Gazdasági informatika

Gráf algoritmusok

Minimális feszítőfa

- ***Prím algoritmus:*** Először egy minimális súlyú élt választunk a feszítőfába, utána pedig a már meglévő feszítőfát bővítjük úgy, hogy annak a csúcsaihoz kapcsolódó (minimális súlyú) éleket adunk a feszítőfához úgy, hogy a kritériumokat az ne sértse.
- ***Kruskal algoritmus:*** A lehetséges (feszítőfa feltételeit nem sértő, azaz kört nem alkotó) és minimális élek hozzáadásával történik a feszítőfa építése. (Nem feltétlenül lesz tehát része egy minimális feszítőfának az összes minimális súlyú él.)

Mindkét algoritmusra igaz tehát, hogy új éleket csak úgy választhatunk be, hogy azok különböző komponensbe tartozó csúcsokat kössenek össze.

Legrövidebb út

- ***Dijsktra:*** Kezdetben a kiindulású csúcshoz rendeljünk 0, a többihez pedig végtelen hosszúságú utat (egészen addig, ameddig nem vizsgáltunk olyan csúcsot, amelyből az adott csúcsba el lehet jutni). Az algoritmus során a csúcsokat egyesével megvizsgáljuk (a legkisebb aktuális költségű (súlyú) lesz mindig a soron következő), vesszük a vizsgált csúcsból kimenő élek súlyát, és azok alapján felülírjuk a többi (még nem vizsgált) csúcshoz tartozó költségeket. A már megvizsgált csúcsokat egy halmazba gyűjtjük, amelynek elemeit már nem kell vizsgálnunk, hiszen az ezen csúcsokhoz tartozó legrövidebb utakhoz már pontos becslésünk van.

Bejárások

- ***Szélességi keresés:*** Ehhez az algoritmushoz sort (FIFO) adatszerkezetet használunk. Tehát az újonnan beolvasott elemek a sor végére kerülnek, kivenni pedig a sor elejéről tudunk. Az első csúcspont (gyökér) beolvasása után minden iteráció során megvizsgáljuk, hogy van-e elem a sorban, ha nincs akkor az algoritmus megáll. Ellenkező esetben kivesszük az első elemet és ennek a szomszédjait (gyerekeit) egyenként a sor végére fűzzük. Minden szomszédról megjegyezzük, hogy ők már bekerültek a sorba (így esetleges kör esetén így ezek a csúcsok nem lesznek újra beillesztve a sorba).
- ***Mélységi keresés:*** Ehhez az algoritmushoz vermet (LIFO) adatszerkezetet használunk. Az újonnan beolvasott elemek a verem végére (tetejére) kerülnek, kivenni pedig a verem tetejéről tudunk. Első iterációban itt is a gyökér csúcsot vesszük ki, és annak a szomszédjait tesszük be a verembe. Következőnek kivesszük a verem utolsó elemét és ennek a szomszédjait szúrjuk a verem végére (fontos itt is a verembe elhelyezett elemek megjegyzése a körök kiküszöbölése végett). A bejárás mindig a gráf egy adott tartományára koncentrálódik, adott irányban mélyül, ameddig lehetséges. A verem kiürülésével fejeződik be az algoritmus.

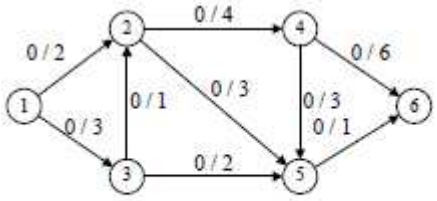
Maximális folyam (Ford-Fulkerson)

A gráf éleihez adott egy-egy kapacitás érték (egy időegység alatt mennyi egység anyagot lehet szállítani). Azoknak az éleknek nulla a kapacitás, amik nem részei a gráfnak. A termelő és a fogyasztó közötti csomópontokra teljesülnie kell az anyagmegmaradás törvényének (nincs termelés, tárolás, fogyás). Fontos, hogy az élen átmenő folyamérték ne haladja meg az élhez rendelt kapacitáskorlátot. Maximális folyamról beszélünk akkor, ha a termelőből a lehető legnagyobb anyagmennyiséget szállítjuk a fogyasztóba a hálózaton úgy, hogy teljesül az anyagmegmaradás, és a kapacitási korlátozásokat sem sértjük meg.

