Eötvös Loránd Tudományegyetem
Természettudományi kar

Algoritmusok ütemezési feladatokra

Készítette:
Nácz Vanessza Kitti
Matematika BSc
elemző szakirány

Témavezető:
Dr. Jordán Tibor
egyetemi tanár
Operációkutatási Tanszék

ELTE
EÖTVÖS LORÁND
TUDOMÁNYEGYETEM

Budapest, 2022

NYILATKOZAT

Név: Nácz Vanessza Kitti

ELTE Természettudományi Kar, szak: Matematika BSc

NEPTUN azonosító: FL1A4I

Szakdolgozat címe:
Algoritmusok ütemezési feladatokra

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2022. 05. 26.

Nácz Vanessza Kitti

a hallgató aláírása

Köszönetnyilvánítás

Legelsősorban szeretném megköszönni Dr. Jordán Tibor tanár úrnak a kitartó munkáját, nélküle ez a szakdolgozat nem készülhetett volna el. Az ütemezéselmélet tárgy keretein belül a matematika egy – számomra – új ágát ismerhettem meg és már az órákon felkeltette a téma iránt az érdeklődésemet, amelyet a konzultációk alkalmával fenn is tartott, végig motivált (például tudományos videós forrásokkal). Sokat segített a szakirodalmak kiválasztásánál, majd a rendelkezésre álló idő beosztásában, végül a dolgozat szakmai ellenőrzésében és a felmerülő kérdéseim tisztázásában.
Továbbá köszönöm a szüleimnek a támogatást, hogy mindenben segítettek a tanulmányaim alatt.

Tartalomjegyzék

1 Bevezetés

Az alábbi fejezetben megvizsgáljuk, hogy egy ütemezés bizonyos értelemben mitől lehet jobb egy másiknál és a megszakíthatóság mennyivel járulhat hozzá a kisebb optimum eléréséhez. Akkor beszélhetünk a megszakíthatóság előnyéről, ha ugyanazon a példán vizsgáltuk meg azokat, azaz ugyanarra az inputra. Megnézzük azt is, hogy egyes példák miért lehetnek érdekesek számunkra.

Egy ütemezési feladat általában a végrehajtandó munkákból áll, amelyekhez tartoznak attribútumok is, mint például a megmunkálási idejük vagy az elérhetőségük időpontja (amikortól elkezdhetünk egy munkát). Emellett adottak gépek, amelyeken elvégezhetjük azokat. Beszélhetünk egy- vagy többgépes ütemezésről attól függően, hogy hány gép áll a rendelkezésünkre. Párhuzamos ütemezéseknél a gépek sebessége egyenlő, tehát ez alapján mindegy, hogy hova kerül fel egy munka. Uniform ütemezéseknél más a helyzet, ott minden egyes géphez hozzárendelünk egy sebességet is, ami kifejezi, hogy milyen gyorsan hajtható rajta végre egy feladat.

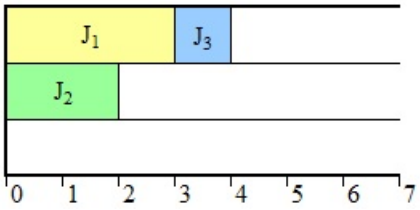
A célfüggvényünk lehet például a teljes átfutási idő vagy az összmegmunkálási idő minimalizálása (ez utóbbit súlyokkal is vehetjük), de ezenfelül számos más minimalizálandó értékünk is lehet. Továbbá tisztáznunk kell az elején, hogy mikor tekintünk egy megoldást megengedettnek, például lehet-e a megoldásunkban megszakítása egy-egy munkának.

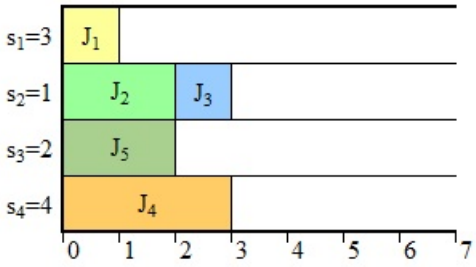
A munkák számát általában n-nel jelöljük és mindegyik munkához egy indexet társítunk (1-től n-ig). A munkák hosszát pedig p_j-vel szokás jelölni, ahol a j számok pozitív egészek.

A nem megszakítható ütemezésekben minden munkát el kell végeznünk teljes egészében és mindegyikükhöz egy folytonos időintervallumot rendelünk. Pontosan ebben az intervallumban kell elvégeznünk az egyik gépen (nem helyezhetjük át). Ezzel ellentétben a megszakításos ütemezésekben a munkákat kisebb darabokra vághatjuk és ezeket külön munkálhatjuk meg, akár külön gépre is áttehetjük azokat, viszont sosem futhatnak egy időben ugyanazon munka részei. Ez életszerű lehet, ha például ugyanazon a tárgyon kell feladatokat végrehajtani különböző műhelyekben, ekkor nem lehet a tárgyunk egyszerre két helyen. Természetesen itt is az összes munkát be kell fejeznünk.

A gépek a nulla időpillanattól fogva elérhetőek egészen addig, amíg már nincs szükségünk rájuk. Egy gép egyszerre csak egy munkával képes foglalkozni és egy munka egy időpillanatban csak egy gépen lehet, azaz egy adott munka részei csak diszjunkt időintervallumokon lehetnek ütemezve. A

munkák ésszerűen vannak megszakítva mindig, azaz a részmunkák mérete valós szám kell hogy legyen. Az alábbi ábrákon bemutatjuk, hogy hogyan szokás jelölni az egygépes, párhuzamos gépes és az uniform gépes ütemezéseket. Legyenek a megmunkálási idők az alábbiak: $p_1 = 3,\ p_2 = 2,\ p_3 = 1,\ p_4 = 12,\ p_5 = 4$.





Az uniform gépek esetében az i. gép sebességét $s_i > 0$-val jelöljük. Ekkor könnyen átgondolható, hogy a j munka megmunkálási ideje az i. gépen épp $\frac{p_j}{s_i}$ lesz. Az ábrán tehát nem a munkák valódi hosszát, hanem a gépek sebessége által módosított méretüket láthatjuk.

Összehasonlítva a megszakítást megengedő ütemezéseket a nem megszakíthatóakkal elmondható, hogy az utóbbinál az összes munka egy-egy géphez és egy-egy folytonos idősávhoz van rendelve, ezeknek megfelelően kell teljes egészében befejeződniük úgy, hogy az adott időintervallumban folyamatos megmunkálás alatt legyenek. A megszakítható ütemezéseknél szintén szükséges az, hogy az összes munkát el tudjuk végezni, viszont azokat véges sok részre vághatjuk. Minden részt külön munkálunk meg, de itt is vigyázni kell, hogy nem futhatnak párhuzamosan ezek. Éppen ezért, ha egy

munkánk van csak, nem segít ha több párhuzamos gép is a rendelkezésünkre áll. Átfutási időnek a legutoljára véget érő munka befejezését értjük, ami a mostani példákban a munka megmunkálási idejének hossza mindkét esetben (megszakíthatóságtól függetlenül).

Itt megjegyezhetjük, hogy van olyan ütemezési modell (*fractional scheduling*), ahol megszakíthatjuk a munkákat és azok részeit egymástól függetlenül ütemezhetjük, akár még párhuzamosan is, de ezt csak lényegesen bonyolultabb feladatoknál használják.

Segíthet viszont a megszakíthatóság, ha több munkánk van, nem csak egy. Például, ha három munkát kell elvégeznünk, amelyek mindegyikének két egység a megmunkálási ideje, akkor a teljes átfutási idő a nem megszakítható esetben négy egységet fog igénybe venni minimum, míg a megszakítás engedélyezése után már elégséges lesz 3 egység is a feladatok befejezéséhez. Tehát $p_1 = p_2 = p_3 = 2$ és a feladatunk pedig a következő: $P2 \mid pmtn,\ p_j = 2 \mid C_{\max}$, de akár az átlagos átfutási időt is minimalizálhatjuk, ugyanúgy javít majd a megszakítás.

A ütemezések jelölésére a szokásos, Graham és munkatársai által bevezetett $\alpha \mid \beta \mid \gamma$ jelöléseket használjuk [7]. Az irodalomjegyzék után mellékletben összefoglalva találhatóak a főbb jelölések.

2 A megszakíthatóság ereje

Az alábbi fejezetben azt definiáljuk, hogy mi számít előnynek a megszakítás megengedésénél. Azt szeretnénk mérni, hogy mennyire tudja a megoldásunkat javítani. A megszakítható változatú ütemezésekben nem kötelező a munkákat részekre osztani, de lehetséges, ezért minden nem megszakítható megoldás a megszakítható változatú feladatunkra is egy lehetséges ütemezést ad.

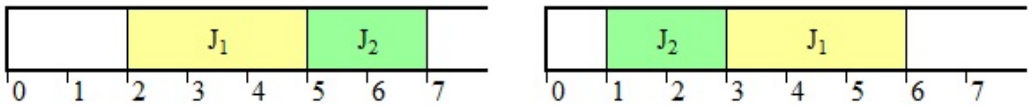
Hasonlítsuk össze az optimális nem megszakítható ütemezést az optimális megszakítható ütemezéssel ugyanarra az I inputra nézve. Jelöljük $OPT_p(I)$-vel a megszakítható változatú feladatunk optimumát, $OPT_n(I)$-nel pedig a nem megszakíthatóét. Ekkor a megszakíthatóság előnye:

$$\sup_I \frac{OPT_n(I)}{OPT_p(I)} \geq 1$$

Mivel minden nem megszakítható megoldás a megszakítható változatú feladatra is érvényes, illetve lehet hogy ez a könnyítés javít az optimumon,

ezért a tört nevezője maximum akkora lehet, mint a számláló. Nem ronthat az eredményünkön a megszakítás lehetővé tétele, így a fentebbi egyenlőtlenség teljesülni fog mindig.

Nézzük meg, hogy mi határozza meg, hogy melyik megoldása lesz a jobb egy adott feladatnak. Ehhez vezessük be a súlyokat, amelyek nem lehetnek negatívak és egy adott munka fontosságát jelzik nekünk. Vizsgáljuk meg az alábbi két ütemezést, hogy mennyiben befolyásolja a súlyok megjelenése az eredményeket. Például: $(p_1 = 2,\ p_2 = 1,\ r_1 = 2,\ r_2 = 1,\ \omega_1 = 11,\ \omega_2 = 2)$



Az átfutási időt nézve látjuk, hogy $C_{\max} = 7$ az első esetben, a másodikban $C_{\max} = 6$, így ezek alapján a második lenne a jobb megoldás. Nézzük meg $\sum C_j$-t is. Az első esetben $\sum C_j = 12$, a másodikban $\sum C_j = 9$, tehát ezek alapján is a második ütemezés a célszerűbb. Viszont, ha figyelembe vesszük a súlyokat már más lesz a helyzet, mert az első esetben $\sum \omega_j C_j = 11 * 5 + 2 * 7 = 69$, míg a másodikban nőni fog ez a súlyozott érték $\sum \omega_j C_j = 2 * 3 + 11 * 6 = 72$. Tehát attól függ, hogy melyik lesz egy optimális megoldás, hogy pontosan mi érdekel minket, mit szeretnénk minimalizálni.

A továbbiakban egy adott m gépet és n munkát tartalmazó I inputra jelölje P a munkák összméretét és q pedig a legnagyobb munka hosszát. Ekkor mind a megszakítható, mind a megszakítás nélküli feladatokban érvényes marad, hogy egygépes ütemezéseknél a teljes átfutási idő minimum P. Ugyanez m egyforma párhuzamos gép esetén $\frac{P}{m}$-re módosul (a megmunkálási idők átlagolása következtében). Kivéve ha q nagyobb ennél, mert akkor legalább ennyi idő szükséges az összes munka elvégzéséhez, mert mindegyiket el kell végeznünk (q-t is) és a párhuzamos futtatása egy munkának nem engedélyezett. Uniform gépek esetén ez az alsó becslés $\frac{P}{\sum_i s_i}$, ahol s_i az i. gép sebességét jelöli. Ugyanúgy itt is figyelembe kell vennünk a leghosszabb munkát, ha azt a leggyorsabb gépre helyezzük, akkor $\frac{q}{\max s_i}$ időt vesz igénybe q megmunkálása minimum. Tehát itt is az utóbbi kettő közül a nagyobb lesz az alsó korlátunk, amelyhez képest az átfutási idő legalább akkora.

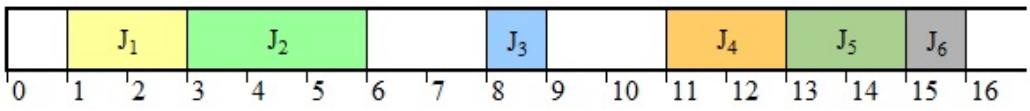
2.1 Példák és egyszerű bizonyítások:

2.1.1 $1 \mid r_j \mid C_j$

Ennek a feladatnak a megoldására létezik megfelelő algoritmus, amely még a megszakítható változatú problémára is optimális megoldást ad. A megszakítás előnye itt 1 lesz, azaz nem kapunk jobb megoldást általa.

