Processzorütemezés

Iványi Antal

2010. március 23.

Tartalomjegyzék

1. A determinisztikus ütemezés alapfogalmai

1.1. Bevezetés

A termelés különböző területein gyakran jelentkezik a következő feladat: adott munkákat kell elvégezni adott eszközök felhasználásával, adott korlátozó feltételek betartása mellett, adott értelemben optimumra törekedve. Az ilyen típusú feladatok jelentősége a nagy értékű termelőeszközök, bonyolult termelési folyamatok terjedésével együtt nő. Ezen feladatkör kutatása nagy lendületet kapott a számítógépek elterjedésével, mivel a gyors műveletvégzési lehetőség kiszélesítette a reális idő alatt megoldható feladatok körét.

A témakör fontosságát mutatja, hogy az összegyűlt ismeretek egy része *ütemezés elmélet* (scheduling theory) néven a számítástudomány önálló ágává fejlődött. Az operációs rendszerekkel kapcsolatos, tipikus ütemezési feladat: kötegelt programfeldolgozás esetén az adott perifériákat adott ideig használó, egymástól független programokból álló köteg minimális idő alatt való futtatása adott konfigurációjú számítógépen. Másik tipikus feladat egymáshoz adott módon kapcsolódó alprogramokból álló program futtatása egyprocesszoros számítógépen vagy több processzort tartalmazó hálózatban.

Ebben a jegyzetben az elvégzendő munkákat (például lefuttatandó programokat) *taszkoknak* vagy *feladatoknak*[1], a rendelkezésre álló eszközöket (például processzorokat, perifériákat) *erőforrásoknak* fogjuk nevezni.

Ha a feladatok és erőforrások jellemző adatai rögzített (rendszerint előre ismert) értékek, akkor *determinisztikus ütemezésről* beszélünk. A nemdeterminisztikus ütemezésben gyakori, hogy a nem rögzített paraméterek eloszlásfüggvényeikkel adottak. Ilyenkor a feladatok végrehajtását jellemző mennyiség várható értékének optimalizálása a tipikus cél.

Ebben a jegyzetben csak a determinisztikus ütemezéssel foglalkozunk. Először összefoglaljuk az alapfogalmakat, alapvető ismereteket (1. fejezet), majd a legegy-

[1] Az angol nyelvű szakirodalom a *task* szót használja.

szerűbb egyprocesszoros eredményeket ismertetjük (2. fejezet). Ezután a maximális befejezési idő és az átlagos befejezési idő optimalizálásával foglalkozunk (3. fejezet). Külön részben foglalkozunk a különböző ütemezési algoritmusok összehasonlításával (4. fejezet).

1.2. Ütemezési probléma

A $\pi = (\rho, \tau, \mu)$ ütemezési probléma egy hármas, melynek elemei a ρ erőforrásrendszer, a τ taszkrendszer és a μ teljesítménymérték.

1.2.1. Erőforrásrendszer

A taszkok egy részének végrehajtásához egyféle erőforrás elegendő. Ezeket az erőforrásokat *processzoroknak* nevezzük: m-mel jelöljük a processzorok számát, $\mathbf{P} = \{P_1, \ldots, P_m\}$-mel pedig a processzorok halmazát, illetve az egyes processzorokat.

Ha a P_i és $P_j \in \mathbf{P}$ processzorokon végrehajtható feladatok halmazai különbözőek, akkor azt mondjuk, hogy P_i és P_j *különböző* processzorok. Ha két processzoron ugyanazok a taszkok hajthatók végre, de a végrehajtási idők nem minden taszkra azonosak, akkor *különböző sebességű* processzorokról beszélünk. Ha két processzoron ugyanazok a taszkok hajthatók végre, és a végrehajtási idők is megegyeznek, akkor a processzorok *azonosak*.

Általános esetben a processzorokon kívül további, úgynevezett *kiegészítő erőforrásokra* is szükség van. Ezek típusainak számát jelöljük s-sel, a típusok halmazát, illetve az egyes típusokat $\mathbf{R} = \{R_1, \ldots, R_s\}$-sel, továbbá jelölje az $\mathbf{r} = (r_1, \ldots, r_s)$ vektor az egyes kiegészítő erőforrás típusokból rendelkezésre álló mennyiségeket.

A $\rho = (\mathbf{P}, \mathbf{R}, \mathbf{r})$ rendezett hármast nevezzük *erőforrásrendszernek*.

1.2.2. Taszkrendszer

A taszkok legfontosabb tulajdonságai rendszerint leírhatók a következő jelölések segítségével. Adott $\rho = (\mathbf{P}, \mathbf{R}, \mathbf{r})$ erőforrásrendszerre vonatkozó taszkrendszer egy $\tau = (\mathbf{T}, <, [t_{ij}], [r_{ik}], \mathbf{b}, \mathrm{d}, \mathbf{p})$ hetes, ahol

1. $\mathbf{T} = \{T_1, \ldots, T_n\}$ a végrehajtandó taszkok halmaza,

2. $<$ egy a $\mathbf{T}$ halmazon értelmezett tranzitív, irreflexív, aszimmetrikus bináris reláció ($< \subseteq \mathbf{T} \times \mathbf{T}$).
   Ha $T_i, T_j \in \mathbf{T}$, akkor $T_i < T_j$ azt jelenti, hogy T_i végrehajtását be kell fejezni, mielőtt T_j végrehajtását elkezdenénk. Ilyenkor azt mondjuk, hogy T_i megelőzi T_j-t. $<$-t a taszkrendszer *megelőzési relációjának* hívjuk.

Ha sem $T_i < T_j$, sem $T_j < T_i$ nem teljesül, akkor azt mondjuk, hogy T_i és T_j függetlenek egymástól. Ha egy adott időpontban minden olyan T_k feladatot végrehajtottunk, amelyre $T_k < T_i$, és a T_i végrehajtásához szükséges erőforrások rendelkezésre állnak, akkor T_i az adott időpontban *végrehajtható* (végrehajtása megkezdhető). A megelőzési reláció gyakran előforduló példái: lánc, fa és független taszkok.

3. $[t_{ij}]$ a *végrehajtási idők* mátrixa: t_{ij} $(1 \le i \le n,\ 1 \le j \le m)$ azt az időt jelenti, amely alatt T_i végrehajtható P_j-n. Annak szokásos jelölése, hogy T_i nem hajtható végre P_j-n: $t_{ij} = \infty$. Általában feltesszük, hogy a mátrix minden eleme pozitív (a taszkok végrehajtásához minden processzoron időre van szükség), és a mátrix minden sorában van véges érték (minden taszk legalább egy processzoron végrehajtható). Ha a processzorok azonosak, akkor $t_{i1} = t_{i2} = \cdots = t_{im}$ $(i = 1, 2, \ldots, n)$. Ilyenkor T_i végrehajtási idejét egyszerűen t_i-vel jelöljük (és végrehajtási idők $[t_{ij}]$ mátrixa helyett végrehajtási idők $\mathbf{t}$ vektora szerepel τ-ban).

4. $[r_{ik}]$ a *kiegészítő erőforrásokra vonatkozó igények* mátrixa: r_{ik} $(1 \le i \le n,\ 1 \le k \le s)$ azt adja meg, hogy a T_i taszk végrehajtásához az R_k erőforrástípusból hány egységre van szükség (a végrehajtás egész időtartamára). Általában feltesszük, hogy $r_{ik} \le r_k$ $(1 \le i \le n,\ 1 \le k \le s)$ teljesül, azaz a taszkok erőforrásigénye kielégíthető. Ha $s = 0$, akkor ezzel a mátrixszal nem kell foglalkozni.

5. $\mathbf{b} = (b_1, \ldots, b_n)$ a *belépési időpont*ok vektora: b_i jelenti azt az időpontot, amely előtt T_i végrehajtása nem kezdhető el.

6. $\mathbf{d} = (d_1, \ldots, d_n)$ a *határidők* vektora: d_i jelenti azt az időpontot, ameddig a T_i taszk végrehajtását be kell fejezni. A határidők lehetnek *szigorúak,* illetve *ajánlottak.* Szigorú határidő esetén csak a taszkok olyan végrehajtása fogadható el, amely biztosítja a határidők betartását, míg az ajánlott határidők túllépése megengedett (bár a túllépésnek hátrányos következményei lehetnek).

7. $\mathbf{p} = (p_1, \ldots, p_n)$ a *súlyok* vektora: p_i a T_i taszk fontosságát jellemzi (a teljesítménymértékek értékén keresztül befolyásolja a taszkok végrehajtását).

1.2.3. Teljesítménymértékek

A taszkok jellemzőit két csoportba oszthatjuk: az egyik csoportba tartozók (mint $b_i, d_i, p_i, t_{ij}, r_{ik}$) az ütemezés elkészítésekor ismert, állandó értékek — a másik csoportba tartozók pedig az ütemezés elkészítése során kapják meg az értéküket (attól

függ az értékük, hogy milyen ütemezést választunk):
Az ütemezést S-sel jelölve vezessük be a következő jelöléseket:

$s_i(S)$	a T_i taszk *kezdési időpontja*
$f_i(S)$	a T_i taszk *befejezési időpontja*
$w_i(S) = s_i(S) - b_i$	a T_i taszk *várakozási ideje*
$n_i(S) = f_i(S) - d_i$	a T_i taszk *pontatlansága*
$l_i(S) = \max(n_i(S), 0)$	a T_i taszk *késése*

Ezen ütemezéstől függő taszkjellemzők segítségével fogjuk megadni a $\mu = \{F_{\max}$, $F_{\min}$, F_{av}, $W_{\max}$, $W_{\min}$, W_{av}, $N_{\max}$, $N_{\min}$, N_{av}, $L_{\max}$, $L_{\min}$, L_{av}, $E_{\max}$, $E_{\min}$, E_{av}, $A_{\max}$, $A_{\min}$, $A_{\text{av}}\}$ teljesítménymértékeket (amelyek az ütemezést jellemzik). Pontosabban csak $F_{\dots}$, $W_{\dots}$, $N_{\dots}$ és $L_{\dots}$ származik közülük a fenti taszkjellemzőkből, $E_{\dots}$ és $A_{\dots}$ pedig rendre az $E(S,t) = |\{T_i : f_i(S) > t\}|$ (S ütemezés mellett a t időpontban befejezetlen taszkok száma), illetve az $A(S,t) = \sum_{1 \le i \le n,\, f_i(S) > t} p_i$ (S ütemezés mellett a t időpontban befejezetlen taszkok súlyainak összege) jellemzőkből származnak.

A teljesítménymértékek indexe azt jelöli, hogy maximális, minimális avagy súlyozott átlagos jellemzőről van-e szó (a taszkjellemzők közül a maximális, minimális, illetve súlyokkal számolt átlagos érték). Tehát például $F_{\max}$ a maximális befejezési idő, $F_{\min}$ a minimális befejezési idő, F_{av} pedig a súlyozott átlagos befejezési idő. (Hasonlóan $W_{\dots}$ a várakozási idő, $N_{\dots}$ a pontatlanság, $L_{\dots}$ pedig a késés taszkonként vett maximuma, minimuma, illetve súlyozott átlaga.) Az $E_{\dots}$ és $A_{\dots}$ jellemzők természetszerűleg nem taszkonként vett maximumot, minimumot, illetve átlagot adnak meg, hanem idő szerintit, ahogy az az alábbi összefoglaló táblázatban látható:

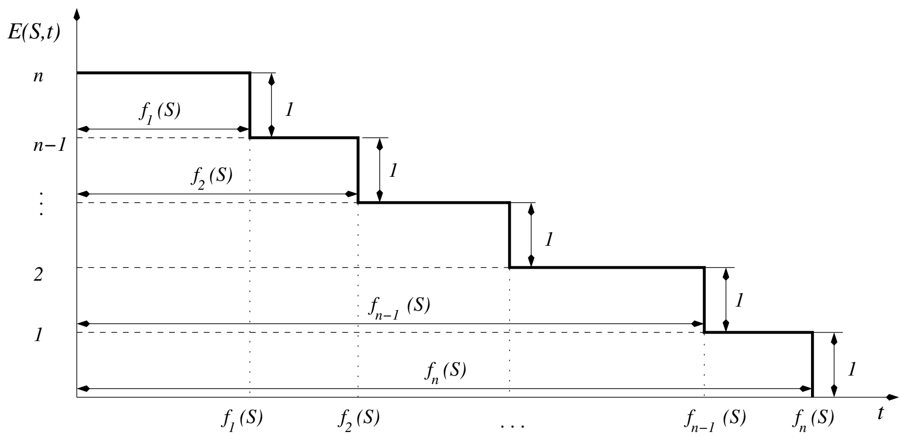
$F_{\max}(S) = \max\limits_{1 \le i \le n} f_i(S)$	$F_{\min}(S) = \min\limits_{1 \le i \le n} f_i(S)$	$F_{\text{av}}(S) = \frac{1}{n} \sum\limits_{i=1}^{n} p_i f_i(S)$
$W_{\max}(S) = \max\limits_{1 \le i \le n} w_i(S)$	$W_{\min}(S) = \min\limits_{1 \le i \le n} w_i(S)$	$W_{\text{av}}(S) = \frac{1}{n} \sum\limits_{i=1}^{n} p_i w_i(S)$
$N_{\max}(S) = \max\limits_{1 \le i \le n} n_i(S)$	$N_{\min}(S) = \min\limits_{1 \le i \le n} n_i(S)$	$N_{\text{av}}(S) = \frac{1}{n} \sum\limits_{i=1}^{n} p_i n_i(S)$
$L_{\max}(S) = \max\limits_{1 \le i \le n} l_i(S)$	$L_{\min}(S) = \min\limits_{1 \le i \le n} l_i(S)$	$L_{\text{av}}(S) = \frac{1}{n} \sum\limits_{i=1}^{n} p_i l_i(S)$
$E_{\max}(S) = \max\limits_{0 \le t \le F_{\max}(S)} E(S,t)$	$E_{\min}(S) = \min\limits_{0 \le t \le F_{\max}(S)} E(S,t)$	$E_{\text{av}}(S) = \frac{1}{F_{\max}(S)} \int\limits_{0}^{F_{\max}(S)} E(S,t)\, \mathrm{d}t$
$A_{\max}(S) = \max\limits_{0 \le t \le F_{\max}(S)} A(S,t)$	$A_{\min}(S) = \min\limits_{0 \le t \le F_{\max}(S)} A(S,t)$	$A_{\text{av}}(S) = \frac{1}{F_{\max}(S)} \int\limits_{0}^{F_{\max}(S)} A(S,t)\, \mathrm{d}t$

Megjegyzés:

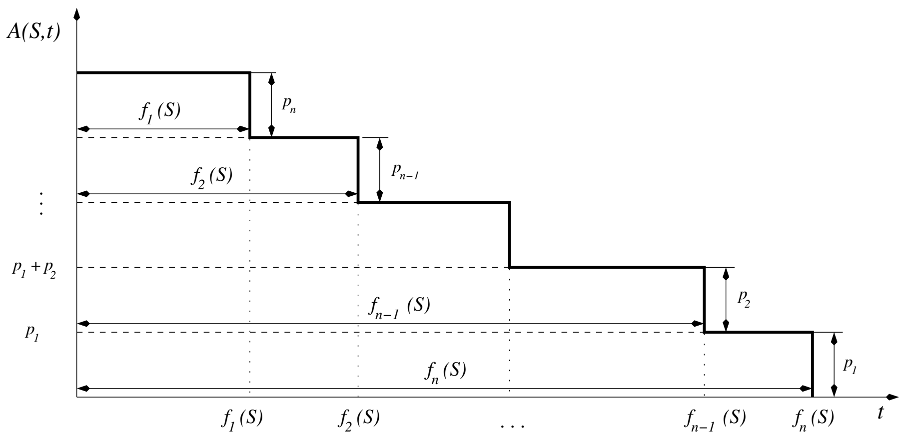
1. $E_{\max}(S) = E(S,0) = n$,
   $A_{\max}(S) = A(S,0) = \sum_{i=1}^{n} p_i$,

2. $E_{\min}(S) = E(S, F_{\max}(S)) = 0$,
   $A_{\min}(S) = A(S, F_{\max}(S)) = 0$,

3. $E_{\mathrm{av}}(S) = \frac{1}{F_{\max}(S)} \sum_{i=1}^{n} f_i(S)$ *(lásd 1.1. ábra)*,
   $A_{\mathrm{av}}(S) = \frac{1}{F_{\max}(S)} \sum_{i=1}^{n} p_i f_i(S)$ *(lásd 1.2. ábra)*.



