih števil (predstavljen v (Hutchinson, 1966) in (Park, 1988)). Torej:

$y_i = (v_1+(v_2-v_1) * rand_i(0,1))*L + rand_i(0,1)$

Vrednost y_i nato zaokrožimo navzdol na najbližje celo število:

$l_i = [y_i]$

Ob takem generiranju vrednosti je sicer možno, da je več dolžin naročil medsebojno enakih in da torej dejansko nismo generirali m različnih naročil ampak kakšno manj, kar ni bistveno.

Za generiranje potreb po posameznem naročilu se celotno naročilo D (D=m*d) razdeli po posameznih naročilih po naslednjem postopku:

$d_i = \max \{1, |đ_i + 0,5|\}$

$$đ_i = \frac{rand_i(0,1)}{\sum_{k=1}^{m} rand_k(0,1)} * D \qquad i= 1,\ldots,m-1$$

$$d_m = \max\{1, D - \sum_{k=1}^{m-1} d_i\}$$

Povpraševanja po dolžini d_i (i=1,…,m-1) so popravljena s prištevanjem 0,5, saj bi bila sicer ta povpraševanja sistematično podcenjena, s tem pa bi bila sistematično precenjena povpraševanja po d_m. Načeloma je možno, da je celotno povpraševanje $\sum_{j=1}^{m} d_i$ nekoliko večje od želenega D_i. Vendar to ni večji problem, saj ima ta razlika zanemarljivo majhen vpliv na povprečno povpraševanje d.

Opisani generator je primeren za generiranje testnih podatkov za standardni problem razreza. Za naš primer pa v taki obliki ni uporaben, saj ne generira različnih dolžin palic v zalogi. Dejstvo, da so vse palice v zalogi različnih dolžin, je ena od glavnih značilnosti splošnega enodimenzionalnega problema razreza.

Tako za generiranje podatkov v nadaljevanju uporaba tega generatorja ne bi bila možna. Zato uporabljamo razširjen in izboljšan generator PGEN (Gradišar, 1999b; Gradišar, 2002). PGEN uvede 5 novih parametrov, tako da ima naslednje parametre:

n - število različnih dolžin naročil

v_1, v_2 - zgornja in spodnja meja za dolžino naročil – vsa naročila so torej dolga med v_1 in v_2.

d - povprečno število kosov posamezne dolžine naročila

p - število različnih standardnih dolžin palic v zalogi

s_1, s_2 - zgornja in spodnja meja za dolžino standardnih palic v zalogi, torej $s_1 \leq L_k \leq s_2$ ($k = 1,..., p$)

m - število nestandardnih dolžin palic v zalogi

u_1, u_2 - zgornja in spodnja meja za dolžino nestandardnih palic v zalogi, torej $u_1 \leq U_j \leq u_2$ ($j = 1,..., m$).

r - število generiranih primerov problema (parameter r torej pomeni, koliko problemov želimo generirati z istimi vrednostmi zgornjih parametrov – tako generirani problemi seveda niso medsebojno enaki)

Zaradi uvedbe teh parametrov je PGEN primeren tudi za probleme z različnimi dolžinami palic v skladišču, kakršen je tudi primer v tem magistrskem delu. Seveda se lahko ta generator uporablja tudi za generiranje podatkov za testiranje metod za reševanje standardnega problema razreza.

Z generatorjem PGEN je mogoče ob uporabi istega semena in istih vhodnih parametrih ponovno zgenerirati identične podatke, te podatke nato uporabiti za reševanje in na koncu primerjati rešitve, pridobljene z različnimi metodami. Vhodni podatki se zgenerirajo na podlagi parametrov kot naključni vzorec enega ali več testnih problemov. Uporaba tega generatorja za medsebojno primerjavo dveh hevrističnih metod je prikazana v (Gradišar, 1999b; Gradišar, 2002), v našem magistrskem delu pa je bil uporabljen za pridobitev zadostnega števila podatkov za ustrezno testiranje eksaktne metode.

V našem primeru smo uporabili naslednje vhodne podatke:

- določanje števila različnih dolžin naročila in povprečne potrebe po posamezni dolžini ter zgornjih in spodnjih mej za naročila:

  število različnih dolžin naročila n zavzame vrednosti 5, 10, 15, v_1 in v_2 zavzemajo naslednje vrednosti (v_1 = 100 in v_2 = 200; v_1 = 200 in v_2 = 400; v_1 = 300 in v_2 = 600) medtem ko povprečna potrebno število kosov posamezne dolžine zavzema vrednosti (d = 5, 10, 15). S kombinacijo teh treh možnosti smo pridobili 27 problemov z različnimi parametri

- določitev standardnih dolžin palic:

  generator PGEN omogoča tudi, da generiramo nekaj standardnih dolžin palic. Ker se v tem magistrskem delu ukvarjamo s splošnim problemom enodimenzionalnega razreza, kjer ni standardnih dolžin palic, je ta parameter postavljen na 0.

- določitev nestandardnih dolžin palic:

  Število nestandardnih palic (m) se spreminja od 5 do 15 v koraku po 5, zgornja meja za dolžino palic se spreminja od 2000 do 6000, spodnja meja pa od 1000 do 3000.

  Podroben prikaz generiranja testnih problemov in način, kako se je določilo seme, sta razvidna iz sheme, ki je prikazana v sliki 2.

Procedura PROGEN:

od $i = 1$ ***do*** 3

od $j = 1$ ***do*** 3

od $k = 1$ ***do*** 3

$n \leftarrow i \cdot 5$

$v_1 \leftarrow j \cdot 100$

$v_2 \leftarrow j \cdot 200$

$d \leftarrow k \cdot 5$

$m \leftarrow k \cdot 5$

$u_1 \leftarrow k \cdot 1000$

$u_2 \leftarrow k \cdot 2000$

$c \leftarrow int(\frac{m}{10}) \cdot 9 + 1$

$seme \leftarrow m + 10 \cdot c \cdot d + 10 \cdot c^2 \cdot v_2 + 1000 \cdot c^2 \cdot v_1 + 1000000 \cdot n$

$r \leftarrow 10$

Poženi *PGEN (n, v_1, v_2, p, s_1, s_2, d, m, u_1, u_2, seme, r)*

naslednji k

naslednji j

naslednji i

Slika 2: Prikaz sheme procedure PROGEN uporabljene za generiranje testnih problemov.

V praksi smo PROGEN proceduro implementirali z uporabo programskega jezika Visual Basic. Generirani parametri s to proceduro so prikazani v tabeli 2.

Za vsak testni primer smo s proceduro PROGEN generirali 10 testnih primerov (r=10). Skupaj je torej za testiranje na razpolago 270 primerov, od tega jih je 150 takih, kjer je dovolj materiala, 120 pa takih, kjer materiala primanjkuje. Še enkrat naj poudarimo, da prikaz parametrov v tabeli 2 (vključno s semenom) omogoča, da vsakdo z uporabo procedure PROGEN pride do istih podatkov in jih nato uporabi za reševanje problemov z morebitnimi novimi metodami ter nato primerja svoje rezultate z rezultati, prikazanimi v tem magistrskem delu.

Poleg samega testiranja metode in njene primerjave s hevrističnimi rešitvami je bil eden od ciljev tudi ugotoviti, kako spreminjanje časovne omejitve vpliva na izboljševanje pridobljene rešitve. Zato se časovna omejitev povečuje od 2 preko 10 do 60 sekund.

zap. št.	n	v_1	v_2	d	m	u_1	u_2	Seme
1	5	100	200	5	5	1000	2000	5102055
2	5	100	200	10	10	2000	4000	510201010
3	5	100	200	15	15	3000	6000	510201515
4	5	200	400	5	5	1000	2000	5204055
5	5	200	400	10	10	2000	4000	520401010
6	5	200	400	15	15	3000	6000	520401515
7	5	300	600	5	5	1000	2000	5306055
8	5	300	600	10	10	2000	4000	530601010
9	5	300	600	15	15	3000	6000	530601515
10	10	100	200	5	5	1000	2000	10102055
11	10	100	200	10	10	2000	4000	1010201010
12	10	100	200	15	15	3000	6000	1010201515
13	10	200	400	5	5	1000	2000	10204055
14	10	200	400	10	10	2000	4000	1020401010
15	10	200	400	15	15	3000	6000	1020401515
16	10	300	600	5	5	1000	2000	10306055
17	10	300	600	10	10	2000	4000	1030601010
18	10	300	600	15	15	3000	6000	1030601515
19	15	100	200	5	5	1000	2000	15102055
20	15	100	200	10	10	2000	4000	1510201010
21	15	100	200	15	15	3000	6000	1510201515
22	15	200	400	5	5	1000	2000	15204055
23	15	200	400	10	10	2000	4000	1520401010
24	15	200	400	15	15	3000	6000	1520401515
25	15	300	600	5	5	1000	2000	15306055
26	15	300	600	10	10	2000	4000	1530601010
27	15	300	600	15	15	3000	6000	1530601515

Tabela 2: Parametri za 27 testnih primerov.

Način testiranja je prikazan z diagramom poteka v sliki 3. V diagramu je uporabljena naslednja notacija:

p= število testnih primerov (p=1….27),

k= število podprimerov, generiranih za vsak testni primer (k=1…10),

f= število različnih časovnih omejitev v eksperimentu (f=1…6).

Sumirani rezultati za vseh 27 primerov so predstavljeni v tabeli 3 (ob upoštevanju UB= min s_i) in tabeli 4 (ob upoštevanju UB= max s_i). V tabeli je v vsaki vrstici združenih 10 podprimerov, ki so bili generirani z istimi parametri. Za vsako časovno omejitev je navedena celotna izguba (vsota izgub v vseh desetih podproblemih), izguba v odstotkih in število optimalno rešenih podprimerov.

Pri postavitvi UB na min s_i je bila v 2 sekundah najdena optimalna rešitev za 57 podprimerov, v 60 sekundah pa za 110 podprimerov. Izguba po 60 sekundah niha od 0 do

2,4%, pri čemer je potrebno poudariti, da je izguba 2,4% v 7. primeru optimalna rešitev (torej tudi z uporabo kake druge metode ne bi mogli najti boljše rešitve). V 7. primeru je namreč število palic, število naročil in povprečno naročilo majhno (vse te tri spremenljivke imajo vrednost 5), predvsem pa je nizko razmerje med povprečno dolžino palice in povprečno dolžino naročila. To pomeni, da imamo razmeroma majhno število možnih kombinacij. Posledica tega je, da je optimalno rešitev razmeroma lahko najti, izguba pri optimalni rešitvi pa je precej velika.

Podobno velja tudi za primer 4. V vseh ostalih primerih je izguba 1% ali manj, kar pomeni, da je eksaktna metoda za ta velikostni razred problema primerna. Pri povečevanju časovne omejitve za reševanje lahko vidimo, da največje zmanjšanje izgube prinese povečanje časa iz 2 na 10 sekund (zmanjšanje izgube za 35%), v nekoliko manjši meri pa tudi povečanje iz 10 na 20 sekund (izguba se zmanjša za slabih 20%). Nadaljnja povečanja razpoložljivega časa sicer privedejo do delnega zmanjšanja izgube, vendar razlike niso več tako očitne (povečanje časovne omejitve za trikrat iz 20 na 60 sekund tako izgubo zmanjša le še za manj kot 15%).

V tabeli 4, kjer je UB=max s_i, so rezultati prikazani samo za primere, kjer je dovolj materiala, saj v primerih s premalo materiala UB ni vključena v model. Zaradi strožje omejitve glede zgornje meje palice, ki ne šteje kot izguba, so izgube razumljivo nekoliko večje, sicer pa veljajo podobne ugotovitve kot za primere z nižje postavljeno zgornjo mejo.

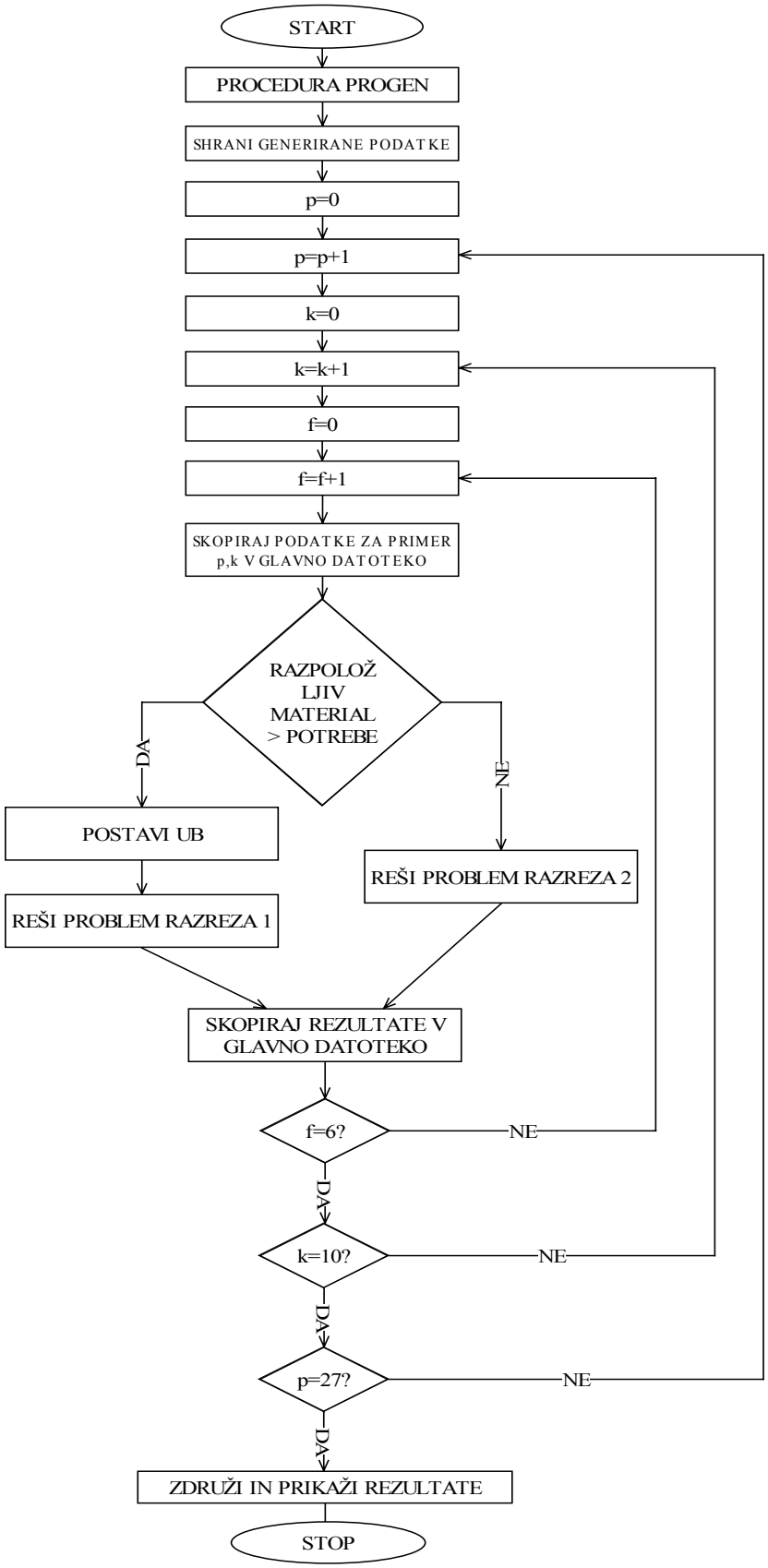
Izguba pri manjših primerih (to so denimo primeri z zaporedno številko 1, 2, 3) je tudi zanemarljivo majhna, za vse te primere najkasneje v 10 sekundah najdemo optimalno rešitev. Pri večjih problemih izguba hitro narašča in tudi povečevanje časovne omejitve (denimo v primeru 24) dostikrat ne privede do bistvene izboljšave rešitve. To je v skladu z ugotovitvami, navedenimi v 1. poglavju tega magistrskega dela.

	2 sekundi			10 sekund			20 sekund			30 sekund			45 sekund			60 sekund		
zap. št.	izguba		opt	izguba		opt	izguba		opt	izguba		opt	izguba		opt	izguba		opt
	cm	%		cm	%		cm	%		cm	%		cm	%		cm	%	
1	1	0,0027%	9	1	0,0027%	10	1	0,0027%	10	1	0,0027%	10	1	0,0027%	10	1	0,0027%	10
2	0	0,0000%	10	0	0,0000%	10	0	0,0000%	10	0	0,0000%	10	0	0,0000%	10	0	0,0000%	10
3	3	0,0026%	8	1	0,0009%	9	0	0,0000%	10	0	0,0000%	10	0	0,0000%	10	0	0,0000%	10
4	1301	1,6761%	5	1028	1,3445%	9	1028	1,3445%	9	1028	1,3445%	9	1028	1,3445%	9	1028	1,3445%	9
5	831	0,4978%	0	351	0,2119%	1	295	0,1759%	2	255	0,1495%	2	228	0,1327%	2	212	0,1229%	2
6	852	0,3552%	0	295	0,1240%	0	83	0,0364%	1	69	0,0301%	1	62	0,0269%	2	62	0,0269%	2
7	1702	2,3912%	9	1702	2,3912%	10	1702	2,3912%	10	1702	2,3912%	10	1702	2,3912%	10	1702	2,3912%	10
8	2884	1,3326%	0	2325	1,0791%	0	1755	0,8112%	0	1541	0,7116%	0	1519	0,6996%	0	1457	0,6706%	0
9	4103	1,1964%	0	3045	0,8837%	0	2555	0,7473%	0	2201	0,6467%	0	1911	0,5595%	0	1780	0,5196%	0
10	157	0,2295%	1	53	0,0730%	2	27	0,0383%	3	27	0,0383%	3	22	0,0305%	4	18	0,0246%	5
11	96	0,0671%	2	25	0,0164%	4	10	0,0065%	6	10	0,0065%	6	8	0,0052%	6	8	0,0052%	6
12	269	0,1189%	0	83	0,0372%	7	64	0,0285%	7	39	0,0175%	8	27	0,0122%	8	23	0,0103%	8
13	99	0,1245%	2	41	0,0515%	4	33	0,0419%	4	17	0,0219%	5	9	0,0113%	6	8	0,0100%	6
14	3022	1,0156%	0	2048	0,6839%	0	1703	0,5687%	0	1666	0,5572%	0	1666	0,5572%	0	1613	0,5393%	0
15	6652	1,4417%	0	2753	0,5913%	0	2565	0,5483%	0	2502	0,5334%	0	2324	0,4923%	0	1833	0,3866%	0
16	266	0,3686%	1	114	0,1569%	5	49	0,0681%	8	49	0,0681%	9	49	0,0681%	9	49	0,0681%	9
17	4011	1,3846%	0	3013	1,0466%	0	2630	0,9136%	0	2580	0,8953%	0	2272	0,7893%	0	2074	0,7236%	0
18	9599	1,6617%	0	7936	1,2251%	0	7041	1,0860%	0	6533	1,0099%	0	6533	1,0099%	0	6533	1,0099%	0
19	29	0,0402%	5	10	0,0142%	9	5	0,0071%	9	3	0,0043%	9	0	0,0000%	10	0	0,0000%	10
20	1179	0,5150%	0	756	0,3203%	0	638	0,2707%	0	557	0,2361%	0	434	0,1827%	1	429	0,1803%	1
21	7753	2,1027%	0	1143	0,3401%	0	600	0,1691%	0	568	0,1599%	0	444	0,1235%	1	438	0,1217%	1
22	141	0,1831%	4	37	0,0477%	5	21	0,0275%	7	4	0,0051%	8	1	0,0013%	9	1	0,0013%	9
23	2580	0,8612%	0	1941	0,6456%	0	1551	0,5126%	0	1357	0,4517%	0	1071	0,3559%	0	1043	0,3468%	0
24	16250	3,0632%	0	13977	2,3567%	0	7639	1,1572%	0	7101	1,0736%	0	6856	1,0387%	0	5615	0,8531%	0
25	427	0,5728%	1	113	0,1457%	2	93	0,1197%	2	93	0,1197%	2	48	0,0636%	2	41	0,0549%	3
26	2037	0,6872%	0	1319	0,4446%	0	1054	0,3547%	0	1054	0,3547%	0	993	0,3325%	0	984	0,3294%	0
27	11150	1,6180%	0	7518	1,0954%	0	5984	0,8730%	0	5984	0,8730%	0	5859	0,8553%	0	5316	0,7742%	0
SKUPAJ	77394	0,8707%	57	51628	0,5678%	87	39126	0,456%	98	36941	0,4334%	102	35067	0,4106%	109	32268	0,3895%	110

Tabela 3: Rezultati za UB= min s_i ob različnih časovnih omejitvah.

