Haliho

Kombinatorikus módszerek az ütemezéselméletben

Matematikus diplomamunka
Vaik Zsuzsanna*
Témavezető: Jordán Tibor

2003

Kivonat

Ebben a diplomamunkában néhány ütemezéselméleti problémát mutatok be. Olyan párhuzamos, illetve uniform gépekre adott ütemezési feladatokat nézek, melyeknél a munkák halmazán megelőzési feltételek adottak. Azaz a munkák halmazán adott egy irányított aciklikus gráf. Legtöbbször a cél az utolsó munka befejezési idejének minimalizálása, de olvashatunk néhány olyan példát is, melyben a maximális késést szeretnénk minimalizálni. Ezen feladatok általában már megelőzési feltételek nélkül is az $\mathcal{NP}$-nehéz problémák közé tartoznak. Azonban, ha még további feltételeket adunk a munkákra, vagy a gépekre, (például, hogy a munkák egységnyi idő alatt futnak, vagy a gépek száma meghatározott), abban az esetben polinom időben megoldhatóvá válhatnak. Ebben a dolgozatban a teljesség igénye nélkül bemutatom, hogy milyen sokféle módszert lehet alkalmazni az ütemezési feladatok megoldására: a gráfelmélet problémáitól kezdve a lineáris programozáson át szinte mindent.

*ELTE TTK, Matematikus V., zsuzska@cs.elte.hu

Tartalomjegyzék

1. Bevezető

A dolgozat 2. fejezetében olyan feladatokat mutatok be, melyeknél párhuzamos gépek vannak, azaz minden gép azonos tulajdonságú, azonos sebességgel működik, s bármely munkát bármely gépre ütemezhetünk. A fejezet első felében olyan problémákat nézek, melyeknél megelőzési feltételek adottak, és cél az utoljára véget érő munka befejezési idejének minimalizálása. Garey és Johnson [10] belátta, hogy ha nincsenek megelőzési feltételeink, a feladat már akkor is $\mathcal{NP}$-nehéz. Továbbá azt is belátták, hogy ha minden munka futási ideje egységnyi idejű, a feladat akkor sem lesz könnyebb. Ezt a problémát egy ismert $\mathcal{NP}$-nehéz gráfelméleti problémára vezették vissza. A következő alfejezetben bemutatom Graham [11] által adott listás ütemezést, amely 2-approximációs algoritmus a $P|prec|C_{max}$ feladatra. A 2.3 fejezetben látható, amennyiben a gépek számára erős megkötést teszünk, ($m = 2$), akkor a feladat polinomiálisan megoldható, sőt Garey és Johnson algoritmusa [9] az általánosabb L_{max} célfüggvényre is optimális megoldást ad. A 2.5 fejezetben, a $P|outtree, r_j, p_j = 1|\sum C_j$ feladatban a munkák befejezési idejének összegét szeretnénk minimalizálni. Erre Brucker, Hurink és Knust adott algoritmikus megoldást. Végül a 2.6 fejezetben a $P|tree, p_j = 1|C_{max}$ probléma és a matroid bázis fedési problémája közötti, Marcotte és Trotter [17] által bebizonyított összefüggést olvashatjuk. A matroidelmélet alapfogalmait a 1.1 fejezetben leírom, azonban csak érintőlegesen, amennyire a bizonyítás megértéséhez feltétlenül szükséges. Ha ezáltal felkeltettem a kedves olvasó érdeklődését a matroidok iránt, ajánlom figyelmébe Frank jegyzetét [8].

A 3. fejezetben uniform gépeken való ütemezési feladatok olvashatók, azaz a gépek nem feltétlenül azonos sebességűek. Ebben a fejezetben is olyan feladatokkal foglalkozom, ahol a munkák között megelőzési feltételek adottak. $Q||C_{max}$ feladat szintén $\mathcal{NP}$-nehéz, (hiszen amennyiben minden sebesség egyenlő, visszajutunk a $P||C_{max}$ feladatra). Azonban erre a feladatra is adható egy 2-approximációs algoritmus, mely egy egyszerű listás ütemezésen alapuló algoritmus. Ennek segítségével könnyen adható a $Q|chains|C_{max}$ esetre egy 2-approximációs algoritmus. Meglepő azonban, hogy a $Q|prec|C_{max}$ feladat megoldására az eddig legjobb ismert approximációs algoritmus faktora $\mathcal{O}(\sqrt{\log m})$. Erre először Chudak és Shmoys adott egy lineáris programozáson alapuló algoritmust [6], majd Bender és Chekuri [1] adott egy másik algoritmust, mely a lineáris programozási feladat helyett a megelőzési gráf egy felbontását használja a munkák a gépekre való kijelöléséhez. Ezeket a bizonyításokat mutatom be a 3.1 ill. a 3.2 alfejezetekben. Végül a $Q2|chain, p_j = 1|C_{max}$ feladatra bemutatom Brucker, Hurink és Kubiak [4] polinomiális algoritmusát.

Köszönöm témavezetőmnek, Jordán Tibornak, hogy felkeltette érdeklődésemet a kombinatorikus optimalizálás e területe iránt, és köszönöm, hogy már több, mint egy éve segít felkészülni a dolgozat megírására.

1.1. Fogalmak, jelölések

A szakdolgozatom néhány ütemezési feladatot mutat be. Nézzük hát, mit takar ez a fogalom, hogy ütemezés: Tegyük fel, hogy adva van m gép, jelölje ezeket M_i, $(i = 1, ..., m)$, továbbá adott n munka, j, $(j = 1, ..., n)$. Ezen munkákat az adott gépeken kell elvégeznünk. Egy ütemezés nem más, minthogy minden munkát kijelölünk egy vagy több gépre, egy –vagy több– időintervallumban, úgy, hogy egy munka egy időpillanatban csak egy gépen fusson, és egy gépen egyszerre legfeljebb egy munka legyen, s az ütemezés végére minden munkát befejezzük. Egy S ütemezésben jelölje $s(j)$ azt az időpontot, mikor S a j munkát gépre rakta. Mivel ebben a dolgozatban a legtöbb esetben nem engedjük meg, hogy megszakítsuk a munkák végzését és esetlegesen egy másik gépen folytassuk, így $s(j)$ egyértelműen meghatározott. Ha megengedünk megszakításokat, legyen $s(j)$ az első olyan pillanat, mikor j-t gépre raktuk. Minden munkára adott, hogy mennyi ideig tart elvégezni egy egységnyi sebességű gépen, azaz minden $j \in J$ munkához adott a p_j futási idő. Jelölje C_j azt az időpillanatot, mikor a $j \in J$ munka befejeződik. Vannak olyan esetek, mikor nem áll azonnal minden munka rendelkezésünkre, ekkor minden munkához adott egy időpillanat, mielőtt nem rakhatjuk gépre, legyen ez r_j $(j = 1, ..., n)$. (Ekkor természetesen egy megengedett S ütemezésre $s(j) \geq r_j$). Előfordul, hogy a munkák halmazán megelőzési feltételek adottak, azaz adott egy antiszimmetrikus reláció a munkák között, $j \prec k$ azt jelöli, hogy a k munka addig nem kerülhet gépre, míg a j munka be nem fejeződött. A reláció tulajdonságaiból következően ezt egy irányított aciklikus $G = (V, A)$ gráffal is megadhatjuk, ahol a $V = \{1, ..., n\}$ megfeleltethető a munkák halmazának, és $(i, k) \in A$ pontosan akkor, ha $i \prec k$. Előfordul, hogy ez a megelőzési gráf egy speciális gráf, pl. ki-, vagy befenyő(k), (akár láncok) uniója. (Ahol egy kifenyő alatt egy olyan irányított gráfot értünk, melynek van egy gyökere, azaz egy olyan pont, melynek a befoka 0 és minden más pont befoka egy. Hasonlóan definiálható a befenyő is). Előfordul, hogy minden $j \in J$ munkához adott egy d_j nem negatív szám, amely a befejezési határidőt jelöli. Azaz szeretnénk, ha a j munka d_j idő előtt befejeződne: $C_j \leq d_j$. Persze ez nem mindig lehetséges, ekkor definiálhatjuk a j munka késését, azaz legyen $L_j = C_j - d_j$.

Még sok más tulajdonságú feladat lehetséges, azonban e dolgozatnak nem célja az ütemezéselméleti problémák teljes választékát bemutatni. Amennyiben a kedves olvasó kíváncsisága ezzel nem csillapult, még további érdekes feladatot és algoritmust talál Brucker [2] könyvében.

Egy S ütemezés akkor megengedett, ha minden munka minden időpillanatban legfeljebb egy gépen fut, és a munkák feltételei nem sérülnek. Persze többféle megengedett ütemezés létezhet, a célunk megkeresni egy adott célfüggvényre optimális megengedett ütemezést. Ebben a dolgozatban legtöbbször az utoljára véget érő munka befejezési idejét szeretnénk minimalizálni ($C_{max} \to \min$), de lesz példa arra, mikor a legnagyobb késés nagyságát minimalizáljuk ($L_{max} \to \min$), vagy a befejezési idők összegének a minimumát keressük ($\sum_{i=1}^{n} C_i \to \min$). A megfelelő optimumokat jelölje C^*_{max}, L^*_{max}.
Használni fogom az ütemezési problémák rövidebb hivatkozására a $\alpha|\beta|\gamma$ jelölést, melyet Graham, Lawler, Lenstra és Rinnooy Kan [12] vezettek be. Az α a gépek tulajdonságát, β az adott munkák tulajdonságát, és végül γ a célfüggvényt jelöli. Ebben a dolgozatban tehát azon esetek fordulnak elő, ahol az $\alpha \in \{P, P2, Pn, Q, Q2\}$, a $\beta \subseteq \{prec, tree, p_j = 1, r_j, chains\}$ és végül $\gamma \in \{C_{max}, L_{max}, \sum C_j\}$.
A dolgozatban használni fogok az alábbi fogalmakat:

- *Listás ütemezés*: tegyük fel, hogy a munkákat valamilyen szempont szerint sorba raktuk; amint szabaddá válik egy gép a listás ütemezés a sorrendben következő elérhető munkát rakja rá.

- *LPT-sorrend*: a munkákat nem növekvő p_j-k szerint rakjuk sorba, azaz $p_1 \geq p_2 \geq \dots \geq p_n$.

- *SPT-sorrend*: a munkákat nem csökkenő p_j-k szerint rakjuk sorba, azaz $p_1 \leq p_2 \leq \dots \leq p_n$.

- *EDD-sorrend*: a munkákat nem növekvő d_j-k szerint rakjuk sorba, azaz $d_1 \geq d_2 \geq \dots \geq d_n$

- *Topologikus sorrend*: a gráf pontjait úgy rakjuk sorba, hogy egy pontból csak az őt sorrendbe megelőző pontokba vezessen él.

- *Matroid*: a matroid olyan véges absztrakt struktúra, amely egy $(S, \mathcal{F})$ párral adható meg, ahol A véges halmaz, $\mathcal{F}$ pedig az S részhalmazainak olyan rendszere, mely kielégíti a következő axiómákat:

  - $\emptyset \in \mathcal{F}$
  - ha $X \subseteq Y \in \mathcal{F}$, akkor $X \in \mathcal{F}$
  - minden $X \subseteq S$ részhalmazra az $\mathcal{F}$ tartalmazásra nézve maximális X-ben fekvő tagjai azonos elemszámúak

Az $\mathcal{F}$ tagjait a matroid *független* halmazainak nevezzük, a nem független halmazt nevezzük *függő*nek. A harmadik axióma szerint minden X részhalmazban a tovább már nem bővíthető független halmazok elemszáma azonos, legyen ez a szám az X halmaz *rang*ja, jelöljük ezt $r(X)$-el. Nyilván minden $X \subseteq S$ halmazra $r(X) \leq |X|$. S maximális függetlenjeit nevezzük a matroid *bázis*ainak.

- *Transzverzális matroid*: Egy $G = (S, T; E)$ páros gráfban S párosítható részhalmazai matroidot alkotnak, s az ilyen matroidot nevezzük transzverzális matroidnak.

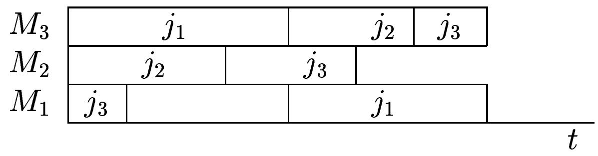
- *α-approximációs algoritmus*: $\mathcal{NP}$-nehéz optimalizálási (minimalizálási) problémára szeretnénk olyan algoritmust adni, amely polinom időben valamilyen szempontból „jó" megoldást ad. Egy algoritmust α-approximációs algoritmusnak nevezünk, amennyiben

  - megengedett megoldást ad polinomiális időben
  - a megoldás értéke bizonyíthatóan nem több, mint az optimális megoldás α-szorosa.

  Ekkor α az algoritmus *faktor*a. Az algoritmus persze annál jobb, minél közelebb van α értéke 1-hez. Az $\mathcal{NP}$-nehéz minimalizálási feladatok három osztályba sorolhatók:

  - bármely $\alpha > 1$-re létezik α-approximációs algoritmus
  - létezik olyan $\beta > 1$, melynél kisebb α-ra nem létezik α-approximációs algoritmus (amennyiben feltesszük, hogy $\mathcal{P} \neq \mathcal{NP}$)
  - semmilyen konstans α-ra nem létezik α-approximációs algoritmus.

Használni fogom a következő jelölést egy ütemezés ábrázolására, amelyben könnyedén leolvashatjuk a vízszintes tengely mentén, hogy mely időpillanatban mely munkát mely gépre ütemeztük.



2. Párhuzamos gépek

2.1. $P|prec, p_j = 1|C_{max}$

Amennyiben a gépek száma nem fix, a feladat $\mathcal{NP}$ nehéz, melyet Garey és Johnson [10] látott be, egy ismert $\mathcal{NP}$-nehéz gráfelméleti problémát visszavezetve a feladatra. Abban az esetben, ha a gépek száma kettő, ($m = 2$), a feladat polinomiálisan megoldhatóvá válik, melyre a 2.3 fejezetben bemutatjuk Garey és Johnson algoritmusát [9]. Az m fix eset bonyolultsága még nyitott kérdés.