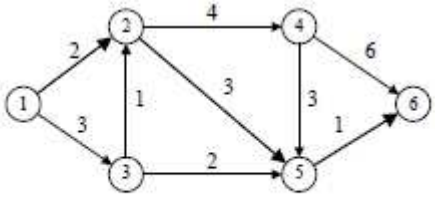
reziduális hálózat: A kiindulási gráf éleihez tartoznak folyam/kapacitás értékek. A reziduális kapacitást meghatározása: kapacitás - folyam érték. A reziduális gráfon oda-vissza élek szerepelnek (odafele él értéke: reziduális kapacitás, visszafele élé: folyam érték). Példa:

Ford-Fulkerson
Kezdetben minden folyam érték legyen 0. Ezután a reziduális hálózaton tetszőleges algoritmussal utat keresünk a termelőből a gyártóba. A kapott úton megvizsgáljuk, hogy mekkora a minimális reziduális kapacitás, és annyival megnöveljük az úthoz tartozó folyamok értékét. A keresett utat javító útnak nevezzük, a folyamat addig megy, ameddig van javító út.

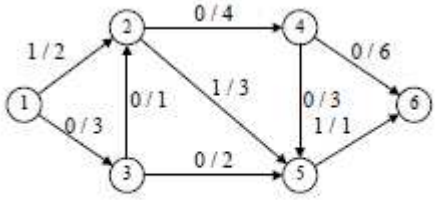
Lényegében először mindig a reziduális hálózaton keresünk úgy útvonalat, hogy az ne tartalmazzon telített élt (a reziduális hálózatban a telített él csak termelő felé mutat visszafele). Ezután növeljük a minimális reziduális kapacitással az úthoz tartozó folyam értékeket a sima hálózati ábrán, ami alapján újrarajzoljuk a reziduális gráfot, amiben a következő iteráció során újabb javító utat keresünk (ha van olyan). Példa:



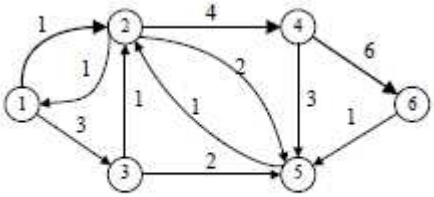
A hálózat az első iteráció előtt



Az induló hálózat reziduális hálózata

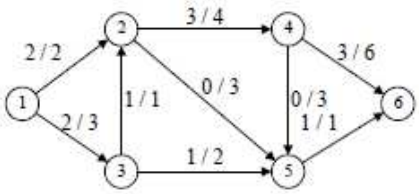


A hálózat az első iteráció után

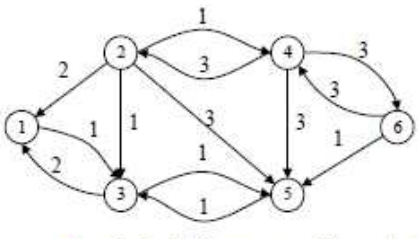


A reziduális hálózat az első iteráció után

Az utolsó (4. iteráció) eredménye pedig:



A hálózat a negyedik iteráció után



A reziduális hálózat a negyedik iteráció után

Lineáris programozási feladat modellezése

Pizzás példa: egy nap sajtból 550, sonkából 150, ananászból 120 adag áll rendelkezésre.
Két féle pizzát készíthetünk:

- Margaréta: 10 sajt, 2 sonka, 0 ananász
- Hawaii: 5 sajt, 4 sonka, 3

Egy margaréta pizzát (x_1) 600 forintért, egy hawaiit (x_2) 800 forintért tudunk értékesíteni.
A feladat az adatok ismeretében a pizzákból származó bevétel maximalizálása, ez alapján a célfüggvény: max $600x_1 + 800x_2$, a megszorítási egyenlőtlenségek pedig:

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

A modell GNU MathProg nyelven:

```
var x1 >= 0;
var x2 >= 0;
s.t. sajt: 10*x1 + 5*x2 <= 550;
s.t. sonka: 2*x1 + 4 * x2 <= 150;
s.t. ananasz: 3*x2 <= 120;
maximize bevetel: 600 * x1 + 800 * x2;
```

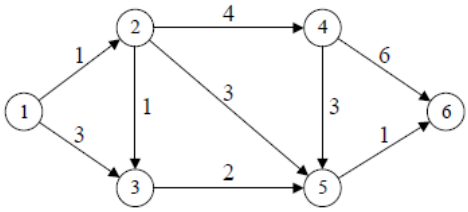
Nevezetes problémákhoz matematikai modell megadása

Legrövidebb út

Minden élhez rendeljünk logikai változót (ennek az értéke dönti el, hogy szerepel-e az adott él a legrövidebb útban). Továbbá adjunk meg pár feltételt: A csúcsokba és csúcsokból vezető élek összege egyenlő kell, hogy legyen (jelen esetben 1 lesz), valamint a kiindulási élek közül feltétlenül ki kell választanunk valamelyiket. Cél az út költségének minimalizálása, amelyhez a következő matematikai modellt adhatjuk meg:

```
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; //x12, x25, x56 értéke 1 lesz, a változóé pedig 0
```