	2 sekundi			10 sekund			20 sekund			30 sekund			45 sekund			60 sekund		
	izguba			izguba			izguba			izguba			izguba			izguba		
Zap. št.	cm	%	opt			opt	cm	%	opt	cm	%	opt	cm	%	opt	cm	%	opt
1	1	0,0027%	10	1	0,0027%	10	1	0,0027%	10	1	0,0027%	10	1	0,0027%	10	1	0,0027%	10
2	2	0,0026%	9	0	0,0000%	10	0	0,0000%	10	0	0,0000%	10	0	0,0000%	10	0	0,0000%	10
3	1	0,0008%	9	0	0,0000%	10	0	0,0000%	10	0	0,0000%	10	0	0,0000%	10	0	0,0000%	10
4	1842	2,4094%	6	1799	2,3524%	8	1799	2,3524%	8	1799	2,3524%	8	1799	2,3524%	9	1799	2,3524%	9
5	1299	0,7972%	0	580	0,3257%	0	531	0,2971%	1	508	0,2833%	1	442	0,2466%	1	313	0,1754%	1
6	720	0,3041%	0	306	0,1248%	0	305	0,1242%	0	294	0,1198%	1	197	0,0808%	1	162	0,0663%	1
8	3594	1,6466%	0	2112	0,9747%	0	1757	0,8068%	0	1501	0,7016%	0	1427	0,6638%	0	1369	0,6364%	0
9	5916	1,6859%	0	4596	1,3164%	0	3629	1,0453%	0	3601	1,0354%	0	3353	0,9612%	0	2601	0,7537%	0
10	106	0,1522%	2	53	0,0748%	2	24	0,0355%	5	24	0,0355%	5	19	0,0277%	6	17	0,0248%	6
11	173	0,1208%	1	65	0,0461%	4	47	0,0333%	5	47	0,0333%	5	18	0,0127%	6	17	0,0120%	6
12	253	0,1095%	2	72	0,0306%	5	68	0,0288%	6	51	0,0213%	7	30	0,0124%	8	0	0,0000%	10
14	2905	0,9699%	0	2175	0,7247%	0	1929	0,6437%	0	1892	0,6321%	0	1878	0,6276%	0	1722	0,5763%	0
15	13879	2,9446%	0	6611	1,4494%	0	5709	1,2615%	0	4543	1,0074%	0	4528	1,0043%	0	2869	0,6296%	0
18	11526	2,0006%	0	7308	1,2592%	0	6963	1,2005%	0	6561	1,1339%	0	6484	1,1194%	0	6444	1,1129%	0
20	7933	2,8766%	0	7282	2,5875%	0	7205	2,5529%	0	642	0,2803%	0	361	0,1551%	1	337	0,1454%	1
21	5344	1,5468%	0	4708	1,3642%	0	4496	1,3068%	1	4469	1,2991%	1	889	0,2529%	1	867	0,2457%	2
24	9685	2,3776%	0	4210	0,9170%	0	4200	0,9149%	0	4101	0,8926%	0	9801	1,8811%	0	9619	1,8464%	0

Tabela 4: Rezultati za UB= max s_i ob različnih časovnih omejitvah.



Slika 3: Diagram poteka za eksperiment.

Hevristične rešitve

Obstoječe hevristične rešitve

Poleg predstavljene eksaktne rešitve obstajata tudi dve hevristični metodi za rešitev istega problema in sicer COLA (Gradišar, 1996; Gradišar, 1997), ki je bila v osnovi razvita za optimizacijo razreza v tekstilni industriji, vendar se lahko uporablja tudi za razrez v drugih panogah, in CUT (Gradišar, 1999). Glede na to da je CUT razširjena in izboljšana verzija COLA-e in da je pokazano, da je ta metoda boljša (Gradišar, 2002), to metodo predstavljamo nekoliko obširneje. COLA deluje po zelo podobnem principu, vendar je nekoliko enostavnejša, zato tudi nekoliko hitrejša, vendar običajno privede do načrta razreza z večjo izgubo.

Pri CUT gre za tako imenovano zaporedno hevristično proceduro (angl. Sequential Heuristic Procedure – SHP), ki deluje tako, da minimizira vpliv končnih pogojev (angl. ending conditions). Ti so namreč pri hevurističnih procedurah dostikrat problem – prav zadnje faze razreza lahko namreč prispevajo večji del k celotni izgubi.

Sama definicija problema je enaka kot pri eksaktni metodi z izjemo treh manjših razlik. Prva je ta, da je pri formulaciji problema za hevristično metodo (kadar je dovolj materiala), postavljen pogoj, da je največ ena palica daljša od zgornje meje, pri eksaktni pa, da največ ena palica ne šteje kot izguba. V magistrskem delu smo že dokazali, da sta formulaciji za obe metodi večinoma enakovredni, razen v nekaterih skrajnih primerih, ko je formulacija za eksaktno metodo ustreznejša, saj drugačna formulacija ne privede do rešitve.

Druga razlika je v primeru, da imamo premalo materiala. Obe hevristični metodi v skladu z modelom minimizirata $\sum_{i=1}^{n} \Delta_i * s_i$ (torej vsoto nenarezanih naročil), medtem ko eksaktna rešitev, predstavljena v tem magistrskem delu minimizira $\sum_{j=1}^{m} \delta_j$ (vsoto ostankov po posameznih palicah). Poleg tega CUT dejansko minimizira $\sum_{i=1}^{n} \Delta_i * s_i$ preko minimiziranja $\sum_{j=1}^{m} t_j$, zato moramo v teh primerih UB nujno postaviti na max s_i, saj le tako dosežemo pravilno delovanje algoritma (Gradišar, 1999, str. 560). Čeprav sta kriterijski funkciji formulirani različno, sta dejansko enakovredni, saj velja:

$$\sum_{i=1}^{n} (\Delta_i * s_i) + \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} * s_j = \sum_{i=1}^{n} s_i * b_i \qquad (1)$$

$$\sum_{j=1}^{m} \delta_j + \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} * s_j = \sum_{j=1}^{m} d_j \qquad (2)$$

Prva enačba pomeni, da je vsota dolžin nenarezanih in narezanih kosov enaka vsoti vseh kosov (to je očitno, saj je vsak kos lahko ali narezan ali nenarezan). Druga enačba pomeni, da je vsota dolžin vseh ostankov in narezanih kosov enaka vsoti dolžin vseh palic (tudi ta enačba v primerih s premalo materiala velja, saj smo v načrtu razreza res uporabili vse palice).

Če sedaj enačbo (1) odštejemo od enačbe (2), dobimo:

$$\sum_{j=1}^{m} \delta_j - \sum_{i=1}^{n} \Delta_i * s_i = \sum_{j=1}^{m} d_j - \sum_{i=1}^{n} s_i * b_i$$

Ker je desna stran te enačbe konstanta, smo torej dokazali, da je minimiziranje $\sum_{j=1}^{m} \delta_j$ (kar minimizira kriterijska funkcija pri eksaktni metodi) ekvivalentno minimiziranju $\Delta_i * s_i$ (to minimizira kriterijska funkcija pri hevristični metodi).

Tretja razlika je, da hevristični metodi omogočata še vključitev parametra Y, ki določa največje število različnih dolžin naročil, ki jih lahko izrežemo iz ene palice. V modelu za eksaktno rešitev problema tega parametra nismo vključili.

Hevristična metoda temelji na postopnem izgrajevanju načrta razreza in na zaporednem obravnavanju palic. Na začetku vse dolžine palic pripadajo skupini nenarezanih palic, skupina narezanih palic je prazna. Nato se v vsakem koraku skupina nenarezanih palic zmanjša za 1, skupina narezanih palic pa poveča za 1 (v vsakem koraku torej poiščemo načrt razreza za eno palico). Ustrezno popravimo tudi število potrebnih naročil posameznih dolžin. Algoritem je sestavljen iz treh korakov:

1. izberi dolžine naročil,
2. izberi palico in jo razreži na izbrana naročila,
3. ponavljaj ta dva koraka, dokler niso narezane vse palice ali izpolnjena vsa naročila.

Pri izbiri naročil in palic se ravnamo po predpostavkah (te predpostavke so se v praksi pokazale za pravilne), da je lažje najti dobro rešitev, če:

1. lahko izbiramo iz čim večjega števila možnih rešitev. Število možnih rešitev je večje, če:
    a. je na razpolago čim večje število različnih dolžin naročil, ki še niso bila izpolnjena v celoti,
    b. razmerje med povprečno dolžino palice in povprečno dolžino naročila je čim večje,
    c. število nenarezanih dolžin naročil je čim večje,
2. razlike med posameznimi možnimi rešitvami so čim večje. To velja, če:
    a. je razmerje med najdaljšo in najkrajšo dolžino naročila čim večje,
    b. je razmerje med najdaljšo in najkrajšo dolžino palice na zalogi čim večje.

Omenjene predpostavke so statistično dokazane (Gradišar, 1999), vendar ni nujno, da veljajo v vsakem primeru. Z upoštevanjem predpostavk dosežemo, da lahko tudi v zadnjih korakih algoritma še vedno izbiramo med razmeroma velikim številom možnih rešitev, kar privede do manjše izgube materiala. To je predvsem pomembno v primerih s premalo materiala, saj imamo v primerih, kjer je materiala dovolj v zadnjih korakih nekaj več svobode, saj nam tudi na koncu še ostane nekaj nenarezanih palic. Na ta način torej zmanjšamo vpliv t. i. končnih pogojev, ki so pogosto problem pri tovrstnih hevrističnih metodah.

Najpomembnejše vprašanje je način izbora dolžin naročil. Načeloma lahko uporabimo 2 proceduri:

1. v vsakem koraku najprej izberemo tista naročila, ki imajo največje število še nenarezanih kosov naročil – tako izberemo prvih Y naročil, urejenih po padajočem številu nenarezanih kosov (Δ_i). Na ta način bo na koncu ostalo več različnih naročil, vendar bodo ostala na koncu tudi najdaljša naročila, kar je v nasprotju s predpostavko 1.2. in 1.3.,
2. v vsakem koraku najprej izberemo najdaljša naročila – izberemo torej prvih Y naročil, urejenih po padajoči dolžini (s_i).

Prvi pristop je primeren predvsem pri majhnih razlikah med najdaljšo in najkrajšo dolžino naročila, drugi pa pri velikih razlikah.

V praksi zato kombiniramo oba pristopa in del naročil izberemo po prvem, del naročil pa po drugem načinu. Uporabljamo naslednje kombinacije:

- f naročil izberemo po proceduri 1, Y-f po proceduri 2 (f=0,…,Y),
- f naročil izberemo po proceduri 2, Y-f po proceduri 1 (f=1,…Y-1),
- naročila lahko izbiramo tudi po padajoči vrednosti produkta $s_i*\Delta_i$.

Vsega skupaj torej imamo 2*Y+1 možnih kombinacij za izbor[6]. Hevristična procedura CUT preizkusi vse kombinacije in izbere tisto, ki privede do najboljše končne rešitve.

Pri izboru palice, ki bo razrezana v vsakem koraku, hevristična procedura deluje tako, da izbere tisto, ki ima najmanjšo izgubo. V primeru, da ima več palic enako izgubo, procedura (v skladu s predpostavko 1.2.) izbere najkrajšo palico s tako izgubo. To dosežemo tako, da najprej poskušamo razrezati kratke palice (palice torej uredimo naraščajoče po dolžini). Ker je verjetno, da bodo boljše rešitve dosežene pri daljših palicah, bi to privedlo do tega, da bi za kasnejše faze preostajale predvsem krajše palice. Za zmanjšanje tega vpliva se lahko določi parameter r – to je tista izguba, ki se pri posamezni palici lahko zanemari. Vse rešitve za posamezno palico z izgubo manjšo od r se torej štejejo kot enakovredne (večji kot je r, pogosteje bodo kratke palice uporabljene na začetku – posledično bo izguba večja na začetku, zato pa manjša na koncu).

Omenjeni algoritem je napisan v programskem jeziku FORTRAN in vsebuje 2500 vrstic kode.

V omenjenih člankih (Gradišar, 1999) je tudi statistična analiza faktorjev, ki vplivajo na kvaliteto rešitve. Ugotovljeno je, da je rešitev tem boljša, čim večje so razlike med posameznimi dolžinami naročil in čim večje je povprečno število kosov izrezanih iz ene palice na zalogi.

Pri tem je potrebno posebej poudariti, da se v teh člankih kot dobra rešitev šteje tista, ki ima majhno odstotno izgubo materiala. V tem magistrskem delu pa kot dobre rešitve problema vzamemo tiste, ki so čim bližje optimalni rešitvi (tako denimo je po tem kriteriju

[6] To je tudi ena od poglavitnih prednosti algoritma CUT pred COLA. COLA namreč preizkusi samo 3 kombinacije za sortiranje in izbor naročil, in sicer sortiranje po preostalih potrebah, sortiranje po dolžini in sortiranje po produktu dolžine in preostalih potreb, ne preizkusi pa tudi vmesnih kombinacij (torej izbor dela naročil po preostalih potrebah, dela naročil pa po dolžini).

primer 4 iz tabele 3 dobro rešen, čeprav ima daleč največjo izgubo od vseh primerov, ker je kar devet od desetih podproblemov tega problema rešeno optimalno).

Primerjava med rezultati pridobljenimi z eksaktno metodo (ob upoštevanju UB=min s_i in časovne omejitve 1 minuta) in metodo CUT so predstavljeni v tabeli 5. Tako kot v prejšnjih tabelah je v eni vrstici združenih 10 primerov z istimi parametri. V drugi vrstici je označeno, ali gre za primere z dovolj materiala (model 1, oznaka D) ali za primere s premalo materiala (model 2, oznaka N). Oznaka D/N pomeni, da je v nekaterih podprimerih dovolj materiala, v drugih pa ne.

	Dovolj materiala?	izguba CUT		izguba eksaktna metoda	
zap. št.		cm	%	cm	%
1	D	8	0,0213%	1	0,0027%
2	D	0	0,0000%	0	0,0000%
3	D	0	0,0000%	0	0,0000%
4	D/N	1182	1,5460%	1028	1,3445%
5	D	28	0,0162%	212	0,1229%
6	D	9	0,0039%	62	0,0269%
7	N	1940	2,7256%	1702	2,3912%
8	D	213	0,0980%	1457	0,6706%
9	D	285	0,0832%	1780	0,5196%
10	D/N	59	0,0807%	18	0,0246%
11	D	0	0,0000%	8	0,0052%
12	D	2	0,0009%	23	0,0103%
13	N	88	0,1103%	8	0,0100%
14	D/N	172	0,0575%	1613	0,5393%
15	D	22	0,0046%	1833	0,3866%
16	N	227	0,3155%	49	0,0681%
17	N	272	0,0949%	2074	0,7236%
18	D/N	541	0,0836%	6533	1,0099%
19	N	7	0,0095%	0	0,0000%
20	D	10	0,0042%	429	0,1803%
21	D	0	0,0000%	438	0,1217%
22	N	47	0,0618%	1	0,0013%
23	N	36	0,0120%	1043	0,3468%
24	D	159	0,0242%	5615	0,8531%
25	N	81	0,1085%	41	0,0549%
26	N	93	0,0311%	984	0,3294%
27	N	112	0,0163%	5316	0,7742%

Tabela 5: Primerjava rezultatov doseženih z eksaktno metodo in rezultatov CUT.

Skupno je eksaktna metoda našla boljšo rešitev v 64 podprimerih, hevristična metoda CUT v 139, medtem ko sta obe metodi v 67 podprimerih našli enakovredno rešitev (torej rešitev z enako izgubo materiala, kar ne pomeni nujno, da gre za enak načrt razreza). V skladu s pričakovanji daje eksaktna metoda boljše rezultate za manjše primere (v primerih, kjer je d<=5 in m<=5 eksaktna metoda najde boljšo rešitev tudi ob zelo kratkih časovnih omejitvah), CUT pa v večjih primerih (d>=10 in m>=10).

Skupna izguba v vseh 270 primerih je precej manjša pri uporabi CUT, saj znaša 5593 cm, medtem ko pri eksaktni metodi znaša 32268 cm. Vendar ima eksaktna metoda (poleg že omenjene primernosti za manjše probleme) nekatere prednosti, kot je na primer ta, da CUT vedno privede do enakega rezultata, medtem ko bi lahko rešitve, pridobljene z eksaktno metodo, izboljšali s povečanjem časovne omejitve ali procesorske moči računalnika. Pri tem je potrebno poudariti, da zaradi NP-polnosti problema pri večjih problemih to povečanje ne bi privedlo do bistvene izboljšave rezultatov ali celo do optimalne rešitve.

