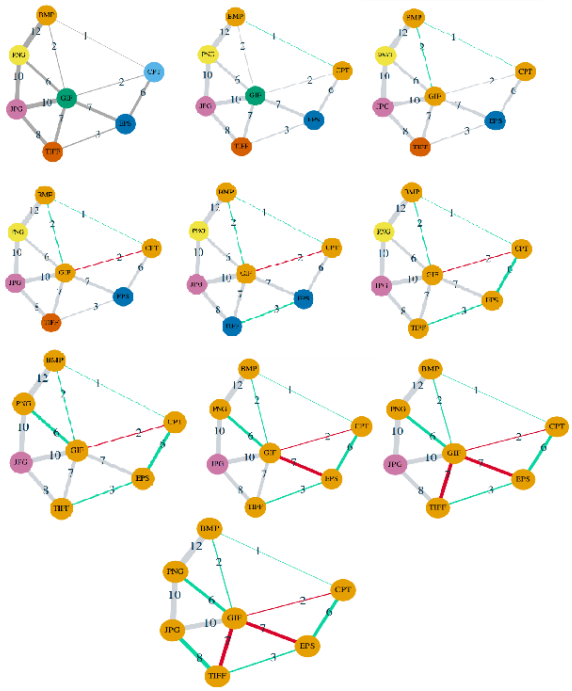
Gráfok:

Legyen A egy halmaz, R ⊆ A×A pedig egy bináris reláció. Az R reláció egy úgynevezett irányított gráffal szemléltethető. A elemeit ilyenkor apró körökkel ábrázoljuk, amiket az irányított gráf csúcsainak hívunk. Az irányított gráf egy a csúcsából nyíl mutat egy b csúcsába akkor és csak akkor, ha (a, b) ∈ R. Ezeket a nyilakat az irányított gráf éleinek nevezzük. Az R⊆A×A relációt szimmetrikusnak mondjuk, ha minden (a, b)∈R esetén (b, a)∈R. Ez azt jelenti, hogy a relációhoz tartozó irányított gráfban ha van egy nyíl két csúcs között, akkor a két csúcs között van az ellenkező irányba mutató él is. Az ilyen relációkat ábrázolhatjuk irányítatlan gráffal is. Ilyenkor a csúcsokat nem oda-vissza mutató nyilak, hanem egyszerű vonalak kötik össze.

Minimális feszítőfa meghatározása – Kruskal algoritmus



Feszítőfa: Részgráfja az eredeti gráfnak és minden csúcsot érint, fa (nincs benne kör), minimális az éleinek száma

Minimális [súlyú] feszítőfa: Ennél kisebb élsúlyú feszítőfa nincs a gráfban, az éleinek az összsúlya minimális

Input súlyozott, irányítatlan gráf

Output minimális feszítőfa1 (feszítőerdő)

1. minden csúcsot külön halmazba teszünk
2. rendezzük az éleket súly szerint növekvő sorrendbe
3. amíg nem minden csúcs esik azonos halmazba vegyük ki az első (legkisebb súlyú) élet:

a. ha a két végpontja különböző halmazban van: bevesszük a minimális feszítőfába + egyesítjük a két halmazt

b. ha azonos halmazban van, akkor kihagyjuk

- a feszítőfa élei: PNG – GIF (6), GIF – BMP (2), BMP – CPT (1), CPT – EPS (6), EPS – TIFF (3), TIFF – JPG (8).
- a feszítőfa súlya: 6 + 2 + 1 + 6 + 3 + 8 = 26

Legrövidebb út meghatározása

- két pont között
- egy pontból mindenhova
- minden pontból egy pontba
- minden pontból mindenhova

(ezek egyre általánosabbak 1<2,3<4)

Input

1. Irányított gráf, amiben nincs negatív súlyú él

2. Kezdőcsúcs

Output Legrövidebb út a kezdőcsúcsból az összes többi csúcsba

0. Megjegyezzük a meglátogatott csúcsokat, meg is jelöljük a kezdőcsúcsot

1. Megjegyezzük az eddigi legrövidebb távolságokat a kezdőcsúcsból Ę ha valaki nem érhető el közvetlenül, ∞-t kap

2. Ha még nem látogattunk meg minden csúcsot:

2.1. látogassuk meg a legközelebb lévő, még meg nem látogatott csúcsot

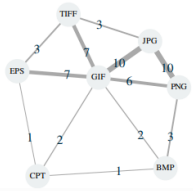
2.2. az élein keresztül frissítsük a távolságot = a csúcs távolsága + az él súlya ha rövidebb az új távolság, felülírjuk

Honnan	Hova	Ár
BMP	CPT	1
BMP	GIF	2
BMP	PNG	3
CPT	EPS	1
CPT	GIF	2
EPS	TIFF	3
EPS	GIF	7
TIFF	JPG	3
TIFF	GIF	7
JPG	PNG	10
JPG	GIF	10
PNG	GIF	6

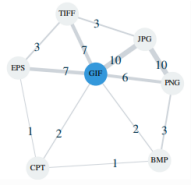
Feladat:

"A szoftverpiacon 7 féle grafikus formátum közötti oda-vissza konverzióra használatos programok kaphatók. Az i-edik és a j-edik között oda-vissza fordítás ideje itt látható. Javasoljunk fordítókat annak megtervezésére, hogy minden egyes formátumról GIF formátumra a lehető leggyorsabban konvertáljunk!" 0. lépés. Rajzoljuk fel a gráfot a táblázat alapján.

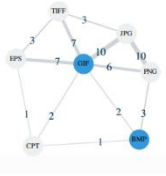
Cúcs	Távolság	Szülő	Iteráció
BMP	Inf		
CPT	Inf		
GIF	0.00		
PNG	Inf		
EPS	Inf		
TIFF	Inf		
JPG	Inf		



Cúcs	Távolság	Szülő	Iteráció
BMP	2.00	GIF	1
CPT	2.00	GIF	1
*GIF	0.00		
PNG	6.00	GIF	1
EPS	7.00	GIF	1
TIFF	7.00	GIF	1
JPG	10.00	GIF	1

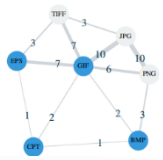


Cúcs	Távolság	Szülő	Iteráció
*BMP	2.00	GIF	1
CPT	2.00	GIF	1
*GIF	0.00		
PNG	5.00	BMP	2
EPS	7.00	GIF	1
TIFF	7.00	GIF	1
JPG	10.00	GIF	1

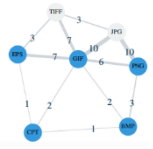


Cúcs	Távolság	Szülő	Iteráció
*BMP	2.00	GIF	1
*CPT	2.00	GIF	1
*GIF	0.00		
PNG	5.00	BMP	2
EPS	3.00	CPT	3
TIFF	7.00	GIF	1
JPG	10.00	GIF	1

Cúcs	Távolság	Szülő	Iteráció
*BMP	2.00	GIF	1
*CPT	2.00	GIF	1
*GIF	0.00		
*PNG	5.00	BMP	2
*EPS	3.00	CPT	3
TIFF	6.00	EPS	4
JPG	10.00	GIF	1

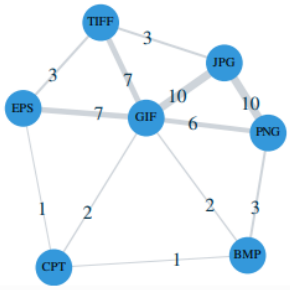


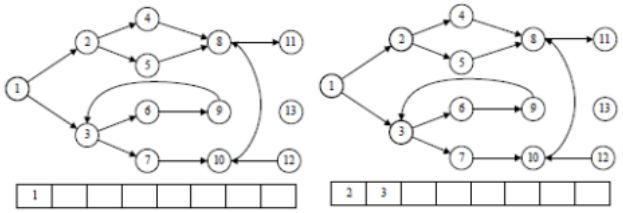
Cúcs	Távolság	Szülő	Iteráció
*BMP	2.00	GIF	1
*CPT	2.00	GIF	1
*GIF	0.00		
*PNG	5.00	BMP	2
*EPS	3.00	CPT	3
TIFF	6.00	EPS	4
JPG	10.00	GIF	1

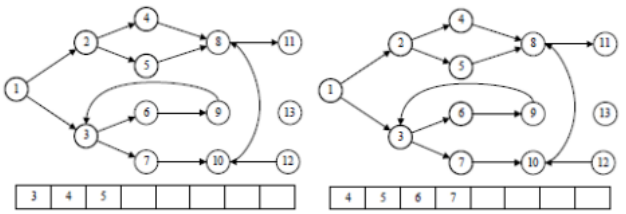


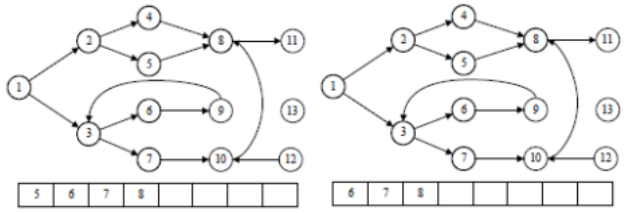
Cúcs	Távolság	Szülő	Iteráció
*BMP	2.00	GIF	1
*CPT	2.00	GIF	1
*GIF	0.00		
*PNG	5.00	BMP	2
*EPS	3.00	CPT	3
*TIFF	6.00	EPS	4
JPG	9.00	TIFF	6

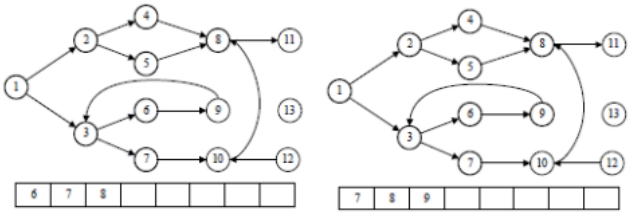
Cúcs	Távolság	Szülő	Iteráció
*BMP	2.00	GIF	1
*CPT	2.00	GIF	1
*GIF	0.00		
*PNG	5.00	BMP	2
*EPS	3.00	CPT	3
*TIFF	6.00	EPS	4
*JPG	9.00	TIFF	6

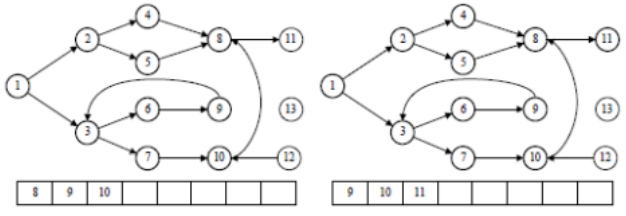










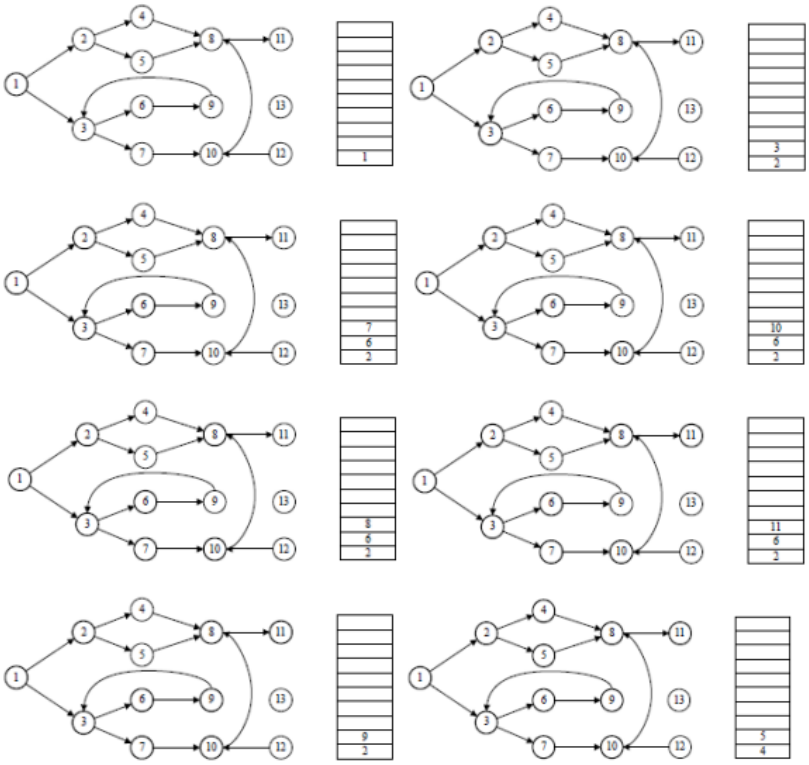


GRÁF BEJÁRÁSOK

Keresés. A két algoritmus egy adott csomópontból indul ki. Nyilvántartjuk azokat a csomópontokat, amelyeknek a szomszédait még nem vizsgáltuk meg, kiválasztunk közülük egyet és megjelöljük ennek a szomszédait. Ezt addig ismételjük, amíg van olyan csomópont amit megjelöltünk, de a szomszédait még nem vizsgáltuk. Ügyelni kell arra, hogy ha léteznek körök a gráfban, akkor így végtelen ciklusba kerülhet az algoritmus, hiszen a kör csomópontjait újra és újra végigjárjuk. Ezt úgy lehet kivédeni, hogy csak azokat a szomszédokat jelöljük meg, amelyeket még nem vizsgáltunk.

Szélességi bejárás (FIFO) →

Sor adatszerkezet: belerakjuk a kezdő csomópontot, minden új elemet a végére fűzünk és csak a sor elején lévő elemet vehetjük ki belőle. Miután az első csúcspontot beillesztettük a sorba, következnek az iterációk.

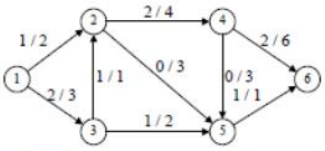


←Mélységi bejárás (LIFO)

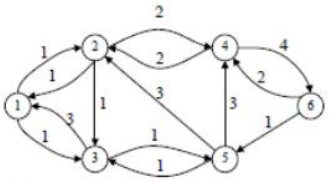
Verem adatszerkezet: végére szúrjuk az új elemet, és a végéből veszünk ki. A legutoljára beszúrt szomszéddal megyünk tovább

MAXIMÁLIS FOLYAM →

A termelőből a lehető legnagyobb anyagmennyiséget szállítjuk a fogyasztóba a hálózaton úgy, hogy teljesül az anyagmegmaradás, és a kapacitási korlátozásokat sem sértjük meg.



3.39. ábra. Hálózat kapacitásokkal és folyam értékekkel



3.40. ábra. Az előző hálózat reziduális hálózata

speciális csúcsok:

- forrás (termelő) (S): innen csak kifele áramlik a folyam (de bármennyi jöhet)
- nyelő (fogyasztó) (T): ide csak befele áramlik folyam (de bármennyi jöhet)

Minden élnek két tulajdonsága van:

- folyam: jelenleg ennyit visz (folyam <= kapacitás!!)
- kapacitás: maximum ennyi egységet bír átvinni időegység alatt (a nem gráfhoz tartozó élek kapacitása 0)

A csúcsokba pont annyi folyik ki, mint amennyi be.

A nem speciális csúcsokra igaz az anyagmegmaradási törvény, nem termel, nem fogyaszt és nem tárol.

Kérdések, amik érdekelnek minket:

- mennyi a maximális folyam, ami át tud menni a hálózaton?
- merre mennyit tudok átvinni?
- hogy tudom meghatározni a maximális folyamot?