A feladathoz tartozó gráf:

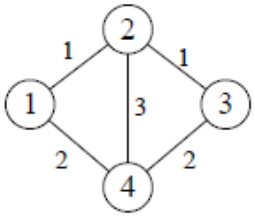


Minimális feszítőfa

Az irányítatlan {u, v} élből alkossunk egy (u, v) és egy (v, u) élt, ahol a két él súlya megegyezik az irányítatlan él súlyával. Minden élhez tartozzon bináris (y(u, v)) és folytonos (x(y, v)) változó. A bináris határozza meg, hogy az adott él beletartozik-e a feszítőfába, a folytonos pedig az éleken szállított anyagok mennyiségét mutatja (a maximálisan szállítható anyagmennyiség eggyel kisebb, mint a csomópontok (fogyasztók) száma). Minden fogyasztó egy egységnyi anyagot fogyaszt, így az anyagáramlás során összesen eggyel kevesebb anyag fog kiáramolni, mint ami beáramlott. Példa a matematikai modellre:

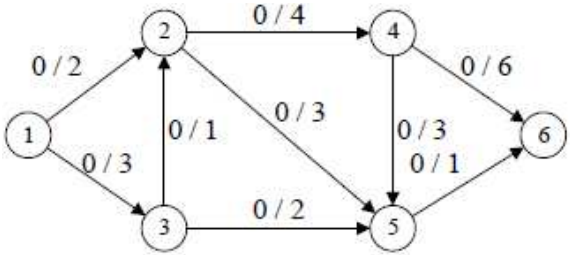
```
var xAB >= 0; var xBA >= 0;
var xBC >= 0; var xCB >= 0;
var xCD >= 0; var xDC >= 0;
var xAD >= 0; var xDA >= 0;
var xDB >= 0; var xBD >= 0;

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);
```

A feladathoz tartozó gráf:



Maximális folyam

A Ford-Fulkerson algoritmus tárgyalásakor vett feltételeket kell megadnunk a modellhez: Az élekhez tartozó folyamértékek nagysága nem haladhatja meg a kapacitásokat, valamint a kezdő és cél csomópontot kivéve teljesülnie kell az anyagmegmaradás feltételének. A következő gráfhoz tartozó matematikai modellt fogjuk meghatározni:



A matematikai modell a következő:

```
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;

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;

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;

maximize folyam: x12 + x13;
```

Folyamathálózathoz matematikai modell megadása

A példához adottak a következő adatok, korlátozások:

Műveleti egységek	Bemenetek	Kimenetek	Fixköltség	Arányos költség
O_1	C(5)	A(4),G(1)	4	1
O_2	D(9)	A(8),B(1)	3	1
O_3	E(4),G(1)	C(5)	2	1
O_4	F(10)	C(1),D(9)	2	0,5

Nyersanyag	Ár
E	0,8
F	1,6

Termék	Feltétel
A	≥ 4
Nyersanyag	**Feltétel**
E	≤ 10

A modellben a folytonos változókat *x*-el, a binárisokat pedig *y*-al jelöljük. Az *i*-edik műveleti egységhez (O_i) rendelt x_i folytonos változó jelöli a műveleti egység méretét (kapacitását), az y_i bináris változó pedig azt, hogy az adott műveleti egység szerepel-e a struktúrában vagy sem. Ha az O_i műveleti egység része a struktúrának, azaz $y_i = 1$, akkor a műveleti egység kapacitását reprezentáló x_i folytonos változó bármilyen értéket felvehet 0, és a műveleti egység felső kapacitáskorlátja, U_i között. U_i az O_i műveleti egység kapacitására vonatkozó felső korlát. Ha a feladatban nincs ilyen korlát definiálva, akkor U_i helyett egy tetszőleges, megfelelően nagy *M* szám használható.

A példa feladatban a célfüggvény a költségek minimalizálását fogja reprezentálni:
min $x_1 + 4y_1 + x_2 + 3y_2 + 4.2x_3 + 2y_3 + 16.5x_4 + 2y_4$

A matematikai modellhez tartozó tovább megkötések, feltételek:

Az anyagegyensúly feltételek a köztes anyagokra:
C anyag: $-5x_1 + 5x_3 + x4 \geq 0$ (5.35)
D anyag: $-9x_2 + 9x_4 \geq 0$ (5.36)
G anyag: $x_1 - x_3 \geq 0$ (5.37)

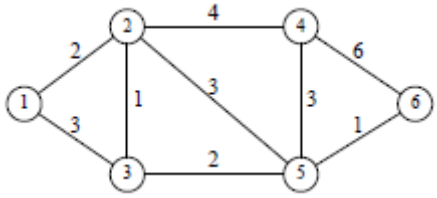
A termékre vonatkozó feltétel:
A termék: $4x_1 + 8x_2 \geq 4$ (5.38)

A nyersanyagokra vonatkozó megkötések:
E nyersanyag: $4x_3 \leq 10$ (5.39)

További, a műveleti egységekre vonatkozó feltételek:
$O1 : x_1 \leq y_1 M$ (5.40)
$O2 : x_2 \leq y_2 M$ (5.41)
$O3 : x_3 \leq y_3 M$ (5.42)
$O4 : x_4 \leq y_4 M$ (5.43)
Az *F* nyersanyagra vonatkozóan nem adtunk meg korlátozó feltételt.