Druga prednost eksaktne metode je, da daje približen odgovor na to, kako daleč je pridobljena rešitev od optimalne, medtem ko CUT na to vprašanje ne odgovarja. Tako imata denimo primera 4 in 18 pri eksaktni metodi podobno odstotno izgubo materiala po 10 sekundah. Glede na to da je v primeru 4 optimalno zagotovo rešeno že 9 od 10 primerov, je jasno, da kakršnokoli povečanje časovne omejitve ali morebitna sprememba metode ne more privesti do bistveno boljših rezultatov, saj se lahko izboljša rešitev le v enem podprimeru.

V primeru 18 pa v 10 sekundah ni optimalno rešen noben primer. Torej bi lahko s povečanjem časa prišli do boljših rezultatov. Dejansko se v primeru 4 ob povečanju časa iz 10 na 60 sekund skupna izguba ne spremeni (ostane pri 1028 cm), medtem ko se v primeru 18 zmanjša iz 7936 cm na 6533 cm. Pri hevristični metodi pa ne moremo ugotoviti razlike med pridobljeno in dejansko optimalno rešitvijo.

Izbor metode

Ker obstaja več metod za rešitev istega problema, se seveda pojavi vprašanje, katero metodo izbrati za posamezen problem. Glede na primerjavo, predstavljeno v tabeli 5, je sicer očitno, da je eksaktna metoda primernejša za manjše probleme (kjer je denimo število palic na zalogi in naročil okoli 5 ali manj), CUT pa za večje probleme (kjer je število palic in naročil večje od 10). Vendar dosedanje analize niso dale odgovora na dve pomembni vprašanji:

- katera metoda naj se uporabi za primere, ki ležijo med zgoraj navedenima mejama,
- katere spremenljivke v modelu najbolj vplivajo na kompleksnost problema in s tem na verjetnost, da bo možno dobiti optimalno rešitev v vnaprej določeni časovni omejitvi.

Kot odgovor na prvo vprašanje bi seveda lahko vsak problem rešili z obema metodama in obdržali boljši rezultat. Vendar bi za to potrebovali dodaten čas in napor. Zato bi bilo dobro imeti način, kako vnaprej predvideti, katera metoda bo prinesla boljši rezultat. Za to smo v tem magistrskem delu uporabili metodo odločitvenih dreves (angl. decision tree). S pomočjo odločitvenih dreves smo poskušali za vsak primer vnaprej napovedati, ali bo eksaktna metoda v dani časovni omejitvi privedla do optimalnega rezultata ali ne.

Predstavitev odločitvenih dreves

Odločitveno drevo je v bistvu grafična predstavitev procedure za klasifikacijo primerov v posamezne razrede. S pomočjo odločitvenega drevesa lahko npr. na podlagi pacientovih simptomov ugotovimo verjetno bolezen (Decision tress, 2002). Ena od glavnih prednosti

odločitvenih dreves je tudi ta, da jih lahko uporabljamo brez računalnika ter natančno vidni razlogi, ki so privedli do odločitve (Lenič, 2002).

Pri odločitvenem drevesu so učni primeri predstavljeni kot par (lastnosti atributov, odločitev). Lastnosti so zbirka več atributov, ki opisujejo posamezen primer (v našem primeru so to denimo število palic, število naročil…). Izbiro atributov, ki jih uporabimo pri gradnji odločitvenega drevesa, moramo narediti vnaprej. Odločitev je znana pri učni množici, ne pa tudi pri objektih, o katerih bomo kasneje sprejemali odločitve (Kokol, 2000, str. 75). V našem primeru torej za objekte v učni množici vemo, ali je eksaktna metoda pripeljala do optimalne rešitve ali ne.

Ti podatki o lastnostih in odločitvah pri posameznih učnih objektih (angl. training objects) v učni množici (angl. training set) nam torej predstavljajo osnovo za izgradnjo odločitvenega drevesa. Namen odločitvenega drevesa je poiskati neka pravila, ki določajo, v kateri razred naj bi spadal posamezen primer. Jasno je, da pri realnih primerih nobena klasifikacija na podlagi nekaj atributov ne more biti 100% natančna, lahko pa poskušamo napovedati, v kateri razred spada posamezen nov primer.

Osnovna ideja in algoritem, po katerem deluje program See5/C5[7], ki smo ga uporabili v tem magistrskem delu, je naslednja (Hunt, 1966; Quinlan, 1993, str. 17-18): imamo T testnih primerov, ki pripadajo različnim razredom (razrede označimo $\{C_1, C_2\ldots, C_k\}$. Imamo tri možnosti:

- T vsebuje več primerov, ki vsi pripadajo razredu C_j. Odločitveno drevo za T je list (angl. leaf), ki kaže, da primeri v tem listu pripadajo razredu C_j,
- T ne vsebuje nobenega primera. V tem primeru to spet predstavlja list. Razred, ki je povezan s tem listom, mora biti sedaj določen na nek drug način. Program See5 povezan razred določi na podlagi najpogostejšega razreda na predhodnem vozlu (angl. parent node),
- T vsebuje več primerov, ki pripadajo različnim razredom. V tem primeru T razdelimo na več podskupin. V tem primeru izberemo tak test A, ki na podlagi preverjanja vrednosti enega atributa, množico T razdeli na več medsebojno izključujočih delov. Tako je sedaj odločitveno drevo za T sestavljeno iz enega vozlišča (angl. node), ki predstavlja test A, in po ene veje za vsako od podmnožic. To proceduro ponavljamo za vsako od podmnožic, dokler ne dobimo končnega odločitvenega drevesa.

Odločilno vprašanje je seveda, kako v vsaki fazi izbrati test A, ki bo razdelil učno podmnožico na dva dela. Cilj je kreiranje takega drevesa, ki bo čim bolje opisalo učno množico in bo torej čim manjše. Tako drevo ima namreč tudi največjo napovedno moč za napovedovanje rezultatov novih primerov.

Načeloma bi lahko izdelali vsa možna drevesa in izbrali najenostavnejšega. Vendar je dokazano, da je to NP-poln problem (Hyafil, 1976) in optimalne rešitve torej za srednje in večje probleme ne moremo najti. Zato se v praksi uporabljajo hevristični algoritmi,

[7] C5 je oznaka za verzijo programa, ki deluje na Unix-u, See5 pa za verzijo, ki deluje v operacijskem sistemu Windows. V magistrskem delu smo uporabili verzijo za Windows.

običajno na podlagi požrešnih metod. Pri vsaki delitvi torej izmed vseh možnih testov izberemo tistega, ki privede do največje pridobitve v informacijski količini (angl. information gain).

Informacijsko količino, potrebno za določitev i-tega stanja sistema, ki se v tem stanju nahaja z verjetnostjo p_i, izračunamo po formuli (Gradišar, 2001, str. 55):

$$I(X) = -\log_2 p_i$$

Če torej izberemo en primer iz skupine S primerov in ugotovimo, da ta primer pripada razredu C_j, je verjetnost za to enaka:

$$p_j = \frac{freq(C_j, S)}{|S|}$$

Informacijska količina za ta primer znaša:

$$I(X) = -\log_2 (\frac{freq(C_j, S)}{|S|}) \text{ bitov.}$$

Če sedaj seštejemo informacijsko količino za vse primere iz skupine S dobimo naslednjo formulo (Quinlan, 1993, str. 21):

$$\inf o(S) = -\sum_{j=1}^{k} \frac{freq(C_j, S)}{|S|} * \log_2 (\frac{freq(C_j, S)}{|S|})$$

info(S) torej meri povprečno količino informacije (oz. entropijo) potrebno za določitev razreda, v katerega spada posamezen element.

Po delitvi množice na več podmnožic lahko sedaj povprečno količino informacije izračunamo kot ponderirano vsoto posameznih povprečnih količin informacij po formuli:

$$\inf o_x(S) = \sum_{i=1}^{n} \frac{|S_i|}{|S|} * \inf o(S_i)$$

Želimo seveda izbrati tako delitev, ki bo privedla do največje pridobitve v informacijski količini. Torej izberemo tisti test, ki ima maksimalno vrednost info(S)-$info_x$(S).

Problem tega kriterija je v tem, da močno favorizira tiste teste, ki imajo večje število možnih izidov (Quinlan, 1993, str. 23). Če bi torej denimo želeli klasificirati študente Ekonomske fakultete po tem, koliko let potrebujejo do diplome, in bi kot atribut med drugim vključili tudi številko indeksa, bi po tem kriteriju učno množico razdelili ravno po številki indeksa na toliko podmnožic, kot je študentov. V vsaki podmnožici bi bil tako samo en študent, entropija vsake podmnožice pa bi bila 0. Očitno pa je, da številka indeksa kot atribut dejansko nima nobene prediktivne vrednosti.

To pomanjkljivost odpravimo tako, da ustrezno popravimo (»normaliziramo«) pridobitev informacijske količine pri testih z več možnimi izidi. Tako namesto absolutne pridobitve v količini informacije izračunamo količnik. Najprej izračunamo potencialno pridobitev v količini informacije, če razdelimo množico T v n podmnožic. To izračunamo po naslednji formuli:

$$split \inf o(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} * \log_2 (\frac{|T_i|}{|T|})$$

Vrednost split info (X) se praviloma povečuje s povečevanjem različnih izidov pri posameznem testu.

Nato količnik izračunamo tako, da delimo pridobitev v informaciji s tem faktorjem:

količnik= (info(S)-info$_x$(S))/split info (X)

Ta količnik torej meri, koliko je koristne (netrivialne) pridobitve v količini informacije

Tako popravljen količnik pridobitve količine informacije praviloma daje bistveno boljše rezultate kot sam kriterij pridobitve informacije (Quinlan, 1988), zato ga uporablja tudi program See5.

Dodaten problem je upoštevanje zveznih spremenljivk pri izdelavi odločitvenega drevesa, saj je načeloma težko določiti mejo, ki naj se uporabi za delitev množice na dva dela. Kot prag se lahko uporabi katerakoli vrednost med v_i in v_{i+1}[8] (če je denimo v_6=5 in v_7=10, katerokoli število med 5 in 10 povzroči enako delitev učne množice). Ena od možnosti je preslikava teh atributov v diskretni prostor tako, da vrednosti razdelimo na intervale in nato na vsak interval gledamo kot na diskretno vrednost. Intervale lahko razdelimo na ekvidistančne dele, lahko jih razdelimo z določitvijo praga ali z dinamično delitvijo (Kokol, 2000).

Program See5 uporablja eno od različic delitve intervala z določitvijo praga, zato to metodo predstavljamo nekoliko obširneje (Kokol, 2000; Quinlan, 1993). Imamo nek zvezni atribut A, ki v učni množici zavzema N različnih vrednosti. Če sedaj razporedimo te vrednosti po velikosti, dobimo množico $\{v_1, v_2, \ldots, v_n\}$. Sedaj torej dobimo N-1 možnih pragov (angl. threshold) za delitev učne množice na dva dela, saj so vse vrednosti med v_i in v_{i+1} medsebojno enakovredne, ker učno množico razdelijo na enak način[9]. Pri delitvi množice torej upoštevamo vse možne pragove in izberemo tistega, ki prinese največji pridobitek pri informaciji. Ta prag seveda izberemo le, v kolikor je ta pridobitek večji od tistega pri morebitni delitvi po vrednosti kakšnega drugega (diskretnega ali zveznega) atributa.

Zaradi več možnih delitev se sicer nekoliko podaljša čas izdelave drevesa, vendar za običajne primere še vedno lahko pričakujemo hiter rezultat v nekaj sekundah. Dodatna prednost je, da pravzaprav ni potrebno preiskati vseh N-1 možnih pragov. V kolikor vrednosti v_i in v_{i+1} pripadata istemu razredu, je namreč dokazano, da nobeden od pragov med tema vrednostma ne more biti optimalen (Fayyad, 1992).

To metodo je uporabljala prejšnja verzija programa C4.5., pri čemer je kot prag bila izbrana manjša od obeh vrednosti[10]. Osnovni problem je v tem, da imajo tako zvezni atributi prednost pri izbiri, po katerem atributu bomo delili učno množico[11]. Do tega pride,

---

[8] Predpostavljamo seveda, da so vrednosti v množici v urejene po velikosti, bodisi padajoče ali naraščajoče

[9] Imamo samo N-1 in ne N+1 možnih pragov, kajti vsak prag, ki je manjši od v_1 ali večji od v_n ne pride v poštev, saj ne razdeli množice na dva dela, ampak vsi primeri ostanejo v eni množici.

[10] Načeloma lahko kot prag izberemo katerokoli vrednost med v_i in v_{i+1}. V praksi se dostikrat uporablja kar sredina tega intervala, torej $\frac{v_1 + v_2}{2}$, medtem ko C4.5. uporablja kar vrednost v_1.

[11] Ravno tako imajo prednost zvezni atributi z večjim številom različnih vrednosti pred tistimi z manjšim (Quinlan, 1996).

ker pri zveznih atributih lahko izberemo enega od N-1 možnih pragov, pri diskretnih pa je na razpolago precej manjše število možnih delitev. Zato je program dostikrat izbral delitev po zveznem atributu, čeprav bi delitev po diskretnem bolje opisala dejansko stanje (Quinlan, 1996; Auer, 1995; Dougherty, 1995).

Možne rešitve tega problema temeljijo na upoštevanju Ginijevega koeficienta, χ^2 kontingenčne statistične tabele in podobnega (Han, 2001, str. 291-292). See5 ta problem odpravi z dvema popravkoma pri izbiri ustreznega testa za delitev učne množice (Quinlan, 1996; str. 79-80) :

1. prvi popravek temelji na principu minimalne potrebne opisne dolžine (angl. Minimum Description Length principle) (Rissanen, 1983). Po tem principu se izbere tista delitev, ki potrebuje minimalno število bitov za opis teorije in izjem. Test po zvezni spremenljivki tako zahteva dodatnih $\log_2$ (N-1) bitov za opis teorije, pri čemer N-1 predstavlja število pragov in s tem število možnih delitev. Za to vrednost torej znižamo po prej navedeni formuli izračunano pridobitev informacije za delitev po zvezni spremenljivki,

2. drugi popravek temelji na tem, da pri izračunu količnika pridobitve v informacijski količini to pridobitev delimo s split info. Slednja je funkcija praga in je maksimalna, kadar je enako število primerov pod in nad pragom. Zato najprej izberemo prag t, ki prinese maksimalno korist, nato pa je končni izbor atributa, ki bo uporabljen za razdelitev množice, narejen na podlagi prilagojene pridobitve v informaciji.

Do sedaj predstavljen način kreiranja odločitvenega drevesa pomeni, da na koncu dobimo zelo razvejano drevo z mnogo listi, saj se drevo deli toliko časa, dokler niso v vsaki podskupini samo primeri, ki pripadajo istemu razredu ali pa nadaljnji testi ne bi prinesli novega izboljšanja. To bi pomenilo preveliko prilagajanje (angl. overfitting) razpoložljivim podatkom, kar bi privedlo do zmanjšanja natančnosti predvidevanja (Quinlan, 1993, str. 35). To pomanjkljivost lahko odpravimo s poenostavljanjem (angl. pruning) odločitvenega drevesa in sicer z uporabo ene od dveh možnosti:

- predhodno klestenje (angl. prepruning ali stopping): običajno to izvedemo tako, da za vsako podskupino določimo najboljši način za delitev v dve podskupini z vidika statistične značilnosti, zmanjšanja števila napak, pridobitve informacije ali po kakem drugem kriteriju. Če taka delitev ne doseže vnaprej določene meje, potem te delitve ne izvedemo, ampak pustimo celo podskupino v enem listu.

  Prednost tega pristopa je predvsem ta, da je nekoliko hitrejši, saj ne izgubljamo časa z določanjem vej v drevesu, ki jih nato niti ne bomo uporabili. Osnovni problem tega pristopa je prav v določitvi višine tega kriterija – previsok kriterij povzroči premalo razvejano drevo, prenizek pa drevesa ne poenostavi dovolj (Breiman, 1984).
- naknadno klestenje (angl. postpruning): pri tem pristopu najprej zgradimo celotno odločitveno drevo, ki ga nato postopoma poenostavimo z odstranjevanjem posameznih vej. To naredimo tako, da del drevesa zamenjamo z enim samim listom ali eno vejo. To lahko naredimo na več načinov, denimo z uporabo algoritma na podlagi stroškovne kompleksnosti (angl. cost complexity). Po tem algoritmu za vsak vozel izračunamo pričakovano stopnjo napake, če bi ta vozel oklestili. Nato izračunamo še pričakovano stopnjo napake za neoklesteno drevo. Če je slednja večja, potem oklestimo ta del

drevesa. Ta postopek ponavljamo toliko časa, dokler ne pridemo do drevesa z najmanjšo pričakovano stopnjo napake (Han, 2001; str. 290).

Odločitvena drevesa smo izbrali kot ustrezno metodo za problem razreza, saj naš problem izpolnjuje pogoje za uspešno uporabo odločitvenih dreves, ki jih navaja Quinlan (Quinlan, 1993, str. 2):

- opis problema z atributi: vsi podatki o posameznem primeru morajo biti izraženi s fiksnim številom lastnosti oziroma atributov. Vsak atribut ima lahko diskretne ali številske vrednosti, bistveno je, da imajo vsi primeri iste atribute.

  Ta pogoj je izpolnjen, saj lahko dejansko vsak primer dobro opišemo z vnaprej določenimi atributi, kot so število palic v zalogi, število naročil, povprečno povpraševanje po posameznem naročilu in različna razmerja (denimo med celotnim materialom in celotnimi potrebami). Več o izbiri atributov, ki smo jih vključili v model, podajamo v nadaljevanju,
- prej definirani razredi: kategorije, v katere bomo razporejali primere, morajo biti določene vnaprej. Gre za tako imenovano nadzorovano učenje.