Rendezzük a munkákat az elérhetőségük szerint ($r_1 \leq r_2 \leq \cdots \leq r_n$). Ütemezzük az első munkát $S_1 = r_1$-től folyamatosan az $[S_1, T_1) = [r_1, r_1 + p_1)$ intervallumon. A többi $i = 2, 3, \ldots n$ munkát sorban tegyük fel a gépre úgy, hogy S_i pillanatban rendeljük a géphez, ahol $S_i = \max\{r_i, T_{i-1}\}$, azaz vagy az elérhetővé válása pillanatában szabad a gépünk, ezért el tudjuk kezdeni a munkálatokat, vagy meg kell várnunk, míg az előtte lévő munka befejeződik. Ekkor $[S_i, S_i + p_i)$ intervallumra kerül az i. munka.

I	1	2	3	4	5	6
p_j	2	3	1	2	2	1
r_j	1	2	8	11	12	13



Ez az algoritmus azért ad optimális megoldást a feladatra, mert ha megnézzük az utolsó blokkot, akkor a benne lévő munkák elérhetési ideje legalább akkora, mint ahol a blokk kezdődik. Ettől az időpillanattól fogva minimum annyi idő szükséges, mint a benne lévő munkák hosszának az összege. Az algoritmus pont ezt adja nekünk, hiszen nincs állásidő a gépen. Jól látszik ezért, hogy nincs jobb megoldása a feladatnak, még akkor sem, ha megengedjük a megszakíthatóságot. A példánkban az utolsó blokk a 11 időpillanatban kezdődik és az ennél nagyobb indexszel – azaz legalább ekkora elérhetési idővel – rendelkező munkák megmunkálási idejének összege $p_4 + p_5 + p_6 = 5$. Az átfutási idő alsó korlátja ezért itt $11 + 5 = 16$ lesz és látjuk, hogy az algoritmussal is ezt kapjuk.

2.1.2 $P \mid\mid C_{\max}$ és $Q \mid\mid C_{\max}$

Ezek a feladatok NP-nehezek. Nem tudunk pontos optimumot adni, hanem azt szeretnénk minél jobban közelíteni. Ehhez approximációs algoritmusokat keresünk. Egy minimalizációs feladatra egy algoritmus közelítő, ha megengedett megoldást ad, polinom idejű és ha az általa kapott értéket elosztjuk az optimummal, hányadosuk nem kisebb mint egy.

Erre a feladatra létezik $(2 - \frac{1}{m})$ közelítő algoritmus. Ehhez a listás ütemezést fogjuk alkalmazni, tehát sorba állítjuk a munkákat és a kezdő időpillanattól fogva, amint van szabad gép, feltesszük a soron következő munkát, amelyet eddig még nem rendeltünk egy géphez sem.

Gondoljuk át, mennyi munkát tudunk teljesíteni t időegység alatt. Mivel m gépünk van, ezért éppen $m * t$ időegységnyi feladatot tudunk azalatt elvégezni. Az összes munkát teljesen be kell fejeznünk, ezért teljesülni fog a következő: $OPT * m \geq \sum_{j=1}^{n} p_j$. Ezt átrendezve a következőt kapjuk:

$$OPT \geq \frac{\sum_{j=1}^{n} p_j}{m}$$

A másik egyenlőtlenség, amit fel tudunk írni az a leghosszabb munkához köthető. Mivel azt is kötelesek vagyunk elvégezni, ezért annál nagyobb kell hogy legyen az optimum.

$$OPT \geq \max_j p_j$$

Legyen S a listás ütemezésünk és k pedig a legkésőbb végződő munka, amelynek kezdő időpillanata t. Ezen időpont előtt biztos, hogy az összes gép végig foglalt volt, mert a listás ütemezés miatt nem lehetséges, hogy úgy legyen állásidő egy gépen, hogy még van szabad munka, amelyet még nem rendeltünk sehova se. Éppen ezért az addig elkészült munkák minimum $m*t$ helyet foglalnak el a gépeken: $\sum_{j \neq k} p_j \geq m * t$. Felhasználva az előbbi két egyenlőtlenséget a következőt kapjuk:

$$C^S_{\max} = t + p_k \leq \frac{\sum_{j \neq k} p_j}{m} + p_k = \frac{\sum_{j=1}^{n} p_j}{m} + (1 - \frac{1}{m}) p_k$$

$$\leq OPT + (1 - \frac{1}{m}) * OPT = (2 - \frac{1}{m}) * OPT$$

A listás ütemezést p_j szerint monoton csökkenő (LPT) sorrendben elkészítve jobb, $\frac{4}{3}$ közelítő algoritmust kapunk a $P \mid\mid C_{\max}$ feladatra.

A $P \mid\mid C_{\max}$ feladat megoldására használjuk az LPT sorrendet. Ha m munkánk lenne, akkor mindegyikük külön gépre kerülne így a legnagyobb munka hosszával lesz egyenlő a teljes átfutási idő, mert ha az utána következő munkákat kiiktatnánk az ütemezésből, akkor sem változna ez az érték. Ha legalább m munkánk van, akkor legalább q (a leghosszabb munka megmunkálási ideje) idő szükséges a munkák ütemezéséhez. Legyen a legutoljára végződő munka hossza p. Ekkor az összes munkára teljesül $p_j \geq p$, hiszen LPT sorrendben rendeztük azokat.

Itt kékkel az adott gépen az összes addigi munkát színeztük be. Jelölje P az összes munka hosszát, $P = \sum_j p_j$ és legyen i az a gép, amelyre p kerül. Nézzük meg, hogy az egyes k gépek a p munka felhelyezése előtt hol állnak le és jelöljük ezt X_k-val. Ekkor, ha az i. gépre kerül az utolsó munka, akkor $C_{\max} = X_i + p$. Mivel az i-t úgy választjuk, hogy a p munka a lehető leghamarabb véget érjen, ezért minden k gépre teljesül, hogy $X_k \geq X_i$. Ha minden géphez – kivéve az i-hez – hozzáadnánk egy p hosszúságú munkát, akkor mindegyik gép legalább $C_{\max}$ ideig foglalt lenne, ezért $P' \geq m * C_{\max}$, azaz $C_{\max} \leq \frac{P'}{m}$. Itt P' a régi és az új munkák együttes méretét jelöli, így $P' = P + (m-1)p$. Ezt és az előbbi egyenlőtlenséget fogjuk felhasználni ahhoz, hogy megnézzük, mennyit tud javítani a megoldásunkon a megszakíthatóság engedélyezése. Tudjuk, hogy ennél a relaxáltnál az optimum legalább $\frac{P}{m}$ lesz majd. Mivel legalább $m+1$ munkánk van, ezért $P \geq (m+1) * p$. A munkák nem növekvő sorrendben követik egymást, ezért mindegyik munka legalább p hosszúságú kell hogy legyen, ezért $p \leq \frac{P}{m+1}$. Ezeket felhasználva $C_{\max}$-ra a

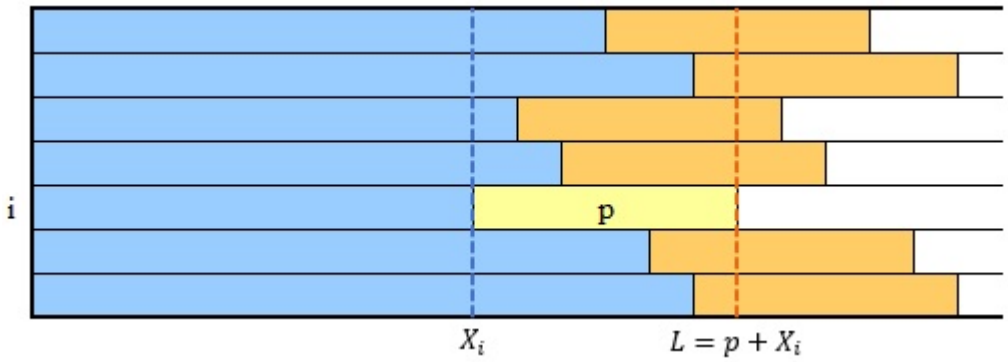
következőket kapjuk:

$$C_{\max} \leq \frac{P'}{m} = \frac{P + (m-1)p}{m} \leq \frac{P + \frac{P(m-1)}{m+1}}{m} = \frac{2m}{m(m+1)} * P.$$

Összehasonlítva ezt az eredményt a megszakítható változatú feladat $\frac{P}{m}$ alsó korlátjával

$$\frac{2m}{m+1}$$

lesz a megszakítás előnye legfeljebb.



Uniform gépekre is van optimális megoldás, sőt több – McNaughtonénál nehezebb – algoritmust találtak rá. Az egyszerűség kedvéért itt is legyenek a munkák nagyságuk szerint rendezve: $p_1 \geq p_2 \geq \cdots \geq p_n$. Feltehetjük, hogy $n \geq m$ különben az $m - n$ leglassabb gépre nem lenne szükségünk egy optimális megoldásnál, ezeket eltávolítanánk, mert megszakíthatóságtól függetlenül egyszerre csak n gép működhet az összes időpillanatban.

Legyen az első k legnagyobb munkának az összmegmunkálási ideje $P_k = \sum_{l=1}^{k} p_l$. Tegyük fel, hogy a gépek a sebességük szerint vannak rendezve, tehát $s_1 \geq s_2 \geq \cdots \geq s_m$. Legyen ekkor az első k leggyorsabb gép összsebessége $S_k = \sum_{l=1}^{k} s_l$. Tudjuk, hogy a megszakítható változatú feladatnál a teljes átfutási idő

$$\max\left\{\frac{P_1}{S_1}, \frac{P_2}{S_2}, \cdots \frac{P_{m-1}}{S_{m-1}}, \frac{P_n}{S_m}\right\}$$

(Átgondolva a párhuzamos gépek esetét, ahol minden $s_l = 1$, $S_k = \sum_{l=1}^{k} s_l = k$.)

Legyen annyi munkánk, ahány gépünk van, azaz m darab, melyek mindegyike $2m-1$ nagyságú. A gépek közül $m-1$ sebessége legyen 2, az utolsóé pedig 1. Nem megszakítható esetben az egyik munka teljes egészében a leglassabb gépre kerül vagy nem használva azt, lesz egy olyan gyors gépünk, amely két munkát fog elvégezni. Mindkét esetben a teljes átfutási idő minimum $2m-1$ egységnyi lesz.

A megszakítható esetben k munka összhossza $P_k = k(2m-1)$, az összsebesség $S_k = 2k$, ha $k < m$, és $S_m = 2m-1$, ha $k = m$. Az előbbi képletbe behelyettesítve:

$$\max\left\{\frac{k(2m-1)}{2k}, \frac{m(2m-1)}{2m-1}\right\} = m.$$

Így a megszakítás előnye ebben az esetben $\frac{2m-1}{m}$ lesz.

Általános esetben hasonlóan nézzük meg a megszakítás előnyét, mint a párhuzamos gépek esetén tettük. Legyen $S = S_m = \sum_i s_i$. Ekkor a megszakítható feladat alsó korlátja $\frac{P}{S}$ lesz. A másik különbség még, hogy az LPT sorrend másképp viselkedik sebességgel rendelkező gépek esetén. Az első m munka így nem feltétlenül az első m gépre kerül fel (például ha egy gép a többinél lényegesen gyorsabb, lehet hogy rá kerül fel az összes munka). Feltehetjük, hogy legalább m munkánk van, különben legalább egy gép szükségtelenné válik számunkra. Így mikor a párhuzamos gépeknél az utolsó munkát arra a gépre helyeztük, ahol a legkisebb lesz általa a teljes átfutási idő, most arra a gépre helyezzük, ahol $\frac{p}{s_i}$-t hozzáadva lesz minimális a $C_{\max}$. Ezzel számolva hasonlóan kijön, hogy a megszakítás előnye itt is $\frac{2m-1}{m}$ legfeljebb.