Megjegyzés Az éleken a folyam és a kapacitás jelölése: “folyam / kapacitás”

ferde szimmetria szabály: f(v, u)=−f(u, v) (n egységnyi anyagot szállítunk u-ból v-be és ugyanezt az anyagmennyiséget a másik irányba is szállítjuk, akkor a folyam értéke u és v között lenullázódik, hiszen végeredményeben ugyanazt kapjuk ha nem szállítunk a kettő között semmit)

Ford-Fulkerson:

Input Gráf a kapacitásokkal **Output** Maximális folyamú gráf

Menet

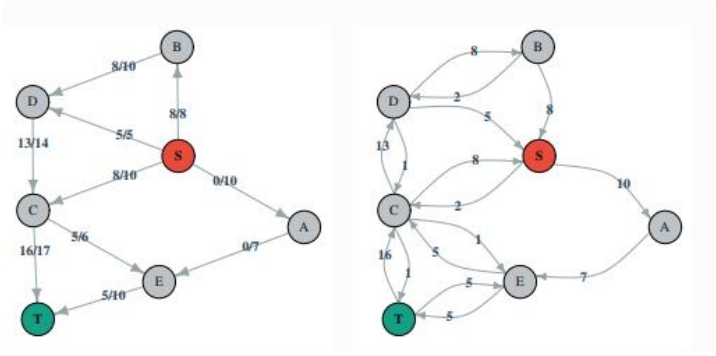
1. Keressünk javítóutat S-ből T-be

2. Az úton növeljük meg minden él folyamát a rendelkezésre álló reziduális kapacitások minimumával

3. Ismételjük ezt addig, amíg nincs több javítóút

Reziduális hálózat:

olyan érték, amely megadja, hogy adott folyam esetén legfeljebb mennyivel növelhetjük a folyamot úgy, hogy a kapacitást ne haladja meg.



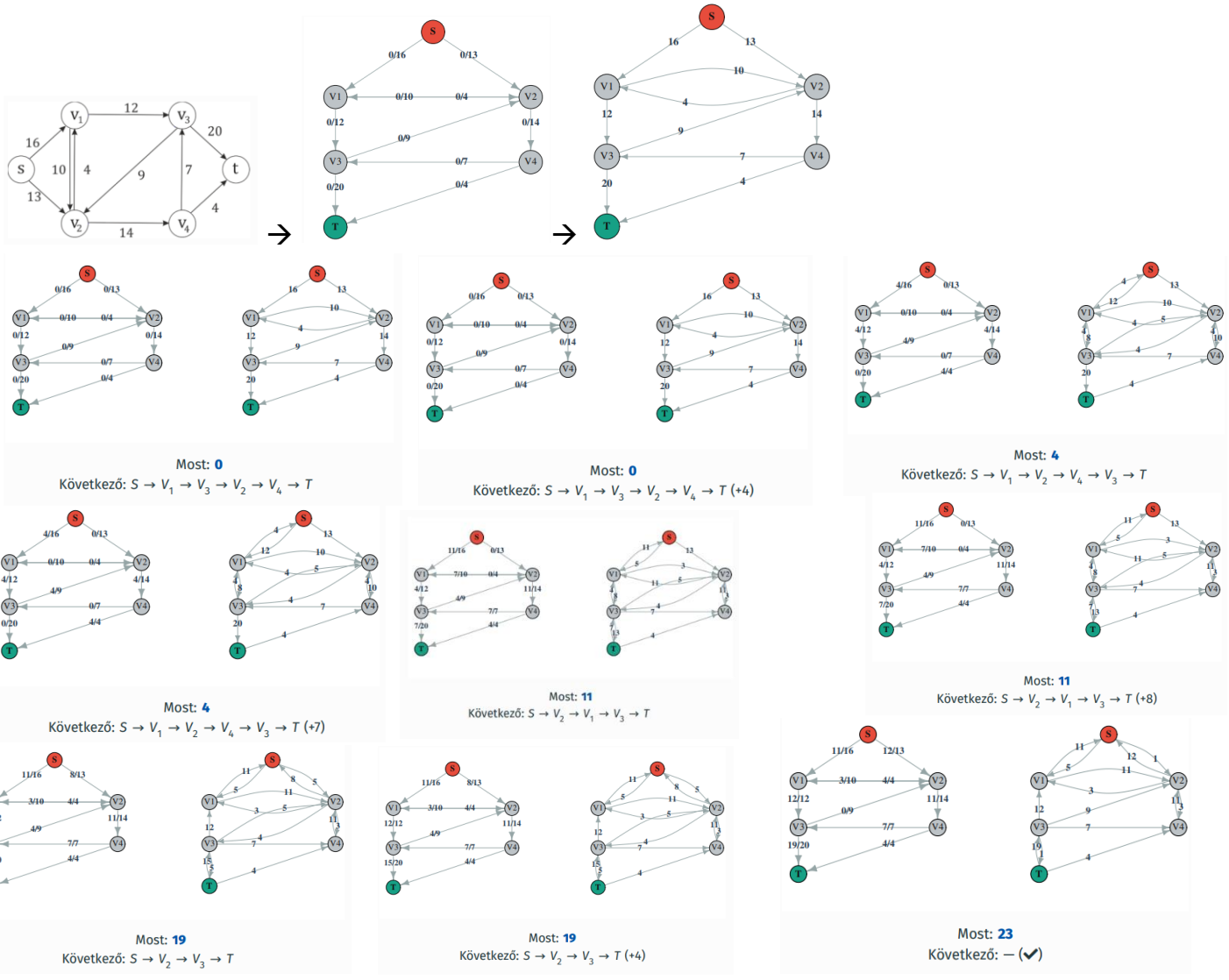
Ez segít leolvasni, hogy melyik irányban mennyivel tudunk egy élen folyamot növelni

Hogy csináljuk:

1. vegyünk egy élet az eredeti folyamban (pl. $A \rightarrow B$)

2. reziduálisban A-ból B-be annyi megy, amennyi **szabad** kapacitás még van abban az irányban

3. reziduálisban B-ből A-ba (hátrafele) annyi megy, amennyit **elhasználtunk**



LINEÁRIS PROGRAMOZÁS

A lineáris korlátozásokkal és célfüggvénnyel adott optimalizálási feladat felírását és megoldását lineáris programozásnak vagy LP-nek nevezzük, magát a feladat ezen formáját pedig lineáris programozási modellnek vagy LP modellnek hívjuk. Létezhet több optimális megoldás is. Ha ez fellép akkor a modellnek alternatív megoldása is van. A szimplex típusú módszerekkel könnyebben találhatunk alternatív megoldásokat.

Cél (Lineáris) függvény minimalizálása (pl. költség) vagy maximalizálása (pl. profit)

Úgy, hogy közben korlátoznak minket:

• csak x egység anyagunk van a gyártáshoz

• ügyfél nem venne az y termékből z-nél többet

• a munkások nem hajlandók napi 18 óránál többet dolgozni

• stb. (minél több korlátozás, a megoldás megkeresése annál tovább tart)
(több változó hozzáadásával az időigény nem nő túlzottan)

Jelenleg:

- Akármennyit süthetünk, mind elfogy.
- A margarétát 600, a hawaii-t 800-ért. Melyikből mennyit készítsünk?
- Maximalizáljuk a bevételt nap végére.
- Nem csak egész pizzát gyárthatunk.
- Nyersanyagok mennyisége korlátozott és vannak olyan hozzávalók, amelyek több pizzára is jók.
- Egy nap sajtból 550, sonkából 150, ananászból pedig 120 adag áll rendelkezésre.

$$x_i \geq 0, \quad i = 1,2$$

$$\max \quad 600x_1 + 800x_2$$

$$10x_1 + 5x_2 \leq 550$$
$$2x_1 + 4x_2 \leq 150$$
$$3x_2 \leq 120$$

4.1. táblázat. A feladat paraméterei

	margaréta	hawaii
sajt	10	5
sonka	2	4
ananász	0	3

Fontos fogalmak:

• döntési változók: $x1$, $x2$, … (pl előállítandó pizzák)

• feltételek: $2x1 - 3x2 \leq 300$

• célfüggvény: max →x $z = 20x1 + 30x2$

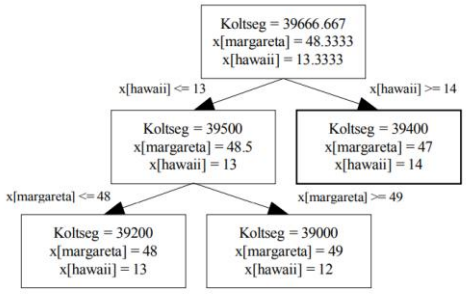
EGÉSZ ÉRTÉKŰ VÁLTOZÓK

Bináris változók a) logikai értékek (↑ és ↓, igaz és hamis) (ekkor is igaz hogy 0 és 1 el jelöljük őket)

b) egész számok (1 és 0; emiatt összeadás, szorzás stb. működik)

Az egész értékű feladatok megoldása jóval nehezebb, mint a csak folytonos változókat tartalmazóké. A csak egész változókat tartalmazó feladatok, egészértékű lineáris problémáknak (integer programming, IP) nevezzük. Ha a változók egy része egész, a többi folytonos, akkor vegyes egészértékű lineáris problémáról (mixed integer linear programming, MILP) beszélünk.

Korlátozások hozzáadása többnyire gyorsít a megoldás menetén. Viszont nő a megoldási idő, ha több egészértékű változót adunk a modellhez.



4.14. ábra. Szétválasztjuk az első gyerek részproblémát

Először eltekintünk az egészértékű kikötéstől. Ezt a módosított modellt hívjuk az eredeti egészértékű feladat relaxációjának, majd vegyünk csak egész értékű pizzákat. Először tehát oldjuk meg a modell relaxációját, ekkor azt kapjuk, hogy a célfüggvény értéke 39666.66667, az x[margareta] 48.3333, az x[hawaii] pedig 13.3333. Mivel a kapott megoldásban az egész változók értékei nem egészek, ezért nem ér véget az algoritmus. Válasszunk ki egy változót, amely bár egész típusú, de a relaxált feladat megoldásában nem egészre jött ki. Legyen ez az x[hawaii]. Oldjuk meg úgy a feladatot, hogy ez a változó ne vehesse fel a 13.3333 értéket: 13-at, vagy annál kevesebbet, illetve 14-et vagy annál többet viszont felvehet. Származtassunk tehát az eredeti modellből két újabb modellt, azzal a módosítással, hogy az egyik esetében egy új korlátozással előírjuk, hogy az x[hawaii] kisebb vagy egyenlő legyen mint 13, a másik változatban pedig nagyobb, vagy egyenlő legyen, mint 14. Azt mondjuk, hogy az x[hawaii] változó mentén szétválasztjuk a feladatot. Oldjuk meg a két módosított modell relaxációját is, majd újra vizsgáljuk meg a kapott változókat.

Először elvégezzük az első szétválasztást (első két sor) Majd szétválasztjuk az első gyerek részproblémát.

KLASSZIKUS PROBLÉMÁK:

Döntési változók:

- élekkel dolgozunk, ezekre veszünk fel változókat
- irányított gráfokkal dolgozunk → ha irányítatlan, mindkét irányban felveszünk irányított élet
- általában a döntés: bevegyük-e az élet

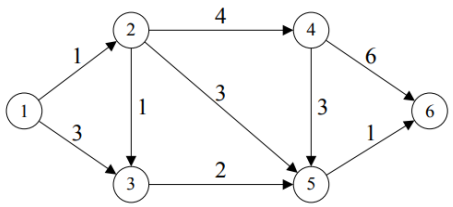
LEGRÖVIDEBB ÚT

Döntési változók: Bevegyük-e az élet (yij) (bináris); Az élsúlyokat jelölje (xij)

Célfüggvény: min $\sum i,j\ dij \cdot yij$ (dij az él súlya)

Feltételek:

- Kiinduló csúcsból (s) eggyel több kimenő, mint bemenő él (foka egy)
- Cél csúcsba (t) eggyel több bemenő, mint kimenő él
- Többi csúcsban ugyanannyi bemenő, mint kimenő (foka kettő)



4.16. ábra. Keressük meg a legrövidebb utat az 1. csomópontból a 6-osba

```
var x12 binary;
var x13 binary;
var x23 binary;
var x24 binary;
var x25 binary;
var x35 binary;
var x45 binary;
var x46 binary;
var x56 binary;

s.t. indulas: x12 + x13 = 1;

s.t. n2: x12 = x23 + x24 + x25;
s.t. n3: x13 + x23 = x35;
s.t. n4: x24 = x45 + x46;
s.t. n5: x25 + x35 + x45 = x56;

minimize ut: x12 + 3*x13 + x23 + 4*x24 + 3*x25 + 2*x35 + 3*x45 +
6*x46 + x56;
end;
```

Mely éleket kell kiválasztani ahhoz, hogy a kiválasztott élek egy minimális hosszúságú utat határozzanak meg. A kezdő csomópontról tovább kell lépni valamelyik szomszédos csomópontra.

Jelölések:

- d + (v) azon csomópontok halmaza, amelyekre van irányított él a v csomópontból,
- d −(v) a v-t megelőző csomópontok halmaza.

A 4.16 ábrán látható gráfon d + (5) = {1} (szerintem ez itt 6 akar lenni csak ugye szar a jegyzet…), d −(5) = {2, 3, 4}.

A modellt megoldva azt kapjuk, hogy az x12, x25 és x56 változók értéke 1, a többi pedig 0, tehát a legrövidebb út az 1, 2, 5, 6 csomópontokon halad keresztül, a hossza pedig 5.

MINIMÁLIS FESZÍTŐFA

Feszítőfát kell kapnunk (összefüggő, körmentes, n-1 él EBBŐL A 3BÓL ELÉG 2 FELTÉTELT TELJESÍTENI → összefüggő, n-1 feltételt választjuk)

Helyette folyamként képzeljük el a gráfot:

- lesz egy kezdőcsúcsunk, amiből jön legfeljebb n folyam
- minden csúcs elnyel egyet a folyamból (tehát belőle eggyel kevesebb jön ki, mint amennyi bement)
- ha valamelyik csúcsnak nem jut folyam, akkor a gráf nem összefüggő

Kétféle változóra is szükségünk lesz:

- xij (egész) ezzel számoljuk, hogy az adott élen mennyi a (képzeletbeli) folyam
- yij (bináris) ez mondja meg, hogy bevegyük-e az adott élet a feszítőfába. (egy élen akkor lehet folyam, ha az él be van választva, tehát y értéke 1)

Cél: Minimalizáljuk a fa súlyát! $z = \min_{\bar{y}} \sum_{i,j} d_{ij} \cdot y_{ij}$ A bináris változót (y) tesszük a célfüggvénybe!

FELTÉTELEK:

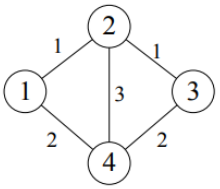
A) Folyam feltételek (minden csúcs egyet elnyel) !!! A kezdőcsúcsra nem veszünk fel ilyet

$$\sum_{i\text{-ből jön él } j\text{-be}} x_{ij} - \sum_{j\text{-ből megy él } i\text{-be}} x_{ij} = 1.$$

B) Felső korlát a folyamra: $x_{ij} \le (n-1) \cdot y_{ij}$

$x_{ij} - (n-1) \cdot y_{ij} \le 0$ (ugyanaz, csak átrendezve)

Megjegyzés: $n - 1$ helyett valami ennél nagyobb számot szoktunk használni max anyagmennyiségre (pl. $n = 8$ csúcsnál 10-et)



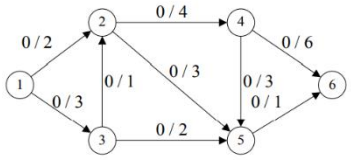
4.19. ábra. A minimális feszítőfa példához tartozó gráf

```
var xAB >= 0;   var xBA >= 0;
var xBC >= 0;   var xCB >= 0;
var xCD >= 0;   var xDC >= 0;
var xAD >= 0;   var xDA >= 0;
var xDB >= 0;   var xBD >= 0;

var yAB binary;   var yBA binary;
var yBC binary;   var yCB binary;
var yCD binary;   var yDC binary;
var yAD binary;   var yDA binary;
var yDB binary;   var yBD binary;

s.t. ab: xAB <= 3*yAB;   s.t. ba: xBA <= 3*yBA;
s.t. bc: xBC <= 3*yBC;   s.t. cb: xCB <= 3*yCB;
s.t. cd: xCD <= 3*yCD;   s.t. dc: xDC <= 3*yDC;
s.t. ad: xAD <= 3*yAD;   s.t. da: xDA <= 3*yDA;
s.t. db: xDB <= 3*yDB;   s.t. bd: xBD <= 3*yBD;

s.t. b: xAB + xDB + xCB - (xBA + xBD + xBC) = 1;
s.t. c: xBC + xDC - (xCB + xCD) = 1;
s.t. d: xAD + xBD + xCD - (xDA + xDB + xDC) = 1;

minimize feszitofa: yAB + yBA + yBC + yCB + 2*(yAD + yDA) + 2*(yDC + yCD)
+ 3*(yBD + yDB);
end;
```

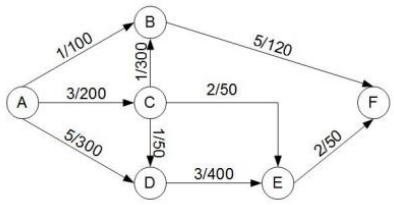
MAXIMÁLIS FOLYAM

Folyam probléma LP-ként:

- élek érdekelnek, folyamot keressük→**egész értékű változók**
- itt az éleken vannak költségek is → **célfüggvény**
- és vannak kapacitások → **feltétel**
- folyam probléma:
  - amennyi befolyik, annyi ki is (kezdő és célpontot kivéve) → **feltétel**



4.23. ábra. Ezen a hálózaton határozzuk meg a maximális folyamot az 1. és 6. csomópontok között



Jelölés

Éleken: egy egység folyam ára / kapacitás az élen.