  V našem primeru smo vnaprej določili dve kategoriji, in sicer:

  1. problem je rešen optimalno v vnaprej določeni časovni omejitvi (razred 1),
  2. optimalna rešitev z eksaktno metodo v časovni omejitvi ni bila najdena (razred 0),
- diskretni razredi: razredi morajo biti medsebojno ostro ločeni – posamezen primer bodisi spada v en razred ali pa ne. Celotno število razredov mora biti bistveno manjše od števila primerov, ki so na razpolago za izgradnjo odločitvenega drevesa. Naš primer izpolnjuje oba pogoja, saj je ločnica med obema razredoma jasna, število testnih primerov pa je (zaradi uporabe generatorja naključnih števil) lahko skoraj poljubno veliko in tako seveda znatno presega število razredov,
- zadostno število podatkov: metoda odločitvenih dreves temelji na različnih statističnih testih, ki ugotavljajo vzorce v podatkih. Zato da ločimo dejanske od morebitnih naključnih vzorcev, je potrebno zadostno število primerov, ki jih lahko uporabimo za kreiranje vzorcev. Koliko je zadostno število, je odvisno od kompleksnosti modela in števila atributov ter razredov.

  Glede na to da lahko z generatorjem naključnih števil generiramo poljubno število primerov in jih rešimo z avtomatično proceduro, je tudi ta pogoj v našem primeru izpolnjen,
- logični modeli za klasifikacijo: Pravila za odločanje se morajo dati izraziti kot skupek logičnih pravil o vrednosti posameznih atributov. Ta metoda torej ni primerna, če želimo posamezne atribute ponderirati in na tej podlagi priti do rezultata.

Praktična uporaba odločitvenih dreves

Pri našem problemu enodimenzionalnega razreza se lahko ta metoda uporabi v različnih primerih in med drugim pomaga pri odgovoru na naslednja vprašanja:

- da ugotovimo, ali je bolje, da posamezen problem rešimo s hevristično ali eksaktno metodo. Primer te uporabe prikazujemo v tem poglavju,
- da statistično ugotovimo, kateri faktorji imajo največji vpliv na časovno kompleksnost pri reševanju problema s predlagano metodo,
- da ugotovimo ustrezno velikost pri določanju podproblema. Dostikrat je možno problem razdeliti na več manjših problemov in rešiti vsakega od teh problemov optimalno ali pa rešiti večino problema hevristično in le manjši del optimalno (to je običajno najpomembnejši del ali del, ki prispeva največ k skupni izgubi). Statistična analiza prikazana v nadaljevanju tega poglavja nam omogoča, da natančneje določimo ustrezno velikost podproblema in faktorje, ki jih moramo pri določitvi podproblema upoštevati. Ugotovitve iz tega poglavja smo uporabili pri izdelavi kombinirane metode za reševanje, ki jo podrobneje predstavljamo v naslednjem poglavju.

Najprej smo želeli torej ugotoviti, pri katerih primerih lahko pričakujemo optimalno rešitev z uporabo eksaktne metode v časovnem intervalu ene minute, ki smo ga arbitrarno določili kot najdaljši čas, ki ga še lahko porabimo za reševanje posameznega problema v praksi. Kot je razvidno iz prejšnjih poglavij, je večinoma jasno, da pri problemih, kjer je palic v zalogi manj kot 5, lahko z veliko verjetnostjo pričakujemo optimalno rešitev v tej časovni omejitvi (oziroma v praksi večinoma še precej prej, običajno v nekaj sekundah). Po drugi strani pri primerih z več kot 10 različnimi palicami v zalogi eksaktna metoda zelo verjetno ne bo dala optimalne rešitve v tem času. Za vmesne primere odgovor ni povsem jasen, ravno tako še nismo ugotovili, kateri faktorji (poleg števila palic) najbolj vplivajo na verjetnost za pridobitev optimalne rešitve.

Zato smo z uporabo odločitvenih dreves preverili primere, pri katerih se število palic v zalogi giblje med 5 in 10. Testne primere smo podobno kot prej pridobili s pomočjo generatorja naključnih števil PGEN, določanje parametrov (njihov pomen je isti kot v sliki 2) pa je prikazano v sliki 4.

od g=1 ***do*** 3
 od h=1 ***do*** 3
  od i=1 ***do*** 3
   od j=1 ***do*** 3
    od k=1 ***do*** 3
     u_1=1000*g
     u_2=2000*g
     m=(j*2)+3
     d=(i*2)+3
     v_1=h*100
     v_2=h*200
     n=(g*2)+3
     seme=1000000*n+1000*v1+10*v2+10*d+m
     r=5
     Poženi PGEN (n, v_1, v_2,, d, m, u_1, u_2, seme, r)
    naslednji k
   naslednji i
  naslednji j
 naslednji h
naslednji g

Slika 4: Shema procedure PROGEN, uporabljene za generiranje testnih problemov za izdelavo odločitvenega drevesa.

Na ta način pridobimo 1215 testnih primerov (po 5 primerov za vsako od 243 uporabljenih kombinacij parametrov, saj je vrednost parametra r enaka 5), kar zadostuje za kreiranje odločitvenega drevesa.

Vsak od navedenih primerov je bil nato rešen z eksaktno metodo na isti način in na istem računalniku kot že primeri v 2. poglavju. Za vsak primer je bil zabeležen celotni čas reševanja, izguba (optimalna rešitev ali najboljša najdena rešitev v 1 minuti, v kolikor primer ni bil rešen optimalno) ter predvsem dejstvo, ali je bil primer rešen optimalno ali ne.

Celoten eksperiment (torej generiranje podatkov, reševanje problemov in sumiranje rezultatov) je vsega skupaj trajal nekaj več kot 10 ur. Procedura za izvajanje eksperimenta je bila napisana v Visual Basic for Applications, MS Excel je bil uporabljen za zbiranje in prikaz pridobljenih rezultatov.

Prvih 30 kombinacij parametrov in skupne izgube ter število optimalno rešenih podproblemov je prikazanih v tabeli 6 (zaradi velikega števila podatkov ne prikazujemo vseh pridobljenih podatkov). Pomen oznak v tabeli je enak kot v prejšnjih tabelah. V vsaki vrstici je združenih 5 podprimerov, ki so bili generirani z uporabo istih parametrov, v

zadnjem stolpcu je prikazano število optimalno rešenih primerov za to kombinacijo. Pri izdelavi odločitvenega drevesa je seveda vsak primer vključen ločeno.

zap št	n	v_1	v_2	d	m	u_1	u_2	seme	skupna izguba	skupna izguba (v %)	optim
1	5	100	200	5	5	1000	2000	5102055	1	0,0053%	5
2	5	100	200	5	5	2000	4000	5102055	0	0,0000%	5
3	5	100	200	5	5	3000	6000	5102055	0	0,0000%	5
4	5	100	200	5	7	1000	2000	5102057	0	0,0000%	5
5	5	100	200	5	7	2000	4000	5102057	0	0,0000%	5
6	5	100	200	5	7	3000	6000	5102057	0	0,0000%	5
7	5	100	200	5	9	1000	2000	5102059	0	0,0000%	5
8	5	100	200	5	9	2000	4000	5102059	0	0,0000%	5
9	5	100	200	5	9	3000	6000	5102059	0	0,0000%	5
10	5	100	200	7	5	1000	2000	5102075	9	0,0301%	4
11	5	100	200	7	5	2000	4000	5102075	0	0,0000%	5
12	5	100	200	7	5	3000	6000	5102075	0	0,0000%	5
13	5	100	200	7	7	1000	2000	5102077	5	0,0181%	3
14	5	100	200	7	7	2000	4000	5102077	0	0,0000%	5
15	5	100	200	7	7	3000	6000	5102077	0	0,0000%	5
16	5	100	200	7	9	1000	2000	5102079	4	0,0138%	3
17	5	100	200	7	9	2000	4000	5102079	0	0,0000%	5
18	5	100	200	7	9	3000	6000	5102079	0	0,0000%	5
19	5	100	200	9	5	1000	2000	5102095	98	0,2976%	2
20	5	100	200	9	5	2000	4000	5102095	0	0,0000%	5
21	5	100	200	9	5	3000	6000	5102095	0	0,0000%	5
22	5	100	200	9	7	1000	2000	5102097	13	0,0350%	3
23	5	100	200	9	7	2000	4000	5102097	0	0,0000%	5
24	5	100	200	9	7	3000	6000	5102097	0	0,0000%	5
25	5	100	200	9	9	1000	2000	5102099	4	0,0116%	4
26	5	100	200	9	9	2000	4000	5102099	0	0,0000%	5
27	5	100	200	9	9	3000	6000	5102099	0	0,0000%	5
28	5	200	400	5	5	1000	2000	5204055	801	2,0828%	4
29	5	200	400	5	5	2000	4000	5204055	6	0,0145%	5
30	5	200	400	5	5	3000	6000	5204055	1	0,0024%	5

Tabela 6: Prvih 150 generiranih kombinacij parametrov in njihove rešitve.

Preden z uporabo programa See5 kreiramo odločitveno drevo, je potrebno odgovoriti še na eno pomembno vprašanje. Potrebno je namreč izbrati spremenljivke, ki jih bomo uporabili kot atribute za izdelavo drevesa. Očitno moramo vključiti tiste spremenljivke, za katere na podlagi dosedanjih rezultatov pričakujemo, da vplivajo na časovno kompleksnost reševanja problemov z eksaktno metodo. Tako smo vključili naslednje spremenljivke:

- število palic v zalogi, število naročil, povprečno povpraševanje po posameznem naročilu: očitno te spremenljivke vplivajo na velikost modela, saj število palic v zalogi in število naročil vplivata na število celoštevilskih spremenljivk ter omejitev v modelu, medtem ko

povpraševanje po posamezni dolžini naročila vpliva na število možnih kombinacij. Vse te spremenljivke so zato vključene kot atributi pri izdelavi odločitvenega drevesa.

- povprečnih dolžin palic in naročil seveda ne bi bilo smiselno vključiti kot absolutne številke, saj absolutno spreminjanje teh vrednosti ne vpliva na model (če bi na primer vse dolžine pomnožili s faktorjem 10, s tem modela ne bi čisto nič spremenili). Zato so te spremenljivke vključene kot del dveh razmerij:

- r – razmerje med povprečno dolžino palice in povprečno dolžino naročila: $\frac{u_1 + u_2}{v_1 + v_2}$. V (Gradišar, 1999) je bilo statistično ugotovljeno, da povečanje tega razmerja večinoma pomeni tudi boljše rešitve pri uporabi hevristične metode; pri tem je mišljena manjša izguba materiala. Želeli smo ugotoviti, kakšen ima vpliv to razmerje pri uporabi eksaktne metode, pri čemer naj poudarimo, da nas tu ne zanima, kakšna je absolutna ali relativna izguba, ampak kakšna je verjetnost, da bo posamezen primer rešen optimalno,

- q – razmerje med celotnim razpoložljivim materialom in celotnimi potrebami: $\frac{\sum_{j=1}^{m} d_j}{\sum_{i=1}^{n} s_i * b_i}$ - večje razmerje med celotnim materialom in celotnimi potrebami naj bi v skladu s pričakovanji pomenilo večjo verjetnost, da bomo pridobili optimalno rešitev. To trditev smo želeli empirično preveriti.

Omenjene podatke smo nato uporabili za generiranje odločitvenega drevesa. Pri tem smo uporabili 70% podatkov za trening, 30% pa kot testne podatke za preverjanje natančnosti odločitvenega drevesa. Razporeditev podatkov v obe skupini je bila naključna in jo je izvedel program See5.

Da bi preprečili pretirano prileganje podatkov, je bil postavljen dodaten pogoj, da mora vsaka veja v drevesu vsebovati vsaj 10 primerov. Pri deljenju drevesa torej ne upoštevamo tistih možnih delitev, ki bi učno množico razdelile tako, da bi ena od podmnožic imela manj kot 10 učnih objektov.

Tako pridobljeno odločitveno drevo je prikazano v sliki 5. Številke v oklepaju na sliki pomenijo, koliko podatkov, uporabljenih za trening, pripada posameznemu listu. Prva številka pomeni število pravilno, druga število napačno klasificiranih primerov.

```
q > 2.18239:
:...m <= 7:  1  (126/2)
:   m > 7:
:   :...n > 5: 1 (76/2)
:       n <= 5:
:       :...q <= 2.86825 : 0 (16/4)
:                q > 2.86825  : 1 (44/1)
q <= 2.18239:
:...r>15: 1 (30/2)
   r<=15
   :...m > 5
          :.....q > 1.69128
          :          :...n <= 5 : (28/2)
          :          :   n > 5  : (50/20)
          :          q <= 1.69128
          :          :...m > 7  : 0 (124/6)
          :              m<=7
          :              :
          :              ...n>=5 : 0 (115/20)
          :                n<=5
          :                :....r <= 5 : 1 (22/4)
          :                           r >  5 : 0 (16/2)
          m<=5
          :...r <=5 : 1 (65/7)
         r > 5
                   :....q>1.6137 : 1 (19)
                       q<=1.6137
                       :..n<=5 : 1 (27/9)
                              n>5
                              :...r <=6.667: 0(20/7)
                                 r > 6.667
                                      :.n<=7: 0(38/17)
                                        n>7
                                        :..q<=0.78137: 1(14/3)
                                                 q >0.78137 : 0 (20/4)
```

Slika 5: Odločitveno drevo za ocenitev verjetnosti, da bo problem rešen optimalno glede na vrednosti parametrov (Trkman, 2002a).

V celoti je odločitveno drevo pravilno predvidelo, ali bo problem rešen optimalno v eni minuti v 86,1% podatkov za trening in 84,1% testnih podatkov, kar predstavlja precej

visok odstotek. To odločitveno drevo lahko brez težav implementiramo kot podprogram v katerem koli programskem jeziku in ga nato uporabimo za avtomatično izbiro ustrezne metode za vsak individualni primer.

Iz slike 5 je tudi razvidno, da ima za to vrsto in velikost problema na kompleksnost največji vpliv razmerje med celotnim materialom in potrebami, precejšen vpliv pa ima tudi število palic. Ti dve spremenljivki se namreč nahajata precej visoko v odločitvenem drevesu. Vpliv števila naročil in povprečnega povpraševanja po posameznem naročilu ima presenetljivo majhen vpliv, saj se nahajata nizko v odločitvenem drevesu. Te ugotovitve smo uporabili v nadaljevanju magistrske naloge pri načinu izbire velikosti podproblema pri kombinirani metodi.

Očitno je, da samo visok delež pravilno klasificiranih testnih podatkov in podatkov za trening ne more biti zadostno zagotovilo, da je odločitveno drevo res ustrezno za izbiro pravilne metode pri problemih take velikosti. Zato smo odločitveno drevo dodatno testirali na podatkih, prikazanih v tabeli 5. Ker so bili tudi rezultati v tabeli 5 pridobljeni na istem računalniku in z istim solverjem, so primerni za testiranje odločitvenega drevesa.

	Dovolj materiala?	izguba CUT		izguba eksaktna metoda			
zap. št.		cm	%	cm	%	izbrana metoda	storjena napaka
1	D	8	0,0213%	1	0,0027%	E	0
2	D	0	0,0000%	0	0,0000%	H	0
3	D	0	0,0000%	0	0,0000%	H	0
4	D/N	1182	1,5460%	1028	1,3445%	E	0
5	D	28	0,0162%	212	0,1229%	H	0
6	D	9	0,0039%	62	0,0269%	H	0
7	N	1940	2,7256%	1702	2,3912%	E	0
8	D	213	0,0980%	1457	0,6706%	H	0
9	D	285	0,0832%	1780	0,5196%	H	0
10	D/N	59	0,0807%	18	0,0246%	H	41
11	D	0	0,0000%	8	0,0052%	E	8
12	D	2	0,0009%	23	0,0103%	E	21
13	N	88	0,1103%	8	0,0100%	E	0
14	D/N	172	0,0575%	1613	0,5393%	H	0
15	D	22	0,0046%	1833	0,3866%	H	0
16	N	227	0,3155%	49	0,0681%	E	0
17	N	272	0,0949%	2074	0,7236%	H	0
18	D/N	541	0,0836%	6533	1,0099%	H	0
19	N	7	0,0095%	0	0,0000%	E	0
20	D	10	0,0042%	429	0,1803%	H	0
21	D	0	0,0000%	438	0,1217%	E	438
22	N	47	0,0618%	1	0,0013%	E	0
23	N	36	0,0120%	1043	0,3468%	H	0
24	D	159	0,0242%	5615	0,8531%	H	0
25	N	81	0,1085%	41	0,0549%	H	40
26	N	93	0,0311%	984	0,3294%	H	0
27	N	112	0,0163%	5316	0,7742%	H	0

Tabela 7: Izbira ustrezne metode na podlagi odločitvenih dreves.

V skladu z drevesom bi morali z eksaktno metodo rešiti primere z naslednjimi zaporednimi številkami iz tabele 5: 1, 4, 7, 11, 12, 13, 16, 19, 21 in 22, medtem ko naj bi ostale rešili s hevristično metodo. Ti podatki so predstavljeni tudi v tabeli 7. V predzadnjem stolpcu oznaka H pomeni, da je bila na podlagi odločitvenega drevesa izbrana hevristična, oznaka E, da je bila izbrana eksaktna metoda. V zadnjem stolpcu je izračunana napaka pri izboru kot razlika med izgubo metode, ki jo je izbralo drevo, in izgubo dejansko najboljše metode za ta primer.

Od teh 27 primerov dajeta v dveh primerih (primera 2 in 3) obe metodi enak rezultat, ki je tudi optimalen. Zato v teh dveh primerih seveda ne moremo ugotavljati, ali je odločitveno drevo izbralo pravo metodo. Od preostalih 25 problemov je drevo pravo metodo izbralo v 20 primerih (80,0%), v 5 primerih pa je naredilo napako. Od teh napak so štiri imele za posledico le manjše povečanje izgube (gre za primer 11, kjer je 8 cm namesto 0 cm; primer 12, kjer je izguba 23 cm namesto 2 cm; primer 25, kjer je izguba 81 cm namesto 41 cm in primer 10, kjer je izguba 59 cm namesto 18 cm), le v enem primeru je bila storjena večja napaka (primer 21, kjer je izguba 438 cm namesto 0 cm, kot bi bila, če bi drevo za reševanje tega problema izbralo hevristično metodo). Tri napake so bile posledica izbora eksaktne namesto hevristične metode, dve napaki pa izbora hevristične namesto eksaktne metode.

Najpomembnejši rezultat je, da bi bila celotna izguba 5593 cm, če bi vse probleme rešili s hevristično metodo, 32268 cm ob reševanju z eksaktno metodo in 5351 cm, če vsak problem rešimo s tisto metodo, ki jo je izbralo odločitveno drevo. To kaže na to, da predlagani pristop dejansko vodi k izboljšanim rezultatom in zmanjšani izgubi materiala. Očitno bi lahko vsak primer rešili z obema metodama in obdržali le boljši rezultat, vendar bi to zahtevalo dodaten napor in predvsem čas. Vnaprejšnja izbira ustrezne metode nam zagotavlja zelo podobne rezultate kot ob reševanju z obema metodama.

