Szerelőszalagok

Diplomamunka

Írta: Villányi Nóra

Alkalmazott matematikus szak

Témavezető:

Kovács Gergely, főiskolai docens
Operációkutatási Tanszék
Eötvös Loránd Tudományegyetem, Természettudományi Kar

Eötvös Loránd Tudományegyetem
Természettudományi Kar
2009

Tartalomjegyzék

1. Bevezetés

A mindennapi ember élete során számtalan eszközt használ. Használati tárgyaink vásárlásakor többnyire nem tartunk igényt az egyediségre. Például autó vásárlásakor nem készíttetünk egy teljesen új modellt, hanem egy meglevő kínálatból választunk. Így a vétel egyszerűbb, gyorsabb, olcsóbb és megbízhatóbb. Az ilyen termékeket tömeggyártással állítják elő, tehát gyártásuk folyamatos, így nem kell megvárni, hogy elkészüljön az új termék, hiszen már a vásárlás pillanata előtt elkészül az áru. A nagy mennyiségű előállításnak köszönhetően a termelés méretgazdaságos, valamint a vásárlói visszajelzéseknek és a belső ellenőrzéseknek köszönhetően a hibák is könnyen javíthatóak. Egy meghatározott eszközből azonban többfélét is találhatunk a piacon. A köztük lévő eltérés abból adódik, hogy ugyanarra a célra szolgáló terméket több különböző gyártó is előállít. A gyártók közötti versenyből az kerül ki győztesen, aki a hasonló minőségű termékek közül a legolcsóbbat kínálja. Ezért céljuk az előállítási költségek minimalizálása, a leggazdaságosabb gyártóberendezés felállítása.

Tömeggyártás esetén szerelőszalagot alkalmaznak. A termelést részműveletekre bontják, és az egyes műveleteket gondosan előkészítik. Az emberek már régen is tudtak több dologból egy hasznos eszközt előállítani. Tömeggyártásban a hatékonyság növelése érdekében már szabványosításra lett szükség. Ennek köszönhetően a különböző forrásokból származó részek az erre szakosodott üzemekben a szükséges megmunkálások után összeállíthatók egy kész termékké. Egy üzemben tehát a termelésirányításért felelős részlegnek az is a feladata, hogy biztosítsa a beépítendő anyagokat és alkatrészeket. Ebben a dolgozatban feltételezzük, hogy ez a feladat már megoldott, vagyis a szükséges alkatrészek rendelkezésre állnak. Vizsgálódásunk csak a minimális költségű gyártó berendezés megtervezésére irányul. A használandó modell alkalmazását egy konkrét példán követjük végig.

A második fejezet a szerelőszalaggal kapcsolatos fogalmakat és jelöléseket tartalmazza.

A harmadik fejezetben megismerkedünk a konkrét kitűzött feladattal és a megoldhatóság feltételeivel.

A negyedik fejezetben azt a viszonylag egyszerű esetet tárgyaljuk, amikor az igényt több egyforma szalag felállításával teljesítjük, míg az ötödik fejezetben kiterjesztjük a feladatot úgy, hogy több különböző szerkezetű gyártósort is alkalmazhatunk egyszerre. A hatodik fejezetben pedig azt vizsgáljuk, hogy különböző korlátozó tényezők hogyan befolyásolják a tervezést.

A példában kapott számok hosszas számolás eredményei. A dolgozathoz csatolt program e számolás programozott változata. A hetedik fejezetben e program működésének és használatának leírása található.

2. Alapfogalmak, jelölések

A szerelőszalag, mint ahogy a neve is mutatja, egy összeszerelő szalag, amely állomásokból áll. A gyártási folyamatot műveletekre bontják, és az egyes műveleteknek egy-egy állomást feleltetnek meg. A gyártási hatékonyság növelése érdekében az állomások közötti szállítást automatizált berendezések végzik. Így kevesebb a gyártási költség, kisebb a hely és munkaerő igény, illetve kevesebb kár (például törés) keletkezik az anyagkezelés közben. Modellünkben feltesszük, hogy ezt a szállítást gumiszalag vagy konvejorsor végzi. Ekkor termék az állomásokon meghatározott sorrendben halad végig, miközben minden állomást pontosan egyszer érint. A termékek az állomásokról egyszerre mozdulnak el, és ilyenkor mindegyik pontosan egy állomással lép hátrébb. A sor végén megkapjuk a kész terméket.

Egy ilyen szalag tervezéséhez ismernünk és használnunk kell a következő fogalmakat:

n – a műveletek száma,
p_α – az α művelet ideje tizedes pontossággal percben kifejezve,
T – naponta a gyártással töltött idő percben kifejezve,
d – naponta a gyártási igény.

Közvetlen előzmények és utózmányok: Világos, hogy a szerelési folyamatban az egyes műveletek csak meghatározott sorrendben követhetik egymást. Vagyis a műveleteknek lehetnek közvetlen előzményei, amelyeket az adott művelet előtt mindenképpen el kell végezni, illetve lehetnek közvetlen utózmányai, amelyeket nem lehet addig elvégezni, amíg az adott műveletet nem hajtottuk végre. Jelölésük:
E_α – az α művelet közvetlen előzményeinek halmaza,
U_α – az α művelet közvetlen utózmányainak halmaza.

Műveleti gráf: Felrajzolhatunk egy $G(N, A)$ irányított gráfot, ahol N a csúcsok, vagyis a műveletek halmaza, A pedig az élek halmaza, ahol egy i csúcsból vezet él egy j csúcsba, ha i a j-nek közvetlen előzménye.

Topológikus rendezés: Legyen $G(N, A)$ egy irányított gráf, ahol $|N| = n$. Ekkor G csúcsainak egy $(u_1, u_2, \ldots, u_n)$ rendezését topológikus rendezésnek nevezzük, ha $\nexists$ olyan $1 \leq i < j \leq n$ indexpár, amelyre $(u_j, u_i) \in A$.

Összes előzmény és utózmány: Egy α művelet összes előzménye azoknak a műveleteknek a halmaza, amelyekből a műveleti gráfban vezet irányított út az α-ba.

Hasonlóan egy α művelet összes utózmánya azoknak a műveleteknek a halmaza, amelyekbe vezet irányított út az α-ból:
E_α^* – az α művelet előzményeinek halmaza,
U_α^* – az α művelet utózmányainak halmaza.

Ciklusidő: A szerelőszalag csak akkor mozdulhat tovább, ha már minden állomás befejezte a munkát. Egy ilyen elmozdulást nevezünk ciklusnak. Minden ciklusban kapunk tehát egy új kész terméket. Két ilyen kibocsátás között eltelt időt nevezzük a szerelőszalag ciklusidejének. Jele: t.

Szalagméret: A naponta kibocsátott kész termékek száma. Jele: s, ahol

$$s = \left\lfloor \frac{T}{t} \right\rfloor.$$

Megjegyezzük, hogy a szalag első indításakor ez az érték kevesebb, hiszen ekkor a szerelőszalag üresen indul, és az első $n-1$ ciklus alatt nem bocsát ki kész terméket. Mi viszont egy ilyen gyártósort akár több éves termelésre tervezünk, és modellünkben feltesszük, hogy az első indítás kivételével minden ciklusban kapunk egy új darabot. Ezért használhatjuk ezt az általános alakot.

3. A feladat és feltételei

3.1. A kitűzött feladat

Mint már említettük, a dolgozat célja az, hogy egy meghatározott gyártási folyamathoz megadja annak a szerelőszalagnak a felépítését, amely felállításának és üzemeltetésének költsége minimális. A tervezést befolyásolja, hogy az igények állandóak vagy változékonyak, illetve az, hogy milyen nyersanyagkezelő rendszert alkalmaznak. Változékony igények esetén tervezéskor nem csak a költség minimalizálást kell szem előtt tartani, hanem a rugalmasságot is, vagyis azt, hogy a kapacitást könnyen tudjuk változtatni. Ez a rugalmasság könnyebben kivitelezhető, ha a szállításra úgynevezett AS/RS-t, vagyis automatizált raktározó és visszanyerő rendszert használnak. Konvejorsor használatával ellentétben egy AS/RS alkalmazásával elhagyható az a megszorítás, hogy a gyártás során a termékek az összes állomásról egyszerre mennek tovább a következő állomásra. Ugyanis egy AS/RS 3 fő részből áll, egy közvetlen elérésű kezelőből, egy raktározási szerkezetből és egy ellenőrző rendszerből. A ellenőrző rendszer utasításokat fogad és továbbít az állomásoktól a közvetlen elérésű kezelőnek. Ezen utasítások arra vonatkoznak, hogy a félkész terméket melyik állomásról melyikre kell elvinni. Ha esetleg a célállomás foglalt, akkor átmenetileg a raktározási szerkezeten helyezi el a munkadarabot. Ezáltal nem csak az egyszerre mozgás megszorítása hagyható el, hanem egyes állomásokból többet is felállíthatunk. Így a kiegyensúlyozottság javítása mellett a kapacitás növelése és csökkentése is egyszerűen megoldható akár egyetlen állomás hozzávételével vagy kiiktatásával. Ekkor azonban tervezéskor az AS/RS teljesítményét is figyelembe kell venni.