1.1. ábra. $\int_0^{F_{\max}(S)} E(S,t)\,\mathrm{d}t$ kiszámítása



1.2. ábra. $\int_0^{F_{\max}(S)} A(S,t)\,\mathrm{d}t$ kiszámítása

2. Egyprocesszoros eredmények

2.1. Bevezetés

Ebben a fejezetben viszonylag egyszerű ütemezési problémákat vizsgálunk, melyekben egyetlen mohó és oszthatatlan processzor van ($m = 1$), kiegészítő erőforrás nincs ($s = 0$), a feladatok függetlenek ($<$ üres), és a taszkok végrehajtása nem szakítható meg. A taszkokhoz ajánlott határidők tartoznak, listát nem alkalmazunk.

A vizsgált ütemezési problémák közös formalizált leírása ezért a következő alakra egyszerűsödik: $\pi = (\rho, \tau, \mu)$, $\rho = (\mathbf{P})$, $\mathbf{P} = \{P_1\}$, $\tau = (\mathbf{T}, \mathbf{t}, \mathrm{d}, \mathbf{p})$, $\mathbf{T} = (T_1, \ldots, T_n)$, $\mathbf{t} = (t_1, \ldots, t_n)$, $\mathbf{d} = (d_1, \ldots, d_n)$, $\mathbf{p} = (p_1, \ldots, p_n)$. Továbbá — mivel egyetlen mohó és oszthatatlan processzor van, és csak megszakítás nélküli algoritmusok jöhetnek szóba, ezért — az ütemezés a taszkok egy permutációjának kiválasztását jelenti.

Ezen feltevések mellett a minimális várakozási idő a taszkok végrehajtási sorrendjétől függetlenül nulla ($W_{\min}(S) = 0$, $\forall S$-re). Ugyancsak a sorrendtől független a maximális befejezési idő: $F_{\max}(S) = \sum_{i=1}^{n} t_i$ mind az $n!$ permutációra.

$F_{\min}$ akkor és csak akkor lesz minimális, ha a taszkok végrehajtását a legkisebb végrehajtási idejű taszkkal kezdjük, míg $F_{\min}$ maximális értéke a legnagyobb végrehajtási idejű taszkkal való kezdés esetén érhető el.

A maximális várakozási idő az utoljára végrehajtott taszk jó megválasztásával befolyásolható: a végrehajtást a legkisebb végrehajtási idejű taszkkal befejezve $W_{\max}$ maximális lesz, míg a legnagyobb végrehajtási idejű taszkkal befejezve minimális lesz.

Jelölések. A továbbiakban az „a taszkokat a β_i szerint növekvő sorrendben hajtjuk végre” kifejezés helyett a „$\beta_i \nearrow$” jelölést, az „akkor és csak akkor, ha” kifejezés helyett pedig a „$\Leftrightarrow$” jelölést is használjuk. Értelemszerű jelentéssel a „$\beta_i \searrow$”, „$\Rightarrow$” és „$\Leftarrow$” jelöléseket is alkalmazzuk. Az adott ütemezés szerint j.-ként végrehajtott feladat indexét pedig i_j-vel ($j = 1, \ldots, n$) jelöljük. Végül a „μ maximális” és „μ minimális” rövidítésére a „$\mu\uparrow$”, illetve „$\mu\downarrow$” jelölések szolgálnak.

Ezekkel a jelölésekkel például az $F_{\min}$-re vonatkozó állítások röviden így írhatók fel: $F_{\min}\downarrow \Leftrightarrow t_{i_1} = \min\limits_{1 \le i \le n} t_i$, illetve $F_{\min}\uparrow \Leftrightarrow t_{i_1} = \max\limits_{1 \le i \le n} t_i$.

A következő három alfejezetben néhány átlagos, maximális, illetve minimális jellemző szélsőértékeinek elérését biztosító feltételt adunk meg.

2.2. Átlagos jellemzők

Az átlagos súlyozott befejezési idő, várakozási idő és pontatlanság szélsőértékei a $\frac{t_i}{p_i}$ szerint rendezett permutációkhoz tartoznak.

2.1. tétel. F_{av} N_{av} és W_{av} akkor és csak akkor lesz minimális (maximális), ha a taszkokat $\frac{t_{i_j}}{p_{i_j}}$ szerint növekvő (csökkenő) sorrendben hajtjuk végre.
Ennek az állításnak tömör írásmódja:

$$F_{\text{av}}, N_{\text{av}}, W_{\text{av}} \downarrow \Leftrightarrow \frac{t_{i_j}}{p_{i_j}} \nearrow, \text{ és } F_{\text{av}}, N_{\text{av}}, W_{\text{av}} \uparrow \Leftrightarrow \frac{t_{i_j}}{p_{i_j}} \searrow .$$

Bizonyítás. a) Mivel a feladatoknak csak véges sok ($n!$) ütemezése van, ezek között kell lennie optimálisnak.

b) Megmutatjuk, hogy F_{av} csak akkor lehet minimális, ha a taszkokat a $\frac{t_{i_j}}{p_{i_j}}$ szerint növekvő sorrendben hajtjuk végre.

Tegyük fel, hogy van olyan S ütemezés, azaz a taszkoknak olyan $T_{i_1}, \ldots, T_{i_n}$ permutációja, amelyre $F_{\text{av}}(S)$ minimális, de ugyanakkor van olyan k ($1 \leq k \leq n-1$) index, amelyre

$$\frac{t_{i_k}}{p_{i_k}} > \frac{t_{i_{k+1}}}{p_{i_{k+1}}} . \tag{2.1}$$

Vizsgáljuk most meg a T_{i_k} és $T_{i_{k+1}}$ taszkok sorrendjének felcserélésével S-ből előállítható S' ütemezést, és a hozzá tartozó $F_{\text{av}}(S')$ átlagos befejezési időt.

F_{av} új értéke két ok miatt különbözik a régitől: T_{i_k} befejezési időpontja a csere miatt $t_{i_{k+1}}$-gyel nőtt, míg $T_{i_{k+1}}$ befejezési időpontja t_{i_k}-val csökkent. Ezért

$$F_{\text{av}}(S') = \frac{1}{n}\left(p_{i_k} t_{i_{k+1}} - p_{i_{k+1}} t_{i_k}\right) + \underbrace{\frac{1}{n}\sum_{i=1}^{n} p_i f_i(S)}_{F_{\text{av}}(S)} . \tag{2.2}$$

(2.1) átrendezésével azt kapjuk, hogy $p_{i_k} t_{i_{k+1}} < p_{i_{k+1}} t_{i_k}$, amiből (2.2) alapján $F_{\text{av}}(S') < F_{\text{av}}(S)$ következik. Mivel ez ellentmond annak, hogy $F_{\text{av}}(S)$ minimális, ezért csak $\frac{t_i}{p_i}$ szerint növekvő sorrendhez tartozhat minimális F_{av}.

c) A szükségesség után most a vizsgált feltétel elégségességét is belátjuk. A bizonyítás b) része szerint tehát az egyik „jó” (növekvő) permutáció adja az optimumot. Ha a $\frac{t_i}{p_i}$ hányados két taszkra nézve azonos, akkor (2.1) és (2.2) alapján sorrendcseréjük F_{av} értékét nem befolyásolja, tehát valamennyi „jó” sorrend minimális F_{av} értékre vezet.
Ezzel a tétel F_{av}re vonatkozó részét bebizonyítottuk.

d) Az átlagos súlyozott várakozási idő és pontatlanság definíciója alapján

$$W_{\mathrm{av}}(S) = \frac{1}{n}\sum_{i=1}^{n} p_i w_i(S) = \frac{1}{n}\sum_{i=1}^{n} p_i \underbrace{(f_i(S) - t_i)}_{s_i(S)} = F_{\mathrm{av}}(S) - \frac{1}{n}\sum_{i=1}^{n} p_i t_i, \tag{2.3}$$

$$N_{\mathrm{av}}(S) = \frac{1}{n}\sum_{i=1}^{n} p_i n_i(S) = \frac{1}{n}\sum_{i=1}^{n} p_i (f_i(S) - d_i) = F_{\mathrm{av}}(S) - \frac{1}{n}\sum_{i=1}^{n} p_i d_i. \tag{2.4}$$

(2.3) és (2.4) szerint W_{av} illetve N_{av} csak egy additív konstanssal különböznek F_{av}-től, tehát akkor és csak akkor lesznek minimálisak, ha F_{av} is az.

e) Az eddigiekben megadtuk annak szükséges és elégséges feltételét, hogy a vizsgált teljesítménymértékek értékei minimálisak legyenek. Ha a b), c) és d) részben a „minimális” helyett „maximális”, „növekvő” helyett „csökkenő”, „$<$” helyett „$>$” és „$>$” helyett „$<$” szerepel, akkor a maximális értékhez tartozó bizonyítást kapjuk.

Most megmutatjuk, hogy a 2.1. tételhez hasonló állítás teljesül E_{av} és A_{av} értékére is. Láttuk, hogy

$$E_{\mathrm{av}}(S) = \frac{1}{F_{\max}(S)}\sum_{i=1}^{n} f_i(S). \tag{2.5}$$

Mivel $F_{\max}(S)$ értéke állandó, így (a 2.5) egyenlőségből és a 2.1. tételből következik, hogy E_{av} akkor és csak akkor minimális, ha a taszkokat t_i növekvő sorrendjében ütemezzük (ugyanis $F_{\mathrm{av}}(S) = \frac{1}{n}\sum_{i=1}^{n} f_i(S)$, így $p_i = 1$, $(i = 1, \ldots, n)$ esetén $E_{\mathrm{av}}(S) = \frac{1}{F_{\max}(S)} n F_{\mathrm{av}}(S)$, amiből következik, hogy $E_{\mathrm{av}}\downarrow \Leftrightarrow F_{\mathrm{av}}\downarrow$, vagyis $E_{\mathrm{av}}\downarrow \Leftrightarrow t_i\nearrow$.

Hasonló gondolatmenet alkalmazható A_{av} esetén is: Láttuk, hogy

$$A_{\mathrm{av}}(S) = \frac{1}{F_{\max}(S)}\sum_{i=1}^{n} p_i f_i(S). \tag{2.6}$$

Ezt felhasználva $A_{\mathrm{av}} = \frac{1}{F_{\max}(S)} n F_{\mathrm{av}}(S)$, amiből — mivel $F_{\max}(S)$ konstans — következik, hogy $A_{\mathrm{av}}\downarrow \Leftrightarrow F_{\mathrm{av}}\downarrow$, vagyis $A_{\mathrm{av}}\downarrow \Leftrightarrow t_i\nearrow$.

Természetesen ugyanígy E_{av} és A_{av} maximumára is levezethetjük a szükséges és elégséges feltételt.

2.3. Maximális jellemzők

A maximális késés és a maximális pontatlanság minimalizálásához elégséges feltételt biztosít a következő állítás.

2.2. tétel. Ha a taszkokat a d_{i_j} ajánlott határidők növekvő sorrendjében ütemezzük, akkor az $N_{\max}$ maximális pontatlanság és az $L_{\max}$ maximális késés minimális lesz.

Bizonyítás. a) Mivel a taszkoknak csak véges sok permutációja van, ezért biztosan van optimális ütemezés.

b) Először a maximális pontatlanságra vonatkozó részt látjuk be. Tegyük fel, hogy van olyan S ütemezés, azaz a taszkoknak olyan $T_{i_1}, \ldots, T_{i_n}$ permutációja, melyre $N_{\max}(S)$ minimális, de ugyanakkor van olyan j index $(1 \le j \le n-1)$, melyre

$$d_{i_j} > d_{i_{j+1}}. \tag{2.7}$$

Ekkor a T_{i_j} és $T_{i_{j+1}}$ taszkok pontatlanságai:

$$n_{i_j}(S) = f_{i_j}(S) - d_{i_j},\ n_{i_{j+1}}(S) = f_{i_{j+1}}(S) - d_{i_{j+1}} = f_{i_j}(S) + t_{i_{j+1}} - d_{i_{j+1}}. \tag{2.8}$$

Innen (2.7) és $t_{i_{j+1}} > 0$ figyelembevételével $n_{i_{j+1}} < n_{i_j}$, és így $N_{\max}(S) = \max\limits_{1 \le k \le n} n_{i_k} \ge n_{i_{j+1}}$.

Cseréljük fel az S ütemezésben a T_{i_j} és $T_{i_{j+1}}$ taszkok sorrendjét, és az így kapott új ütemezést jelöljük S'-vel. A továbbiakban az S'-höz tartozó jellemzők jelét vesszővel, az S-hez tartozók jelét pedig vessző nélkül használjuk. Az új ütemezéssel a pontatlanságok:

$$n'_{i_j} = f'_{i_j} - d_{i_j} = f_{i_j} + t_{i_{j+1}} - d_{i_j},\ n'_{i_{j+1}} = f'_{i_{j+1}} - d_{i_{j+1}} = \underbrace{f_{i_j} + t_{i_{j+1}}}_{f_{i_{j+1}}} - t_{i_j} - d_{i_{j+1}}. \tag{2.9}$$

Figyeljük meg, hogy $n'_{i_j} < n_{i_{j+1}}$ és $n'_{i_{j+1}} < n_{i_{j+1}}$, ezért $N_{\max}(S') \le N_{\max}(S)$, tehát a cserétől az $N_{\max}$ értéke nem nőtt.