Feladat

Az irányító problémája, hogy minél olcsóbban juttasson el "A" pontból "F"-be 100 egységnyi anyagot.

Egyik élhez tartozó folyam nagysága sem haladhatja meg az él kapacitását!

Korlátozások:

←Minden folyamhoz tartozzon egy-egy folytonos változó, a folyamértékek meghatározásához.

```
var x12 >= 0;
var x13 >= 0;
var x24 >= 0;
var x25 >= 0;
var x32 >= 0;
var x35 >= 0;
var x45 >= 0;
var x46 >= 0;
var x56 >= 0;
```

← Minden változóra meg kell adni egy-egy korlátot, amely a kapacitás alapján megszabja a változó, azaz a folyam felső korlátját

```
s.t. c12: x12 <= 2;
s.t. c13: x13 <= 3;
s.t. c24: x24 <= 4;
s.t. c25: x25 <= 3;
s.t. c32: x32 <= 1;
s.t. c35: x35 <= 2;
s.t. c45: x45 <= 3;
s.t. c46: x46 <= 6;
s.t. c56: x56 <= 1;
```

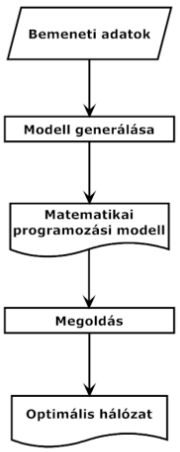
Anyagmegmaradást biztosító korlátozások: 2, 3, 4 és 5-ös csúcsokra→

```
s.t. kibe2: x12 + x32 = x24 + x25;
s.t. kibe3: x13 = x32 + x35;
s.t. kibe4: x24 = x45 + x46;
s.t. kibe5: x25 + x35 + x45 = x56;
```

Célfüggvény: A maximális folyam értéke megegyezik a kezdő csúcsból kiinduló, vagy a cél csúcsba befolyó összes anyag mennyiségével. →

```
maximize folyam: x12 + x13;
end;
```

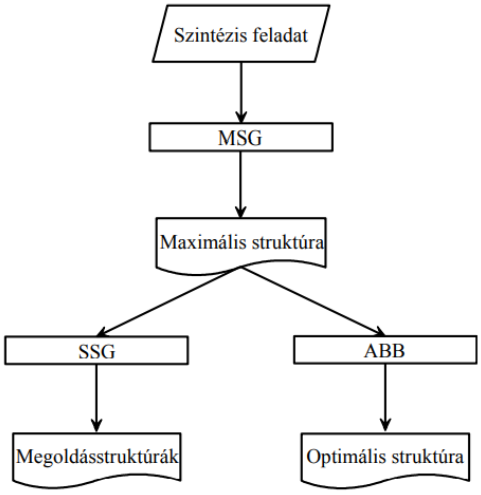
5. FEJEZET – FOLYAMATHÁLÓZAT – SZINTÉZIS ÉS OPTIMALIZÁLÁS



5.1. ábra. A hagyományos algoritmikus módszerek főbb lépései

P-GRÁF MÓDSZERTAN

A nagyméretű feladatok optimumának meghatározására alkalmas.



5.3. ábra. A P-gráf módszertan főbb lépései

ALAPFOGALMAK

M: modellezendő gyártási folyamatban definiált anyagok véges, nem üres halmaza.

Az M anyagaihoz kapcsolódó szintézis feladat: (P = termékek (azok az anyagok, amelyeket a folyamat során valamilyen szempont szerint optimálisan szeretnénk előállítani), R = nyersanyagok (amelyek a gyártás kezdetekor rendelkezésünkre állnak), O = rendelkezésre álló, gyártás során felhasználható műveleti egységek) . Matematikailag az M, P és R halmazok közötti reláció a következőképpen fejezhető ki:

$$P \subset M, R \subset M \text{ és } P \cap M = \emptyset.$$

Az O halmaz elemei, azaz a műveleti egységek végzik az egyes anyagok közötti átalakításokat.

Egy műveleti egységet matematikailag egy rendezett párral modellezünk, azaz (α, β) ∈ O, ahol α jelöli az egység bemenetei anyagainak, β pedig a kimenetei anyagainak a halmazát.

példát:

$$M := \{A, B, C, D, E, F, G, H, I, J, K\}$$

$$P := \{A\}$$

$$R := \{D, F, H, I\}$$

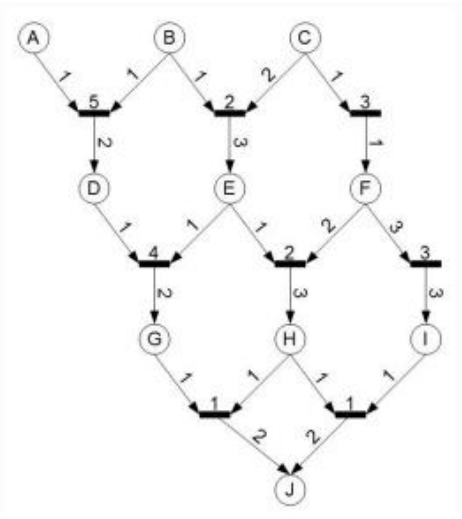
illetve

$$O := \{(\{B\}, \{A, E\}), (\{C\}, \{A, J\}), (\{D, E\}, \{B\}), (\{E, F\}, \{B\}), (\{F, G\}, \{C, K\}), (\{H\}, \{E\}), (\{I, J\}, \{G\})\} = \{O_1, O_2, O_3, O_4, O_5, O_6, O_7\}$$

10 anyagot és 7 műveleti egységet tartalmazó példában az A anyagot szeretnénk gyártani a kiinduláskor rendelkezésre álló D, F, H, I anyagokból.

- az anyagokat, amik se nem nyersanyagok, se nem végtermékek, (a példában B,C, E, G, J) nem szoktuk külön halmazként feltüntetni
- az anyagok, amik műveleti egységek kimeneti, illetve bemenetei anyaghalmazának IS elemei, a köztes anyagok, kiinduláskor nem állnak rendelkezésünkre, a gyártás során műveleti egységek állítják elő őket, illetve a termék előállításhoz fel is használják őket. (a példában B,C, E, G, J) (g-t O7 gyártja és O5 használja fel.)
- az anyagok, amiket a gyártás során műveleti egységek állítanak elő, de más műveleti egységek nem használják fel és nem elemei a P halmaznak sem, melléktermékek. (K anyag, O5 gyárt, de nem használja fel más műveleti egység, és nem eleme a termékek halmazának)

P-GRÁF (Process-graph, P-graph)

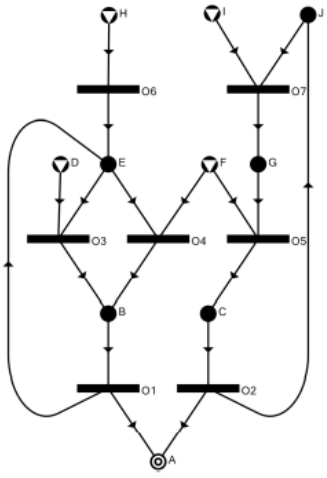


• csúcsok típusai:

- anyagok, termékek (kör)
- feldolgozók, gépek (műveleti egység) (vonal)

Két azonos típusú csúcs nem kapcsolódhat egymáshoz közvetlenül

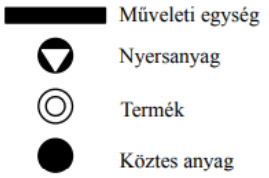
Egy anyag és egy műveleti egység típusú csúcs között akkor és csak akkor megy él, ha a kapcsolatuk része a reprezentálandó folyamatnak.



5.4. ábra. A hálózatszintézis-feladathoz tartozó P-gráf

• élek (gépekre vonatkozóan)

- bemenő: mennyi kell egy anyagból
- kimenő: mennyit állít elő



5.5. ábra. Szimbólumok

Az élek irányai megegyeznek a folyamat előrehaladásának irányával.

• elnevezések (általában)

- legfelső termékek: alapanyagok
- legutolsó termékek: végtermék
- többi: köztes termékek

Amire figyelni kell:

- nem használhatunk több köztes anyagot annál, amennyi van (kevesebbet lehet)
- a köztes anyagokra kell feltételeket felvenni
- lényegében mint a folyam problémánál

Speciális, irányított páros gráf. Páros gráf, ha a gráf csúcsai kettéválasztható 2 halmazba, amelynek elemei között egyik se szomszédos. pl.:

P-gráf egy olyan (m, o) pár, m halmaz elemei az anyag típusú csúcsok, az o halmaz elemei pedig a műveleti egység típusú csúcsok. A gráf élei az halmaz elemei, ahol

$$A_1 := \{(x,y) | y = (\alpha, \beta) \in o \text{ és } x \in \alpha\}$$

és

$$A_2 := \{(y,x) | y = (\alpha, \beta) \in o s x \in \beta\}.$$

- x: az anyag típusú csúcsok
- y: a műveleti egység típusú csúcsok
- α: azon anyag típusú csúcsok halmaza, amelyek**ből** mutat irányított él a műveleti egység típusú csúcsokba
- β azon anyag típusú csúcsok halmaza, amelyek**be** mutat irányított él a műveleti egység típusú csúcsokból.
- A1 élhalmaz minden eleme anyag típusú csúcsokból műveleti egység típusú csúcsokba mutat,
- A2 élhalmaz minden eleme műveleti egység típusú csúcsból mutat anyag típusú csúcsba.
- Két P-gráf, (m1, o1) és (m2, o2) unióját és metszetét, melyek szintén P-gráfok, a következőképpen definiáljuk:

$$(m_1, o_1) \cup (m_2, o_2) = (m_1 \cup m_2, o_1 \cup o_2)$$

$$(m_1, o_1) \cap (m_2, o_2) = (m_1 \cap m_2, o_1 \cap o_2)$$

  Továbbá az (m1, o1) gráf az (m2, o2) gráf részgráfja, azaz

$$(m_1, o_1) \subseteq (m_2, o_2)$$

ha

$$m_1 \subseteq m_2 \text{ és } o_1 \subseteq o_2.$$

KOMBINATORIKUSAN LEHETSÉGES MEGOLDÁSSTRUKTÚRÁK AXIÓMÁI

Az alábbi axiómák fogalmazzák meg azokat a tulajdonságokat, amelyeket EGYSZERRE alkalmazva egy valós, gyakorlatban megvalósítható folyamatot reprezentáló struktúrának rendelkeznie kell:

- (S1) Minden legyártandó termék, azaz P minden eleme szerepel a struktúrában.
- (S2) Egy a struktúrában szereplő anyag akkor és csak akkor nyersanyag, ha egyetlen a struktúrában szereplő műveleti egység sem állítja elő.
- (S3) Minden a struktúrában szereplő műveleti egység a szintézis feladatban definiált.
- (S4) Minden a struktúrában szereplő műveleti egységből vezet út legalább egy legyártandó termékhez.
- (S5) Ha egy x anyag része a struktúrának, akkor létezik a struktúrában olyan műveleti egység, amelynek x anyagot felhasználja vagy előállítja.

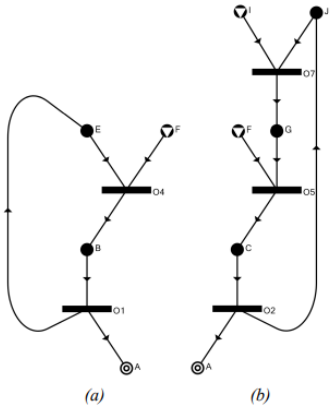
Ha egy szintézis feladathoz tartozó P-gráf kielégíti ezeket az axiómákat, akkor a P-gráfot a probléma egy megoldásstruktúrájának mondjuk. Tekintsük az alábbi szintézis feladatot a megoldásstruktúrák koncepciójának az illusztrálására:

$$M := \{A, B, C, D, E, F, G, H, I\}$$

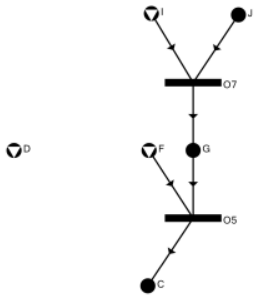
$$P := \{A\}$$

$$R := \{D, F, H\}$$

$$O := \{(\{C\}, \{A, I\}), (\{B\}, \{A, E\}), (\{D, E\}, \{B\}), (\{E, F\}, \{B\}), (\{F, G\}, \{C\}), (\{H, I\}, \{G\}), \}$$



5.6. ábra. Két kombinatorikusan lehetséges megoldásstruktúra



5.7. ábra. Ez a struktúra nem elégíti ki az axiómákat

Ha egy műveleti egység típusú csúcs része egy megoldásstruktúrát reprezentáló P-gráfnak, akkor a műveleti egység bemeneti és kimenetei anyaghalmazainak összes eleme is a gráfhoz tartozik.

Egy megoldásstruktúra nem tartalmazza feltétlenül az M anyaghalmaz összes elemét, illetve az R nyersanyaghalmazból sem kell feltétlenül mindent felhasználnia a gyártás során.

Axiómák teljesülése:

1.) Mivel a gyártandó termék, A, mindkét struktúrában szerepel, mindkét esetben teljesül.

2.) Az első struktúrában az F, a második struktúrában az F és G csúcsok miatt teljesül, nyersanyagként egyedül ezekbe a csúcsokba nem vezet él műveleti egységekből, azaz ezeket az anyagokat nem gyártja semmi.

3.) Az első struktúra két, a második pedig három műveleti egységet tartalmaz, amelyek mindegyike definiálva van a szintézisfeladatban.

4.) Mindkét struktúrában minden műveleti egység típusú csúcsból vezet út a legyártandó termékhez.

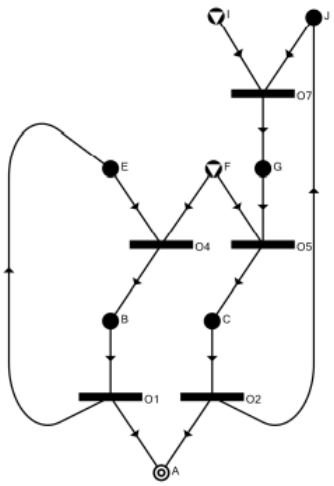
5.) Mindkét struktúrában minden egyes anyag típusú csúcs legalább egy műveleti egységnek a kimenete, vagy a bemenete.

Ez a két struktúra kielégíti az összes axiómát, az 5.7 ábrán a 3as kivételével egyik sem teljesül.

