HEGESZTÉSSEL KOMBINÁLT CSŐVÁGÁSI FELADAT[1]

ÁGOSTON KOLOS CSABA – NYÍRI JÁNOS
Budapesti Corvinus Egyetem

A vágási problémát (angolul cutting stock) sokan és mélyrehatóan tanulmányozták. Az alapfeladatot az évek során többféle irányban általánosították. Ebben a cikkben mi is bemutatjuk az egydimenziós probléma egy lehetséges továbbfejlesztését. A mi esetünkben (iparági szabványnak megfelelően) engedélyezett a toldás (hegesztés), de egy egység csak maximum két darabból rakható össze. Bemutatunk egy algoritmust, amely a problémát vegyes egészértékű LP feladatok sorozataként oldja meg. Az algoritmus viszonylag gyorsan lefut, és skálázható.

1 Bevezetés

A vágási problémát először Kantorovics (1939) fogalmazta meg. Később ilyen feladatokkal Gillmore és Gomory (1961) is részletesen foglalkozott, átfogó elemzéssel támogatva a megoldási folyamatokat. Cikkükben a problémát egészértékű LP feladatként fogalmazták meg, amelynek megoldásához vágási tervekre (eredeti szóhasználatban: activity) volt szükség, és az egészértékű programozási feladat megmondta, hogy melyik vágási tervet hányszor kell alkalmazni. Hozzájárulásuk legfontosabb eredménye az volt, hogy olyan eljárást mutattak fel, amelyhez nem szükséges az összes vágási terv előzetes létrehozása, ugyanis az algoritmus a lényeges vágásokat menet közben generálja (egy ún. oszlopgenerálási módszer segítségével). Ugyanakkor az algoritmus nem feltétlenül generál egészértékű megoldást is egyben.

Az eredeti feladat lényegében egydimenziós feladat: csövek vagy olyan szalagok vágásához lehetett felhasználni, ahol a szélesség adott. A gyakorlatban előforduló problémák azonban az eredeti modell kiterjesztését motiválták. Legjellemzőbb kiterjesztés a dimenziószám növelése volt: vizsgáltak két- és többdimenziós problémákat. A többdimenziós problémák esetén lényeges kérdés, hogy milyen vágások engedélyezettek: egyszerűbb esetben a vágásoknak végig kell haladniuk az anyagban (pl. üveget nem szokás félig levágni), ezekre a szakirodalomban a 'guillotine' probléma megnevezés honosodott meg. Textíliák esetében viszont nem jelent problémát, ha a vágás nem ér végig az anyag teljes szélességén. Operációkutatási szempontból ez a kétféle vágási technika eltérő kezelést implikál.

A vágási problémákról összefoglalót ad Cheng et al (1994). Ipari alkalmazásról lásd például Stadtler (1990).

[1]Beérkezett: 2016. április 14. E-mail: kolos.agoston@uni-corvinus.hu.

Napjainkban elterjedtek az ún. Sprinkler tűzvédelmi rendszerek. A rendszer tervezésének, kivitelezésének és karbantartásának specifikációját a MSZ EN 12845:2004+ A2:2009 számú szabvány írja le. Egy adott megrendelés esetén az előzetesen méretre vágott csöveket a helyszínen szerelik össze. A feladat érdekességét az adja, hogy a szabvány szerint ezeket a csöveket lehet hegeszteni, de csövenként maximum egyszer.

A 2. fejezetben definiáljuk a problémát, a 3. fejezetben bemutatjuk, hogy hogyan lehet a hegesztéssel kombinált feladatot visszavezetni az alapfeladatra, ugyanakkor felhívjuk a figyelmet a visszavezetés korlátaira is. A 4. fejezetben felírjuk a hegesztéssel kombinált vágási problémát vegyes egészértékű LP feladatként, amely futási ideje csillagászati mértékű is lehet. Az 5. fejezetben megadunk egy közelítő eljárást, amely elfogadható idő alatt lefut. Numerikus futtatások eredményei a 6. fejezetben szerepelnek, melyet az összefoglalás követ.

2 A hegesztéssel kombinált vágási feladat ismertetése

Adott egy L^d lista, amely a d átmérőhöz tartozó levágandó csöveket tartalmazza. Mivel különböző átmérőjű csöveket nem lehet összehegeszteni, így minden átmérőhöz külön feladat tartozik, ezért a d index nem feltétlenül jelenik meg a későbbiekben, kivéve a paraméterekben. A levágandó csövek hossza 0 és 12 000 mm között változhat, amelyet l_i jelöl ($i \in L^d$). Iparági szabvány, hogy a csöveket lehet hegeszteni, de darabonként maximum egyszer. A hegesztés költsége w^d —amely függhet az átmérőtől— az üzletmenet során változhat. Ha sok a megrendelés, akkor magas költséggel számolhatunk, mert a megnövekedett munkaráfordítás miatt más megrendeléseket vissza kell utasítani, így a hegesztés rezsiköltségéhez viszonylag nagy alternatíva költség is járul. Ha kevés a megrendelés, akkor csak a viszonylag alacsony rezsiköltséget kell figyelembe venni.

A csöveket 6 000 mm hosszú alapanyagokból kell méretre vágni, majd szükség esetén a ledarabolt csöveket összehegeszteni. Az egyértelmű szóhasználat miatt hívjuk a továbbiakban az alapanyagként szolgáló 'csövet' rúdnak, míg a listában szereplőt csőnek, habár ez nem teljesen szabatos. A rudak ára természetesen függ az átmérőtől: p^d.