Tétel 2.1. *Annak eldöntése, hogy $P|prec, p_j = 1|C_{max}$ feladatban létezik-e $C_{max} \leq 3$ értékű ütemezés, $\mathcal{NP}$-teljes probléma.*

Bizonyítás:
Tekintsük a következő problémát, amelyről tudjuk, hogy $\mathcal{NP}$-teljes: Legyen adott egy $G = (V, E)$ gráf, és $k \in \mathbb{Z}$. Kérdés, hogy létezik-e olyan $T \subseteq V$, mely teljes részgráfot feszít és $|T| = k$? (Az ilyen részgráfot nevezzük továbbiakban *klikk*-nek.)
Megmutatjuk, hogy ez a kérdés visszavezethető egy $P|prec, p_j = 1|C_{max}$ feladatra. Készítsük el a következő ütemezési feladatot:
Legyen $l := \binom{k}{2}$, $k' := |V| - k$, $l' := |E| - l$, $m := \max\{k, l + k', l'\} + 1$
Legyenek a munkák a következők:
Minden $v \in V$ illetve, minden $e \in E$ élhez rendeljünk egy-egy j_v ill. j_e munkát, és ezen kívül legyen adott még három munka-halmaz:
$X = \{x_1, x_2, ..., x_{m-k}\}$, $Y = \{y_1, y_2, ..., y_{m-k'-l}\}$, $Z = \{z_1, z_2, ..., z_{m-l'}\}$.
Ekkor összesen $k + k' + l' + l + m - k + m - k' - l + m - l' = 3m$ munkák van.
A munkák közötti megelőzési feltételek legyenek a következők:
Ha v pont az e él egyik végpontja, akkor legyen $j_v \prec j_e$, illetve minden $x \in X, y \in Y, z \in Z$-re: $x \prec y \prec z$.

Lemma 2.2. *Pontosan akkor létezik olyan ütemezés, melyre C_{max} értéke legfeljebb 3, ha a G gráfban létezik klikk.*

Bizonyítás:
Elsőnek megmutatjuk, hogy ha a G gráfban létezik klikk (legyen ez $T = (V_T, E_T)$), akkor létezik olyan ütemezés, melyre a C_{max} legfeljebb 3. (Adunk egy 3 hosszú ütemezést).
Ütemezzük az első időegységre az X halmaz munkáit, és még azon pontoknak megfelelő munkákat, melyek benne vannak a klikkben: (j_v: $v \in T$). Második időegységre ütemezzük az Y halmaz munkáit, és a gráf azon pontjaihoz tartozó munkákat, melyek még nem voltak, (amelyek nincsenek benne a klikkben), illetve azon élekhez tartozó munkákat, mely élek benne vannak

a klikk által feszített részgráfban. A harmadik időegységre ütemezzük a Z halmaz munkáit, és a maradék élekhez tartozó munkákat.
Megmutatjuk, hogy ez egy megengedett ütemezés, azaz

- Minden időegységben legfeljebb m munkát ütemeztünk:
  - első időegységben: $|X| + |V_T| = m - k + k = m$
  - második időegységben:
    $|Y|+|V\backslash V_T|+|E_T| = m-k'-l+|V|-k+\binom{k}{2} = m-k'-l+k'+l = m$
  - harmadik időegységben:
    $|Z| + |E\backslash E_T| = m - l' + |E| - \binom{k}{2} = m - l' + l' = m$
- A megelőzési feltételek nem sérülnek: X, Y és Z halmazok munkái között nyilván teljesülnek a feltételek, és a gráf pont és élek közötti összefüggéseiből adódó megelőzési feltételek sem sérültek, ugyanis az összes klikk-élhez tartozó munka a második időegységben kerül a gépekre, s ezidőre a klikk-pontokhoz tartozó munkák már befejeződtek, míg a nem klikk élekhez tartozó munkák a harmadik időegységben kerültek a gépekre, amikorra már az összes olyan munka befejeződött, amely ponthoz lett rendelve.

Tehát az ütemezés megengedett, és $C_{max} = 3$.
Most megmutatjuk, hogy ha adott egy ütemezés, melyre $C_{max} \leq 3$, akkor létezik a G gráfban klikk. Tekintsünk egy ilyen S ütemezést.
Először tegyünk néhány megfigyelést:

- Az S ütemezés szerint a megelőzési feltételek miatt az X halmaz munkái az első időegységben, az Y halmaz munkái a második időegységben, míg a Z halmaz munkái a harmadik időegységben kell, hogy fussanak a gépeken.
- Mindhárom időegységben minden gép foglalt, mert $3m$ munka van, és m gép.
- Az első időpillanatban X halmaz munkáin kívül csak pontokhoz tartozó munkák futhatnak. (A megelőzési feltételek miatt egy élhez tartozó munka csak akkor kerülhet gépre, mikor az ő végpontja már végzett, azaz leghamarabb a második időpillanatban.) Ezen munkákhoz tartozó pontok halmaza legyen T. Az előző megfigyelés miatt $|T| = k$.

Ekkor tegyük fel indirekt, hogy nincs klikk a G gráfban, azaz T nem egy teljes részgráf, így kevesebb, mint $\binom{k}{2} = l$ élt feszít. Ekkor, ha az S az összes nem T-beli ponthoz tartozó munkát a második időegységben ütemezte, akkor is kell,

hogy legyen olyan T által nem feszített élhez tartozó munka, mely szintén a második időegységben fut. (Mivel az összes gép foglalt, ugyanakkor a T által feszített élek száma és Y halmaz elemszáma kisebb, mint m a definíció szerint.) Legyen az $e \in E$ él olyan, mely nincs benne a T által feszített részgráfban, és a hozzátartozó munka a második időegységben fut. Ekkor e egyik végpontjához tartozó munka biztos nem az első időegységben fut, és ez megsérti a megelőzési feltételeket. Tehát nem lehet az S ütemezés megengedett, azaz ellentmondáshoz jutottunk.
Ezzel a tételt bebizonyítottuk. $\square$

2.2. $P|prec|C_{max}$

Láttuk, hogy ez a probléma már a $p_j = 1$ megkötéssel is $\mathcal{NP}$-nehéz. Graham [11] megmutatta, hogy már egy egyszerű listás ütemezés 2-approximációs megoldását adja a feladatnak. Azaz, ha munkákat valamilyen szempont szerint sorbarendezzük, majd a megelőzési feltételek figyelembevételével a gépekre ütemezzük, akkor egy olyan approximációs algoritmust kapunk, melynek faktora 2.
Legyen j munka a t időpillanatban *elérhető*, ha j minden elődje, azaz minden $k \prec j$ munka, elkészült már a t időre. Ekkor amennyiben,
$\mathcal{C} = \{j_{t_1}, ..., j_{t_k}\}$ egy lánc a részben rendezésben, akkor következő alsó korláthoz jutunk:

$$C^*_{max} \geq \sum_{i=1}^{k} p_{t_i}$$

Ezt felhasználva a következő eredményt kapjuk:

Tétel 2.3. *A listás ütemezés 2-approximációs algoritmust ad a $P|prec|C_{max}$ feladatra.*

Bizonyítás:
Legyen j_1 az a munka, mely utoljára ér véget. Legyen j_2 a j_1 elődei közül az utolsónak végződő ... és így tovább, legyen j_k az utolsó a sorban, amelynek nincs elődje. Ekkor ezen munkák egy láncot alkotnak, legyen ez:
$\mathcal{C} = \{j_1, j_2, ..., j_k\}$. Legyen továbbá A azon időtartam, amíg a $\mathcal{C}$ lánc valamely munkája épp valamelyik gépen van, s legyen B azon időtartam, mikor egy olyan munka sincs a gépeken, mely $\mathcal{C}$-hez tartozik. Ekkor

$$C_{max} = A + B = \sum_{j \in \mathcal{C}} p_j + \frac{\sum p_j}{m} \leq C^*_{max} + C^*_{max} = 2C^*_{max}$$

Megjegyzés: B-ben nincs állásidő, különben $\mathcal{C}$ valamely munkáját a szabad gépre rakhatnánk, hiszen listás ütemezés szerint ütemeztünk. $\square$

2.3. $P2|prec, p_j = 1|L_{max}$

Ha figyelembe vesszük, hogy amennyiben $d_j = 0$, a feladat megegyezik a $P2|prec, p_j = 1|C_{max}$ esettel, ez a probléma magában foglalja az előző feladat egy speciális esetét. (Mikor a gépek száma pontosan 2, ez a megkötés az $\mathcal{NP}$-nehéz feladatot polinomiálisan megoldhatóvá teszi.)
Tehát legyen adva két egyforma gép, J munkák halmaza, melyre minden $j \in J$ munkára $p_j = 1$, és a munkák halmazán adott egy irányított aciklikus gráf.
Elsőnek adunk egy algoritmust mely ad egy ütemezést, majd megmutatjuk, hogy ez egy optimális ütemezés lesz.

Algoritmus

1. fázis: Számoljuk ki minden j munkára a d'_j módosított határidőt a következőképp:
Legyen minden $i \in J$ munkára

$$S(i) := \{j \in J : \ j \text{ elérhető } i\text{-ből}\},$$

azon munkák halmaza, melyek szigorúan csak i után kerülhetnek sorra. Majd minden $r \in \mathbb{R}$-re, legyen

$$g(i, r) := |\{k \in S(i) : d'_k \le r\}|,$$

ekkor legyen

$$d'_i := \min\left\{d_i, \min\{d'_j - \left\lceil \frac{g(i, d'_j)}{2} \right\rceil : j \in S(i)\}\right\}.$$

Megjegyzés:

- a d'_i-k kiszámíthatóak egy topológikus sorrend szerint visszafele.
- a módosított határidők kiszámítására a következő megfigyelés motivál minket: ha minden olyan j munka elkészül d'_j határidőre, melyre $i \prec j$, akkor az i-edik munkának is el kell készülnie d'_i határidőre.

2. fázis Rakjuk d'_j-k szerint EDD sorrendbe a munkákat, majd eszerint következzen a listás ütemezés.

Lemma 2.4. *Egy megengedett ütemezésben pontosan akkor készül el minden j munka d_j határidőre, amennyiben minden i munka elkészül d'_i határidőre.*

Bizonyítás:
Ha minden munka elkészül a módosított határidőkre, akkor nyilván minden munka elkészül az eredeti határidőkre, mivel $d'_i \leq d_i$.
Másik irány bizonyításához indirekt tegyük fel, hogy van olyan munka, ami elkészül az eredeti határidőre, de nem készül el a módosított határidőkre. Legyen j olyan munka, amely késik d'_j-re, és ezek közül is olyan, melyre $S(j)$ minimális méretű. Nyilván j nem lehet nyelő, mert ha j nyelő, akkor $d'_j = d_j$. Így $S(j)$ nem üres. Ha $j' \in S(j)$, akkor $S(j') \subseteq S(j)$, ezért, a feltevés szerint minden $j' \in S(j)$ elkészül a módosított határidőre nézve, de a d'_j definíciója miatt ekkor j is el kell készüljön d'_j-re. □

Tétel 2.5. *Ha létezik olyan ütemezés, amelyben minden munka elkészül a megadott határidőre, akkor az algoritmus ilyet ad.*

Bizonyítás:
Indirekt tegyük fel, hogy az algoritmus által adott S ütemezésben a j munka késik d_j-hez képest. Mivel $d'_j \leq d_j$, ezért j késik a módosított határidőre nézve is. Legyen j olyan munka, mely késik a módosított határidőre nézve és kezdési ideje minimális.
Megfigyelés:
Bármely $t < s(j)$ időpontban létezik olyan k munka a gépeken, melyre $d'_k \leq d'_j$, (ez következik az algoritmus választási szabályából és a módosított határidők definíciójából). Ekkor két eset lehet:

1. létezik olyan $t \leq s(j)$ időegység, amelyben pontosan egy olyan k munka kerül a gépre, melyre $d'_k \leq d'_j$. Legyen $[t, t+1]$ az utolsó ilyen időintervallum. Ekkor igaz lesz a következő két tulajdonság:

   (a) $[t, t+1]$ időben mindig két olyan munka fut a gépeken, melyek módosított határideje legfeljebb d'_j;

   (b) $[t+1, s(j)]$ időben minden munka benne van $S(k)$-ban is. (az algoritmus választási szabálya miatt).

   Ekkor (a) és (b) miatt

   $$d'_k \leq d'_j - \left\lceil \frac{g(k, d'_j)}{2} \right\rceil \leq s(k)$$

   így k is késik d'_k-re nézve. Ez ellentmondás, hiszen feltettük, hogy j a legkorábban kezdődő olyan munka, mely késik.

2. Minden $t < s(j)$-hez pontosan két olyan munka létezik, mely a $[t, t+1]$-ben fut, és amelyeknek a módosított határideje nem nagyobb a j munka

módosított határidejénél. Ekkor azonban j tetszőleges ütemezésben késik a módosított határidőre! Így az előző lemma miatt j tetszőleges ütemezésben késik az eredeti határidőre is, tehát ellentmond a tétel feltételének. □

Lemma 2.6. *Ha a d határidőkhöz a d' módosított határidők tartoznak, és $l \in \mathbb{R}$, akkor a $d + l$ határidőkhöz a $d' + l$ módosított határidők tartoznak.*

Bizonyítás:
Nyelőkre nyilvánvaló az állítás, mert ott megegyezik a módosított határidő az eredetivel. Egyébként meg a topologikus sorrendben visszafele vizsgálva következik a módosított határidők definíciójából. □

Tétel 2.7. *Az algoritmus optimális ütemezést talál a*

$$P2|prec, p_j = 1|L_{max} \text{ feladatra.}$$

Bizonyítás:
Legyen L^*_{max} az optimum. Pontosan akkor létezik S ütemezés, melyre minden $j \in J$-re $C_j - d_j \leq L^*_{max}$, ha a $d_j + L^*_{max}$ határidőre létezik késés nélküli ütemezés. Viszont, ha létezik ilyen ütemezés, akkor tudjuk, hogy az algoritmusunk ilyet talál, és az előző lemma miatt az algoritmusunk ugyanazt az ütemezést adja az eredeti határidőkre is. □

2.4. $P|tree, p_j = 1|C_{max}$

A $P|prec, p_j = 1|C_{max}$ egy másik speciális esete, mikor a részben rendezés gráfja egy ki-, vagy egy befenyő. Ez a feladat is polinomiálisan megoldható. Sőt, itt is többet igazolunk.

2.4.1. $P|tree, p_j = 1|C_{max}$

Amikor az F fenyő egy s gyökerű befenyő, Hu [13] a következő algoritmust adta:

1. Minden j munkához kiszámoljuk az $l(j)$ „szint” értéket, ami legyen az egyértelműen meghatározott j-s út hossza az F fenyőben. $[l(s) := 0]$

2. a $t = 0$ időpillanattól kezdve hajtsuk végre a következőket:
   Legyen L_t az éppen előd-nélküli munkák halmaza (a forrás, illetve az elérhetők), rakjuk őket sorba szintek szerint csökkenően. Ekkor ütemezzük a $[t, t + 1]$ időre az L_t halmazból az első $\min\{|L_t|, m\}$ munkát.