Legyen S(P, R, O) az a halmaz, amely a (P, R, O)szintézisfeladat összes megoldás-struktúráját tartalmazza. Ekkor a megoldás-struktúrákra vonatkozóan az alábbi tételt fogalmazhatjuk meg:

1. **Tétel. A megoldásstruktúrák zártak az unióra, azaz két megoldásstruktúra uniója is megoldásstruktúra.**

Az előző példa megoldásstuktúráinak unióját ábrázolja az 5.8 ábra, ami maga is egy megoldásstruktúra.

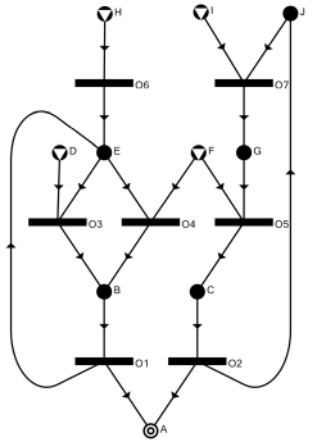


5.8. ábra. Az 5.6 ábrán látható struktúrák uniója

MAXIMÁLIS STRUKTÚRA

Mivel a megoldásstruktúrák halmaza véges és zárt az unióra, a halmaznak lesz egy eleme, μ(P, R, O), amely az összes megoldásstruktúra uniója, azaz $\mu(P, R, O) = \bigcup_{\sigma \in S(P,R,O)} \sigma$ feltéve ha a megoldásstruktúrák

halmaza nem üres, azaz S(P, R, O) ∅. Ekkor a halmaz μ(P, R, O) elemét a (P, R, O) szintézisfeladat maximális struktúrájának (szuperstruktúrának) nevezzük, minden csúcsa és éle része legalább egy megoldás-struktúrának, illetve minden megoldásstruktúra P-gráfja részgráfja a maximális struktúrát reprezentáló P-gráfnak.



5.9. ábra. A szintézisfeladathoz tartozó maximális struktúra

2. Tétel. A maximális struktúra maga is megoldásstruktúra, azaz μ(P, R, O) ∈ S(P, R, O).

A maximális struktúra a feladat összes lehetséges megoldását tartalmazza, így az optimálisat is. Meghatározásával lecsökkentjük a keresési teret, mivel az optimális megoldás meghatározása során elegendő csupán a kombinatorikusan lehetséges megoldásstruktúrákat megvizsgálnunk.

AZ MSG ALGORITMUS (Maximal Structure Generation)

A szintézis feladathoz tartozó maximális struktúrát a P-gráf módszertan az MSG algoritmus segítségével határozza meg. Az algoritmus a fent már ismertetett axiómákon alapszik. Számítógépes implementációjának fő lépései:

1.) definiáljuk magát a (P, R, O) szintézis-feladatot az M anyaghalmaz, a P termékhalmaz, az R nyersanyaghalmaz illetve az O, a lehetséges műveleti egységek halmazának megadásával. Az M halmaznak nem csak a köztes anyagokat, de a termékeket és a nyersanyagokat, azaz P és R elemeit is tartalmaznia kell.
2.) meghatározunk egy kezdeti struktúrát, vagy kiindulási hálózatot úgy, hogy összekapcsoljuk a műveleti egységeket a hozzájuk tartozó közös anyag típusú csúcsok mentén.
3.) eltávolítjuk a hálózatból azokat az anyagokat és műveleti egységeket, amelyek nem lehetnek részei a maximális struktúrának, mert megsértik az axiómák valamelyikét. Ez az algoritmus redukciós szakasza. Kikerülnek azok a műveleti egységek, amelyek nyersanyagokat gyártanak, illetve azok az anyagok, amelyek nem számítanak nyersanyagnak, de mégsem állítja elő őket semmi. Ez iteratívan történik, mivel előfordulhat, hogy bizonyos anyagok és műveleti egységek eltávolítása újabb anyagok és műveleti egységek eltávolítását vonja maga után.
4.) az algoritmus építő szakasz fázisában lépésről lépésre építjük fel a hálózatot a termékektől kiindulva először azokat a műveleti egységeket hozzáadva, amelyek a termékeket gyártják, majd azokat a műveleti egységeket, amelyek ezek bemeneteit gyártják, egészen addig, amíg el nem érünk a nyersanyagokig. Ha szemléletesen szeretnénk fogalmazni, akkor a lebontás felülről lefelé, a nyersanyagoktól a termékek felé haladva, az építés pedig alulról fölfelé, azaz a termékektől a nyersanyagok felé haladva történik.

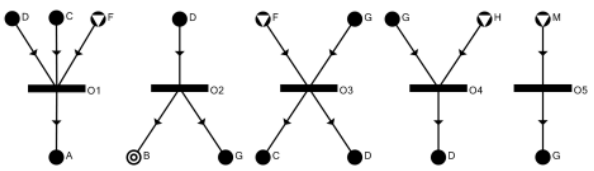
A szuperstruktúrát persze nem elég önmagában meghatározni, hanem lehetőleg minél hatékonyabban szeretnénk ezt megtenni. Az MSG ennek is eleget tesz, mivel a futási ideje polinomiális. Az MSG algoritmus működését az alábbi hipotetikus példán szemléltetjük:

$$M = \{A, B, C, D, F, G, H, L, M\}$$

$$P = \{B\}$$

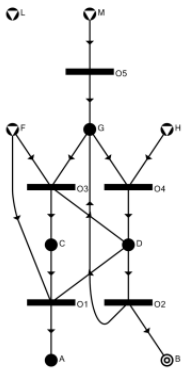
$$R = \{F, H, L, M\}$$

$$O = \{(\{C, D, F\}, \{A\}), (\{D\}, \{B, G\}), (\{F, G\}, \{C, D\}), (\{G, H\}, \{D\}), (\{M\}, \{G\})\} = \{O_1, O_2, O_3, O_4, O_5\}$$



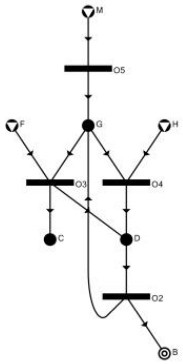
5.10. ábra. A példához tartozó műveleti egységek

A gyártás során a B terméket szeretnénk előállítani a rendelkezésre álló F, H, L, M nyersanyagokból. Definíciónak megfelelően, P és R elemei szerepelnek az M halmazban is.



5.11. ábra. A szintézisfeladathoz tartozó kiindulási struktúra

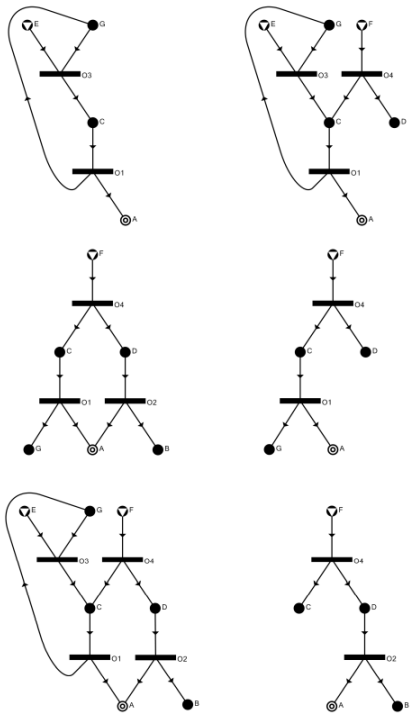
A kiindulási hálózatot a műveleti egységek közös kimeneteik és bemeneteik összekapcsolásával kapjuk. Ebben a példában O3-at és O1-et C-n keresztül, O3-at és O4-et O1-hez és O2-höz D-n keresztül végül O2-t és O5-öt O3-hoz és O4-hez G-n keresztül kapcsoljuk össze. Az eredményül kapott kiindulási hálózat: 5.11 ábra. A redukciós szakaszban az L nyersanyagot kivesszük a struktúrából, mert sérti az



5.12. ábra. A szintézisfeladathoz tartozó maximális struktúra

ötös axiómát, illetve az O1 műveleti egységet is, mert sérti a négyes axiómát. Az O1 műveleti egység eltávolításából következik, hogy az A anyag sem marad része a struktúrának, mivel nem lesz, ami előállítja. A többi anyag és műveleti egység nem sért egyetlen axiómát sem, részei maradnak a struktúrának. Az eredményül kapott struktúra az 5.12 ábrán látható, amely megfelel a maximális struktúrának is, amit az algoritmus építő szakaszának végrehajtása után kapunk.

AZ SSG ALGORITMUS (Solution Structure Generation)



5.13. ábra. Az SSG által generált megoldás-struktúrák

Az MSG eredményeként kapott maximális struktúra tartalmazza az összes kombinatorikusan lehetséges megoldásstruktúrát, amelyek alkalmasak annak a gyártási folyamatnak a modellezésére, amely során a megadott termékeket állítjuk elő a rendelkezésre álló nyersanyagokból. A maximális struktúra tartalmazza többek között a megadott célfüggvény, általában a költség szerinti optimális megoldást is. Nincs azonban általánosan elfogadott szabály arra, hogy melyik megoldás az optimális. A szükséges további vizsgálathoz ad segítséget az SSG algoritmus, amely minden kombinatorikusan lehetséges struktúrát, közte a maximális struktúrát, amely maga is megoldásstruktúra, pontosan egyszer generál. Az algoritmus egyik számítógépes megvalósítás a döntési leképezéseken alapszik. A döntési leképezések során arról döntünk, hogy mely anyagot mely műveleti egységgel vagy egységekkel gyártunk, azaz mely műveleti egységeket vonjuk be egy adott megoldás-struktúrába, így arról is döntünk, mely műveleti egységeket zárjuk ki az adott struktúrából. A döntések során ügyelnünk kell a konzisztenciára is, ha egy műveleti egységről már döntöttünk egy anyagra vonatkozóan, hogy nem kerül be a struktúrába, akkor egy másik anyagra vonatkozó döntés során már nem választhatjuk be. Egy struktúrában szereplő műveleti egységnek minden kimeneti anyagát elő kell állítania. Az SSG algoritmus döntési leképezésen alapuló megvalósítása rekurzívan hívja magát. Az algoritmus működését ábrázoló hipotetikus példa:

$$M = \{A, B, C, D, E, F, G\}$$
$$P = \{A\}$$
$$R = \{E, F\}$$
$$O = \{(\{C\}, \{A, G\}), (\{D\}, \{A, B\}), (\{E, G\}, \{C\}), (\{F, C\}, \{D\})\} =$$
$$= \{O_1, O_2, O_3, O_4\}$$

A VEGYES- EGÉSZ MATEMATIKAI PROGRAMOZÁSI MODELL

A modellben szereplő folytonos változókat x-el, a bináris változókat pedig y-al jelöljük. Ezeket a változókat a műveleti egységekhez rendeljük hozzá. Az xi folytonos változó jelöli az Oi műveleti egység méretét vagy kapacitását, az yi bináris változó pedig azt, hogy az adott műveleti egység szerepel-e a struktúrában, vagy sem (y=1 szerepel, y=0 nem). Ha yi = 1, akkor a műveleti egység kapacitását reprezentáló xi folytonos változó bármilyen értéket felvehet 0, és a műveleti egység felső kapacitáskorlátja, Ui között. Formálisan: $x_i \leq y_i U_i$ ahol Ui az Oi műveleti egység kapacitására vonatkozó felső korlát. Ha a feladatban nincs ilyen korlát definiálva, akkor Ui helyett egy tetszőleges, megfelelően nagy M szám használható. A célfüggvényt a költség minimalizálására írjuk fel. Folyamat teljes költsége = beruházási költségek összege + műveleti egységek működtetésének a költsége+ nyersanyagok ára. A modellben a műveleti egységek költségét egyszerűen az a + bx összefüggéssel adjuk meg, ahol „x” jelöli az adott műveleti egység méretét vagy kapacitását, az „a” fix költséget, „b” pedig a proporcionális költséget.

xi ≤ yiUi korlátozási mellett további feltételeket:

- anyagegyensúlyokra (köztes anyagokra, mindegyikből min. annyit gyártsunk, amennyit felhasználunk később)
- termékekre (ált. alsó korlát, min. előállítandó mennyiség)
- nyersanyagokra (felső korlát, max. ennyink van).

Vegyes-egész matematikai programozási modellt az előző, SSG szemléltetését bemutató példára: Szükséges információk:

- melyik műveleti egység milyen anyagokat használ fel illetve állít elő,
- miből mennyit (5.1 az anyagoknál zárójelben).
- műveleti egységek költségparaméterei (5.2)
- nyersanyagok árai (5.3)
- termékekre és nyersanyagokra vonatkozó megkötések (5.4).

Így az alábbi matematikai programozási modell írható fel. →

5.1. táblázat. A műveleti egységek kimeneti és bemeneti anyagai

Műveleti egységek	Bemenetek	Kimenetek
O_1	C(5)	A(4),G(1)
O_2	D(9)	A(8),B(1)
O_3	E(4),G(1)	C(5)
O_4	F(10)	C(1),D(9)

5.2. táblázat. A műveleti egységek költségparaméterei

Műveleti egységek	Fixköltség	Arányos költség
O_1	4	1
O_2	3	1
O_3	2	1
O_4	2	0,5

5.4. táblázat. A termékekre és nyersanyagokra vonatkozó feltételek

Termék	Feltétel
A	≥ 4
Nyersanyag	Feltétel
E	≤ 10

A feladathoz tartozó költségfüggvény:

$$\min \quad x_1+4y_1+x_2+3y_2+4.2x_3+2y_3+16.5x_4+2y_4$$

Az anyagegyensúly feltételek a köztes anyagokra:

$$\begin{aligned} C \text{ anyag:} \quad & -5x_1+5x_3+x_4 \geq 0 \\ D \text{ anyag:} \quad & -9x_2+9x_4 \geq 0 \\ G \text{ anyag:} \quad & x_1-x_3 \geq 0 \end{aligned}$$

A termékre vonatkozó feltétel:

$$A \text{ termék:} \quad 4x_1+8x_2 \geq 4$$

A nyersanyagokra vonatkozó megkötések:

5.3. táblázat. A nyersanyagok árai

Nyersanyag	Ár
E	0,8
F	1,6

$$E \text{ nyersanyag:} \quad 4x_3 \leq 10$$

További, a műveleti egységekre vonatkozó feltételek:

$$\begin{aligned} O_1: \quad & x_1 \leq y_1 M \\ O_2: \quad & x_2 \leq y_2 M \\ O_3: \quad & x_3 \leq y_3 M \\ O_4: \quad & x_4 \leq y_4 M \end{aligned}$$

Az F nyersanyagra vonatkozóan nem adtunk meg korlátozó feltételt.

AZ ABB ALGORITMUS (Accelerated Branch-and-Bound)

Mint vegyes-egész programozási feladatnak, ennek a modellnek a megoldására is használhatóak az általános branch-and-bound típusú módszerek. Bár a feladat optimális megoldása ezekkel a módszerekkel is meghatározható, hatékonyságuk tovább javítható, mivel a megoldás keresésekor nem veszik figyelembe a szintézis feladatok speciális tulajdonságait. A P-gráf módszertan az optimális megoldás meghatározására ennek megfelelően egy speciális korlátozás és szétválasztás típusú algoritmust, az ABB-t használja. Az ABB algoritmusnak az SSG algoritmushoz hasonlóan többféle implementációja létezik. Az egyik megvalósítás a már SSG-nél is említett döntési leképezéseken alapszik. Ennek a megközelítésnek az az előnye, hogy az ABB csak a kombinatorikusan lehetséges struktúrákat vizsgálja az optimális megoldás keresése során. Az algoritmus alkalmaz egyéb kombinatorukis gyorsításokat is, mint például az úgynevezett neutrális kiterjesztést.