A kérdés az, hogyan lehetne elvégezni egy L^d listához tartozó csövek méretre vágását és hegesztését úgy, hogy az összköltség (nyersanyagár plusz hegesztési költség) minimális legyen? A nyersanyagárat az elhasznált rudak darabszáma alapján kell kalkulálni, a keletkező hulladékot nem lehet felhasználni egy másik megrendeléshez.

3 Visszavezetési kísérlet

Első lehetőségként vizsgáljuk meg, hogy vissza tudjuk-e vezetni a feladatot hegesztés nélküli darabolási feladatra!

Gillmore és Gomory (1961) által megadott algoritmus nem kezeli a hegesztéseket; az alapanyag hosszúsága viszont többféle is lehet. A mi esetünkben az alapanyag hossza 6 000 mm, de egy hegesztéssel készíthetünk egy 12 000 mm hosszú rudat, melynek ára $2p^d + w^d$; két hegesztéssel készíthetünk egy 18 000 mm hosszú rudat, amelynek ára $3p^d + 2w^d$ ... Tehát úgy tűnik, hogy visszavezettük a feladatot hegesztés nélküli vágási feladatra.

De ez a visszavezetés nem tökéletes. Legyen pl. 3 db 10 000 mm hosszú leszabandó cső. Ekkor Gillmore és Gomory (1961) algoritmusa (vagy valamely továbbfejlesztett változata) le fogja darabolni ezt egy 30 000 mm hosszú rúdból vágási hulladék nélkül. Viszont a 30 000 mm hosszú rúd elkészítéséhez 4 hegesztés kell, a 3 csövet pedig maximum 3-szor lehet hegeszteni (csövenként egyszer). Az ilyen jellegű problémákat megnyugtató módon nem tudjuk kezelni a visszavezetés során.

További probléma, hogy nagy teljesítményű (szabad felhasználású) implementációja nem érhető el a darabolási feladatnak.

Lehetőség lenne az oszlopgeneráció módosítására is, de ez sem feltétlenül célravezető, hiszen a toldás lehetősége miatt a megoldandó feladat mérete megnő, ami akár nagyon nagy futási időt is okozhat (ugyanis egészértékű feladat megoldása szükséges hozzá).

A cikkben bemutatott algoritmus egyfajta kísérletnek is tekinthető, hogy milyen teljesítményre lehet számítani, ha eltérünk az oszlopgenerálós technikától. Az implementáció az üzleti életben hasznos segítséget jelentett, a megoldással a megrendelő elégedett volt. A bemutatásra kerülő programozási feladatok jól hasznosíthatóak egy módosított oszlopgenerálós algoritmus megalkotásakor.

4 Globális optimum

A 2. fejezetben leírt feladat optimális megoldása megadható egy vegyes egészértékű programozási feladatként. Legyen K olyan nagy szám, hogy a feladat K rúd segítségével már biztosan elvégezhető. Jelölje x_{ik}, $i \in L^d$, $k = 1 \dots K$, hogy az i levágandó cső mekkora részét vágjuk le a k rúdból. Jelölje r_k bináris változó, hogy a k rudat felhasználtuk-e (van-e olyan i index, amelyre x_{ik} pozitív). Legyenek h_{ik} bináris indikátorváltozók, hogy x_{ik} változók pozitívak-e vagy sem.

$$\sum_{k=1}^{K}\left(p^d r_k + \sum_{i\in L^d} w^d h_{ik}\right) \quad \rightarrow \quad \min$$

f. h.:

$$\begin{array}{rcll}
x_{ik} - 6\,000 h_{ik} & \leq & 0 & \forall i \in L^d, k = 1 \ldots K \\
\sum\limits_{i\in L^d} x_{ik} - 6\,000 r_k & \leq & 0 & k = 1 \ldots K \\
\sum\limits_{k=1}^{K} x_{ik} & = & l_i & \forall i \in L^d \\
\sum\limits_{i\in L^d} x_{ik} & \leq & 6\,000 & k = 1 \ldots K \\
\sum\limits_{k=1}^{K} h_{ik} & \leq & 2 & \forall i \in L^d \\
x_{ik} & \geq & 0 & \forall i \in L^d, k = 1 \ldots K \\
h_{ik} & \in & \{0;1\} & \forall i \in L^d, k = 1 \ldots K \\
r_k & \in & \{0;1\} & k = 1 \ldots K
\end{array} \tag{1}$$

Az (1) feladat megfogalmazásánál figyelni kell arra, hogy a célfüggvényben a hegesztés költségét egyszer akkor is hozzáadjuk a költséghez, ha nincs hegesztés. Tehát a kapott célfüggvény értékből le kell vonni a csövek számaszor a hegesztés költségét (de ez egy fix összeg, az optimum helyét nem változtatja meg).

Az (1) programozási feladat megoldható vegyes egészértékű solverek segítségével. A felírásban szereplő nagyszámú bináris változó miatt a szükséges idő extrém nagyra nőhet.

A numerikus számításokat egy Intel Duo 2,33 GHz processzorral, 2 GB RAM-mal rendelkező számítógépen futattuk; az operációs rendszer Windows 7 Enterprise volt, az LP solver pedig a 4.55 verziószámú GLPK.