Tétel 2.8. *Az algoritmus optimális ütemezést ad.*

Bizonyítás:
Legyenek $V_0, V_1, ..., V_r$ a szinthalmazok. Amennyiben j szintje legalább i, akkor

$$C^*_{max} \geq i + \left\lceil \frac{\sum_{k=i}^{r} |V_k|}{m} \right\rceil$$

Belátjuk, hogy létezik olyan j munka, melyre $i = l(j)$ választással egyenlőség áll. Legyen j a legkorábban ütemezett olyan munka, amelynek s-től való „távolsága" az ütemezésben éppen $l(j)$. (Ilyen munka létezik, pl. s ilyen, tehát vehetjük ezen tulajdonságúak közül a legkorábban ütemezettet).
Megfigyelések:

- Mielőtt j-t ütemeztük, az előző időegységben ütemezett munkák egyike sem elődje j-nek. (j választása miatt, ugyanis ha lenne ilyen, akkor az a munka szintje pont eggyel kisebb lenne, mint $l(j)$, és pont eggyel korábban lett ütemezve, tehát azt a munkát választottuk volna j helyett.)

- Mielőtt j-t ütemeztük, az előző időegységben minden gép foglalt és ezen munkák szintje mindig legalább $l(j)$.

- Minden $s(j)$ előtti időegységben foglalt minden gép, és ezen munkák szintje legalább $l(j)$

A fenti megfigyelésekből következik, hogy $i = l(j)$ választással egyenlőség áll, tehát az algoritmus által adott ütemezésre a $C_{max} = i + \left\lceil \frac{\sum_{k=i}^{r} |V_k|}{m} \right\rceil$, amiből következik az optimalitás. □

Ez a gondolatmenet átvihető arra az esetre is, mikor a megelőzési feltételek gráfja nem be-, hanem kifenyő, vagy egy be-, vagy kifenyves. Megjegyzendő azonban, hogy amennyiben a gráf kifenyő(k) és befenyő(k) (vegyes)uniója, a feladat ismét $\mathcal{NP}$-nehézzé válik.

2.4.2. $P|tree, p_j = 1|L_{max}$

Ebben az esetben már nem mindegy, hogy kifenyővel vagy befenyővel adott a megelőzési feltételek gráfja, ugyanis Brucker, Garey és Johnson [3] belátták, hogy a kifenyő eset már $\mathcal{NP}$-teljes bonyolultságú. Visszavezették a feladatra a pontfedés problémáját. Konstruáltak egy $G = (V, E)$ gráfot, melyben annak eldöntése, hogy létezik-e V-nek egy k-pontú részhalmaza, mely minden élt lefog, ekvivalens azzal, hogy $P|outtree, p_j = 1|L_{max}$ problémának létezik-e olyan ütemezése, melyre egy munka sem késik, azaz $L_{max} = 0$.

Amennyiben a megelőzési feltételek gráfja befenyő, a feladat polinomiálisan megoldható:
Tudjuk, hogy amennyiben $i \prec k$, akkor $s(i) + 1 \leq s(k) \leq d_k - 1$. Ez motiválja következő módosított határidők bevezetését: $d'_i := \min\{d_i, d_k - 1\}$. Ezt könnyedén kiszámolhatjuk minden i munkára a következő algoritmussal:

1. A gyökérre legyen a $d' = d$.

2. Legyen $j \in J$ olyan, melyre még nem módosítottuk a határidőt, de a rákövetkező $j \prec k$ munkát már igen. Ekkor, ha nem létezik ilyen munka, akkor készen vagyunk. Ha létezik, akkor:

3. $d'_j := \min\{d_j, d'_k - 1\}$, majd menjünk a 2. lépésre.

Rakjuk a munkákat nem növekvő d'-k szerint sorba. Amíg tudunk, rendeljünk az időegységekhez munkákat úgy, hogy:

- egy időegységbe legfeljebb m munkát rendeljünk;

- sorba haladva a munkákon mindig azt a munkát rendeljük a gépre, melynek az elődjei már egy korábbi időegységben végeztek.

Ez polinomiálisan megoldható.

Lemma 2.9. *Pontosan akkor létezik olyan ütemezés, melyre minden munka elkészül a módosított határidőre, ha létezik olyan ütemezés, melyben az eredeti határidőre is elkészül minden munka.*

Bizonyítás:
Tegyük fel, hogy az S listás ütemezés szerint valamely munka késik az eredeti határidőre. Mivel pontosan akkor létezik az eredeti d_j-kre megengedett ütemezés, ha a módosított határidőkre nézve létezik, ezért S valamelyik módosított határidőre nézve is késik. Legyen $i \in J$ a késők közül az S által legkorábban ütemezett. Legyen t_0 azon legkésőbbi, de $\lfloor d'_i \rfloor$-nél kisebb egész időpillanat, mikor m-nél kevesebb olyan munka fut a gépeken, melynek a módosított határideje nem nagyobb d'_i-nél.
Ha nem létezik ilyen t_0, készen vagyunk, ugyanis legfeljebb $m\lfloor d'_i \rfloor + 1$ munka készülhet el d'_i időpontig. Így nincs olyan ütemezés, melyben minden munka elkészül a módosított határidőkre. Tehát feltehető, hogy t_0 létezik. A listás ütemezés szabályából, illetve i és t_0 választásából következően minden olyan t_0 után ütemezett j munkának, melyre $d'_j \leq d'_i$, van egy elődje, mely t_0 időpontra ütemezett. (Különben t_0 időpontra ütemeztük volna). Két eset lehetséges:

Amennyiben $t_0 = \lfloor d'_i \rfloor - 1$: legyen $k \prec i$ azon elődje i-nek, mely t_0 időpontra lett ütemezve. (Ilyen természetesen van, különben i-t is t_0 időpontra ütemezhettük volna). Mivel k elkészül $\lfloor d'_i \rfloor - 1$ időpillanatra, és a a módosított határidők definíciója szerint $d'_k \leq d'_i - 1 < \lfloor d'_i \rfloor$, ezért k szintén késik a módosított határidejéhez képest, ami ellentmond i választásának.

Amennyiben $t_0 < \lfloor d'_i \rfloor - 1$: $t_0 + 1$ időpontban pontosan m olyan munka lett ütemezve, amelynek a módosított határideje nem nagyobb d'_i-nél, és mindegyiknek van egy t_0-ban kezdődő elődje. Ez viszont ellentmond a befenyő struktúrának. □

Lemma 2.10. *Ha a d határidőkhöz a d' módosított határidők tartoznak, és $l \in \mathbb{R}$, akkor a $d + l$ határidőkhöz a $d' + l$ módosított határidők tartoznak.*

Bizonyítás:
A gyökérre nyilvánvaló az állítás, mert ott megegyezik a módosított határidő az eredetivel. A módosított határidők definíciójából indukcióval belátható az állítás. □

Tétel 2.11. *Az algoritmus optimális ütemezést talál a*

$$P|intree, p_j = 1|L_{max} \textit{ feladatra.}$$

Bizonyítás:
Legyen L^*_{max} az optimum. Pontosan akkor létezik S ütemezés, melyre minden $j \in J$-re $C_j - d_j \leq L^*_{max}$, ha a $d_j + L^*_{max}$ határidőre létezik késés nélküli ütemezés. Ha létezik ilyen ütemezés, akkor tudjuk, hogy az algoritmusunk ilyet talál, és az előző lemma miatt az algoritmusunk ugyanazt az ütemezést adja az eredeti határidőkre is. □

Megjegyzés: Amennyiben az eredeti határidők megegyeznek, a fent bemutatott algoritmus megegyezik az előző, 2.4.1 alfejezetben bemutatott szint-algoritmussal.

2.5. $P|outtree, r_j, p_j = 1|\sum C_j$

E feladatra Brucker, Hurink és Knust [5] adtak polinomiális megoldást, mégpedig a feladat két relaxációját vizsgálták, majd ezt felhasználva adtak egy algoritmust az eredeti problémára:
Feltesszük, hogy a rendelkezésre álló idők egészek, azaz $r_j \in \mathbb{Z}^+$ és kompatibilisek a megelőzési relációkkal, azaz $r_i + 1 \leq r_j$, ha $i \prec j$.
Tekintsük két speciális esetét a problémának: Az egyik, mikor eltekintünk a megelőzési feltételektől, a másik pedig mikor feltesszük, hogy annyi gépünk van, mint munkánk, azaz $m = n$.

2.5.1. $P|p_j = 1, r_j|\sum C_j$

Legyenek $\tilde{r}_1 < \tilde{r}_2 < ... < \tilde{r}_k$ a különböző rendelkezésre álló idők, és legyen $\tilde{r}_{k+1} := \infty$. Legyen S_ν azon munkák halmaza, melyek rendelkezésre álló ideje $\tilde{r}_\nu$. A probléma optimális ütemezését adja a következő algoritmus:

Algoritmus I.

1. $S := \emptyset$;
2. $\nu :=$ 1-től k-ig hajtsuk végre:
   - $t := \tilde{r}_\nu$; $S := S \cup S_\nu$;
   - Amíg $t < \tilde{r}_{\nu+1}$ és $S \neq \emptyset$ legyen:
     - $m_t := \min\{m, |S|\}$;
     - ütemezzünk m_t darab $S_t \subseteq S$ munkát t időpontra;
     - $S := S \backslash S_t$; $t := t + 1$;

Megjegyzés: minden olyan pillanatban, mikor m_t nem definiált, legyen $m_t := 0$.

2.5.2. $Pn|p_j = 1, r_j|\sum C_j$

Ebben az esetben minden S_ν-beli munkát a megjelenési időpontjában, azaz $\tilde{r}_\nu$ időpontban ($\nu = 1, ..., k$), gépre rakhatunk. Ez az ütemezés megengedett, (mivel feltettük, hogy a rendelkezésre álló időpontok kompatibilisek a megelőzési feltételekkel), és természetesen optimális is lesz.

Az eredeti feladat megoldása
Legyen

$$\hat{m}_t := \begin{cases} |S_\nu| & \text{ha } t = \tilde{r}_\nu, \\ 0 & \text{különben.} \end{cases}$$

1. futassuk le az I. algoritmust,
2. majd toljuk balról jobbra a munkákat következőképpen:
   A következő két feltétel végig igaz lesz:
   (a) a megelőzési feltételek nem romlanak el
   (b) $\sum_{\nu \leq \tau} \hat{m}_\nu \geq \sum_{\nu \leq \tau} m_\nu$ minden τ időpillanatban.

Kezdetben mindkét feltétel igaz. Tegyük fel, hogy valamely ν -re $\hat{m}_\nu < m_\nu$. Ugyanis, ha nincs ilyen ν érték, akkor $n = \sum \hat{m}_\nu \geq \sum m_\nu = n$, azaz minden ν-re $\hat{m}_\nu = m_\nu$ és ezzel készen vagyunk. Legyen t a legkisebb olyan index, melyre $\hat{m}_t < m_t \leq m$. Ekkor (b) miatt létezik $t' < t$, melyre $\hat{m}_{t'} > m_{t'}$ és $\hat{m}_\tau = m_\tau$ minden $t' < \tau < t$-re. Sőt, m_t definíciója miatt $m_\tau = m$ minden $t' < \tau < t$, ezért $\hat{m}_{t'} \geq \hat{m}_{t'+1} = \hat{m}_{t'+2} = ... = \hat{m}_{t-1} > \hat{m}_t$. Eltolhatunk egy munkát $[\tau - 1, \tau]$-ból, $[\tau, \tau+1]$-be, minden $\tau = t, t-1, ..., t'+1$-re úgy, hogy (a) igaz maradjon, mivel $[\tau-1, \tau]$-ben maximum egy munkával van több, mint $[\tau, \tau+1]$-ben és minden munkának csak egy elődje van. Tehát mindig található előd nélküli munka $[\tau, \tau+1]$-ben, amit elrakhatunk. Tehát (b) teljesül, mert $\sum_{\nu<t'} \hat{m}_\nu \geq \sum_{\nu<t'} m_\nu$ és $\hat{m}_\nu > m_\nu$. Ebből következik, hogy miután egy munkát elmozdítottunk $[t', t'+1]$-ból $[t'+1, t'+2]$-be, (b)-ben az egyenlőtlenség még mindig igaz marad $\tau = t'$-re, és $\tau > t'$-re szintén, mivel $\nu > t'$-re a $\hat{m}_\nu$ változatlan maradt.

Egy lépésben a $\max\{0, m_\nu - \hat{m}_\nu\}$ értéke $\nu = t$-re 1-gyel nő, illetve a többi ν-re nem változik, azaz a $\sum_\nu \max\{0, m_\nu - \hat{m}_\nu\}$ minden lépésben eggyel nő. Mivel az elején $\sum_\nu \max\{0, m_\nu - \hat{m}_\nu\} \leq \sum_\nu m_\nu = n$, ezért legfeljebb n iteráció után elérjük az eredeti probléma optimális értékét. Mivel minden iteráció $\mathcal{O}(n)$ időben lefuttatható, a problémánk megoldható $\mathcal{O}(n^2)$ időben.

2.6. $P|tree, p_j = 1|C_{max}$ és a matroid-fedés probléma

Már láttuk a 2.4.1 fejezetben erre a feladatra Hu szintalgoritmusát. Legyen $J_l := \{i \in J : l(i) \leq l\}$ és l^* legyen a maximális szint. Ekkor Hu [13] a következő minimax tételt látta be:

$$\max_{0 \leq l \leq l^*} \left\lceil \frac{|J_l|}{D - l} \right\rceil = \{\text{ a szükséges min. gépszám, hogy D időre minden munka elkészüljön}\}$$

Megjegyezzük, hogy $D > l^*$, különben a probléma nem megoldható.
Edmonds [7] a matroid bázisfedési-problémára adott egy teljesen hasonló minimax tételt:

$$\max_{\emptyset \neq S \subseteq E} \left\lceil \frac{|S|}{r(S)} \right\rceil = \{\text{min. bázis, melyeknek uniója } E\},$$

ahol r a matroid rangfügvénye. Marcotte és Trotter [17] megmutatta, hogy e hasonlóság nem véletlen, azaz ez a két minimax-tétel egymásból levezethető.