NEVEZETES FELADATOK MEGOLDÁSA FOLYAMATSZINTÉZISSEL

A folyamatszintézis eszközeivel megoldhatóak azok a feladatok is, melyek elemi gráf algoritmusokkal. A gráf leképezhető folyamatgráfként, a keresés célja pedig optimalizálási célként. A gráf minden esetben hasonló módon írható át a folyamatszintézis nyelvére. A csúcsokhoz anyagokat, az irányított élekhez pedig műveleti egységeket rendelünk. A csúcsok egy {1, 2, 3, 4, 5, 6} halmazát például az M = {m1, m2, m3, m4, m5, m6} anyaghalmaz jelöli, míg az irányított élek egy {(1, 2), (1,3), (2,3)} halmazát az O = {o12 = ({m1},{m2}), o13 = = ({m1},{m3}), o23 = ({m2},{m3})} műveleti egység halmaz jelöli. Irányítatlan gráf esetén az élek mindkét lehetséges irányát külön-külön rögzíteni kell. Tehát irányítatlan élek egy {{1, 2}, {1,3}} halmazát az O={({m1},{m2}), ({m2},{m1}), ({m1},{m3}), ({m3},{m1})} műveleti egység halmaz adja meg. Az algoritmusok célját az anyagokra és műveleti egységekre vonatkozó korlátokkal és költség paraméterekkel tudjuk kijelölni.

MINIMÁLIS FESZÍTŐFA SZINTÉZISE

5.5. táblázat. Anyagok

Név	Típus	Alsó korlát	Felső korlát
m1	nyersanyag		5
m2	termék	1	
m3	termék	1	
m4	termék	1	
m5	termék	1	
m6	termék	1	

5.6. táblázat. Műveleti egységek

Név	Bemenő anyag	Kimenő anyag	Fix költség
o12	m1	m2	2
o21	m2	m1	2
o13	m1	m3	3
o31	m3	m1	3
o23	m2	m3	1
o32	m3	m2	1
o24	m2	m4	4
o42	m4	m2	4
o25	m2	m5	3
o52	m5	m2	3
o35	m3	m5	2
o53	m5	m3	2
o45	m4	m5	3
o54	m5	m4	3
o46	m4	m6	6
o64	m6	m4	6
o56	m5	m6	1
o65	m6	m5	1

Összefüggő gráf (Ami ezt biztosítja: egyik csúcsot forrásként(nyersanyagként), többit nyelőként (kötelező termék) definiálunk), minden csúcsot tartalmaz, az élek súlyának összege minimális. A termékekre szükséges pozitív mennyiségi alsó korlát beállítása, hogy a hozzá vezető műveletek legalább egyikének mérete ne lehessen nulla. Ezután az élek súlyát, mint a leíró műveleti egységek fix költségét adjuk meg. Kruskal algoritmus bemutatásánál szereplő feladat folyamatszintézis átírása az 5.5 és 5.6 táblázatokban látható. A feszítőfát azok az élek adják, melyekhez tartozó műveleti egységek szerepelnek a szintézis feladat optimális megoldásában. Ilyen megoldást ad például az ABB algoritmus szoftver megvalósítása.

LEGRÖVIDEBB ÚT SZINTÉZISE

5.8. táblázat. Műveleti egységek

Név	Bemenő anyag	Kimenő anyag	Fix költség
o12	m1	m2	1
o21	m2	m1	1
o13	m1	m3	3
o31	m3	m1	3
o23	m2	m3	1
o32	m3	m2	1
o24	m2	m4	4
o42	m4	m2	4
o25	m2	m5	3
o52	m5	m2	3
o35	m3	m5	2
o53	m5	m3	2
o45	m4	m5	3
o54	m5	m4	3
o46	m4	m6	6
o64	m6	m4	6
o56	m5	m5	1
o65	m6	m5	1

5.7. táblázat. Anyagok

Név	Típus	Alsó korlát	Felső korlát
m1	nyersanyag		5
m2	köztes anyag		
m3	köztes anyag		
m4	köztes anyag		
m5	köztes anyag		
m6	termék	1	

A hasonlóan határozható meg, mint a feszítőfa, de a legrövidebb úthoz csak a kiindulási és a cél csúcsot összekötő gráfot kell meghatározni, melyben az élek súlyának összege minimális. Így a folyamatszintézis feladatban a kiindulási csúcsot adjuk meg nyersanyagként, a cél csúcsot pedig kötelező termékként valamely pozitív előállítandó mennyiséggel. Az élek súlyát ismét a műveleti egységek fix költségeként definiáljuk.

A Dijkstra algoritmusnál bemutatott példára a folyamatszintézis feladat (5.7 és 5.8). A legrövidebb utat azok az élek adják, melyekhez tartozó műveleti egységek szerepelnek a szintézis feladat optimális megoldásában. Ilyen megoldást ad például az ABB algoritmus szoftver megvalósítása.

MAXIMÁLIS FOLYAM SZINTÉZISE

5.9. táblázat. Anyagok

Név	Típus	Alsó korlát	Felső korlát	Ár
m1	nyersanyag		∞	0
m2	köztes anyag		0	
m3	köztes anyag		0	
m4	köztes anyag		0	
m5	köztes anyag		0	
m6	termék		∞	1

5.10. táblázat. Műveleti egységek

Név	Bemenő anyag	Kimenő anyag	Kapacitás felső korlát
o12	m1	m2	1
o13	m1	m3	3
o32	m3	m2	1
o24	m2	m4	4
o25	m2	m5	3
o35	m3	m5	2
o45	m4	m5	3
o46	m4	m6	6
o56	m5	m5	1

A gráf éleihez rendelt tulajdonság a kapacitás, ami mellett a lehető legnagyobb mennyiséget kell egy forrásból a nyelőbe juttatni. Az éleket reprezentáló műveleti egységeknek maximális kapacitása lesz, mely megegyezik az megfelelő él kapacitásával. Motiváció: a nyelőn megjelenő anyagmennyiséget egy olyan termékkel írjuk le, aminek van egy pozitív ára, minél nagyobb a termelés, annál nagyobb a profitot. Hogy az éleken ne folyjon olyan anyagmennyiség, melynek nincs szerepe a maximális folyamban, mert csak közbülső csúcshoz vezet, korlátozzuk nullára a közbülső anyagpontokon megmaradó anyagmennyiséget az anyag felső korlátjával. A Ford-Fulkerson algoritmusnál megismert példát megadó szintézis feladat leíarása (5.9 és 5.10).

A maximális folyamot a szintézis feladat optimális megoldásában szereplő műveleti egységek által reprezentált élek adják, felhasznált kapacitásuk pedig megegyezik a megfelelő műveleti egységek optimális kapacitásával. Ilyen megoldást ad például az ABB algoritmus szoftver megvalósítása.

6. FEJEZET – ÜTEMEZÉS

Ha a rendszer tartalmaz olyan konkurens folyamatokat, melyek ugyanazokat az erőforrásokat igénylik és nem áll rendelkezésre megfelelő számú erőforrás, akkor az erőforrások ütemezésével lehet a folyamatokat kiszolgálni.

Egy ütemezési feladat többféleképpen definiálható. A fejezet első felében ütemezési feladatnak nevezzük azt a feladatot, ahol a lehető legrövidebb idő alatt kell adott mennyiségű termékeket előállítani a rendelkezésre álló szakaszos működésű berendezések felhasználásával, ahol egy terméket taszkok adott sorrendű végrehajtásával lehet előállítani. A gyakorlatban egy taszk elvégzésére több berendezés is alkalmas lehet, a közülük való választás is az ütemezési feladathoz tartozik. A fejezet második részében adott időn belül kell majd a lehető legnagyobb profitot elérni.

Egy szakaszos működésű rendszerben a **taszkokra** a következő tulajdonságok teljesülnek:

- Adott bemenetből adott kimenetet állít elő.
- A teljes bemenetnek a taszk indulásakor rendelkezésre kell állnia.
- A kimenet csak a taszk befejezése után áll rendelkezésre.
- Egy taszk adott ideig tart, ahol az idő függhet a használt berendezéstől.
- Nem megszakítható.

Egy ütemezési feladat matematikai modelljének megfelelő leírásához rengeteg változót kell bevezetni, amely modell nehezen vagy gyakran egyáltalán nem oldható meg általános célú megoldó szoftverekkel.

FLOW SHOP, JOB SHOP, OPEN SHOP

Ütemezési feladatokat széles körű előfordulásuk és összetett jellegük miatt sokféle szempont alapján osztályozhatjuk, pl. a termékek elkészítéséhez szükséges műveletek, feladatok végrehajtásának módja alapján. Így megkülönböztethetünk „flow shop", „job shop" és „open shop" ütemezési feladatokat. A taszkokat ebben a fejezetben műveleteknek (operation) hívjuk, ezek csoportját pedig munkának (job), amely egy termék előállításához szükséges műveleteket tartalmazza a gyártás sorrendjében. Feltételezzük, hogy a különböző munkákhoz tartozó műveletek között nincs sorrendi feltétel, kivéve azokat a műveletek, amelyeket ugyanazon gépekkel (machine) lehet elvégezni. A fejezetben a következő feltételezésekkel élünk:

1. A műveletek nem megszakíthatóak, minden művelethez pontosan egy végrehajtó gép tartozik
2. Nem lehet egy géppel egyszerre több műveletet végezni.

- J = {J1, … , Jn} a munkák halmaza
- M = {M1, . . . , Mm} a gépek halmaza.
- Minden Jj munka cj darab műveletből áll
- (O1j, O2j, . . . , Ocj,) ahol cj munkánként eltérő lehet, és cj lehet nagyobb mint m (mert lehetnek olyan műveletei egy munkának, amelyeket ugyanaz a gép hajt végre)
- pij : Oij művelet működési ideje
- pj: Jj munka működési idejének vektora.
- mij (i=1, 2, . . . , cj , j=1, 2, . . . , n) gépek hozzárendelését műveletekhez számokkal jelöljük, ahol Oij műveletet az mij gépen kell elvégezni.

A cél a műveletek optimális sorrendjének meghatározása úgy, hogy a rendszer teljes működési ideje minimális legyen.Ezekkel a jelölésekkel tudjuk definiálni a flow shop, job shop és open shop feladatokat.

Flow shop:

- minden munka minden gépen végigmegy pontosan ugyanolyan sorrendben. Az általánosság elvesztése nélkül feltételezhetjük, hogy a továbbiakban a sorrend M1, M2, . . . , Mm.
- Tipikus példa rá a futószalag, ahol a munkások és a munkaállomások jelentik a gépeket.
- A feladat jellemezhető a következőképen cj =m és mij =i (i = 1, 2, . . . , m, j = 1, 2, . . . , n).

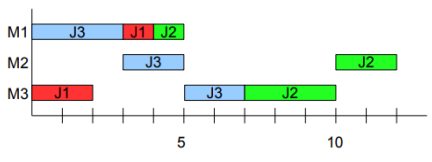
Job shop:

- nincsenek megkötések a műveletek számára és a hozzájuk rendelt gépekre.
- A gépek sorrendje minden munkára különböző lehet, valamint a felhasznált gépek is munkánként különbözhetnek.
- A munkák műveleteinek a sorrendje rögzített.

Open shop: (hasonló a job shophoz)

- a munkák műveleteit bármilyen sorrendben végre lehet hajtani a megfelelő gépeken,
- a munkákhoz tartozó műveletek sorrendje nincs megkötve, de mindet el kell végezni

6.1. példa. Egy job shop feladatban három munka van, amelyek sorrendben kettő, három és három műveletből állnak. Az első munka két műveletből áll, ahol az első munkát a 2- es géppel lehet elvégezni 2 időegység alatt, a másodikat pedig az 1-es géppel 1 időegység alatt. A második munka három műveletből áll, ahol a gépek sorrendje 1, 3, 2, a hozzájuk tartozó működési idő pedig sorrendben 1, 3, 2 időegység. A harmadik munka szintén három műveletből áll, itt a gépek sorrendje 1, 2 és 3, a működési idők pedig sorrendben 3, 2, 2. A korábbiakban bevezetett jelölésekkel ez az alábbi formában írható le:



6.1. ábra. A 6.1. példa egy megoldása Gantt diagrammal

$$J_1 : (m_{11}, m_{21}) = (2, 1) \qquad p_1 = (2, 1)$$
$$J_2 : (m_{12}, m_{22}, m_{32}) = (1, 3, 2) \qquad p_2 = (1, 3, 2)$$
$$J_3 : (m_{13}, m_{23}, m_{33}) = (1, 2, 3) \qquad p_3 = (3, 2, 2)$$

(szerintem itt szar az ábra, mert a J1nek semmi köze az M3hoz, az szerintem az M2nél kellene hogy legyen)

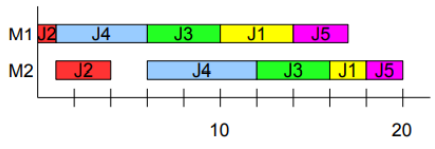
Pl.: az 1-es gép (M1) 0 időpillanatban elkezd dolgozni a J3 munkán, ami tart 3 időegységig, majd 1 időegységet tölt a J1 munkával, végül J2 munkának a műveletén dolgozik 4 időpillanattól kezdve az 5 időpillanatig.

Az egy munkához tartozó műveleteket nem lehet egyszerre elvégezni (Gannt diagrammok rajzolásnál figyeljünk, hogy egy géphez tartozó műveletek nem fedhetik egymást), valamint flow shop és job shop esetén a munkához tartozó műveletek sorrendje a feladatban rögzített. Flow shop feladatoknál egyszerű megmutatni, hogy ha minden működési idő pozitív, akkor van olyan optimális megoldás, ahol a munkák működési sorrendje megegyezik az első két műveleten, valamint megegyezik a működési sorrend az utolsó két műveleten is. Ebből következik, hogy háromgépes flow shop feladatoknál ugyanaz a gépek használati sorrendje a munkákhoz, mivel az elsőn ugyanaz a sorrend mint a másodikon valamint a másodikon ugyanaz a sorrend, mint a harmadikon. Ha legalább négy gép van, akkor lehet olyan optimális megoldás, hogy a gépek működési sorrendje különbözik a munkákon. 2-gépes flow shop feladatok hatékonyan megoldhatóak Johnson algoritmusát használva:

begin
$N_1 \leftarrow \{j : p_{1j} < p_{2j}\}$
$N_2 \leftarrow \{j : p_{1j} \geq p_{2j}\}$
rendezzük N_1-et nem csökkenő sorrendbe p_{1j} szerint
rendezzük N_2-t nem növekvő sorrendbe p_{2j} szerint
optimális megoldásban a rendezett N_1 halmaz elemeit a rendezett N_2 követi
end

6.2. példa. Adott egy öt munkát tartalmazó két gépes flow shop feladat a

- munkák működési idők vektorai: p1=(4, 2), p2=(1, 3), p3=(4, 4), p4=(5, 6), p5=(3, 2).
- Ezek alapján felírhatjuk a halmazokat: N1 = {2, 4}, N2 = {1, 3, 5}.
- Majd ezeket rendezzük: N1 = {2, 4}, N2 = {3, 1, 5} és a rendezett halmazokat összefűzve végül
- megkapjuk az optimális megoldást: 2, 4, 3, 1, 5



6.2. ábra. A 6.2. példa optimális megoldása

Több speciális flow shop feladat (amelyben több, mint két gép van) hatékonyan megoldható specializált algoritmusokkal. Egy nagy része a feladatoknak kezelhető a Johnson algoritmussal, vagy annak kis változtatásával. Például Johnson bebizonyította, hogy három gépes esetben, ha a második (középső) gép legnagyobb működési ideje jóval kisebb, mint az első vagy az utolsó gép legkisebb ideje (az uralkodik rajta), akkor egy optimális megoldást kaphatunk a Johnson algoritmus segítségével, ha a második gépet „összevonjuk” azzal, amelyik uralkodik rajta.

ÜTEMEZÉSI FELADAT ÁLTALÁNOS MEGFOGALMAZÁSA

A továbbiakban a követező alapfogalmakat használjuk:

• Terméknek (product) nevezzük a gyártó rendszerben előállítani kívánt anyagot.

• A berendezés (equipment unit) a termékek előállításához felhasználható eszköz, erőforrás. A berendezés megújuló erőforrás, azaz használat után újra rendelkezésre áll.