Azonban mint ahogy már említettük a mi modellünkben gumiszalagot vagy konvejorsort használunk az állomások közötti szállításra, és feltesszük, hogy az igények állandóak. A minimalizáláskor azt használjuk ki, hogy nem szükséges minden műveletet külön állomáshoz rendelni, hanem egyes tevékenységek összevonhatók és egy helyen elvégezhetők. Ezáltal csökken a gyártósor hely és munkaerő igénye, változik egy gyártósor felállításának és üzemeltetésének költsége, ciklusideje, illetve kibocsátási képessége. Ekkor egy szétosztást *megengedettnek* nevezünk, ha az előzmény és utózmány feltételek teljesülnek, és a leghosszabb tevékenységű állomás ideje nem haladja meg a ciklusidőt. Ahhoz, hogy megtaláljuk az ideális szerkezetet, ismernünk kell az összevonás következményeit és feltételeit, valamint az összköltség elemeit és szerkezetét.

3.2. Az összevonás következményei

Összevonásokat takarékossági okokból alkalmazunk. Az ilyen irányú hatások majd a költség szerkezet ismertetése után válnak világossá. De az egyesítésnek más következményei is adódnak. Nem feleltethetjük meg kölcsönösen egyértelműen egymásnak az állomásokat és a műveleteket. Ezért érdemes bevezetni három újabb jelölést:

k – az állomások száma,

$$x_{\alpha j} = \begin{cases} 1 & \text{ha az } \alpha \text{ műveletet a } j. \text{ állomáson végezzük el,} \\ 0 & \text{különben,} \end{cases}$$

q_j – a j. állomáshoz rendelt műveletek összideje, azaz $q_j = \sum_{\alpha=1}^{n} x_{\alpha j} p_\alpha$.

Ha minden műveletet külön állomásra helyezünk, akkor a ciklusidő lényegében a leghosszabb művelet idejével egyezik meg. Ha viszont összevonásokat alkalmazunk, akkor a ciklusidő a leghosszabb tevékenységű állomás ideje lesz:

$$t = \max\{q_j : j = 1, \ldots, k\}.$$

Itt érdemes megjegyezni, hogy két fajta összevonási részfeladatot különböztethetünk meg. Az első esetben az állomások száma, vagyis k adott, és a ciklusidőt kell minimalizálni, a második esetben a ciklusidő adott, és az állomások számát kell minimalizálni. Hogy melyik részfeladatot kell megoldani, azt a teljes feladat határozza meg.

Most pedig nézzük meg, hogy mi szükséges ahhoz, hogy a gyártási folyamat végrehajtható legyen, illetve milyen feltételei vannak a műveletek csoportosításának.

3.3. Feltételek

A 2. fejezetben definiáltuk a topológikus rendezés fogalmát. Erre azért volt szükségünk, mert a műveleteknek a szalagon topológikus sorrendben kell követniük egymást. Ha nem létezik ilyen sorrend, akkor nem lehet a szétosztást úgy végrehajtani, hogy minden előzmény és utózmány feltétel teljesüljön. Nézzük a létezés szükséges és elégséges feltételét:

Tétel: *Egy $G(N, A)$ irányított gráf csúcsainak akkor és csak akkor létezik topológikus rendezése, ha G nem tartalmaz irányított kört.*

Bizonyítás: Először tegyük fel, hogy létezik topológikus rendezés, és indexeljük a csúcsokat ennek megfelelően: $(u_1, \ldots, u_n)$. Továbbá indirekt tegyük fel, hogy a gráf tartalmaz irányított, k hosszúságú kört. Ezen kör csúcsai legyenek: $u_{i_1}, u_{i_2}, \ldots, u_{i_k}$, ahol $1 \leq i_1 < i_2 < \cdots < i_k \leq n$. A körben minden csúcsból pontosan egy él indul ki, ezért az olyan (u_{i_s}, u_{i_t}) alakú élek száma, amelyekre $1 \leq s < t \leq k$ legfeljebb $k-1$. A kör viszont összesen k élt tartalmaz, vagyis biztosan kell benne lennie egy olyan (u_{i_s}, u_{i_t}) élnek, amelyre $1 \leq t < s \leq k$. Ez viszont ellentmond annak, hogy a csúcsokat topológikus sorrendben indexeltük.

A másik irány bizonyításához tegyük fel, hogy G nem tartalmaz irányított kört, és igazoljuk indirekt, hogy van olyan csúcs, amibe nem vezet él. Tegyük fel, hogy nincs ilyen, és válasszunk egy tetszőleges u_1 csúcsot. Az ide mutató él kezdőpontja legyen u_2. Ebbe is megy él, ennek kezdőpontja legyen u_3. Mivel minden pontba vezet él, ezért ezt az eljárást végtelen lépésig folytathatjuk. G-nek viszont csak véges sok csúcsa van, így véges sok lépés után egy csúcs ismétlődni fog, ami irányított kör létezését jelentené.

Legyen most tehát u_1 egy olyan csúcs, amibe nem vezet él. Jegyezzük ezt fel, majd töröljük u_1-et is és a belőle kiinduló éleket is. Ekkor ismét egy olyan G' gráfot kaptunk, ami nem tartalmaz irányított kört, vagyis van olyan u_2 csúcsa, amibe nem vezet él G'-ben. Tehát G-ben csak u_1-ből vezethet él u_2-be. Jegyezzük fel u_2-t és töröljük a belőle kiinduló élekkel együtt. Ismételjük ezt az eljárást mindaddig, amíg az utolsó csúcsot is kitöröljük. Igazoltuk ezzel, hogy létezik a csúcsoknak topológikus rendezése, és egy ilyen rendezést meg is adtunk.

■

Ezzel megadtunk a gyártási folyamat végrehajthatóságának feltételét. Nézzük, hogy milyen feltételeket kell figyelembe venni a műveletek szétosztásakor. A most következő modellt Bowman [3] vezette be.

- Minden műveletet hozzá kell rendelni valamelyik állomáshoz, azaz

$$\sum_{j=1}^{k} x_{\alpha j} = 1 \qquad \alpha = 1, \ldots, n.$$

- Egyetlen művelet sem kerülhet korábbi állomásra, mint valamely előzménye, azaz ha az α előzménye a β-nak és β-t az i. állomáson végezzük el, akkor az α-t már valamely $j \leq i$ állomáson el kell végezni

$$\sum_{j=1}^{i} x_{\alpha j} \geq x_{\beta i} \qquad \beta = 1, \ldots, n, \quad i = 1, \ldots, k \; \forall \alpha \in E_\beta.$$

- Egyik állomáson sem lehet hosszabb a megmunkálási idő, mint a ciklusidő, azaz

$$q_j = \sum_{\alpha=1}^{n} p_\alpha x_{\alpha j} \leq t \qquad j = 1, \ldots, k.$$

  Erre a feltételre akkor van szükség, amikor adott ciklusidő mellett az állomások számának minimalizálása a cél, hiszen a másik esetben a szétosztásból származtatjuk a ciklusidőt.

Egy konkrét feladat esetén természetesen adódhatnak további feltételek, például megszabhatják a két művelet között eltelt idő minimumát vagy maximumát, de ebben a dolgozatban csak az általános esetet, illetve a konkrét példánk plusz feltételeit tárgyaljuk.

3.4. Költség szerkezet

A feltételekhez hasonlóan a költségeknél is általánosításokkal élünk. A termék előállításának teljes költsége nagyon sok részből tevődik össze. Az üzemterület megvásárlásától vagy bérleti díjától kezdve, az alkatrész beszerzésen és a gyártósor üzemeltetésen át, a kész termék elszállításáig minden a gyártó kiadását képzi. Ezek közül azonban érezzük, hogy egyesek nem befolyásolják a gyártósor tervezését (például alkatrész beszerzés), és vannak olyanok is, amelyek az összköltséget tekintve elhanyagolhatók. Ennek megfelelően az itt tárgyalt modellben a költség elemek számát négyre csökkentettük. Bár látható majd, hogy ez akár tovább csökkenthető háromra is. A szalagtervezéshez kapcsolódó elemek közül ezek nagyságrendjükkel is befolyásolják a költséget. Vegyük sorba ezen komponenseket, de előtte vezessünk be ide kapcsolódó jelöléseket:

y – a szerelőszalag élettartama években kifejezve,
sz_j – a j. állomás szerszámozási költsége,
l_c – átlagos éves munkabér,
b – a terület vételi egységára,
o – a terület fenntartásának egységára,
g – a szerelőszalag alapterülete,
m – az anyagkezelés egységára,
h – a szerelőszalag hossza méterben,
c – a szerelőszalag teljes költsége.