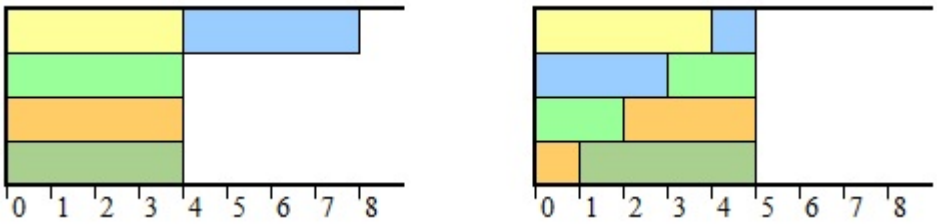
2.1.3 $P \mid pmtn \mid C_{\max}$

Ennek a feladatnak a megoldásához az egyik leghíresebb algoritmust fogjuk használni. Legyen

$$D = \max\left\{\frac{\sum_{j=1}^{n} p_j}{m}, \max_j p_j\right\},$$

amelyhez képest az optimum legalább ekkora. A McNaughton algoritmust használva elsőször határozzuk meg a munkák egy sorrendjét, majd kezdjük el egyenként folyamatosan ütemezni ezeket. Az első gépet töltsük fel a munkákkal egészen a D időpillanatig, és ha egy munka túlcsordulna rajta, szakítsuk

azt meg és folytassuk a következő gépen. Így összesen $m * D$ mennyiségű munkát tudunk ütemezni. Azt már tudjuk, hogy az optimum nem lehet kisebb D-nél, már csak azt kéne belátnunk, hogy ezen idő alatt be is tudjuk fejezni az összes munkát. Ehhez először az lenne szükséges, hogy mindegyik munka beleférjen a D időegységbe: $\sum_{j=1}^{n} p_j \leq mD$. Ezt m-mel elosztva könnyebben láthatjuk, hogy a D érétket pont így választottuk, tehát ez teljesül. Másrészt szükséges lenne, hogy $p_j \leq D$ igaz legyen, ezzel elkerülve azt, hogy egy hosszabb munka esetleg egy időben több gépen folyamatban van. Ez utóbbi is teljesülni fog, hiszen a legnagyobb p_j-vel rendelkező munkánál sem lehet kisebb a D, mert így választottuk azt.



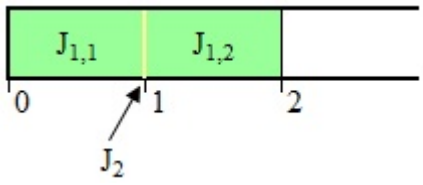
2.1.4 Átlagos (súlyozott) átfutási idő

Az egygépes feladatok sokkal bonyolultabbak, ha a teljes (súlyozott) átfutási időt szeretnénk minimalizálni. Viszonylag korán – 1966-ban – Rothkopf bebizonyította azt a meglepő állítást, miszerint párhuzamos gépek esetén a megszakíthatóság előnye 1.

A továbbiakban először olyan feladatokon keresztül fogjuk a különböző korlátokat megvizsgálni, amelyekben egy gép áll a rendelkezésünkre és adottak az elérhetőségi idők. Epstein és Levin 2016-ban az alsó korlátot dolgozta ki erre a feladatra, a felső korlátnál egy korábbi eredményre támaszkodtak, ahol ez a korlát $\frac{\epsilon}{\epsilon-1} \approx 1.581$. Uniform gépeknél, ha nincsenek súlyok, akkor a felső korlát körülbelül 1.39795 lesz, míg ez két gép esetén 1.2-re módosul Epstein, Levin, Soper és Strusevich szerint.

Láttuk korábban, ha a teljes átfutási időt szeretnénk minimalizálni, akkor nem segít nekünk a megszakíthatóság engedélyezése. Nézzük meg, hogy most, az átlagos átfutási idő minimalizálásánál miért visz minket előrébb. Legyen két munkánk, amelyek közül a hosszabbik $p_1 = 2$ és az elérhetési ideje $r_1 = 0$. A rövidebb munka tulajdonságai legyenek a következők $p_2 = 0$ és $r_2 = 1$. Itt csupán az egyszerűség kedvéért engedjük meg a 0 megmunkálási időt, de nézhetnénk egy egységnyi munkát és egy nagyon nagy munkát is

helyette. Megszakítható esetben az optimális megoldás az alábbi:

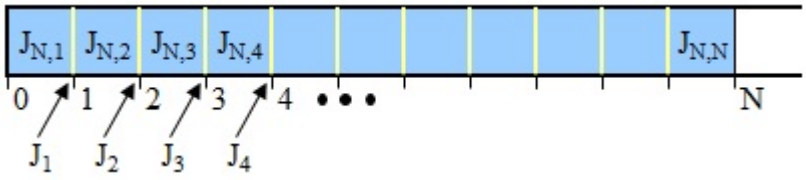


Látjuk, hogy a hosszabbik munkát megszakítjuk egy kis időre annak érdekében, hogy a rövidebbet be tudjuk fejezni. Ekkor $\sum C_j = 1 + 2 = 3$ optimális, mert mindegyik munka $r_j + p_j$ időben készül el.

Ha nem szakíthatjuk meg a munkákat, két lehetőségük van és mindkét esetben látjuk, hogy $\sum C_j = 4$.

Ez egy $\frac{4}{3}$-os alsó korlátot ad nekünk a megszakítás előnyére az átlagos átfutási időre (súlyok nélkül).

Nézzük meg az alsó korlátot, ha a súlyozott átlagos átfutási időt szeretnénk minimalizálni. Legyen egy nagyon nagy munkánk, melynek mérete $p_N = N$ és elérhető $r_N = 0$-tól. Legyen ezen kívül $N - 1$ darab kicsi ($p_j = 0$) munkánk, amelyek elérhetősége $r_j = 1, 2, \ldots, N - 1$. Az előző példához hasonlóan itt is egy optimális megszakítható ütemezés minden munkát a lehető leghamarabb elvégez, azaz $r_j + p_j = j$-re, ami az adott munka indexe is egyben. Az átlagos átfutási idő így $\sum_{j=1}^{N} j\omega_j$ lesz.



Nem megszakítható esetben is hasonlóan ütemezhetünk mint az előbb. N lehetőségünk van, attól függően, hogy a hosszú munka előtt hány rövid

munkát ütemezünk $(0, 1, \ldots, N-1)$. Számoljuk ki az összes esetre ilyenkor a $\sum C_j\omega_j$ értékét. Látjuk, hogy az i. féle ütemezésnél ez a következő lesz:

$$\sum_{j=1}^{i-1} j\omega_j + (N+i-1)(\omega_i + \omega_{i+1} + \cdots + \omega_N).$$

Olyan súlyokat szeretnénk az adott ütemezésekhez rendelni, hogy az összes N-féle ütemezés értéke egyenlő legyen. Így mindegy lesz, hogy hogyan ütemezünk a nem megszakítható esetben. Ehhez válasszuk a súlyokat a következőnek: $\omega_i = a^i$ minden $i = 1, 2, \ldots N-1$-re és $\omega_N = Na^N$, ahol $a = 1 - \frac{1}{N}$. Ekkor, ha N tart a végtelenhez, akkor a megszakítás előnyének a hányadosa $\frac{\epsilon}{\epsilon-1} \approx 1.581$-hez fog tartani.

Súlyok nélkül a nem megszakítható változatú feladatra még régebben Scharge bizonyította, hogy polinomidőben megoldható és az SRPT-sorrend (a munkákat megmunkálási idő szerint nem csökkenő sorrendbe téve) optimális megoldást ad rá.

Uniform gépek esetén megszakítható és nem megszakítható feladatokra is van optimális algoritmus, ha az átlagos átfutási időt szeretnénk minimalizálni. Megszakítható változatú feladat könnyen megoldható a "*staircase*" algoritmussal. Ekkor minden időegység alatt az m legrövidebb munkát tesszük rá a gépekre (ha kevesebb munka van, akkor a leggyorsabb gépeket használjuk). Ezen belül úgy rendezzük a munkákat, hogy a legrövidebb fusson a leggyorsabb gépen, a második legrövidebb a második leggyorsabbon és így tovább. Amikor a legkisebb munka elkészült, a második legkisebb átkerül a leggyorsabb gépre és annak a helyére kerül a harmadik legkisebb munka és így tovább. Ez azért lesz célravezető, mert mindegyik munkát olyan hamar szeretnénk befejezni, ahogy csak tudjuk és nincsenek súlyok, amik befolyásolhatnák ezt az elrendezést. A következő képen egy példát láthatunk a lépcsős algoritmusra, ahol a munkák mérete $p_j = \{4, 6, 7, 11, 13, 14\}$ és így $\sum C_j = 27$.

Ehhez az első fejezethez egy online videós forrást használtam. A *Scheduling seminar* nevű csatorna heti rendszerességgel tart online tanórákat, ahova mindig az adott terület szakembereit hívják meg. Az általam feldolgozott előadást Leah Epstein tartotta, akinek a nevéhez fűződnek a fentebb említett különböző korlátok, amelyeket munkatársaival bizonyítottak [1].

3 Limitált megszakíthatóság

Limitált megszakíthatóságnak azt nevezzük, ha a megadott munkák megszakíthatóak ugyan, viszont a későbbiek folyamán nem lehet azokat áthelyezni másik gépre. Kötelesek vagyunk mindig azon a gépen végezni a munkát, amelyikre először rátettük. Ez életszerű lehet, ha időt vesztünk azzal, hogy átvisszük az adott feladatot például egy másik műhelybe.

3.1 $P2 \mid prec \mid C_{\max}$ *és* $P2 \mid prec,\ pmtn \mid C_{\max}$

Nézzük meg azt a feladatot, amikor két párhuzamos gép áll a rendelkezésünkre. A munkák a megmunkálási időkkel és a megelőzési feltételekkel adottak és a teljes átfutási időt szeretnénk minimalizálni. Hasonlítsuk össze, mi történik az optimumokkal, ha megengedjük a megszakíthatóságot és ha nem. Coffman és Garey [6] bebizonyította, hogy

$$\frac{OPT_N}{OPT_P} \leq \frac{4}{3},$$

ahol az OPT_N a nem megszakítható eset optimumát, az OPT_P ugyanazon feladat megszakítható változata optimumát jelöli. Az alábbi példában legyen három munkánk, azonos 2 egység megmunkálási idővel.

nem megszakítható *megszakítható*

Láthatjuk, hogyha nem megengedett a megszakíthatóság, akkor legalább 4 egység szükséges a három munka elvégzésére, ellenben ha megszakíthatóvá tesszük a feladatokat, már 3 egységnyi időintervallum alatt is be tudjuk fejezni azokat. (A megoldás megengedett is, mert ugyanaz a munka nem fut egy időben két különböző gépen.) Itt a két optimum hányadosa épp az alábbi:

$$\frac{OPT_N}{OPT_P} = \frac{4}{3}.$$

Azaz a korábbi egyenlőtlenség teljesülhet egyenlőséggel is, ezért biztos, hogy nem létezik kisebb felső korlát.

3.2 $P2 \mid prec \mid C_{\max}$ *és* $P2 \mid prec,\ limited\ pmtn \mid C_{\max}$

Az előző esethez képest abban tér el a jelenlegi feladatunk, hogy korlátozzuk a megszakíthatóságát a munkáknak, ezért egyértelmű, hogy a sima megszakításos feladat optimumával vagy megegyezik a jelenlegi optimum vagy nagyobb lesz annál. $OPT_L \geq OPT_P$, ahol az OPT_L a korlátozottan megszakítható feladat optimuma.

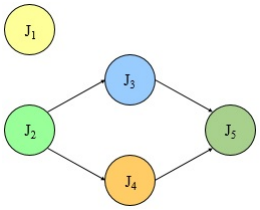
$$\frac{OPT_N}{OPT_L} \leq \frac{OPT_N}{OPT_P} \leq \frac{4}{3}$$

3.3 $P2 \mid prec,\ pmtn \mid C_{\max}$ *és* $P2 \mid prec,\ limited\ pmtn \mid C_{\max}$

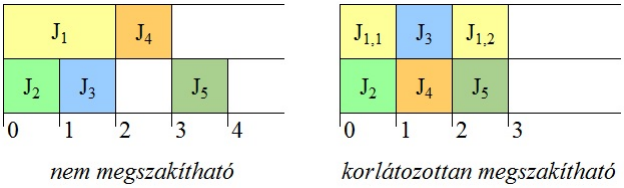
Összehasonlítva a korlátozott megszakíthatóság optimumát a nem megszakítható esetével, nyilvánvaló hogy az utóbbi legalább akkora mint az előbbi. Ezt felhasználva kapjuk az alábbi egyenlőtlenségeket:

$$\frac{OPT_L}{OPT_P} \leq \frac{OPT_N}{OPT_P} \leq \frac{4}{3}$$

Az alábbi példán keresztül szemléltethetjük, hogy $\frac{4}{3}$-nál kisebb felső korlát nem létezik erre. Legyen két gépünk és öt munkánk, amelyek megmunkálási ideje p=[2, 1, 1, 1, 1]. Adottak a megelőzési feltételek is:



Jól látható, hogyha az első munkát nem tudjuk megszakítani, akkor az átfutási idő legalább négy egység lesz, ellenben megszakítással egy egységgel csökkenthetjük $C_{\max}$-ot.



A fejezet kidolgozásakor *A note on limited preemption* című tudományos írásra támaszkodtam, amelyet E. G. Coffman, Jr. és S. Even publikált [5].

4 Amikor a megszakítás nem vezet jobb megoldásra

A következőkben megnézzük, mi történik akkor, ha párhuzamos gépeken szeretnénk egy egységnyi munkákat elvégezni úgy, hogy adottak a rendelkezésre állási idők és az *outtree* feltételek. A befejezési idők összegét szeretnénk minimalizálni, azaz az átlagos átfutási időt vizsgáljuk. Ekkor létezik polinom időben megoldható algoritmusa a feladatunknak, sőt az is bizonyított, hogy ha megszakíthatjuk a munkákat, az optimum nem fog változni.

