

# *Gyártórendszerek dinamikája*

## ***Gyártásütemezés: megoldás a kritikus út módszerrel II., dinamikus programozás***

Werner Ágnes

*Villamosmérnöki és Információs Rendszerek Tanszék*

e-mail: [wenera@almos.uni-pannon.hu](mailto:wenera@almos.uni-pannon.hu)

*Erőforrás-korlátozás nélküli statikus ütemezés*

*Kritikus út módszer*

# Kritikus út módszer - CPM

---

CPM: Critical Path Method

## **Adott**

- az ütemezendő elemi műveletek halmaza (activity)
- minden elemi művelet előfeltételei (szintén elemi műveletek)  
*parciális rendezés (egymásutániség)*
- minden elemi művelet *végrehajtási ideje*

Táblázatba rendezhető

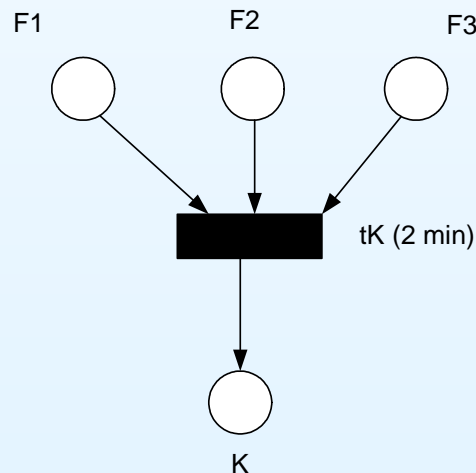
## **Grafikus leírás: CPM gráf**

- *élei megfelelnek az elemi műveleteknek (egy-egy értelmű megfeleltetés)*
- csúcsok megfelelnek eseményeknek (rendszer-állapotok bekövetkezésének)

## A CPM gráf és a Petri háló kapcsolata

**A CPM gráfbeli művelet** megfelel egy olyan átmenetnek egy **időzített Petri hálóban**, amelynek

- bemeneti eseményei (előfeltételei) a CPM gráfbeli művelet előfeltételei
- tüzelési ideje a művelet végrehajtási ideje
- következménye csak egy feltételből áll



**A CPM gráf az időzített Petri háló előfordulási gráfja**

# A kritikus út kiszámítása – 1

---

Algoritmikusan, két menetben

- I. Előrefelé haladó számítás: legkorábbi bekövetkezési idők
  1. A kezdő esemény kezdeti idejét nullára állítjuk
  2. Minden műveletet akkor kezdünk, ha az előfeltételek teljesültek
  3. Minden esemény legkorábbi bekövetkezési ideje ( $E_i$ ) a beléje vezető műveletek befejeződési idejeinek ***maximuma***
  
- II. Visszafelé haladó számítás: legkésőbbi bekövetkezési idők
  1. A befejező esemény legkésőbbi bekövetkezési idejét a legkorábbira állítjuk
  2. Minden művelet legkésőbbi bekövetkezési idejét a vég-esemény legkésőbbi bekövetkezési idejéből számítjuk, levonva belőle a művelet végrehajtási idejét
  3. Minden esemény legkésőbbi bekövetkezési ideje ( $L_i$ ) a belőle induló műveletek kezdő idejeinek ***minimuma***

## A kritikus út kiszámítása – 2

---

### Csúszási idők

1. Eseményekre: a legkésőbbi és a legkorábbi bekövetkezési idő különbsége,  $S_i = L_i - E_i$
2. Az  $i$ -edik eseményből a  $j$ -edikbe vezető műveletre:

$$S_{ij} = L_j - E_i - t_{ij}$$

ahol  $t_{ij}$  a művelet végrehajtási ideje

**Kritikus út: azon műveletekből (élekből) áll, amelyek csúszási ideje nulla**

# ***Dinamikus programozás - Gyártásütemezés***

# Dinamikus programozás

---

Egy dinamikus programozási algoritmus kifejlesztése 4 lépésre bontható fel:

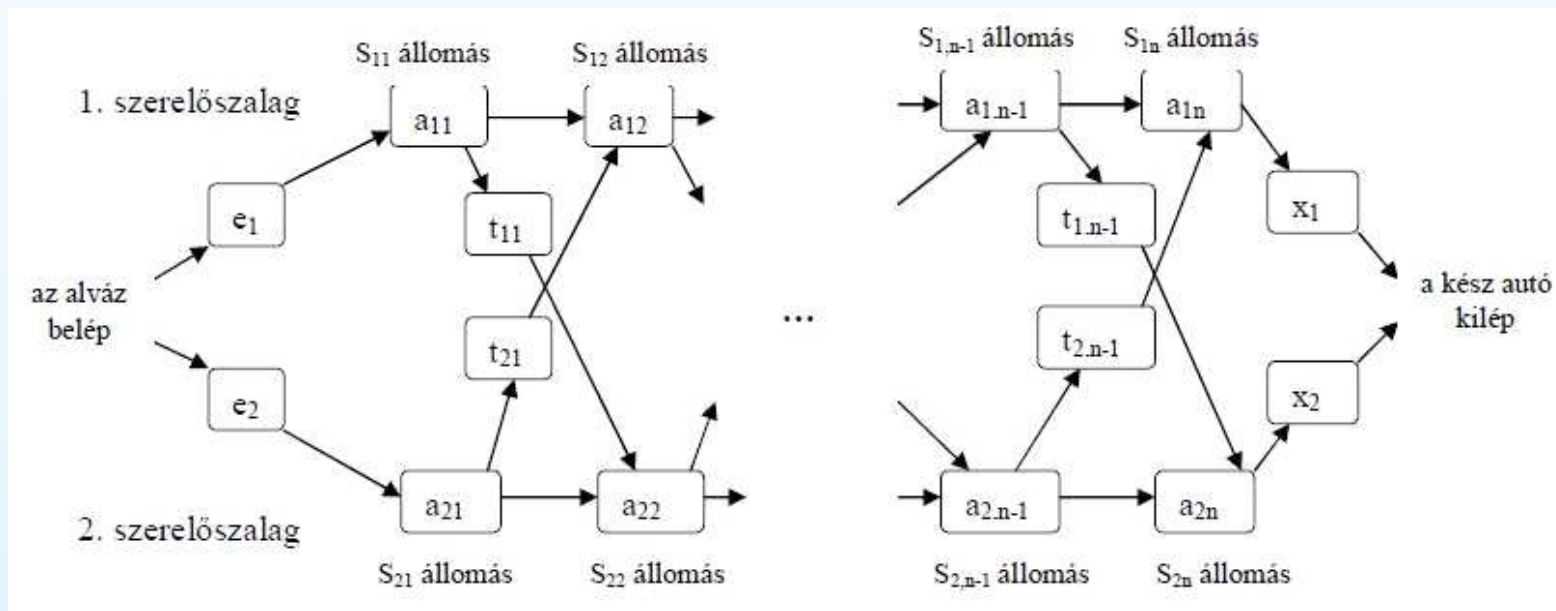
1. *Jellemezzük az optimális megoldás szerkezetét.*
2. *Rekurzív módon definiáljuk az optimális megoldás értékét.*
3. *Kiszámítjuk az optimális megoldás értékét alulról felfelé történő módon.*
4. *A kiszámított információk alapján megszerkesztünk egy optimális megoldást.*

A dinamikus programozás minden egyes részfeladatot és annak **minden részfeladatát pontosan egyszer oldja meg, az eredményt egy táblázatban tárolja**, és ezáltal elkerüli az ismételt számítást, ha a részfeladat megint felmerül.