• A taszk (task) egy olyan tevékenység, amely nem bontható résztevékenységekre és adott idő alatt adott bemenetekből adott kimeneteket generál. Egy taszk végrehajtásához egy vagy több berendezés áll rendelkezésre, ahol a végrehajtási idő függ a felhasznált berendezéstől. Egy berendezés egy időben csak egy taszkhoz rendelhető, azaz nem lehet egy berendezéssel egyszerre több taszkot végrehajtani. Egy taszkhoz csak egy berendezés rendelhető, azaz a taszkok nem megoszthatóak.

• A recept (recipe) tartalmazza azokat a minimális információkat, melyek a termékek előállításához szükségesek. Ezek a termékek előállításához szükséges taszkok hálózata, a taszkokhoz rendelhető berendezések (a működési idővel együtt) és az előállítandó termékmennyiségek. A receptben, az eddigiektől eltérően lehetnek több bemenettel illetve kimenettel rendelkező taszkok is, azaz a termékek előállítása nem feltétlenül egy vonal mentén történik.

• Szakaszos termelő rendszerekben a termelés adagokban (batch) történik. Ez azt jelenti, hogy ha az előállítandó termék mennyisége több, mint amennyi a recept egyszeri végrehajtása során keletkezik, akkor a recept többszöri megismétlésével állítják elő a kívánt termékmennyiséget. A recept egyszeri végrehajtását, ütemezését jelenti egy adagnyi termék előállítása.

6.3. példa. Tekintsük a következő ütemezési feladatot, ahol három terméket kell előállítani három lépésben. A termékek előállítási módjai (receptjei) és a felhasználható berendezések a 6.1 táblázatban láthatóak (például az A termék első taszkjához az E1 berendezés használható, melynek működési ideje 6 perc). A szükséges termékmennyiségek adagok számával adottak a 6.2 táblázatban.

6.1. táblázat. Recept a 6.3. példához

	A termék		B termék		C termék	
Taszk	Berendezés	Idő [perc]	Berendezés	Idő [perc]	Berendezés	Idő [perc]
1	E1	6	E3	9	E2	7
2	E2	9	E3	15	E1	17
3	E1	14	E2	16	E3	8

6.2. táblázat. Adagok száma a 6.3. példához.

Termék	A	B	C
Batch-ek száma	1	1	1

Meg kell még határozni, hogy van-e lehetőség a keletkező köztes termékek tárolására a taszkok között. Ilyen lehet egy tároló berendezés vagy maga az anyagot előállító berendezés. Ha van ilyen lehetőség, akkor fontos, hogy milyen mennyiségben lehet tárolni az anyagot és a tárolókat hol (mely taszkok vagy berendezések között) lehet használni; ha nincs, akkor a gyártó berendezésben várakozhat-e az anyag. Ezek alapján a következő tárolási stratégiák léteznek:

• **Nincs lehetőség köztes tárolásra** (A köztes termék tárolásához nem áll rendelkezésre tároló):

◦ NIS tárolási stratégia (non intermediate storage policy): A taszk elvégzése után a köztes terméket az azt előállító berendezésben tárolhatjuk, ilyenkor a berendezést csak akkor lehet használni egy másik taszk végrehajtásához, ha a keletkezett anyagot már áttöltöttük egy másik, értelemszerűen a következő taszkot végrehajtó berendezésbe. Az egyik leggyakoribb eset.

◦ ZW tárolási stratégia (zero wait policy): az őt előállító berendezésben sem tárolhatjuk. Ilyenkor az anyagot azonnal át kell tölteni a következő berendezésbe. Ez gyakran előfordulhat olyan esetekben, amikor nem stabil anyagot gyártunk, vagy az anyag tulajdonságai megkövetelik.

• **Van lehetőség köztes tárolásra:**

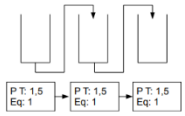
◦ UIS tárolási stratégia (unlimited intermediate storage policy): A köztes termékeket „végtelen” mennyiségben lehet tárolni. A taszk elvégzése után a berendezésből a köztes terméket tároljuk egy, gyakorlati szempontból „végtelen” kapacitású tárolóban, azaz a berendezést tisztítás után rögtön tudjuk használni. Ilyen tároló lehet egy megfelelően nagy raktárhelyiség, ahol a keletkezett félkész termékek elhelyezhetők.

◦ FIS tárolási stratégia (finite intermediate storage policy): A rendszer véges számú és méretű tárolót tartalmaz. Minden tároló csak két előre meghatározott berendezés között használható. Szintén FIS-nek nevezik azt az esetet, amikor a tárolók nem berendezésekhez rendeltek, hanem anyagokhoz. Azaz egy tároló berendezés csak bizonyos típusú anyagot tárolhat.

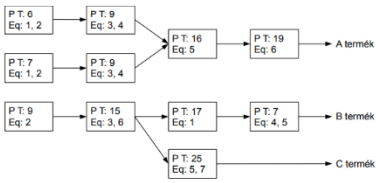
◦ CIS tárolási stratégia (common intermediate storage policy): Ebben az esetben is véges számú és méretű tároló áll rendelkezésre. Ez a tárolási stratégia abban különbözik a FIS stratégiától, hogy itt a tárolók helye nem rögzített, valamint arra is figyelni kell, hogy ha több anyagot tárolunk benne, akkor a tárolások között ki kell tisztítani a berendezést.

◦ MIS tárolási stratégia (mixed intermediate storage policy): Az előző négy tárolási stratégia keveréke, a rendszer különböző pontjain különböző stratégiák lehetnek.

S-GRÁF LEÍRÁS

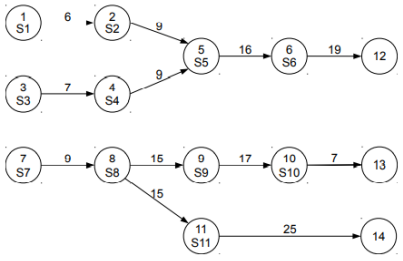


6.3. ábra. Három lépésben előállítható termék előállítási módja és receptjének hagyományos leírása.



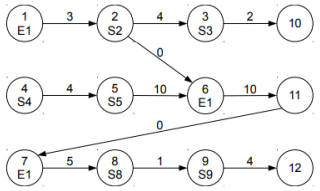
6.4. ábra. Három termék előállításának hagyományos leírása.

A termékek előállítását leíró receptet hagyományosan lehet ábrázolni egy irányított gráffal, ahol a gráf csomópontjai jelentik a taszkokat, a közöttük lévő élek pedig ezek sorrendjét. A működési idők és a taszkokhoz rendelkezésre álló berendezések a megfelelő csomópontokban adottak. A 6.3 ábra mutatja egy három lépésben (három taszkkal) előállítható termék hagyományosan megadott előállítási módját és receptjének reprezentációját. Összetett receptek is leírhatóak ezen a módon, pl. a 6.4 ábrán látható receptben az A terméket két köztes anyag összekeverésével, majd a keverék további feldolgozásával állítható elő.



6.5. ábra. Gráf leírás a 6.4 ábrán látható recepthez.

Ebben a hagyományos leírásban az élek a taszkok sorrendjét adják meg, minden egyéb információ a csomópontokhoz van rendelve. A 6.4 ábrán látható receptnek a gráf leírása a 6.5 ábrán látható, ahol Si azoknak a berendezéseknek a halmaza, amelyekkel az i csomóponttal reprezentált taszkot végre lehet hajtani (például S1 = {E1, E2}), valamint egy-egy további csomópont jelöl minden terméket. Ebben a leírásban az élek súlya alsó korlátot jelent a két kapcsolódó taszk kezdési idejének különbségére. Egy taszk működési ideje a hozzá rendelhető berendezésektől függően különböző lehet, ebben az esetben az él súlya a felhasználható berendezések működési idői közül a legkisebb lesz. Ez az érték a feladat megoldása során természetesen változhat a berendezések taszkhoz való hozzárendelése során.



6.6. ábra. Az E1 berendezés 1-6-7 működési sorrendjének megadása NIS esetben

Bármely recept egy körmentes irányított gráffal ábrázolható. Tegyük fel, hogy minden termékre adott a legyártandó mennyiség, valamint adott, hogy melyik taszk melyik berendezéssel/berendezésekkel hajtható végre. A taszkok sorrendje ábrázolható egy irányított gráffal, ahol a sorrend élekkel adott úgy, hogy az élek kötik össze az egy berendezéshez tartozó taszkokat jelölő csomópontokat. Az **élek** a taszkok **működési idejével súlyozottak** és megadják ezek működési sorrendjét. Az élek súlya azért van súlyozva a működési idővel, mert ezekre az élekre is igaz, hogy alsó korlátot jelentenek a kapcsolódó taszkok kezdési idejének különbségére és a berendezésnek előbb végre kell hajtania az aktuális taszkot és csak utána kezdheti el a következőt. Ez a fajta leírás jól használható UIS típusú feladatok megoldására, de nem megfelelő NIS tárolási stratégia esetén. Az UIS stratégia feltételezi, hogy a berendezések a taszkok végeztével azonnal rendelkezésre állnak, azaz a köztes termékek, amelyek a taszk végrehajtása során keletkeznek, kikerülnek a berendezésből és eltárolásra kerülnek, amíg a következő taszk el nem kezdődik. A legtöbb szakaszos működésű folyamatban ezt nem lehet biztosítani, azaz NIS esetet kell figyelembe vennünk. NIS stratégiánál a berendezések nem állnak rendelkezésre egy taszk befejezése után egészen addig, amíg a tárolt anyag át nem töltődik a következő taszkot végrehajtó berendezésbe. A gráfban ezt a további feltételt is reprezentálni kell illetve lehet egy vagy több él segítségével a következő módon. Jelölje τ azokhoz a taszkokhoz tartozó csomópontokat, amelyek a recept szerint a j csomóponthoz tartozó taszkot közvetlenül követik. Ha az Ei berendezés a k taszkot végzi el közvetlenül a j után, akkor egy nulla súlyú (vagy váltási idővel súlyozott) élet húzunk τ minden eleméből k-ba. Ezt a fajta leírást nevezzük S-gráfnak. A 6.6 ábrán látható az E1 berendezés 1-6-7 működési sorrendjének megadása S-gráf segítségével. Az ábráról leolvasható, hogy az E1 berendezés először végrehajtja az 1 csomópont által reprezentált taszkot, majd megvárja a 2-es taszkot végrehajtó valamelyik berendezést (S2 halmaz) és áttölti bele az anyagot, utána végrehajtja a 6-os taszkot, amivel előállít egy terméket, végezetül a 7-es taszkot hajtja végre és áttölti az anyagot az S8 halmazból a 8-as taszkhoz rendelt berendezésbe. Mivel

a példában a váltási élek súlya mindenhol nulla, tehát nincs szükség az E1 berendezés tisztítására a következő taszk előtt.

Matematikai jelölések: egy irányított G gráfot egy (N, A) párral adhatunk meg, ahol N véges halmaz, a csomópontok halmaza és A az élek halmaza ($A \subseteq N \times N$). Egy S-gráfnak két típusú éle van (léteznek recept-élek és ütemezési-élek). Így egy S-gráfot G(N, A1, A2) formában adhatunk meg, ahol N a csomópontok, A1 a recept-élek, A2 pedig az ütemezésiélek halmaza feltéve, hogy $A1 \subseteq N \times N$, $A2 \subseteq N \times N$ és $A1 \cap A2 = \emptyset$; továbbá minden $(i, j) \in A1 \cup A2$ élhez tartozik egy nem negatív érték c(i, j), az él súlya. Ha egy él i-ből j-be mutat ($(i, j) \in A1 \cup \cup A2$), akkor az a gyakorlatban azt jelenti, hogy az j csomóponthoz tartozó taszk legalább c(i, j) idővel később kezdi a működését, mint az i csomóponthoz tartozó taszk. Speciális S-gráfot vezetünk be a recepthez (recept-gráf) és a megoldáshoz (ütemezési-gráf).

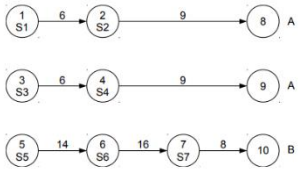
RECEPT-GRÁF

Egy ütemezési feladat receptje megadja minden termékhez a termékekhez tartozó taszkok sorrendjét, a köztük lévő anyagáramokat, valamint az egyes taszkokhoz felhasználható berendezések halmazát a hozzájuk tartozó működési idővel. Ezeknek az információknak a receptgráfban szerepelniük kell ahhoz, hogy a receptet megfelelően le tudjuk vele írni. A termékek gyártását leíró recept alapján a recept-gráf alapját jelentő taszk-hálózat felépítéséhez a következő lépéseket hajtjuk végre:

1. Rendeljünk egy-egy csomópontot minden taszkhoz (taszk-csomópont) és egy-egy csomópontot minden termékhez (termék-csomópont).
2. Egy él (recept-él) mutasson minden taszkot reprezentáló taszk-csomópontból a receptben utána következő taszkot reprezentáló taszk-csomópontba, valamint a terméket gyártó taszkokhoz tartozó csomópontokból a megfelelő termék-csomópontba.
3. Egy recept-él súlya legyen egyenlő az él kezdő csomópontjához tartozó taszk működési idejével abban az esetben, ha a taszkhoz csak egy berendezés áll rendelkezésre. Ha több berendezés is rendelkezésre áll, akkor a működési idők közül a legkisebbel egyezzen meg az él súlya.
4. Ha egy termékből nagyobb mennyiségre van szükség, mint amennyit a recept alapján egyszerre elő lehet állítani (több adagra van szükség), akkor a termék előállításában résztvevő taszkok taszk-csomópontjai, a termék-csomópont, valamint a közöttük lévő élek annyiszor kerülnek be a gráfba, ahány adag már minimálisan elegendő a szükséges anyagmennyiséghez.

Az így keletkezett gráfot hívjuk taszk-hálózatnak, ahol Nt jelöli a taszk-csomópontok, Np pedig a termék-csomópontok halmazát ($Nt \cap Np = \emptyset$). Fontos megjegyezni, hogy több adag esetén minden taszkhoz több taszk-csomópont is tartozik. Ha a feladat szempontjából fontos, hogy egy taszkhoz több csomópont is tartozik, akkor ezen csomópontok által reprezentált termelési folyamatot (és továbbiakban magát a csomópontot is) tevékenységnek nevezzük, egyébként a taszk kifejezést használjuk. A továbbiakban nem vesszük figyelembe a több bemenettel rendelkező taszkok bemeneteinek időzítését. Ebben az esetben a taszk-hálózat alkotja a recept-gráfot, ahol minden a receptben található strukturális információ adott. Ha G(N, A1, A2) egy recept-gráf, akkor körmentes és $A2 = \emptyset$. Legyen Ni($\subset$Nt) azon csomópontok halmaza, amelyek az i berendezéssel végrehajthatók. Feltehetjük, hogy minden taszkhoz létezik legalább egy berendezés, amivel végre lehet hajtani, azaz a taszk-csomópontok halmaza megadható a következő módon $Nt = N1 \cup N2 \cup \cdots \cup Nn$, ahol Ni és Nj (i, j = 1, 2, . . . , n) tartalmazhat azonos elemeket, azaz metszetük nem szükségszerűen üres.

6.4. példa. Tegyük fel, hogy két terméket kell előállítanunk, A-t és B-t, A-ból két adagot, Bből egyet. A-t két egymás utáni lépésben lehet előállítani, ahol az első lépés az S1, a második pedig az S2 halmazban lévő berendezések bármelyikével végrehajtható 6 illetve 9 időegység alatt. B-t három egymást követő lépésben lehet előállítani az S3, az S4 illetve az S5 halmazokban lévő berendezésekkel 14, 16 illetve 8 időegység alatt. A feladat recept-gráfja a 6.7 ábrán látható.