Legyen n a munkák halmaza ($j = 1, \ldots, n$) és mindegyikre igaz, hogy $p_j = 1$. Ezeket a munkákat szeretnénk m párhuzamos gépen ($M_1, \ldots M_m$) ütemezni. Egy gép egy adott időpillanatban csak egy munkán dolgozhat és bármely munka kerülhet bármelyik gépre. A rendelkezésre állási idők (r_j) egészek. Ezek mellett még megelőzési feltételek is adottak a munkák között, például $i \to j$, amely azt jelenti, hogy a j munkát akkor kezdhetjük csak el,

ha már az i munkát teljesen befejeztük. A következtetéseinket az *outtree* (kifenyő) megelőzésekre fogjuk korlátozni, ahol maximum egy előfeltétele lehet egy munkának. Célunk, hogy egy olyan megengedett megoldást találjunk, ami minimalizálja a befejezési idők összegét, $\sum C_j$-t. Két problémát fogunk megnézni, először a $P \mid p_j = 1,\ r_j,\ outtree \mid \sum C_j$ feladatot, ahol nem engedjük meg a megszakíthatóságot, majd a $P \mid p_j = 1,\ r_j,\ outtree,\ pmtn \mid \sum C_j$ verzióját, ahol megállíthatunk egy munkafolyamatot bármikor és később visszatérhetünk rá egy másik vagy ugyanazon a gépen, ahol előtte dolgoztunk vele, viszont egy munka egy időben csak egy gépen futhat.

Számos más komplexitási eredmény is létezik megszakítható és nem megszakítható párhuzamos gépes problémákra. Például, ha nincsenek megadva a hozzáférési idők, a $P \mid p_j = 1,\ outtree \mid \sum C_j$ feladat megoldható a HU algoritmussal $P \mid p_j = 1,\ tree \mid \sum C_{\max}$ feladatként. Egy érdekes kérdés a megszakítható és a nem megszakítható feladatokra, hogy vajon milyen esetben tud jobb optimális megoldást adni az, ha megengedjük a feladatok megszakítását. McNaughton bebizonyította, hogy a megszakítás nem vezet jobb eredményre akkor, ha a megmunkálási idők tetszőlegesek, megelőzési feltételek nincsenek, se elérhetési idők és a súlyozott összegét szeretnénk minimalizálni a befejezési időknek ($P \mid pmtn \mid \sum w_j C_j$) [8].

Baptiste és Timkovsky 2001-ben komplex bizonyításukkal belátták, hogy a $P2 \mid p_j = 1, r_j,\ outtree,\ pmtn \mid \sum C_j$ feladatnál a megszakítás engedélyezése nem szükséges, nem kapunk jobb megoldást általa. Sőt, azt is sejtették, hogy ez az állításuk több gép esetén is teljesülni fog majd. Ezt a feltevést fogjuk most belátni úgy, hogy először adunk egy polinomiális algoritmust a nem megszakítható változatú feladatra, majd megmutatjuk, hogy a megoldási módszer szintén optimális marad a megszakíthatóság engedélyezése után is. Ez a későbbi bizonyítás jóval egyszerűbb, mint amit ezelőtt beláttak a kétgépes feladatra.

4.1 $P \mid p_j = 1,\ r_j,\ outtree \mid \sum C_j$

Ebben a fejezetben az előbb említett polinomiális algoritmust fogjuk megnézni a nem megszakítható változatú feladatra. Feltesszük, hogy a rendelkezésre állási idők, r_j-k egészek és megfelelnek az *outtree* megelőzési feltételeinek, tehát mivel egységnyi hosszúságú munkáink vannak $r_i + 1 \leq r_j$ teljesül az összes $i \to j$ megelőzési feltétellel rendelkező munkapárra.

A következőkben két relaxációját fogjuk megnézni a feladatunknak. Az első esetben minden megelőzési feltételt kiveszünk, tehát a $P \mid p_j = 1,\ r_j \mid$

$\sum C_j$ feladatot fogjuk megvizsgálni. Legyenek az $\tilde{r}_1 < \tilde{r}_2 < \cdots < \tilde{r}_k$ a különböző rendelkezésre állási idők és definiáljuk $\tilde{r}_{k+1} = \infty$. Jelölje S_v azon munkák halmazát, melyek elérhetési ideje $\tilde{r}_v$. Egy optimális ütemezést lehet az alábbi algoritmussal készíteni.

Legyen S kezdetben egy üres halmaz. Menjünk végig az összes v időponton 1-től k-ig. Az éppen adott $t = \tilde{r}_v$ időpillanatot véve adjuk hozzá a már meglévő S halmazhoz az S_v halmazt. Ezután amíg S nem üres halmaz és a $t < \tilde{r}_v$, legyen $m_t = \min\{m, |S|\}$ és ütemezzük az m_t munkákból álló $S_t \subseteq S$ halmazt t időpillanattól. Miután ezt megtettük, vegyük ki a már ütemezett munkákat az S halmazból, azaz $S = S \setminus S_t$, így megmaradhatnak azok a munkák, amelyeket elérünk az t időpillanatban, de túl kevés gépünk van ahhoz, hogy az összeset ütemezzük ezek közül. Ezután a t-t növeljük eggyel és nézzük meg ismét, hogy S üres halmaz-e, illetve hogy t kisebb-e $\tilde{r}_{v+1}$-nél, azaz még nem jött-e új elérhető munkánk, amit be kéne venni a halmazba.

Az algoritmus azon részében, mikor egy bizonyos v-t vizsgálunk, a munkák az $[\tilde{r}_v, \tilde{r}_{v+1}]$ intervallumon vannak ütemezve. Ezt úgy készítjük el, hogy először hozzáadjuk az S_v halmazt azon munkák halmazához, amelyek még elérhetőek ezen kívül a $t = \tilde{r}_v$ időpillanatban (korábban már a rendelkezésünkre álltak, de eddig még nem került rájuk sor). Ezek közül, azaz az S halmazból a maximális számú munkát ütemezzük a t időpillanattól kezdve az egységnyi hosszúságú időintervallumok mindegyikén (amíg még nem válnak elérhetővé új munkák). Az összes olyan t időpillanatban, ahol nincs definiálva m_t, legyen $m_t = 0$. Ez akkor fordulhat elő, ha már az S halmaz összes elemét ütemeztük, viszont nem vált még elérhetővé új feladat.

Az m_t azt fogja nekünk megmondani, hogy az adott egységnyi hosszúságú intervallumon maximum hány munkát tudunk elvégezni. Ennek az értékét úgy tudjuk meghatározni, hogy megnézzük hány gépünk van, illetve hogy hány munkát kéne az intervallumon elvégezni, és amelyik ezek közül a kisebb, annyit tudunk maximum feltenni a gépekre. Később kivonjuk az ebben az időintervallumban ütemezett munkák számát S-ből, így megtudjuk, mennyi munkánk maradt még hátra és később, a t növelésénél ehhez adjuk hozzá az újonnan elérhetővé váló munkák számát, így biztos, hogy sor kerül az összesre.

A második relaxációban kicseréljük a gépek számát (m) a munkákéra (n). Ebben a könnyítésben az összes munkát tudjuk egy időben, $\tilde{r}_v$-től ($v = 1, \ldots, k$) ütemezni a S_v halmazbeliek közül. A kapott ütemezés megengedett (ha persze az elérhetési idők kompatibilisek a megelőzési feltételekkel) és

biztosan optimális a feladatra, ha n gépünk van. Legyen

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

Ahhoz, hogy egy optimális ütemezést készítsünk az eredeti problémára, módosítsuk a második könnyített feladat ütemezését – amiben tehát $\hat{m}_t$ munka van ütemezve mindegyik $[t, t+1]$ intervallumon – úgy, hogy olyan megengedett megoldást kapjunk, ahol pontosan m_t munka van ütemezve a t periódusban (anélkül, hogy megszegnénk a megelőzési feltételeket). A kapott ütemezés optimális lesz az eredeti feladatra nézve, ha megegyezik az optimuma az első könnyített feladat optimumával. Ezt változtatást úgy tudjuk megtenni, hogy iteratívan mozgatjuk a munkákat balról jobbra. A módosítások alatt változatlannak kell maradnia az alábbiaknak:

1. az *outtree* megelőzési feltételekre figyelünk,
2. $\sum_{v \le \tau} \hat{m}_v \ge \sum_{v \le \tau} m_v$ teljesül az összes τ időpillanatra.

Ezeknek a feltételeknek már a legelejétől kezdve teljesülniük kell.

Feltehetjük, hogy néhány v-re teljesül $\hat{m}_v < m_v$. Különben, $n = \sum_v \hat{m}_v \ge \sum_v m_v = n$ következményeként $\hat{m}_v = m_v$ minden v-re és akkor készen is lennénk. Legyen t a legkisebb időindex úgy, hogy $\hat{m}_t < m_t \le m$. A 2. feltétel miatt lesz egy indexünk $t' < t$ úgy, hogy $\hat{m}_{t'} > m_{t'}$ és $\hat{m}_\tau = m_\tau$, ha $t' < \tau < t$ (mert a 2. pontbeli szummás egyenlőtlenségnek mindig fent kell állnia, így ha van olyan amelyik kisebb, kell lennie olyannak is, amelyik nagyobb). Továbbá az m_t definíciója miatt $m_\tau = m$, ha $t' \le \tau < t$ (különben nem azt választottuk volna t-nek). A t időpont előtt tehát kellett hogy legyen olyan eset, ahol több feladat volt, mint ahány gép. Az előbbiekből következik

$$\hat{m}_{t'} \ge \hat{m}_{t'+1} = \hat{m}_{t'+2} = \cdots = \hat{m}_{t-1} > \hat{m}_t.$$

Sikeresen fogunk munkát mozgatni a $[\tau - 1, \tau]$ intervallumból $[\tau, \tau + 1]$, ha $\tau = t, t-1, \ldots, t'+1$, ha továbbra is teljesül az 1. feltétel. Ezt a következőképpen tudjuk biztosítani.

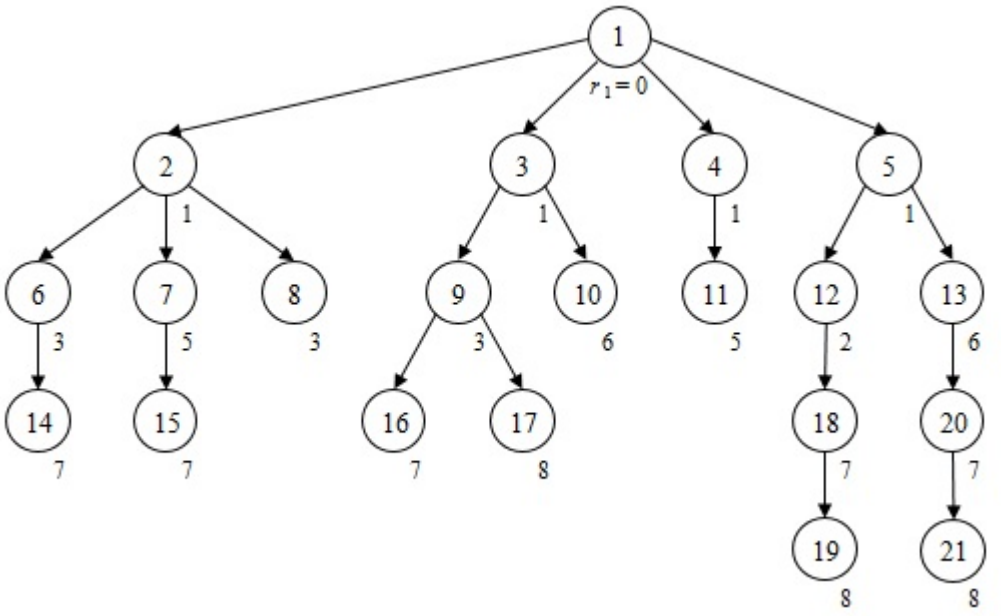
Amíg a $[\tau - 1, \tau]$ időintervallumban legalább eggyel több munka van ütemezve, mint a $[\tau, \tau + 1]$ intervallumban és mindegyik munkának van legalább egy előfeltétele, addig mindig tudunk a $[\tau - 1, \tau]$ intervallumon találni egy olyan munkát, amelynek nincsen utódja a $[\tau, \tau+1]$ intervallumban (mivel

mindegyiknek csak egy szülője lehet), amely munkát ezért az eggyel jobbra levő egységnyi intervallumba tudunk helyezni. Ha $\sum_{v<t'} \hat{m}_v \geq \sum_{v<t'} m_v$ és $\hat{m}_{t'} > m_{t'}$, a második feltétel még mindig teljesülni fog egy munka a $[t', t'+1]$ intervallumból $[t'+1, t'+2]$-ba történő mozgatása után. Ez teljesül az összes $\tau > t'$-re, mert a mozgatással az $\hat{m}_v$ értékek nem csökkennek, ha $v > t'$.