A későbbiekben az *1. táblázatban* szereplő feladatot fogjuk szemléltetésként használni.

Azonosító	Hossz	Azonosító	Hossz
c1	4680	c17	4680
c2	4000	c18	4680
c3	4680	c19	2000
c4	7200	c20	4500
c5	4680	c21	5096
c6	5000	c22	4680
c7	5250	c23	7000
c8	4680	c24	4660
c9	4680	c25	4500
c10	6000	c26	5500
c11	5260	c27	4500
c12	7200	c28	4660
c13	5660	c29	7000
c14	4680	c30	4680
c15	4500	c31	6000
c16	4680		

1. táblázat. Szemléltető példa adatai

Ha az 1. táblázatban csak az első 5 csővel futtattuk le a modellt, és 6 rudat engedélyeztünk, akkor is 187,7 másodperc volt szükséges az optimális megoldás megtalálásához, ami elég tekintélyes idő a feladat méretéhez képest. Ha az első 6 csővel futtattuk, 1 óra (3600 másodperc) sem volt elég az optimum megtalálásához.

Érdemes figyelembe venni, hogy a feladat futási ideje jelentősen csökkenthető, ha a korlátok közé hozzávesszük a $r_{k-1} \geq r_k$ feltételeket, így már mind a két eset 1 másodpercen belül megoldhatóvá vált. A *2. táblázat* mutatja a futási időket a csövek számának növekedésével.

Csövek száma	Futási idő (sec)
5	> 1
6	> 1
7	410
8	< 3600

2. táblázat. A szemléltető példában szereplő első néhány csővel futtatott feladat futási ideje másodpercben

A teljes modell futtatásához szükséges idő beláthatatlan. Legalább 15 másodpercre volt szükség ahhoz, hogy a program az első egészértékű lehetséges megoldást megtalálja. Ennél jóval nagyobb méretű problémák esetén egy lehetséges egészértékű megoldás megtalálása várhatóan jóval több időt igényel, és a folyamat kevéssé befolyásolható. Megelégednénk egy közelítő optimummal is, ha azt viszonylag gyorsan megkaphatnánk. További elvárás, hogy az algoritmus skálázható legyen.

5 Közelítő algoritmus

Az előző fejezet alapján levonhatjuk azt a következtetést, hogy nem célszerű egyetlen nagy méretű feladatot felírni.

Vegyes egészértékű programozási feladatok tanulmányozása során jutottunk arra a megállapításra, hogy kb. 30-50 bináris változót tartalmazó feladatot tudnak gyorsan megoldani a solverek. Természetesen ez csak egy hozzávetőleges szám, ezen belül is sok függ a feladat struktúrájától. Szeretnénk olyan algoritmust létrehozni, ami akár sok vegyes egészértékű feladatot megold, de a mondott 30-50 bináris változónál nem tartalmaz többet, de ennél célravezetőbb, ha külső paraméterrel kontrolálható a bináris változók (maximális) száma. Legyen ez a paraméter *pmb*!

Az ötlet az, hogy bontsuk két fázisra az algoritmust. Az első fázisban csak a vágási hulladék minimalizálása a cél, és előállítunk sok lehetséges vágási tervet, amelyek közül a második fázisban kiválasztjuk, melyik esetén lesz az összköltség minimális.

Az első fázist is további alfázisokra bontjuk: 1/1 alfázis, 1/2 alfázis ... Az alfázisokat az különbözteti meg, hogy hány rudat tekintünk egyszerre.

5.1 1/1 alfázis

Kezdjük az 1/1 alfázissal! Adva van L^d lista, és van egy 6 000 mm hosszú rúd. Szeretnénk a levágandó csövek közül kiválasztani néhányat, hogy a legkisebb vágási hulladék keletkezzen. Mint látni fogjuk ez gyakorlatilag egy hátizsák probléma. Mivel egyetlen rúd van, a hegesztésnek nincs értelme. Egyetlen probléma, hogy az L^d listában elképzelhető, hogy több elem van, mint a *pmb* paraméter értéke. Vegyünk ki véletlenszerűen (visszatevés nélkül) maximum *pmb* elemet a L^d listából, legyen ez az I_1 lista.

Mit kezdjünk a 6 000 mm-nél hosszabb csövekkel? Ezeket a csöveket egyszer mindenképpen hegeszteni kell, ez a hegesztés nem befolyásolja az optimumot. Ezért úgy járunk el, hogy levesszük belőlük a 6 000 mm-es részt, és csak a maradékot szerepeltetjük a listában.

Az 1/1 alfázishoz tartozó feladat: jelölje d_i bináris változó, hogy kiválasztásra kerül-e az i elem a listában.

$$\begin{aligned} & \sum_{i \in I_1} l_i d_i \quad \rightarrow \quad \max \\ \text{f. h.:} \quad & \\ & \sum_{i \in I_1} l_i d_i \quad \leq \quad 6\,000 \\ & \qquad d_i \quad \in \quad \{0;1\} \qquad \forall i \in I_1 \ . \end{aligned} \tag{2}$$

A feladat optimális megoldásaként kapunk egy vágási tervet, amelynek összhossza remélhetőleg közel van a 6 000 mm-hez. Tároljuk el ezt a vágási tervet (javaslatot)! Azokat a csöveket, amelyeket ebben az iterációban levágtunk (azaz ahol az optimális megoldásban 1 érték szerepel), kivesszük az I_1 listából, és ha lehet, az eredeti L^d listából újabbakkal pótoljuk. Ha ez már nem lehetséges, akkor az I_1 lista elemszáma csökken.