- T_c – **Szerszámozási költség:** A szerelőszalag felállításakor az állomások kialakításának összköltsége:

$$T_c = \sum_{j=1}^{k} sz_j \ .$$

- L_c – **Munkaerő költség:** A gyártósort kezelő emberek számának és az átlagos kezelőnkénti éves bérek szorzata. Modellünkben az egyszerűség kedvéért minden munkaállomáson egy kezelő dolgozik. Így tehát ennek az elemnek az értéke:

$$L_c = k \cdot l_c \cdot y \ .$$

- S_c – **Terület költség:** Az alapterület lineáris függvényeként fejezhető ki. A szalagtervezéskor elegendő kizárólag a szerelőszalag és a hozzá közvetlenül kapcsolódó rész alapterületét figyelembe venni, mert az ezen kívüli részek mérete független a gyártósor szerkezetétől, és ezért költsége állandó. A terület vételi árának egységköltsége függ a gyártási környezettől. Például egy átlagos gyárnál ez a költség 1100\$-1600\$/m^2, míg egy úgynevezett tiszta terem esetén az egységár 4300\$-5400\$/m^2. Emellett évente felmerülnek egyéb költségek is, mint például közművek, adók, biztosítás, amelyeket szintén az alapterület függvényében számítanak fel. Ezek a kiadások hagyományos gyártási környezet esetén évente körülbelül 30\$-40\$/m^2 tesznek ki. Összegezve:

$$S_c = g \cdot b + g \cdot o \cdot y \ .$$

- M_c – **Anyagkezelési költség:** Ez alatt a félkész termékek szerelőszalagon belüli szállítását végző berendezés felállítási költségét értjük:

$$M_c = m \cdot h \ .$$

Most már elegendő előismerettel rendelkezünk ahhoz, hogy megoldjuk a kitűzött feladatot. A leegyszerűsített feladat összegezve tehát a következő: El szeretnénk kezdeni gyártani egy terméket. Ehhez meg kell vásárolnunk a gyártás helyszínét, és installálni kell egy berendezést, amely a megfelelő alkatrészekből előállítja termékünket. Célunk, hogy az előállítás költsége minimális legyen. Ismerjük a gyártási folyamatot, az egységköltségeket, és tudjuk, hogy naponta hány darabot kell legyártani. Meg kell keresni tehát az ideális szerkezetű szerelőszalagot. Elképzelhető persze, hogy nem elég egyetlen gyártósor az igény kielégítéséhez. Ekkor felállíthatunk több azonos szerkezetű, vagy több, de különböző szerkezetű szerelőszalagot. Tekintsük először az előbbi esetet.

4. Minimalizálás azonos szerkezetű szalagok esetén

A korábbiakban már láttuk, hogy a napi kibocsátás a gyártással töltött idő és a ciklusidő hányadosa. Tervezéskor csak a ciklusidőt tudjuk befolyásolni, ezért a kibocsátás egy szerelőszalagon akkor maximális, ha a ciklusidő minimális. Ez pedig akkor teljesül, ha minden műveletet különböző állomáson végzünk el. Ekkor tehát a ciklusidő a leghosszabb művelet ideje lesz.

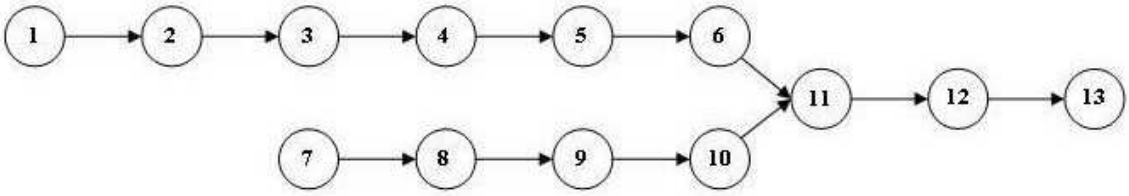
Általában viszont az így nyert kibocsátás sem fedezi az igényt. Ekkor egy újabb gyártósor felállítására van szükség. Tervezéskor ezért első lépésben el kell dönteni, hogy egyforma szalagokat alkalmazunk, vagy különbözőket. Az előbbivel a költség legalább akkora, mint az utóbbival, mert azt nem kötjük ki, hogy minden felállítandó berendezésnek különböző szerkezetűnek kell lenni, csupán megengedjük az eltérőséget. Eszerint gazdaságosabb, ha nem ragaszkodunk az azonossághoz, mégis sok alkalmazás ezt követeli meg. Ennek egyik oka, hogy itt csak egy szalagot kell a mérnököknek megtervezni, és ennek mintájára el lehet a többit is készíteni. Ezáltal olcsóbb és egyszerűbb lesz a tervezés és a kivitelezés.

Nézzük meg most részletesen, hogy hogyan válasszuk ki ebben az esetben az ideális szalagméretet. Ehhez meg kell határozni, hogy milyen méretű szalagokat tudunk előállítani, illetve ezek költségeit, majd értékelni kell az eredményeket, és kiválasztani az ideális szerkezetet.

Ezt a folyamatot végigkövetjük egy konkrét példán keresztül. A példában szereplő folyamatidők, szerszámozási költségek és az egyéb költségelemek értéke a [2]-beli 8.1-es példából származnak. [2]-ben más feltételeknek kell eleget tennie a szalgoknak. Vagyis ott egy állomásból egy szalagon belül több is lehet, a gyártási hatékonyság folyamatosan fejlődik, és más költségelemeket is figyelembe vesznek. Valamint ott ez a példa csak arra szolgál, hogy igazolják, hogy műveletek összevonásával csökkenthetők a költségek. Ennek érdekében kiszámítják a költségét annak a szalagnak, amelyben egy állomásra csak egy művelet kerül, és egy olyannak, amelyben a szomszédos műveleteket összevonják.

Ezzel szemben mi ezekre az adatokra végig vezetjük a teljes tervezési folyamatot. Nem csak szomszédos állomások összevonását engedjük meg, hanem eljárásunkban bármely két, illetve több művelet összevonható azzal a feltétellel, hogy a kialakult csoportosításban és állomáshoz rendelésben teljesülnek az előzmény és utózmány feltételek. Ez alapján előállítjuk a lehetséges szerelőszalagokat, és meghatározzuk azonos és különböző szalagokkal is azt a kombinációjukat, amivel teljesíthető az igény, valamint minimális a gyártóberendezés felállításának és üzemeltetésének költsége.

1. Példa: Egy termék elkészítéséhez 13 művelet elvégzésére van szükség. Ezek egymásra épülését az *1. ábra* mutatja. A műveleteket egy topológikus sorrendjük szerint számoztuk.



1. ábra. Műveleti gráf

Az *1. táblázat* tartalmazza a műveletek végrehajtási idejét és szerszámozási költségét.

művelet	**idő** (min)	**szerszámozási költség** ($k)
1	2, 9	128
2	3, 6	11
3	2, 2	56
4	3, 8	101
5	2, 5	75
6	3, 5	32
7	3, 5	85
8	2, 3	110
9	3, 1	70
10	5, 4	125
11	3, 5	125
12	3, 0	4
13	3, 5	50

1. táblázat. Műveleti idők és szerszámozási költségek

5 éves termelésre számítunk, naponta 18, 6 órában. Az igény 1150db/nap. Műveletek összevonásakor általában is feltehető, hogy a szerszámozási költségek additívak. Egy-egy művelet elvégzéséhez $1\,m$ hosszú munkaterület szükséges, és minden állomás után $0, 5\,m$ hosszúságú helyet kell kihagyni. Az állomások szélessége $1\,m$, és a gyártósorhoz további $1 - 1\,m$ szélességű rész van fenntartva az anyagkezelő rendszernek, a

kezelőknek, valamint karbantartásra és gyalogos közlekedésre.

Az átlagos éves munkabér 35000\$. A terület megvásárlásának és berendezésének költsége $1507\$/m^2$, és évente a fenntartás további $43\$/m^2$. Az anyagkezelő rendszer egységára pedig $1310\$/m$.

Nézzük meg az egyes költségelemek alakulását:

- **Szerszámozási költség:** Az *1. táblázat* alapján számolható.

$$T_c = \sum_{j=1}^{k} sz_j = 972000\$.$$

- **Munkaerő költség:**

$$L_c = k \cdot l_c \cdot y = k \cdot 35000\$ \cdot 5 = k \cdot 175000\$.$$

- **Terület költség:**

$$g = h \cdot (1 + 1 + 1 + 1 + 1) = 5 \cdot h,$$

$$S_c = g \cdot b + t_a \cdot o \cdot y = 5 \cdot (13 \cdot 1 + 0,5 \cdot k) \cdot 1507\$/m^2 + 5 \cdot (13 \cdot 1 + 0,5 \cdot k) \cdot 43\$/m^2 \cdot 5,$$

$$S_c = 4305\$ \cdot k + 111930\$.$$

- **Anyagkezelési költség:**

$$M_c = m \cdot h = 1310\$ \cdot (13 \cdot 1 + 0,5 \cdot k) = 655\$ \cdot k + 17030\$.$$

Összegezve ezeket azt kapjuk, hogy a teljes költség csak az állomások számától függ:

$$c = T_c + L_c + S_c + M_c = 179960\$ \cdot k + 1100960\$.$$

□

Feltételezhető, hogy az összköltség mindig olyan alakra hozható, amely csak az állomások számától függ. Ezáltal tudjuk, hogy mennyibe kerül egy k állomásból álló szerelőszalag. Viszont n művelet k állomásra való szétosztása abban az esetben, ha nincsenek előfeltételek $k^n - (k-1)^n$ féleképpen lehetséges, vagyis ennyi szalagnak lesz ugyanannyi a költsége. Az előfeltételek alkalmazásával ez a szám csökken, de még mindig sok olyan kombináció marad, amelynek költsége egyforma. Számunkra az a leggazdaságosabb, ha adott költség mellett a legtöbbet tudjuk előállítani. Azt pedig már tudjuk, hogy ehhez a ciklusidőt kell minimalizálni. Meg kell tehát nézni $\forall k \in (1, \ldots, n)$-re a hozzá tartozó minimális ciklusidőt.