Pri tem je vendarle potrebno upoštevati, da sklepi v tem poglavju veljajo za tovrsten problem in za približni velikostni razred, iz katerega prihajajo obravnavani problemi. Za drugačen problem ali bistveno drugačno velikost istega problema (oziroma ob reševanju na bistveno hitrejšem ali počasnejšem računalniku, ob drugi časovni omejitvi, z drugačno metodo ali drugim solverjem itd.) ne moremo trditi, da na čas reševanja z eksaktno metodo vplivajo iste spremenljivke in to na isti način, kot pri primerih tega velikostnega razreda.

Zato naše odločitveno drevo ni splošno ustrezno za vsako velikost problema. Za drugačne probleme bi morali tovrstno analizo ponoviti, pri čemer nam obstoječe procedure za izvajanje eksperimenta in generatorji naključnih števil omogočajo, da to naredimo zelo enostavno, hitro in brez pretiranega napora.

Kombinacija hevristične in eksaktne metode

Do sedaj smo v magistrskem delu že predstavili hevristično in eksaktno metodo za rešitev problema razreza ter način, kako lahko izberemo ustrezno metodo glede na karakteristike posameznega problema. Obe metodi imata klasične pomanjkljivosti navedene v literaturi – eksaktna metoda ni primerna za večje primere, hevristična pa tudi pri manjših primerih dostikrat ne privede do optimalne rešitve. Tudi če je rešitev dejansko optimalna, tega

metoda ne ugotovi (razen pri izgubi 0). Ker obe metodi rešujeta isti problem, se hitro pojavi ideja, da bi metodi povezali.

V tem delu tako predstavljamo način, kako lahko s kombiniranjem hevristične metode za rešitev večine problema in eksaktne metode za rešitev manjšega dela dosežemo precej boljše rezultate kot s katero koli od treh prej predstavljenih metod.

Osnovna ideja tovrstne kombinacije je združitev prednosti hevristične metode (predvsem dejstva, da zelo hitro pridemo do dobre rešitve tudi pri večjih problemih) ter eksaktne metode (možnost, da pridemo do optimalne rešitve za manjše probleme). Hkrati želimo čim bolj zmanjšati slabosti obeh metod (torej dejstvo, da hevristična metoda praviloma ne daje optimalnih rešitev, in dejstvo, da eksaktna metoda ni primerna za večje probleme zaradi hitro rastoče časovne zahtevnosti).

Osnovna ideja te kombinacije je, da celoten problem najprej rešimo s CUT. Kot že omenjeno, je namreč pokazano, da je CUT boljši od obeh hevurističnih metod (Gradišar, 2002), zato smo ga izbrali za uporabo v novi metodi. CUT je namreč razširjena in izboljšana verzija prejšnje hevristične metode COLA (Gradišar, 1996; Gradišar, 1997).

Pri reševanju problema s CUT je običajno, da večino izgube povzroči nekaj slabše narezanih palic – zaradi vpliva končnih pogojev so to običajno nazadnje narezane palice. Zato te palice v 2. fazi ponovno narežemo z uporabo eksaktne metode, ki je uporabna, saj z njo poiščemo načrt razreza samo za manjše število palic, večji del problema pa rešimo s hevristično metodo.

Končni načrt razreza je tako kombinacija rezultata 1. (za večino palic) in 2. (za tiste palice, ki smo jih narezali ponovno) faze. Metodo smo poimenovali C-CUT (angl. combined cutting) in je prvič predstavljena v (Gradišar, 2002a).

V opisu metode uporabljamo naslednje nove spremenljivke:

p – število palic, ki jih je v načrtu razreza uporabila metoda CUT,

f – število palic, uporabljenih v 1. fazi, ki jih bomo vključili v podproblem za 2. fazo,

g – število neuporabljenih palic iz 1. faze, ki jih bomo vključili v podproblem za 2. fazo,

b'_i – število kosov dolžine s_i, ki jih bomo vključili v podproblem za drugo fazo.

Algoritem za rešitev ima naslednje korake in se nekoliko razlikuje za primere s preveč in premalo materiala. Algoritem je prikazan tudi v sliki 6.

1. korak: Rešimo problem razreza z metodo CUT in shranimo tako pridobljen rezultat

2. korak: če je rešitev zanesljivo optimalna, potem prenehaj in prikaži rešitev 1. faze kot končno rešitev.

Rešitev je zagotovo optimalna, če je izpolnjen eden od naslednjih pogojev:

(1) $\sum_{j=1}^{m} t_j = 0$ (za primere z dovolj materiala)

(2) $\sum_{j=1}^{m} \delta_j = 0$ (za primere s pomanjkanjem materiala)

(3) $(0 < (\sum_{j=1}^{m} d_j - \sum_{i=1}^{n} (s_i * b_i)) < UB) \wedge (\sum_{j=1}^{m} t_j = \sum_{j=1}^{m} d_j - \sum_{i=1}^{n} (s_i * b_i))$

Prva dva pogoja sta jasna – rešitev, pri kateri je izguba 0, je očitno zagotovo optimalna. Tretji pogoj morda zahteva nekoliko razlage – da je rešitev optimalna, lahko trdimo tudi pri primerih, pri katerih je materiala dovolj, vendar razpoložljiv material presega potrebe za manj kot UB. V kolikor je v takem primeru izguba enaka razliki med razpoložljivim materialom in potrebami, je taka rešitev zanesljivo optimalna. Tudi če bi vse ostanke po posameznih palicah združili v eni palici, bi bil tak ostanek še vedno krajši od zgornje meje in bi tako štel kot izguba. Od 270 testnih primerov je en tak, ki izpolnjuje 3. pogoj in lahko trdimo, da je rešitev optimalna, čeprav izguba ni 0.

Ob tem je potrebno poudariti, da je izpolnitev enega od navedenih pogojev zadostni, ne pa tudi potrebni pogoj za optimalnost rešitve. Kot že omenjeno, ne CUT ne COLA ne pokažeta, ali je rešitev, pridobljena s tema metodama, optimalna ali ne.[12] Zato se kot optimalne rešitve štejejo le tiste, ki izpolnjujejo enega od navedenih pogojev, vse ostale rešitve (med njimi po vsej verjetnosti tudi nekaj optimalnih, česar pa ne moremo dokazati) vstopijo v naslednjo fazo.

3. korak: v tem koraku določimo, katere palice in naročila bomo vključili v podproblem, ki ga bomo nato rešili z eksaktno metodo. Omenjene palice in naročila izberemo na podlagi rezultatov iz 1. faze, pri čemer se postopek nekoliko razlikuje za primere z dovolj in za primere s premalo materiala.

Za primere z dovolj materiala je postopek naslednji:

- *vse uporabljene palice ($y_j=0$) sortiramo po padajočem δ_j ($j=1...p$)*

Glede na to da so vse dolžine palic različne in da vrstni red, v katerem režemo palice, ni pomemben[13], lahko to sortiranje izvedemo, ne da bi kakorkoli vplivali na rešitev ali na manjšo splošno uporabnost metode. Palice sortiramo padajoče zato, da zagotovimo, da bomo v naslednjem koraku, ko bomo izbrali prvih nekaj palic iz te skupine, v podproblem vključili tiste palice, ki največ prispevajo h končni izgubi. Zelo pomembno je, da palice sortiramo po δ_j in ne po t_j. Le pri takem sortiranju namreč zagotovimo, da bomo v podproblem vključili tudi palico (v kolikor taka palica v posameznem primeru seveda obstaja), kjer je $u_j=1$, $\delta_j> 0$ in $t_j=0$ – gre torej za palico, ki ni bila narezana do konca, pri kateri je ostanek večji od UB in ki ne šteje kot izguba. Če te palice v podproblem ne bi vključili, bi bilo namreč možno, da bi pri končni rešitvi (ko bi združili rešitev hevristične metode in rešitev podproblema) dobili dve palici, ki bi imeli vrednost $u_j = 1$, kar bi bilo v nasprotju z omejitvijo 5.

- *vse neuporabljene palice ($y_j=1$) sortiramo po padajočem d_j ($j=p+1...m$)*

V podproblem vključimo tudi nekaj neuporabljenih palic, torej takih, ki niso bile vključene v prvotni načrt razreza. Osnovna ideja je v tem, da s tem povečamo število možnih kombinacij in posledično tudi možnost, da bo rešitev podproblema boljša od rešitve, ki jo je pridobil CUT.

---

[12] V (Gradišar, 1999, str. 567) se kot zadosten pogoj za optimalnost rešitve sicer navaja pogoj, da je skupna izguba manjša od najkrajše dolžine naročila, vendar to dejansko ni niti zadosten niti potreben pogoj za optimalnost.

[13] Vrstni red bi bil pomemben, v kolikor bi bilo več palic enake dolžine in bi v model vključili tudi stroške za nastavljanje rezila za različne vzorce ali ob podobnih razširitvah modela.

Neuporabljene palice sortiramo od največje do najmanjše zato, ker s tem povečamo razmerje med $\sum_{j=p+1}^{p+g} d_j$ in $\sum_{i=1}^{n} (s_i * b_i')$ ter tudi razmerje med povprečno dolžino palice in povprečno dolžino naročila v vzorcu. Statistične analize prikazane v tem magistrskem delu in tudi v (Gradišar, 1999) so pokazale, da povečanje teh razmerij poveča tudi možnost, da bo končno pridobljena rešitev boljša.

- *izberemo podmnožico palic in sicer tako, da izberemo prvih f uporabljenih in prvih g neuporabljenih palic iz prej sortiranih seznamov*

Statistična analiza z odločitvenimi drevesi je pokazala, da ima število palic precej večji vpliv na verjetnost, da bomo v določenem času našli optimalno rešitev kot pa število naročil ali povprečno povpraševanje po posameznem naročilu. Zato je število palic v podproblemu določeno vnaprej, medtem ko se število naročil in potreba po posameznem naročilu izračuna v naslednjem koraku. Vrednost f in g določimo glede na čas, ki je na razpolago za rešitev podproblema, kakovost uporabljenega solverja in procesorsko moč računalnika, na katerem poteka izdelava načrta razreza.

- *ugotovimo podmnožico naročil in izračunamo potrebno število kosov posameznega naročila*

Potrebno število kosov posameznega naročila enostavno izračunamo tako, da seštejemo vse kose, ki so bili izrezani iz tistih palic, ki smo jih vključili v podproblem. To torej izračunamo po naslednji formuli:

$$b_i' = \sum_{j=1}^{f} x_{ij} \forall i$$

Možno je, da za kako od naročil velja, da je $b'_i = 0$ – da torej ta dolžina naročila ni nastopala v nobeni od palic, ki smo jih vključili v podproblem. V tem primeru takšnega naročila seveda ne vključimo v podproblem.

Za primere s premalo materiala je postopek naslednji:

- *vse uporabljene palice (y_j=0) sortiramo po padajočem δ_j (j=1…p)*

Seveda so v primerih s premalo materiala vse palice uporabljene v načrtu razreza, neuporabljenih palic ne more biti (saj niti z uporabo vseh palic nismo uspeli v celoti izpolniti naročila).

- *izberemo prvih f palic s tako sortiranega seznama*

Motiv za izbiro teh palic je podoben kot pri prvem primeru – izberemo tiste palice, ki imajo največjo izgubo in so torej največ prispevale tudi k celotni izgubi pri hevristični rešitvi.

- *ugotovimo dolžine naročil, ki jih moramo vključiti v podproblem, in potrebno število kosov posamezne dolžine*

To izračunamo z naslednjo enačbo:

$$b_i' = \sum_{j=1}^{f} x_{ij} + (b_i - \sum_{j=1}^{m} x_{ij})$$

Iz enačbe je razvidno, da moramo poleg tistih kosov, ki so bili izrezani iz palic, ki smo jih vključili v podproblem, upoštevati tudi tiste kose, ki so pri hevristični rešitvi ostali nenarezani. Podobno kot pri prvem primeru dolžin naročil, kjer je $b'_i = 0$, ne vključimo v model.

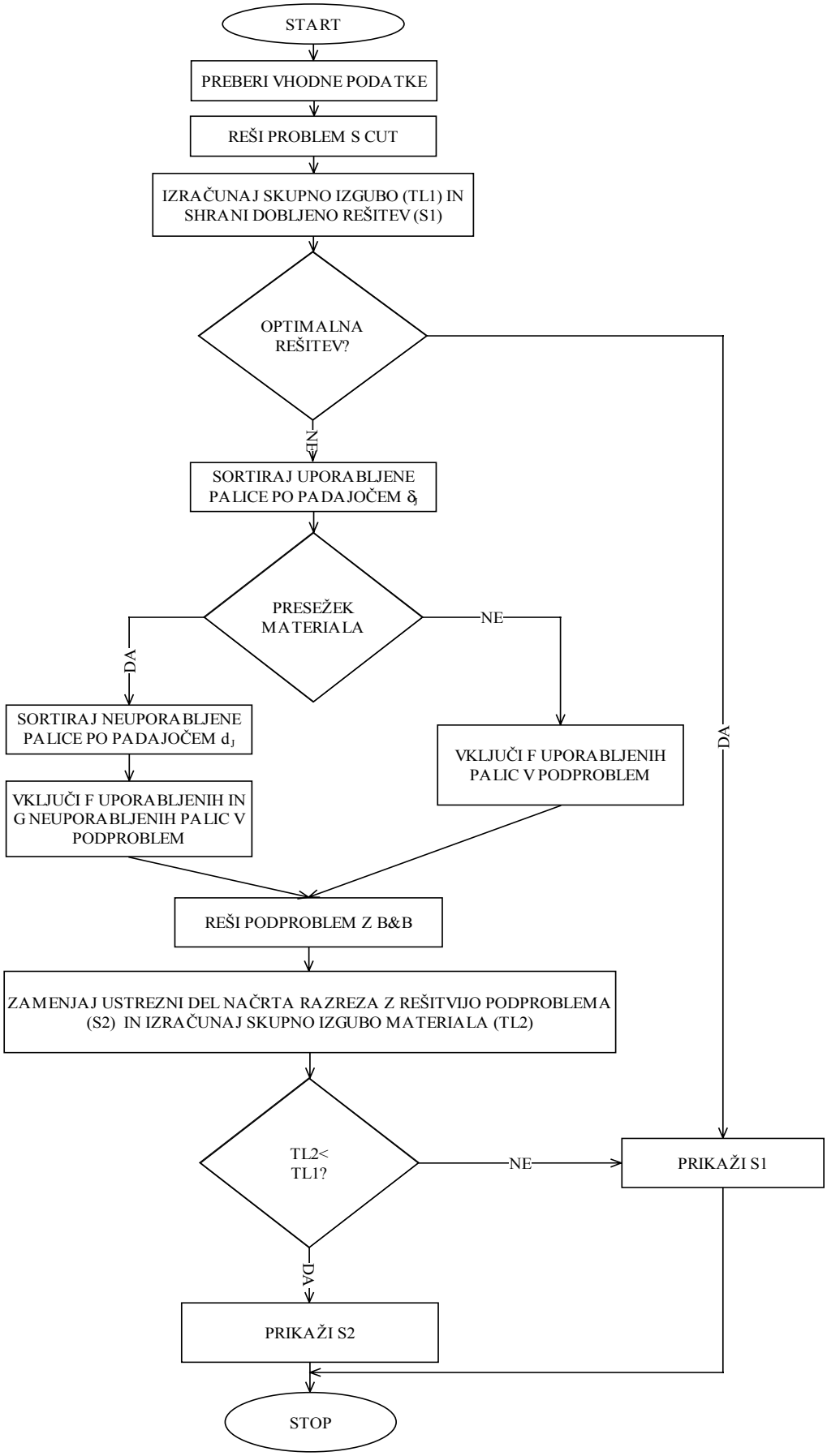
4. korak: Poiščemo rešitev podproblema z eksaktno metodo.

Način reševanja se ne razlikuje za primere s premalo in preveč materiala, moramo pa uporabiti ustrezen model, ki je bil predstavljen že v poglavju z eksaktno rešitvijo. Za reševanje lahko uporabimo katerikoli razpoložljiv solver za celoštevilčno linearno programiranje.

Parametra f in g sicer običajno določimo tako, da lahko pričakujemo, da bomo optimalno rešitev našli ob dani časovni omejitvi, kakovosti solverja in procesorski moči računalnika. Kljub temu je možno, da v določenem času ne najdemo optimalne rešitve ampak le neko začasno rešitev podproblema. V kolikor je ta začasna rešitev vseeno boljša od rešitve, ki jo je za podproblem našel CUT, obdržimo to rešitev, sicer pa kar hevristično rešitev.

5. korak: zamenjamo ustrezen del načrta razreza z rešitvijo podproblema, ki smo jo našli v 4. koraku, in ugotovimo, ali je tako pridobljena rešitev boljša od tiste, ki jo je našel CUT. Zaradi ustreznega algoritma rešitev, ki jo pridobi C-CUT, zagotovo ustreza vsem omejitvam v modelu. V kolikor je bila najdena optimalna rešitev podproblema, je nova rešitev seveda vsaj tako dobra kot prejšnja. Na koncu še prikažemo boljšo od obeh rešitev

Ta natančno opisan algoritem je prikazan tudi v sliki 6.



Slika 6: Prikaz algoritma metode C-CUT.

Če je podproblem v 4. koraku rešen optimalno, je nova rešitev vsaj enakovredna obstoječi, običajno pa je kar precej boljša, kot je razvidno tudi iz nadaljevanja te magistrske naloge (tabela 8). Rešitev podproblema je v prvi vrsti (če seveda predpostavimo, da so časovna omejitev, kvaliteta solverja in procesorska moč uporabljenega računalnika dani) odvisna od določitve parametrov f in g.

Večanje teh parametrov veča tudi kompleksnost problema in s tem verjetnost, da optimalne rešitve ne bomo našli v določeni časovni omejitvi. Zato morata biti f in g postavljena tako, da lahko optimalno rešitev pričakujemo s precejšnjo verjetnostjo. Povečanje parametra f hkrati tudi zmanjša razmerje med $\sum_{j=1}^{m} d_j$ in $\sum_{i=1}^{n} s_i * b_i^{'}$.

Po drugi strani nizke vrednosti f in g pomenijo, da rešitev podproblema ne more biti bistveno boljša od rešitve, ki jo je pridobil CUT, saj ne preostane več dosti možnih kombinacij za izdelavo načrta razreza.