A (2) feladatot annyiszor futtatjuk, amíg az I_1 lista ki nem ürül. Így az eredeti L^d listában szereplő összes csövet leszabtuk valamilyen módon. A generált vágási terveket a *3. táblázat* tartalmazza.

Ez az alfázis adott esetben kiváltható lenne pl. Gillmore és Gomory (1961) által leírt algoritmussal (ami vélhetően jobb eredményt adna), de egyrészt nem találtunk beágyazható programot, másrészt a feladat bonyolultságát a toldások lehetősége adja. Mivel ebben az alfázisban még nincs hegesztés, az itt nyert előnyök kevésbé számítanak a végső megoldásnál.

Vágás sorszáma	Csövek azonosítói	Összhossz	Hegesztések száma
v1	c10	6 000	0
v2	c31	6 000	0
v3	c2, c19	6 000	0
v4	c6, c23	6 000 (+6 000)	1
v5	c12, c30	5 880 (+6 000)	1
v6	c1, c4	5 880 (+6 000)	1
v7	c3, c29	5 680 (+6 000)	1
v8	c13,	5 660	0
v9	c26	5 500	0
v10	c11	5 260	0
v11	c7	5 250	0
v12	c21	5 096	0
v13	c9	4 680	0
v14	c8	4 680	0
v15	c14	4 680	0
v16	c16	4 680	0
v17	c17	4 680	0
v18	c18	4 680	0
v19	c22	4 680	0
v20	c5	4 680	0
v21	c28	4 660	0
v22	c24	4 660	0
v23	c20	4 500	0
v24	c25	4 500	0
v25	c27	4 500	0
v26	c15	4 500	0

3. táblázat. Az 1/1 alfázis során generált vágási tervek

5.2 1/2 alfázis

Ha készen vagyunk az 1/1 alfázissal visszatesszük az összes elemet az L^d listába. Az 1/2 alfázis esetén két 6 000 mm-es rúd áll rendelkezésre. Itt már értelmet nyer a hegesztés, de mivel két rúd van, nem kell külön kikötni, hogy csak maximum egyszer lehet hegeszteni, ez magától teljesül. Most is csak arra koncentrálunk, hogy a vágási hulladék minimális legyen, a hegesztés költségét nem vesszük figyelembe.

Kiveszünk most is véletlenszerűen egy maximum *pmb* elemet tartalmazó listát az L^d listából, és eltároljuk a I_2 listában. Legyen d_i jelentése ugyanaz, mint az 1/1 alfázisban! Jelentse x_{i1} és x_{i2} hogy az i levágandó csőhöz mennyit használunk el az 1., illetve 2. rúdból!

$$
\begin{aligned}
& \sum_{i \in I_2} l_i d_i && \rightarrow && \max && \\
\text{f. h.:} & && && && \\
& \sum_{i \in I_1} x_{i1} && \leq && 6\,000 && \\
& \sum_{i \in I_1} x_{i2} && \leq && 6\,000 && \\
& x_{i1} + x_{i2} - l_i d_i && = && 0 && \forall i \in I_2 \\
& x_{i1}, x_{i2} && \geq && 0 && \forall i \in I_2 \\
& d_i && \in && \{0;1\} && \forall i \in I_2\,.
\end{aligned}
\tag{3}
$$

Vágás sorszáma	Csövek azonosítói	Összhossz	Hegesztések száma
v27	c2, c10, c19	12 000	0
v28	c6, c29	12 000	1
v29	c3, c4	11 880	1
v30	c9, c12	11 880	1
v31	c23, c30	11 680	1
v32	c13, c31	11 660	0
v33	c11, c26	10 760	1
v34	c7, c21	10 346	1
v35	c1, c22	9 360	1
v36	c5, c18	9 360	1
v37	c14, c16	9 360	1
v38	c8, c17	9 360	1
v39	c24, c28	9 320	1
v40	c15, c25	9 000	1
v41	c20, c27	9 000	1

4. táblázat. Az 1/2 alfázis során generált vágási tervek

A (3) feladat optimális megoldása nem feltétlenül marad optimális, amint a hegesztés költségével is számolunk. Nézzük a 4. táblázatban a v33 vágást! A (3) feladat optimális megoldása esetén van egy hegesztés: az első rudat felvágjuk 5 500+500 módon, a másodikat 4 760+1 240 módon, majd az 500 mm-es és 4 760 mm-es darabot összehegesztjük. Természetesen ezt a részfeladatot meg lehetne hegesztés nélkül is oldani. Mivel a célfüggvényben nem szerepeltetjük a hegesztés költségét (ez jelentősen megnövelné a futási időt), ezért a modell számára ez a két megoldás ekvivalens. Egyrészről itt is látszik, hogy az algoritmus nem feltétlenül találja meg a globális optimumot: ez az ára a gyorsaságnak. Másrészről a jelen példa esetén látható, hogy az 1/1 alfázisban szerepel az a vágás, hogy 5 500+500 (v9 vágás) és 5 260+740 (v10 vágás), tehát ebben a konkrét esetben ez nem jelent tényleges hátrányt, mert az 1/1 alfázis v9 és v10 vágása együttesen dominálja a v33 vágást: ugyanúgy két rudat használunk fel mindkét esetben az 5 500 és az 5 260 mm-es csövek előállításához, de a v33 vágási javaslat esetén ehhez társul egy hegesztés költsége is.