Nevezetes feladatok megoldása folyamatszintézissel

Definiáljunk egy gráfot a feladatokhoz a következőképpen:
A csúcsok {1, 2, 3, 4, 5, 6} halmazát az M = {m1, m2, m3, m4, m5, m6} anyaghalmaz jelölje.
A gráfban irányítatlan élek vannak, így az élek mindkét lehetséges irányát rögzíteni kell:
{(1,2), (2,1), (1,3), (3,1), (2,3), (3,2), (2,4), (4,2), (2,5), (5,2), (3,5), (5,3), (4,5), (5,4), (4,6), (6,4), (5,6), (6,5)}
A műveleti egységek halmaza pedig:
O = {o12 = ({m1},{m2}), o21 = ({m2}, {m1}),..., o56 = ({m5},{m6}), o65 = ({m6}, {m5})}



Minimális feszítőfa szintézise
Minimális feszítőfa egy olyan összefüggő gráf, amely minden csúcsot tartalmaz, és az élek súlyának összege minimális. Jelöljük a kiindulási csúcsot nyersanyagnak, minden más csúcs pedig legyen termék. A termékekre és a nyersanyagra adjuk meg a szükséges pozitív mennyiségi korlátokat: A nyersanyag felső korlátja legyen a termékek száma, termékek alsó korlátja legyen 1 (a hozzájuk vezető műveletek legalább egyikének a mérete lehet nulla). Ezután az élek súlyát, mint a leíró műveleti egységek fix költségét adjuk meg. Ez alapján a következő táblázatokkal reprezentálhatjuk a feladatot:

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	m5	1
o65	m6	m5	1

A feszítőfát azok az élek adják, amelyekhez tartozó műveleti egységek szerepelnek a szintézis feladat optimális megoldásában.

Legrövidebb út szintézise

A legrövidebb út hasonlóképpen határozható meg, mint a feszítőfa, azzal a különbséggel, hogy a legrövidebb úthoz csak a kiindulási és a cél csúcsot összekötő gráfot kell meghatározni, amelyben az élek súlyának összege minimális. Ennek megfelelően, 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 költségeként definiáljuk. Ezek alapján a következő táblázatokat írhatjuk fel:

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	

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

A legrövidebb utat azok az élek adják, amelyekhez tartozó műveleti egységek szerepelnek a szintézis feladat optimális megoldásában.

Maximális folyam szintézise

A maximális folyamoknál a gráf éleihez rendelt tulajdonság nem a költség, hanem a kapacitás. Ezen kapacitás mellett kell a lehető legnagyobb mennyiséget egy forrásból a nyelőbe juttatni. Az éleket reprezentáló műveleti egységnek itt nem a költsége lesz, hanem maximális kapacitása, amely megegyezik a megfelelő él kapacitásával. A maximális folyamot az motiválja, hogy a nyelőn megjelenő anyagmennyiséget egy olyan termékkel írjuk le, aminek van egy pozitív ára, tehát a minél nagyobb termelése egyre nagyobb profitot hoz. Annak érdekében, hogy az éleken ne folyjon olyan anyagmennyiség, amelynek nincs szerepe a maximális folyamban, mert csak közbülső csúcshoz vezet, korlátozzuk nullára a közbülső anyagpontokon megmaradó megmaradandó anyagmennyiséget az anyag felső korlátjával. Ezeknek az információknak a birtokában a következő táblázatokat adhatjuk meg a feladathoz:

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

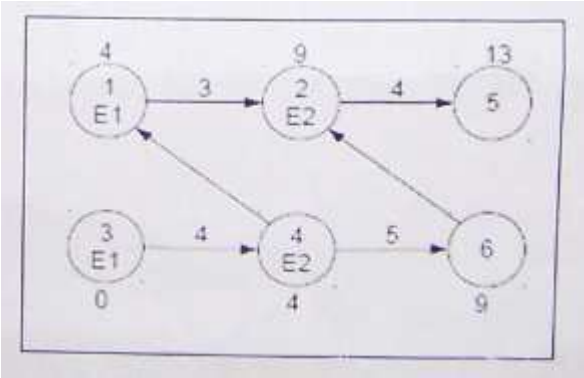
A maximális folyamot a szintézis feladat optimális megoldásában szereplő műveleti egységek által reprezentált élek adják, felhasznált kapacitásuk pedig megegyezik a megfelelő műveleti egységek optimális kapacitásával.