Očitno je, da lahko postavimo f=g=0. V tem primeru se celoten problem reši hevristično z metodo CUT. Po drugi strani lahko postavimo tudi f=p in g=m-p, kar pomeni, da v podproblem vključimo vse palice v zalogi (in posledično seveda tudi vse kose naročil, ki jih potrebujemo). Tako se celoten problem reši z eksaktno metodo. To kaže na to, da je C-CUT dejansko razširitev obstoječih metod za rešitev tega problema in ob ustrezni postavitvi parametrov omogoča, da uporabimo tudi samo eno od metod, ki jih kombinira.

Kot že nekajkrat omenjano, je izbor parametrov f in g odvisen od:

- značilnosti problema (razmerja med $\sum_{j=1}^{m} d_j$ in $\sum_{i=1}^{n} s_i * b_i$ – torej med celotnim materialom in celotnimi potrebami; razmerja med povprečno d_j in povprečno s_j, torej med povprečno dolžino palice in povprečno dolžino naročila),
- procesorske moči računalnika,
- časovne omejitve za rešitev v 2. fazi,
- kvalitete solverja v 2. fazi.

Kot so pokazali eksperimenti, prikazani v prejšnjih poglavjih (vsi eksperimenti so se izvajali na istem računalniku in z uporabo istega solverja), je ustrezna vrednost za vsoto parametrov f in g nekje med 5 in 10. Pri večjem številu palic, vključenih v podproblem (na primer 15 vključenih palic ali več), bo rešitev, pridobljena z eksaktno metodo v eni minuti, veliko predaleč od optimalne. Pri manjšem številu rešitev ne bo mogla biti bistveno boljša od rešitve pridobljene s CUT.

Jasno je, da lahko trdimo, da je predlagana metoda znatno boljša od obstoječih hevrističnih metod le po obsežnem testiranju na praktičnih primerih. Zato smo metodo testirali na problemih že prej objavljenih v literaturi (Gradišar, 2002). V tem članku je bilo generiranih 270 problemov z generatorjem problemov PGEN in rešenih tako s CUT kot s COLA algoritmom. Kot že omenjeno, nam uporaba generatorja PGEN ob poznavanju uporabljenega semena omogoča, da zgeneriramo enake probleme in jih nato rešimo.

Podrobnosti o parametrih problemov in načinu za določanje semena so torej enake kot v (Gradišar, 2002) in so razvidne iz dinamične programske sheme procedure PROGEN, ki je prikazana v sliki 7.

Procedura PROGEN:

Od i= 1 ***do*** 3

Od j= 1 ***do*** 3

Od k=1 ***do*** 3

$n \leftarrow i \cdot 5$

$v_1 \leftarrow j \cdot 100$

$v_2 \leftarrow j \cdot 300$

$p \leftarrow 0$

$s_1 \leftarrow 0$

$s_2 \leftarrow 0$

$d \leftarrow k \cdot 10$

$m \leftarrow i \cdot j \cdot 10$

$u_1 \leftarrow j \cdot 500$

$u_2 \leftarrow j \cdot 1500$

$seme \leftarrow n \cdot 100000000$

$seme \leftarrow seme + v_1 \cdot 100000$

$seme \leftarrow seme + v_2 \cdot 1000$

$seme \leftarrow seme + d \cdot 100$

$seme \leftarrow seme + m$

$r \leftarrow 10$

kliči *PGEN (n, v_1, v_2, p, s_1, s_2, d, m, u_1, u_2, seme, r)*

naslednji

naslednji

naslednji

Slika 7: Shema procedure PROGEN, uporabljena za pridobitev 270 testnih podprimerov.

Kot smo že ugotovili v tem magistrskem delu, lahko optimalno rešitev z eksaktno metodo pričakujemo, v kolikor je število palic manj kot 10. Zato smo v primerih, kjer je preveč materiala, vrednost spremenljivke f postavili na 6, g pa na 2.

V kolikor je število palic, kjer je δ_j>0, manjše kot 6, smo f zmanjšali na število palic z izgubo ostankom večjim od 0, hkrati pa ustrezno povečali g, tako da je bil enak min(8-f, število neuporabljenih palic v prvotnem načrtu razreza). To se sicer običajno zgodi le v manjših primerih.

V primerih, kjer je materiala premalo, mora g očitno biti enak 0, tako da je f enak 8.

Praktični eksperiment, katerega rezultati so predstavljeni v naslednjih slikah in tabelah, je potrdil, da so bile take nastavitve za vrednosti f in g ustrezne in da opisana metoda dejansko vodi k manjši izgubi materiala. Časovna omejitev za drugo fazo algoritma je bila 60 sekund, pri čemer je bila večina primerov rešenih že v krajšem času.

Kot že omenjeno, je CUT algoritem napisan v programskem jeziku FORTRAN. Program za branje rezultatov prve faze, obdelavo podatkov in pripravo podatkov za drugo fazo je napisan v programskem jeziku Visual Basic for Applications (omenjen program po potrebi omogoča tudi ročno izbiro palic, ki jih želimo vključiti v podproblem za drugo fazo). V 4. koraku algoritma podproblem rešimo z MPL/CPLEX. Vsi koraki algoritma so bili izvedeni na istem računalniku kot vsi preostali testi v tem magistrskem delu (AMD, 1300 MHz).

V naslednjih slikah je prikazana rešitev drugega generiranega podproblema (prvi podproblem je bil optimalno rešen že v prvi fazi, torej samo s CUT algoritmom, zato tu prikazujemo rešitev drugega podproblema, ki je šel skozi obe fazi algoritma). V sliki 8 je prikazana rešitev 1. faze (torej rešitev pridobljena s CUT-om), v sliki 9 rešitev podproblema, pridobljena z eksaktno metodo, v sliki 10 pa končna združena rešitev C-CUT-a.

PODATKI O NAROČILIH

zap. št. naročila	dolžina	št. kosov
1	144	2
2	194	3
3	249	18
4	157	15
5	129	12

PODATKI O PALICAH V ZALOGI

zap. št. palice	dolžina
1	2663
2	1862
3	1805
4	1963
5	2461
6	1963
7	2196
8	1532
9	1954
10	1158

REZULTATI RAZPOREJENI PO ZAPOREDNI ŠTEVILKI NAROČIL

zap. št. naročila	zap. št. palice	št. kosov
1	9	1
1	1	1
2	7	2
2	1	1
3	7	6
3	8	3
3	9	6
3	1	3
4	2	2
4	7	2
4	8	5
4	9	2
4	1	4
5	2	12

PODATKI O OSTANKIH

zap. št. palice	δ_j
1	950
2	0
7	0
8	0
9	2

IZGUBA

zap. št. palice	izguba (t_j)
1	0
2	0
7	0
8	0
9	2

skupna izguba= 2 cm

Slika 8: Začetna rešitev pridobljena s CUT algoritmom.

V skladu z algoritmom in rezultati v sliki 8 so v 2. fazi v podproblem vključene palice z naslednjimi indeksi: 1, 3, 4, 5, 6, 9. Iz tega primera je tudi razvidna pomembnost sortiranja

palic po padajočem δ_j in ne po t_j. Če bi palice sortirali po padajoči izgubi, potem palica 1 ne bi bila vključena v model, kar bi v končni fazi pomenilo, da bi celotna rešitev, ki bi jo ponudil algoritem C-CUT, kršila omejitev 5. S takim izborom palic za drugo fazo smo zagotovili, da bo končna rešitev (tudi ko bomo kombinirali rezultate 1. faze in optimalne rešitve podproblema) ustrezala pogojem v modelu – da bo torej največ ena uporabljena palica vrnjena v skladišče.

PODATKI O NAROČILIH

zap. št. naročila	dolžina	št. kosov
1	144	2
2	194	1
3	249	9
4	157	6

PODATKI O PALICAH V ZALOGI

zap. št. palice	dolžina
1	2663
3	1805
4	1963
5	2461
6	1963
9	1954
10	1158

REZULTATI RAZPOREJENI PO ZAPOREDNI ŠTEVILKI NAROČIL

zap. št. naročila	zap. št. palice	št. kosov
1	3	1
1	5	1
2	3	1
3	3	4
3	5	5
4	3	3
4	5	3

IZGUBA

zap. št. palice	δ_j	izguba (t_j)
3	0	0
5	601	0

Slika 9: Rešitev v drugi fazi algoritma.

Eksaktna rešitev podproblema v 2. fazi je bila najdena v 0,17 sekunde in je prikazana v sliki 9. Iz slike 9 je tudi razvidno, kakšne so bile potrebe po posameznih dolžinah naročil v 2. fazi.

REZULTATI RAZPOREJENI PO DOLŽINAH NAROČIL

zap. št. naročila	zap. št. palice	št. kosov
1	3	1
1	5	1
2	7	2
2	3	1
3	7	6
3	8	3
3	3	4
3	5	5
4	2	2
4	7	2
4	8	5
4	3	3
4	5	3
5	2	12

skupna izguba = 0 cm

Slika 10: Končni načrt razreza pridobljen s C-CUT-om.

Dodamo naj, da je pri eksaktni rešitvi podproblema ostanek pri palici 5 enak 601 centimetrov in torej ne šteje kot izguba, saj je daljši od UB (249 cm). Skupna izguba je torej enaka 0 cm, kar pomeni, da smo zagotovo našli optimalno rešitev. Boljša rešitev od tiste, ki jo je ponudil algoritem CUT, je bila najdena tako, da se je v načrtu razreza namesto palic z indeksom 1 in 9 uporabilo palice z indeksom 3 in 5, kar je omogočilo zmanjšanje izgube za 2 centimetra.

Navedeni primer je namenjen predvsem za ilustracijo uporabe predlagane metode. Na podlagi enega samega primera seveda ne moremo sklepati o uporabnosti in učinkovitosti metode.

Zato so v tabeli 8 predstavljeni združeni podatki za vseh 270 primerov. V vsaki vrstici tabele je skupna izguba ugotovljena kot vsota izgub pri 10 problemih, ki jih je generirala procedura PROGEN z istimi parametri. Odstotek izgube je izračunan kot povprečje 10 odstotnih izgub.

zap. št.	n	v_1	v_2	d	m	u_1	u_2	seme	izguba CUT (cm)	izguba CUT (%)	izguba C-CUT (cm)	izguba C-CUT (%)
1	5	100	300	10	10	1000	3000	510301010	12	0,0160981	9	0,0120736
2	5	100	300	20	10	1000	3000	510302010	248	0,1423806	129	0,0740609
3	5	100	300	30	10	1000	3000	510303010	86	0,0426632	35	0,0173629
4	5	200	600	10	20	2000	6000	520601020	20	0,0126105	14	0,0088274
5	5	200	600	20	20	2000	6000	520602020	105	0,0287008	84	0,0229606
6	5	200	600	30	20	2000	6000	520603020	571	0,1120517	523	0,1026323
7	5	300	900	10	30	3000	9000	530901030	16	0,0070396	7	0,0030798
8	5	300	900	20	30	3000	9000	530902030	166	0,0313493	118	0,0222844
9	5	300	900	30	30	3000	9000	530903030	365	0,0481739	332	0,0438185
10	10	100	300	10	20	1000	3000	1010301020	8	0,0044288	0	0,0000000
11	10	100	300	20	20	1000	3000	1010302020	303	0,0794727	212	0,0556047
12	10	100	300	30	20	1000	3000	1010303020	52	0,0124638	0	0,0000000
13	10	200	600	10	40	2000	6000	1020601040	10	0,0029437	1	0,0002944
14	10	200	600	20	40	2000	6000	1020602040	18	0,0024302	6	0,0008101
15	10	200	600	30	40	2000	6000	1020603040	350	0,0312979	292	0,0261114
16	10	300	900	10	60	3000	9000	1030901060	5	0,0009259	0	0,0000000
17	10	300	900	20	60	3000	9000	1030902060	18	0,0015379	10	0,0008544
18	10	300	900	30	60	3000	9000	1030903060	324	0,0184679	238	0,0135659
19	15	100	300	10	30	1000	3000	1510301030	19	0,0069757	0	0,0000000
20	15	100	300	20	30	1000	3000	1510302030	276	0,0487203	96	0,0169462
21	15	100	300	30	30	1000	3000	1510303030	47	0,0077090	12	0,0019683
22	15	200	600	10	60	2000	6000	1520601060	13	0,0023932	0	0,0000000
23	15	200	600	20	60	2000	6000	1520602060	45	0,0038501	11	0,0009411
24	15	200	600	30	60	2000	6000	1520603060	126	0,0071349	83	0,0047000
25	15	300	900	10	90	3000	9000	1530901090	9	0,0011424	2	0,0002539
26	15	300	900	20	90	3000	9000	1530902090	11	0,0006482	0	0,0000000
27	15	300	900	30	90	3000	9000	1530903090	37	0,0013871	22	0,0008248
SKUPAJ									3260	0,0249999	2236	0,0159250

Tabela 8: Primerjava rezultatov med metodo CUT in metodo C-CUT.

Skupno je CUT zagotovo našel optimalno rešitev v 70 primerih. Tako je vsega skupaj v 2. fazo vstopilo 200 primerov. V teh 200 primerih je bila boljša rešitev najdena v 155 primerih, od tega je bila rešitev v 91 primerih zagotovo optimalna. Skupaj je bilo torej optimalno rešenih 161 od 270 primerov (oziroma 59,6%). Boljša rešitev je bila torej najdena v več kot 75% prvotno neoptimalno rešenih primerih, skupno število optimalno rešenih problemov se je več kot podvojilo.

Kot je razvidno iz tabele, se je skupna izguba zmanjšala za 31,4% (iz 3260 centimetrov na 2236 cm). Iz vsega napisanega je jasno, da C-CUT privede do bistveno boljših rezultatov kot CUT, ob tem pa ohrani nizko časovno kompleksnost, ki je ena od poglavitnih prednosti CUT-a v primerjavi z eksaktno metodo. Čas, potreben za rešitev, se namreč v primerjavi s CUT poveča največ za 1 minuto, saj je le-ta postavljena kot omejitev za reševanje podproblema, medtem ko se za druge operacije v algoritmu (kot je na primer izbira palic, ki tvorijo podproblem) porabi zanemarljivo malo časa.

Zaradi nizke časovne kompleksnosti je C-CUT tudi boljši od eksaktne metode za reševanje tega tipa problema, saj je primeren tudi za večje probleme, kjer je število palic večje od 15. Edino za reševanje zelo majhnih primerov je primernejša eksaktna metoda.

Razširitve problema

Model od upoštevanju oportunitetnih stroškov

V primerih s premalo materiala je jasno, da bodo nekatera naročila ostala neizpolnjena. Do sedaj smo predpostavljali, da je edini cilj, da čim bolj zmanjšamo izgubo materiala, medtem ko je vseeno, katera naročila ostanejo neizpolnjena. V tem poglavju problem razširimo s predpostavko, da so nekatera naročila pomembnejša od preostalih in bi njihova morebitna neizpolnitev za podjetje pomenila večji strošek kot neizpolnitev preostalih naročil. Poleg tega upoštevamo še, koliko časa posamezno naročilo čaka na izvedbo – s tem preprečimo, da bi v primeru konstantnega pomanjkanja materiala naročila z nizko prioriteto čakala na razrez neskončno dolgo.

Hkrati ne smemo pozabiti na osnovni cilj pri kreiranju vsakega načrta razreza – to je čim manjša izguba materiala. Za razliko od vseh dosedanjih modelov, razvitih v tem magistrskem delu, se torej sedaj srečujemo s trikriterijskim problemom, kjer imamo tri cilje:

- izpolniti predvsem tista naročila, ki so pomembnejša,
- doseči čim manjšo izgubo materiala,
- razrezati predvsem naročila, ki že dolgo čakajo na izvedbo.

Pri obravnavi tega problema uporabljamo podoben pristop, kot je predstavljen v (Wäscher, 1990), ki smo ga ustrezno popravili in prilagodili za naš primer. V tem članku se minimizirajo celotni stroški razreza, v našem primeru pa minimiziramo oportunitetne stroške. Za vsako dolžino naročila ocenimo, kakšne oportunitetne stroške povzroči neizpolnitev posameznega kosa tega naročila. Nato vrednost oportunitetnih stroškov pomnožimo s številom neizpolnjenih naročil te dolžine in tako dobimo skupne oportunitetne stroške neizpolnitve naročil.

Ker sedaj ni vseeno, katera naročila ostanejo nenarezana, za kriterijsko funkcijo ne moremo več uporabiti $\sum_{i=1}^{n} \delta_j$. Zato uporabimo funkcijo $\sum_{i=1}^{n} \Delta_i * s_i$, pri čemer dolžino posameznega naročila zamenjamo z oportunitenimi stroški tega naročila. Tako dobimo izraz:

$$\sum_{i=1}^{n} \Delta_i * op_i$$

Oportunitetne stroške lahko denimo izračunamo po formuli[14]:

$$op_i = b_i * (1 + m * \sqrt{čas_i}) * (1 + n * prioriteta_i)$$

Seveda moramo ustrezno popraviti tudi omejitev 3, tako da dobimo naslednji model:

(1) min $\sum_{i=1}^{n} \Delta_i * op_i$ (minimiziraj število nenarezanih kosov, ponderirano z oportunitetnimi stroški)

s pogojem, da

(2) $\sum_{i=1}^{n} (s_i * x_{ij}) < d_j \qquad \forall j$ (nahrbtnik omejitev)

(3) $\sum_{j=1}^{m} x_{ij} + \Delta_i = b_i \quad \forall i$ (omejitve povpraševanja – posamezne dolžine naročila ne smemo narezati več, kot je povpraševanje po njej)

(4) $x_{ij} \geq 0$, celoštevilčni $\forall i, j$

$\Delta_i \geq 0 \qquad \forall i$

Na ta način smo torej spet prišli do običajnega enokriterijskega problema, ki ga lahko spet rešimo s poljubnim solverjem za linearno programiranje.

S to formulo v bistvu korigiramo vrednost b_i (dolžino posameznega naročila) za nek faktor, ki ga določimo na podlagi dosedanjega časa čakanja na razrez in prioritete posameznega naročila. m in n sta faktorja, ki določata relativno prioriteto pri razrezu in s katerima lahko določamo, kaj nam je pri načrtu razreza pomembneje (čim manjša izguba, čim boljša izpolnitev naročil z dolgim čakalnim časom ali čim boljša izpolnitev naročil z visoko prioriteto). Višja vrednost teh uteži pomeni večjo relativno pomembnost tega faktorja.

Seveda lahko določimo tudi m=n=0. V tem primeru je vrednost kriterijske funkcije enaka kot v prejšnjih primerih in spet minimiziramo samo izgubo materiala.