Elsőnek nézzük meg, miként vezethető le Hu állításából a matroid bázisfedésére adott minimax-tétel. Ehhez definiáljunk egy transzverzális matroidot:
Egy megengedett ütemezésben egy gép legfeljebb $D - l$ munkát teljesíthet azon munkák közül, melyeknek szintje legalább l. Ugyanis amennyiben egy gépen legalább $D - l$ munka fut a J_l munkahalmazból, akkor van olyan l-szintű munka, mely $D - l$ idő után kerül a gépre, ami azt eredményezi, hogy a befenyő gyökere – a 0-szintű munka – D idő után kerül csak a gépekre.
Nevezzük a munkák egy S halmazát *megengedett gépelőzmény*nek, ha $|S \cap J_l| \leq D - l$, minden $0 \leq l \leq l^*$-re. Ezek lesznek majd a transzverzális matroid függetlenjei. Hogy ezt megmutassuk, definiáljunk egy páros gráfot a következőképp: $G := (J \cup T; E)$, ahol $J = \{1, ..., n\}$, azaz minden munkának megfeleltetünk egy pontot a gráfban, és hasonlóan minden időegységhez is rendeljünk egy pontot: $T = \{1, ..., D\}$. A gráf élhalmaza legyen: $E = \{(i, t) : D - l(i) \geq t\}$.

Tétel 2.12. *A megengedett gépelőzmények ennek a transzverzális matroidnak a függetlenjeit adják.*

Bizonyítás:
Definíció szerint egy megengedett gépelőzmény kijelöl néhány l^*-szintű munkát az $1, ..., D - l^*$ időintervallumokba, majd kijelöl néhány l^* illetve $l^* - 1$ szintű munkát a $1, ..., D - l^* + 1$ időintervallumokba, ... és így tovább. Ezen munkák időintervallumokba való beosztása a G gráf egy párosítását adja, melynél a párosításbeli J-pontok pont a megengedett gépelőzmény pontjai. Fordítva, egy párosítás J-pontjai meghatároznak egy munkahalmazt, melyben legfeljebb $D - l^*$ darab l^* szintű munka szerepel, legfeljebb $D - l^* + 1$ darab munka szerepel l^* és $l^* - 1$ szintekből, ... és így tovább. így a munkák egy megengedett gépelőzményt alkotnak.

Jelöljük J_l J-re vett komplementerét $\bar{J}_l$-lel. A matroid rangfüggvény tulajdonságából és a gráf definíciójából könnyen belátható az alábbi két egyenlőtlenség:

$$r(S) \leq |S|,$$
$$r(S) \leq |S \cap \bar{J}_l| + D - l,$$

Innen

$$r(S) \leq \min\{|S|; |S \cap \bar{J}_l| + D - l; l = 1, ..., l^*\} \quad (1)$$

Sőt, ez egyenlőséggel teljesül, s hogy ezt belássuk, tekintsük a következő definíciót:
Legyen l, $(0 \leq l \leq l^*)$, S-re nézve *kritikus index*, amennyiben létezik olyan S megengedett gépelőzmény, melyre $|S \cap J_l| = D - l$, azaz az $1, 2, ..., D - l$-időpontoknak megfelelő T-pontok szintén bepárosíthatók $S \cap J_l$-be.

Lemma 2.13. *Ha nem létezik kritikus index, akkor* $r(S) = |S|$*, ha létezik, akkor legyen* k *minimális kritikus index. Ekkor* $r(S) = |S \cap \bar{J}_k| + D - k$ *azaz* $S \cap J_k$*-ből* $(D - k)$ *pont bepárosítható* $1, ..., (D - k)$*-ba.*

Bizonyítás:
Amennyiben nem létezik S-re nézve kritikus index, akkor l^* sem kritikus. Mivel S megengedett gépelőzmény, a $S \cap J_{l^*}$-beli J-pontok bepárosíthatók a $1, ..., |S \cap J_{l^*}|$ pontokba, ahol $|S \cap J_{l^*}| < D - l^*$. Mivel $l^* - 1$ sem kritikus, ezért $S \cap (J_{l^*-1} \backslash J_{l^*})$-beli J-pontok (azaz az $l^* - 1$ szintű pontok), $|S \cap J_{l^*}| + 1, ..., |S \cap J_{l^*-1}|$ T-pontokba párosíthatók,
(ahol $|S \cap J_{l^*-1}| < D - l^* + 1$). Folytatva ezt a gondolatmenetet, J-pontok egy párosítását kapjuk, melyek mind S-beliek, tehát $r(S) = |S|$.
Most tegyük fel, hogy létezik kritikus index. Mivel k kritikus, tudjuk, hogy $D - k$ darab J-pont az $S \cap J_k$ halmazból bepárosítható az $1, ..., D - k$ ponthalmazba. Ha $k = 0$, akkor $\bar{J}_k = \emptyset$, tehát $r(S) = D - k$.
Ha $k > 0$, akkor $0, ..., k - 1$ nem kritikus, akkor a fenti gondolatmenet szerint $S \cap \bar{J}_k$ ponthalmaz bepárosítható $D - k + 1, ..., D$ pontokba, tehát $r(S) = |S \cap \bar{J}_k| + D - k$. Ezzel igazoltuk a lemma állítását. □

Tehát a megengedett gépelőzmények egyenlőséggel teljesítik (1)-t, így valóban a matroid függetlenjeit adják. Innen már könnyen beláthatjuk a fentebb kimondott minimax egyenlőtlenséget. Legyen $\emptyset \neq S \subseteq J$, ekkor vagy nem létezik kritikus index, s ekkor

$$\frac{|S|}{r(S)} = \frac{|S|}{|S|} = 1 \leq \left\lceil \frac{J_l}{r(J_l)} \right\rceil$$

illetve, ha létezik, akkor

$$\frac{|S|}{r(S)} = \frac{|S \cap \bar{J}_l| + |S \cap J_l|}{|S \cap \bar{J}_l| + D - l} \leq \frac{|S \cap J_l|}{D - l} \leq \frac{|J_l|}{D - l} \leq \frac{|J_l|}{r(J_l)}$$

Tehát innen kapjuk, hogy

$$\max_{0 \leq l \leq l^*} \left\lceil \frac{|J_l|}{D - l} \right\rceil = \{\text{min. gépelőzmények száma, melynek uniója } J\}$$

A fordított levezetéshez legyen

$$M = \max_{0 \le l \le l^*} \lceil |J_l|/(D - l^*) \rceil,$$

majd Hu által adott szint-algoritmust használva jelöljük ki a munkákat M gépre: amint egy gép szabad lesz, a megmaradó munkák közül azt ütemezzük rá, melynek a szintje legnagyobb. Megmutatjuk, hogy így az összes J_l-beli munka $D - l$ időre elkészül, (minden $l = l^*, l^* - 1, ..., 0$). Az $l = 0$ eset adja majd a kérdéses mini-max tételt. A bizonyításunk induktív lesz. Mivel $M \ge \lceil |J_{l^*}|/(D - l^*) \rceil$, következik, hogy $M(D - l^*) \ge |J_{l^*}|$, tehát minden l^* szintű munka elkészül $D - l^*$ időre. Legyen $l^* > l \ge 0$, és tegyük fel, hogy minden J_{l+1} szintű munka elkészül $D - (l+1)$ időre. Tegyük fel, hogy van olyan l szintű munka, mondjuk j, mely $D - l$ idő után kerül a gépekre. Mivel $M(D - l) \ge |J_l|$, s j csak később kerül a gépekre, ezért egy $[t - 1, t]$ időben ($0 < t \le D - l$) van munka nélküli gép, vagy van olyan munka a gépeken, melynek szintje kisebb, mint $l(j)$. Mivel J_{l+1} munkái elkészülnek $D - l - 1$ időre, és j munka $D - l$ idő után lett ütemezve, a szint-algoritmus M darab l-szintű munkát ütemez a $[D - l - 1, D - l]$ időintervallumba. Azaz $t < D - l$. A szint-algoritmus választási szabálya szerint minden olyan l-szintű munkának, mely $[D - l - 1, D - l]$ alatt fut, van egy elődje a korábbi $[t - 1, t]$ időintervallumban. De ebben az időben legfeljebb $M - 1$ munka lehet l-nél nagyobb szintű. Azaz M darab l szintű munkának, melyek mindegyikének van elődje, összesen legfeljebb $M - 1$ darab elődje van. Ez ellentmond a megelőzési gráfunk befenyő szerkezetének. Tehát minden l-re igaz az állítás, így $l = 0$-ra is, így beláttuk Hu minimax tételét. $\square$

3. Uniform gépek

3.1. $Q|prec|C_{max}$

Láttuk, hogy ha azonos sebességű gépeink vannak, akkor egy egyszerű listás ütemezés 2-approximációs algoritmust ad, és a mai napig nincs ennél jobb faktorú algoritmus. Uniform gépek esetén sokáig a legjobb ismert algoritmus egy $\mathcal{O}(\sqrt{m})$-approximációs algoritmus volt [14], majd Chudak és Shmoys adtak $\mathcal{O}(\log m)$-approximációs algoritmust [6]:
A párhuzamos gépeken való listás ütemezés azért volt célrevezető, mert találtunk a munkáknak egy olyan láncát, melyre csak akkor volt szabad gép, ha abban az időpillanatban a lánc valamely munkája éppen valamely gépen futott. S ez két alsó korlátot is szolgáltatott a célfüggvényünkre. Adná magát az ötlet, hogy az uniform gépek esetében is valami hasonlóval próbálkozzunk, azonban sajnos a lánc összfutási idejéből adódó alsó korlát nem marad igaz, csak abban az esetben, ha a leggyorsabb gépre ütemeznénk az összes munkáját, ami persze nem biztosított. Ellenben, ha tudnánk előre, hogy mely munkát milyen sebességű gépre ütemezzünk, akkor pedig nem lehetünk biztosak abban, hogy minden gép foglalt lesz, így a második alsó korlát nem lesz igaz. Azonban, ha megengedjük, hogy némely gép munka nélkül legyen, s helyette feltesszük, hogy legalább egy adott sebességgel futó összes gép foglalt, egy egész jó becsléshez juthatunk.

Legyen m_k azon gépek száma, melyek $\bar{s}_k$ sebességgel futnak, ahol $k = 1, ..., K$; legyen továbbá $\bar{s}_1 > \bar{s}_2 > ... > \bar{s}_K$. Tegyük fel, hogy minden $j \in J$ munkához adott azon gép – $k(j))$ – indexe, amelyre ütemezzük. Ekkor az $\bar{s}_k$ sebességű gépek összműködési ideje:

$$D_k = \frac{1}{m_k} \sum_{j:k(j)=k} \frac{p_j}{\bar{s}_k}.$$

Legyen

$$C = \max_{\mathcal{C} \text{ lánc}} \sum_{j \in \mathcal{C}} \frac{p_j}{\bar{s}_{k(j)}}.$$

Legyen a sebességen alapuló listás ütemezésben a munkák rendre $1, 2, ..., n$ sorba rendezve. Amint egy munka befejeződik, mindazon gépekre ütemezzünk munkát, ahol épp nincsen. A munka nélküli gépeket sorba rakjuk. Egy $\bar{s}_k$ sebességű gépre elsőnek azon még nem szerepelt j munkát választjuk, amelyre $k(j) = k$, és amelynek már az összes elődje végzett. Ha ilyen munka nem létezik, akkor a gép egészen addig munka nélkül marad, amíg a következő munka nem végez.

Tétel 3.1. *Ha minden munkára meghatározott a $k(j)$ index, $(j = 1, ..., n)$, akkor a sebességen alapuló listás ütemezési algoritmus megoldására*

$$C_{max} \leq C + \sum_{k=1}^{K} D_k.$$

Bizonyítás:
A tétel bizonyítása általánosítása Graham [11] bizonyításának. Felosztjuk a $[0, C_{max}]$ időintervallumot $K+1$ darab τ_k téglára, $(k = 0, 1, ..., K)$:
Legyen j_1 az a munka, ami C_{max} időre készül el. Készítsük el a következő láncot, amely j_1 munkával ér véget: minden $t = 1, ..., n$-re legyen j_{t+1} a j_t azon elődje, amely az ütemezésben legkésőbb ér véget. Ha j_t-nek nincs elődje, akkor $\mathcal{C}$ lánc teljes. Legyen τ_0 azon időintervallum, ami alatt $\mathcal{C}$ munkái futnak. Jelöljük $\tau_{k(j)}$-vel azt az intervallumot, ami akkor végződik, mikor a $j \in \mathcal{C}$ munka kezdődik. Így felosztottuk az időintervallumot 0-tól C_{max}-ig téglákra.
Elsőnek is figyeljük meg, hogy τ_0 hossza maximum C, mivel az algoritmusunk minden j munkát egy $\bar{s}_{k(j)}$ sebességű gépre ütemez, $(j = 1, ..., n)$. Megfigyelhetjük azt is, hogy nincs $\bar{s}_k$ sebességű álló gép a τ_k intervallumban. Ha τ_k intervalllum hossza t_k, akkor ez idő alatt $\bar{s}_k m_k t_k$ egységnyi munkát kell teljesítenie ezeken a gépeken, sőt összesen csak $\sum_{j:k(j)=k} p_j$ egységet kell teljesítenie az $\bar{s}_k$ sebességű gépeknek, ezért

$$t_k \leq \frac{1}{m_k \bar{s}_k} \sum_{j:k(j)=k} p_j = D_k, \; k = 1, ..., K.$$

Ezzel bebizonyítottuk a tétel állítását. □

Szeretnénk kijelölni a munkákat a gépekhez oly módon, hogy a felső korlát, $C + \sum_{k=1}^{K} D_k$ ne legyen túl nagy. Bemutatjuk a feladatunk egy relaxációját, amely kiszámolja, hogy mely munkát milyen sebességű gépre ütemezzünk, hogy az összidő D legyen.
Vezessük be a következő változókat: legyen x_{kj} értéke 1, ha j munka egy $\bar{s}_k$ sebességű gépen fut, és 0, ha nem, $(k = 1, ..., K$ és $j = 1, ..., n)$ továbbá jelölje C_j változó értéke azt az időpillanatot, mikor a j munka véget ér,$(j = 1, ..., n)$. Ez utóbbiak természetesen nem feltétlenül egészek.
Ekkor a következők teljesülnek a változókra:

- minden munka fut valamely gépen, ezért (2)
- bármely $\bar{s}_k$-ra, az azonos sebességű gépek összemunkaideje nem haladhatja meg D-t, ezért (3)

- egyetlen munka sem fejeződhet be az átfutási idejénél hamarabb (4)
- minden $j' \prec j$ relációra, $C_j - C_{j'}$ érték nem lehet kisebb, mint a j futási ideje, így kapjuk (5)-t
- amennyiben az ütemezés hossza D, úgy minden munka véget ér addigra, innen kapjuk (6)-t
- s végül (7) biztosítja majd a megfelelő változók egészértékűségét.