5.3 1/3 alfázis

Ha készen vagyunk az 1/2 alfázissal, megint visszatesszük az összes elemet az L^d listába. Az 1/3 alfázis esetén már három 6 000 mm-es rúd áll rendelkezésre. Innentől kezdve vigyázni kell arra is, hogy egy csövet csak maximum két részből hegeszthetünk össze. Ez a tény a feladatban szereplő bináris változók számát jelentősen megnöveli.

Kiveszünk most is véletlenszerűen egy maximum *pmb* elemet tartalmazó listát az L^d listából és eltároljuk az I_3 listában. Legyen d_i és x_{ik} jelentése ugyanaz, mint az 1/2 alfázisban. Legyenek h_{ik} bináris indikátorváltozók,

hogy x_{ik} változók pozitívak-e vagy sem.

$$\begin{aligned}
&\sum_{i\in I_2} l_i d_i && \to && \max && \\
\text{f. h.:}& && && && \\
&\sum_{i\in I_3} x_{i1} && \le && 6\,000 && \\
&\sum_{i\in I_3} x_{i2} && \le && 6\,000 && \\
&\sum_{i\in I_3} x_{i3} && \le && 6\,000 && \\
&x_{i1}+x_{i2}+x_{i3}-l_i d_i && = && 0 && \forall i\in I_3 \\
&x_{ik}-6000h_{ik} && \le && 0 && k=1\ldots 3,\ \forall i\in I_3 \\
&h_{i1}+h_{i2}+h_{i3} && \le && 2 && \forall i\in I_3 \\
&x_{i1},x_{i2},x_{i3} && \ge && 0 && \forall i\in I_3 \\
&d_i,h_{i1},h_{i2},h_{i3} && \in && \{0;1\} && \forall i\in I_3
\end{aligned} \tag{4}$$

Vágás sorszáma	Csövek azonosítói	Összhossz	Futási idő (sec)
v42a	c2, c23, c29	18 000	$<1(<1)$
v43a	c19, c26, c27	18 000	$<1(<1)$
v44a	???	?	$>3\,600(<1)$

5. táblázat. A (4) feladattal generált néhány vágás futási ideje

A (4) feladat futásigénye jelentősen megnőtt, lásd *5. táblázatot*. Jó lenne olyan módon megfogalmazni a feladatot, hogy ne növekedjen a bináris változók száma. Erre akkor van lehetőségünk, ha nem vesszük figyelembe az összes lehetséges vágást, így a futási idő akár nagyságrendekkel is csökkenthető.

Bontsuk az I_3 listát két diszjunkt részre: I_3^1 és I_3^2. Mivel az I_3 listát véletlenszerűen töltöttük fel, ezért elégséges pl. ha az I_3^1 lista az I_3 lista első felét tartalmazza, az I_3^2 pedig a másodikat. A korlátozás legyen az, hogy az I_3^1 lista elemeit csak az első és második rúdból toldhatjuk össze, az I_3^2 elemeit pedig csak az első és harmadik rúdból. Ekkor nem kell külön feltételként kikötni, hogy maximum egy hegesztés lehetséges. Tehát a megoldandó feladat:

$$\begin{aligned}
&\sum_{i\in I_2} l_i d_i && \to && \max && \\
\text{f. h.:}& && && && \\
&\sum_{i\in I_3} x_{i1} && \le && 6\,000 && \\
&\sum_{i\in I_3^1} x_{i2} && \le && 6\,000 && \\
&\sum_{i\in I_3^2} x_{i3} && \le && 6\,000 && \\
&x_{i1}+x_{i2}-l_i d_i && = && 0 && \forall i\in I_3^1 \\
&x_{i1}+x_{i3}-l_i d_i && = && 0 && \forall i\in I_3^2 \\
&x_{i1},x_{i2},x_{i3} && \ge && 0 && \forall i\in I_3 \\
&d_i && \in && \{0;1\} && \forall i\in I_3\ .
\end{aligned} \tag{5}$$

Ha megkaptuk az optimális megoldást, akkor a levágott csöveket kivesszük az I_3 listából, és ha tudjuk, feltöltjük új elemekkel. Ha már nincs új elem,

akkor a lista mérete csökken. Akár így, akár úgy, az I_3^1 és I_3^2 listákat mindenképpen újra kell generálni, de mint említettük, ez lehet az I_3 lista első és második fele egyszerűen.

Megoldottuk a v44a feladathoz tartozó listát az (5) feladat segítségével is. A futási idő 1,7 másodperc volt, az optimum pedig 17 880. Látható, hogy a futási idő jelentős mértékben lecsökkent, de a vágási hulladék is nagyobb volt ebben az esetben, mint ameddig a (4) feladat megoldása során 1 óra alatt eljutottunk. Alternatív eljárás lehetne, hogy a (4) feladatot oldjuk meg és csak pár másodperces futást engedélyezünk, remélve, hogy eljutunk (egy nem túl rossz) egészértékű lehetséges megoldásig ennyi idő alatt. Mi inkább az (5) feladat mellett döntöttünk, a rövidebb futási idő miatt, de a kérdés akár további vizsgálódások tárgya is lehet.