4.1. Ciklusidő minimalizálás

Mivel nagyon hosszadalmas lenne megkonstruálni mind a $k^n-(k-1)^n$ lehetőséget, és mindegyikhez kiszámolni a ciklusidőt, egy leszámlálás típusú algotritmust használunk, melynek alapötlete [1]-ből származik.

Ott a feladat az, hogy rögzített ciklusidő esetén a műveleteket a lehető legkevesebb állomásra ossza szét. Ehhez kiindulásként vesz egy tetszőleges heurisztikus módszerrel előállított megengedett megoldást, majd minden lépésben ezt javítja. Legyen a mindenkori legjobb megoldásban az állomások száma $\hat{n}+1$. Tehát első lépésben $\hat{n}+1$ a heurisztikus megoldásban kapott állomásszám lesz. A következő lépésben egy olyan megengedett megoldást keresünk, amelyben az állomásszám legfeljebb $\hat{n}$.

A felhasznált algoritmus minden leszámlálás típusú algoritmushoz hasonlóan kétféle lépésből áll. Az egyik az úgynevezett *konstrukciós lépés*, a másik pedig az *ágcsere*. Az előbbi elnevezése arra utal, hogy itt a cél az előbbieknél jobb megengedett megoldás konstruálása. Ágcserét akkor alkalmazunk, ha menetközben kiderül, hogy ez nem lehetséges. Ekkor a félig kész megoldást kis mértékben megváltoztatjuk, de csak annyira, hogy ne hagyjunk ki egyetlen lehetőséget sem. Ezután folytatjuk a konstrukciót.

Ha sikerült legfeljebb $\hat{n}$ állomásra szétosztani a műveleteket, akkor tovább csökkentjük az állomásszámot, és újra alkalmazzuk az algoritmust. Ha viszont minden lehetséges csoportosítást végignéztünk, azaz már ágcserét sem tudunk alkalmazni, akkor ehhez a ciklusidőhöz az ideális szétosztás az lesz, amit az előző lépésben kaptunk. Ilyen módszerrel tehát minden t ciklusidőre megtalálhatjuk a műveletek legjobb csoportosítását.

Emlékeztetőül, a mi feladatunk az, hogy megtaláljuk a legjobb ciklusidő-állomásszám párosításokat. Használhatnánk tehát a fent leírt eljárást, és minden lehetséges ciklusidőre meghatározhatnánk a minimális állomásszámot. Ekkor tehát az állomásszámokat a $t \in \{max\{p_\alpha\}, \ldots, \sum_{\alpha=1}^{n} p_\alpha\}$ értékekhez keressük, hiszen a legrövidebb ciklusidőt akkor kapjuk, ha minden műveletet külön állomáson végzünk el, a leghosszabbat pedig akkor, ha minden műveletet egy állomáson hajtunk végre. Az előbbi esetben a ciklusidő a leghosszabb művelet ideje lesz, az utóbbi esetben pedig a műveletek összideje. Ezzel a módszerrel tehát a ciklusidő és az állomásszám felhasználásával a szalagméretekhez határozzuk meg a költségeket.

A problémát az jelenti, hogy a lehetséges t intervallum nagyon nagy is lehet. Látható, hogy ez első példánk esetében $(42,8-5,4)\cdot 10 = 374$ ciklusidőt jelent. Viszont $k \in \{1,\ldots,13\}$, hiszen itt 13 művelet van összesen. Így általánosan is

elmondható, hogy nagyon sok ciklusidőhöz ugyanaz az állomásszám fog tartozni. Tehát még ki kellene választani a t - k párosítások közül is azt, ami a leggazdaságosabb, vagyis minden k-hoz a minimális t-t. Ez nem is jelentene olyan sok többlet műveletet, viszont adott esetben a 374 ciklusidőhöz a műveletek szétosztásásnak meghatározása nagyon sok felesleges műveletet igényel. Ezért a hatékonyság növelése érdekében úgy módosítjuk az eljárást, hogy összességében kevesebbszer kelljen végrehajtani a szétosztást.

Mi inkább a költséghez határozzuk meg a maximális szalagméretet. Ezt azért tehetjük meg, mert láttuk, hogy modellünkben a költség az állomások számától függ. Tehát a lehetséges $k \in \{1, \ldots, n\}$ állomásszámokhoz meghatározzuk a minimális ciklusidőt, és ezáltal megkapjuk a szalagméretet is. Az eredeti eljárással ellentétben nem egy megengedett megoldás javításával minimalizálunk, hanem az ideális, azaz a tökéletesen kiegyensúlyozott szerelőszalag ciklusidejéből indulunk ki. A szalag akkor *tökéletesen kiegyensúlyozott*, ha minden állomás tevékenysége ugyanannyi ideig tart. Általában ekkora ciklusidővel nem oszthatók szét a műveletek, csak ettől nagyobbal. Eljárásunk tehát az ideális ciklusidőből indul ki, és ezt addig rontja, míg megengedett megoldást nem talál.

Induljunk ki tehát a k állomásszámhoz tartozó ideális ciklusidőből. k állomás esetén akkor minimális a ciklusidő, ha a szerelőszalag tökéletesen kiegyensúlyozott. Ekkor tehát

$$t = \left\lceil \frac{10 \cdot \sum_{\alpha=1}^{n} p_\alpha}{k} \right\rceil : 10.$$

A tizedesekre felfelé kerekítés használható, mivel a műveleti időket is tizedes pontossággal adtuk meg. Általában viszont nincs a műveleteknek ilyen csoportosítása. Ha tehát nem tudjuk a szétosztást ezzel a minimális ciklusidővel elkészíteni, akkor növeljük $0,1$ perccel a ciklusidőt, és ennek megfelelően ismét megpróbáljuk kialakítani az állomásokat. Ezzel a módszerrel véges sok növelés után megkapjuk a lehető legjobb felbontást az adott k-ra.

Mikor egy t-hez próbálunk csoportosítást keresni, akkor az eredeti eljárás szerint szisztematikusan haladunk a lehetőségeken egészen addig, amíg nem találunk megengedett megoldást, vagy amíg végig nem néztünk minden lehetőséget. Ehhez tehát a konstrukciós lépést és az ágcserét felváltva alkalmazzuk. Az alábbiakban ismertetjük az egyes lépések elvét, a megvalósítást a hetedik fejezet tartalmazza.

Megengedett megoldás konstruálása

Jelölje az α művelet elhelyezése esetén $l(\alpha)$ annak a legkorábbi állomásnak a sorszámát, amire α a jelenlegi konstrukció mellett helyezhető, valamint x_α annak az állomásnak a számát, amelyikhez az α műveletet rendeltük, azaz $x_\alpha := j$, ha $x_{\alpha j} = 1$. Egy topológikus sorrendben vesszük a műveleteket és egyesével megpróbáljuk elhelyezni őket. Minden egyes műveletet a lehető legkorábbi állomásra szeretnénk tenni úgy, hogy közben teljesüljön a következő két feltétel:

- Egyik művelet sem kerülhet előrébb, mint valamelyik előzménye:

$$l(\alpha) = max\{j : x_\beta = j\ ,\ \beta \in E^*_\alpha\}.$$

- Az állomáson elhelyezett műveletek összideje nem haladhatja meg a ciklusidőt, azaz csak olyan j állomás választható, amelyre

$$\sum_{\beta : x_\beta = j} p_\beta + p_\alpha \leq t,$$

ennek megfelelően:

$$x_\alpha := min\{j \in [l(\alpha), k] : t - \sum_{\beta : x_\beta = j} p_\beta \geq p_\alpha\}.$$

Ha viszont

$$\{j \in [l(\alpha), k] : t - \sum_{\beta : x_\beta = j} p_\beta \geq p_\alpha\} = \emptyset,$$

akkor a műveletek jelenlegi állomásokhoz való rendelésével a szétosztást t ciklusidővel nem lehet befejezni. Ekkor kerül sor az ágcsere lépésre. Ha az ágcsere sikeresen elvégezhető, akkor a vele kapott hozzárendelést folytatjuk a konstrukciós lépéssel.

Ágcsere

Az ágcsere lényegében azt jelenti, hogy a már állomáshoz rendelt műveletek közül a legutolsó olyat, amit a jelenlegi állomásától hátrébb is tudtunk volna rakni, azt a pillanatnyi állomása utáni első lehetséges állomásra rakjuk, és az ezt követő összes művelet állomáshoz való rendelését töröljük. Ha egyetlen művelet sem helyezhető hátrébb, akkor minden lehetséges esetet megvizsgáltunk, és nem sikerült a t ciklusidővel megvalósítani a szétosztást. Ekkor kerül tehát sor t növelésére.