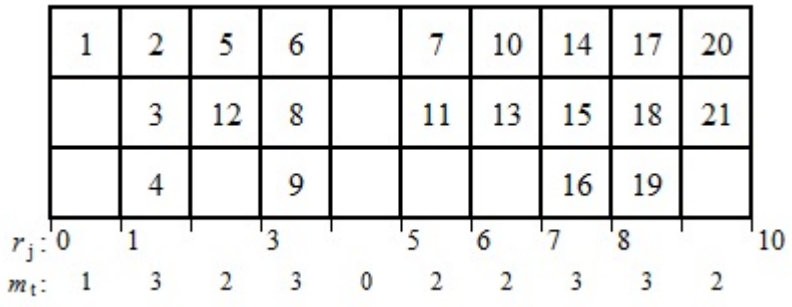
A változtatások egy lépésében a $\max\{0, m_v - \hat{m}_v\}$ értéke eggyel csökken, ha $v = t$ és a többi v értéket változatlanul hagyja. Továbbá $\sum_v \max\{0, m_v - \hat{m}_v\}$ is eggyel csökken minden egyes lépésben. Annak a ténynek a következtében, hogy kezdetben $\sum_v \max\{0, m_v - \hat{m}_v\} \leq \sum_v m_v = n$ teljesült, ezután legfeljebb n lépés után elérkezünk egy optimális megoldásához az eredeti feladatunknak. Mivel minden lépés végrehajtható $O(n)$ időben, ezért az egész feladat $O(n^2)$ időben teljesíthető.

Példa:

Legyen $m = 3$ a gépek száma, a munkáké pedig $n = 21$ és teljesüljenek az alábbi megelőzési feltételek és elérhetési idők a munkákra:

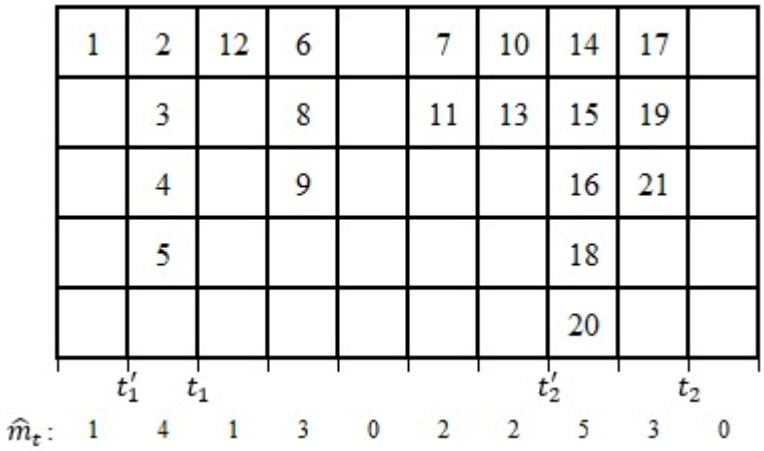


Alkalmazva az előbbi algoritmust első lépésben a $P \mid p_j = 1,\ r_j \mid \sum C_j$ feladatra az alábbi ütemezést kapjuk:

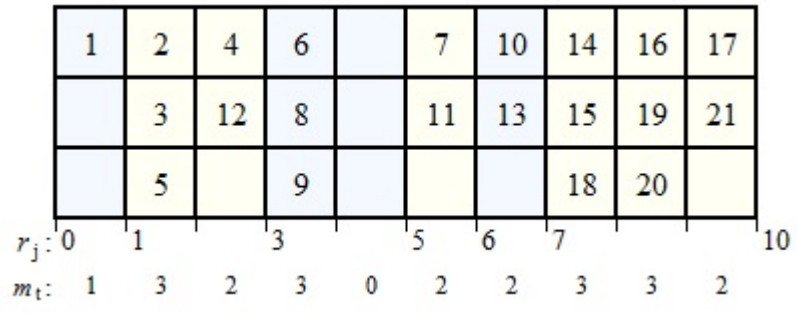


Itt látjuk, hogy mely megelőzési feltételeink nem teljesülnek: $5 \to 12$, $18 \to 19$ és $20 \to 21$.

Egy optimális ütemezést arra az esetre, ha $m = n$ az alábbi ábrán láthatunk:



Ezt módosítva kapjuk az alábbi ütemezést, ahol a következő munkákat mozgattuk: 4-es munkát az $[1, 2]$ intervallumból a $[2, 3]$-ba, a 17-es és 21-es munkákat a $[8, 9]$-ból a $[9, 10]$-be és végül a 16-os és 20-as munkát a $[7, 8]$-ból a $[8, 9]$-be.



4.2 $P \mid p_j = 1,\ r_j,\ outtree,\ pmtn \mid \sum C_j$

Ebben a fejezetben megmutatjuk, hogy a $P \mid p_j = 1,\ r_j,\ outtree \mid \sum C_j$ feladatra létezik optimális ütemezés, amelyben nem szükséges a feladatok megszakítását engedélyezni. Ezenfelül az előbb látott algoritmus megoldja a mostani problémát is. Követeljük meg ehhez a következőket:

1. Az optimális megoldás értéke a relaxált $P \mid p_j = 1,\ r_j,\ pmtn \mid \sum C_j$ feladatra egy alsó korlátot ad a megfelelő $P \mid p_j = 1,\ r_j,\ outtree,\ pmtn \mid \sum C_j$ feladatnak.

2. Ugyanazon példákra a $P \mid p_j = 1,\ r_j,\ pmtn \mid \sum C_j$ és a $P \mid p_j = 1,\ r_j \mid \sum C_j$ feladatok optimális megoldása ugyanaz.

3. Adott egy I példa a $P \mid p_j = 1,\ r_j,\ outtree \mid \sum C_j$ feladatra, ahol a rendelkezésre állási idők kompatibilisek az *outtree* megelőzési feltételekkel. Egy optimális megoldása a megfelelő $P \mid p_j = 1,\ r_j \mid \sum C_j$ feladatnak ugyanazon elérhetőségi időkkel, átalakítható I-nek egy megengedett megoldásává anélkül, hogy az értéket megváltoztatnánk.

Az első pont egyértelmű, mivel ha több feltételt szabunk ki a feladatban, akkor az optimum értéke csak nőhet vagy ugyanaz maradhat. A harmadik pontot az előző fejezetben beláttuk. Ahhoz, hogy a második pontban leírtakat is bebizonyítsuk, szedjük szét blokkokra a $P \mid p_j = 1,\ r_j \mid \sum C_j$ – vagy a megfelelő $P \mid p_j = 1,\ r_j,\ outtree \mid \sum C_j$ – feladatra alkalmazott algoritmus által létrehozott ütemezést. Ezek a blokkok a következők alapján legyenek definiálva.

Legyenek $i_0 = 1 < i_1 < \cdots < i_q$ indexei v-k, $v = 0, \ldots, q-1$.

- i_{v+1} az első olyan index, amely nagyobb i_v-nél, oly módon, hogy az algoritmus az összes j munkát úgy ütemezi, hogy $\tilde{r}_{i_v} \leq r_j \leq \tilde{r}_{i_{v+1}}$ az $[\tilde{r}_{i_v}, \tilde{r}_{i_{v+1}}[$ intervallumon belül, és

- legalább egy gép áll minimum egy egységnyit az $[\tilde{r}_{i_v}, \tilde{r}_{i_{v+1}}[$ intervallumon belül.

Vegyük figyelembe, hogy a v blokkban lévő munkák nem lehetnek hamarabb ütemezve, mint a megfelelő $[\tilde{r}_{i_v}, \tilde{r}_{i_{v+1}}[$ intervallum, mivel az elérhetőségi idejük nagyobb vagy egyenlő $\tilde{r}_{i_v}$-vel. Az $[\tilde{r}_{i_v}, \tilde{r}_{i_{v+1}}[$ intervallumokhoz tartozó részütemezéseket hívjuk blokkoknak. Az előbbi példában az előző képen láthatjuk, hogy 6 blokkunk van a $[0,1[, [1,3[, [3,5[, [5,6[, [6,7[, [7,10[$ intervallumokon.

Most már a blokkokra indukcióval beláthatjuk a 2-es pont állítását. Ha csak egy blokk létezik, akkor nézzük azt a relaxált feladatot, amelyben a rendelkezésre állási időket figyelmen kívül hagyjuk ($P \mid p_j = 1,\ pmtn \mid \sum C_j$). McNaughton megmutatta, hogy a megszakíthatóság nem szükségszerű, ha a blokkra létezik egy optimális nem megszakítható ütemezés. Az így kapott ütemezés teljesen beleillik a blokkunk által meghatározott időintervallumba. Továbbá egy optimális megoldása a $P \mid p_j = 1 \mid \sum C_j$ feladatnak ugyanolyan felépítéssel rendelkezik, mint a $P \mid p_j = 1,\ r_j \mid \sum C_j$ feladat algoritmus által létrehozott ütemezése, amely megengedett, sőt optimális is a $P \mid p_j = 1,\ r_j,\ pmtn \mid \sum C_j$ feladatra.

Feltehetjük, hogy ez igaz $k-1$ blokkig és nézzük meg mi történik a k. blokknál. Gondoljunk a relaxált változatára a feladatnak, amelyikben az első blokk és az azt követő $k-1$ külön van optimálisan ütemezve. Indukcióval látjuk, hogy a megszakíthatóság redundáns minden egyes alproblémára és ezek ütemezése beleillik a blokkjuknak megfelelő időintervallumukba. Ezen felül egyik ütemezésnek sincs átfedése, szóval az így összerakott ütemezés megfelelő lesz k blokkra is. Tehát az blokkokból összeállított ütemezés optimális a megszakítható és a nem megszakítható feladatra is.

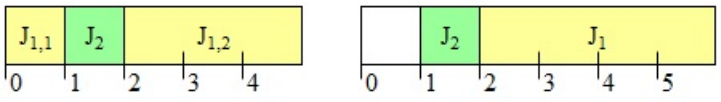
A fejezet elkészítésekor egy P. Brucker, J. Hurink és S. Knust által írt cikkre támaszkodtam [3].

5 További példák a megszakíthatóság redundanciájára

Az alábbi fejezetben olyan eseteket fogunk megnézni, amelyekben a megszakítás hiába ad nekünk nagyobb szabadságot, mégsem segít minket a jobb ütemezés elérésében. Azaz ugyanaz a megszakítható és a nem megszakítható feladat optimuma. A rendelkezésre álló inputokat tekintsük egész számoknak.

Egygépes feladatokat nézve például, ha nincsenek rendelkezésre állási idők vagy megelőzési feltételek, tehát a kezdő időpillanattól elérhető az összes munka és bármilyen sorrendben ütemezhetőek, akkor könnyű átgondolnunk, hogy nem segít nekünk a megszakíthatóság, hiszen állásidő nélkül ütemezünk és $C_{\max} = \sum p_j$ mindig. A szokásos jelölést használva: $1 \mid\mid C_{\max}$ optimuma megegyezik az $1 \mid pmtn \mid C_{\max}$ optimumával. Bonyolultabb a feladat, ha a rendelkezésre állási idők is adottak ($1 \mid r_j \mid C_{\max}$). Meg lehet mutatni, hogyha a megmunkálási idők mindegyike egy időegységnyi ($1 \mid r_j,\ p_j = 1 \mid C_{\max}$), akkor a megszakítás szintén nem vezet jobb megoldásra, mert akkor van értelme megállítani egy munkát, ha egy új érkezik, viszont erre nem lesz szükség, hiszen a megmunkálási idők mindegyike 1 egységnyi és a rendelkezésre állási idők egészek.

Ezzel szemben viszont könnyedén tudunk olyan egygépes kisebb példákat mutatni, amikor segít nekünk a megszakítás. Ilyen az $1 \mid r_j \mid L_{\max}$ és az $1 \mid r_j \mid \sum C_j$ feladat is. Legyenek a megmunkálási idők p=[4,1], a rendelkezésre állási idők r=[0,1] és a határidők d=[5,2]. Ekkor az alábbi ábráról leolvasva láthatjuk, hogy a megszakítás segített nekünk, kisebb lett az átfutási idő vagy a késés.



megszakítható *optimális nem megszakítható*

A megszakítható esetben az $L_{\max} = 0$, a másik feladatban már $L_{\max} = 1$, ugyanígy a megmunkálási idők összegére nézve az első esetben $\sum C_j = 7$ az optimum, ha megszakíthatóak a munkák, míg ha nem, akkor $\sum C_j = 8$.

Párhuzamos gépek esetén más a helyzetünk, mert miután egy munkát megszakítottunk, lehetséges, hogy később már másik gépre kerül át. Itt is tudunk olyan esetet találni, ahol segít nekünk a megszakítás, például a $P2 \mid p = 2 \mid C_{\max}$ feladatban, amelyet az előbbi 3.1-es fejezetben már láttunk.