Ütemezés (Gantt-diagram)

A feladat során lényegében egy ütemezési gráfot (S-gráf) leíró Gantt-diagramot adunk meg. Adott az ütemezési gráf, vizsgáljuk meg a gráf csúcsait:

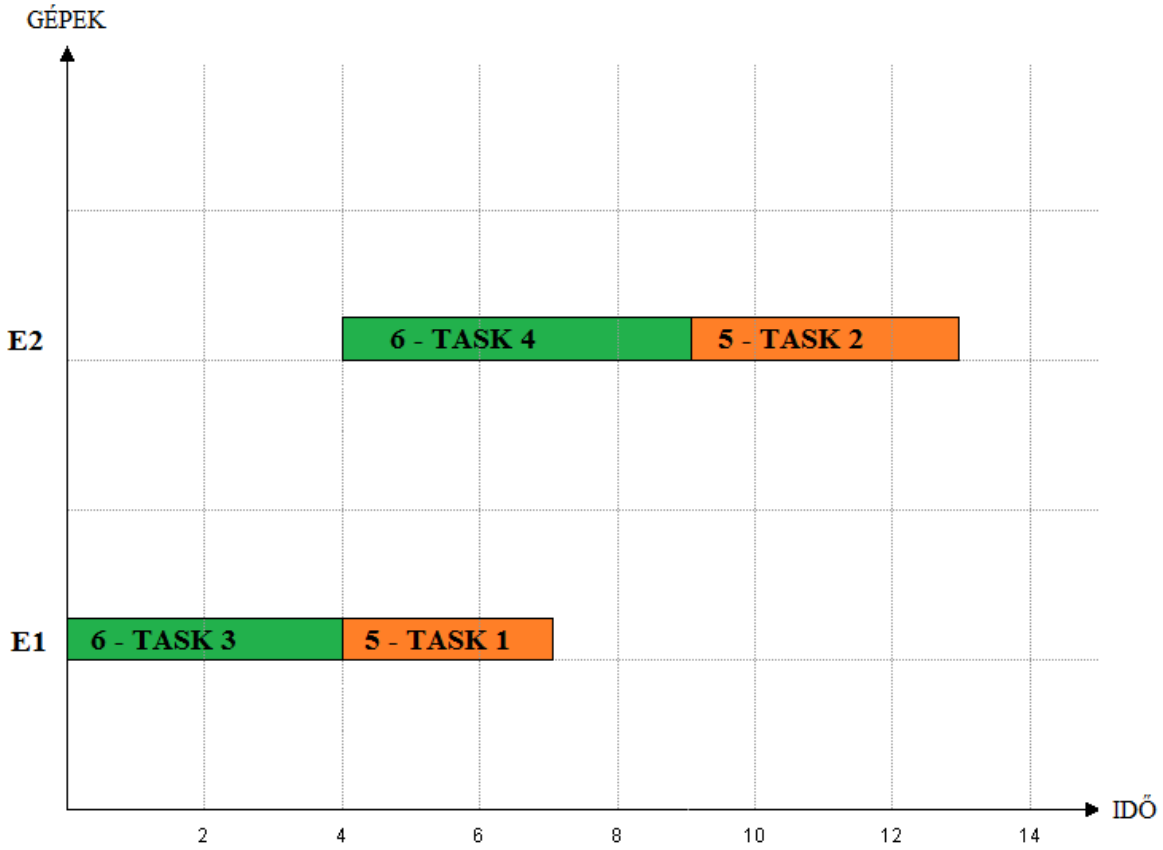
- E_i, $i \geq 1$, jelöli, hogy melyik gép hajtja végre az adott TASK-ot.
- Az E_i fölötti szám jelöli a TASK-ok sorszámát ($TASK_i$, $i \geq 1$).
- A csúcsok alatt, illetve felett található számok jelzik, hogy az adott csúcsot elérve egységnyi időben hol tart a végrehajtás.
- Az irányított éleken lévő számok a $TASK_i$ végrehajtásához szükséges időt jelzik.
- Látható, hogy a csúcsok sorokba vannak rendezve, a sorok végén található csúcs címéki definiáljak az egyes termékeket, a sorban lévő többi csúcs pedig az előállításukhoz szükséges TASK-ok sorozata.