Véges sok hasonló cserével elérhető, hogy a feladatok határidők szerint növekvő sorrendjéhez jussunk. Ha akár egyetlen cserével a korábbinál kisebb pontatlanságot kapunk, akkor az ellentmondás állításunk helyességét bizonyítja. Ha minden csere után változatlan marad a maximális pontatlanság, akkor is igaz, hogy az így megadott algoritmus minimalizálja $N_{\max}$-ot.

c) Hasonlóképpen láthatjuk be, hogy a határidők szerint növekvő sorrend esetén $L_{\max}$ minimális lesz. Itt is igaz, hogy csak véges sok ütemezés van.

Ismét tegyük fel, hogy van olyan $T_{i_1}, \ldots, T_{i_n}$ permutáció, melyre $L_{\max}$ minimális, és ugyanakkor van olyan j index, melyre (2.7) fennáll. Cseréljük fel a T_{i_j} és $T_{i_{j+1}}$ taszkokat. A csere előtti ütemezést jelöljük S-sel, a csere utánit S'-vel. Ekkor a csere előtti késések:

$$l_{i_j} = \max(0, f_{i_j} - d_{i_j}), \quad l_{i_{j+1}} = \max(0, f_{i_j} + t_{i_{j+1}} - d_{i_{j+1}}. \tag{2.10}$$

Mivel most (2.7) és $t_{i_{j+1}} > 0$ alapján $l_{i_{j+1}} \geq l_{i_j}$, ezért $L_{\max}(S) = \max\limits_{1 \leq i \leq n} l_i \geq l_{i_{j+1}}$. Az új késések értéke:

$$l'_{i_j} = \max(0, f_{i_j} + t_{i_{j+1}} - d_{i_j}), \quad l_{i_{j+1}} = \max(0, f_{i_j} + t_{i_{j+1}} - t_{i_j} - d_{i_{j+1}}). \tag{2.11}$$

Mivel most $l'_{i_j} < l_{i_{j+1}}$ és $l'_{i_{j+1}} < l_{i_{j+1}}$, ezért $L_{\max}(S') \leq L_{\max}(S)$.

Véges sok hasonló cserével a határidők szerint növekvő taszk sorrendet kapunk. Akár volt olyan csere, amely után a maximális késés csökkent, akár minden cserénél változatlan maradt $L_{\max}$ értéke, mindkét esetben igaz, hogy a tétel $L_{\max}$ra vonatkozó részét is bizonyítottuk.

Tekintsük most azokat a T_1, T_2 és T_3 feladatokat, melyekre $d_1 = 1$, $d_2 = 2$, $d_3 = 3$, a végrehajtási idők pedig $t_1 = t_2 = t_3 = 10$. Ezeket a taszkokat a határidők növekvő sorrendjében végrehajtva $N_{\max}(S) = L_{\max}(S) = \max(9, 18, 27) = 27$, míg az előzőtől eltérő T_2, T_1, T_3 sorrend esetén ugyancsak $N_{\max}(S) = L_{\max}(S) = \max(8, 19, 27) = 27$. A példából látható, hogy — a 2.1. tételben tapasztaltakkal ellentétben — még különböző jellemzők (itt: határidők) esetén sem egyértelmű az optimális sorrend.

A teljesség kedvéért vizsgáljuk meg annak feltételét is, hogy $N_{\max}$ illetve $L_{\max}$ maximális legyen. A maximális pontatlanság akkor és csak akkor lesz maximális, ha utoljára a legkisebb határidejű taszkot hajtjuk végre. $L_{\max}$ maximalizálásakor két esetet kell megkülönböztetnünk: ha a legkisebb határidő sem kisebb, mint a végrehajtási idők összege, akkor $L_{\max}(S) = 0$ mind az $n!$ lehetséges ütemezésre. Ha viszont van a végrehajtási idők összegénél kisebb határidő, akkor a szükséges és elégséges feltétel $d_{i_n} = \min\limits_{1 \leq i \leq n} d_i$ lesz.

2.4. Minimális jellemzők

A minimális pontatlanság és a minimális késés minimalizálásához jó elvnek tűnhet a tartalékidők ($sp_i = d_i - t_i$) alapján való ütemezés. Némiképp meglepő ezért a következő állítás.

2.3. tétel. Ha a taszkokat a tartalékidők növekvő sorrendjében ütemezzük, akkor az $N_{\min}$ minimális pontatlanság és az $L_{\min}$ minimális késés maximális lesz.

Bizonyítás. a) Mivel a taszkoknak csak véges sok permutációja van, az optimális megoldás létezik.

b) Megmutatjuk, hogy a tartalékidők növekvő sorrendje esetén $N_{\min}$ maximális lesz. Tegyük fel, hogy van olyan S ütemezés, azaz a taszkoknak olyan $T_{i_1}, \ldots, T_{i_n}$ sorrendje, amelyre $N_{\min}$ maximális, de ugyanakkor van olyan j index ($1 \leq j \leq n-1$), amelyre

$$sp_{i_j}(S) > sp_{i_{j+1}}(S). \tag{2.12}$$

Cseréljük fel az S ütemezésben a T_{i_j} és a $T_{i_{j+1}}$ taszkokat (így kapjuk az S' ütemezést). A felcserélt taszkok eredeti pontatlanságai:

(2.13)

$$n_{i_j} = f_{i_j} - d_{i_j} = f_{i_j} - t_{i_j} - sp_{i_j}, \quad n_{i_{j+1}} = f_{i_{j+1}} - d_{i_{j+1}} = f_{i_j} + t_{i_{j+1}} - d_{i_{j+1}} = f_{i_j} - sp_{i_{j+1}}.$$

Innen (2.12) és $t_{i_j} > 0$ figyelembevételével $n_{i_j} < n_{i_{j+1}}$, és így $N_{\min} = \min_{1 \leq i \leq n} n_i \leq n_{i_j}$. A csere utáni pontatlanságok:

(2.14)

$$\begin{aligned} n'_{i_j} &= f'_{i_j} - d_{i_j} = f_{i_j} + t_{i_{j+1}} - d_{i_j} = f_{i_j} + t_{i_{j+1}} - t_{i_j} - sp_{i_j} = n_{i_j} + t_{i_{j+1}}, \\ n'_{i_{j+1}} &= f'_{i_{j+1}} - d_{i_{j+1}} = \underbrace{f_{i_j} + t_{i_{j+1}}}_{f_{i_{j+1}}} - t_{i_j} - d_{i_{j+1}} = f_{i_j} - t_{i_j} - sp_{i_{j+1}}. \end{aligned}$$

Mivel $n'_{i_j} > n_{i_j}$ és $n'_{i_{j+1}} > n_{i_j}$, így $N_{\min}(S') \geq N_{\min}(S)$. Tehát a cserétől a minimális pontatlanság értéke nem csökkent, így — S optimalitása miatt — ugyanakkora maradt.

Véges sok cserével elérhető, hogy a taszkoknak a tartalékidők szerint növekvő sorrendjéhez jussunk. Tehát a tartalékidők növekvő sorrendjében való ütemezés maximalizálja $N_{\min}$-t.

c) Hasonló módon látjuk be, hogy a növekvő tartalékidők szerinti ütemezés esetén $L_{\min}$ maximális. Tegyük fel, hogy van olyan S optimális permutáció és ahhoz olyan j ($1 \leq j \leq n-1$) index, melyre (2.12) fennáll. Ekkor

(2.15)

$$\begin{aligned} l_{i_j} &= \max(0, f_{i_j} - d_{i_j}) = \max(0, f_{i_j} - t_{i_j} - sp_{i_j}), \\ l_{i_{j+1}} &= \max(0, f_{i_{j+1}} - d_{i_{j+1}}) = \max(0, f_{i_j} + t_{i_{j+1}} - d_{i_{j+1}}) = \max(0, f_{i_j} - sp_{i_{j+1}}). \end{aligned}$$

Innen (2.12) és $t_{i_j} > 0$ alapján $l_{i_j} \leq l_{i_{j+1}}$, így $L_{\min} = \min_{1 \leq k \leq n} l_k \leq l_{i_j}$. Cseréljük fel most a T_{i_j} és $T_{i_{j+1}}$ taszkokat. Az így kapott S' ütemezés szerinti késések:

(2.16)

$$l'_{i_j} = \max(0, \underbrace{f_{i_j} + t_{i_{j+1}}}_{f'_{i_j}} - d_{i_j}) = \max(0, f_{i_j} + t_{i_{j+1}} - t_{i_j} - sp_{i_j}),$$

$$l'_{i_{j+1}} = \max(0, \underbrace{\overbrace{f_{i_j} + t_{i_{j+1}}}^{f_{i_{j+1}}} - t_{i_j}}_{f'_{i_{j+1}}} - d_{i_{j+1}}) = \max(0, f_{i_j} - t_{i_j} - sp_{i_{j+1}}).$$

Mivel $l_{i_j} \geq l'_{i_j}$ és $l_{i_j} \geq l'_{i_{j+1}}$, így $L_{\min}(S') \geq L_{\min}(S)$. Ebből S optimalitása miatt a két utóbbi érték egyenlősége következik.

Véges sok hasonló cserével elérhető a növekvő tartalékidők szerinti sorrend, amiből állításunk következik.

Tekintsük most újra a 2.3. rész végén vizsgált taszkrendszert. Legyen az S ütemezés a T_1, T_2, T_3 permutációval adott, míg az S' ütemezés a T_1, T_3, T_2 sorrendhez tartozzon. Most a tartalékidők $sp_1 = -9$, $sp_2 = -8$, $sp_3 = -7$, azaz S a 2.3. tétel szerinti ütemezés. Ezekkel az adatokkal $N_{\min}(S) = L_{\min}(S) = \min(9, 18, 27) = 9$ és $N_{\min}(S') = L_{\min}(S') = \min(9, 17, 28) = 9$. A példából látható, hogy a tételben szereplő feltétel valóban csak elégséges, de nem szükséges.

Vizsgáljuk meg most annak feltételeit is, hogy ez a két teljesítménymérték minimális értéket vegyen fel. $N_{\min}$ akkor és csak akkor lesz minimális, ha először a legnagyobb tartalékidejű taszkot hajtjuk végre. $L_{\min}$ esetében két lehetőség van attól függően, hogy van-e nemnegatív tartalékidejű taszk. Ha van, úgy $L_{\min}$ akkor és csak akkor lesz minimális, ha van olyan taszk, amelynek végrehajtását határidőre befejezzük (ekkor a minimális késés nulla). Ha viszont minden tartalékidő negatív, úgy az a keresett szükséges és elégséges feltétel, hogy a legnagyobb tartalékidejű taszkot hajtsuk végre először.

2.5. Egyprocesszoros eredmények összefoglalása

A 2.1. táblázatban összefoglaltuk az egyprocesszoros ütemezési eredményeket. A táblázatból hiányzik az átlagos súlyozott késés szélsőértékeihez tartozó feltétel, mivel nem ismeretes olyan polinomiális algoritmus, amely ezt az ütemezési problémát megoldaná.

Sorsz.	Minimum feltétele	Maximum feltétele
1.	$W_{\min} = 0$	ütemezéstől függetlenül
2.	$E_{\min} = 0$	ütemezéstől függetlenül
3.	$A_{\min} = 0$	ütemezéstől függetlenül
4.	$F_{\max} = \sum_{i=1}^{n} t_i$	ütemezéstől függetlenül
5.	$E_{\max} = n$	ütemezéstől függetlenül
6.	$A_{\min} = \sum_{i=1}^{n} p_i$	ütemezéstől függetlenül
7.	$F_{\min} \downarrow \Leftrightarrow t_{i_1} = \min_{1 \le j \le n} t_j$	$F_{\min} \uparrow \Leftrightarrow t_{i_1} = \max_{1 \le j \le n} t_j$
8.	$W_{\max} \downarrow \Leftrightarrow t_{i_n} = \max_{1 \le j \le n} t_j$	$W_{\max} \uparrow \Leftrightarrow t_{i_n} = \min_{1 \le j \le n} t_j$
9.	$F_{\text{av}} \downarrow \Leftrightarrow \frac{t_{i_j}}{p_{i_j}} \nearrow$	$F_{\text{av}} \uparrow \Leftrightarrow \frac{t_{i_j}}{p_{i_j}} \searrow$
10.	$W_{\text{av}} \downarrow \Leftrightarrow \frac{t_{i_j}}{p_{i_j}} \nearrow$	$W_{\text{av}} \uparrow \Leftrightarrow \frac{t_{i_j}}{p_{i_j}} \searrow$
11.	$N_{\text{av}} \downarrow \Leftrightarrow \frac{t_{i_j}}{p_{i_j}} \nearrow$	$N_{\text{av}} \uparrow \Leftrightarrow \frac{t_{i_j}}{p_{i_j}} \searrow$
12.	$E_{\text{av}} \downarrow \Leftrightarrow \frac{t_{i_j}}{p_{i_j}} \nearrow$	$E_{\text{av}} \uparrow \Leftrightarrow \frac{t_{i_j}}{p_{i_j}} \searrow$
13.	$A_{\text{av}} \downarrow \Leftrightarrow \frac{t_{i_j}}{p_{i_j}} \nearrow$	$A_{\text{av}} \uparrow \Leftrightarrow \frac{t_{i_j}}{p_{i_j}} \searrow$
14.	$N_{\max} \downarrow \Leftarrow d_{i_j} \nearrow$	$N_{\max} \uparrow \Leftrightarrow d_{i_n} = \min_{1 \le j \le n} d_j$
15.	$L_{\max} \downarrow \Leftarrow d_{i_j} \nearrow$	$L_{\max} = 0 \Leftarrow F_{\max} \le \min_{1 \le j \le n} d_j$
		$N_{\max} \uparrow \Leftrightarrow d_{i_n} = \min_{1 \le j \le n} d_j$
16.	$N_{\min} \downarrow \Leftrightarrow sp_{i_1} = \max_{1 \le j \le n} sp_j$	$N_{\min} \uparrow \Leftarrow sp_{i_j} \nearrow$
17.	$L_{\min} = 0 \Leftrightarrow \exists j : f_{i_j} \le d_{i_j}$, ha $\exists i : sp_i \ge 0$	$L_{\min} \uparrow \Leftarrow sp_{i_j} \nearrow$
	$L_{\min} \downarrow \Leftrightarrow sp_{i_1} = \max_{1 \le j \le n} sp_j$, egyébként	
18.	$L_{\text{av}} \downarrow ???$	$L_{\text{av}} \uparrow ???$

2.1. táblázat. Egyprocesszoros eredmények összefoglalása

3. Maximális befejezési idő minimalizálása

3.1. Bevezetés

Ebben a fejezetben olyan ütemezési problémákat tárgyalunk, melyekben a maximális befejezési idő minimalizálása a cél. Ebből adódik, hogy például a határidők és a súlyok nem befolyásolják a taszkok ütemezését.