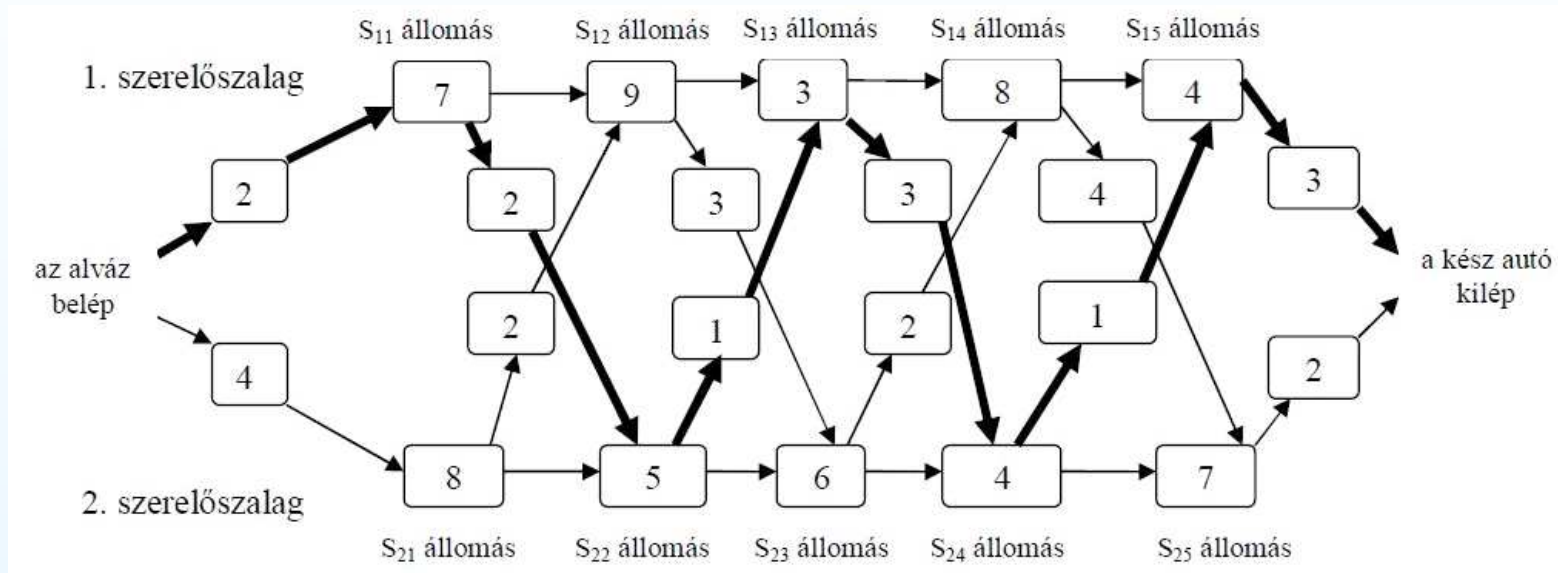


# Szerelőszalag ütemezése

Mindegyik szalagnak  $n$  állomása van, melyek indexei  $j = 1, \dots, n$ .  $S_{ij}$  jelöli az  $i$ -edik ( $i = 1$  vagy  $2$ ) szalag  $j$ -edik állomását. Az első szalag  $j$ -edik állomása ( $S_{1j}$ ) ugyanazt a funkciót látja el, mint a második szalag  $j$ -edik állomása ( $S_{2j}$ ). Az  $S_{ij}$  állomáson a műveleti időt  $a_{ij}$  jelöli.



# Szerelőszalag ütemezése -Példa



A teljes leszámolás számítási ideje  $\Omega(2n)$ . Ez nagy  $n$  mellett elfogadhatatlan.

## 1. lépés: a legrövidebb gyártási út szerkezete

Tekintsük a lehetséges legrövidebb utat, ahogy egy alváz a kiindulási helyzetből túljut az  $S_{1j}$  állomáson.

Ha  $j = 1$ , akkor csak egy lehetőség van, és így könnyű meghatározni a szükséges időt.

$j = 2, \dots, n$  esetén két lehetőség van:

- Az alváz érkezhét közvetlenül az  $S_{1,j-1}$  állomásról, amikor ugyanannak a szalagnak a  $(j - 1)$ -edik állomásáról a  $j$ -edik állomására való átszállítás ideje elhanyagolható.
- Az alváz  $S_{2,j-1}$  felől érkezik, az átszállítás ideje  $t_{2,j-1}$ .

Tegyük fel, hogy az  $S_{1j}$  állomáson való túljutás leggyorsabb módja az, hogy oda az  $S_{1,j-1}$  felől érkezünk.

Az alváznak a legrövidebb módon kell túljutnia az  $S_{1,j-1}$  állomáson, mert ha volna egy gyorsabb mód, hogy az alváz túljusson a  $S_{1,j-1}$  állomáson, akkor ezt használva magán  $S_{1,j}$ -n is hamarabb jutna túl, ami **ellentmondás**.

## 1. lépés: a legrövidebb gyártási út szerkezete

---

Általánosabban fogalmazva azt mondhatjuk, hogy a szerelőszalag ütemezésének  
*(hogy megtaláljuk a legrövidebb módot az  $S_{ij}$  állomáson való túljutásra)*

optimális megoldása tartalmazza egy részfeladat  
*(vagy az  $S_{1,j-1}$  vagy az  $S_{2,j-1}$  állomáson való leggyorsabb áthaladás)*

optimális megoldását.

Erre a tulajdonságra **optimális részstruktúraként** fogunk hivatkozni.

**Azaz részfeladatok optimális megoldásaiból megszerkeszthető a feladat optimális megoldása.**

## 2. lépés: a rekurzív megoldás

Az optimális megoldás értékét a részfeladatok optimális értékeiből rekurzívan fejezzük ki.

A részfeladatok legyenek mindkét szalag  $j$ -edik állomásán való leggyorsabb áthaladás megkeresésének a feladatai  $j = 1, \dots, n$  mellett.

Jelölje  $f_i[j]$  azt a legrövidebb időt, ami alatt az alváz túl tud jutni az  $S_{ij}$  állomáson.