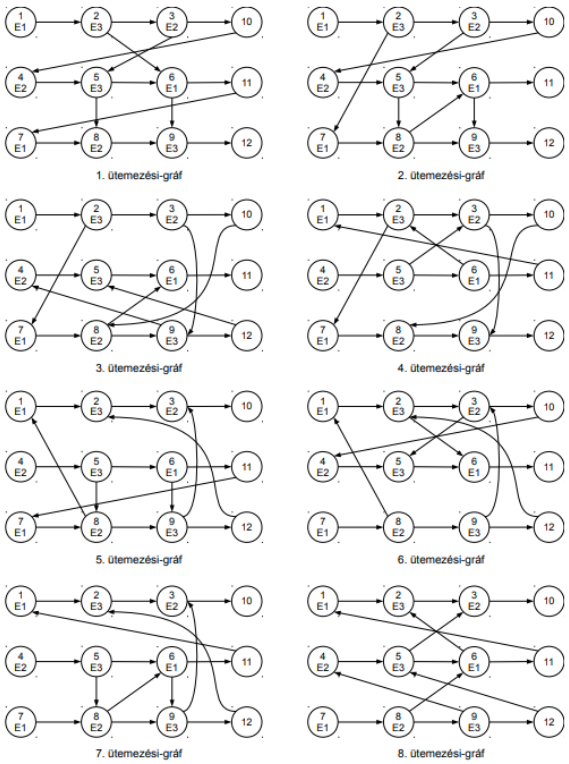


6.7. ábra. A 6.4. példa recept-gráfja: két adag az A termékből és egy a B-ből.

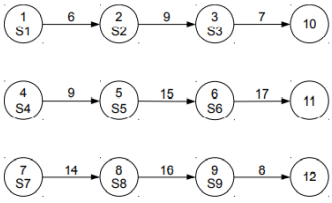
ÜTEMEZÉSI-GRÁF

Az ütemezési-gráf olyan speciális S-gráf, amely egy megoldást reprezentál; az ütemezési feladat minden megoldásához létezik egy ütemezési-gráf és ez a gráf minden megoldáshoz különböző. A G ′ (N, A1, A2) S-gráfot a G(N, A1, ∅) recept-gráfhoz tartozó ütemezési-gráfnak nevezünk, ha a recept-gráf minden csomópontja (taszkja) ütemezve van, figyelembe véve a berendezés-taszk hozzárendelést. Egy megfelelő keresési módszerrel az optimális megoldáshoz tartozó ütemezési-gráf és berendezés-taszk hozzárendelés megtalálható. Megfelelő keresési módszerrel nem csak az optimális, hanem az összes megoldás generálható.

6.5. példa. Három termék előállítása a 6.8 ábrán látható recept-gráf szerint történik.



6.9. ábra. A 6.5. példa ütemezési-gráfjai



6.8. ábra. Recept-gráf a 6.5. példához.

Minden taszkhoz egy berendezés áll rendelkezésre, ahol S1 = {E1}, S2 = {E3}, S3 = {E2}, S4 = {E2}, S5 = {E3}, S6 = {E1}, S7 = {E1}, S8 = {E2}, S9 = {E3}. A feladatnak nyolc különböző megoldása létezik, amelyek ütemezési-gráfjai a 6.9 ábrán láthatóak.

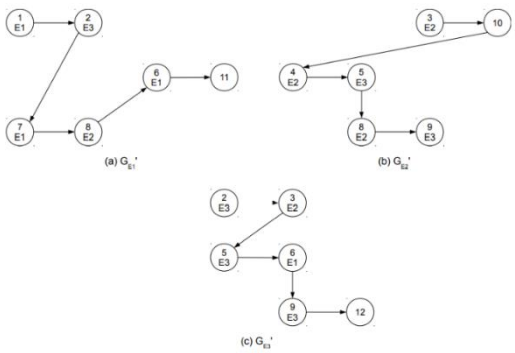
Az ütemezési-gráf formális leíráshoz tételezzük fel, hogy a G(N, A1, ∅) recept-gráfhoz adott egy G ′ (N, A1, A2) ütemezési-gráf, ahol A2 ⊆N×N. Jelölje Mi (i=1, 2, . . . , n) azoknak a csomópontoknak a halmazát, amelyekhez tartozó taszkokat az i berendezés hajtja végre, azaz azokat a csomópontokat, amelyekhez az i berendezést hozzárendeltük. Feltételezzük, hogy a megoldásban egy taszkot pontosan egy berendezéssel lehet végrehajtani, azaz Mi ∩ Mj = = ∅ (i j, i, j = 1, 2, . . . , n). Ebből következik, hogy M1, M2, . . . , Mn egy partícionálása a taszk csomópontok halmazának (Nt = N1∪N2∪· · ·∪Nn) úgy, hogy Mi ⊆ Ni (i = 1, 2, . . . , n). Ahhoz, hogy az ütemezési-gráf korrekt matematikai leírását meg tudjunk adni, szükségünk van egy speciális S-gráf bevezetésére, amelyet komponens-gráfnak nevezünk. A komponens-gráfok mutatják meg az egyes berendezéseknek a megoldáshoz tartozó működési sorrendjét. Ebből következik, hogy a komponens-gráf része az aktuális ütemezési-gráfnak. Formálisan az i berendezéshez tartozó komponens-gráf G ′ i (N ′ i , A1i, A2i) (⊆ G ′ (N, A1, A2)) egy olyan S-gráf (i = 1, 2, . . . , n), ahol

• N ′ i tartalmazza G ′ -ből az Mi összes csomópontját és az összes olyan csomópontot, amelybe Mi-ből induló recept-él mutat. Azaz tartalmazza azokat a csomópontokat, amelyhez az i berendezés hozzá lett rendelve, valamint a receptben ezeket követőket, ahova az elkészült anyagot át kell tölteni.

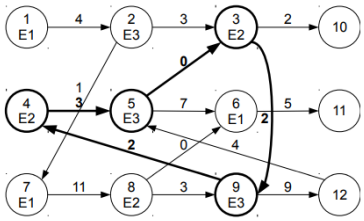
• $N_i' = M_i \cup \{k : k \in N \text{ és } \exists j \in M_i, \text{ hogy } (j,k) \in A_1\}$ A1i tartalmazza az összes olyan recept-élet G ′ -ből, amely Mi-ből indul. Azaz az összes olyan recept-élet, amely összeköti a i berendezéshez rendelt csomópontokat a receptben utána $A_{1i} = \{(j,k) : (j,k) \in A_1 \text{ és } j \in M_i\}$ következőkkel és amelyen a berendezés működési ideje szerepel.

• A2i tartalmazza az összes olyan ütemezési-élet G ′ -ből, amely A1i valamely élének végpontjából Mi valamelyik elemébe mutat. Azaz az összes olyan ütemezési élet, amely az i berendezés működési sorrendjére vonatkozó döntéseket jelöli. $A_{2i} = \{(j,k) : (j,k) \in A_2, k \in M_i \text{ és } \exists l \in M_i, \text{ hogy } (l,j) \in A_{1i}\}$

6.5. példa (további vizsgálat). A 2. ütemezési-gráf (6.9 ábra) komponens-gráfjai G ′ Ei (i = = 1, 2, 3) a 6.10 ábrán láthatóak a következő berendezés-taszk hozzárendeléssel: ME1 = = {1, 6, 7}, ME2 = {3, 4, 8}, ME3 = {2, 5, 9}.



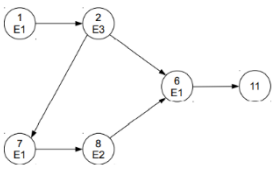
A komponens-gráf segítségével a következőképpen definiáljuk az ütemezési-gráfot. A G(N, A1, ∅) recept-gráfhoz és az Mi (i = 1, 2, . . . , n) berendezés-taszk hozzárendeléshez tartozó G ′ (N, A1, A2) S-gráf ütemezési-gráf, ha teljesíti a következő négy feltételt.



6.11. ábra. Irányított kör az S-gráfban.

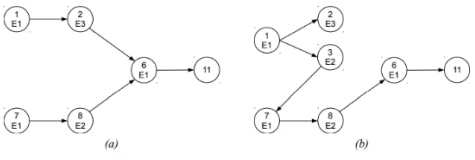
- (SG1) G ′ nem tartalmaz irányított kört.
- (SG2) G ′ i komponens-gráf által meghatározott rendezés teljes minden Mi∪{j} halmazon, ahol j ∈ N ′ i és i = 1, 2, …,n.
- (SG3) Az N ′ i (i = 1, 2, . . . , n) halmazok minden eleméből legfeljebb egy A2i-beli él indul.
- (SG4) A G ′ ütemezési-gráf megegyezik komponens-gráfjainak uniójával, azaz G ′ = ∪n i=1 G ′ i

Az ütemezési-gráfok és a megvalósítható ütemezések közötti kapcsolat azon alapul, hogy az (i, j) él az S-gráfban azt mutatja, hogy a j csomópont által reprezentált taszk leghamarabb c(i, j) idővel később kezdődhet el, mint az i által reprezentált. Ebből következik, hogy ha az (SG1) feltétel nem teljesül, akkor a megoldás nem megvalósítható, mivel vagy az időbeliség vagy a NIS tárolási stratégia feltételei nem teljesülnek. A 6.11 ábrán látható S-gráf irányított kört tartalmaz, mely vastag vonallal van jelölve. Ha a kör mentén végig nézzük az élek jelentését, akkor azt tapasztaljuk, hogy a 4-es taszknál 3 időegységgel később kezdheti el a munkáját az 5-ös, annál nem kezdődhet hamarabb a 3-as, azután leghamarabb 2-vel kezdődhet a 9-es, majd 2-vel a 4-es. Ez azt jelenti, hogy a 4-es taszk 7 időegységgel később kezdheti el a munkát a 4-es taszknál, azaz ellentmondáshoz jutottunk.



6.13. ábra. Felesleges ütemezési-élek az S-gráfban.

Tegyük fel, hogy (SG1) feltétel teljesül, de (SG2) nem. Ekkor két esetet különböztethetünk meg: nincs teljes rendezés az Mi halmazon, vagy teljes rendezés van az Mi halmazon, de van olyan j(∈ N ′ i), hogy ezzel kiegészítve Mi-t nincs teljes rendezés. Az első esetben, ha nincs teljes rendezés az Mi halmazon, akkor létezik két olyan taszk, amelyek működési sorrendje nem egyértelműen meghatározott. Például a 6.12a ábra S-gráfja teljesíti a komponens-gráf definícióját, de az E1 berendezéshez tartozó 1. és 7. csomópontok között nincs út, ezért ezek működési sorrendjéről nem tudunk semmit. A második esetben a NIS stratégia sérül. Például a 6.12b ábra komponens-gráfján látható, hogy bár az E1 berendezéshez tartozó csomópontok (1, 6, 7) között teljes a rendezés, de hiányzik egy ütemezési-él a 2-es és a 7-es csomópont között, így az E1 berendezésnek nem kell megvárnia a 2. taszkon dolgozó E3 berendezést az anyag áttöltéséhez. Az (SG3) és (SG4) feltételek teljesülése nem szükséges ahhoz, hogy az ütemezési-gráf által reprezentált megoldás megvalósítható és teljes legyen, de az optimális megoldás mindig megtalálható az (SG3) és (SG4) által leszűkített keresési térben. (SG3) biztosítja, hogy ne legyenek redundáns ütemezési-élek a gráfban, azaz a minimális számú éllel valósuljon meg az ütemezés. A 6.13 ábrán látható komponens-gráfban a 2. és 6. csomópontok közötti ütemezésiél redundáns, nem hordoz információt, mivel az E1 berendezés mindenképpen csak az 1 taszk végrehajtása után kezdheti meg a 6. taszkot, ha már a 7 taszkot is végrehajtotta. Az (SG4) feltétel pedig kiszűri azokat az éleket, amelyek nem tartoznak egyik berendezés ütemezéséhez sem, azaz minden él vagy a recept-gráf recept-éle vagy valamely berendezés ütemezéséhez tartozó ütemezési-él. Az előbbeikből következik, hogy az (SG3) és (SG4) feltételek teljesülése esetén az S-gráf minimális abban az értelemben, hogy él elhagyásával nem ír le megoldást.



6.12. ábra. Az SG2 szabály megsértése

ALGORITMUS ÜTEMEZÉSE S-GRÁFFAL

A fejezetben Sanmartí és társai által bevezetett alapalgoritmust mutatjuk be, amely általánosan használható bármilyen típusú ütemezési feladatra. Az algoritmus bemenetként megkapja a feladat recept-gráfját, kimenete pedig egy optimális ütemezés ütemezési-gráfja. Az algoritmusban a cél az optimális megoldás (ütemezési-gráf) megtalálása a receptből (recept-gráf) kiindulva részfeladatok (általános S-gráf) sorozatán keresztül. Definíció alapján a recept-gráf mindig részgráfja a hozzá tartozó összes ütemezési-gráfnak úgy, hogy a gráfok csomópontjai nem változnak, és a recept-éleknek is legfeljebb a súlya növekedhet a berendezés-taszk hozzárendelés alapján. Az előző fejezetben leírtak szerint egy ütemezési-gráf minden éle, amely nem tartozik a kiinduló recept-gráfhoz, ütemezési-él. Ha figyelembe vesszük az (SG1)- (SG4) feltételeket és a lehetséges berendezés-taszk hozzárendeléseket, akkor a recept-gráf összes, a feltételeknek megfelelő kiterjesztése ütemezési-élekkel elvezet bennünket az összes ütemezési-gráfhoz. Mivel a lehetséges kiterjesztések száma véges, az összes ütemezési-gráf a hozzá tartozó berendezés-taszk hozzárendelésekkel véges lépésben előállítható. Az S-gráf leírás jó alap egy szétválasztás és korlátozás (branch-and-bound, B&B) típusú algoritmushoz.

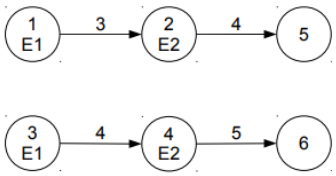
SZÉTVÁLASZTÁSI LÉPÉS

A szétválasztás és korlátozás algoritmus képes előállítani bármely recept-gráfhoz a hozzá tartozó összes ütemezési-gráfot, ahol minden részfeladathoz egy Sgráf és egy részleges berendezés-taszk hozzárendelés tartozik, valamint egy csomópont a keresőfában. A keresőfa gyökeréhez a recept-gráf tartozik, a leveleihez pedig ütemezési-gráfok. A szétválasztás során minden részfeladatnál kiválasztunk egy berendezést. A részfeladat minden gyerek részfeladatában a berendezést hozzárendeljük egy megfelelő, még be nem ütemezett taszkhoz (természetesen mindegyik részfeladatban másikhoz), valamint a kiválasztott taszkot beütemezzük a berendezés aktuálisan utolsó lépésének. Mivel a taszkok működési ideje függhet a felhasznált berendezéstől, ezért egy berendezés hozzárendelése egy taszkhoz módosíthatja a taszkot jelölő csomópontból kiinduló recept-él (vagy elágazás esetén receptélek) súlyát. Az egyszerűség kedvéért feltételeztük, hogy a megoldásban egy taszkot pontosan egy berendezés hajthat végre. Az algoritmus megvalósításánál nagy szabadságot ad a keresési stratégia kiválasztása. Például, hogy melyik berendezést választjuk ki ütemezésre nagyban befolyásolhatja az algoritmus hatékonyságát. Gyakorlatban a berendezések „leterheltségének" sorrendjében való ütemezés általában jó stratégiának bizonyult. Ezen kívül a részfeladat kiválasztása is jelentős hatással van az algoritmus futási idejére. A számítógépes megvalósításnál érdemes mélységben először keresést alkalmazni, az egy szinten lévő részfeladatok közül a legjobb alsó korlátút választva.