A vizsgált problémákban kiegészítő erőforrások nem szerepelnek, így a formális modell a következő: $\pi = (\rho, \tau, \mu)$, $\rho = (\mathbf{P})$, $\mathbf{P} = \{P_1, \ldots, P_m\}$, $\tau = (\mathbf{T}, <, [t_{ij}])$, $\mu = F_{\max} \downarrow$.

A processzorok az esetek többségében azonosak lesznek.

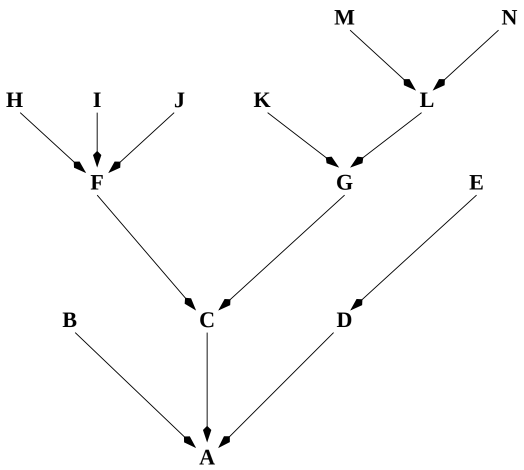
A vizsgált ütemezési problémák mindegyikének megoldására ismeretes polinomiális algoritmus. A fejezet végén azt is megmutatjuk, hogy a korábban vizsgált anomáliák mértéke mekkora lehet.

3.2. Fastruktúrájú taszkrendszerek

Tegyük fel, hogy adott m azonos processzor, amelyek mohóak. A taszkok legyenek megszakíthatatlanok, egységnyi végrehajtási idejűek, és a megelőzési relációt fa adja meg.

Először tekintsük a 3.1. ábrán bemutatott példát, ahol a taszkokat az A, ..., N betűkkel jelöltük, és az egységnyi végrehajtási időket elhagytuk. Becsüljük meg $F_{\max}(S)$ értékét. Mivel 14 taszk van, ezért $m \geq 1$ mohó processzor esetén $F_{\max}(S) \leq 14$ bármely szóba jövő algoritmusra. Mivel van öt taszkot tartalmazó lánc (például N$<$L$<$G$<$C$<$A), ezért $F_{\max} \geq 5$. Tehát ha $m \geq 1$, akkor $5 \leq F_{\max}(S) \leq 14$ minden A algoritmusra.

A pontosabb becsléshez rögzítsük a processzorok számát: legyen $m = 3$. 14 megszakíthatatlan taszk végrehajtásához 3 processzoron még független taszkok esetén is legalább 5 időegységre van szükség, azaz ismét azt kaptuk, hogy $F_{\max}(S) \geq 5$. Ha a megelőzési korlátozásokat is figyelembe vesszük, akkor látjuk, hogy az A taszkot az



3.1. ábra. Fastruktúrájú taszkrendszer

összes többi taszk megelőzi: mivel 13 taszk 3 processzort legalább 5 időegységre lefoglal, ezért $F_{\max}(S) \geq 6$. Az eddigieket összesítve azt kapjuk, hogy $6 \leq F_{\max}(S) \leq 14$ minden szóba jövő S ütemezésre.

Az egyik lehetséges ütemezési stratégia az, hogy a taszkokat lexikografikus sorrendben (nevük ábécé szerinti sorrendjében) hajtjuk végre. Ekkor a 3.2. ábrán bemutatott S_{lex} ütemezést kapjuk, amelyre $F_{\max}(S_{\text{lex}}) = 8$.

P_1	B	D	F	N	L	G	C	A
P_2	E	I	K	-	-	-	-	-
P_3	H	J	M	-	-	-	-	-

3.2. ábra. Az S_{lex} lexikografikus ütemezés

Egy másik lehetséges megközezelítés, hogy az A befejező feladattól legtávolabbi feladatot igyekszünk ütemezni (több jelölt esetén például az előbbi, lexikografikus módon választva. Ekkor a 3.3. ábrán bemutatott S_{fur} ütemezést kapjuk, amelyre $F_{\max}(S_{\text{fur}}) = 6$.

Mivel korábban láttuk, hogy a maximális befejezési idő legalább 6, ezért S_{fur} optimális ütemezés. A továbbiakban megmutatjuk, hogy a most vizsgált ütemezési problémák esetén a „legtávolabbit először” elv mindig optimális ütemezésre vezet.

Ehhez először vezessük be a taszkok szintjének fogalmát. Tegyük fel, hogy a taszkokat azonos processzorokon hajtjuk végre, azaz a végrehajtási idő minden processzoron azonos. A (definíciójuk szerint körmentes) megelőzési gráfokban egy *taszk szintjét* úgy definiáljuk, mint a taszkból a befejező taszkokhoz vezető utak hossza-

P_1	M	I	L	G	C	A
P_2	N	J	E	B	-	-
P_3	H	K	F	D	-	-

3.3. ábra. Az S_{fur} „legtávolabbit először” ütemezés

inak maximuma. Megjegyezzük, hogy egy út hosszán az út mentén lévő taszkok végrehajtási időinek összegét értjük.

Amint arról már volt szó, ebben a részben a végrehajtási idők egységnyiek, a megelőzési gráf pedig fa, ezért egy taszk szintjét egyszerűen a taszkból az egyetlen befejező taszkhoz vezető egyetlen út mentén levő taszkok száma adja.

Szintes ütemezésről akkor beszélünk, ha valahányszor ütemezendő taszkot kell választanunk, mindannyiszor a végrehajtható és még nem ütemezett taszkok közül a legmagasabb szintűek egyikét választjuk. Ez például a következőképpen valósítható meg: Először az n taszkhoz alkalmas módon hozzárendeljük az $1, 2, \ldots, n$ címkéket, majd a taszkokat címkéik csökkenő sorrendjében felsorolva prioritási listát készítünk, és ezzel a listával *listás ütemezést* végzünk. Ez utóbbi azt jelenti, hogy ha ütemezendő taszkot kell választani, azt a lista sorrendjében keressük.

3.1. algoritmus (LEV). Legyen $\tau = (\mathbf{T}, <, \mathbf{t})$ fastruktúrájú vagy erdőstruktúrájú taszkrendszer, amely egységnyi végrehajtási idejű, megszakíthatatlan taszkokat tartalmaz. Ekkor a fastruktúrájú τ $S_{\text{LEV}}(\tau)$ szintes ütemezése azonos processzorokon a következőképpen állítható elő.
Definiálunk egy $C : \mathbf{T} \to \{1, \ldots, n\}$ „címkéző” leképezést (1–3. lépés), majd a címkék segítségével listásan ütemezünk (4–5. lépés).

1. lépés. Ha $\mathrm{T} \in \mathbf{T}$ a taszkrendszer befejező feladata, akkor $C(\mathrm{T}) := 1$.

2. lépés. Tegyük fel, hogy az $1, \ldots, j-1$ $(j \leq n)$ címkéket már hozzárendeltük bizonyos taszkokhoz. Legyen γ_j azon taszkok halmaza, amelyeknek

a) még nincs címkéjük, és

b) közvetlen rákövetkezőjüknek már van címkéje.

Ha ebben a halmazban T az egyetlen taszk, vagy T rákövetkezőjének a címkéje a legkisebb, akkor $C(\mathrm{T}) := j$. Ha viszont több taszk rendelkezik (azonos) legkisebb címkéjű közvetlen rákövetkezővel, akkor ezek bármelyikét (pl. a lexikografikusan legkisebbet) választhatjuk T-nek, és $C(\mathrm{T}) := j$.

3. lépés. Ha van még címkézetlen taszk, akkor folytassuk a 2. lépéstől, egyébként pedig a 4. lépéstől.

4. lépés. Indexeljük meg a taszkokat oly módon, hogy a $(\mathrm{T}_n, \ldots, \mathrm{T}_1)$ listára fennálljon $C(\mathrm{T}_j) = j$ minden $j = 1, \ldots, n$ címkére.

5. lépés. A $(\mathrm{T}_n, \ldots, \mathrm{T}_1)$ lista segítségével m azonos processzoron adódó ütemezést nevezzük S_{LEV}-nek.

Erdőstruktúrájú τ esetén τ-t kiegészítjük egy 0 végrehajtási idejű taszkkal, melynek τ befejező taszkjai lesznek a közvetlen megelőzői, és erre az új taszkrendszerre alkalmazzuk a fenti lépéseket.

Példa. A 3.1. ábrán szereplő fa esetén egyértelműen adódik, hogy $C(\mathrm{A}) = 1$. Ekkor az a) tulajdonsággal A-n kívül minden taszk rendelkezik, közülük B, C és D rendelkezik a b) tulajdonsággal is. Mindhárom taszk közvetlen rákövetkezőjének címkéje azonos (1), így mindhárom választható T-nek. Legyen például $C(\mathrm{B}) = 2$. Ezután C és D alkotják γ_3-at. Legyen $C(\mathrm{C}) = 3$. Most $\gamma_4 = D, F, G$, és az 1, 3 címkék összehasonlításával $C(\mathrm{D}) := 4$. Ezután $\gamma_5 = \{E, F, G\}$, és F, G közül például F-et választva $C(\mathrm{F}) := 5$. Hasonlóképpen folytatva $C(\mathrm{G}) = 6$, $C(\mathrm{E}) = 7$, $C(\mathrm{H}) = 8$, $C(\mathrm{I}) = 9$, $C(\mathrm{J}) = 10$, $C(\mathrm{K}) = 11$, $C(\mathrm{L}) = 12$, $C(\mathrm{M}) = 13$ és $C(\mathrm{N}) = 14$ adódik.

Ez a címkézés az (N, M, L, K, J, I, H, E, G, F, D, C, B, A) listára, az pedig a 3.4. ábrán látható ütemezésre vezet.

P_1	N	L	H	F	C	A
P_2	M	J	E	D	B	-
P_3	K	I	G	-	-	-

3.4. ábra. Az S_{LEV} ütemezés Gantt-diagramja

A LEV ütemezési algoritmussal kapcsolatban megjegyezzük, hogy a taszkok címkézése nem egyértelmű: ha a γ_j-beli taszkok közül többnek a közvetlen rákövetkezője azonos (és így a vizsgált címkék is azonosak), akkor ezen taszkok közül T tetszőlegesen választható az algoritmus 2. lépésében.

3.1. tétel. Ha egységnyi végrehajtási idejű, megszakíthatatlan taszkokat tartalmazó fastruktúrájú taszkrendszert azonos processzorokra a LEV algoritmussal (3.1. algoritmus) ütemezünk, akkor a maximális befejezési idő minimális lesz.

Bizonyítás. a) Egyszintes (egyetlen taszkot tartalmazó) és kétszintes taszkrendszerekre az állítás közvetlenül adódik.

b) A továbbiakban tegyük fel, hogy a vizsgált taszkrendszerek legalább három szintet tartalmaznak. Azt is tegyük fel, hogy a tétel állításával ellentétben S_{LEV} nem minden esetben optimális. Ekkor van olyan, a taszkok n számát tekintve minimális taszkrendszer, amelyre a szintes ütemezés nem optimális. Egy ilyet $\tau_{\min}$-nel jelölve tehát $F_{\max}(S_{\mathrm{LEV}}(\tau_{\min})) = F > F_{\max}(S_{\mathrm{OPT}}(\tau_{\min})) = F_0$. Természetesen $\tau_{\min}$ megválasztásából következik, hogy az n-nél kevesebb taszkot tartalmazó taszkrendszerek esetén LEV optimális ütemezést ad.

c) Mivel a végrehajtandó taszkok száma rögzített, ezért az üres periódusok száma meghatározza F-et. A befejező taszk mindig az utolsó időegységben hajtódik végre. Mivel S_{LEV} nem optimális, a $[0, F-2]$ intervallumban van üres periódus (egyébként S_{LEV} optimális lenne). Tegyük fel, hogy a p-edik időegységben van üres periódus, ahol $1 \leq p \leq F-2$. Ha $p < F-2$, akkor a p-edik időegység után végrehajtott taszkok a p-edik időegység elején még nem kezdhetők el, hiszen egyébként a processzorok mohósága miatt valamelyiket elkezdtük volna. Tehát ezen taszkok valamelyik megelőzője a p-edik időegységben hajtódott végre. Mivel a fastruktúra miatt minden taszknak legfeljebb egy közvetlen rákövetkezője van, így a $(p+1), \ldots, (F-2)$-edik időegységekben végrehajtott taszkok számai nem haladhatják meg (külön-külön sem) a p-edik időegységben végrehajtott taszkok számát, és így az $(F-2)$-edik időegységben is van legalább egy processzor, amelyik nem dolgozik.

d) Legyen $\tau_{\min}$ befejező taszkja T. S_{LEV} szerint T az F-edik időegységben hajtódik végre, ezért T-nek van olyan közvetlen megelőzője, amelyik az $(F-1)$-edik időegységben hajtódik végre — legyen R egy ilyen taszk. Mivel az $(F-2)$-edik időegységben is van üres periódus, így R-nek van olyan közvetlen megelőzője, amelyik ebben az időegységben hajtódik végre (ha nem lenne, akkor R a processzorok mohósága miatt már az $(F-2)$-edik időegységben végrehajtódna) — legyen Q egy ilyen taszk.

   Tekintsük most azt a τ' taszkrendszert, amelyet $\tau_{\min}$-ből kapunk T és annak közvetlen megelőzői eltávolításával. Ha az $S_{\mathrm{LEV}}(\tau_{\min})$-nek megfelelő Gantt-diagramban minden eltávolított taszkot üres periódussal helyettesítünk, akkor az S' ütemezéshez jutunk, amely τ' szintes ütemezésének felel meg. Ugyanis:

   i.) T az utolsó, F-edik időegységben hajtódik végre, így — mivel minden más feladatnak rákövetkezője — T üres periódussal való helyettesítése nem változtatja meg a megmaradó ütemezés szintes szerkezetét.

   ii.) Legyen most P egy olyan közvetlen megelőzője T-nek, amelyik a k-adik időegységben $(1 \leq k \leq F-1)$ hajtódik végre. Ha P-t eltávolítjuk, akkor a k-adik periódusban S' készítésekor csak olyan taszk ütemezhető, amelynek szintje azonos P szintjével, vagy annál kisebb: ugyanis, ha S' szerint egy P-nél magasabb szintű taszk P után hajtódik végre, akkor a taszkot S_{LEV} összeállításakor P előtt kellett vizsgálnunk, és annak kellett fennállnia, hogy az a taszk nem hajtható végre. Mivel a P-vel azonos vagy nála alacsonyabb szinten lévő taszkokat eltávolítottuk, így a P eltávolításával felszabaduló üres periódusban egyetlen taszk sem hajtódhat végre.