Čas čakanja se v vsakem naslednjem obdobju povečuje za 1, oportunitetni stroški pa se povečujejo s korenom časa čakanja. Prioriteto lahko določimo arbitrarno na podlagi tega, kakšno škodo bi podjetju povzročila neizpolnitev posameznega naročila. Ta škoda je lahko bodisi zastoj v nadaljnji proizvodnji zaradi pomanjkanja materiala, izguba dobička pri nadaljnji prodaji, kazen za neizpolnitev naročila določena v pogodbi ali kaj četrtega.

Prioriteta je vključena iz jasnih razlogov – pomembna naročila želimo izpolniti čim prej. Čas čakanja je dodan kot faktor v enačbo, zato da v primeru konstantnega pomanjkanja materiala manj pomembna naročila na izpolnitev ne bi čakala neskončno dolgo.

Seveda je v enačbi za izračun oportunitetnih stroškov vključena tudi dolžina naročila – s tem zagotovimo, da model upošteva tudi izgubo materiala kot enega od faktorjev pri optimizaciji.

---

[14] To formulo lahko načeloma določimo arbitrarno, odvisno od tega kakšno pomembnost pripisujemo posameznim faktorjem, v našem primeru času in prioriteti posameznega naročila. Po potrebi bi lahko v formulo vključili tudi druge relevantne faktorje.

V tako opredeljeno enačbo lahko po potrebi dodajamo tudi druge faktorje, ki vplivajo na to, da je izpolnitev posameznega naročila pomembnejša od preostalih (kot na primer, če bi želeli prej izpolniti daljša naročila zaradi zmanjšanja izgub v kasnejših razrezih).

Jasno je, da vsako povečanje obeh ponderjev ali vključitev novih kriterijev v model povečuje izgubo materiala, saj le-ta sedaj ni več edini faktor, ki vpliva na izdelavo načrta razreza. V primerjavi z modelom, predstavljenim v drugem poglavju, je sedaj izguba materiala večja (ali v najboljšem primeru enaka), kot je bila v prejšnjem modelu.

Uporabo razširjenega modela razreza prikazujemo na praktičnem primeru. Tako kot prejšnji primeri je bil tudi ta rešen z uporabo programa MPL/CPLEX na osebnem računalniku (AMD, 1300 MHz).

PODATKI O NAROČILIH

zap. št. naročila	dolžina	št. kosov	čakalni čas	prioriteta	oport. stroški
1	144	22	2	1	266,6222
2	194	11	2	1	359,1994
3	249	29	0	1	323,7000
4	157	37	0	3	298,3000

PODATKI O PALICAH V ZALOGI

zap. št. palice	dolžina
1	2663
2	1805
3	2461
4	1963

IZGUBA

zap. št. palice	izguba
1	7
2	1
3	1
4	0

REALIZACIJA

zap. št. naročila	načrt	realizacija	razlika
1	22	20	2
2	11	1	10
3	29	0	29
4	37	37	0

Slika 11: Rešitev problema enodimenzionalnega razreza ob upoštevanju oportunitetnih stroškov v 1. obdobju.

Vrednost ponderjev m in n je bila postavljena na 0,3, predstavljamo pa razrez v dveh obdobjih, kjer je bilo obakrat premalo materiala. Vsi podatki o dolžinah palic, dolžinah, potrebnem številu, prioriteti in čakalnem času posameznih naročil ter številu narezanih kosov so predstavljeni v sliki 11 (za 1. obdobje) ter sliki 12 (za 2. obdobje) – natančnega načrta razreza tokrat ne navajamo.
V drugem obdobju se čas čakanja pri naročilih, ki so preostala iz prvega obdobja, poveča za 1, v proizvodnjo pride nov delovni nalog za izpolnitev novega naročila (13 kosov dolžine 188 s prioriteto 2 in časom čakanja 0), poleg tega v skladišče pridejo še nove dolžine palic (vsi podatki so razvidni tudi iz slike 12)

PODATKI O NAROČILIH

zap. št. naročila	dolžina	št. kosov	čakalni čas	prioriteta	oport. stroški
1	144	2	3	1	284,4720
2	194	10	3	1	383,2470
3	249	29	1	1	420,8100
5	188	13	0	2	300,8000

PODATKI O PALICAH V ZALOGI

zap. št. palice	dolžina
5	2518
6	1638
7	2019
8	1791

IZGUBA

zap. št. palice	izguba
5	0
6	0
7	9
8	9

REALIZACIJA

zap. št. naročila	načrt	realizacija	razlika
1	2	2	0
2	10	10	0
3	29	20	9
5	13	4	9

Slika 12: Rešitev problema enodimenzionalnega razreza ob upoštevanju oportunitetnih stroškov v 2. obdobju.

Skupna izguba materiala je torej 27 cm (9 cm v prvem ter 18 cm v drugem obdobju). Če bi problem reševali brez upoštevanja oportunitetnih stroškov, bi prišli do izgube 0 cm, pri čemer bi za razliko od predstavljenih rezultatov v slikah 11 in 12 narezali nekaj več kosov naročila 3 in 5, zato pa bi slabše izpolnili naročila 1 in predvsem 4 (slednje ima med vsemi največjo prioriteto). Vidimo, da se je izguba materiala povečala le za nekaj centimetrov, saj 27 cm predstavlja približno 0,16% celotnega materiala. To verjetno ni previsoka cena za boljšo izpolnitev preostalih dveh ciljev. S povečanjem ponderjev m in n bi dosegli še boljše izpolnjevanje teh ciljev, seveda ob še nekoliko višji izgubi.

Ta model je možno razširiti tudi tako, da se izguba materiala ne poveča, hkrati pa vendarle upoštevamo tudi druge faktorje, ki so podrejeni glavnemu cilju. Dostikrat je namreč možno, da obstaja več različnih rešitev istega problema, ki imajo enako izgubo. Brez dodatnega kriterija se pač naključno izbere ena od ekvivalentnih rešitev.

Ko govorimo o naključnem izboru, je seveda potrebno povedati, da je algoritem, po katerem deluje razveji&omeji deterministične in (ob istih podatkih in istih nastavitvah za izbor vozlov, ki jih bomo razmejili, in drugih nastavitvah v programu) vedno privede do iste rešitve. Z naključnim izborom mislimo na to, da nobena od teh rešitev z isto izgubo materiala nima prednosti in da je bila do sedaj odločitev o izboru med ekvivalentnimi rešitvami prepuščena programu. Z ustrezno spremembo modela lahko zagotovimo, da bo v primeru obstoja več rešitev z enako izgubo vedno izbrana tista rešitev, ki bolje zadovoljuje preostale cilje.

Če želimo zagotoviti, da se izguba ne bo povečala niti za centimeter, mora biti razlika med oportunitetnimi stroški in dejansko dolžino posameznega naročila tako majhna, da bodo preostali faktorji upoštevani le pri izbiri med rešitvami z isto izgubo materiala. Ta razlika mora biti torej tako majhna, da nobena rešitev A, ki ima večjo izgubo kot rešitev B, ne bo imela boljše vrednosti nove kriterijske funkcije od rešitve B. Dodatni kriteriji se bodo tako uporabili le, če bosta imeli obe rešitvi isto izgubo. Povedano drugače: veljati mora naslednje:

$$\left(\sum_{j=1}^{m}\delta_{jA} > \sum_{j=1}^{m}\delta_{jB}\right) \Leftrightarrow \left(\sum_{i=1}^{n}\Delta_{iA} * op_i > \sum_{i=1}^{n}\Delta_{iB} * op_i\right)$$

V zgornji enačbi se indeks A nanaša na vrednosti spremenljivk pri rešitvi A, indeks B pa na vrednosti pri rešitvi B (oportunitetni stroški so pri obeh rešitvah izračunani na enak način).

Izpolnitev zgornjega pogoja najlažje dosežemo tako, da faktorja m in n nastavimo tako nizko, da bo veljalo:

$$\sum_{i=1}^{n}\left|(b_i - op_i) * s_i\right| < 1$$

Z izpolnitvijo tega pogoja smo dosegli, da bo doprinos razlike med oportunitetnimi stroški in dolžino posameznega naročila skupno manjši kot 1. Tako nobena rešitev, ki je po modelu za primere s premalo materiala (brez upoštevanja stroškov) slabša od druge, od te rešitve ne more biti boljša po vključitvi stroškov. Če je ena rešitev slabša od druge, to

namreč pomeni, da je slabša vsaj za 1, saj so vrednosti kriterijske funkcije pri modelu brez upoštevanja stroškov celoštevilčne.

S tem smo dosegli želeno: med rešitvami z enako izgubo materiala ne izbiramo več naključno, ampak izberemo tisto rešitev, ki bolje izpolnjuje dodatne pogoje (kar smo omenili že prej, velja tudi tu – ti dodatni pogoji so lahko tudi kaj drugega kot čas čakanja na razrez in prioriteta posameznega naročila). Nekaj praktičnih preizkusov kaže na to, da tak pristop dejansko privede do drugačne rešitve ob isti izgubi materiala in nekoliko daljšem času reševanja. Eden od preprostih primerov je predstavljen v sliki 13 (podatki za ta primer), sliki 14 (rešitev brez upoštevanja oportunitetnih stroškov) in sliki 15 (rešitev ob upoštevanju oportunitetnih stroškov). Oba ponderja m in n sta bila postavljena na 0,0000002. Kot je razvidno iz obeh slik, je izguba v obeh primerih 0 cm, vendar je pri drugi rešitvi narezano bistveno več (15 namesto 21) kosov dolžine številka 4, ki ima najvišjo prioriteto. To je posledica različnega razreza tretje palice, medtem ko je bil pri obeh rešitvah za prvi dve palici najden isti načrt razreza.

PODATKI O NAROČILIH

zap. št. naročila	dolžina	št. kosov	čakalni čas	prioriteta	oport. stroški
1	144	22	1	1	144,0001
2	194	11	1	1	194,0001
3	249	29	0	1	249,0000
4	157	37	0	8	157,0003

PODATKI O PALICAH V ZALOGI

zap. št. palice	dolžina
1	2663
2	1805
3	2461

Slika 13: Podatki za primer.

REZULTATI RAZPOREJENI PO ZAPOREDNI ŠTEVILKI PALICE

zap. št. palice	zap. št. naročila	št. kosov
1	1	4
1	2	3
1	3	1
1	4	8
2	1	1
2	2	1
2	3	4
2	4	3
3	1	11
3	2	0
3	3	1
3	4	4

REALIZACIJA

zap. št. naročila	načrt	realizacija	razlika
1	22	16	6
2	11	4	7
3	29	6	23
4	37	15	22

Slika 14: Rešitev brez upoštevanja oportunitetnih stroškov (model za premalo materiala).

REZULTATI RAZPOREJENI PO ZAPOREDNI ŠTEVILKI PALICE

zap. št. palice	zap. št. naročila	št. kosov
1	1	4
1	2	3
1	3	1
1	4	8
2	1	1
2	2	1
2	3	4
2	4	3
3	1	1
3	2	0
3	3	3
3	4	10

REALIZACIJA

zap. št. naročila	načrt	realizacija	razlika
1	22	6	16
2	11	4	7
3	29	8	21
4	37	21	16

Slika 15: Rešitev ob upoštevanju oportunitetnih stroškov.

Model ob upoštevanju stroškov skladiščenja

Do sedaj smo v tem magistrskem delu, tako kot praktično v vsej razpoložljivi literaturi s tega področja, predpostavljali, da so število in dolžina palic v skladišču fiksni – da torej na to ne moremo vplivati, ampak moramo pač poiskati optimalen načrt razreza za dane podatke.

V praksi sicer najpogosteje drži, da moramo v trenutku razreza uporabiti količine, ki so na razpolago v skladišču. Vsekakor pa podjetje dolgoročno običajno lahko vpliva na to, koliko materiala naroča pri dobaviteljih in kakšna je raven zalog v skladišču.

Obravnava posameznih modelov za planiranje zalog bi presegala okvir te magistrske naloge, zato se ukvarjamo predvsem z vplivom višine zalog na oblikovanje načrta razreza in izgubo pri tem.

Višje stanje zalog (več palic v skladišču) pomeni manjšo verjetnost, da posameznega naročila ne bomo mogli izpolniti in da bo moralo podjetje nositi s tem povezane oportunitetne stroške, predstavljene v prejšnjem poglavju. Tudi v primerih, ko je materiala dovolj, večje število daljših palic poveča verjetnost, da bo možno najti načrt razreza z majhno izgubo. Tako namreč obstaja večje število kombinacij, ravno tako pa se poveča razmerje med razpoložljivim materialom in potrebami.

Z vidika načrtovanja razreza je torej dobro čim višje stanje zalog v skladišču. Po drugi strani to povišuje nekatere stroške. Kot prvo gre za stroške povezane z držanjem zalog, kot so na primer stroški vezanega kapitala v zalogah, stroški skladiščenja, pakiranja zalog, možnih okvar, kraj, možnost zastaranja in pokvarljivosti, stroški zavarovanja, administrativni stroški, davek na premoženje in podobno (Thomas, 1980, str. 52; Rusjan, 1999, str. 137-138; Plossl, 1967, str. 52-53). Poleg tega moramo upoštevati še izgubo fleksibilnosti v proizvodnji, saj je v primeru visokih zalog težko in predvsem drago uvesti spremembe, ker bi le-te zahtevale odpis velikega dela zalog (Thomas, 1980, str. 102-104). Kot zadnje lahko zaloge povečujejo tudi nestabilnost v sistemu celotne proizvodne verige, saj majhna sprememba zalog pri končnem proizvajalcu lahko povzroči velike spremembe pri prvem proizvajalcu v verigi (Forrester, 1961).

Z razširjenim modelom želimo torej minimizirati vsoto stroškov skladiščenja in izgub pri razrezu, pri čemer seveda vnaprej ne vemo, koliko in kakšna naročila bomo potrebovali v posameznem obdobju. Z ocenjevanjem stroškov zalog pri stohastičnem povpraševanju se je ukvarjalo že dosti raziskovalcev (denimo (Ferbar, 1998)), zato nas tu zanima predvsem vpliv višine zalog na izdelavo načrta razreza.

V kolikor je vnaprej znano, kakšne bodo potrebe po materialu posameznih dolžin, lahko seveda tudi natančno planiramo, koliko palic posamezne dolžine potrebujemo za izvedbo razreza in skladiščenja z minimalnimi stroški za podjetje[15].

Običajno seveda tega podatka v trenutku, ko se odločamo za naročilo materiala pri dobavitelju, nimamo. Zato je ta problem rešljiv predvsem z uporabo različnih simulacij na podlagi razpoložljivih (običajno nepopolnih) podatkov o stroških skladiščenja, prihodnjih

[15] Če nam dobavitelj omogoča naročilo poljubnega materiala v poljubnih dolžinah, lahko v takem primeru seveda naročimo točno tak material, kot ga potrebujemo, in s tem pridemo do načrtov razreza z izgubo 0.

potrebah po materialih, možnosti za naročanje in podobnem. Planiranje materialnih potreb je področje, ki izvajanje simulacij, na podlagi katerih lahko preverjamo ustreznost dobavnih rokov za posamezna naročila, že omogoča (Ferbar, 1998, str. 7).

Povezava tega in problema razreza v ustrezen sistem in izvedba simulacij v taki kombinaciji je zaenkrat še precej neraziskano področje. Oceniti moramo torej vse stroške, povezane s skladiščenjem in razrezom, ter vpliv višjega stanja zalog na izgube v razrezu. Nato moramo najti tako stanje zalog in tak načrt razreza, ki bo te skupne stroške minimiziral.

Odločanje o višini zalog z vidika stroškov zalog in izgub pri razrezu je zato nedvomno področje, ki predstavlja še veliko možnosti za nadaljnje raziskovalno delo, predvsem pri povezavi stroškov zalog in izgub pri razrezu v enoten model.

Sklep

V magistrskem delu smo predstavili problem razreza, ki se pojavlja v številnih industrijskih procesih, v podobnih oblikah (npr. kot problem pakiranja) pa tudi v drugih gospodarskih panogah. Po krajši predstavitvi drugih metod za reševanje različnih oblik tega problema in doseganje raznolikih ciljev, ki jih želimo doseči v posameznih verzijah tega problema, smo se podrobneje osredotočili na splošni problem razreza, kjer so vse palice v zalogi lahko različnih dolžin.

Glavni cilj je doseči tak načrt razreza materiala v določeno število naročil vnaprej znanih dolžin, da bo izguba materiala čim manjša, saj s tem tudi pripomoremo k manjši porabi materiala in s tem nižjim stroškom za podjetje.

Za ta problem je bila do sedaj najbolj razširjena in uveljavljena metoda CUT, ki na podlagi hevristične procedure vodi do zelo dobrih rezultatov v veliki večini problemov. Vendar tako pridobljene rešitve večinoma niso optimalne. Zato smo v magistrskem delu predstavili metodo, ki privede do optimalnih rešitev. Vendar je ta metoda zaradi opisane NP-polnosti tega problema primerna samo za manjše probleme, kjer velikost problema ne preseže določene meje.

Ker meja velikosti problema, kjer je primernejša eksaktna kot hevristična metoda, ni natančno določena, smo v magistrskem delu uporabili pristop na podlagi odločitvenih dreves za izbor primerne metode v vsakem posameznem primeru. Pri tem pristopu na podlagi testnih podatkov izdelamo odločitveno drevo, ki z dovolj veliko zanesljivostjo lahko napove, ali bo določen nov problem v dani časovni omejitvi rešen optimalno ali ne. Na praktičnem primeru smo pokazali, da izbor metode na podlagi odločitvenega drevesa res vodi do manjše izgube, kot če bi uporabili samo eno ali drugo metodo.