5.1 $P \mid p_j = 1,\ r_j \mid \sum w_j U_j$ *és* $P \mid p_j = 1,\ intree \mid \sum w_j U_j$

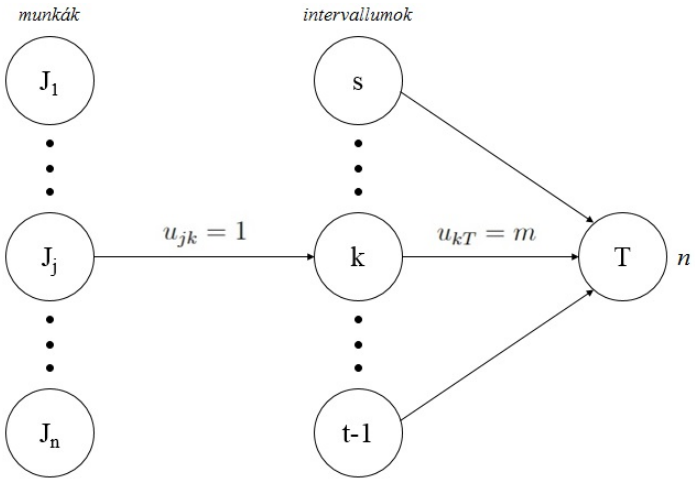
Vizsgáljuk párhuzamos gépeken az egységnyi hosszú munkák ütemezését. Először a rendelkezésre állási idők adottak és a késő munkák súlyozott összegét szeretnénk minimalizálni.

A feladat megoldásához először nézzük meg a *time window* problémát, amikor minden j munkának tudjuk a határidejét ($P \mid p_j = 1,\ r_j,\ d_j \mid -$) és szeretnénk ezeket elvégezni az $[r_j,\ d_j]$ időintervallumban, mert ha ezt meg tudjuk tenni, akkor létezik megengedett megoldása az ütemezésünknek, ahol mindegyik feladat a megfelelő időintervallumban készül el. Akkor van megoldása a *time window* problémának, ha a megfelelő megszakítható feladatnak is. Ezt felhasználva belátható, hogy az eredeti ütemezési feladatunkban a megszakíthatóság hozzávétele nem segít nekünk az optimum csökkentésében.

Lemma: A *time window* problémának ($P \mid p_j = 1,\ r_j,\ d_j \mid -$) van megengedett megoldása akkor, és csak akkor, ha a megfelelő megszakítható feladatnak van megengedett megoldása.

Bizonyítás:

A megszakítható változatú feladatunk ($P \mid p_j = 1,\ r_j,\ d_j,\ pmtn \mid -$) megegyezik az alábbi folyam feladattal, ahol a maximum kapacitása a (j,k) élnek éppen u_{jk}:



Az első oszlopban a munkákat láthatjuk, amelyekről tudjuk, hogy mindegyik egységnyi hosszú, ezek lesznek a források. A második oszlop csúcsai időintervallumokat jelölnek, mégpedig úgy, hogy a $[k, k+1[$ idősávnak a k. csúcs felel meg. A legkisebb ilyen k-t az összes munka közül a legkorábban hozzáférhető elérési ideje határozza meg ($s := \min r_j$), míg a legnagyobbat a legkésőbbi határidő időpontja ($t := \max d_j$). Az ábrán azért $t-1$-ig számozzuk az időintervallumokat jelölő csúcsokat, mert ez a $[t-1,\ t[$ -et jelöli és t-ig mehetünk el maximum, az a legkésőbbi határidőnk. Akkor van él a munkák és az időintervallumok között, tehát $(j,\ k)$ él akkor létezik, ha a j munka elérési időpillanata legalább k és a határideje legfeljebb $k+1$ ($r_j \leq k \leq d_j - 1$). Ekkor ennek a kapacitása $u_{jk} = 1$, ezzel lehetővé tesszük, hogy megszakítás nélkül is teljesíteni tudjuk az adott feladatot egy egységnyi időintervallum alatt. Berajzoltunk a végére egy nyelő csúcsot is (T-t), ahova minden időintevallumhoz tartozó pontból megy él és mindegyik kapacitása éppen m, $u_{kT} = m$. Ez azért szükséges, hogy egyszerre, egy egységnyi időintervallumon belül az m gépen futtatható $1 * m$ egységnyi kapacitásunkat ne tudjuk túllépni.

Látjuk, hogy akkor van megoldása a megszakítható változatú ütemezési feladatnak, ha létezik megfelelő folyam ezen a gráfon. Ezt jelöljük x_{jk}-val, ami tehát azt mondja meg nekünk, hogy a j csúcsból mennyi egységnyi folyik át a k csúcsba ebben a folyamban. Ekkor létezik ehhez megengedett ütemezés is úgy, hogy a j munkát x_{jk} ideig hagyjuk a $[k,\ k+1]$ időintervallumban a gépeken. Ezt meg tudjuk tenni a McNaughton algoritmussal (mindegyik k intervallumra). Elsősorban azért, mert teljesül, hogy semelyik munka sem hosszabb az intervallum hosszánál, így nem fog egyszerre több gépen futni ($x_{jk} \leq 1$), illetve nem léphetjük túl az adott intervallumban lefuttatható munkák maximumát ($m * 1$) az ide kerülő munkákkal, mert

$$\sum_{j=1}^{n} x_{jk} \leq m.$$

A lemma abból az állításból következik, hogy ha van a folyam feladatnak megengedett megoldása, akkor van egész számokkal vett megoldása is.[4] □

Tétel: A $P \mid p_j = 1,\ r_j \mid \sum w_j U_j$ feladatnál a megszakíthatóság engedélyezése nem vezet jobb megoldásra.

Bizonyítás:

Vegyük egy optimális megoldását a $P \mid p_j = 1,\ r_j,\ pmtn \mid \sum w_j U_j$ feladatnak és legyen S azon munkák halmaza, amelyeket időben be tudunk fejezni. Az előbbi lemmából következik, hogy egy nem megszakítható ütemezésben is el tudjuk készíteni határidőre az S-beli munkákat, illetve ekkor a nem megszakítható feladat optimuma szintén optimális a megszakítható változatú feladatra nézve is. □

A következő feladatban használni fogjuk a be-fenyő (*intree*) kifejezést, amellyel a megelőzési feltételeinket tudjuk korlátozni, mégpedig úgy, hogy az a csúcs, amelyik legutoljára végződő munkát jelöli, bármelyik másik pontból elérhető és rajta kívül az összes többi csúcsból egy él indul ki. Tehát egy munka (a legutolsót leszámítva) csak egy másik munkának lehet a közvetlen megelőzési feltétele.

Lemma: A $P \mid p_j = 1,\ intree \mid \sum w_j U_j$ feladatra a megszakítás megengedése nem vezet jobb eredményre.

Bizonyítás:

Legyen S a $P \mid p_j = 1,\ intree,\ pmtn \mid \sum w_j U_j$ feladat optimális ütemezése és legyen ekkor L az időben befejezett munkák halmaza. Legyen S' azon részütemezése S-nek, amely az L halmaz munkáiból áll és megoldja ezáltal a *time window* problémát a $[0,\ d_j]$ időintervallumokra. Jelöljük k-val a legkésőbbi határidőt az L-beliek közül ($k = \max_{i \in L} d_i$). Ezután, ha a fordított ütemezését vesszük az S'-nek, akkor az megoldja a $P \mid p_j = 1,\ r_j = k - d_j,\ outtree,\ pmtn \mid C_{\max} \leq k$ feladatot (ezt nevezzük el Q-nak). Mivel az S' ütemezésben mindegyik munka befejeződött a határidejéig és azon túl biztos nem nyúlt, ezért a fordított ütemezésben elérhetővé legkorábban a korábbi határidejétől válhat egy munka az új ütemezésben. Ez az időpillanat k-tól, azaz a mostani kezdőpillanattól $k - d_j$ időegységnyire található.

Brucker és szerzőtársai bizonyításához hasonlóan – amit a 4-es fejezetben láthattunk – meg lehet mutatni, hogy a $P \mid p_j = 1,\ r_j,\ outtree,\ pmtn \mid C_{\max} \leq k$ feladatnak van optimális nem megszakítható megoldása.[3] Továbbá Q-nak is van nem megszakítható megoldása, amelynek a megfordított ütemezése egy nem megszakítható optimális ütemezése a $P \mid p_j = 1,\ intree,\ pmtn \mid \sum w_j U_j$ feladatnak. □

5.2 $P \mid p_j = 1 \mid \sum T_j$

Ebben a fejezetben a $P \mid p_j = 1 \mid \sum T_j$ feladatról fogjuk belátni, hogy a megszakíthatóság lehetővé tétele nem vezet jobb megoldásra. Ennek a bizonyítása megszakítható és nem megszakítható feladatok optimumának több tulajdonságán alapszik, amelyeket a következő lemmákban fogunk megnézni. Az egyszerűség kedvéért a határidejük alapján számozzuk a munkákat $d_1 \leq d_2 ... \leq d_n$.

Lemma: A $P \mid p_j = 1, \mid \sum T_j$ feladatnak létezik optimális megoldása úgy, hogy $C_1 \leq C_2 ... \leq C_n$.

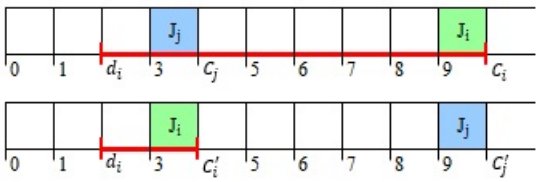
Bizonyítás:

Legyen $d_i \leq d_j$, viszont indirekt tegyük fel, hogy $C_j < C_i$. Ha az i és a j munkát kicseréljük, akkor a T_i értéke az alábbival csökken:

$$\Delta_1 = \max\{0, \min\{C_i - C_j, C_i - d_i\}\}$$

Itt azért kell egy bizonyos érték és a nulla szám maximumát néznünk, mert ha az i. munka idő előtt elkészül már az első esetben is, akkor a minimum a $C_j - d_i$ érték lesz, ami negatív, de az nem lehetséges. Belegondolva, ha a megcserélés előtt is már $T = 0$, akkor hamarabb befejezve az i. munkát sem lesz jobb eredményünk, ugyanúgy 0 marad a T.

Nézzük meg, hogy mi történik akkor, ha nem tudjuk befejezni határidőre az i. munkát. Ekkor két estünk van. Az első, ha megcserélve a két feladatot már időre el tudjuk készíteni az i munkát (azaz $C_j \leq d_i$ és $T_i' = 0$), akkor a régi késés értékével tudjuk csökkenteni a mostanit, azaz $C_i - d_i$-vel. A másik esetben, hiába hozzuk előrébb az i. munkát, így sem tudjuk időre elkészíteni azt. Ekkor a régi késés $C_i - d_i$ volt, a mostani $C_i' - d_i = C_j - d_i$, mivel mindegyik munka egységnyi hosszú. Ezeket kivonva egymásból $[C_i - d_i - (C_j - d_i) = C_i - C_j]$ kapjuk a másik tagot. Például: $d_i = 2,\ C_i = 10,\ C_j = 4$



Ekkor a két munkát megcserélve: $C_i' = 4,\ C_j' = 10$. Itt jól látszik az ábrán, hogy a két piros szakasz hosszának a különbsége éppen $C_i - C_j$.

A másik esetnél is láthatjuk az alábbi példában, ahol $d_i = 7$, hogy felcserélés után már időben befejeződik az i. munka, ezért a csere előtti késéssel fog csökkenni T_i, amelynek értéke $C_i - d_i$.

Hasonlóan átgondolva a T_j értéke a következővel csökken:

$$\Delta_2 = \max\{0, \min\{C_i - C_j, C_i - d_j\}\}$$

Tudjuk, hogy $d_i \leq d_j$, ezért $\Delta_2 \leq \Delta_1$, így ilyen cseréket végrehajtva egy optimális megoldáson, az optimum nem nőhet. □

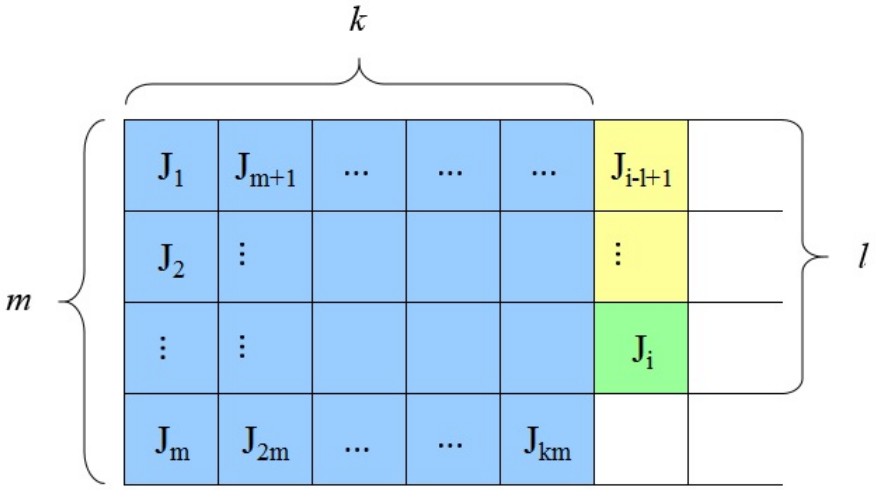
Nézzük meg, hogy miért nem vezet jobb megoldásra a megszakíthatóság engedélyezése a $P \mid p_j = 1 \mid \sum T_j$ feladatnál.