Mivel a taszkok szintje τ'-ben pontosan 2-vel kisebb, mint volt $\tau_{\min}$-ben, így S' valóban szintes ütemezés.

e) Mivel $\tau_{\min}$-ből két szintet elhagytunk, így $F_{\max}(S'(\tau')) = F' \leq F - 2$. Az előbb említett Q taszk az $(F-2)$-edik időegységben hajtódik végre. Mivel Q a T taszk egyik közvetlen megelőzőjének közvetlen megelőzője, amelyik az $(F-2)$-edik időegységben hajtódik végre, így Q szintje 3. Ezért az $(F-2)$-edik időegység előtt nem lehet olyan időegység, amelyben csak kettes szintű taszkok hajtódtak végre. Így F' értéke pontosan $F-2$.

f) τ'-höz vegyünk hozzá egy új taszkot, amelynek közvetlen megelőzői pontosan τ' befejező taszkjai. Az így kapott τ'' taszkrendszer struktúrája ugyyancsak fa. Ha az S' ütemezést úgy egészítjük ki S'' ütemezéssé, hogy az $(F-1)$-edik időegységben a τ'' befejező feladatát hajtjuk végre, akkor azt kapjuk, hogy $F_{\max}(S''(\tau'')) = F'' = (F-2)+1 = F-1$.

g) Mivel τ'-t $\tau_{\min}$-ből két szinteltávolításával kaptuk, így τ'-ben legfeljebb $n-2$, és így τ''-ben legfeljebb $n-1$ taszk van, azaz τ'' kisebb, mint $\tau_{\min}$. Ha $\tau_{\min}$-t optimálisan ütemezzük, akkor befejezőú taszkja az F_0-adik időegységben hajtódik végre. Mivel az (F_0-1)-edik időegységben csak a befejező taszk közvetlen megelőzői hajtódhatnak végre S_{OPT} szerint, így $F_{\max}(S_{\mathrm{OPT}}(\tau')) = F_0' \leq F_0 - 2$. Amikor τ'-t egy új befejező taszkkal egészítjük ki, akkor a maximális befejezési idő legfeljebb egy egységgel nő, azaz $F_{\max}(S_{\mathrm{OPT}}(\tau'')) = F_0'' \leq F_0 - 1$.
Mivel $F_0 < F$, $F_0'' \leq F_0 - 1$ és $F'' = F-1$, innen $F_0'' < F''$, azaz S'' nem optimális. Mivel τ'' kisebb, mint $\tau_{\min}$, ellentmondásba kerültünk $\tau_{\min}$ definíciójával, ami a bizonyítandó tétel helyességét igazolja.

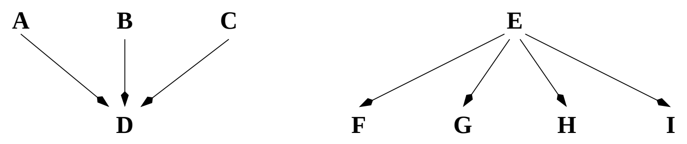
3.1. következmény. Ha egységnyi végrehajtási idejű, megszakíthatatlan taszkokat tartalmazó, erdő struktúrájú taszkrendszert azonos processzorokra a LEV algoritmussal ütemezünk, akkor a maximális befejezési idő minimális lesz.

Bizonyítás. A végrehajtandó taszkrendszert kiegészítjük egy új, egységnyi idejű taszkkal, amelynek a közvetlen megelőzői pontosan az eredeti taszkrendszer befejező taszkjai. Az így kapott τ' taszkrendszer megfelel a 3.1. tétel feltételeinek. Ha LEV nem minimalizálja $F_{\max}$-ot τ-ra, akkor nem minimalizálhatja az új taszkrendszerre sem — ezért a 3.1. tételt alkalmazva τ'-re, megkapjuk a következmény állítását.

3.2. következmény. Ha egységnyi végrehajtási idejű, megszakíthatatlan taszkokat tartalmazó, antierdő struktúrájú taszkrendszert azonos processzorokra a LEV algoritmussal ütemezünk (az alább ismertett módon), akkor a maximális befejezési idő minimális lesz.

Bizonyítás. Fordítsuk meg az entierdőben az élek irányítását. Az így kapott erdőstruktúrájú feladatrendszerre alkalmazzuk a LEV algoritmust. Könnyen belátható, hogy az így kapott $S_{\text{LEV}}(\tau')$ ütemezést visszafelé olvasva az eredeti taszkrendszer szintes ütemezéséhez jutunk. Ha az antierdő struktúrájú eredeti taszkrendszernek lenne a visszafelé olvasással kapottnál jobb ütemezése, akkor annak megfordításával az erdő struktúrájú τ'-re $S_{\text{LEV}}(\tau')$-nél jobb ütemezést kapnánk, ami ellentmondana a 3.1. következménynek.

Ha egy taszkrendszer gráfjában fa és antifa is van (az ilyen irányított gráfot nevezhetjük dzsungelnek), akkor egy ütemezés szintes volta már nem garantálja az optimális megoldást. Ezt illusztrálja a 3.5. ábrán lévő dzsungel struktúrájú taszkrendszer. Ekkor egy szintes ütemezés eredményezheti az (A, B, C, E, D, F, G, H, I) listát, amelyre három processzor esetén $F_{\max} = 4$, míg az (E, A, B, C, D, F, G, H, I) lista esetén az optimális $F_{\max} = 3$ értéket kapjuk.



3.5. ábra. Dzsungel struktúrájú taszkrendszer

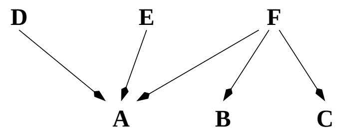
3.3. Taszkrendszerek végrehajtása két processzoron

Ebben a pontban a következő ütemezési problémát vizsgáljuk: $\pi = (\rho, \tau, \mu)$, ahol $\rho = (\mathbf{P})$, $\mathbf{P} = \{P_1, P_2\}$, $\tau = (\mathbf{T}, <, \mathbf{t})$, $\mathbf{T} = \{\mathrm{T}_1, \ldots, \mathrm{T}_n\}$, $\mathbf{t} = (1, \ldots, 1)$, $\mu = F_{\max}\downarrow$.

A 3.6. ábrán bemutatott egyszerű taszkrendszer S' és S'' ütemezése is megfelel a szintes elvnek. Ez a példa bizonyítja, hogy tetszőleges struktúra esetén a szintesség még két processzorra sem garantálja az optimális ütemezést.

Tekintsük most azt a taszkrendszert, melyben $\mathbf{T}$={A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q}, és $<$={(D,B), (D,C), (E,C), (F,A), (F,C), (G,A), (G,D), (G,E), (H,D), (H,E), (H,F), (I,D), (I,F), (J,F), (K,G), (L,G), (L,H), (M,K), (M,L), (N,I), (N,M), (O,N), (P,N), (Q,J), (Q,N)} (3.7. ábra).

Vizsgáljuk meg, hogyan lehetne a szintes algoritmust úgy általánosítani, hogy biztosítsa az optimális ütemezést tetszőleges struktúra esetén. Ehhez megfogalma-



S' :

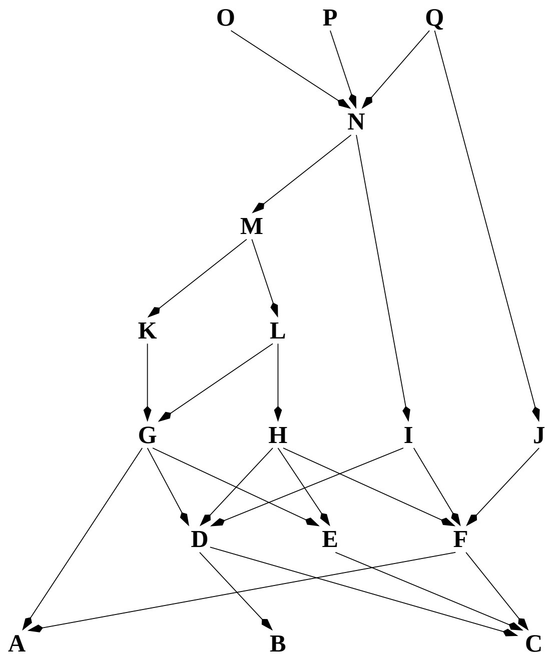
P_1	D	F	A	C
P_2	E	-	B	-

S'' :

P_1	E	D	A
P_2	F	C	B

3.6. ábra. Taszkrendszer, és két lehetséges szintes ütemezése

zunk bizonyos célszerűnek látszó tulajdonságokat. A T taszk címkéjét (indexét) most $C(\mathrm{T})$, szintjét pedig $L(\mathrm{T})$ jelöli.



3.7. ábra. Taszkrendszer a LEV algoritmus általánosításához

3.1. tulajdonság. Ha a Q taszk szintje nagyobb, mint az R taszké, akkor az indexe is legyen nagyobb, azaz teljesüljön $\forall \mathrm{Q,R} \in \mathbf{T}: \ L(\mathrm{Q}) > L(\mathrm{R}) \rightarrow C(\mathrm{Q}) > C(\mathrm{R})$.

Ha például minden szinten páros számú taszk van, akkor ez az elv (tulajdonság) önmagában is elegendő az optimális ütemezés megtalálásához.

3.2. tulajdonság. Ha a Q taszk közvetlen rákövetkezőinek halmaza valódi részhalmazként tartalmazza az R taszk közvetlen rákövetkezőinek halmazát, akkor $C(\mathrm{Q}) > C(\mathrm{R})$.

Ezek a tulajdonságok elegendőek arra, hogy a 17-es indexet Q-hoz rendeljük, a 15-öst és 16-ost pedig O-hoz és P-hez. A hatodik szinten csak az N taszk van, ezért az kapja a 14-es indexet. Ezután az M taszk a 13-as indexet kapja, az L a 12-est, a K pedig a 11-est.

Hogyan hasonlítsuk össze a harmadik szinten lévő taszkokat, mindenekelőtt a G-t és H-t? G-nek A, H-nak pedig F a harmadik rákövetkezője a közös D és E mellett, így a 3.2. tulajdonság nem elég a választáshoz. Az egyik lehetőség az, hogy H-hoz rendeljük a nagyobb indexet, mivel több második szinten lévő rákövetkezője van, és a 3.1. tulajdonságot teljesítő ütemezésben az első szint taszkjai csak akkor kerülhetnek ütemezésre, ha a második szint minden taszkját ütemeztük már. Így tehát $C(\mathrm{H})$:=10, $C(\mathrm{G})$:=9, $C(\mathrm{I})$=8 és $C(\mathrm{J})$=7. D és F közül megint nem tudunk a követő halmazok alapján dönteni, legyen mondjuk $C(\mathrm{D})$=6 és $C(\mathrm{F})$=5, végül $C(\mathrm{E})$:=4, $C(\mathrm{A})$:=3, $C(\mathrm{B})$:=2 és $C(\mathrm{C})$:=1. Jelölje $S_{\mathrm{GLEV}}(\tau)$ az ezen lista alapján adódó ütemezést, és nevezzük GLEV algoritmusnak az eljárást, amivel a listát (és az ütemezést) előállítottuk — vagyis a fenti két tulajdonság figyelembevételével működő ütemezési algoritmust. Megmutatjuk, hogy bármely $S(\tau)$ ütemezés esetén $F_{\max}(S(\tau)) \geq F_{\max}(S_{\mathrm{GLEV}}(\tau))$, vagyis hogy GLEV optimális.

Ehhez először definiáljuk a LEV algoritmus általánosított változatát, a GLEV algoritmust.

3.2. algoritmus (GLEV). Legyen $\tau = (\mathbf{T}, <, \mathbf{t})$ tetszőleges struktúrájú taszkrendszer, amely egységnyi végrehajtási idejű, megszakíthatatlan taszkokat tartalmaz. Ekkor τ $S_{\mathrm{GLEV}}(\tau)$ szintes ütemezése azonos processzorokon a következőképpen állítható elő.

Definiálunk egy $C : \mathbf{T} \to \{1, \ldots, n\}$ „címkéző” leképezést (1–3. lépés), majd a címkék segítségével listásan ütemezünk (4–5. lépés).

1. lépés. Legyen $\mathrm{T} \in \mathbf{T}$ a taszkrendszer tetszőleges befejező taszkja, és legyen $C(\mathrm{T}) := 1$.

2. lépés. Tegyük fel, hogy az $1, \ldots, j-1$ $(j \leq n)$ címkéket már hozzárendeltük bizonyos taszkokhoz. Legyen γ_j azon taszkok halmaza, amelyeknek

a) még nincs címkéjük, és

b) nincs címkézetlen közvetlen rákövetkezőjük.

3. lépés. Most kiválasztjuk γ_j azon elemét, amelyik a j címkét fogja kapni.

Ehhez minden $\mathrm{T} \in \gamma_j$ taszkhoz hozzárendeljük a közvetlen rákövetkezőinek indexeit csökkenő sorrendben tartalmazó listát, melyet $\overline{\mathrm{T}}$-vel jelölünk.

Legyen most $\mathrm{T} \in \gamma_j$ olyan taszk, amelyre $\overline{\mathrm{T}} \leq \overline{\mathrm{Q}}$ teljesül minden $\mathrm{Q} \in \gamma_j$ taszkra, ahol a $\leq$ lexikografikus összehasonlítást jelent. Ekkor legyen $C(\mathrm{T}) := j$.

4. lépés. Ha van még címkézetlen taszk, akkor folytassuk a 2. lépéstől, egyébként pedig a 4. lépéstől.

5. lépés. Indexeljük meg a taszkokat oly módon, hogy a $(\mathrm{T}_n, \ldots, \mathrm{T}_1)$ listára fennálljon $C(\mathrm{T}_j) = j$ minden $j = 1, \ldots, n$ címkére.

6. lépés. A $(\mathrm{T}_n, \ldots, \mathrm{T}_1)$ lista segítségével m azonos processzoron adódó ütemezést nevezzük S_{GLEV}-nek.

Mivel a taszkok egységnyi végrehajtási idejűek, ezért végrehajtásuk olyan intervallumokban történik, amelyek végpontjai egész számok. Jelöljük $\lambda(\mathrm{T})$-vel azt az időegységet, amelyben T végrehajtódik.