Példa feladat, egy tetszőleges ütemezési gráfra:



A Gantt-diagram sikeres előállításához szögezzünk le pár dolgot: Egy gép egyszerre csak egy TASK-ot végezhet, egy adott TASK egyetlen konkrét termék előállításához kapcsolódik, egy termék előállításán egyidejűleg egyetlen gép dolgozhat, viszont különböző TASK-okon dolgozhat párhuzamosan több gép is. Továbbá a TASK-ok végrehajtási sorrendjének követnie kell az ütemezési gráf irányított éleiből előálló útvonalakat.

Ez alapján az ütemezési gráfhoz a következő Gantt-diagram adható meg:



Fogalmak

2 pontos definíciók

Komponens gráf: 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 az egyes berendezéseknek a megoldáshoz tartozó működési sorrendjét.

Irányított gráf: Másik neve digráf. G = (V, A) párral adható meg, ahol V a csúcsok halmaza, $A \subseteq V \times V$ pedig az élek halmaza. Az éleket rendezett párok reprezentálják.

Irányítatlan gráf: G = (V, E) párral adható meg. Legfeljebb kételemű halmaza a csúcsoknak.

Súlyozott gráf: Ha a gráf csúcsait összekötő élekhez súlyokat rendelünk. Ábrázoláskor a súlyokat az élekre írjuk.

Descartes-szorzat: Az A és B halmazok Descartes-szorzata az összes olyan (a,b) rendezett pár halmaza, ahol $a \in A$ és $b \in B$. Jelölése: $A \times B$. Például:
{1,3,9} x {b,c,d} = {(1,b), (1,c), (1,d), (3,b), (3,c), (3,d), (9,b), (9,c), (9,d)}

Gantt diagram: A gépek a függőleges tengelyen, az idő a vízszintes tengelyen van ábrázolva.

Reziduális hálózat: Olyan gráf, amely tartalmazza az eredeti összes csúcspontját, valamint olyan irányított éleket, amelyekre a reziduális kapacitás nagyobb, mint 0. Tartalmazhat olyan éleket is, amelyeket az eredeti gráf nem. 5/20 súly az élen (5 egység szállítása, 20 kapacitás) → 5 kerül az él negáltjára, 20-5=15 pedig az eredeti irányú élre

Páros gráf (él csak két halmaz között van): Egy gráf akkor páros, ha a csúcshalmaza partícionálható két diszjunkt halmazra úgy, hogy az azonos halmazban lévő csúcsok közül egymással semelyik kettő sem szomszédos.

Lineáris programozás: A lineáris korlátozásokkal és célfüggvénnyel adott optimalizálási feladat felírását és megoldását nevezzük LP-nek.

IP: Az olyan feladatokat, amelyek csak egész változókat tartalmaznak, egészértékű lineáris problémáknak (integer programing, IP) nevezzük.

MILP (Mixed Integer Linear Programming): Vegyes egészértékű lineáris probléma, nem minden változóra van egész értékű megkötés. A legrövidebb út, a minimális feszítőfa és a maximális folyam is megoldhatóak vele.

FIFO (first in, first out): Minden új elemet a végére fűzünk és csak a sor elején lévő elemet vehetjük ki belőle.

LIFO (last in, first out): Olyan elv, mely szerint ami utolsónak megy be, elsőnek jön ki a veremből. A mélységi bejárásnál alkalmazzák.

Verem: Olyan adatszerkezet, melynek végére szúrjuk az új elemeket és a végéről szedjük ki a következő elemet is (LIFO elv – „last in, first out")

Legrövidebb út: Ki kell használni, hogy az út egy olyan részgráfja a fának, amelynek az első és utolsó csomópontját kivéve minden csomópont foka kettő, az elsőé és utolsóé pedig egy. Egyszerűbben: mely éleket kell kiválasztani ahhoz, hogy a kiválasztott élek egy minimális hosszúságú utat határozzanak meg.

2 pontos, tárolási osztályokhoz, ütemezéshez kapcsolódó definíciók (kifejtős is lehet)

Flow shop: Egy flow shop feladatban minden munka minden gépen végigmegy pontosan ugyanolyan sorendben. Tipikus példa flow shop feladatra a futószalag, ahol a munkások jelentik a gépeket.

Job shop: A feladatban nincsenek megkötések a műveletek számára és a hozzájuk rendelt gépekre. A gépek sorrendje minden munkára különböző lehet, valamint a felhasznált gépek is munkánként különbözhetnek.

Open shop: Ütemezési feladat, amely hasonló a job shop feladathoz, az eltérés abban van, hogy amíg a job shop feladatoknál a munkák műveleteinek a sorrendje rögzített, addig az open shop feladatokban a munkák műveleteit bármilyen sorrendben végre lehet hajtani a megfelelő gépeken. Azaz a munkákhoz tartozó műveletek sorrendje nincs megkötve, de mindet el kell végezni.

Termék: A gyártó rendszerben elő állítani kívánt anyag (product).