Tehát:

$$\sum_{k=1}^{K} x_{kj} = 1,\ j = 1, ..., n, \tag{2}$$

$$\frac{1}{m_k \bar{s}_k} \sum_{j=1}^{n} p_j x_{kj} \leq D,\ k = 1, ..., K, \tag{3}$$

$$\sum_{k=1}^{K} \frac{p_j}{\bar{s}_k} x_{kj} \leq C_j,\ j = 1, ..., n, \tag{4}$$

$$\sum_{k=1}^{K} \frac{p_j}{\bar{s}_k} x_{kj} \leq C_j - C_{j'},\ \text{ha } j' \prec j \tag{5}$$

$$C_j \leq D,\ j = 1, ..., n, \tag{6}$$

$$x_{kj} \in \{0, 1\},\ k = 1, ..., K,\ j = 1, ..., n. \tag{7}$$

Ekkor a relaxáció a következő: Minimalizáljuk D-t, a fenti (2)-(7) feltétellel. Ezen vegyes-egészértékű lineáris programozási feladat optimális megoldása alsó korlát lesz az optimális ütemezés hosszára, C^*_{max}-ra. A feladat minden megoldására a sebességen alapuló listás ütemezés egy $(K+1)D$ hosszúságú megoldást ad: (5) és (6) feltételek következtében minden lánc hossza legfeljebb D, és a (3) feltétel miatt minden k-ra, az $\bar{s}_k$ sebességű gépek D_k összműködési ideje legfeljebb D, $(k = 1, ..., K)$. Valójában 3.1 tétel többet állít: ezen korlát eléréséhez nem szükséges, hogy olyan egészértékű megoldásokat kapjunk, ami (3)-t kielégíti; legyen igaz a következő:

$$\sum_{k=1}^{K} \frac{1}{m_k \bar{s}_k} \sum_{j=1}^{n} p_j x_{kj} \leq KD \tag{8}$$

Az algoritmusunk felhasználja azt a tényt, hogy ez a gyengébb egyenlőtlenség elegendő lesz. Helyettesítsük a (7)-es feltételünket a következővel:

$$x_{kj} \geq 0,\ k = 1, ..., K,\ j = 1, ..., n. \tag{9}$$

A lineáris programozási feladat a következő lesz: minimalizáljuk D-t az (2), (3), (4), (5), (6) és (9) feltételek mellett. Jelöljük ezt a feladatot LP-vel. Ez a feladat polinom időben megoldható, legyen az optimális megoldás $\bar{x}_{kj}$ $(k = 1, ..., K,\ j = 1, ..., n)$, és $\bar{C}_j$ $(j = 1, ..., n)$, és az optimum $\bar{D}$, ami természetesen alsó korlátja lesz C^*_{max}-nak. Az LP optimális megoldását használjuk fel, hogy kijelöljük a munkákat a gépekre. Minden j munkára, $(j = 1, ..., n)$, kiszámoljuk az –átlagos– $\bar{p}_j$ futási idejét az LP által kijelölt gépeken:

$$\bar{p}_j = \sum_{k=1}^{K} (p_j/\bar{s}_k) x_{kj},\ j = 1, ..., n.$$

Legyen továbbá B_j azon (rossz) gépek indexhalmaza, melyek túl lassúak a j munkához:

$$B_j = \{k : p_j/\bar{s}_k > (\sqrt{K} + 1)\bar{p}_j\}.$$

Minden j munkára, $(j = 1, ..., n)$, legyen $k(j)$ az a $k \notin B_j$ index, melyre $p_j/(\bar{s}_k m_k)$ minimális, azaz melyre $\bar{s}_k m_k$ a legnagyobb. Megmutatjuk, hogy ezt a $k(j)$-t használva $C + \sum_{k=1}^{K} D_k \leq 2(K+1)\bar{D}$ lesz. S mivel $\bar{D} \leq C^*_{max}$, ezért a sebességen alapuló listás ütemezésünk egy $2(K+1)$-approximációs algoritmust ad.

Lemma 3.2. *Az algoritmus által kijelölt $k(j)$ értékekre, $(j = 1, ..., n)$, és bármely $\mathcal{C}$ láncra*

$$\sum_{j \in \mathcal{C}} p_j/\bar{s}_{kj} \leq (\sqrt{K} + 1)\bar{D}.$$

Bizonyítás:
Legyen $\mathcal{C}$ lánc a következő: $j_1 \prec j_2 \prec ... \prec j_r$. Elsőnek belátjuk, hogy

$$\sum_{j \in \mathcal{C}} \bar{p}_j \leq \bar{D}. \tag{10}$$

S mivel minden j munkához úgy választottuk a $k(j)$ indexet, hogy legfeljebb $(\sqrt{K} + 1)\bar{p}_j$ időegység alatt fut a gépen, ez elég is lesz a lemma állításához. Mivel $\bar{x}$ kielégíti (4)-t, $\bar{p}_{j1} \leq \bar{C}_{j1}$. Továbbá, $\bar{x}$ kielégíti (5)-t is, és így, mivel $j_{i-1} \prec j_i, i = 2, ..., r$, ezért $\bar{p}_j \leq \bar{C}j_i - \bar{C}_{j_{i-1}}, (i = 2, ..., r.)$. Ha összeadjuk ezeket a feltételeket, kapjuk, hogy

$$\sum_{j \in \mathcal{C}} \bar{p}_j \leq \bar{C}_{j_r}.$$

Végül (6) miatt $\bar{C}_{j_r} \leq \bar{D}$, és ezért (10) teljesül. $\square$

Lemma 3.3. *Legyen* $\hat{x}_{kj} = 1$ *ha* $k = k(j)$, $(j = 1, ..., n)$ *és legyen* $\hat{x}_{kj} = 0$ *különben. Ekkor*

$$\sum_{k=1}^{K} \frac{1}{m_k \bar{s}_k} \sum_{j=1}^{n} p_j \hat{x}_{kj} \leq \frac{\sqrt{K}+1}{\sqrt{K}} \bar{D}.$$

Bizonyítás:
Tekintsük a problémát egy szimplex felett:

$$\min \sum_{j=1}^{N} c_j y_j$$

$$\sum_{j=1}^{N} h_j = 1, y_j \geq 0, \ j = 1, ..., N.$$

Ebben az esetben könnyen ellenőrizhető, hogy létezik egy optimális y^* megoldás, ami egészértékű: legyen $c_{j^*} = \min_j c_j$ és legyen $y^*_{j^*} = 1$, és összes többi komponense 0.
Hasonlóan, $\hat{x}$ optimális megoldása a következő lineáris programnak:

$$\min \sum_{k=1}^{K} \frac{1}{m_k \bar{s}_k} \sum_{j=1}^{n} p_j x_{kj} \tag{11}$$

$$\sum_{k=1}^{K} x_{kj} = 1, \ j = 1, ..., n, \tag{12}$$

$$x_{kj} = 0, \ \text{ha } k \in B_j, \ j = 1, ..., n, \tag{13}$$

$$x_{kj} \geq 0, \ k = 1, ..., K, \ j = 1, ..., n. \tag{14}$$

Célunk egy olyan $\tilde{x}$ megengedett megoldást előállítani, melynél a célfüggvény értéke legfeljebb $\frac{\sqrt{K}+1}{\sqrt{K}}\bar{D}$. Lin és Vitter [16] technikáját használva kapjuk LP optimális $\bar{x}$ megoldását. Ekkor legyen $\alpha_j := \sum_{k \notin B_j} \bar{x}_{kj}$, $(\alpha_j > \frac{\sqrt{K}}{\sqrt{K}+1})$. Majd legyen

$$\tilde{x}_{kj} := \begin{cases} 0 & \text{ha } k \in B_j, \\ \frac{\bar{x}_{kj}}{\alpha_j} & \text{különben.} \end{cases}$$

Könnyen ellenőrizhető, hogy $\tilde{x}$ megengedett megoldása a (12)-(14)-nek. Továbbá $\tilde{x}_{kj} \leq \frac{\sqrt{K}+1}{\sqrt{K}}\bar{x}_{kj}$, (minden $k = 1, ..., K, j = 1, ..., n$). Tehát (3)-ből kapjuk, hogy

$$\frac{1}{m_k \bar{s}_k} \sum_{j=1}^{n} p_j \tilde{x}_{kj} \leq \frac{\sqrt{K}+1}{\sqrt{K}} \bar{D}, \; k = 1, ..., K. \tag{15}$$

Ebből következik, hogy $\tilde{x}$ megengedett megoldáshoz tartozó célfüggvény értéke, (11), legfeljebb $\frac{\sqrt{K}+1}{\sqrt{K}} K\bar{D}$. Mivel $\hat{x}$ optimális megoldás, ezért a hozzátartozó célfüggvényérték szintén legfeljebb $\frac{\sqrt{K}+1}{\sqrt{K}} K\bar{D}$. □

Kombinálva ezt a lemmát a 3.1-es tétellel, kapjuk a következő eredményt:

Tétel 3.4. *A kijelölő algoritmust kombinálva a sebességen alapuló listás ütemezéssel egy* $(K+2\sqrt{K}+1)$*-approximációs algoritmust kapunk a* $Q|prec|C_{max}$ *problémára, ahol* K *a különböző gépsebességek számát jelöli.*

Következmény 3.5. *A kijelölő algoritmus a sebességen alapuló listás ütemezéssel kombinálva egy olyan megoldást ad, melyre a* C_{max} *értéke* $\bar{D}$*, az* LP *lineáris programozási feladatnak optimális megoldásának* $(K+2\sqrt{K}+1)$ *hányadán belül marad.*

Megmutatjuk, hogy az általános problémát hogyan vezessük vissza arra az esetre, mikor legfeljebb $\log m$ különböző sebességű gép van (azaz $K = \log m$). Közben az LP feladat optimuma legfeljebb a négyszeresére változhat, s így a 3.5-os következmény értelmében egy $\mathcal{O}(\log m)$-approximációs algoritmust kapunk. Feltehetjük, hogy az m gép sebessége s_i, $(i = 1, ..., m)$, K különböző sebességből áll, ahol $1 = \bar{s}_1 > \bar{s}_2 > ... > \bar{s}_K$ (természetesen a $K = m$ legrosszabb eset is előfordulhat). Ekkor az algoritmus két lépésből áll:

- Elsőnek kiszórjuk azokat a gépeket, melyek a sebessége a leggyorsabb gép sebességéhez viszonyítva legalább $c(m)$-szer lassabbak, ahol $c(m)$ egy m-től függő konstans.
- Minden megmaradt gép sebességét lekerekítjük a legközelebbi $1/2$ hatványig. (Vagy tetszés szerinti $(1/\alpha)$ hatványig). Így egyik gép sem lesz gyorsabb, mint eredetileg volt.

Legyen k_0 az a legkisebb index, ami nem nagyobb $1/m$-nél, azaz, $\bar{s}_k \leq 1/m$, $(k = k_0 + 1, ..., K)$. Nevezzünk „gyorsnak" egy gépet, ha sebessége nagyobb, mint $1/m$, a többi gép legyen „lassú". Bármely munka, mely lassú gépre van

ütemezve, ütemezhető egy gyors gépre is:
Ugyanis nem több, mint m lassú gép van, ezért a lassú gépek összsebessége kevesebb, mint 1, ami épp s_1. Ez azt jelenti, hogy bármely 1 sebességű gép legfeljebb kétszer annyi idő alatt elvégezheti a lassú gépek munkáját is. Azaz LP feladatunk bármely megengedett $\bar{x}_{kj}$, $(k = 1, ..., K;\ j = 1, ..., n)$, $\bar{C}_j$, $(j = 1, ..., n)$, és $\bar{D}$ megoldásához konstruálhatunk egy $\tilde{x}$ megengedett megoldást, melynek célfüggvény értéke legfeljebb $2\bar{D}$, és csak a gyors gépeket használja. Legyen $\tilde{D} = 2\bar{D}$, és minden j munkára, $(j = 1, ..., n)$, legyen

$$\tilde{x}_{1j} = \bar{x}_{1j} + \sum_{k_0}^{K} \bar{x}_{kj},$$

$$\tilde{x}_{kj} = \bar{x}_{kj},\ (k = 2, ..., k_0),\ \text{és } \tilde{C}_j = \bar{C}_j.$$

Ez egy olyan megengedett megoldása lesz az LP feladatnak, mely csak a gyors gépeket használja.
Tehát feltehetjük, hogy bármely gépsebességre: $\bar{s}_k \in (1/m, 1]$. Minden i gépre, kerekítsük le a sebességét a legközelebbi 1/2 hatványig; azaz minden gépet, melynek sebessége $\bar{s}_i \in (2^{-k}, 2^{-k+1}]$ helyettesítsük egy olyan géppel, melynek sebessége 2^{-k}. Ekkor a $\tilde{x}_{kj}$, $(k = 1, ..., k_0;\ j = 1, ..., n)$, $2\tilde{C}_j$, $(j = 1, ..., n)$ és $2\tilde{D} = 4\bar{D}$ megengedett megoldása lesz ennek a lekerekített LP feladatnak. Kerekítés után legfeljebb $\log m$ különböző sebesség lesz, tehát egy $4(\log m + 2\sqrt{\log m} + 1)$ approximációs algoritmust kaptunk a $Q|prec|C_{max}$ feladatra.
Megjegyzés:
Persze tovább optimalizálhatunk, amennyiben megkeressük a legjobb $c(m)$ konstans, ill. azt a hatványt, melyre a meghagyott gépek a sebességét érdemes kerekíteni. Azaz például, ha az $1/(\beta m)$-nél lassabb gépeket elhagyjuk, és minden más sebességet a $(\alpha^{-k}, \alpha^{-k+1}]$ intervallumban lekerekítünk α^{-k}-ra, (ahol $\alpha > 1$, és $\beta \geq 1$), abban az esetben a különböző sebességek száma legfeljebb $\log_\alpha(\beta m)$, és a kerekítés az LP feladatunk célfüggvényértékét legfeljebb $\alpha(1 + 1/\beta)$-szeresére növelhette. Azaz a legjobb α és β választás mellett a következő tételt kapjuk:

Tétel 3.6. *Ha K a különböző gépsebességek száma, akkor létezik egy $\min\{K + 2\sqrt{K} + 1, 1.89 \log m + \Theta(\sqrt{\log m})\}$-approximációs algoritmus a $Q|prec|C_{max}$ feladatra.*

3.2. $Q|prec|C_{max}$ mégegyszer

Fentebb bemutattunk a problémára egy lineáris programozással kombinált algoritmust, most nézzük meg Bender és Chekuri [1] által adott algoritmust, mely szintén $\mathcal{O}(\log m)$-approximációs, ellenben a lineáris programozási feladat helyett kombinatorikus módszert használ.
Az előző részhez hasonlóan hagyjuk el azokat a gépeket, melyeknek sebessége legfeljebb $1/m$-szerese a leggyorsabb gépnek, majd kerekítsük le a többi gép sebességét a legközelebbi $1/2$ hatványig.
Használni fogjuk a követező fogalmakat:
Legyen a részben rendezett halmaz egy $\mathcal{P} = (P_1, P_2, ...P_r)$ sorba rendezett láncokból álló partíciója *láncfelbontás*. Egy *maximális láncfelbontás* olyan, melyre P_1 a leghosszabb lánc, és $(P_2, ...P_r)$ egy maximális láncfelbontás a maradék munkahalmazon.
Megjegyzendő, hogy a maximális láncfelbontásnál nem mindegy a láncok sorrendje, használni fogjuk azt a tényt, hogy $|P_1| \geq |P_2| \geq ... \geq |P_r|$.
Legyen $\mathcal{P} = (P_1, P_2, ..., P_r)$ egy láncfelbontása a munkáknak. Legyen a $\mathcal{P}$-hez és $s_1 \geq s_2 \geq ... \geq s_m$ gépsebességekhez tartozó $L_{\mathcal{P}}$ mennyiség a következőképp adva:

$$L_{\mathcal{P}} = \max_{1 \leq j \leq \min\{r,m\}} \frac{\sum_{i=1}^{j} |P_i|}{\sum_{i=1}^{j} s_i}.$$

A definíció nevezőjében szereplő szumma nem a sebesség osztályokra megy, hanem a gépekre. Egy adott $j \leq m$-re a $\frac{\sum_{i=1}^{j} |P_i|}{\sum_{i=1}^{j} s_i}$ mennyiség a $\mathcal{P}$ felbontás első j láncának a j leggyorsabb gépen történő átlagos futási idejét jelöli.