Nézzük az (5) feladat korlátait! Legyen az I_3 listában négy 4 500 mm hosszú cső és a többi legyen 5 000 mm! Az optimális az lenne, ha két 6 000 mm-es rudat szétvágnánk $4\,500 + 1\,500$ módon, a harmadikat pedig $3\,000 + 3\,000$ módon. Így kapnánk két 4 500 mm hosszú csövet hegesztés nélkül, és kettőt $3\,000 + 1\,500$ módon, egy-egy hegesztéssel. Ha a négy 4 500 mm hosszú csőből nem 2-2 kerül az I_3^1 és I_3^2 listába (aminek valószínűsége 62,5%), akkor nem a 4 500 mm hosszú csövek kerülnek kiválasztásra, hanem három 5 000 mm-es cső.

Az I_3^1 és I_3^2 listák újra generálásakor van újabb esély, arra hogy a 2-2 eloszlás létrejöjjön. Ha ez nem következik be, akkor a 4 500 mm hosszú csöveket is csak 1 500 mm vágási hulladékkal tudjuk elkészíteni.

Végezetül a *6. táblázat* tartalmazza az 1/3 alfázisban (az (5) feladat által) generált vágásokat.

Vágás sorszáma	Csövek azonosítói	Összhossz	Hegesztések száma	Futási idő (sec)
v42	c2, c6, c19, c23	18 000	2	0,9
v43	c11, c13, c29	17 920	2	2,5
v44	c12, c30, c31	17 880	2	1,4
v45	c4, c21, c26	17 796	2	0,8
v46	c7, c10, c17	15 930	2	0,4
v47	c1, c16, c22	14 040	2	0,1
v48	c8, c14, c18	14 040	2	$> 0,1$
v49	c3, c9, c24	14 020	2	$> 0,1$
v50	c5, c15, c28	13 500	1	$> 0,1$
v51	c20, c25, c27	13 500	1	$> 0,1$

6. táblázat. Az 1/3 alfázis során generált vágási tervek

5.4 További alfázisok

Nem feltétlenül állunk meg az 1/3 alfázisnál (ez paraméter kérdése). Ha befejeztük az 1/j-1 alfázist és tovább szeretnénk menni, a követett módszer analóg: visszatesszük az összes elemet az L^d listába, majd véletlenszerűen kiválasztunk maximum *pmb* elemet az I_j listába. Az I_j listát $j-1$ diszjunkt részre osztjuk: $I_j^1 ... I_j^{j-1}$. A megoldandó vegyes egészértékű programozási

feladat:

$$\begin{array}{rcll} \sum\limits_{i\in I_2} l_i d_i & \rightarrow & \max & \\ \text{f. h.:} & & & \\ \sum\limits_{i\in I_j} x_{i1} & \leq & 6\,000 & \\ \sum\limits_{i\in I_j^m} x_{i(m+1)} & \leq & 6\,000 & m=1\ldots j-1 \\ x_{i1}+x_{i(m+1)}-l_i d_i & = & 0 & m=1\ldots j-1, \quad \forall i\in I_j^m \\ x_{i1}, x_{i(m+1)} & \geq & 0 & m=1\ldots j-1, \quad \forall i\in I_j^m \\ d_i & \in & \{0;1\} & \forall i\in I_j\ . \end{array} \tag{6}$$

Az optimális megoldásban szereplő csövekhez tartozó indexeket kivesszük az I_j listából, és ha van még cső az L^d listában, akkor a helyükre beteszünk újakat, ha nincs; akkor csökken a lista mérete. Újradefiniáljuk a $I_j^1 ... I_j^{j-1}$ listákat és lefuttatjuk megint a (6) vegyes egészértékű programozási feladatot.

A *7. táblázatban* bemutatjuk az 1/4 alfázishoz tartozó vágásokat, és úgy döntünk, hogy a példánkban ezzel véget is ér az első fázis.

Vágás sorszáma	Csövek azonosítói	Összhossz	Hegesztések száma	Futási idő (sec)
v52	c10, c15, c19, c25, c29	24 000	2	2,1
v53	c4, c12, c21, c27	23 996	3	2,8
v54	c1, c2, c18, c20, c31	23 860	3	1,3
v55	c7, c11, c13, c23	23 170	3	0,4
v56	c5, c6, c17, c26	19 860	2	0,2
v57	c3, c8, c16, c30	18 720	2	$> 0,1$
v58	c9, c14, c22, c24	18 700	2	$> 0,1$
v59	c28	4 660	0	$> 0,1$

7. táblázat. Az 1/4 alfázis során generált vágási tervek

5.5 2. fázis

Az alfázisok során keletkezett J vágási ajánlat. A második fázisban ki kell válogatni közülük azokat, amelyek összköltsége a legkisebb. A b_{ij} paraméter értéke legyen 1, ha az i cső a j vágásban szerepel ($j = 1 \ldots J$); 0, ha nem. Jelölje y_j bináris változó, hogy a j. vágási javaslat szerepel-e a végső vágási tervben. Legyen c_j a j. vágás költsége. A vágás költsége megkapható, ha összeadjuk a vágáshoz elhasznált rudak (anyag)költségét, és a hegesztések költségét. A második fázisban megoldandó feladat:

$$\begin{array}{rcll} \sum\limits_{j=1}^{J} c_j y_j & \rightarrow & \min & \\ \text{f. h.:} & & & \\ \sum\limits_{j=1}^{J} b_{ij} y_j & \geq & 1 & \forall i\in I^d \\ y_j & \in & \{0;1\} & j=1\ldots J\ . \end{array} \tag{7}$$

Ha egy rúd ára 4200 Ft és a hegesztés költsége pedig 80 Ft, akkor a következő vágások kerültek be a végső vágási tervbe: v4, v5, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v19, v20, v22, v25, v29, v52, v54 v59. Így összességében 29 rudat használtunk fel, és 8 hegesztésre volt szükség. Ha a hegesztés költsége megváltozik 2000 Ft-ra, akkor már a végső vágási tervhez 30 rúdra volt szükség, de csak 4 hegesztésre.