Glede na znane pomanjkljivosti eksaktne (predvsem hitro povečevanje časa reševanja za večje probleme) in hevristične (neoptimalnost rezultatov) metode smo obe združili na tak način, da smo ohranili čim več prednosti obeh metod in hkrati omejili njune pomanjkljivosti. Na podlagi predstavljenih rezultatov lahko trdimo, da nova metoda, poimenovana C-CUT, dejansko vodi do precej boljših rezultatov za to vrsto problema kot obe do sedaj poznani metodi. Pri tem se je čas reševanja s to metodo v primerjavi z metodo

CUT le nekoliko podaljšal, v primerjavi z eksaktno metodo pa je bistveno krajši. Edino pri manjših problemih je ustreznejša uporaba eksaktne metode.
V do sedaj predstavljenih problemih smo kot edini cilj priprave načrta razreza upoštevali izgubo materiala. V zadnjem poglavju magistrske naloge predstavimo enostaven način, kako lahko v obstoječo metodo vključimo tudi druge faktorje, ne da bi pri tem zanemarili izgubo materiala kot enega od kriterijev za izbor končnega načrta razreza materiala. To pokažemo na primeru vključitve prioritete posameznega naročila in časa čakanja pri odločanju o tem, katere dolžine naročila bodo narezane v določenem časovnem obdobju.
S spreminjanjem vrednosti ponderjev lahko enostavno določamo relativno pomembnost posameznega faktorja. Ta pristop omogoča tudi vključitev morebitnih drugih faktorjev, ki jih glede na naravo problema želimo upoštevati pri izdelavi posameznega načrta razreza. Predstavimo tudi način, kako lahko ponderje postavimo tako, da izguba materiala ne bo večja kot pri optimalni rešitvi brez dodatnih omejitev, te dodatne omejitve pa bodo vseeno vsaj delno upoštevane. Tako kot vse ostale metode v tem magistrskem delu tudi to predstavimo na numeričnem primeru.
V zadnjem delu magistrskega dela prikažemo še možno razširitev problema z vključitvijo stroškov skladiščenja v problem razreza. Praktično vsa obstoječa literatura s področja razreza namreč razrez obravnava izolirano od ostalih poslovnih procesov v podjetju. Tako gledanje je običajno nekoliko preozko, saj ima podjetje tudi možnost vpliva na politiko zalog in s tem posledično na dolžine palic, ki so na razpolago za pripravo načrta razreza v vsakem obdobju. To je gotovo tema, ki daje še številne možnosti za nadaljnje raziskovanje.

Literatura

1. Abraham P. M., Kirby S. J., Ng Y. G.: Production schedule of a float glass process plant. System Engineering Project Report, School of Engineering Science and Industrial Management, University of Liverpool, 1976.
2. Antonio Julien, Chauvet Fabrice, Chu Chengbin, Proth Jean-Marie: The cutting stock problem with mixed objectives: Two heuristics based on dynamic programming. Amsterdam: European Journal of Operational Research, 114(1999), str. 395-402.
3. Armbuster Michael: A solution procedure for a pattern sequencing problem as part of one-dimensional cutting stock problem in the steel industry. Amsterdam: European Journal of Operational Research, 141(2002), str. 328-340.
4. Auer Peter, Holte Robert, Maass Wolfgang: Theory and application of agnostic pac-learning with small decision trees. Proceedings of Twelfth International Conference on Machine Learning, San Francisco: Morgan Kaufmann, 1995, str. 21-29.
5. Belov Gleb, Scheithauer Guntram: A cutting plane algorithm for one-dimensional cutting stock problem with multiple stock lengths. Amsterdam: European Journal of Operational Research, 141(2002), str. 274-294.

6. Bhadury Joy, Chandrasekaran Ramaswamy: Stock cutting to minimize cutting length. Amsterdam: European Journal of Operational Research, 88(1996), str. 69-87.
7. Breiman Leo, Friedman Jerome, Olshen Richard, Stone Charles: Classification and Regression Trees. Wadsworth: Belmont, 1984.
8. Carnieri Celso, Mendoza Guillermo, Luppold William: Optimal cutting of dimension parts from lumber with a defect: A heuristic solution procedure. Madison: Forest Products Journal, 43(1993), 9, str. 66-75.
9. Carvalho Valerio, Rodrigues Guimaraes: An LP-based approach to a two-stage cutting stock problem. Amsterdam: European Journal of Operational Research, 84(1995), str. 580-589.
10. Carvalho Valerio: LP models for bin packing and cutting stock problem. Amsterdam: European Journal of Operational Research, 141(2002), str. 253-273.
11. Chu Chengbin, Antonio Julien: Approximation algorithms to solve real-life multicriteria cutting stock problems. Baltimore: Operations Research, 47(1999), 4, str. 495-508.
12. Coffman Edward, Garey Richard, Johnson David.: Approximation algorithms for bin packing – An updated survey. Algorithm Design for Computer Systems Design, New York: Springer-Verlag, 1984, str. 49-106.
13. Degraeve Zeger, Vandebroek Martina: A mixed integer programming model for solving a layout problem in the fashion industry. Management Science, 44(1998), 3, str. 301-310.
14. Degraeve Zeger, Schrage Linus: Optimal Integer Solutions to Industrial Cutting Stock Problems. Informs Journal on Computing, 11(1999), 4, str. 406-419.
15. Doughtery James, Kohavi Ron, Sahami Mehran: Supervised and unsupervised discretization of continuous features. Proceedings of Twelfth International Conference on Machine Learning, San Francisco: Morgan Kaufmann, 1995, str. 194-202.
16. Dudzinski Krzysztof, Walukiewicz Stanislaw: Exact methods for the knapsack problem and its generalizations. Amsterdam: European Journal of Operational Research, 28(1987), str. 3-21.
17. Dyckhoff Harald: A typology of cutting and packing problems. Amsterdam: European Journal of Operational Research, 44(1990), str. 145-159.
18. Faggioli Enrico, Bentivoglio Carlo: Heuristic and exact methods for the cutting sequencing problem. Amsterdam: European Journal of Operational Research, 110(1998), str. 564-575.
19. Fayyad Usama, Irani Keki : Multi-interval discretization of continuous-valued attributes for classification learning. Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence. San Francisco: Morgan Kaufmann, 1992, str. 1022-1027

20. Ferbar Liljana: Nadgradnja modela planiranja materialnih potreb (MRP) z vključitvijo teorije odločitev in teorije iger pri stohastičnem povpraševanju. Doktorska disertacija. Ljubljana: Ekonomska fakulteta, 1998. 130 str.
21. Ferreira Soeiro, Neves Antonio, Fonseca Castro: A two-phase roll cutting problem. Amsterdam: European Journal of Operational Research, 44(1990), str. 185-196.
22. Fisher M. L: Worst case analysis of heuristic algorithm. Management Science 26(1980), str. 1-17.
23. Forrester Jay: Industrial Dynamics. Cambridge: MIT Press, 1961. 464 str.
24. Garey Michael, Johnson David: Computers and Intractability: A guide to the Theory of NP-Completeness. New York: W. H. Freeman. 340 str.
25. Gass Saul: Linear Programming, Methods and Applications. New York: McGraw-Hill, 1985. 532 str.
26. Gau Thomas, Wäscher Gerhard: CUTGEN1: A problem generator for the Standard One-Dimensional Cutting Stock Problem. Amsterdam: European Journal of Operational Research, 84(1995), str. 572-579.
27. Gilmore P. C., Gomory R. E.: A linear programming approach to the cutting stock problem. Baltimore: Operations Research, 9(1961), str. 849-859.
28. Gilmore P. C., Gomory R. E.: A linear programming approach to the cutting stock problem, Part II. Baltimore: Operations Research, 11(1963), str. 863-888.
29. Gradišar Miro, Jesenko Jože: Optimization of One Dimensional Cutting in Clothing Industry. Ljubljana: Informatica, 20(1996), str. 211-221.
30. Gradišar Miro, Jesenko Jože, Resinovič Gortan: Optimization of roll cutting in clothing industry. New York: Computer & Operations Research, 24(1997), str. 945-953.
31. Gradišar Miro, Resinovič Gortan, Jesenko Jože, Kljajić Miroljub: A sequential heuristic procedure for one-dimensional cutting. Amsterdam: European Journal of Operational Research, 114(1999), str. 557-568.
32. Gradišar Miro, Kljajić Miroljub, Resinovič Gortan: A hybrid approach for optimization of one-dimensional cutting. Amsterdam: European Journal of Operational Research, 119(1999a), str. 165-174.
33. Gradišar Miro, Resinovič Gortan, Kljajić Miroljub: Evaluation of algorithms for one-dimensional cutting. Working paper No. 90, Faculty of Economics, University of Ljubljana, 1999b. 14 str.
34. Gradišar Miro, Resinovič Gortan: Informatika v poslovnem okolju. Ljubljana: Ekonomska fakulteta, 2001. 508 str.
35. Gradišar Miro, Trkman Peter, Indihar Štemberger Mojca: Exact solution of general one-dimensional cutting stock problem. Working paper No. 123, Faculty of Economics, University of Ljubljana, 2001a. 17 str.
36. Gradišar Miro, Resinovič Gortan, Kljajić Miroljub: Evaluation of algorithms for one-dimensional cutting. New York: Computers & Operations Research, 29(2002), str. 1207-1220.

37. Gradišar Miro, Trkman Peter: A combined approach for solution of general one-dimensional cutting stock problem. Working paper No. 124, Faculty of Economics, University of Ljubljana, 2002a. 15 str.
38. Han Jiwaei, Kamber Micheline: Data mining: concepts and techniques. San Francisco: Morgan Kaufmann, 2001. 550 str.
39. Hifi Mhand: Exact algorithms for the guillotine strip cutting/packing problem. New York: Computers & Operations Research, 25(1998), str. 925-940.
40. Hinxman A. I.: The trim-loss and assortment problems: a survey. Amsterdam: European Journal of Operational Research, 44(1980), str. 175-184.
41. Hunt Earl, Marin Janet, Stone Philip: Experiments in Induction. New York: Academic Press, 1966.
42. Hutchinson David: A new uniform pseudorandom number generator. Communications of the ACM, 9(1966), str. 432-433.
43. Hyafil Laurent, Rivest Ronald.: Constructing optimal binary decision tress is NP-complete. Amsterdam: Information Processing Letters, 5 (1976), 1, str. 15-17.
44. Johnston Robert: Rounding algorithms for cutting stock problems. Singapore: Asia-Pacific journal of operational research, 3(1986), str. 166-171.
45. Kokol Peter, Hleb Babič Špela, Podgorelec Vili, Zorman Milan: Inteligentni sistemi. Maribor: Fakulteta za elektrotehniko, računalništvo in informatiko, 2001. 211 str.
46. Kozak Jernej: Podatkovne strukture in algoritmi. Ljubljana: Društvo matematikov, fizikov in astronomov SR Slovenije, 1986. 388 str.
47. Land A. H., Doig A. G.: An Automatic Method of Solving Discrete Linear Programming Problems. Evanston: Econometrica, 28 (1960), str. 496-520.
48. Lenič Mitja, Kokol Peter, Yamamoto Ryuichi: Internet storitev za podporo pri odločanju v medicini. Dnevi slovenske informatike 2002, Ljubljana: Slovensko društvo Informatika, 2002, str. 501-505.
49. Liang Ko-Hsin, Yao Xin, Newton Charles, Hoffman David: A new evolutionary approach to cutting stock problems with and without contiguity. New York: Computers & Operations Research, 29 (2002), str. 1641-1659.
50. Lodi Andrea, Martello Silvano, Vigo Daniele: Heuristic and meta heuristic approaches for a class of two-dimensional bin packing problems. Informs Journal on Computing, 11(1999), str. 345-357.
51. Lodi Andrea, Martello Silvano, Vigo Daniele: Recent advances on two-dimensional bin packing problems. Amsterdam: Discrete Applied Mathematics, 123(2002), str. 379-396.
52. Nilsson Nils: Problem Solving Methods in Artificial Intelligence. New York: McGraw-Hill, 1971.
53. Papadimitriou Christos, Steiglitz Kenneth: Combinatorial optimization. New Jersey: Englewood Cliffs, 1982. 496 str.

54. Park Stephen, Miller Keith: Random number generators: Good ones are hard to find. Communications of the ACM, 31(1988), str. 1192-1201.
55. Paull Allan.: Linear Programming: A Key to Optimum Newsprint Production. Pulp and Paper Magazine of Canada, 57(1956), 1, str. 85-90.
56. Pierce John: Some Large-Scale Production Scheduling Problems in the Paper Industry. Englewood Cliffs: Prentice-Hall, 1964. 255 str.
57. Pisinger David: Heuristics for the container loading problem. Amsterdam: European Journal of Operational Research, 141(2002), str. 382-392.
58. Plossl George, Wight Oliver: Production and Inventory Control. Englewood Cliffs: Prentice Hall, 1967. 432 str.
59. Quinlan Ross: Decision trees and multi-valued attributes. Oxford: Machine Intelligence, 11(1988), str. 305-318.
60. Quinlan Ross.: C 4.5: Programs for Machine Learning. San Mateo: Morgan Kaufmann Publishers, 1993. 302 str.
61. Quinlan Ross: Improved Use of Continuous Attributes in C4.5. San Francisco: Journal of Artificial Intelligence Research, 4(1996), str. 77-90.
62. Rissanen Jorma: A universal prior for integers and estimation by minimum description length. Seattle: Annals of Statistics, 11(1983), str. 416-431.
63. Rönnqvist Mikael: A method for the cutting stock problem with different qualities. Amsterdam: European Journal of Operational Research, 83(1995), str. 57-68.
64. Rusjan Borut: Management proizvodnje. Ljubljana: Ekonomska fakulteta, 1999. 296 str.
65. Scheitauer Guntram, Terno Johannes: The modified integer round-up property for one-dimensional cutting stock problem. Amsterdam: European Journal of Operational Research, 84(1995), str. 562-571.
66. Schilling Gordian, Georgiadis Michael: An algorithm for the determination of optimal cutting patterns. New York: Computers & Operations Research, 29(2002), str. 1041-1058.
67. Schrijver Alexander: Theory of linear and integer programming. Chichester: John Wiley & Sons, 1986. 471 str.
68. Stadtler Hartmut: A one-dimensional cutting stock problem in the aluminium industry and its solution. Amsterdam: European Journal of Operational Research, 44(1990), str. 209-223.
69. Suliman Saad: Pattern generating procedure for the cutting stock problem. Amsterdam: International Journal of Production Economics, 74(2001), str. 293-301.
70. Sweeney Paul, Paternoster Elizabeth.: Cutting and packing problems: A Categorised, Application-Orientated Research Bibliography. New York: Journal of the Operational Research Society, 43(1992), str. 691-706.
71. Thomas Adin: Stock Control in Manufacturing Industries. Hampshire: Gower Press, 1980. 221 str.

72. Trkman Peter, Gradišar Miro: Eksaktna rešitev problema enodimenzionalnega razreza. Dnevi slovenske informatike 2002. Ljubljana: Slovensko društvo Informatika, 2002, str. 355-359.
73. Trkman Peter, Gradišar Miro: Choice of method for General One-Dimensional Cutting Stock Problem. 2nd WSEAS International Conference on Simulation, Modeling and Optimization, september 2002a. Sprejeto v objavo
74. Umetani Shunji, Yagiura Mutsunori, Ibaraki Toshihide: One-dimensional cutting stock problem to minimize the number of different patterns. Amsterdam: European Journal of Operational Research, 2002. Sprejeto v objavo.
75. Vahrenkamp Richard: Random search in the one-dimensional cutting stock problem. Amsterdam: European Journal of Operational Research, 95(1996), str. 191-200.
76. Vanderbeck Francois: Exact algorithm for minimizing the number of setups in the one-dimensional cutting stock problem. Baltimore: Operations Research, 48(2000), str. 915-926
77. Vasko Francis, Newhart Dennis, Stott Kenneth: A hierarchical approach for one-dimensional cutting stock problems in the steel industry that maximizes yield and minimizes overgrading. Amsterdam: European Journal of Operational Research, 114(1999), str. 72-82.
78. Wagner Brett: A genetic algorithm for one-dimensional bundled stock cutting. Amsterdam: European Journal of Operational Research, 117(1999), str. 368-381.
79. Wang Pearl, Wäscher Gerhard: Cutting and packing, editorial. Amsterdam: European Journal of Operational Research, 141(2002), str. 239-240.
80. Wäscher Gerhard: An LP-based approach to cutting stock problems with multiple objectives. Amsterdam: European Journal of Operational Research, 44(1990), str. 175-184.
81. Wäscher Gerhard, Gau Thomas: Generating Almost Optimal Solutions for the Integer One-dimensional Cutting Stock Problem. Working Paper No. 94/06, Institut für Wirtschaftswissenschaften, Technische Universität Braunschweig, 1994.
82. Westerlund Tapio, Harjunkoski Iiro, Isaksson Johnny: Solving a Production Optimization Problem in a Paper-Converting Mill with MILP. Oxford: Computers & Chemical Engineering, 22(1998), str. 563-570.
83. Winston Wayne: Operations research: applications and algorithms. Belmont: Duxbury Press, 1993. 1318 str.
84. Yuen Boon, Richardson Ken: Establishing the optimality of sequencing heuristics for cutting stock problems. Amsterdam: European Journal of Operational Research, 84(1995), str. 590-598.
85. Zak Eugene: Row and column generation technique for a multistage cutting stock problem. New York: Computers & Operations Research, 29(2002), str. 1143-1156.

Viri

1. Decision trees. [URL: http://cognet.mit.edu/MITECS/Entry/utgoff], MIT Encyclopedia of Cognitive Science, 3. 9. 2002.
2. NP-complete. [URL: http://c2.com/cgi/wiki?NpComplete], 4. 9. 2002

SLOVAR TUJIH IZRAZOV

aspiration level – iskanje sprejemljive rešitve
branch & bound – razveji & omeji
branching – razvejevanje
candidate solution – možna rešitev
cost complexity – stroškovna kompleksnost
cut-off – prekinitev iteracije
cutting stock problem – problem razreza
decision tree – odločitveno drevo
ending condidions – končni pogoji
fathomed node – prečrtan vozel
feasible point – možna rešitev
greedy algorithm – požrešni algoritem
implicit enumeration – implicitno oštevilčenje
information gain – pridobitev v informacijski količini
item oriented – posamično obravnavanje
killed node – prečrtan vozel
knapsack – nahrbtnik
leaf – list
LP relaxation – linearna relaksacija
Minimum Description Length – minimalna potrebna opisna dolžina
mixed integer programming – reševanje mešanega celoštevilčnega problema
node – vozel
NP complete – NP poln
order length – dolžina naročila
overfitting – preveliko prilagajanje odločitvenega drevesa šumu v podatkih
overgrading – razrez iz boljšega materiala, kot je zahtevano
parent node – predhodni vozel
pattern oriented - metoda vzorcev
postpruning – naknadno klestenje
prepruning – predhodno klestenje
problem reduction – razdelitev problema
pruning – poenostavljanje
repeated exhaustion reduction – ponavljanje vzorca
sampling – vzorčenje
search tree – iskalno drevo
Sequential Heuristic Procedure – zaporedna hevristična procedura

state-space search – preiskava grafa
stock length – dolžina palice
stopping – predhodno klestenje
threshold – prag
training object – učni objekt
training set – učna množica
trim loss – izguba materiala
upper bound – zgornja meja za izgubo