Mint korábban is, minden időegységben először a P_1 processzorhoz rendelünk taszkot. Ekkor igaz a következő állítás.

3.1. lemma. Ha a GLEV algoritmussal kapott ütemezés szerint T a P_1 processzoron hajtódik végre, akkor **T** minden T-től különböző, T-nél nem korábban végrehajtott elemének indexe kisebb, mint $C(\mathrm{T})$, azaz $\forall \mathrm{R} \in \mathbf{T}: \ \mathrm{R} \neq \mathrm{T} \wedge \lambda(\mathrm{R}) \geq \lambda(\mathrm{T}) \rightarrow C(\mathrm{R}) < C(\mathrm{T})$.

Bizonyítás. Amikor a T taszkot GLEV szerint a P_1 processzorhoz rendeltük, akkor a még ütemezetlen taszkok vagy ütemezhetőek voltak, vagy nem. Mivel T a P_1 processzoron a $\lambda(\mathrm{T})$-edik időegységben hajtódik végre, a többi ütemezhető taszk indexe kisebb volt, mint $C(\mathrm{T})$. A nem ütemezhető taszkok csak valamelyik – az adott időpontban ütemezhető – megelőzőjük befejezése után válnak végrehajthatóvá. Mivel GLEV szerint a taszkok csak akkor kaphatnak címkét, amikor már minden megelőzőjük kapott címkét (és a címkék csökkenő sorrendben kerülnek kiosztásra), így minden taszk címkéje kisebb, mint a megelőzőinek címkéje. Tehát a nem ütemezhető taszkok is rendelkeznek a fenti tulajdonsággal.

A továbbiakban fontos szerepet játszanak a *domináló taszkok*, az *ugró taszkok*, és a segítségükkel definiált *követő halmazok*, amelyeket az alábbi (FOL) algoritmus segítségével adunk meg.

3.3. algoritmus (FOL). Legyen $S_{\mathrm{GLEV}}(\tau)$ a τ taszkrendszer GLEV által előállított ütemezése. Ekkor az adott ütemezéshez tartozó k természetes számot, a $\mathrm{D}_k, \ldots, \mathrm{D}_1$, D_0 domináló taszkokat, a $\mathrm{J}_k, \ldots, \mathrm{J}_1$, J_0 ugró taszkokat, valamint az F_0, $F_1, \ldots, F_k$ követő halmazokat meghatározó FOL algoritmus a következő lépésekből áll.

1. lépés. Legyenek Q, illetve R a P_1, illetve P_2 processzoron az utolsó időegységben végrehajtott taszkok (R üres periódus is lehet). Ekkor $\mathrm{D}_0 := \mathrm{Q}$ és $\mathrm{J}_0 := R$.

2. lépés. Tegyük fel, hogy $\mathrm{D}_{j-1}, \ldots, \mathrm{D}_0$ és $\mathrm{J}_{j-1}, \ldots, \mathrm{J}_0$ már ismert, D_j és J_j még nem. Ekkor J_j (ha van ilyen) az a T taszk vagy üres periódus, amelyre

a) $C(\mathrm{T}) < C(\mathrm{D_{j-1}})$, ahol feltesszük, hogy az üres periódus indexe nulla;

b) $\lambda(\mathrm{T}) < \lambda(\mathrm{D}_{j-1})$;

c) az előző két feltételnek eleget tevő taszkok közül $\lambda(\mathrm{T})$ a lehető legnagyobb.

3. lépés. Ha J_j létezik, akkor D_j legyen az a taszk, amely a $\lambda(\mathrm{J}_j)$-edik intervallumban a P_1 processzoron hajtódik végre, és menjünk a 2. lépésre.

4. lépés. Legyen $k = j - 1$. Ekkor $F_k := \{\, T \mid \lambda(\mathrm{T}) < \lambda(\mathrm{D}_k)\} \cup \{\mathrm{D}_k\}$, $F_i := \{\, T \mid \lambda(\mathrm{D}_{i+1}) < \lambda(\mathrm{T}) < \lambda(\mathrm{D}_i)\} \cup \{\mathrm{D}_i\}$, $(i = 0, 1, \ldots, k-1)$.

A következő, 3.8. ábra mutatja a definícióban szereplő taszkok közötti kapcsolatot. A J_i taszk megelőzi a D_{i-1}-et, „elébe ugrik” abban az értelemben, hogy korábban hajtódik végre annak ellenére, hogy az indexe kisebb. A következő lemmában megmutatjuk, hogy a D_i taszk dominál az F_{i-1}-beli taszkok felett abban az értelemben, hogy megelőzi őket.

	F_i		F_{i-1}	
D_{i+1}	...	D_i	...	D_{i-1}
J_{i+1}	...	J_i	...	J_{i-1}

3.8. ábra. Az ugró és domináló taszkok, és a követő halmazok

3.2. lemma. A D_i $(i = 1, \ldots, k)$ taszk megelőzi az F_{i-1} követő halmaz minden elemét.

Bizonyítás. A D_i és a J_i taszkok egyidejűleg hajtódnak végre. J_i-t úgy választottuk meg, hogy $C(\mathrm{J_i}) < C(\mathrm{D_{i-1}})$ teljesüljön, és J_i a lehető legközelebbi legyen (balról, azaz a korábban végrehajtott taszkok közül) D_{i-1}-hez. Ezért minden F_{i-1}-beli T taszkra teljesül, hogy $C(\mathrm{D_{i-1}}) < C(\mathrm{T})$, és az egyenlőtlenség tranzitivitása miatt fennáll $C(\mathrm{J_i}) < C(\mathrm{T})$. Mivel az F_{i-1}-beli taszkok indexe nagyobb, mint $C(\mathrm{J_i})$, ezért az ütemezés során előbb vizsgáltuk őket, mint J_i-t. Mivel helyettük J_i-t ütemeztük, így akkor ők nem voltak ütemezhetőek. Ebből következik, hogy D_i minden ilyen T taszknak megelőzője (nem feltétlenül közvetlen megelőzője).

Most megmutatjuk, hogy nemcsak D_i, hanem F_i minden eleme dominál F_{i-1} elemei felett.

3.3. lemma. Az F_i halmaz ($i = 1, \ldots, k$) bármely eleme megelőzi az F_{i-1} halmaz bármely elemét.

Bizonyítás. Tegyük fel, hogy az állítás nem igaz. Akkor van olyan i index ($1 \le i \le k$), hogy egy F_i-beli Q taszk nem előz meg egy F_{i-1}-beli R taszkot. Ekkor Q-nak vagy nincs rákövetkezője F_i-ben (a eset), vagy van (b eset):

a) Ekkor a 3.2. lemma szerint Q különbözik D_i-től, és $\mathrm{D}_i <$ R. F_i definíciója szerint $C(\mathrm{Q}) > C(()\mathrm{D}_i)$, mivel egyébként Q szóba jött volna J_{i+1}-ként, és nem lehetne F_i eleme.
Legyen $\overline{\mathrm{D}_i} = (n_1, \ldots, n_r)$ (a D_i rákövetkezőinek indexeit tartalmazó lista), és δ_{i-1} legyen azon F_{i-1}-beli taszkok halmaza, amelyeknek nincs közvetlen megelőzőjük F_{i-1}-ben. Ekkor δ_{i-1} elemeinek indexei jelen vannak $\overline{\mathrm{D}_i}$-ban, hiszen ezek az elemek D_i közvetlen rákövetkezői. F_{i-1} azon elemeinek indexei, amelyek nem tartoznak δ_{i-1}-hez, nincsenek jelen $\overline{\mathrm{D}_i}$-ban, mivel a megelőzési gráfban nincsenek redundáns élek. F_{i-1} elemeinek indexei nem kisebbek $C(()\mathrm{D}_{i-1})$-nél, viszont $F_{i-2}, \ldots, F_0$ elemeinek indexei és a $\mathrm{J}_{i-1}, \ldots, \mathrm{J}_0$ taszkok indexei a 3.1. lemma szerint kisebbek $C(()\mathrm{D}_{i-1})$-nél, ezért δ_{i-1} elemeinek indexei az első $|\delta_{i-1}|$ helyet foglalják el $\overline{\mathrm{D}_i}$-ban. Mivel $C(\mathrm{Q}) > C(()\mathrm{D}_i)$, ezért $\overline{\mathrm{D}_i} \le \overline{\mathrm{Q}}$, ezért nem üres $\overline{\mathrm{D}_i}$ esetén $\overline{\mathrm{Q}}$ sem lehet üres lista. Mindez csak úgy lehetséges, hogy Q közvetlen megelőzője δ_{i-1} valamennyi elemének; ekkor viszont Q megelőzője F_{i-1} valamennyi elemének, ami ellentmond R választásának.

b) Ha Q-nak van rákövetkezője F_i-ben, akkor van olyan Z rákövetkezője is, amelynek már nincs rákövetkezője F_i-ben. Ekkor a már bizonyított a) rész szerint Z megelőzi F_{i-1} bármely elemét, és így a megelőzés tranzitivitása miatt Q is megelőzi F_{i-1} minden elemét, azaz R-et is. Ez az ellentmondás teljessé teszi a lemma bizonyítását.

A 3.3. lemma segítségével bizonyítható GLEV optimalitása.

3.2. tétel. Ha egységnyi végrehajtási idejű, megszakíthatatlan taszkokat tartalmazó taszkrendszert két azonos processzorra a GLEV algoritmussal ütemezünk, akkor a maximális befejezési idő minimális lesz.

Bizonyítás. Legyen $S_{\mathrm{GLEV}}(\tau)$ a GLEV algoritmussal kapott ütemezés, és $F_0, \ldots, F_k$ az ütemezéshez tartozó követőhalmazok. Ekkor a 3.3. lemma szerint F_i bármely eleme megelőzi F_{i-1} bármely elemét ($i = 1, \ldots, k$), és így az F_{i-1}-beli taszkok végrehajtása csak az F_i-beli taszkok végrehajtásának befejezése után kezdődhet el. A követő halmazok definíciója szerint ezen halmazok elemszáma páratlan. Legyen

F_i elemszáma $2m_i - 1$. Ekkor F_i elemeinek két processzoron való végrehajtásához legalább m_i időegységre van szüksége bármely algorimusnak, azaz $F_{\max}(S_{\mathrm{A}}(\tau)) \geq \sum_{i=1}^{k} m_i$ bármely A algoritmus esetén. Mivel $F_{\max}(S_{\mathrm{GLEV}}(\tau)) = \sum_{i=1}^{k} m_i$, így GLEV valóban minimalizálja $F_{\max}$értékét.

A tétellel kapcsolatban vizsgáljuk meg azt a taszkrendszert, amelyben $\mathbf{T} = \{$A, B, C, D, E, F, G, H$\}$, és $< = \{$(E,D), (F,A), (F,B), (F,C), (F,E) (G,D), (G,E), (H,D), (H,E)$\}$. GLEV szerint az indexek például ábécésorrendben rendelhetők a taszkokhoz, és akkor a 3.9. ábra bal oldalán szereplő Gantt-diagram szerint $F_{\max}(S_{\mathrm{GLEV}}(\tau)) = 5$, szemben a jobb oldalon bemutatott $F_{\max}(S_{\mathrm{OPT}}(\tau)) = 4$ értékkel.

S_{GLEV} :

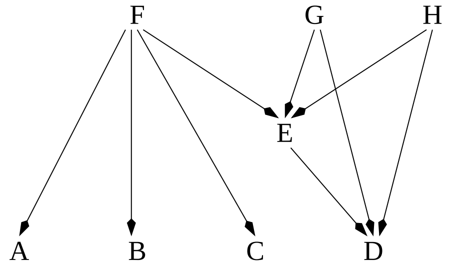
P_1	H	F	E	D	B
P_2	G	-	-	C	A

S_{OPT} :

P_1	F	G	E	D
P_2	H	C	A	B

3.9. ábra. A GLEV algoritmussal kapott ütemezés, és egy optimális ütemezés

A bal oldali ütemezést úgy kaptuk, hogy a taszkrendszer gráfját a 3.10. ábrán láthatónak tételeztük fel. A példa nem mond ellent a 3.2. tételnek, ugyanis a 3.10.



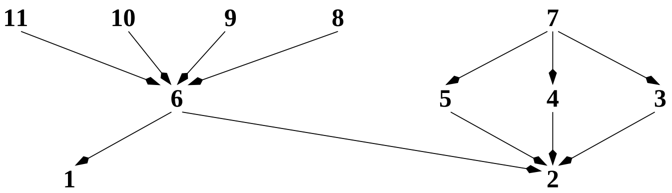
3.10. ábra. A taszkrendszer redundáns éleket tartalmazó gráfja

ábrán lévő gráf a redundáns (H,D) és (G,D) éleket is tartamazza, ami miatt az indexelés hibás lett. A két élt elhagyva GLEV optimális megoldást ad.

A 3.11. ábrán bemutatott taszkrendszer azt mutatja, hogy például három processzor esetén GLEV nem garantálja $F_{\max}$optimális értékét: az ábra szerint az általános szintes algoritmus alkalmazása esetén 5, míg optimális esetben csak 4 időegység kell a taszkrendszer végrehajtásához.

3.4. Független feladatok

Ebben a pontban a következő ütemezési problémát vizsgáljuk: $\pi = (\rho, \tau, \mu)$, ahol $\rho = (\mathbf{P})$, $\mathbf{P} = \{P_1, \ldots, P_m\}$, $\tau = (\mathbf{T}, \mathbf{t})$, $\mathbf{T} = \{\mathrm{T}_1, \ldots, \mathrm{T}_n\}$, $\mathbf{t} = (t_1, \ldots, t_n)$, a processzorok azonosak, a taszkok megszakíthatóak, $\mu = F_{\max} \downarrow$.



S_{GLEV} :

P_1	11	8	6	3	2
P_2	10	7	5	1	-
P_3	9	-	4	-	-

S_{OPT} :

P_1	7	5	4	2
P_2	11	9	3	1
P_3	10	8	6	-

3.11. ábra.
Taszkrendszer 3-processzoros ütemezéshez; a GLEV algoritmussal kapott ütemezés, és egy optimális ütemezés

Legyen a végrehajtási idők összege X. Mivel a legkedvezőbb eset az állásidő nélküli ütemezés lenne, így tetszőleges A ütemezési algoritmus esetén fennáll $F_{\max}(S_{\mathrm{A}}) \geq \frac{X}{m}$.

Legyen $t_{\max} = \max_{1 \leq i \leq n} t_i$. Mivel egy feladat végrehajtásán egyidejűleg legfeljebb egy processzor dolgozhat, ezért tetszőleges A ütemezési algoritmus esetén fennáll $F_{\max}(S_{\mathrm{A}}) \geq t_{\max}$.