Mint említettük, az algoritmus részletes leírása a 7. fejezetben található.

Eredményül tehát megkapjuk egy k állomásszámhoz a ciklusidőt és ezáltal a szalagméretet, valamint a műveletek csoportosítását is. Ezzel a módszerrel megkaphatjuk minden $k \in \{1, \ldots, n\}$ esetén a maximális szalagméreteket. A fenti példa esetén a különböző szalagok költségét, ciklusidejét, méretét és szerkezetét a *2. táblázat* tartalmazza. Ezután már csak az a dolgunk, hogy kiválasszuk a számunkra legmegfelelőbb szerkezetet, és meghatározzuk, hogy hány darabra van belőle szükség.

Állomások	**Ciklusidő** (perc)	**Szalagméret** (db/nap)	**Összköltség** ($)
1	42,8	26	1.280.920
2	21,4	52	1.460.880
3	14,8	75	1.640.840
4	11,4	97	1.820.800
5	9,5	117	2.000.760
6	8	139	2.180.720
7	7	159	2.360.680
8	6,5	171	2.540.640
9	6	186	2.720.600
10	5,8	192	2.900.560
11	5,4	206	3.080.520
12	5,4	206	3.260.480
13	5,4	206	3.440.440

2. táblázat. Ciklusidők, szalagméretek és összköltségek

Mivel a 10. művelet lényegesen hosszabb, mint a többi, a 11 és 12 állomásra bontás esetén a ciklusidő ugyanannyi, mint amikor minden műveletet külön állomásra tettünk. Ezért a napi 206 darabot 3 különböző árú szerelőszalaggal is elérhetjük. Természetesen itt a legolcsóbbat, vagyis a 11 állomást tartalmazót választjuk. A másik kettőt a továbbiakban el is hagyjuk.

4.2. Szalagválasztás

Mivel ebben a szakaszban csak egyfajta szalagot kell választanunk, és azt sokszorosítani, egyszerűen meghatározhatjuk, hogy melyik összetétel lesz a leggazdaságosabb. Kiszámoljuk, hogy az egyes méretekből hány darab fedezi a napi igényt, és így mennyi lesz a teljes igény kielégítésének a költsége. Példánk esetében az itt kapott értékeket a *3. táblázat* mutatja.

k	**Szalag méret**(db/nap)	**Szalag költség**(m $)	**Szalagok száma**(db)	**Teljes költség**(m $)	**Többlet** (db/nap)
1	26	1,28	45	**57,61**	20
2	52	1,46	23	**33,6**	46
3	75	1,64	16	**26,25**	50
4	97	1,82	12	**21,85**	14
5	117	2	10	**20**	20
6	139	2,18	9	**19,63**	101
7	159	2,36	8	**18,89**	122
8	171	2,54	7	**17,78**	47
9	186	2,72	7	**19,04**	152
10	192	2,9	6	***17,4***	2
11	206	3,08	6	**18,48**	86

3. táblázat. Teljes költségek

A *3. táblázatból* leolvasható, hogy 1150-es igény esetén 6 db 10 állomásból álló szalag felállítása a leggazdaságosabb, és ekkor naponta csak 2-vel több termék készül, mint szükséges lenne. Ez pedig a következő művelet szétosztással érhető el.

Állomás	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
Műveletek	7	1, 8	2, 3	4	5, 9	6	10	11	12	13

5. Minimalizálás különböző szerkezetű szalagok esetén

Mikor az igény kielégítésekor megengedjük különböző méretű szalagok egyidejű alkalmazását, akkor az ideális szalagok kiválasztását tekinthetjük úgy, mint egy matematikai programozási feladatot. Tegyük fel, hogy l lehetséges szerelőszalagot tudunk készíteni. Legyen ekkor $i = 1, 2, \ldots, l$ esetén:

s_i – az i. szalag mérete,
c_i – az i. szalag költsége,
x_i – az s_i méretű szalagok száma,
d – az igény.

A kiválasztási probléma:

$$\sum_{i=1}^{l} s_i x_i \geq d \qquad i = 1, \ldots, l,$$

$$x_i \geq 0,$$

$$x_i \in \mathbb{Z},$$

$$\sum_{i=1}^{l} c_i x_i \to min$$

Ez a probléma hátizsák feladatként is ismert, és megoldható dinamikus programozási technikával. A következőkben erre adunk egy eljárást, melyben először két függvény értékeit határozzuk meg, majd ezekből leolvassunk az ideális összetételt.

5.1. Függvények

A megoldáshoz szükségünk lesz tehát két segédfüggvényre, f-re és h-ra:

- $f_i(x)$ a minimális költség egy x igényre, ha az $1, \ldots, i$ szalagokat használhatjuk,
- $\{h_i(x) = j\}$ azt jelenti, hogy az $1, \ldots, i$ szalagok használata esetén egy s_j méretű szalagot kell az eddigiekhez hozzávenni ahhoz, hogy teljesítsük az x igényt.

f értékét rekurzívan számolhatjuk:

$$f_i(x) = min\{f_{i-1}(x), f_i(x - s_i) + c_i\}.$$

Ha a 2. tagban $x - s_i$ negatív, akkor $f_i(x - s_i) = 0$, hiszen ekkor az i. szalag önmagában is képes fedezni az igényt.

f értékével egyidejűleg megkapjuk h következő értékét is. Ha $f_i(x - s_i) + c_i$ a kisebb, akkor egy s_i méretű szalag felállítása gazdaságosabb, ezért $h_i(x) = i$. Ha az $f_{i-1}(x)$ kisebb, akkor viszont az i. szalagot nem vesszük figyelembe, vagyis $h_i(x) = h_{i-1}(x)$.
Nézzük a függvényértékek meghatározására szolgáló algoritmust.

Algoritmus

1. Meghatározzuk az $\{s_i\}$ szalagméreteket.

2. Kiszámoljuk a $\{c_i\}$ költségeket.

3. Legyen $f_0(0) = h_0(0) = 0$ és $i = 0$.

4. $i = i + 1$.

5. A rekurzív képlet alapján meghatározzuk minden 0 és d közötti x igényre a megfelelő $f_i(x)$ és $h_i(x)$ értékeket.

6. Ha $i = l$, vagyis a maximális méretű szalagot is figyelembe vettük, akkor kész vagyunk. Különben tekintjük a következő méretű szalagot is, és azzal is meghatározzuk az f és h értékeket, vagyis visszalépünk a 4. lépésre.

Az algoritmus első két lépését az előző fejezetben ismertetett módon hajtjuk végre. A többi lépés megértését segíti a [2] 8.3-as példája. Az 1. példa esetében nem részletezzük az algoritmus menetét, hiszen ott a lehetséges szalagméretek és igény miatt rendkívül sok függvényértéket kell kiszámítani, és az eredményeket két 11×1150-es táblázatban foglalhatnánk össze. A következő példában szereplő speciális szalagméreteknek és igénynek köszönhetően a függvényértékek jól ábrázolhatók.

2. Példa Tegyük fel, hogy az igény 900, és 3 szalagméret közül választhatunk, és ismerjük ezek költségét is:

	s_i	c_i
$i = 1$	100	20
$i = 2$	200	30
$i = 3$	300	50

A számítások eredményét a következő két táblázatban foglalhatjuk össze. A *4. táblázat* mutatja az $\{f_i(x)\}$ értékeit és az *5. táblázat* a $\{h_i(x)\}$ értékeit $x = 100, 200, \ldots, 900$ esetén. Általános esetben a számítást minden 0 és 900 közötti x-re elvégezzük, itt viszont a szalagok kombinációival csak ezeket az x értékeket állíthatjuk elő. Ezért elég erre a 9 mennyiségre elvégezni a számításokat.

x	100	200	300	400	50	600	700	800	900
$i = 1$	20	40	60	80	100	120	140	160	180
$i = 2$	20	30	50	60	80	90	110	120	140
$i = 3$	20	30	50	50	70	80	100	100	120

4. táblázat. Az $f_i(x)$ költségek

x	100	200	300	400	50	600	700	800	900
$i = 1$	1	1	1	1	1	1	1	1	1
$i = 2$	1	2	2	2	2	2	2	2	2
$i = 3$	1	2	2	3	3	3	3	3	3

5. táblázat. A $h_i(x)$ indexek

A számítás logikájának megértéséhez nézzük, az $i = 3$ és $x = 700$ esetet. El kell dönteni, hogy a következő két lehetőség közül melyik gazdaságosabb:

- Csak az 1. és 2. szalagot használjuk és a 3.-at nem. Ekkor a költség:

$$f_2(700) = 110.$$

- Kiválaszthatjuk a 3. szalagot is. Ekkor a költség:

$$f_3(700 - s_3) + c_3 = f_3(300) + 50 = 50 + 50 = 100.$$

Azt kaptuk tehát, hogy a második esetben lesz kisebb az érték, ezért $f_3(700) = 100$ és $h_3(700) = 3$.