Tétel 3.7. *Legyen $\mathcal{P} = (P_1, P_2, ..., P_r)$ a munkák megelőzési gráfjának egy láncfelbontása (gyakorlatban egy maximális láncfelbontása). Legyen $AL = (\sum_{j=1}^{n} p_j)/(\sum_{i=1}^{m} s_i)$ az átlagos futási idő. Ekkor*

$$C^*_{max} \geq \max\{AL, L_{\mathcal{P}}\}.$$

Sőt ez az alsó korlát azon ütemezésre is érvényes, mikor megszakításokat is megengedünk.

Bizonyítás:
A maximális mennyiség, amit a gépek egységnyi idő alatt el tudnak végezni, legfeljebb $\sum_{i=1}^{m} s_i$, mit pontosan akkor érünk el, ha minden gép foglalt. Ebből következik, hogy az összidő legalább $AL = (\sum_{j=1}^{n} p_j)/(\sum_{i=1}^{m} s_i)$, azaz: $C^*_{max} \geq AL$. Megmutatjuk, hogy minden $1 \leq j \leq m$-re:

$$C^*_{max} \geq \frac{\sum_{i=1}^{j} |P_i|}{\sum_{i=1}^{j} s_i}.$$

Tekintsük az első j láncot. Tegyük fel, hogy úgy módosítjuk a bemenő adatainkat, hogy csak az első j lánc munkái adottak. Könnyen ellenőrizhető, hogy e módosított feladat egy alsó korlátja, alsó korlát lesz az eredeti feladatra is. Mivel egy láncból egyszerre csak egy munka lehet a gépeken, ezért a módosított feladatra csak a leggyorsabb j gép lesz érdekes. A $(\sum_{i=1}^{j} |P_i|)/(\sum_{i=1}^{j} s_i)$ kifejezés, a módosított feladat átlagos futási ideje, amint az már megmutattuk alsó korlát lesz. Tehát a tételt ezzel bebizonyítottuk. Mivel az átlagos futási idő abban az esetben is alsó korlát lesz, mikor megszakításokat is megengedünk, ezért a tételünk ebben az esetben is igaz lesz. $\square$

Tétel 3.8. *A maximális lánc-kompozíció $\mathcal{O}(n^3)$ időben kiszámítható.*

Bizonyítás:
Elsőként készítsük el a gráf tranzitív-lezártját: minden pontjából futassunk le egy szélességi-keresést. (Ez összesen $\mathcal{O}(n^3)$ időben megtehető.) Majd keressük meg a leghosszabb láncot ebben a pontsúlyozott irányított gráfban, erre létezik $\mathcal{O}(n^2)$ időben futó algoritmus, melyet legfeljebb n-szer kell meghívnunk. Ebből már következik a feladat állítása. $\square$

Megjegyzendő, hogy amennyiben minden p_j azonos, akkor a futási idő $\mathcal{O}(n^2\sqrt{n})$.

Az approximációs algoritmus

Az approximációs algoritmus, melyet itt bemutatunk, a maximális láncfelbontáson alapul majd. Az algoritmus minden munkára megmondja, melyik gépen fusson, majd alkalmazzuk a sebességen alapuló listás ütemezést, amint azt az előző megoldásban is tettük.

ALGORITMUS

1. **Számoljuk** ki a munkák egy maximális lánc-kompozicióját, legyen ez $\mathcal{P} = (P_1, ..., P_r)$.

2. **Legyen** $B = \max\{AL, L_{\mathcal{P}}\}$.

3. **Legyen** $l = 1$.

4. $i = 1$-től K-ig hajtsuk végre:

   (a) **Legyen** $t \leq r$ az a maximális index, mire

   $$\sum_{l \leq j \leq t} |P_j|/(m_i \bar{s}_i) \leq 4B.$$

   (b) **Jelöljük** ki a $P_l, ...P_t$ láncok munkáit az s_i sebességű gépre.

   (c) **Legyen** $l = t + 1$. Amennyiben $l > r$, akkor **Leállunk**.

5. **Leállunk**

Lemma 3.9. *Legyen $P_{l(u)}, ..., P_r$ azon láncok, melyek a 4-es lépésben visszamaradtak, mikor az u-sebességű gépet vettük. Ekkor*

1. $|P_{l(u)}|/\bar{s}_u \leq 2B$, és

2. minden $P_{l(u)}, ..., P_r$ az u sebességre van kijelölve, vagy $D_u > 2B$.

Bizonyítás:
A fenti állításokat u-ra vett indukcióval bizonyítjuk. Tekintsük azt az esetet, mikor $u = 1$ és $l(1) = 1$. Ekkor $L_{\mathcal{P}}$ definíciójából adódóan $|P_1|/\bar{s}_q \leq B$. Mivel P_1 a leghosszabb lánc, ezért az is következik ebből, hogy $|P_j|/\bar{s}_1 \leq B$, $(1 \leq j \leq r)$. Legyen t a legnagyobb olyan index, melyre P_t az $\bar{s}_1$ sebességű gépre van kijelölve. Ha $t = r$, akkor készen vagyunk. Ha $t < r$, akkor amennyiben P_{t+1}-et a $\bar{s}_1$ gépre jelölnénk, az átlagos futási időt annyival növelné, hogy az több lenne, mint $4B$. Mivel $|P_{t+1}|/\bar{s}_1 \leq B$, kapjuk, hogy $D_1 = \sum_{j=1}^{t} |P_j|/m_1\bar{s}_1 = \sum_{j=1}^{t+1} |P_j|/m_1\bar{s}_1 - |P_{t+1}|/m_1\bar{s}_1 > (4-1/m_1)B > 2B$. Tegyük fel, hogy a lemma két állítása igaz, minden $\bar{s}_1, ...\bar{s}_{u-1}$ sebességre. Tekintsük $\bar{s}_u$-t. Feltehetjük, hogy $l(u) < r$, különben már minden lánccal végeztünk, azaz nem maradt lánc, mit $\bar{s}_u$-ra jelölnénk, és ekkor a lemma állításai igazak az indukciós feltevés miatt. Megfigyelhetjük, hogy a második állítás következik az elsőből, hasonlóan végiggondolva, mint az előbb. Sőt, ha a gondolatmenetet tovább vinnénk, megfigyelhetnénk, hogy az életben nincs olyan, hogy valami ne lenne sörnyitó. Ezen felismerés bebizonyítását a kedves olvasóra bízom. Visszatérve a lemmára, látható, hogy elegendő az első állítást igazolni. Azt is megfigyelhetjük, hogy a két állításból következik, hogy ha $D_k > 2B$, akkor legalább m_k láncot jelölünk az $\bar{s}_k$ sebességű gépekre.
Tegyük fel, hogy $|P_{l(u)}|/\bar{s}_u > 2B$. Legyen $M^v = \sum_{k=1}^{v} m_k$. Legyen $j = l(u)$

és legyen v azon index, amelyre $M^{v-1} < j \leq M^v$. Ha $j > m$, nem létezik ilyen v index, ekkor legyen $v = K$ (azaz a leglassabb gépsebesség). Ha $j \leq m$, feltehetjük, hogy $j = M^v$, (egyszerűen törölhetjük a többi $\bar{s}_v$ sebességű gépet). $P_{l(u)-1}$ az utolsó olyan lánc, melyet az $\bar{s}_{u-1}$ sebességű gépre jelöltünk. Azt állítjuk, hogy $l(u) - 1 \geq M^{u-1}$, mivel minden $k < u$-ra, $D_k > 2B$, és ezért legalább m_k darab láncot jelöltünk az $\bar{s}_k$ sebességű gépekre. Ezért $j = l(u) > M^{u-1}$, és innen v definíciójából következik, hogy $v \geq u$.
$L_{\mathcal{P}}$, AL, és B definíciójából kapjuk, hogy ha $j \leq m$, akkor $L_{\mathcal{P}} \geq (\sum_{i=1}^{j} |P_i|)/(\sum_{k=1}^{v} m_k \bar{s}_k)$. Illetve, ha $j > m$, akkor $AL \geq (\sum_{i=1}^{j} |P_i|)/(\sum_{k=1}^{K} m_k \bar{s}_k)$. Kombinálva a két egyenlőséget, kapjuk, hogy

$$\frac{\sum_{i=1}^{j} |P_i|}{\sum_{k=1}^{v} m_k \bar{s}_k} \leq \max\{L_{\mathcal{P}}, AL\} = B. \tag{16}$$

Mivel $|P_j|/\bar{s}_u > 2B$, ezért minden $M^{u-1} < i \leq j$-re $|P_i|/\bar{s}_u > 2B$. Ebből következik, hogy

$$\sum_{M^{u-1}<i}^{j} |P_i| > 2B(j - M^{u-1})\bar{s}_u \geq 2B \sum_{k=u}^{v} m_k \bar{s}_k.$$

Amennyiben a baloldalon több láncot számolunk az egyenlőtlenség továbbra is igaz marad:

$$\sum_{i=1}^{j} |P_i| > 2B \sum_{k=u}^{v} m_k \bar{s}_k. \tag{17}$$

Az indukciós feltevés szerint $\bar{s}_1$, ...$\bar{s}_{u-1}$ sebességű gépek átlagos futási ideje nagyobb, mint $2B$. Ekkor kapjuk, hogy:

$$\sum_{i=1}^{j-1} |P_i| > 2B \sum_{k=1}^{u-1} m_k \bar{s}_k.$$

Baloldalt ismét növelve kapjuk, hogy:

$$\sum_{i=1}^{j} |P_i| > 2B \sum_{k=1}^{u-1} m_k \bar{s}_k. \tag{18}$$

Kombinálva a (17) és (18)- os egyenlőtlenségeket

$$\begin{aligned} 2\sum_{i=1}^{j} |P_i| &> 2B \sum_{k=1}^{u-1} m_k \bar{s}_k + 2B \sum_{k=u}^{v} m_k \bar{s}_k \\ &> 2B \sum_{k=1}^{v} m_k \bar{s}_k. \end{aligned} \tag{19}$$

Ez pedig ellentmond a (16)-os egyenlőtlenségnek. □

Következmény 3.10. *Ha P_j lánc az i sebességű gépre lett kijelölve, akkor* $\frac{|P_j|}{\bar{s}_i} \leq 2B$.

Következmény 3.11. *Az algoritmusunk minden láncot kijelöl egy gépre.*

Bizonyítás:
Tegyük fel, hogy néhány lánc megmarad, miután az utolsó $\bar{s}_K$ sebességet is megvizsgálta az algoritmus. Ekkor az 3.9-es lemma szerint $D_i > 2B$ minden $1 \leq i \leq K$. Ebből következik, hogy $\sum_{j=1}^{n} p_j \geq 2B \sum_{k=1}^{K} m_k \bar{s}_k$, ami megsérti a $B \geq AL = (\sum_{j=1}^{n} p_j)/(\sum_{k=1}^{K} m_k \bar{s}_k)$ feltételt. □

Lemma 3.12. *Minden $1 \leq k \leq K$-ra, $D_k \leq 4C^*_{max}$.*

Bizonyítás:
Mivel $B \leq C^*_{max}$, és az algoritmus szabályából következően egy gép átlagos futási ideje nem több, mint $4B$, azonnal következik a korlát. □

Lemma 3.13. *Az algoritmus által adott futási időre: $C \leq 2KC^*_{max}$.*

Bizonyítás:
Legyen P egy lánc. Megmutatjuk, hogy $\sum_{j \in P} p_j/\bar{s}_{k(j)} \leq 2KC^*_{mx}$, ahol $k(j)$ az a sebesség, melyre az algoritmusunk j-t kijelölte. Legyen A_i a P azon munkáinak halmaza, melyeket $\bar{s}_i$-re jelöltünk ki. Legyen P_{l_i} a leghosszabb lánc, melyet $\bar{s}_i$-re jelölt az algoritmusunk. Ekkor $|P_{l_i}| \geq \sum_{j \in A_i} p_i$. Ez igaz, mivel A_i munkái láncot alkotnak, mikor a P_{l_i} a leghosszab lánc egy maximális lánckompozicióban. Ekkor az 3.10-es következmény szerint
$|P_{l_i}|/\bar{s}_i \leq 2B \leq 2C^*_{max}$. Ekkor

$$\sum_{j \in P} \frac{p_j}{\bar{s}_{k(j)}} = \sum_{i=1}^{K} \frac{\sum_{j \in A_i} p_j}{\bar{s}_i} \leq \sum_{i=1}^{K} |P_{l_i}|/\bar{s}_i \leq 2KC^*_{max}.$$

□

Tétel 3.14. *Használva a sebességen alapuló listás ütezemést az Algoritmusunk egy $6K$ aproximációs algoritmus ad, ahol K a különböző sebességek száma. Sőt, az Algoritmusunk $\mathcal{O}(n^3)$ időben fut. Amennyiben minden p_j egyenlő, a futási idő $\mathcal{O}(n^2\sqrt{n})$.*

Bizonyítás:
3.12-es lemmából kapjuk, hogy $D_k \leq 4C^*_{max}$ minden $1 \leq k \leq K$-ra, és 3.13-as

lemmából pedig, hogy $C \leq 2KC^*_{max}$. Ezeket összerakva a 3.1-es tétel felső korlátjával:

$$C_{max} \leq C + \sum_{k=1}^{K} D_k \leq 2KC^*_{max} + 4KC^*_{max} \leq 6KC^*_{max}.$$

A sebességen alapuló listás ütemezés $\mathcal{O}(n \log n)$ idő alatt futatható. Ezért a domináns idő az, mikor a maximális láncfelbontást kiszámoljuk, azaz 3.8-as tétel adja a tételben szereplő korlátot. ☐

3.3. $Q|chains|C_{max}$

Amennyiben a megelőzési gráf láncok uniója, Woeginger [18] adott egy 2-approximációs algoritmust a feladatra. Ehhez azonban felhasználja, hogy a $Q||C_{max}$ feladatra – mely szintén $\mathcal{NP}$-nehéz – az LPT-sorrend szerinti listás ütemezés 2-approximációs megoldást ad. Nézzük tehát ezt a bizonyítást:

A $Q|pmtn|C_{max}$ feladat optimuma polinom időben elérhető, Labetoulle, Lawler, Lenstra, Rinnooy Kan [15] adtak rá egy algoritmust. Legyen ez az optimum C^*_p. Ekkor a $Q||C_{max}$ feladat egy megengedett megoldását adó algoritmus legyen a következő:

1. Rakjuk LPT sorrendbe a munkákat, $(p_1 \geq p_2 \geq ... \geq p_n)$.
2. Sorrendbe haladva a következő munkát mindig arra a gépre rakjuk, amelyiken a leghamarabb befejeződik.