3.4. algoritmus (CONT). A fenti jelöléseket használja a „folytonos” algoritmus:

1. lépés. Legyen $F = \max(t_{\max}, \frac{X}{m})$, és ütemezzük a T_1 taszkot a P_1 processzorra a $[0, t_1]$ intervallumban.

2. lépés. Ha minden taszkot ütemeztünk, akkor STOP.

3. lépés. Tegyük fel, hogy eddig a $\mathrm{T}_1, \ldots, \mathrm{T}_{i-1}$ taszkokat ütemeztük, és hogy a $P_1, \ldots, P_{j-1}$ processzorok teljesen (a $[0, F]$ intervallumban) foglaltak, a P_j processzor pedig csak részben foglalt (a $[0, f]$ intervallumban, ahol $0 \leq f < F$).

4. lépés. Ha $t_i \leq F - f$, akkor ütemezzük T_i-t P_j-re, az $[f, f + t_i]$ intervallumban, egyébként pedig ütemezzük T_i $F - f$ végrehajtási idejű T'' részét P_j-re az $[f, F]$ intervallumban, és $t_i - (F - f)$ végrehajtási idejű T' részét P_{j+1}-re a $[0, t_i - F + f]$ intervallumban.

3.3. tétel. Ha független, megszakítható taszkokat azonos processzorokra CONT segítségével ütemezünk, akkor $F_{\max}$ minimális lesz, értéke pedig $F_{\max}(S_{\mathrm{CONT}}) = \max(t_{\max}, \frac{X}{m})$.

Bizonyítás. a) Mivel tetszőleges A ütemezési algoritmus esetén $F_{\max}(S_{\mathrm{A}}) \geq t_{\max}$ és $F_{\max}(S_{\mathrm{A}}) \geq \frac{X}{m}$, így $F_{\max}(S_{\mathrm{A}}) \geq F = \max(t_{\max}, \frac{X}{m})$.

b) CONT ezt az okvetlenül szükséges értéket biztosítja, mivel egyrészt F elég nagy ahhoz, hogy CONT szerint egy taszkon egyszerre legfeljebb egy processzor dolgozzon (T′ és T″ sávjai ne fedjék át egymást), másrészt F elég nagy ahhoz, hogy T_n is időben befejeződjék (T_n sávja ne „lógjon ki”).

Mit mondhatunk a megszakítások számáról, ha ragaszkodunk $F_{\max}$ minimális értékéhez? *Megszakításról* akkor beszélünk, ha egy T taszkot egy P processzoron elkezdünk végrehajtani, és a végrehajtást adott t időpontban annak ellenére felfüggesztjük, hogy a taszkot még nem fejeztük be. Egy T taszk egy adott t időpontban *aktív*, ha van olyan P processzor, amelyik a t időpontban éppen a T taszk végrehajtását végzi.

Tekintsük például azt a taszkrendszert, amelynek futási időit az $(1, 2, 3, 4, 5, 6)$ vektor tartalmazza, és $m = 3$ processzorunk van? A CONT algoritmus $F_{\max}(S_{\mathrm{CONT}}) = \max(t_{\max}, \frac{X}{m}) = \max(6, 21/3 = 7)$ időegység alatt hajtja végre a hat taszkot, és közben két megszakítást okoz. Ugyanakkor az 1 és 6, a 2 és 5, valamint a 3 és 4 futási idejű taszkokat párosítva a 6 taszk megszakítás nélkül is végrehajtható.

Általában azt mondhatjuk, hogy a CONT egyrészt minden taszkot legfeljebb egyszer szakít meg, másrészt minden processzoron legfeljebb egy megszakítást okoz. Sőt az is igaz, hogy CONT legfeljebb $m - 1$ megszakítást okoz, mivel az első processzoron soha nem okoz megszakítást, továbbá legfeljebb $n - 2$ megszakítást okoz, mivel az első processzoron elkezdett és az utolsó processzoron befejezett taszkot biztosan nem szakítja meg.

Érdekes ezzel kapcsolatban a következő tétel, amely azt mutatja, hogy $m - 1$ a lehető legjobb korlát.

3.4. tétel. Legyen S_{CONT} egy n független taszkból álló taszkrendszer CONT által előállított ütemezése m azonos processzoron. Ekkor

- S_{CONT} legfeljebb $m - 1$ megszakítást tartalmaz;
- $m-1$ a lehető legjobb korlát, mert minden $m \geq 1$ processzorszámhoz van olyan taszkrendszer, amelyet minden olyan algoritmus legalább $m-1$ megszakítással hajt végre, amwelyik biztosítja, hogy $F_{\max}$ minimális legyen.

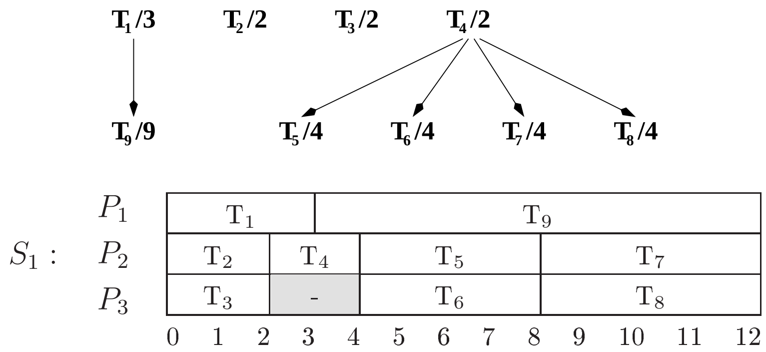
Bizonyítás. CONT szerint minden taszk legfeljebb két részben hajtódik végre. Ha két részben, akkor legfeljebb két aktív periódusban és két szomszédos processzoron úgy, hogy a megszakítás a nagyobb indexű processzoron lesz, ezért CONT ütemezésében az első processzoron biztosan nincs megszakítás, a többi processzoron pedig legfeljebb egy van. Ezzel a tétel első állítását beláttuk.

Tekintsünk egy olyan taszkrendszert, amelyben a taszkok száma eggyel nagyobb, mint a processzorok száma, azaz legyen $n = m + 1$, a taszkok végrehajtási ideje pedig legyen m. Tegyük fel, hogy egy A algoritmus ütemezésében legfeljebb $m - 2$ megszakítás van. Akkor lesz legalább két olyan processzor (legyenek ezek például P_1 és P_2), amelyeken egyáltalán nincs megszakítás. MIvel minden taszk hossza egységesen m, ezért ezen a két processzoron a $[0, m]$ intervallumban folyik egy-egy taszk (legyenek ezek a T_1 és T_2 taszkok) végrehajtása. CONT ütemezése alapján tudjuk, hogy az optimális ütemezésben nincs üres periódus, ezért ezek a processzorok az $[m, m + 1]$ intervallumban is dolgoznak. Tegyük fel, hogy a $[t, t']$ intervallumban (ahol $m < t < t' < m + 1$) egyik a T_3, másik a T_4 taszkot hajtja végre. Ekkor a többi $m-2$ processzornak csak $m+1-4 = m-3$ taszk marad, ami kevés. Így lesz a Gantt-diagramban üres periódus, az ütemezés nem lesz optimális, ami ellentmondás.

3.5. Ütemezési anomáliák I.

Ebben a pontban azt mutatjuk be, hogy egy ütemezési probléma „könnyítése” is kiválthatja a hatékonysági jellemzők romlását.

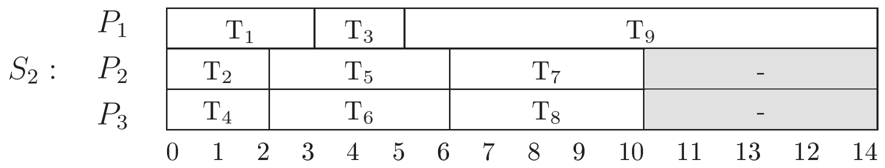
3.1. példa. Tekintsük az alábbi τ_1 taszkrendszert, és annak az $L = (\mathrm{T}_1, \ldots, \mathrm{T}_9)$ listával kapott S_1 ütemezését három ($m = 3$) azonos processzoron. Ekkor (3.12. ábra) $F_{\max}(S_1) = 12$, amiről könnyen belátható, hogy optimális érték.



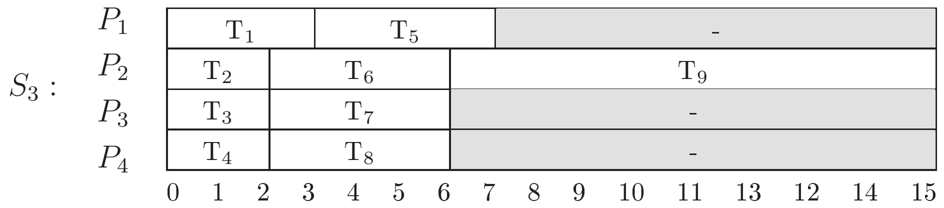
3.12. ábra. A τ_1 taszkrendszer, és annak optimális ütemezése

3.2. példa. Ütemezzük az előbbi τ_1 taszkrendszert $m = 3$ azonos processzorra az $L' = (\mathrm{T}_1, \mathrm{T}_2, \mathrm{T}_4, \mathrm{T}_5, \mathrm{T}_6, \mathrm{T}_3, \mathrm{T}_9, \mathrm{T}_7, \mathrm{T}_8)$ listával. Ekkor a kapott S_2 ütemezésre (3.13. ábra) $F_{\max}(S_2) = 14$.

3.3. példa. Ütemezzük a τ_1 taszkrendszert az L listával $m' = 4$ processzorra. Az eredmény $F_{\max}(S_3) = 15$ (ld. 3.14. ábra).

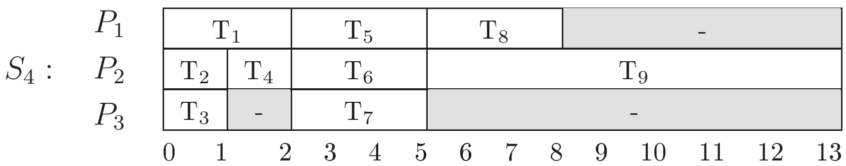


3.13. ábra. A τ_1 taszkrendszer ütemezése az L' lista esetén



3.14. ábra. τ_1 ütemezése az L listával $m' = 4$ processzorra

3.4. példa. Csökkentsük τ_1-ben a végrehajtási időket eggyel. Ütemezzük az így kapott τ_2 taszkrendszert az L listával $m = 3$ processzorra. Az eredmény: $F_{\max}(S_4) = 13$ (ld. 3.15. ábra).



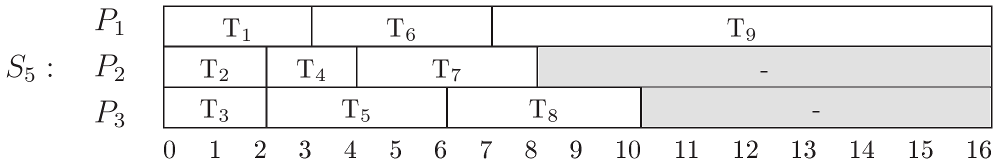
3.15. ábra. τ_2 ütemezése az L listával $m = 3$ processzorra

3.5. példa. Enyhítsük a τ_1 taszkrendszerben a precedencia-korlátozásokat: hagyjuk el a $(\mathrm{T}_4, \mathrm{T}_5)$ és a $(\mathrm{T}_4, \mathrm{T}_6)$ éleket a gráfból. Az így kapott τ_3 taszkrendszer S_5 ütemezésének eredménye a 3.16. ábrán látható: $F_{\max}(S_5) = 16$.

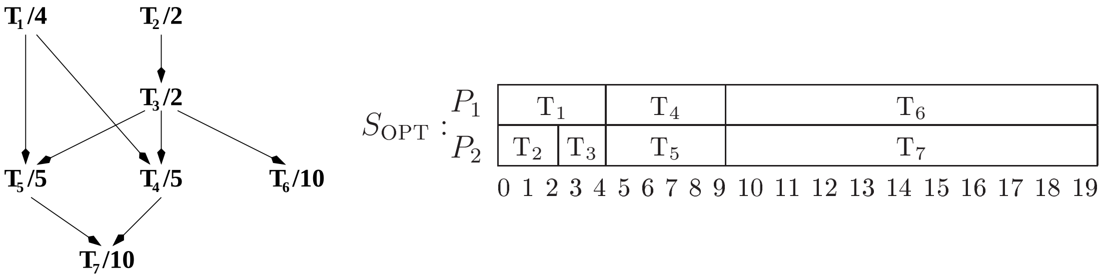
A következő példa azt mutatja, hogy $F_{\max}$ növekedése nem csak a lista rossz megválasztása miatt következhet be.

3.6. példa. Legyen a τ taszkrendszer és annak S_{OPT} optimális ütemezése a 3.17. ábra szerinti. Ekkor $F_{\max}(S_{\mathrm{OPT}}) = 19$.

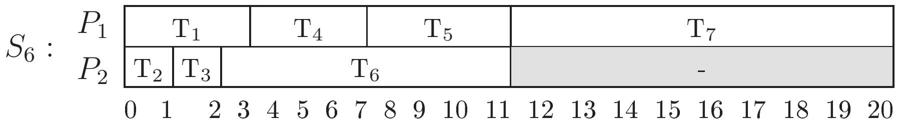
Könnyen belátható, hogy ha most a végrehajtási időket 1-gyel csökkentjük, akkor a kapott τ' taszkrendszeren $F_{\max}(S_6) = 20$ értéknél egyetlen listával sem érhetünk el jobbat (3.21. ábra).



3.16. ábra. A τ_3 taszkrendszer ütemezése $m = 3$ processzorra



3.17. ábra. A τ taszkrendszer, és annak S_{OPT} ütemezése két processzoron



3.18. ábra. A τ' taszkrendszer optimális listás ütemezése

Ezen példák után az ütemezési paraméterek hatását jellemző relatív korlátot adunk meg. Tegyük fel, hogy adott τ és τ' taszkrendszerekre $\mathbf{T}' = \mathbf{T}$, $<' \subseteq <$, $\mathbf{t}' \leq \mathbf{t}$. A τ taszkrendszert egy L, a τ' taszkrendszert pedig egy L' lista alkalmazásával ütemezzük — mégpedig előbbit m, utóbbit pedig m' azonos processzorra. Az így kapott S, illetve S' ütemezésekre legyen $F_{\max}(S) = F$ és $F_{\max}(S') = F'$.

3.5. tétel. A fenti feltételek mellett $\dfrac{F'}{F} \leq 1 + \dfrac{m-1}{m'}$.

Bizonyítás. Tekintsük a vesszős paraméterekhez (S'-höz) tartozó D' ütemezési diagramot. Legyen a $[0, F')$ intervallum pontjainak két – A és B – részhalmazának definíciója a következő: $A = \{t \in [0, F) \mid \text{a } t \text{ időpontban minden processzor foglalt}\}$, $B = [0, F') \setminus A$. Érdemes megemlíteni, hogy mindkét halmaz diszjunkt, félig nyílt (balról zárt, jobbról nyílt) intervallumok uniója.