Legyen $\overline{S}$ egy optimális megoldása a nem megszakítható feladatunknak és legyen a munkák befejezési ideje $\overline{C}_1, \ldots, \overline{C}_n$. Az előbb beláttuk, hogy ez az optimális megoldás lehet a határidők szerint nem csökkenő sorrendben történő ütemezéssel is. Ekkor az i. munkát felírhatjuk a következők szerint: $i = km + l$, ahol $1 \leq l \leq m$ és $k \geq 0$. Az i. munka hamarabb nem kerülhetett a gépre, mert előtte egyetlen gép sem volt szabad. Az i-t megelőző k időintervallumban $k * m$ munkát tudtunk befejezni, illetve a mostani időintervallumban az l. helyre kerül az i felülről nézve, azaz az l. gépen hajtjuk végre azt. A befejezési ideje ilyenkor az alábbi: $\overline{C}_i = k + 1$. Sőt, az előzőek alapján $i/m \leq \overline{C}_i = \lceil i/m \rceil$, ahol az egyenlőség akkor teljesül, ha épp az utolsó (m.) gépre tettük a munkát. Továbbá, ha $i \geq m$, akkor az alábbi összegzést kapjuk:

$$\sum_{j=i-m+1}^{i} \overline{C}_j = l(k+1) + (m-l)k = i.$$

Itt m darab befejezési időt adunk össze, mégpedig az i-től visszafelé haladva, önmagát is belevéve, éppen ezért a j-t legkorábban az első oszlop utolsó – m. gépen lévő – munkájától számíthatjuk, mert ennél korábban nem lesz (indexelés szerint) az adott munka előtt már $(m-1)$ ütemezett feladatunk.

Az i-vel egy oszlopban lévők befejezési ideje megegyezik, mindegyiké $k+1$ és összesen l ilyen munka van. Ez adja az első tényezőt. A második tag az i előtti oszlopban az m darab munkából még fennmaradó $(m-l)$ munka befejezési idejét adja össze, melyek mindegyike k. Az alábbi ábrán könnyebben elképzelhetjük ezeket:



Másrészt legyen $\tilde{S}$ egy tetszőleges optimális megoldása a megfelelő megszakítható változatú feladatnak $(P \mid p_j = 1,\ pmtn \mid \sum T_j)$ és legyenek a befejezési idők $\tilde{C}_1, \ldots, \tilde{C}_n$. Az előbb beláttak alapján ekkor feltehetjük, hogy $\tilde{C}_1 \leq \cdots \leq \tilde{C}_n$.

Lemma:

$$\tilde{C}_i \geq \max\{1, i/m\}$$

és

$$\sum_{j=i-m+1}^{i} \tilde{C}_j \geq i,\ i \geq m$$

Bizonyítás:

A lemma első része egyértelműen következik az elérhető gépek kapacitásából. Amikor az i. munka befejeződik, már az összes előtte lévő is véget ért, amelyek összmegmunkálási ideje i, mert egységnyi hosszúságúak. Ezek elvégzésére legalább i/m időegység szükséges, de mivel megszakíthatóak a munkák, lehet hogy az i-nél nagyobb indexű munkába is belekezdtünk már,

ezért ez egy alsó korlátot ad nekünk. Az i/m hányados, ha kisebb 1-nél ($1 \leq i < m$), azaz a legelső időintervallumban szerepelnek, akkor a legjobb esetben is ezen i munkák megszakítás nélkül ütemezve legkorábban az 1 időpillanatban lehetnek kész, ezért kell az 1 és az i/m maximumát néznünk.

A lemma második részének bizonyításánál első lépésként $i = n$-re nézzük meg az állítást. Jelölje L_j az $\tilde{S}$ ütemezésben a j gép ($j = 1, \ldots, m$) leállási idejét, amikor már nem kerül rá több munka. Feltehetjük, hogy a j. gépen állásidő nélkül ütemezünk a $[0, L_j]$ időintervallumban és a leállási idők az alábbi sorrendben követik egymást: $L_1 \geq \cdots \geq L_m$. Tudjuk, hogy

$$\sum_{j=1}^{m} L_j = \sum_{j=1}^{n} p_j = n,$$

ezért elegendő belátnunk, hogy

$$\sum_{j=n-m+1}^{n} \tilde{C}_j \geq \sum_{j=1}^{m} L_j = i.$$

Mivel az első gép dolgozik a legtovább, ezért a j. gép leállásának pillanatáig az összes $1, \ldots j$ gép még foglalt, sőt az L_j időpillanatig még legalább j munka nincs kész (különben már valamelyik gép leállt volna és egyszerre egy munka csak egy gépen lehet), ezért a j. legnagyobb befejezési idő legalább L_j:

$$\tilde{C}_{n-j+1} \geq L_j.$$

Ebből már következik a fentebbi egyenlőtlenség.

A többi esetre – amikor $i < n$ – gondolhatunk az $\tilde{S}$ egy részütemezéseként, ahol csak 1-től i-ig szerepelnek a munkák. Ezt könnyen átalakíthatjuk egy új ütemezéssé, ahol az $1, \ldots, i$ munkákat állásidő nélkül ütemezzük úgy, hogy a befejezési idők nem csökkenő sorrendben követik egymást az indexelés szerint. Ekkor az előbbi bizonyítást erre is levezethetjük és így tovább. □

A továbbiakban megmutatjuk, hogy az $\overline{S}$ megoldás nem rosszabb az $\tilde{S}$-nél. Ennek a bizonyításnak az alapjául a következő lemma fog szolgálni, ami megmutatja, hogy ha van egy olyan i munka, amelynek késése az $\overline{S}$ ütemezésben nagyobb, mint az $\tilde{S}$-ben, akkor találhatunk egy r indexet ($r < i$) úgy, hogy az összkésése az $r, r+1, \ldots, i$ munkáknak az $\overline{S}$ ütemezésben maximum akkora lehet, mint ezen összeg az $\tilde{S}$ ütemezésben nézve.

Lemma: Minden i munkához tudunk találni egy olyan r munkát ($i-m+1 \leq r \leq i$), amire teljesül, hogy

$$\sum_{j=r}^{i} \overline{T}_j \leq \sum_{j=r}^{i} \tilde{T}_j.$$

(Ahol a $\overline{T}_j$ az $\overline{S}$ ütemezésen belüli késést jelöli, a $\tilde{T}_j$ pedig az $\tilde{S}$-hez tartozót.)

Bizonyítás:

Esetszétválasztást alkalmazunk.

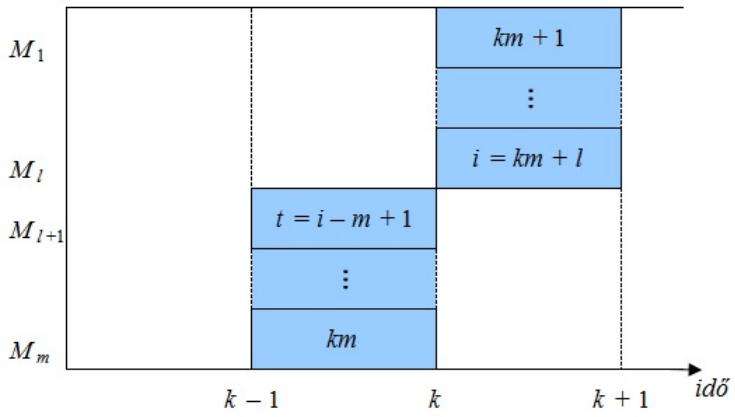
1. Ha $\overline{C}_i \leq \tilde{C}_i$, akkor i-nek választva az r-et pont a lemmát kapjuk eredményül.

2. Ha $\overline{C}_i > \tilde{C}_i$ és i nem késik az $\tilde{S}$ ütemezésben, akkor az előző lemma és az azt megelőző egyenlőség ($\overline{C}_i = \lceil i/m \rceil$) alapján a következőt kapjuk

   $$\overline{C}_i = \lceil i/m \rceil \leq \lceil \tilde{C}_i \rceil,$$

   mert a lemma szerint $i/m \leq \tilde{C}_i$, ezért ez az egyenlőtlenség a felső egészrészüket véve is teljesülni fog. Mivel d_i egész szám, ezért ez azt eredményezi, hogy az i nem késik $\overline{S}$ ütemezésben sem. Ha $r = i$-t választjuk, akkor a lemma eredményét kapjuk.

3. Az utolsó eset, ha $\overline{C}_i > \tilde{C}_i$ és i késik az $\tilde{S}$ ütemezésben. Ahhoz, hogy meghatározzuk az r indexet, nézzük csak a következő m munkát $t := i-m+1, i-m+2, \ldots, i = km+l$. Ezen munkák részütemezése az $\overline{S}$-en belül a következő ábrán látható.



Amíg i késik $\tilde{S}$-ben, a következőt kapjuk:

$$d_i < \tilde{C}_i < \overline{C}_i = k + 1,$$

sőt az alábbi is teljesülni fog:

$$d_t \leq d_{t+1} \leq \cdots \leq d_i \leq k \leq \overline{C}_t \leq \cdots \leq \overline{C}_i,$$

ezért a $t, \ldots i$ egyike se készül el időben $\overline{S}$-ben.

Továbbá legyen $\overline{T}$ a következő:

$$\overline{T} := \sum_{j=t}^{i} \overline{T}_j = \sum_{j=t}^{i} \max\{0, \overline{C}_j - d_j\} =$$

(Ennek a maximuma mindig a $\overline{C}_j - d_j$, mivel ez sosem lesz negatív, mert $\overline{C}_j$ legalább akkora, mint d_j.)

$$= \sum_{j=t}^{i} (\overline{C}_j - d_j) = \sum_{j=t}^{i} \overline{C}_j - \sum_{j=t}^{i} d_j = i - \sum_{j=t}^{i} d_j$$

Az előbb már beláttuk, hogy $\sum_{j=i-m+1}^{i} \overline{C}_j = i$, innen a legutolsó következtetés.

(a) Ezen belül az első eset, ha minden $j = t, \ldots, i$ munkára teljesül, hogy $\tilde{C}_j \geq d_j$. Ekkor ezek közül semelyik munka sincs időben befejezve az $\tilde{S}$ ütemezésben és a következőt kapjuk:

$$\tilde{T} := \sum_{j=t}^{i} \tilde{T}_j = \sum_{j=t}^{i} \max\{0, \tilde{C}_j - d_j\} = \sum_{j=t}^{i} \tilde{C}_j - \sum_{j=t}^{i} d_j \geq i - \sum_{j=t}^{i} d_j$$

Itt az utolsó egyenlőtlenség szintén az előző lemmából következik. Az előző két állításból következik, hogy $\tilde{T} \geq \overline{T}$, ami a késések összegét adja a $t, \ldots, i$ munkáknak $\overline{S}$ ütemezésben, amely tehát nem haladja meg az $\tilde{S}$ ütemezésben lévő késések összegét. Az r-et $t = i - m + 1$-nek választva teljesül a lemmabeli állítás erre az alesetre vonatkoztatva.

(b) Az utolsó eset, ha létezik olyan s munka a $\{t, \ldots, i-1\}$-beliek közül, amelyre teljesül $\tilde{C}_s < d_s$, azaz az s időben befejeződik $\tilde{S}$-ben. Jelölje ez az s a legnagyobb indexű ilyen munkát. Ebből következik, hogy az s-nél nagyobb indexű ($j = s+1, \ldots, i$) munkákra teljesül, hogy $\tilde{C}_j \geq d_j$. Továbbá:

$$\tilde{C}_j \leq \tilde{C}_s < d_s \leq k \leq \overline{C}_j,$$

ahol $j = t, \ldots, s$.
Ebből következik:

$$\sum_{j=s+1}^{i} \tilde{T}_j = \sum_{j=s+1}^{i} \tilde{C}_j - \sum_{j=s+1}^{i} d_j = \sum_{j=t}^{i} \tilde{C}_j - \sum_{j=t}^{s} \tilde{C}_j - \sum_{j=s+1}^{i} d_j$$

$$\geq i - \sum_{j=t}^{s} \tilde{C}_j - \sum_{j=s+1}^{i} d_j = \sum_{j=t}^{i} \overline{C}_j - \sum_{j=t}^{s} \tilde{C}_j - \sum_{j=s+1}^{i} d_j$$

$$= \sum_{j=s+1}^{i} (\overline{C}_j - d_j) + \sum_{j=t}^{s} (\overline{C}_j - \tilde{C}_j) \geq \sum_{j=s+1}^{i} \overline{T}_j$$

Ha r-et $s+1$-nek választjuk, akkor pont a lemmában szereplő eredményt kapjuk. □

Az előző lemmát felhasználva beláthatjuk, hogy a $P \mid p_j = 1 \mid \sum T_j$ feladatnál a megszakíthatóság engedélyezése nem csökkenti az optimumot.