□

5.2. Visszaléptetés

A függvényértékek meghatározása után már csak az a dolgunk, hogy leolvassuk a szalagok ideális kombinációját és a termelés költségét. Ezt h, illetve f visszafelé léptetésével valósítjuk meg. Kövessük végig a második példán ennek logikáját. Mivel $h_3(900) = 3$, ezért egy $s_3 = 400$ méretű szalagra lesz szükség. Ezen felül még megmaradt egy $900 - 400 = 500$ nagyságú igény. $h_3(500) = 3$, ezért egy újabb $s_3 = 400$ méretű szalag szükséges. Végül még 100 darab legyártása szükséges, $h_3(100) = 1$ miatt ehhez egy $s_1 = 100$ méretű szalagot veszünk hozzá az eddigiekhez. Azt kaptuk tehát, hogy a 2. példában a 900 igény teljesítésének leggazdaságosabb módja, ha 2 darab 400-as és egy darab 100-as szalagot állítunk fel az üzemben. Ekkor az összköltség: $2 \cdot c_3 + c_1 = 2 \cdot 50 + 20 = 120$, ami éppen $f_3(900)$ értéke.

Az eljárás segítségével tehát meg tudjuk határozni a szerelőszalagok ideális összetételét abban az esetben, amikor különböző méretű szalagokat is használhatunk az igény teljesítéséhez. Az előző fejezetben pedig azt az esetet oldottuk meg, amikor csak egyforma szalagokat használhatunk. Hasonlítsuk most össze e két esetet.

5.3. Összehasonlítás

Nézzük meg az 1. példa esetén, hogy mekkora az eltérés a költségek, illetve a többlettermelés között a két esetben.

Emlékeztetőül az igény $1150\,db/nap$ és a *2. táblázat* mutatja a lehetséges szalagméreteket és ezek költségeit. Egyforma szalagok használatakor azt kaptuk, hogy az $s_{10} = 192$ méretűből van szükség 6 darabra, az összköltség ekkor 17.403.360$, és 2-vel gyártunk többet naponta, mint amennyi szükséges.

A fent leírt eljárás alkalmazásával határozhatjuk meg ezeket az adatokat különböző méretű szalagok használatakor. Eszerint ekkor az $s_9 = 186$ méretűből négyet és az $s_{11} = 206$ méretűből kettőt kell felállítani. Az összköltség 17.043.440$ és a többlettermelés 6 darab naponta.

Látható, hogy ebben a példában a két eset között a többlettermelés beli különbség elhanyagolható. Viszont különböző méretű szerelőszalagok használatával a költség 17.403.360$ − 17.043.440$ = 359.920$-ral csökkenthető, ha nincsen megszorítás a méretekre.

6. Minimalizálás méretkorlátozás esetén

Előfordulhat azonban, hogy például helyhiány vagy vezetőségi megszorítás hatására egy felsőkorlát adódik a felállítható szalagok méretére. Legyen ez a felső korlát b, azaz ebben az esetben legfeljebb b méretű szerelőszalagokat használhatunk. Egy kiválasztási problémát egy b felső korláttal jelöljünk $P(b)$-vel. A feladat formálisan ekkor a következő:

$$\sum_{i=1}^{l} s_i x_i \geq d \qquad i = 1, \ldots, l,$$

$$x_i \geq 0,$$

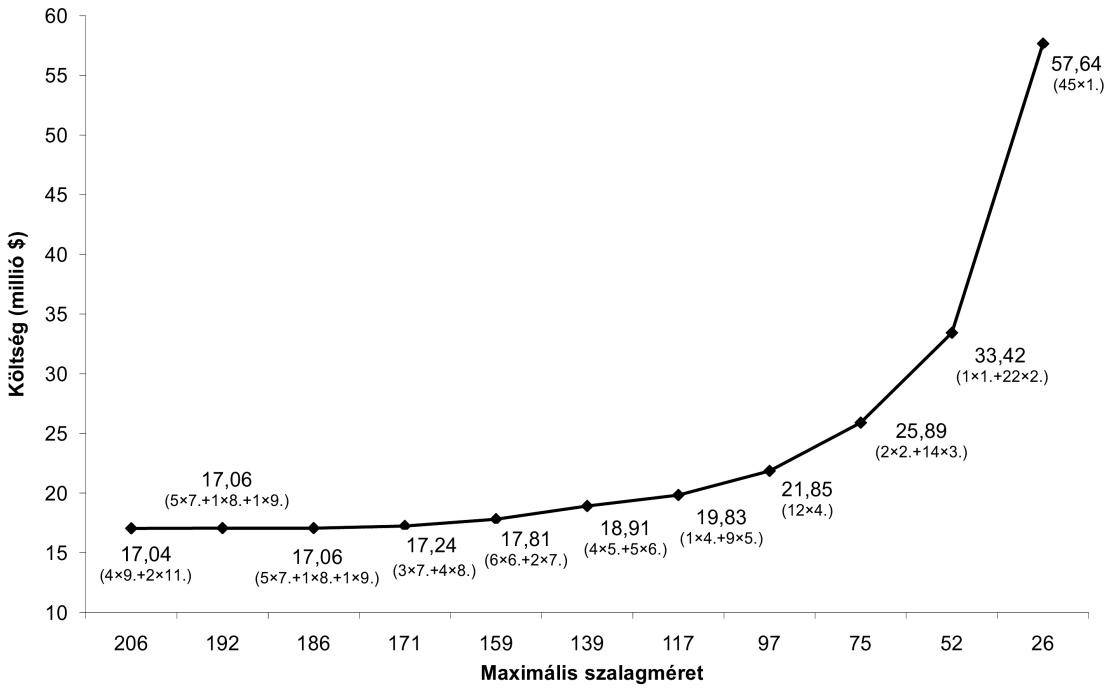
$$x_i \in \mathbb{Z},$$

$$c_i \leq b,$$

$$\sum_{i=1}^{l} c_i x_i \to min,$$

ahol l a méretkorlátozás figyelembe vétele nélkül készíthető szalagok száma.

Egy $b_1 > b_2$ esetén a $P(b_2)$ megoldástere részhalmaza a $P(b_1)$ megoldásterének. Ezért az utóbbi problémának jobb megoldása lesz, vagyis az összköltség kevesebb lesz. A *2. ábra* mutatja az 1. példa költségnövekedését méretkorlátozás esetén.



2. ábra. Költségek méretkorlátozás esetén

Látható, hogy nem minden korlát jelent új eredményt. A 192-es és 186-os korlát ugyanazt az eredményt adja.

7. Program

Az előző fejezetekben megoldást adtunk arra, hogy egy termék gyártására megtaláljuk a legolcsóbb összetételű gyártóberendezést. Egy konkrét példán végig is követtük a számolás menetét, pontosabban az eredményeit, hiszen a műveletek mechanikus végrehajtása, még az egyszerű esetekben is sok időt vesz igénybe. Ezért készült a dolgozathoz egy program, amely szintén feltételezi, hogy egy szerelőszalag költsége egy konstansból és egy állomásszámtól függő részből áll. A műveleti idők, valamint a műveleti sorrend bevitele után megadja az elérhető szalagméreteket, ezek költségeit, majd a megadott igényre összeállítja a minimális költségű kombinációt, figyelembe véve az esetleges méretkorlátozásokat is. A program nagyobb részei: az adatok beolvasása, ebből az utózmány mátrix meghatározása, a lehetséges szalagméretek meghatározása, az f és a h függvények értékeinek kiszámítása és végül a méretkorlátok figyelembe vételével az ideális kombináció elkészítése.

7.1. Adatbeolvasás

A program körülbelül 15 műveletet tud kezelni egy percen belül. A művelet szétosztás ideje exponenciális gyorsasággal nő.

Először azt kell megadnunk, hogy percben kifejezve átlagosan mennyi időt fordítanak naponta a termelésre, majd a költségek megadása történik. A 4. fejezetben ismertetett költségszerkezetnél feltételeztük, hogy a költség olyan alakra hozható, amely csak az állomások számától függ. Ez alapján egy két tagú összegként írható fel, ahol az egyik tag konstans, a másik tag pedig az állomásszámtól függ. A gyártásra fordított idő után ezt a két költségelemet kéri a program.

Ezután a műveletek számát kell megadni, majd az egyes műveletek végrehajtási idejét. Fontos megjegyezni, hogy a műveleteket tizedes pontossággal kell megadni. Általában nem is szoktak ettől pontosabb időket használni. A program tizedesvesszőként csak a pontot értelmezi.

Az ezt követő részben a műveleti gráfot kell megadni. Ehhez először meg kell határozni a műveleteknek egy topológikus sorrendjét, és az adatokat eszerint az indexelés szerint kell bevinni. Vagyis a program csak akkor használható, ha sikerült egy ilyen sorrendet kialakítani. Ekkor a gráf megadása oly módon történik, hogy az utolsó művelet kivételével először megkérdezi az aktuális művelet közvetlen utózmányainak számát, majd egyesével bekéri ezen utózmányok sorszámát. Természetesen a topológikus sorrendből következik, hogy egy művelet minden utózmányának sorszáma nagyobb, mint a művelet sorszáma. Ezt a program is figyeli, és csak ilyen utózmány megadását engedélyezi. A legnagyobb indexű művelet biztosan nem

előzménye egyik műveletnek sem, ezért nem kér az utolsó művelethez utózmányt. Ezzel megadtuk a kiinduláshoz szükséges adatokat.