Berendezés: A termék előállításához felhasználható eszköz, erőforrás. A berendezés megújuló erőforrás, azaz használat után újra rendelkezésre áll.

Task: Egy olyan tevékenység, amely nem bontható résztevékenységekre és adott idő alatt adott bemenetekből adott kimeneteket generál. Egy task végrehajtásához egy vagy több berendezés áll rendelkezésre, ahol a végrehajtási idő függ a felhasznált berendezéstől.

Recept: Tartalmazza azokat a minimális információkat, amelyek a termékek előállításához szükségesek. Ezek a termékek előállításához szükséges task-ok hálózata, a task-okhoz rendelhető berendezések (működési idővel együtt) és az előállítandó termékmennyiségek. A termékek előállítása párhuzamosan is történhet.

Nincs lehetőség köztes tárolásra:

- *NIS tárolási stratégia:* A köztes termékek tárolásához nem áll rendelkezésre tároló berendezés. A task elvégzése után a köztes termékeket az azt előállító berendezésben tárolhatjuk, ilyenkor a berendezést csak akkor lehet használni egy másik task végrehajtásához, ha a keletkezett anyagot már áttöltöttük egy másik, értelemszerűen a következő task-ot végrehajtó berendezésbe.
- *ZW tárolási stratégia:* A köztes termék tárolásához nem áll rendelkezésre tároló berendezés valamint 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:

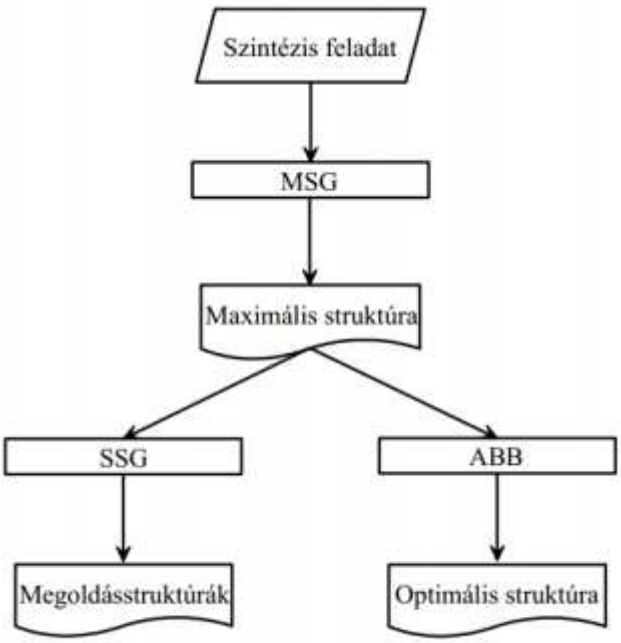
- *UIS tárolási stratégia:* A köztes termékeket "végtelen" mennyiségben lehet tárolni. A task elvégzése után a berendezésből a köztes terméket tároljuk egy, gyakorlati szempontból "végtelen" kapacitású tárolóban, azaz a berendezést tisztítás után rögtön tudjuk használni. Pl.: raktár.
- *FIS tárolási stratégia:* A rendszer véges számú és méretű tárolót tartalmaz. Minden tároló csak két előre meghatározott berendezés között használható. Pl.: folyékony anyagokat gyártó rendszer kiépített csőhálózattal. FIS-nek nevezik azt is, ha a tároló egység nem berendezéshez, hanem az anyag típusához rendelhető.

CIS tárolási stratégia: 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.

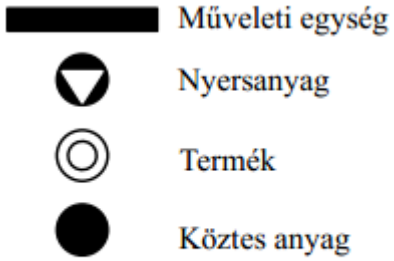
MIS tárolási stratégia: Az előző négy tárolási stratégia keveréke, a rendszer különböző pontjain különböző stratégiák lehetnek.

5 pontos, kifejtős elméleti kérdések

P-gráf (Process-graph, P-graph):



5.3. ábra. A P-gráf módszertan főbb lépései



5.5. ábra. Szimbólumok

1. Az anyagokat a gráf anyag típusú, a műveleti egységeket a műveleti egység típusú csúcsai reprezentálják.
2. Egy anyag és egy műveleti egység típusú csúcs között akkor és csak akkor megy él, ha a megfelelő anyag és a műveleti egység kapcsolata része a reprezentálandó folyamatnak.
3. Az élek irányai megegyeznek a folyamat előrehaladásának irányával.