**Célunk annak a legrövidebb időnek a meghatározása, ami alatt az alváz az egész üzemen keresztül tud haladni.**

Ezt az időt  $f^*$  jelöli.

$$f^* = \min\{f_1[n] + x_1, f_2[n] + x_2\}$$

$$f_1[1] = e_1 + a_{11}$$

$$f_2[1] = e_2 + a_{21}$$

## 2. lépés: a rekurzív megoldás

Hogyan kell kiszámolni az  $f_i[j]$ -t  $j = 2, \dots, n$  és  $i = 1, 2$  esetén?

$$f_1[j] = \min\{f_1[j-1] + a_{1j}, f_2[j-1] + t_{2,j-1} + a_{1j}\}, \text{ ha } j = 2, \dots, n.$$

$$f_2[j] = \min\{f_2[j-1] + a_{2j}, f_1[j-1] + t_{1,j-1} + a_{2j}\}, \text{ ha } j = 2, \dots, n.$$

A következő rekurzív egyenletet kapjuk:

$$f_1[j] = \begin{cases} e_1 + a_{11}, & \text{ha } j = 1, \\ \min\{f_1[j-1] + a_{1j}, f_2[j-1] + t_{2,j-1} + a_{1j}\}, & \text{ha } j \geq 2 \end{cases}$$
$$f_2[j] = \begin{cases} e_2 + a_{21}, & \text{ha } j = 1, \\ \min\{f_2[j-1] + a_{2j}, f_1[j-1] + t_{1,j-1} + a_{2j}\}, & \text{ha } j \geq 2 \end{cases}$$

## 2. lépés: a rekurzív megoldás

Az  $f_i[j]$  mennyiségek a részfeladatok optimális értékei.

Annak érdekében, hogy vissza tudjuk keresni a legrövidebb utat, definiáljuk  $l_i[j]$ -t mint azt a szerelőszalagot, amelyiknek a  $j - 1$ -edik állomását használtuk az  $S_{ij}$ -n való leggyorsabb keresztülhaladáskor.

Itt  $i = 1, 2$  és  $j = 2, \dots, n$ .

(Nem definiáljuk  $l_i[1]$ -et, mert  $S_{i1}$ -t egyik szalagon sem előzi meg másik állomás.)

Az  $l^*$  szalag definíció szerint az, amelyiknek az utolsó állomását használjuk az egész üzemen való áthaladáskor.

Az  $l_i[j]$  értékek segítségével lehet nyomon követni a legrövidebb utat.

### 3. lépés: a legrövidebb átfutási idő kiszámítása

---

Sokkal jobban járunk, ha az  $f_i[j]$  értékeket más sorrend szerint számoljuk, mint az a rekurzív módszerből adódik.

Vegyük észre, hogy  $j \dots 2$  esetén  $f_i[j]$  csak  $f_1[j - 1]$  -től és  $f_2[j - 1]$  -től függ.

Ha az  $f_i[j]$  értékeket az állomások  $j$  indexének növekvő sorrendjében számoljuk, akkor a legrövidebb átfutási idő meghatározása  $\Theta(n)$  ideig tart.



### 3. lépés: a legrövidebb átfutási idő kiszámítása

#### ALGORITMUS1(a,t,e,x,n)

1.  $f_1[1] := e_1 + a_{11}$      $f_2[1] := e_2 + a_{21}$
2.    for  $j := 2$  to  $n$
3.        do if  $f_1[j - 1] + a_{1j} \leq f_2[j - 1] + t_{2,j-1} + a_{1j}$
4.            then  $f_1[j] := f_1[j - 1] + a_{1j}$      $l_1[j] := 1$
5.            else  $f_1[j] := f_2[j - 1] + t_{2,j-1} + a_{1j}$      $l_1[j] := 2$
6.        if  $f_2[j - 1] + a_{2j} \leq f_1[j - 1] + t_{1,j-1} + a_{2j}$
7.            then  $f_2[j] := f_2[j - 1] + a_{2j}$      $l_2[j] := 2$
8.            else  $f_2[j] := f_1[j - 1] + t_{1,j-1} + a_{2j}$      $l_2[j] := 1$
9.        if  $f_1[n] + x_1 \leq f_2[n] + x_2$
10.            then  $f^* = f_1[n] + x_1$      $l^* = 1$
11.            else  $f^* = f_2[n] + x_2$      $l^* = 2$

## 4. lépés: a legrövidebb átfutási idejű út

---

$f_i[j]$ ,  $f^*$ ,  $l_i[j]$  és  $l^*$  kiszámítását követően meg kell szerkeszteni az üzemen való legrövidebb áthaladást biztosító utat.

Az eljárás az út állomásait az indexek csökkenő sorrendjében nyomtatja ki.

ALGORITMUS2(l,n)

1.  $i := l^*$
2. print  $i$ "-edik szalag"  $n$ "-edik állomás"
3. for  $j := n$  downto 2
4. do  $i := l_i[j]$
5. print  $i$ "-edik szalag (" $j - 1$ ")-edik állomás"