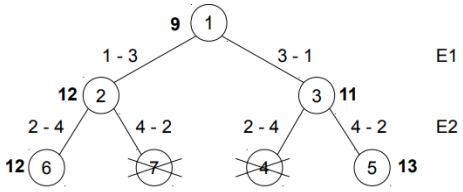
AZ ALGORITMUS KORLÁTOZÁSI LÉPÉSE

A korlátozás során először egy megvalósíthatósági vizsgálatot (feasibility test) végzünk el. Ha az aktuális részfeladat megvalósíthatónak (feasible) bizonyul, akkor alsó korlátot szabunk a részfeladatból elérhető megoldások működési idejére. Ha a köztes termékek várakozási ideje két taszk között nem korlátozott, akkor az S-gráf által reprezentált megoldás csak akkor megvalósítható, ha a gráfban nincs irányított kör, azaz az S-gáf egy ütemezési-gráf. Ezek alapján a megvalósíthatósági vizsgálat elvégezhető egy körkereső algoritmus segítségével, melyre léteznek hatékony polinomiális algoritmusok. Ha a köztes termékek várakozási ideje korlátozva van (ZW tárolási stratégia), akkor további vizsgálat szükséges, pl. egy lineáris programozási (LP) feladat felírása és megoldása. Egy részfeladat S-gráfja mindig részgráfja bármely leszármazott részfeladat S-gráfjának, mivel a gráfot a szétválasztás során legfeljebb éllel bővítjük, de soha nem törlünk belőle. Mivel egy új (nemnegatív súlyú) él hozzáadása egy gráfhoz nem csökkentheti a leghosszabb utat, ezért egy ütemezési-gráf bármely részgráfjához tartozó leghosszabb út alsó korlát az ütemezési-gráf leghosszabb útjára. Egy ütemezési-gráf bármely részgráfjához tartozó leghosszabb út alsó korlátot ad az ütemezési-gráfhoz tartozó megoldás értékére, mivel minden (i, j) él egy S-gráfban azt jelenti, hogy a j csomópont által reprezentált taszk elvégzése leghamarabb c(i, j) idővel később kezdődhet el az i csomópont által reprezentált taszk elkezdéséhez képest. Így egy

AZ ALGORITMUS MŰKÖDÉSÉNEK SZEMLÉLTETÉSE



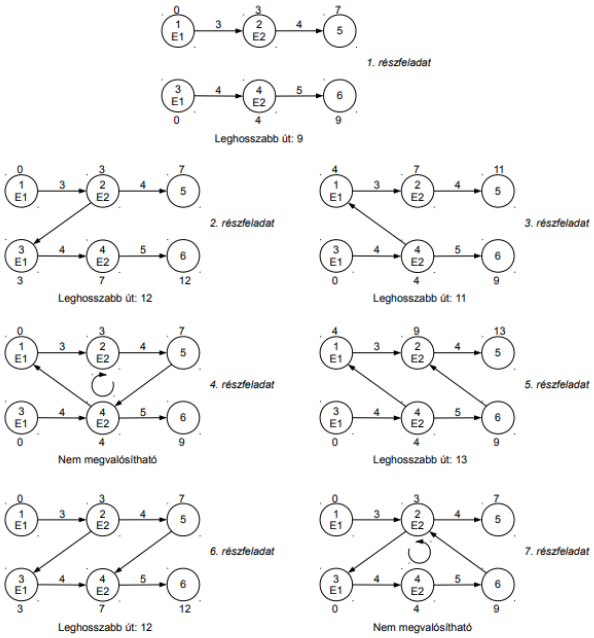
6.14. ábra. Recept-gráf a szemléltető példához.



6.15. ábra. A szemléltető példához tartozó keresőfa.

A Sanmartí és társai által publikált alapalgoritmus a leghosszabb út kereső algoritmust használja, ezért most mi is. A példában (6.14 ábra) két terméket kell előállítani két lépésben két berendezéssel.

A példához tartozó kereső fa a 6.15 ábrán látható. A csomópontok a részfeladatokat jelölik és a generálási sorrend szerint számoztuk őket. A szétválasztásokat (döntéseket) a taszkok sorrendjével jelöljük, amelyeket meghatároznak. A megoldást leegyszerűsítettük annyiban, hogy a két taszk-berendezés hozzárendelés és a sorrendjük meghatározása egy lépésben történik meg. Az alsó korlát minden részfeladat csomópontja mellett félkövéren szerepel. A nem megvalósítható részfeladatok csomópontjai át vannak húzva. A részfeladatokhoz tartozó S-gráfok a 6.16 ábrán láthatóak.



6.16. ábra. A szemléltető példa részfeladataihoz tartozó S-gráfok

S-GRÁF MÓDSZERTAN KITERJESZTÉSEI

Az általunk vizsgált két szempont közül az első esetben a szétválasztás és korlátozás algoritmus keresési tér bejárási módszerét módosítjuk, a másikban pedig a célfüggvényt változtatjuk meg, amely a keresési tér és a hozzá kapcsolódó megoldási módszer megváltoztatását is magával vonja.

TASZK ALAPÚ DÖNTÉSI STRATÉGIA

Az alapalgoritmusban „berendezés alapú döntési stratégiát” (EQ-SG algoritmus) használtunk. Ebben az esetben döntésnél a részfeladat egy ütemezendő berendezésének a kiválasztása után hozzáfűzünk a berendezés végrehajtási sorának végére egy még be nem ütemezett taszkot. A gyermek részfeladatok az összes lehetséges taszk hozzáfűzést figyelembe véve keletkeznek. Az ebben a fejezetben bemutatandó „taszk alapú döntési stratégiát” használó algoritmus (TA-SG algoritmus) ugyanazt a keresési teret járja be, mint a berendezés alapú és bizonyos feladatokra hatékonyabb ütemezőt biztosít ([1]). A taszk alapú döntéseket használó algoritmus szétválasztás lépésében a döntések új alapötlete az, hogy rendeljünk hozzá egy még be nem ütemezett taszkhoz egy lehetséges berendezést és szúrjuk be a taszkot a berendezés végrehajtási sorrendjébe figyelembe véve az összes lehetőséget. Az algoritmus, mint minden szétválasztás és korlátozás elvén működő algoritmus, egy kereső fát valamilyen módon bejárva határozza meg a feladat optimális megoldását. A kereső fa csúcspontjaihoz tartoznak a részfeladatok. Minden részfeladatnak tárolnia kell a kereső fa gyökeréből indulva a részfeladathoz vezető éleken meghozott döntéseket. A kereső fa gyökeréhez a recept-gráf tartozik, leveleihez pedig az ütemezési-gráfok ugyanúgy, mint a berendezés alapú ütemezésnél. A szétválasztási eljárás a döntési lépésében két műveletet hajt végre: kiválaszt egy olyan taszkot, amely még nincsen beütemezve, majd a kiválasztott taszkot beilleszti egy megfelelő berendezés aktuális ütemezésébe. A taszkot végrehajtani tudó összes lehetséges berendezést hozzá kell rendelni a taszkhoz, azaz befűzzük az összes rendelkezésre álló berendezés aktuális ütemezésébe az új taszkot, ahol minden hozzárendelés egy új részfeladatot jelent. Tegyük fel, hogy egy i taszkhoz egy berendezés áll rendelkezésre valamint ehhez a berendezéshez már hozzá van rendelve k darab taszk (j1 , j2 , . . . , j k) sorrendben. Ebben az esetben legfeljebb k + 1 darab gyerek részfeladat lesz, ahol a berendezés működési sorrendje rendre (i, j1 , j2 , . . . , j k), (j1 , i, j2 , . . . , j k), (j1 , j2 , i, . . . , j k), . . ., (j1 , j2 , .

. . , i, j k), (j1 , j2 , . . ., j k , i), pontosabban ezek közül azok, amelyek a megvalósíthatósági teszten megfelelnek. Abban az esetben, ha az ütemezésre kiválasztott taszkot két különböző berendezéssel lehet végrehajtani, mely berendezések jelenlegi ütemezése már k1, illetve k2 darab sorba fűzött taszkot tartalmaz, akkor az új taszk valamely berendezéshez való hozzárendelése és befűzése a berendezések taszk végrehajtási sorába (k1 + 1) + (k2 + 1) darab új gyermek részfeladatot eredményez, mivel egymástól függetlenül mindkét berendezéssel végre lehet hajtani a taszkot. Mindkét berendezés esetén az új ütemezendő taszkot be lehet fűzni valamelyik jelenleg is szereplő taszk elé, vagy pedig a taszk végrehajtási sor legvégére, azaz összesen láncolt lista elemszáma plusz egy új lehetőség áll fent. Az EQ-SG algoritmus esetén a szülő részfeladat S-gráfjának leghosszabb útja része a gyermek részfeladat S-gráfjának, hiszen a gyermek részfeladat S-gráfjában ugyanazok az élek szerepelnek, legfeljebb a recept-élek súlya növekedhet. Ezért teljesül, hogy a gyermek részfeladat alsó korlátja nem kisebb, mint a szülő részfeladaté. A TA-SG algoritmus során a gyermek részfeladatban a berendezések ütemezésének változásakor ütemezési-él törlődik a szülő részfeladat S-gráfjához képest, ami miatt előfordulhat, hogy a szülő részfeladat S-gráfjának leghosszabb útja nem szerepel a gyermek részfeladat S-gráfjában. A TA-SG algoritmus akkor oldja meg helyesen a feladatot, ha a következő feltétel teljesül. Ha az ütemezési feladat receptjének váltási idejeire teljesül a háromszög egyenlőtlenség, azaz bármely berendezés esetén teljesül, hogy tij ≤ tik + tkj, ahol tij, tik, tkj a berendezés váltási ideje az i, j, az i, k és a k, j taszkok között. Ilyenkor a szülő részfeladatból a taszk alapú döntésen alapuló szétválasztás eljárással előállított bármely gyermek részfeladatnak az alsó korlátjai nem kisebbek, mint a szülő részfeladat alsó korlátja. A feltétel a gyakorlatban azt jelenti, hogy két taszk közötti tisztítást nem kiváltani két összeségében rövidebb idejű tisztítással. Formálisan ennél élesebb korlátot is lehet adni, amelyre még mindig helyesen működne az algoritmus, de a gyakorlatban ez a feltétel mindig teljesül. A taszk alapú döntéseket alkalmazó szétválasztási eljárás nem befolyásolja az EQ-SG algoritmus korlátozás eljárását, ezért változtatás nélkül használhatóak az EQ-SG algoritmusban bevezetett körkereső és leghosszabb út kereső algoritmust használó korlátozás eljárás. A korlátozás lépésben a részfeladat megvalósíthatóságát az S-gráf körmentességének vizsgálatával döntjük el. Amenynyiben a részfeladat megvalósítható ütemezést jelöl, akkor a leghosszabb út kereső algoritmussal számolhatjuk ki a részfeladatból elérhető ütemezések végrehajtási idejének alsó korlátját.

PROFIT MAXIMALIZÁLÁSA

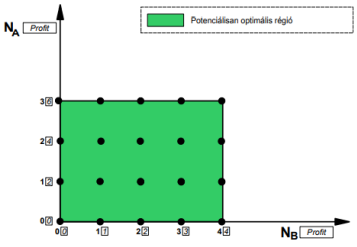
Célunk adott idő alatt a lehető legnagyobb profitot (hasznosságot) elérni és ehhez meghatározni, hogy melyik termékből mennyi adagot kell/lehet legyártani. Ebben az esetben az előállítandó termékmennyiségek nem adottak, hanem a termékek hasznossági mutatóját és egy időhorizontot kell bevezetni, ahol a hasznossági mutató megadja, hogy egy adagnyi termék mekkora hasznossággal (profittal) jár, az időhorizont pedig felső korlátot jelent a rendszer működési idejére.

A megoldást egy irányított keresési módszerrel határozzuk meg, aminek hatékonysága két dolgon alapul:

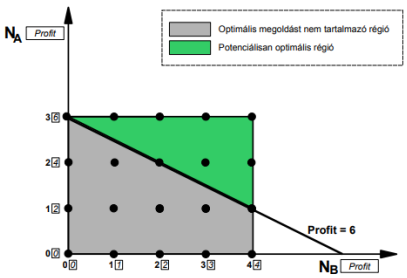
- A termékek adagjainak (batch-jeinek) száma rögzített a keresési tér minden pontjában és ez az érték minden pontban eltérő, így kizárjuk a redundanciát.
- A keresési térből egy előfeldolgozás során kizárjuk azon csomópontokat, melyek olyan adagszámokhoz tartoznak, amelyek nem lehetnek optimálisak.

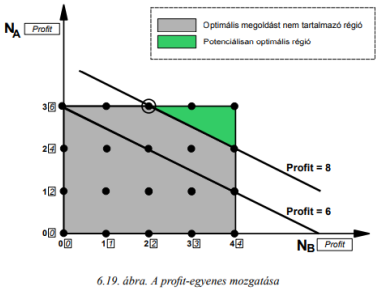
Mivel a keresési tér minden pontjában az adagok száma rögzített, ezért használhatjuk a korábban megismert S-gráf algoritmust. Ezekben a részfeladatokban elég, ha az algoritmus talál egy, az időhorizonton belül megvalósítható megoldást az adott adagokkal. Ha találunk ilyet, a többi részfeladatot már nem is kell megvizsgálni.

Példa: két termékünk van (A és B). Az A termék egy adagjának a profitja (hasznossága) legyen 2 egység (RA = 2), B terméknél ez legyen 1 (RB = 1). Továbbá tételezzük fel, hogy ha csak egy terméket gyártunk, akkor A termékből az időhorizonton belül 3 adagot tudunk előállítani, B-ből pedig 4-et. Ekkor a keresési teret a 6.17 ábrán látható módon ábrázolhatjuk, ahol a függőleges tengelyen az A termék adagjainak számát (NA), a vízszintesen a B termék adagjainak számát (NB) jelöljük. Az adagszámok után bekeretezve az aktuális profit értéke található.

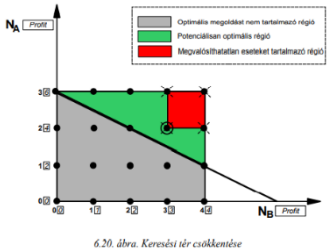


6.17. ábra. Keresési tér két termék esetén





6.19. ábra. A profit-egyenes mozgatása



6.20. ábra. Keresési tér csökkentése

A függőleges és a vízszintes tengelyek metszéspontja tartozna egy szokásos szétválasztás és korlátozás algoritmus keresőfájának gyökeréhez. Ez a metszéspont nulla adagot jelent minden termékből. A kereséséi tér vízszintes és a függőleges határát termékek maximális adagszáma adja meg, amit az időhorizont alatt elő lehet állítani belőlük. Természetesen a keresési tér csak a csúcsokat tartalmazza nem a teljes bejelölt területet. Mivel minden terméknél az egy adaghoz tartozó profit fix, ezért minden csomópontban a profitot az NARA +NBRB lineáris egyenlettel számolhatjuk. A 6.17 ábrán látható, hogy az A termékből 3 adag előállítása esetén 6 egységnyi profitot tudunk elérni, ezért, mivel ez egy megoldást jelöl, az ennél kisebb értékkel rendelkező csomópontokat nem kell figyelembe vennünk. A keresési térben a 6 egységi profitnál kisebb értékű csomópontok az NARA +NBRB = 6 egyenes (profit-egyenes) alatt találhatóak, ezeket kizárhatjuk a keresésből (lásd 6.18 ábra). Ha egy csomópont a vonalon fekszik, azt sem kell megvizsgálni, mivel az nem adhat jobb megoldást a már megtalált 4 adagnyi A-nál. A keresés során a profit-egyenes helye változhat, amikor találunk egy az aktuálisnál jobb megoldást. Az új profit-egyenes az eredetivel párhuzamos lesz, mivel ha például az új profit értéke 8, akkor az egyenes egyenlete NARA +NBRB = 8 lesz (6.19 ábra).

Ha a keresés során nem megvalósítható csomópontra (részfeladatra) találunk, akkor a keresési teret tovább csökkenthetjük. Tegyük fel, hogy az A-ból 2 adag és B-ből 3 adagot jelentő csomópont nem megvalósíthatónak bizonyult. Ez azt jelenti, hogy ekkora mennyiségű terméket nem lehet az időhorizont alatt előállítani. Ebből viszont az is következik, hogy többet sem lehet, azaz minden olyan csomópontot, amely legalább 2 adag A-t és 3 adag B-t tartalmaz, kizárható a keresési térből. Ez a keresési terünkben egy téglalap alakú területet jelent az aktuális csomópontunk felett jobbra (6.20 ábra).