Axiómák:

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

1. Minden legyártandó termék, azaz P minden eleme szerepel a struktúrában.
2. Egy a struktúrában szereplő anyag akkor és csak akkor nyersanyag, ha egyetlen a struktúrában szereplő műveleti egység sem állítja elő.
3. Minden a struktúrában szereplő műveleti egység a szintézis feladatban definiált.
4. Minden a struktúrában szereplő műveleti egységből vezet út legalább egy legyártandó termékhez.
5. 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 megoldás struktúrájának mondjuk.

Ütemezés:

Igaznak kell lenniük:

- Egy taszk 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.
- A taszk nem megszakítható.

Maximális struktúra: Más néven szuperstruktúra. Feltéve ha a megoldásstruktúrák halmaza véges és zárt az unióra, valamint nem üres, a halmaznak lesz egy eleme, amely az összes megoldásstruktúra uniója, ez a maximális struktúra. A maximális struktúra minden csúcsa és éle része legalább egy megoldás-struktúrának, illetve minden megoldás-struktúra P-gráfja részgráfja a maximális struktúrát reprezentáló gráfnak. A maximális struktúra a feladat összes lehetséges megoldását tartalmazza, így az optimálisat is. Önmaga is egy megoldás struktúra. Tulajdonképpen a maximális struktúra meghatározásával lecsökkentjük a keresési teret, mivel az optimális megoldás meghatározása során elegendő csupán a kombinatorikusan lehetséges megoldásstruktúrákat megvizsgálnunk.

SSG (Solution Structure Generation): A megadott szintézisfeladat összes kombinatorikusan lehetséges megoldás struktúráját, közte a maximális struktúrát, amely maga is megoldásstruktúra, pontosan egyszer generálja. Az algoritmusnak többféle számítógépes megvalósítása is létezik.
Egyik megvalósítás a döntési leképezéseken alapszik. A **döntési leképezés**ek során arról döntünk, hogy mely anyagot mely műveleti egységgel vagy egységekkel gyártunk, azaz mely műveleti egységeket vonjuk be egy adottmegoldás-struktúrába. Egyben arról is döntünk, hogy mely műveleti egységeket zárjuk ki az adott struktúrából. Egy műveleti egységnek, ha szerepel a struktúrában, minden kimeneti anyagát előkell állítania. Az SSG algoritmus döntési leképezésen alapuló megvalósítása rekurzívan hívja magát.

ABB (Accelerated Branch-and-Bound): Az optimális megoldás meghatározására egy speciális korlátozás és szétválasztás típusú algoritmus. Többféle implementációja létezik. Az egyik megvalósítás döntési leképezéseken alapszik. Ennek a megközelítésnek az az előnye, hogy az ABB csak a kombinatorikusan lehetséges struktúrákat vizsgálja az optimális megoldás keresése során.
Vegyes-egész programozási feladat, amelynek megoldására is használhatóak az általános Branch-and-Bound típusú módszerek. Bár a feladat optimális megoldása ezekkel a módszerekkel is meghatározható, hatékonyságuk még tovább javítható, mivel a megoldás keresése során nem veszik figyelembe a szintézis feladatok speciális tulajdonságait.

MSG (Maximal Structure Generation): A szintézis feladathoz tartozó maximális struktúra meghatározására jó. A már ismertetett axiómákon alapszik, és számítógépes implementációja az alábbi négy főbb lépésből áll:

- Az **első lépés**ben definiáljuk magát a (P, R, O) szintézis-feladatot az M anyaghalmaz, a P termékhalmaz, az R nyersanyag halmaz 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.
- A **második lépés**ben meghatározunk egy kezdeti struktúrát, vagy kiindulási hálózatot úgy, hogy a műveleti egységeket közös kimeneteik és bemeneteik szerint összekapcsoljuk.
- A **harmadik lépés**ben 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 szakasz*a. 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.
- A **végső fázis** az algoritmus *építő szakasza*. Ebben a fázisban 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.

Alapalgoritmus ütemezésre S-gráffal: Az S-gráf leírás jó alap egy szétválasztás és korlátozás (branch-and-bound, B&B) típusú algoritmushoz.
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.
A **korlátozás** során először egy megvalósíthatósági vizsgálatot végzünk el. Ha az aktuális részfeladat megvalósíthatónak bizonyul, akkor meghatározunk egy alsó korlátot a részfeladatból elérhető megoldások működési idejére. Ha a köztes termékek várakozási ideje két taszk között nem korlátozott, akkor az S-gráf által reprezentált megoldás akkor és csak akkor megvalósítható, ha a gráf nem tartalmaz irányított kört, azaz az S-gáf egy ütemezési-gráf. Ezek alapján a megvalósíthatósági vizsgálat elvégezhető egy körkereső algoritmus segítségével, melyre léteznek hatékony polinomiális algoritmusok.

S-gráf: 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.

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. Egy S-gráfot a 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.