A program elkészíti az utózmány mátrixot, majd szétosztja a műveleteket, és kiírja a kapott eredményeket. Állomásszámonként megjelenik a minimális ciklusidő, és sorban az x_α értékek, vagyis az, hogy az egyes műveleteket melyik állomáshoz rendelte. Végül összegezve láthatjuk a lehetséges szalagméreteket és ezek költségeit.

Ezt követően van lehetőség az igény megadására. Ekkor dönthetjük el azt is, hogy szabunk-e felső korlátot a felhasználható szalagméretekre, és ha igen, akkor mennyi legyen ez.

Ezzel a feladat minden paraméterét megadtuk, már csak az eredmény kiírása következik. Megkapjuk a különböző méretű szalagok minimális költségű kombinációját, ennek költségét és azt, hogy ekkor ténylegesen hány darabot állítottunk elő naponta.

A továbbiakban lehetőség van arra, hogy a kezdeti adatok újbóli megadása nélkül más igény illetve méretkorlát esetén is megtudjuk az ideális kombinációt.

Most pedig nézzük meg, hogy hogyan állítja elő a program az eredményeket.

7.2. Utózmány mátrix meghatározása

Először is vezessük be a használandó jelöléseket:
$v[i][j]$: az aktuális utózmány mátrix; értéke 1, ha az aktuális lépés szerint a j művelet utózmánya az i-nek, különben pedig 0,
$t[i][j]$: az előző lépés szerinti ideiglenes utózmány mátrix.

Ez a részalgoritmus $n-1$ lépésből áll, ahol n továbbra is a műveletek száma. Az első lépésben a bemenő adatok alapján kitölti a közvetlen utózmány mátrixot. Azaz $v[i][j]$ értékét 1-re állítja, ha az i. művelet közvetlen utózmányai között megadtuk a j. műveletet is. Az ezt követő lépésekben az előző lépés szerinti ideiglenes t utózmány mátrix alapján módosítja a v mátrixot.

1. **Procedure** Utózmány mátrix
2. **Begin**
3. **for** $i = 1$ **to** n **step** 1 **do**
4. **begin**
5. **for** $j = 1$ **to** n **step** 1 **do**
6. **begin**

```
7.          if (t[i][j] = 1) then
8.          begin
9.              v[i][j] := 1;
10.             for l = 1 to n step 1 do
11.                 if (t[j][l] = 1) then v[i][l] := 1;
12.         end
13.       end
14.   end
15. end
```

Például tekintsük a 2. lépést. Ekkor a t mátrix a közvetlen utózmányok mátrixa, azaz $t[i][j] = 1$, ha a j az i-nek közvetlen utózmánya. A v mátrixot úgy alakítjuk ki, hogy azokon az i, j helyeken, ahol már a $t[i][j]$ is 1 volt, ott a $v[i][j]$ is 1 lesz, ezen kívül ha azt látjuk, hogy egy i elemnek t szerint utózmánya egy j elem, akkor a v-ben az i utózmányait kiegészítjük a j utózmányaival is (7-12. sor).

Előzmény mátrixot is hasonlóan készíthetnénk, de felesleges lenne, hiszen az utózmány mátrix transzponáltja éppen az előzmény mátrix. Így a műveletek szétosztása során az előzményeket és az utózmányokat is ismerjük.

7.3. Műveletek szétosztása

Mint ahogy már az előző fejezetekben leírtuk, ennek a résznek az a feladata, hogy minden $k \in \{1, \ldots, n\}$ állomásszám esetén meghatározza azt a művelet csoportosítást, amivel a ciklusidő minimális lesz. Emlékeztetőül az ideális, azaz a tökéletesen kiegyensúlyozott szalag ciklusidejéből indul ki, és ha így a szétosztást nem lehet elvégezni, akkor növeli a ciklusidőt 0,1 perccel. Ennek megfelelően e programrész több egymásba ágyazott ciklusból tevődik össze.

```
1. Procedure Szétosztás
2. Begin
3.     for k = 1 to n step 1 do
4.     Begin
```

```
5.      ciklusidő(k);
6.      jóciklusidő := false;
7.      while (jóciklusidő= false) do
8.      Begin
9.        for j = 1 to n step 1 do q_j := 0;
10.       szétoszt(ciklusidő, k);
11.       if siker then jóciklusidő := true;
12.       else ciklusidő= ciklusidő+1;
13.     end
14.   end
15. end
```

A külső ciklus $k = 1$-ről indulva az állomások számát növeli, közben minden lehetséges k-ra előállítja a csoportosítást (3. sor). Tehát ez felelős azért, hogy minden k-ra végrehajtódjon a szétosztás. Ciklusmagjában első lépésként az aktuális állomásszámmal meghatározza a kiindulási ciklusidőt (5. sor). Hogy kiküszöbölje az egészekkel és tizedesekkel egyszerre való számolás nehézségeit, a tényleges műveleti idők és ciklusidő tízszeresével számol, majd csak az eredmények kiírásánál alakítja vissza az időket tizedessé. Ezért fontos, hogy a műveleti időket tizedes pontossággal adjuk meg. Így tehát a kiindulási ciklusidő a következő:

$$ciklusidő(k) = \left\lceil \frac{10 \cdot \sum_{\alpha=1}^{n} p_\alpha}{k} \right\rceil,$$

ahol p_α továbbra is az α művelet ideje.

A következő ciklus szerepe, hogy az adott k mellett megtalálja a minimális ciklusidőt és a hozzá tartozó művelet felosztást. A 9. sorral éri el, hogy a próbálkozások elején minden állomás üres legyen. Az itt szereplő q_j továbbra is a j. állomáshoz rendelt műveletek összideje, azaz $q_j = \sum_{\alpha=1}^{n} x_{\alpha j} p_\alpha$. Ezután történik meg a tényleges szétosztás, melynek algoritmusát az alábbiakban részletezzük. Ha sikerült az aktuális *ciklusidő*-vel és k-val kiosztani a műveleteket, akkor ez a ciklus lezárul (11. sor), különben pedig növeljük az aktuális ciklusidőt (12. sor).

A szétosztási algoritmus: Az adatbevitelnél használt topológikus sorrend szerint sorban próbálja állomásokhoz rendelni a műveleteket. A konstrukciós lépésben egy adott művelet esetén megkeresi az első olyan állomást, ahová még befér az aktuális művelet. Ha találunk ilyen állomást, akkor megvizsgálja, hogy ezzel a hozzárendeléssel nem kerül-e előrébb az aktuális művelet, mint valamelyik előzménye. Ha nem, akkor hozzárendelheti ezt az állomást a művelethez, és ugyanígy próbálja elhelyezni a következő műveletet. Ha viszont az előzmények hátrébb vannak, akkor ez az állomás nem lesz megfelelő, meg kell keresni a következő olyan helyet, ahová befér a művelet. Ha már megvizsgált minden állomást, de sehová sem tudta elhelyezni a műveletet, akkor kerül sor az ágcserére.

Az ágcsere algoritmusának ismertetésekor j jelöli a konstrukciós lépésben utolsónak állomáshoz rendelt művelet indexét. Mint ahogy a 4. fejezetben szerepelt, az ágcsere lényege, hogy a már állomáshoz rendelt műveletek közül az utolsó olyat, amit még lehet későbbi állomáshoz rendelni, azt a pillanatnyi állomása utáni első lehetséges helyre toljuk, és az ezt követő összes művelet állomáshoz való rendelését töröljük.

1. **Procedure** Ágcsere
2. **Begin**
3. $tolható := false;\ \beta := j - 1;$
4. **while** $(tolható = false\ and\ \beta > 0)$ **do**
5. **begin**
6. **if** $(x_\beta < k)$ **then**
7. **begin**
8. $\alpha := x_\beta;$
9. **if**$(q_{\alpha+1} + t_\beta < ciklusidő)$
10. **begin**
11. $x_\beta := \alpha + 1;$
12. $q_\alpha := q_\alpha - t_\beta;$
13. $q_{\alpha+1} := q_{\alpha+1} + t_\beta$
14. $tolható := true;$

15. $\acute{a}gcsere := true$;

16. $j := \beta + 1$;

17. **end**

18. **else**

19. **begin**

20. $x_\beta := \alpha + 1$;

21. $q_\alpha := q_\alpha - t_\beta$;

22. $q_{\alpha+1} := q_{\alpha+1} + t_\beta$

23. **end**

24. **end**

25. **else**

26. **begin**

27. $x_\beta = 0$;

28. $q_k = q_k - t_\beta$;

29. $\beta = \beta - 1$;

30. **end**

31. **end**

32. **if** $(\beta = 0)$ **then** $\acute{a}gcsere := false$;