5.6 Skálázási lehetőségek

Az algoritmus skálázható a megrendelői igényeknek és a feladat méretének megfelelően. Eddig bemutatásra került a *pmb* paraméter, ami a maximális bináris változók számát mutatja az első fázis feladataiban. Minél nagyobb a paraméter értéke, annál nagyobb a lista, így több lehetőséget tud egyszerre figyelembe venni, és így vélhetőleg kisebb hulladékkal tud vágási javaslatokat előállítani. Ugyanakkor a paraméter növekedésével jelentősen megnőhet a futási idő. Futtatási tapasztalok alapján a paramétert 30 és 50 között érdemes választani.

A másik lehetőség, hogy hány alfázist engedélyezünk az első fázisban. 2 alfázist legalább érdemes, mert csak a 2. alfázisban jelennek meg a hegesztések. Az első két alfázis viszonylag gyorsan lefut, a 3. alfázistól megnőhet a futási idő. 5-6 alfázisnál többet nem érdemes választani.

Felfigyelhetünk arra, hogy sokszor egy későbbi alfázisban generált vágási terv előáll több korábbi alfázisban generált vágási tervek összegeként. Például az v27 vágási terv megegyezik az v1 és v3 vágási terv összegével. Ha bizonyos csöveket sikerült jól összepárosítani —azaz kicsi a vágási hulladék—, akkor ezeket a csöveket már nem feltétlenül tesszük vissza a következő alfázis elején a listába. Tehát definiálunk egy küszöbértéket, és ha egy vágási javaslat esetén a hulladék mértéke ennél kisebb, akkor a vágási javaslathoz tartozó csöveket már nem tesszük vissza az L^d listába a következő alfázistól. Így elérhetjük, hogy a 3., 4., stb. alfázisokban —ahol hosszabb ideig is eltarthat egy feladat megoldása—, már kevesebb cső van, így összességében kevesebb feladatot kell megoldani.

Természetesen ennek a megoldásnak is ára van. Legyen pl. 3 db 2 000 mm hosszú levágandó cső és 3 db 4 000 mm hosszú. Az optimális megoldás könnyen kitalálható: a 6 000 mm-es rudakat szét kell vágni 4 000 + 2 000 módon. De az algoritmus lehet, hogy az 1/1 fázisban kiválasztja a 3 db 2 000 mm-es csövet, 0 vágási hulladékkal. Ha ezeket a csöveket nem tesszük vissza, akkor az optimális megoldást a további alfázisok sem képesek megtalálni. Tehát még a 0 vágási hulladékú javaslatokhoz tartozó csöveket is érdemes lehet visszatenni a listába.

Összességében elmondható, hogy minél kisebb a vágási hulladékhoz tartozó küszöbérték, valószínűsíthetően annál kisebb összköltségű lesz a végső vágási terv, de annál hosszabb ideig tart az algoritmus futása. Az algoritmus futása közben érdemes kis értékre állítani a vágási hulladékot, pl.: 150 mm. A konkrét érték természetesen a nyersanyag árának és a vágás költségének arányától is függ. Minél nagyobb (relatíve) a hegesztés költsége, annál na-

gyobb lehet a vágási hulladékhoz a küszöbszám.

Az utolsó skálázási lehetőség, hogy hány másodperces futási időt engedélyezünk az alfázisokban egy-egy programozási feladat megoldásához. Az algoritmus működőképes marad, ha a megoldó nem találja meg az optimális megoldást, csak egy egészértékű lehetséges megoldást. A 6. és 7. táblázatok adatai mutatják, hogy a futási idők jellemzően kicsik, ugyanakkor ritkán nagyon hosszú futási idők is adódhatnak. Tapasztalatok szerint 1 másodperc alatt mindig volt lehetséges egészértékű megoldás. 10 másodpercnél általában nem érdemes hosszabb futási időt engedélyezni, hacsak nem nagyon speciális a megoldandó feladat.

6 Numerikus eredmények

Az előző szakaszban bemutatott algoritmust implementáltuk, hogy az algoritmust numerikusan is tesztelni tudjuk.

Az előző pont során bemutatott példát futtattuk a következő paraméterekkel: a nyersanyagár 4200 Ft rudanként, a hegesztés költsége 80 Ft, a bináris változók maximális száma 31, az első fázisban 5 alfázis engedélyezett, a vágási hulladékhoz 150 mm a küszöbszám és maximum 5 másodperc engedélyezett egy vegyes egész értékű feladat futásához.

Az algoritmus teljes futásához 2,2 másodpercre volt szükség. A végső vágási terv 29 rudat használ fel és 6 hegesztés szükséges hozzá, tehát még jobb tervet kaptunk, mint a bemutatott példában.