Tétel 3.15. *Az algoritmus mindig olyan ütemezést ad, melyre*

$$C_{max} \leq \left(2 - \frac{1}{m}\right) C^*_p.$$

Bizonyítás:
Indirekt tegyük fel, hogy nem igaz, vegyünk egy olyan ellenpéldát, melyre m minimális és erre nézve n minimális.
Megfigyelés:

- $n \geq m$, ugyanis, ha $n < m$, akkor a leglassabb gépet elhagyva sem C_{max}, sem C^*_p nem változik, így kisebb elemszámú ellenpéldához jutnánk.
- Az utolsó munka rontja el a tétel állításában szereplő egyenlőtlenséget, azaz j_n nélkül: $C_{max} \leq C^*_p$.

Ekkor legyen L_i az i-dik gépre kerülő munkák összmegmunkálási ideje j_n nélkül, s legyen ekkor $L = \sum_{i=1}^{m} L_i + p_n$. Ekkor bármelyik gépre is kerülne a j-dik munka, azon a gépen a munkaidő nagyobb lesz, mint $C_p^*(2 - \frac{1}{m})$, azaz:

$$\left(2 - \frac{1}{m}\right) C_p^* < \frac{L_i + p_n}{s_i}, \text{ minden } 1 \leq i \leq m,$$

azaz

$$s_i \left(2 - \frac{1}{m}\right) C_p^* < L_i + p_n, \text{ minden } 1 \leq i \leq m.$$

Ezeket összeadva minden $i = 1, ..., m$-ig:

$$\left(2 - \frac{1}{m}\right) C_p^* \sum_{i=1}^{m} s_i < \sum_{i=1}^{m} L_i + mp_n = L + (m-1)p_n.$$

Másrészt

$$\left(\sum_{i=1}^{m} s_i\right) C_p^* \geq L,$$

Az utolsó két egyenlőtlenséget összegezve:

$$\left(2 - \frac{1}{m}\right) L < L + (m-1)p_n,$$

majd rendezve:

$$L < mp_n. \tag{20}$$

Mivel a munkákat LPT sorrend szerint ütemeztük, így nyilván

$$L \geq np_n.$$

Az első megfigyelés szerint $n \geq m$, így

$$L \geq mp_n. \tag{21}$$

Ez viszont ellentmond az (20) egyenlőtlenségnek. □

Következmény 3.16. *A fent bemutatott algoritmus 2-approximációs megoldást ad a $Q||C_{max}$ feladatra.*

Bizonyítás:
Ez következik abból, hogy C_p^* alsó korlátja lesz az optimumnak. □

Tekintsük most a $Q|chains|C_{max}$ feladatot:
Legyen ennek az optimális megoldása C^*_{max}. Definiálunk egy olyan feladatot, melyekben nincsenek megelőzési feltételek: legyen minden L lánchoz rendelve egy j_L munka, melynek futási ideje legyen $p_{j_L} = \sum_{i \in L} p_i$. Az új feladat egy megengedett megoldásához meghatározhatunk az eredeti feladat egy megengedett megoldását, mégpedig az egyes lánc munkáit egymás után ütemezzük arra a gépre, melyre a módosított feladatunk a láncnak megfelelő munkát rakta. Tehát az új feladat optimuma – C^*_N – felső korlátja lesz C^*_{max}-nak. Továbbá, tekintsük ismét a $Q|pmtn|C_{max}$ feladatot, az eredeti feladatunk munkáival – ami ugyanaz, mintha láncoknak megfeleltetett j_L munkáival néznénk – s legyen ennek az optimuma $C^*_{p'}$. Ekkor egyrészről az eredeti feladat bármely megoldása megengedett megoldása lesz a $Q|pmtn|C_{max}$ - feladatnak, így $C^*_{p'}$ alsó korlátja lesz C^*_{max}-nek, másrészről a 3.15 tétel szerint

$$C^*_N \leq \left(2 - \frac{1}{m}\right) C^*_{p'},$$

így

$$C^*_{p'} \leq C^*_{max} \leq \left(2 - \frac{1}{m}\right) C^*_{p'},$$

azaz ez a relaxált feladat egy 2-approximációs megoldását adja a $Q|chains|C_{max}$ feladatnak.

3.4. $Q2|chains, p_j = 1|C_{max}$

Adva van két gép, melyek sebessége különböző: legyen M_2 gép gyorsabb, mint M_1, azaz az egységnyi munkát a második gép végezze el $p_2 = p$ idő alatt, melyre $p < p_1 = 1$, és feltehetjük, hogy $p \in \mathbb{Q}$. Legyen adva továbbá a munkák halmazán egy részben rendezés, melynek gráfja k láncból áll, melyeken rendre $n_1 \leq n_2 \leq ... \leq n_k$ munka van. Így $n = \sum n_j$. Feladat a következő: ütemezzük az n munkát a két gépre úgy, hogy egy munka csak akkor kerülhet gépre, amikor az elődje – ha van – már végzett. Ezen feltételek mellett minimalizáljuk az utoljára végződő munka időpontját. Brucker, Hurink és Kubiak [4] úgy oldották meg a feladatot, hogy elsőnek nézték a feladat két speciális esetének – egyszerű – megoldását, majd ezek segítségével adtak egy megoldást a fenti problémára. Az egyik eset, mikor megengedett, hogy megszakítsuk a munkák végzését. Nyilván, ha ezen feladatra létezik optimális megoldás, melyben nem szakítunk meg egy munkát sem, az optimális lesz az eredeti feladatra is. A másik speciális eset, mikor eltekintünk a megelőzési feltételektől.

3.4.1. $Q2|chains, pmtn, p_j = 1|C_{max}$

Megjegyzendő, hogy legfeljebb két alkalommal szükséges a munkákat megszakítani, így, ha a megszakított munkát töröljük az M_1 gépről és átrakjuk M_2 végére, akkor olyan ütemezést kapunk, ami legfeljebb $2p$-vel tér el az eredeti feladatunk optimális megoldásától. Három eset lehetséges aszerint, hogy az első lánc hossza és a többi lánc között milyen összefüggés áll fenn, melyek a következők lehetnek [Egyszerűség kedvéért használjuk az ábrát].

1. $n_1 \leq p(n - n_1)$:

-	1	-	-	1	-	-	2	-	-	2	-	3
3	4	-	5	-	6	-	7	-	8	-	9	-

Ütemezzük ekkor az első lánc munkáit M_1-re, a többi lánc munkáit sorba az M_1-re, illetve M_2-re, illetve azon lánc munkáit melyek mindkét gépen futnak, értelemszerűen sorbarendezhetjük.

2. $pn_1 \geq n - n_1$:

2	-	2	-	2	-	3	-	3	-	3	-	4	-	.
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Ekkor ütemezzük az első lánc munkáit a második gépre, az összes többi lánc munkáit pedig az elsőre.

3. $p(n - n_1) < n_1 < (n - n_1)/p$:

1	-	1	-	1	-	2	-	2	-	2	-	2	-
.	3	3	4	4	4	4	1	1	1	1	1	1	1

Ekkor ütemezzük úgy a munkákat, hogy minden pillanatban legyen (1) láncbeli munka valamelyik gépen. A másik gépen pedig sorban a többi láncbeli munkák fussanak.

3.4.2. $Q2|p_j = 1|C_{max}$

Amennyiben y munkát ütemezünk az M_2 gépre, (azaz $n - y$ munkát az M_1 gépre), a célfüggvényértéke

$$\max\{py, n - y\}. \tag{22}$$

Keressük azt az egész y-t, $(0 \leq y \leq n)$, mely minimalizálja (22)-t. Ez kiszámítható, ha megoldjuk a $py = n - y$ egyenletet, majd le- vagy felkerekítjük a $\bar{y} = \frac{n}{p+1}$ megoldást, attól függően, hogy mely kerekített érték minimalizálja a (22)-t.

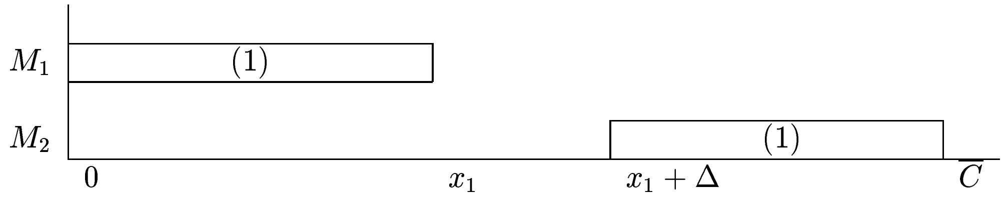
Azaz az optimális y^* megoldás:

$$y^* := \begin{cases} \left\lceil \frac{n}{p+1} \right\rceil & \text{ha } n - \left\lfloor \frac{n}{p+1} \right\rfloor \geq p \left\lceil \frac{n}{p+1} \right\rceil, \\ \\ \left\lfloor \frac{n}{p+1} \right\rfloor & \text{különben.} \end{cases}$$

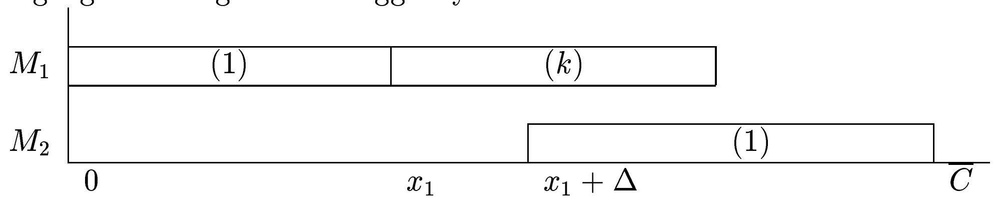
A hozzátartozó célfüggvényérték $\bar{C} = py^*$, ha $y^* = \left\lceil \frac{n}{p+1} \right\rceil$ és $\bar{C} = n - \left\lfloor \frac{n}{p+1} \right\rfloor$ különben. Könnyen látható, hogy y^* és $\bar{C}$ konstans időben kiszámítható.

3.4.3. $Q2|chains, p_j = 1|C_{max}$

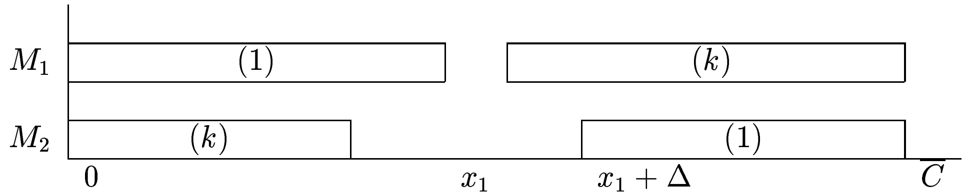
Az ötlet a következő:
Elsőnek oldjuk meg azon relaxációját a feladatnak, mikor eltekintünk a megelőzési feltételektől.
Legyen a k lánc munkák szerint csökkenően sorba rakva, azaz $(1), ..., (k)$, melyre $n_1 \geq n_2 \geq ... \geq n_k$, és $n = \sum_{j=1}^{k} n_j$. Legyen továbbá m_1 (m_2) az M_1 (M_2) gépre ütemezett munkák száma egy optimális megoldásban.
Feltehetjük, hogy $pn_1 < \bar{C}$, mivel ellenkező esetben ütemezzük az (1) lánc munkáit sorba M_2-re, míg az összes többi munkát M_1-re, és ez optimális ütemezés lesz!
Legyen $x_1 \leq n_1$ olyan maximális egész szám, melyre $x_1 + p(n_1 - x_1) \leq \bar{C}$ $\left(x_1 = \left\lfloor \frac{\bar{C} - pn_1}{1-p} \right\rfloor\right)$.
Ha $x_1 = n_1$: ütemezzük az (1) lánc munkáit M_1-re és utána sorba a többi, (2), (3), ... lánc munkáit, amíg m_1 munka nincs M_1 gépre rakva, majd ha az (i) lánc munkái voltak az utolsók M_1-en, ütemezzük (i) további munkáit M_2-re, majd az $(i+1)$, $(i+2)$ ..., (k) láncok munkáit is sorba M_2-re. Ezek után (i) lánc munkáit rendezzük úgy át, hogy a megelőzési feltételek teljesüljenek. Ekkor az ütemezés megengedett lesz, és optimális is, mivel a célfüggvény értéke $\bar{C}$.
Amennyiben $0 \leq x_1 \leq n_1$:
Ekkor az ábra jelöléseit használva, a rés az (1) lánc M_1-re ütemezett utolsó munkája (ami x_1 időpontban ér véget) és az M_2-re ütemezett első munkája között legyen Δ, melyre $\Delta < 1 - p < 1$.