33. **end**

A 4. sorból látszik, hogy addig keres az állomások között visszafelé (19. sor), míg nem talál egy tolhatót, vagy míg el nem fogynak a műveletek. Akkor lehet egy műveletet hátrébb rakni, ha az aktuális állomásának száma még kisebb, mint az összes állomás száma (6. sor). Ha ez teljesül, akkor meg kell nézni, hogy a következő állomáson van-e még annyi hely, hogy ez a művelet oda átkerüljön (9. sor). Ha van, akkor eggyel hátrébb kerül (11. sor), előző állomásán nő, új állomásán viszont csökken a még felhasználható idő (12., 13. sor). Itt tehát az eltolás, és ezáltal az ágcsere is sikeres volt, a konstrukcios lépés a kövekező művelettel folytatódik (14-16. sor). Ha viszont nem áll rendelkezésre elegendő idő, akkor is elvégzi átmenetileg

az áthelyezést (20-22. sor), viszont mást nem csinál. Így újra le fog futni a ciklus, mert még nem sikerült helyesen eltolni, és még van meg nem vizsgált lehetőség. Szintén az ideiglenesen eltolt műveletet próbálja hátrébb helyezni, és közben törli az ideiglenes helyéről. Ezt addig ismétli, míg nem talál neki valóban szabad helyet, vagy meg nem vizsgál minden állomást. Így látható, hogy nem követ el hibát azzal, hogy átmenetileg egy állomáson túllépi a ciklusidőt, hiszen ha a ciklus elején az aktuális művelet a legutolsó állomáson van, akkor törli az állomás elhelyezését (27-29. sor), és ezzel az átmeneti hibákat is korrigálja. Ha azért áll le a ciklus, mert már minden műveleten végigért (32. sor), akkor nem sikerült egyetlen elemet sem hátrébb tolni. Ekkor az aktuális ciklusidővel nem oldható meg a műveletek szétosztása. Ebben az esetben tehát növeli $0,1$ perccel a ciklusidőt, és újra próbálkozik.

Közben az előzmények és utózmányok nem sérültek, mert a műveleteket topológikus sorrendben számoztuk. Vagyis egy művelet elhelyezésekor már minden előzményét hozzárendelte egy állomáshoz, és utózmányait pedig csak ezután próbálja elhelyezni.

A ciklusok lefutása után megkapjuk a lehetséges szalagméreteket és ezek költségeit is. Már csak a megfelelő kombinációt kell összeállítani az igény teljesítéséhez.

7.4. Szalagválasztás

Az ötödik fejezet szerint a megfelelő szalagok kiválasztásához szükség van két segédfüggvényre, f-re és h-ra, ahol
$f_i(x) -$ a minimális költség egy x igényre, ha az $1, \ldots, i$ szalagokat használhatjuk,
$\{h_i(x) = j\} -$ azt jelenti, hogy az $1, \ldots, i$ szalagok használata esetén egy s_j méretű szalagot kell az eddigiekhez hozzávenni ahhoz, hogy teljesítsük az x igényt.

Azt is tudjuk, hogy ezen függvények értékét rekurzívan állíthatjuk elő:

$$f_i(x) = min\{f_{i-1}(x), f_i(x - s_i) + c_i\}.$$

Valamint ha $f_i(x - s_i) + c_i$ a kisebb, akkor $h_i(x) = i$. Ha az $f_{i-1}(x)$ kisebb, akkor $h_i(x) = h_{i-1}(x)$.

A program a rekurzív képletnek megfelelően felépíti az $n \times d$-es $f[i][x]$ és $h[i][x]$ mátrixokat. Ennek algoritmusát egyszerűsége miatt nem részletezzük. Két ciklussal végigmegy a mátrix sorain és oszlopain, és a már meglevő elemek alapján meghatározza az új elemeket. A mátrixok ismeretében már visszaléptetéssel ki tudja választani a szükséges szalagokat. Ezelőtt viszont meg kell adni, hogy van-e méretkorlátozás a használható szalagokra. Ha nem akarunk korlátot szabni, az ekvivalens azzal, hogy a legnagyobb méretű szalag méretét választjuk felső korlátnak,

hiszen így minden szalag használatát megengedjük. Ezért a kiválasztás algoritmusának ismertetésekor azt feltételezzük, hogy szeretnénk korlátot bevezetni. Közben felhasználjuk a szétosztási algoritmusban előállított méret és költség vektorokat, valamint kitöltjük a kombinációs vektort:
$c[i]$ – az i. szalag költsége,
$s[i]$ – az i. szalag mérete,
kombináció$[i]$ – a felállítandó i. indexű szalagok száma.

1. **Procedure** Visszaléptetés
2. **Begin**
3. $index := 1$;
4. **while** $(s[index + 1] \leq korlát)$ **do** $index = index + 1$;
5. $maradék := d$;
6. **for** $i = 1$ **to** n **do** *kombináció*$[i] := 0$;
7. **while** $(maradék > 0)$ **do**
8. **Begin**
9. $l := h[index][maradék]$;
10. $kombinació[l] = kombinació[l] + 1$;
11. $költség := költség + c[l]$;
12. $méret := méret + s[l]$;
13. $maradék := maradék - s[l]$;
14. **end**
15. **end**

Első lépésben azt kell meghatározni, hogy melyik az a legmagasabb indexű szalag, amelynek mérete még kisebb, mint a korlát (3., 4. sor). Ezután ha egy szalagot kiválasztunk, akkor egyidejűleg módosítjuk a maradék kielégítendő igényt (9., 10., 13. sor), ami kezdetben megegyezik a tényleges igénnyel (5. sor), illetve folyamatosan számoljuk a kombináció költségét és az elért kapacitást (11., 12. sor).

Ezzel minden számítást elvégezte a program, kiírja az eredményeket, és lehetőségünk van új igényre is kérni az összeállítást.

8. Összegzés

A dolgozat célkitűzése az volt, hogy bemutassa a szerelőszalagok működését, és segítséget nyújtson egy olyan összetett feladat megoldásához, mint a szerelőszalagok tervezése. A tervezést több tényező befolyásolja, például más-más szemléletet igényel aszerint, hogy az igények állandóak vagy változékonyak, illetve aszerint, hogy milyen nyersanyagkezelő rendszert használnak.

Mi a modellünkben feltettük, hogy az igény állandó, a szerelőszalagokon minden állomásból pontosan egy található, és a termékek egyszerre mozdulnak el az állomásokról. Mint láttuk, még ekkor is több lehetőség közül választhatunk. Az igényt kielégíthetjük egyforma és különböző szalagokkal is. Az ideális szalagszerkezet meghatározásához nem csak algoritmust adtunk, hanem a [2]-ben található gyártási folyamatok egyikére az algoritmus alkalmazásával össze is tudtuk állítani a minimális költségű gyártóberendezést egyforma és különböző szalagokkal is. Ehhez felhasználtuk az ott szereplő kiindulási adatokat, az eredményeket pedig az általunk felállított modell alapján készített programmal számoltuk. A [2]-ben nem szerepel erre a gyártási folyamatra az ideális összeállítás, csak egy másik folyamatra adja meg a végeredményeket, ennek a másik folyamatnak viszont nem ismerjük a kiindulási adatait. Ha összehasonlítjuk a kapott eredményeket egyforma és különböző szalagok esetén, akkor elmondható, hogy különböző szalagok használata esetén lényegesen kisebb költségeket érhetünk el. Kitértünk arra az esetre is, mikor a használható szalagok mérete felülről korlátos. Megnéztük, hogy példánkban hogyan alakulnak a költségek akkor, ha csökkentjük a felállítható szalagok maximális méretét. A kapott eredmény megfelelt a várhatónak, vagyis ilyen korlátozások bevezetésekor a költségek jelentősen megugrottak.

Összeségében tehát sikerült egy jól alkalmazható modellt és eljárást nyújtani a tervezéshez abban az esetben, ha a fenti feltételek teljesülnek. Valamint ezen túlmenően készítettünk egy olyan programot is, amely a modell alapján tetszőleges gyártási folyamatra, költségekre, termelési időre és igényre meg tudja határozni a lehetséges szalagméret-szalagköltség párokat, és megadja az ezek eléréséhez szükséges művelet-csoportosításokat. Majd ezekből a szalagokból összeállítja az ideális kombinációt az igény kielégítéséhez abban az esetben is, ha méretkorlátozást akarunk használni.

Láthattuk, hogy más feltételekkel is építhető gyártóberendezés. A modellünktől eltérő speciális esetekben viszont figyelembe kell venni az eltéréseket, és ennek megfelelően változtatni a megoldáson.

Hivatkozások

[1] Vizvári Béla, Bevezetés a termelésirányítás matematikai módszereibe, egyetemi jegyzet,*ELTE Budapest* (1994): 133-168.

[2] We-Min Chow, Assembly Line Design: Methodology and Applications, *Marcel Dekker, Inc., New York and Basel* 1990.

[3] Bowman, E.H., Assembly line balancing by linear program, *Operation Research*, 8(1960) 385-389.

[4] Talbot, F.B., Patterson, J.H., An Integer Programing Algorithm with Network Cuts for Solving the Single Model Assembly Line Balancing Problem, *Management Science* 30(1984).

[5] Gilmore, P.C., and R.E. Gomery, The Theory of Computation of Knapsack Function., *Journal of Operations Research Society of America* 1966.