Legyen T_{j_1} egy olyan taszk, melynek a végrehajtása D' szerint az F' időpontban fejeződik be (azaz $f_{j_1} = F'$). Ekkor két lehetőség van: a T_{j_1} taszk s_{j_1} kezdőpontja vagy B belső pontja, vagy nem az.

1. Ha s_{j_1} B belső pontja, akkor B definíciója szerint van olyan processzor, amelyre $\varepsilon > 0$ mellett igaz, hogy az $[s_{j_1} - \varepsilon, s_{j_1})$ intervallumban nem dolgozik. Ez azonban csak úgy fordulhat elő, ha van olyan T_{j_2} taszk, amelyre $\mathrm{T}_{j_2} < \mathrm{T}_{j_1}$ és $f_{j_2} = s_{j_1}$ (a eset).

2. Ha s_{j_1} nem belső pontja B-nek, akkor vagy $s_{j_1} = 0$ (b eset), vagy $s_{j_1} > 0$. Ha van B-nek s_{j_1}-nél kisebb eleme (c eset), akkor legyen $x_1 = \sup\{x \mid x < s_{j_1}$ és $x \in B\}$, egyébként pedig legyen $x_1 = 0$ (d eset). Ha $x_1 > 0$, akkor A és B konstrukciójából következik, hogy van olyan processzor, amelyhez talaálható olyan T_{j_2} taszk, amelynek ebben az intervallumban még tart a végrehajtása, és amelyre $\mathrm{T}_{j_2} < \mathrm{T}_{j_1}$.

A két esetet összegezve tehát vagy van olyan $\mathrm{T}_{j_2} < \mathrm{T}_{j_1}$ taszk, amelyre $y \in [f_{j_2}, s_{j_1})$ esetén $y \in A$ (a vagy c eset), vagy pedig minden $x < s_{j_1}$ számra $x \in A$ és $x < 0$ egyike fennáll (a vagy d eset).

Ezt az az eljárást ismételve olyan $\mathrm{T}_{j_r}, \mathrm{T}_{j_{r-1}}, \ldots, \mathrm{T}_{j_1}$ taszklánchoz jutunk, amelyre igaz, hogy $x < s_{j_r}$ esetén vagy $x \in A$ vagy $x < 0$. Ezzel megmutattuk, hogy léteznek olyan taszkok, amelyekre

$$\mathrm{T}_{j_r} <' \mathrm{T}_{j_{r-1}} <' \cdots <' \mathrm{T}_{j_1}, \tag{3.1}$$

továbbá minden $t \in B$ időpontban van olyan processzor, amelyik dolgozik, és éppen a lánc valamelyik elemét hajtja végre. Ebből következik, hogy

$$\sum_{\phi \in S'} t'(\phi) \leq (m' - 1) \sum_{k=1}^{r} t'_{j_k}, \tag{3.2}$$

ahol ϕ-vel az üres periódusokat jelöltük, és így a baloldali összeg az S'-ben lévő összes üres periódusra vonatkozik.

(3.1) és $<' \subseteq <$ alapján $\mathrm{T}_{j_r} < \mathrm{T}_{j_{r-1}} < \cdots < \mathrm{T}_{j_1}$, és így

$$F \geq \sum_{k=1}^{r} t_{j_k} \geq \sum_{k=1}^{r} t'_{j_k}. \tag{3.3}$$

Mivel

$$mF \geq \sum_{i=1}^{n} t_i \geq \sum_{i=1}^{n} t'_i, \tag{3.4}$$

és $F' = \dfrac{1}{m'}\left(\sum_{k=1}^{n} t'_k + \sum_{\phi\in S'} t'(\phi)\right)$, így (3.2), (3.3) és (3.4) alapján
$F' \le \dfrac{1}{m'}\Big(mF + (m'-1)F\Big)$, ahonnan $\dfrac{F'}{F} \le 1 + \dfrac{m-1}{m'}$.

A következő példák nemcsak azt mutatják meg, hogy a tételben szereplő korlát a lehető legjobb, hanem azt is, hogy az adott korlát (legalábbis aszimptotikusan) a paraméterek bármelyikének változtatásával elérhető.

3.6. Ütemezési anomáliák II.

3.7. példa. Ebben a példában a lista változik, $<$ üres, m tetszőleges. A végrehajtási idők a következők:

$$t_i = \begin{cases} 1, & \text{ha } i = 1,\dots,m-1; \\ m, & \text{ha } i = m; \\ m-1, & \text{ha } i = m+1,\dots,2m-1. \end{cases}$$

Ha ezt a τ_4 taszkrendszert m processzorra az $L = (\mathrm{T}_1,\dots,\mathrm{T}_n)$ listával ütemezzük, akkor a 3.19. ábrán lévő $S_7(\tau_4)$ optimális ütemezést kapjuk.

S_7 :

T_1	T_{m+1}
T_2	T_{m+2}
$\vdots$	$\vdots$
T_{m-1}	T_{2m-1}
T_m	

3.19. ábra. Az $L = (\mathrm{T}_1,\dots,\mathrm{T}_n)$ listához tartozó $S_7(\tau_4)$ ütemezés

Ha az L lista helyett az $L' = (\mathrm{T}_{m+1},\dots,\mathrm{T}_{2m-1},\mathrm{T}_1,\dots,\mathrm{T}_{m-1},\mathrm{T}_m)$ listát alkalmazzuk, akkor a 3.20. ábrán látható $S_8(\tau_4)$ ütemezést kapjuk. Ebben az esetben $F = F_{\max}(S_7) = m$, $F' = F_{\max}(S_8) = 2m-1$, így $\frac{F'}{F} = 2 - \frac{1}{m}$; tehát a lista változtatásával elértük, hogy a tétel állításában az egyenlőség teljesüljön, vagyis a $\le$ jel jobb oldalán szereplő kifejezés nem csökkenthető.

3.8. példa. Ebben a példában a végrehajtási időket csökkentjük. A lista mindkét esetben az $L = L' = (\mathrm{T}_1,\dots,\mathrm{T}_n)$. Itt és a fejezet további részében ε tetszőleges kis pozitív számot jelöl. Az eredeti végrehajtási időket a $\mathbf{t} = (t_1,\dots,t_n)$ vektor

S_8 :	T_{m+1}				T_m
	T_{m+2}				-
	$\vdots$				$\vdots$
	T_{2m-1}				-
	T_1	T_2	$\dots$	T_{m-1}	-

3.20. ábra. Az L' listához tartozó $S_8(\tau_4)$ ütemezés

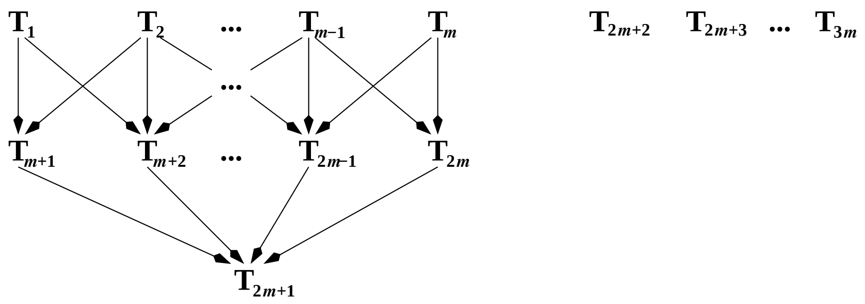
tartalmazza, ahol

$$t_i = \begin{cases} 2, \varepsilon & \text{ha } i = 1, \dots, m; \\ 1, & \text{ha } i = m+1, \dots, 2m; \\ m-1, & \text{ha } i = 2m+1, \dots, 3m. \end{cases}$$

Az új végrehajtási idők pedig:

$$t_i' = \begin{cases} t_i - \varepsilon, & \text{ha } i = 1, \dots, m-1; \\ t_i, & \text{ha } i = m, \dots, 3m. \end{cases}$$

A τ_5, illetve a módosított τ_5' taszkrendszer megelőzési gráfját a 3.21. ábra mutatja, az $S_9(\tau_5)$ (optimális) ütemezés és az $S_{10}(\tau_5')$ ütemezés pedig a 3.22. ábrán látható. Itt



3.21. ábra. τ_5 és τ_5' azonos gráfja

$F = F_{\max}(S_9(\tau_5)) = m + 2\varepsilon$ és $F' = F_{\max}(S_{10}(\tau_5') = 2m - 1 + \varepsilon$, ezért ε csökkenésével $\frac{F'}{F}$ tart a $2 - \frac{1}{m}$ értékhez ($\lim_{\varepsilon \to 0} \frac{F'}{F} = 2 - \frac{1}{m}$). Tehát a végrehajtási időket változtatva tetszőlegesen megközelíthetjük a tételben szereplő korlátot.

3.9. példa. Ebben a példában a megelőzési korlátozásokat gyengítjük. A τ_6 taszkrendszer megelőzési gráfját a 3.23. ábra mutatja. A taszkok végrehajtási ideje pedig: $t_1 = \varepsilon$, $t_i = 1$, ha $i = 1, \dots, m^2 - m + 1$, és $t_{m^2-m+2} = m$. τ_6-nak az

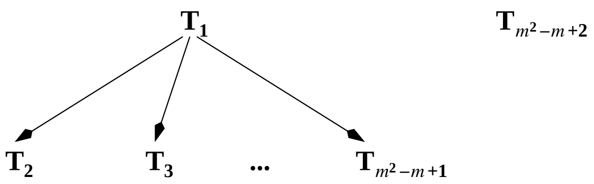
S_9 :

T_1	T_{m+1}	T_{2m+1}
T_2	T_{m+2}	T_{2m+2}
$\vdots$	$\vdots$	$\vdots$
T_{m-1}	T_{2m-1}	T_{3m-1}
T_m	T_{2m}	T_{3m}

S_{10} :

T_1	T_{2m}		T_{2m-1}	T_{2m+1}
T_2	T_{2m+3}		-	
$\vdots$	$\vdots$		$\vdots$	
T_{m-1}	T_{3m}		-	
T_m	T_{m+1}	$\dots$	T_{2m-1}	-

3.22. ábra. Az $S_9(\tau_5)$ és $S_{10}(\tau_5')$ ütemezések



3.23. ábra. A τ_6 taszkrendszer gráfja

$L = (T_1, \dots, T_{m^2-m+2})$ listához tartozó, optimális $S_{11}(\tau_6)$ ütemezését a 3.24. ábra tartalmazza. Az összes megelőzési korlátozás elhagyásával kapjuk τ_6-ból a τ_6' taszkrendszert. A 3.25. ábra mutatja az $S_{12}(\tau_6')$ ütemezést.

S_{11} :

T_1	T_2	T_{m+1}	$\dots$	T_{m^2-2m+2}
T_{m^2-m+2}				-
-	T_3	T_{m+2}	$\dots$	T_{m^2-2m+3}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
-	T_m	T_{2m-1}	$\dots$	T_{m^2-m+1}

3.24. ábra. Az $S_{11}(\tau_6)$ optimális ütemezés

3.10. példa. Ezúttal a processzorok számát növeljük: m-ről m'-re. A τ_7 taszkrend-

S_{12} :

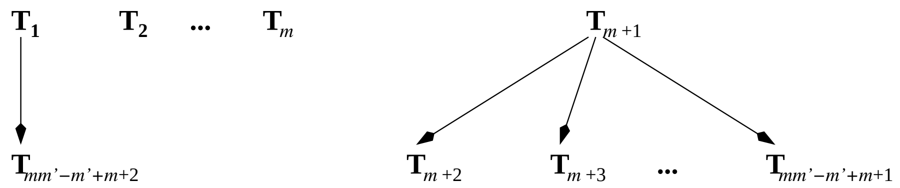
T_1	T_{m+1}	$\cdots$	T_{m^2-m+1}	-
T_2	T_{m+2}	$\cdots$	T_{m^2-m+2}	
T_3	T_{m+3}	$\cdots$	-	
$\vdots$	$\vdots$	$\ddots$	$\vdots$	
T_{m-1}	T_{2m+1}	$\cdots$	-	
T_m	T_{2m}	$\cdots$	-	

3.25. ábra. Az $S_{12}(\tau_6')$ ütemezés

szer gráfját a 3.26. ábra mutatja, a végrehajtási idők pedig:

$$t_i = \begin{cases} \varepsilon, & \text{ha } i = 1, \ldots, m+1; \\ 1, & \text{ha } i = m+2, \ldots, mm' - m' + m + 1; \\ m', & \text{ha } i = mm' - m' + m + 2. \end{cases}$$

A taszkrendszer optimális ütemezését m, illetve m' processzoron a 3.27. illetve a 3.28. ábra mutatja. A maximális befejezési időket összehasonlítva itt is a kívánt aszimptotikus értéket kapjuk: $F = F_{\max}(S_{13}(\tau_7)) = m' + 2\varepsilon$, $F' = F_{\max}(S_{14}(\tau_7)) = m' + m - 1 + \varepsilon$, így $\frac{F'}{F} = 1 + \frac{m-1-\varepsilon}{m'+2\varepsilon}$, valamint $\lim_{\varepsilon \to 0} = \frac{F'}{F} = 1 + \frac{m-1}{m'}$.



3.26. ábra. τ_7 rákövetkezési gráfja

S_{13} :

T_1	T_{m+1}	T_{m+2}	$\cdots$	T_a
T_2	$T_{mm'-m'+2}$			-
T_3	-	T_{m+3}	$\cdots$	T_b
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
T_m	-	T_{2m}	$\cdots$	T_c

3.27. ábra.
Az $S_{13}(\tau_7)$ optimális ütemezés ($a = mm' - m' + 3$, $b = a + 1$, $c = mm' - m' + m + 1$)

S_{14} :

T_1	T_{m+2}	$\cdots$	T_a	$\mathrm{T}_{mm'-m'+m+2}$
T_2	T_{m+3}	$\cdots$	T_{a+1}	-
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
T_b	T_c	$\cdots$	T_d	-
-	$\vdots$	$\ddots$	$\vdots$	$\vdots$
-	T_e	$\cdots$	T_f	-

3.28. ábra.
Az $S_{14}(\tau_7)$ optimális ütemezés ($a = mm' - 2m' + m + 2$, $b = m + 1$, $c = 2m + 2$, $d = mm' - 2m' + 2m + 2$, $e = m + m' + 1$, $f = mm' - m' + m + 1$)

Ezekkel a példákkal bebizonyítottuk a következő tételt.

3.6. tétel. A 3.4. tételben szereplő korlát a négy paraméter ($\mathbf{t}$, $m, <, L$) bármelyikének a változtatásával tetszés szerinti pontossággal megközelíthető.