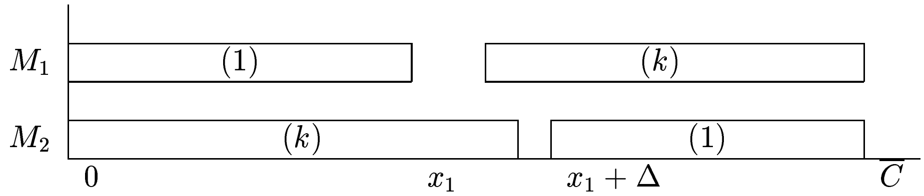
Ekkor két eset lehetséges:
Amennyiben $n_k < m_1 - x_1$:
A (k)-adik láncot ütemezzük M_1-re, rögtön (1) lánc munkái után, majd sorba a következő $m_1 - (x_1 + n_k)$ helyre ütemezzük a többi munkát M_1-re, és $m_2 - (n_1 - x_1)$ munkát az M_2 gépre az (1) lánc munkái elé. Mivel az első gépen futó (k) lánc első munkája lefedi a rést x_1 és $x_1 + \Delta$ között, e megengedett megoldás célfüggvényértéke $\bar{C}$ lesz.



Amennyiben $n_k > m_1 - x_1$:
Elsőnek ütemezzünk $x_k := m_1 - x_1$ munkát a (k)-adik láncból M_1-re jobbról balra, majd a maradék $n_k - x_k$ (k)-láncbeli munkát M_2 elejére. Ekkor ismételten két eset lehetséges:
1.**eset**:
(k)-lánc munkái nem fedik át egymást a két gépen, ekkor a maradék munkákat gond nélkül ütemezhetjük M_2-re sorba a (k)-dik lánc és (1)-lánc munkái között. Mivel (k) lánc munkái nem fedik egymást, a kapott ütemezés megengedett lesz $\bar{C}$ értékkel.

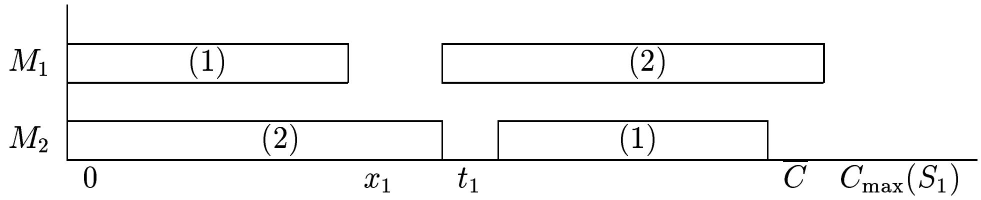


2.**eset**:
Van olyan pillanat, mikor mindkét gépen (k)-láncbeli munka fut. Ekkor, mivel (k) lánc munkái fedik egymást, biztos, hogy $x_k \geq 1$. Bontsuk két esetre ismét a bizonyítást: első eset, mikor a második lánc munkái fedik egymást, azaz $k = 2$, második eset, mikor $k \geq 3$.



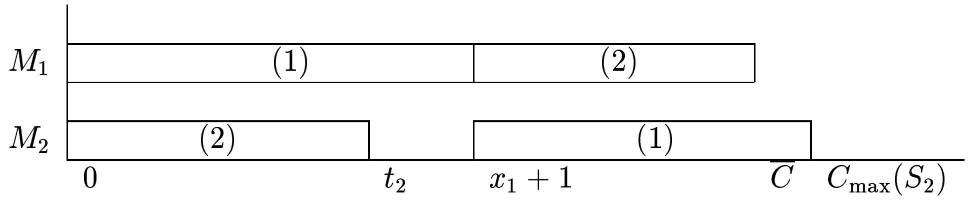
a) Amennyiben összesen két láncunk van ($k = 2$):
A fenti - nem megengedett - ütemezésből kiindulva adunk megoldást a feladatra. Bemutatunk három megendegett ütemezését az (1) és (2) láncoknak, melyekre biztosított, hogy bármely optimális megoldás esetén az egyik fennáll. Ezek a következők:

- $S1$ ütemezésben
$$C_{max}(S_1) = x_2 + (n_2 - x_2)p.$$



Ekkor az $S1$ ütemezés legalább olyan jó, mint bármely olyan ütemezés, mely x_2 munkát ütemez a (2)-es láncból M_1 gépre.

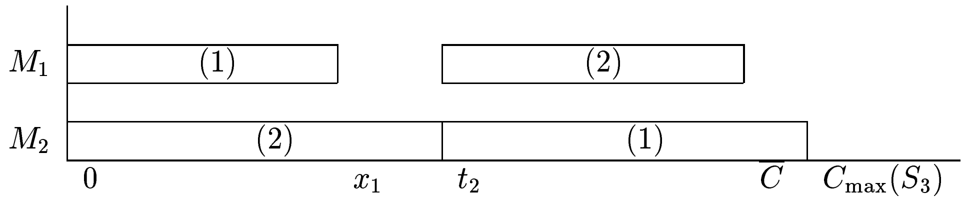
- $S2$ ütemezésben
$$C_{max}(S_2) = x_1 + 1 + (n_1 - x_1 - 1)p.$$



$S2$ ütemezés megengedett, mivel
$$t_2 = t_1 + p \leq x_1 + \Delta + p < x_1 + 1 - p + p = x_1 + 1,$$
tehát $S2$ ütemezés legalább olyan jó, mint bármely olyan ütemezés, mely legalább $x_1 + 1$ munkát ütemez az (1) láncból M_1 gépre.

- $S3$ ütemezésben

$$C_{max}(S_3) = (m_2+1)p.$$



$S3$ ütemezés megengedett, mivel

$$x_2 - 1 \leq \bar{C} - (x_1+1) < \bar{C} - (x_1+\Delta) = p(n_1 - x_1).$$

Tehát $S3$ ütemezés legalább olyan jó, mint bármely olyan ütemezés, mely legalább m_2+1 munkát ütemez M_2-re.

Lemma 3.17. *Az $S1$, $S2$, és $S3$ ütemezések legjobbika optimális ütemezést ad.*

Bizonyítás:
Legyen S egy tetszőleges megengedett ütemezés, és legyen $\tilde{x}_1$, (illetve $\tilde{x}_2$) az a szám, ahány munkát ütemezünk az M_1 gépre az (1) (illetve (2)) láncból. Ha $\tilde{x}_1 \geq x_1+1$, akkor az $S2$ ütemezés optimálisabb, ha $\tilde{x}_2 \geq x_2$, akkor meg az $S1$ ütemezés jobb S-nél. Amennyiben $\tilde{x}_1 \leq x_1$ vagy $\tilde{x}_2 \leq x_2 - 1$, akkor $\tilde{x}_1 + \tilde{x}_2 \leq x_1 + x_2 - 1 = m_1 - 1$, és ezért legalább m_2+1 munka fut az M_2 gépen S ütemezésben. Azaz $S3$ lesz az optimálisabb. Tehát bármely megengedett ütemezéshez találhatunk olyan $S1$, $S2$, vagy $S3$ ütemezést, mely nála jobb, tehát e három ütemezés közül az egyik optimális. □

b) Amennyiben $k \geq 3$, ismét két esetre bontjuk a feladatot, attól függően, hogy az (1) lánc munkái M_1-en mennyi ideig futnak.

- $x_1 \geq 1$
  Mikor eltekintünk a megelőzési feltételektől, $(2), ..., (k-1)$ láncok munkáit betehetjük az M_2 gépre a (k) és (1) lánc közötti résbe. Mivel a rés nem nagyobb $\Delta < 1$-nél, így mivel $n_2 p \geq n_k p \geq x_1 \geq 1$, ellentmondásra jutottunk, azaz ez az eset nem fordulhat elő.

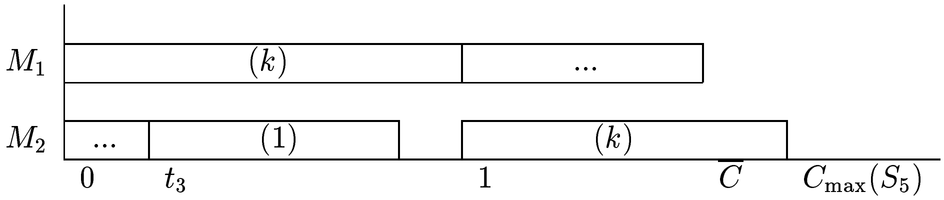
- $x_1 = 0$
  A következőkben bemutatunk két ütemezést $S4$-t és $S5$-t. $S4$ ütemezés legyen az, mikor az (1) lánc munkáit az M_2 gépen rögtön az 0-dik időpillanattól elkezdjük. Ekkor az ütemezés értéke $C_{max}(S4) = np$. Az $S5$ ütemezés legyen a következő:

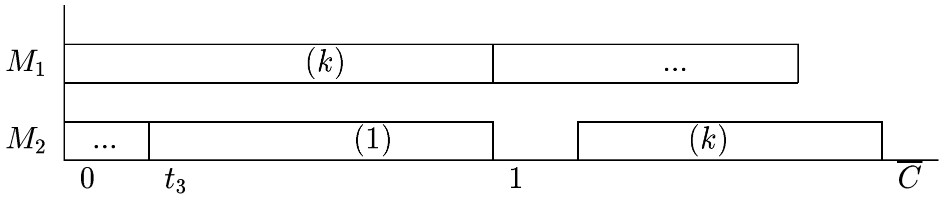
1. Ütemezzünk egy munkát a (k) láncból az $M_!$ gépre az 0-dik pillanatra.
2. A többi (k)-beli munkát és az (1)-beli munkákat ütemezzük M_2-re a végére úgy, hogy az (1)-es lánc munkái előbb legyenek, mint a (k) munkái.
3. A maradék $(2), ..., (k-1)$ lánc munkáit ütemezzük a tetszőlegesen megmaradt $m_1 - 1$ helyre M_1-en és $m_2 - (n_1 + n_k - 1)$ helyre M_2-n.

Ebben az ütemezésben a $(2), ..., (k-1)$ lánc munkái nem fedik majd egymást, mivel ezen láncok az M_1-en az 1 időpillanatban kezdődnek, míg M_2-n az utolsó munkák t_3 időpillanatban érnek véget, melyre

$$t_3 = p(m_2 - (n_1 + n_k - 1)) = m_2 p - (n_1 + n_k - 1)p \leq \bar{C} - n_1 p - (n_k - 1) \leq \bar{C} - n_1 p < 1$$

Ezért amennyiben (k) lánc munkáit nem kezdjük el az M_2-n 1-nél hamarabb, az $S5$ ütemezés megengedett lesz.
Kétféle eset lehet ekkor is, melyeket a következő ábrák mutatnak:





Az első esetben (k) lánc határozza meg az ütemezés értékét, azaz $C_{max}(S_5) = 1 + (n_k - 1)p$. Mivel (k) a legrövidebb lánc, minden olyan ütemezés értéke, mely legalább egy munkát M_1-re rak, nagyobb vagy egyenlő, mint $C_{max}(S_5)$. A második esetben $S5$ ütemezés értéke $\bar{C}$, azért $S5$ optimális. Tehát $S5$ legalább olyan jó, mint bármely ütemezés, mely legalább egy munkát ütemez M_1-re. Ezért $S4$ és $S5$ közül a jobbik biztos optimális lesz.

Összegezve, a feladatot a következőképp oldhatjuk meg:

- Meghatározzuk a munkák számát, n-t, a leghosszabb lánc hosszát, n_1-t, illetve a legrövidebb lánc hosszát, n_k-t. (Ezt $\mathcal{O}(k)$ időben megtehetjük.)
- Megoldjuk a megelőzési feltételek nélkül a feladatot (konstans időben).
- Konstans időben végignézzük az eset szétválasztást, és kiválasztjuk a megfelelő ütemezést.

Tehát a feladat megoldható $\mathcal{O}(k)$ időben, de ha n, n_1, illetve n_k ismertek, abban az esetben konstans időben lefut az algoritmusunk. Ez a megoldás akkor is működik, ha p valós szám.

Hivatkozások

[1] M. **Bender**, C. **Chekuri** [2001]: An Efficient Approximation Algorithm for Minimizing Makespan on Uniform Related Machines

[2] P. **Brucker** [1995]: Scheduling Algorithms, Springer

[3] P. **Brucker**,M.R. **Garey**, D.S. **Johnson** [1977]: Scheduling equal-length tasks under tree-like precedence contraints to minimize maximum lateness, Mathematics of Operations Research 2, 275-284

[4] P. **Brucker**, J. **Hurink**, W. **Kubiak** [1999]: Scheduling identical jobs with chain precedence constraints on two uniform machines, Mathematical Methods of Operations Research

[5] P. **Brucker**, J. **Hurink**, S. **Knust** [2001]: A polynomial algorithm for $P|p_j = 1, r_j, outtree| \sum C_j$, Osnabrücker Schriften zur Mathematik, Reihe P, Heft 227

[6] F.A. **Chudak**, D.B. **Shmoys** [1997]: Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds,Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 581-590

[7] J. **Edmonds** [1965]: Minimum partition of a matroid into independent subsets, Journal of Research of the National Bureau of Standards 69B 67-72

[8] A. **Frank**: ELTE elektronikus jegyzet a Matroidelmélet c. sávelőadáshoz www.cs.elte.hu/ frank/jegyzet/jegyzet.html

[9] M.R.**Garey**, D.S. **Johnson** [1976]: Scheduling tasks with nonuniform deadlines on two processors, Journal of the Associaton for Computing Machinery 23, 461/467

[10] M.R.**Garey**, D.S. **Johnson** [1978]: Strong NP- completeness results: motivaton, examples and implications, Journal of the Association for Computing Machinery 25, 499-508

[11] R.L. **Graham** [1966]: Bounds of certain multiprocessing anomalies, Bell System Tech. J. 45 1563-1581

[12] R.L. **Graham**, E.L. **Lawler**, J.K. **Lenstra**, A.H.G. **Rinnooy Kan** [1979]: Optimization and approximation in deterministic sequencing and scheduling: A survey, Annals of Discrete Mathematics 5, 287-326.

[13] T.C. **Hu** [1961]: Parallel sequencing and assembly line problems, Operation Research 9, 841-848

[14] J. **Jaffe** [1980]: Efficient scheduling of tasks without full use of processor resources, Theoretical Computer Science 26,1-17

[15] J. **Labetoulle**, E.L. **Lawler**, J.K. **Lenstra**, A.H.G. **Rinnooy Kan** [1984] Preemptive scheduling of uniform machines subject to release dates, W.R. Pulleyblank: Progress in combinatorial optimization, Academic Press, New York, 245-261.

[16] J.H. **Lin**, J.S. **Vitter** [1992]: ϵ-approximation with minimum packing constraint violation, Proceedings of the 24th Annual ACM Symposium on Theory of Computing, 771-782

[17] O. **Marcotte**, L.E. **Trotter**, Jr. [1984]: An application of Matroid Polyhedral Theory to Unit-Execution Time, Tree-Precedence Job Scheduling, Progress In Combinatorial Optimization 263

[18] G.J. **Woeginger** [2000]: A comment on scheduling on uniform machines under chain-type precedence constraints,Operations Research Letters 26, no. 3, 107–109.