Nézzünk egy nagyobb méretű feladatot. A *8. táblázat* tartalmazza a levágandó csöveket. A feladatban összesen 334 cső szerepel, ami nagyságrendi különbséget jelent a mintapéldához képest.

Cső hossza	Darabszám
6 002	43
5 116	36
4 731	45
4 434	44
4 330	6
3 800	160

8. táblázat. Numerikus példa adatai

Ha a korábbi paraméteregyüttessel futtattuk le az algoritmust, akkor 252 rudat kell vásárolni és 183 hegesztés szükséges hozzá, ami 1 073 040 Ft összköltséget jelent. Az algoritmus futási ideje 18 perc körül volt, ami, bár jelentős növekedés az előzőhöz képest, üzletileg még vállalható.

Itt használhatjuk ki az algoritmus skálázhatóságát. Ha szükséges, akkor az algoritmus rövidebb idő alatt is futtatható. A korábbi 5 másodperc helyett először csak 3 másodpercet, majd 1 másodpercet engedélyeztünk egy vegyes egészértékű programozási feladat futásához. A futási idő így először 13 percre, majd 5 percre rövidült. A konkrét esetben sem a felhasznált rudak száma, sem a hegesztések száma nem változott meg, de ez a megfigyelés nyilván nem általánosítható. Az üzleti feladatok során —főleg a nagyméretűeknél— 1

másodperc futási idővel szoktuk futtatni az algoritmust, a fejezet további részében is ezt az értéket használjuk.

Ha tovább szeretnénk rövidíteni a futási időt, akkor lehetőségünk van az alfázisok számának csökkentésére. Megváltoztattuk előbb 4-re, majd 3-ra az engedélyezett alfázisok számát. A futási idő lecsökkent 4 és fél percre, majd 3 és fél percre (208 másodperc). 4 alfázis esetén az elhasznált rudak száma nem változott, de a hegesztések száma megnövekedett eggyel, ami 80 forintos összköltség emelkedést eredményezett, ami 1 tized ezreléknek felel meg. 3 alfázis esetén az elhasznált rudak száma hárommal növekedett (255-re), a hegesztéseké pedig tizenhattal csökkent (167-re), ami a kiindulási állapothoz képest 11 320 ft. költségnövekedést okoz (kb. 1%).

Alternatív lehetőség, hogy nem az alfázisok számát, hanem a hulladékhoz használt küszöbértéket emeljük meg (így kevesebb cső kerül a későbbi alfázisokba és kevesebb vegyes egészértékű feladatot kell megoldani). A küszöbértéket (5 alfázis engedélyezése mellett) először 300-ra, majd 450-re növeltük. Így a futási idő lecsökkent 4 és fél percre (a két eset között pár másodperc volt a különbség, természetesen a 450-es küszöbérték javára). 300-as küszöbérték mellett az algoritmus 253 rudat használt el és 177 hegesztést javasolt; az összköltség 3 720 forinttal emelkedett meg, ami 0,3%-os emelkedés a kiindulási állapothoz képest. 450 köszöbérték esetén a rudak száma és a hegesztések száma ugyanaz volt, mint 300-as küszöbérték esetén.

7 Összefoglalás

A cikkben az irodalomban ismert vágási probléma egy lehetséges továbbfejlesztését ismertettük. A megoldott vágási problémának az a specialitása, hogy engedélyezett a toldás, de darabonként maximum egyszer. A feladatot vegyes egészértékű feladatok sorozatára bontottuk: az első fázis lehetséges vágási terveket generál, ezek közül a második fázis törekszik a legalacsonyabb költségűt kiválasztani. Az algoritmus nem feltétlenül találja meg a globális optimumot, csak egy ehhez közeli megoldást ad. A felhasználó paraméterek beállításával dönthet, hogy inkább gyors, vagy inkább pontos végeredményt szeretne.

A leírt algoritmust implementáltuk, és bemutattuk, hogy a futási idők elfogadhatóak a mindennapi üzleti gyakorlatban.

Irodalom

1. Cheng, C. H., Feiring, B. R. és Cheng T. C. E. (1994): The cutting stock problem – a survey. *International Journal of Production Economics*, vol 36. Iss. 3, pp. 291–305.
2. Gilmore, P. C. és Gomory, R. E. (1961): A linear programming approach to the cutting-stock problem. *Operations Research* 9, pp. 849–859.
3. Gilmore, P. C. és Gomory, R. E. (1963): A linear programming approach to the cutting-stock problem - Part II. *Operations Research* 11, pp. 863–888

4. Kantorovics, L. V. (1939): *Mathematical methods of organizing and planning production.* Leningrad State University.
5. Stadtler, H. (1990): A one-dimensional cutting stock problem in the aluminium industry and its solution. *European Journal of Operational Research* 44, pp. 209–223.

CUTTING STOCK PROBLEM WITH THE POSSIBILITY OF WELDING

Cutting stock problem is widely studied in operations research. In this paper we discuss a possible extension of the one-dimensional cutting stock problem: welding is allowed —according to industrial standard—, however every piece can be jointed from maximum two parts. We present an algorithm which solve the problem with successive use of mixed integer LP. The running time is acceptable for business usage, and it is controlled by various parameters (i.e. decision maker can influence whether she/he would like a faster or a more exact solution).