Tétel: A $P \mid p_j = 1 \mid \sum T_j$ feladatra a megszakíthatóság nem vezet jobb ütemezésre.

Bizonyítás:

Tegyük fel indirekt, hogy létezik olyan megszakítható ütemezés, amely jobb a nem megszakítható feladat optimumánál. Legyen $\tilde{S}$ a jobb ütemezés, melyben az i minimális értékekre teljesül

$$\sum_{j=i+1}^{n} \tilde{T}_j \geq \sum_{j=i+1}^{n} \overline{T}_j.$$

Az előző lemmát felhasználva kapunk egy olyan $r \leq i$ indexet, amelyre $\sum_{j=r}^{i} \tilde{T}_j \geq \sum_{j=r}^{i} \overline{T}_j$. Ebből következik:

$$\sum_{j=r}^{n} \tilde{T}_j \geq \sum_{j=r}^{n} \overline{T}_j.$$

Ez ellentmond i minimalitásának, mert n-re is teljesül, így nem igaz az a feltevés, hogy létezik jobb megszakítható ütemezés. □

5.3 $P2 \mid p_j = 1,\ r_j \mid \sum w_j T_j$

Ebben a fejezetben megnézzük, hogy mi történik akkor, ha két gépünk van, mindegyik munka egységnyi hosszú, elérhetőségi idejüket is ismerjük és a súlyozott, nemnegatív késésüket szeretnénk minimalizálni. Azt állítjuk, hogy a megszakíthatóság nem vezet ilyenkor se jobb eredményre. Először tegyük fel, hogy csak három munkánk van és vizsgáljuk csak ezekre az állítást.

Lemma: Minden esetben létezik három munkára nézve a $P2 \mid p_j = 1,\ pmtn \mid \sum w_j T_j$ feladatnak nem megszakítható optimális megoldása.

Bizonyítás:

Feltehetjük hogy a munkák a határidejük szerint vannak rendezve, $d_1 \leq d_2 \leq d_3$. Tekintsük az alábbi három esetet:

1. Első esetben legyen a $d_3 = 0$, azaz mindhárom munka határideje 0. Ekkor $\sum \omega_j T_j = \sum \omega_j C_j$ és az eredmény McNaughton tételéből következik [8].

2. A második esetben legyen $d_3 \geq 2$. Tudunk adni ilyenkor is optimális ütemezést. Az első két munka határideje maximum d_3 és lehet hogy kisebbek 2-nél és esetleg késni fognak (ha $d_1 = 0$ vagy $d_2 = 0$), viszont mindkettőt a kezdő időpillanattól indítani tudjuk, ennél gyorsabban nem lehet befejezni őket. Ekkor az első és a második munkát a $[0, 1]$ időintervallumban végezzük el és a harmadik munka az $[1, 2]$ intervallumban kerül a gépre. Tehát az utolsó munka késése $T_3 = 0$ lesz és az első két munka optimálisan lesz ütemezve (még megszakítható esetben is).

3. Harmadik esetben nézzük meg, mi történik $d_3 = 1$ esetén. Tegyük fel, hogy k a legkisebb súllyal rendelkező az $\{1, 2, 3\}$ munkák közül. Ekkor ennek a munkának kell hátulra kerülnie, azaz az $[1, 2]$ időintervallumba, a másik kettő pedig a $[0, 1]$-ban végezzük el, hogy optimális ütemezést kapjunk. Ez abból következik, hogyha k hamarabb végződne, akkor a $\omega_k T_k$ csökkenne, viszont a többi munka késése növekedne és legalább akkora súllyal kerülnének számításba, mint amennyivel a k csökkentette az összeget, hiszen mindegyik munka határideje maximum egy lehet ($d_1 \leq d_2 \leq d_3 \leq 1$) és a $\omega_k = \min\{\omega_1, \omega_2, \omega_3\}$. □

Nézzük meg egy példa nem megszakítható esetét véve először: $d_1 = 1,\ d_2 = 1,\ d_3 = 1$. Ekkor, ha optimálisan ütemezünk, két munkát tudunk időre elvégezni, az utolsó késése 1 lesz, ezért célszerű annak a legkisebb súllyal rendelkező munkának lennie. Megszakítható esetben ugyanez lesz az optimum, mert ha megszakítanánk az egyik időre elvégzett munkát, akkor látjuk, hogy annak a késése nő (amennyivel a másiké csökken) és nagyobb súllyal szorzódik, mint a k tette azelőtt.

Mivel a határidők egészek (ráadásul maximum 1 lehet mindegyik) és a munkák hossza egy időegységnyi, célszerű lesz a késéseket is egész értékekre állítanunk azért, hogy mindig a következő legkisebb súlyú munkához vegyük a lehető legtöbbet a késésekből, megszakítás miatt ne kerüljön át egy nagyobb súlyú munkához az adott késésmennyiség egy töredéke.

A továbbiakban ezt a lemmát fogjuk használni és a segítségével megmutatjuk, hogy az első időegység alatt a $P2 \mid p_j = 1,\ r_j,\ pmtn \mid \sum w_j T_j$ feladatnak az optimális megoldásában nincsen megszakítás.

Lemma: Vegyük a $P2 \mid p_j = 1,\ r_j,\ pmtn \mid \sum w_j T_j$ feladat egy lehetséges példáját és legyen $r := \min_{j=1}^{n} r_j$. Tegyük fel, hogy létezik legalább három darab i munka, melyre $r_i = r$ teljesül, illetve legalább négy munka, melyre $r_i \leq r + 1$ igaz. Ekkor létezik egy optimális megoldás, melyben kettő munka megszakítás nélkül elvégezhető az $[r, r + 1[$ időintervallumban.

Bizonyítás:

Legyen S egy optimális megoldása a megszakítható változatú feladatunknak és legyen I az alábbi munkák halmaza $I = \{i \mid r_i \leq r + 1\}$. Ezen halmaz elemeit $\{1,2,3,4\}$ rendezzük a munkák befejezési idejei szerint, azaz $C_1 \leq C_2 \leq C_3 \leq C_4$. Ha a négy munkát folyamatosan tudjuk ütemezni, akkor C_4 minimum értéke $r + 2$ lesz, de lehet hogy csak később tudjuk befejezni, mert

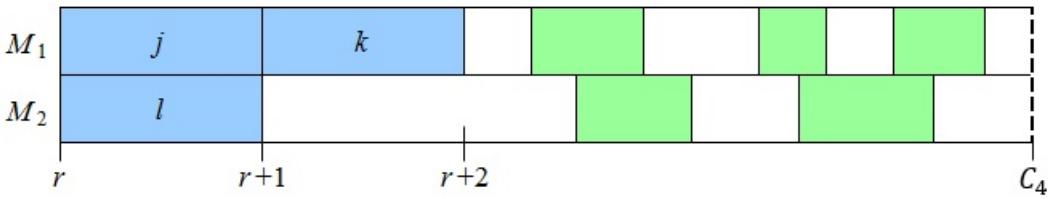
egy később végződő munkába is belekezdtünk már. Továbbá az összes többi $i \in I$ munkára, ami nem ezen négy munkák közé tartozik $C_4 \leq C_i$ teljesül.

Legyen az I' halmaz azon munkák összessége, melyek egy része $[r, r+2]$-on van ütemezve. Mivel minden elérhetési idő egész, ezért $r_i \leq r+1$ minden $i \in I'$ munkára. Azaz I' részhalmaza I-nek.

Hozzunk létre egy új ütemezést az alábbi lépések szerint:

1. Szedjük ki az összes $i \in I' \cup \{1,2,3\}$ részmunkát, amelyek $[r, C_4[$ intervallumon találhatóak. Az egyazon munkához tartozó részeket csatlakoztassuk egymáshoz és készítsünk egy-egy részmunkát belőlük.

2. Ütemezzük optimálisan az $\{1,2,3\}$ munkákat $[r, r+2[$ intervallumon. Ezek közül kettő (jelöljük j-vel és l-lel) az $[r, r+1]$ időintervallumra kerül és a harmadik (nevezzük k-nak) kerüljön az első gépre (M_1) az $[r+1, r+2]$ intervallumon.



   (Itt zölddel jelöljük a többi munka által megszakított részeket. Fehérrel hagyjuk az üres idősávokat.)

3. Harmadik lépésként rendezzük az I'-ben lévő részmunkákat a megmunkálási idejük szerint nem növekvő sorrendben balról jobbra haladva feltöltve az üres részeket $[r+1, C_4[$ intervallumon először az M_1, majd az M_2 gépen (McNaughton algoritmus szerint).

Az így kapott ütemezés megvalósítható, mert ha az M_1 gép végén túlfolyik az utolsó munka, tehát nem fejeződik be az utolsó időintervallumban, akkor az M_2 gépen fog folytatódni $r+1$-től, azaz az adott munkának nem lesz időbeli átfedése a két gépen, így helyes ütemezést kapunk. Az új ütemezés is optimális, egyrészt azért, mert az utolsó előtti lemmában beláttuk, hogy az $\{1,2,3\}$ munkák ütemezése optimális akkor is, ha a megszakítás megengedett. Másrész pedig a befejezési ideje az összes többi munkának nem csökken, azaz $C_4 \leq C_i$ minden $i \in I' \cup I$. □

Tétel: A $P2 \mid p_j = 1,\ r_j \mid \sum w_j T_j$ feladatra a megszakíthatóság nem vezet jobb megoldásra.

Bizonyítás:

Elegendő megmutatnunk, hogy létezik úgy optimális S megoldása a $P2 \mid p_j = 1,\ r_j,\ pmtn \mid \sum w_j T_j$ feladatnak, hogy az $[r, r+1[$ intervallumon a munkák nincsenek megszakítva. Aztán indukcióval következtetünk, hogy arra a problémára, amelyet S ütemezésben az $[r, r+1[$ időintervallumon lévő munkák eltávolításával kapunk, létezik optimális megoldás megszakítás nélkül. Ez az állítás igaz olyan problémákra, ahol $r_i = r$ elérhetőséggel kevesebb mint 3 darab i munkát találunk vagy kevesebb mint 4 olyan i munkánk van, ahol $r_i \leq r + 1$. Különben az állításunk az utolsó lemmából következik. □

A fejezetet a *How Useful are Preemptive Schedules* című cikk alapján dolgoztam ki, amelyet P. Brucker, S, Heitmann és J. Hurink publikált [2].

Irodalomjegyzék

[1] Scheduling seminar - Leah Epstein. *[Scheduling seminar] Leah Epstein (University of Haifa) — The Benefit of Preemption*. URL: `https://www.youtube.com/watch?v=Ik7106Ul9Uw&t=2161s`. (accessed: 2022.04.29.)

[2] P. Brucker, S. Heitmann, and J. Hurink. "How Useful are Preemptive Schedules?" In: 1605 (Nov. 2001), pp. 407–412.

[3] P. Brucker, J. Hurink, and S. Knust. "A polynomial algorithm for $P \mid p_j = 1,\ r_j,\ outtree \mid \sum C_j$". In: *Mathematical Methods of Operations Research* 56 (Apr. 2002), pp. 407–412.

[4] V. Chvatal. *Linear Programming*. Theorem 21. New York: Freeman, 1981.

[5] E.G. Coffman, Jr., and S. Even. "A note on limited preemption". In: *Parallel Processing Letters* 08.01 (1997), pp. 3–6.

[6] E.G. Coffman, Jr., and M.R. Garey. "Proof of the 4/3 Conjecture for Preemptive vs. Nonpreemtive Two-Processor Scheduling". In: *Journal of the ACM* 40.5 (Nov. 1993), pp. 991–1018.

[7] R.L. Graham et al. *Annals of Discrete Mathematics 5*. Elsevier, 1979. Chap. Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey, pp. 287–326.

[8] R. McNaughton. "Scheduling with deadlines and loss functions". In: *Management science* 6 (1959), pp. 1–12.

Melléklet

A $J = \{j_1, j_2, \ldots, j_n\}$ munkákat szeretnénk $M = \{M_1, M_2, \ldots, M_m\}$ gépeken ütemezni. Adottak a megmunkálási idők (p_j), a súlyok (ω_i) és a határidők (d_j).

$\alpha \mid \beta \mid \gamma$

1. α, azaz a gépek
   (a) 1 – egy gép
   (b) P – párhuzamos gépek
   (c) P2 – két párhuzamos gép
   (d) Q – uniform gépek
   (e) R – független gépek
2. β, azaz a feltételek
   (a) r_j – a j munka rendelkezésre állási ideje
   (b) *pmtn* – megszakíthatóság
   (c) *prec* – megelőzési feltételek
   (d) *outtree* – ki-fenyő
   (e) *intree* – be-fenyő
   (f) $p_j = 1$ – minden munka egységnyi hosszúságú
3. γ, azaz a minimalizálandó célfüggvény
   (a) $C_{\max}$ – teljes átfutási idő
   (b) $\sum C_j$ – átlagos átfutási idő
   (c) $\sum \omega_j C_j$ – súlyozott átlagos átfutási idő
   (d) $L_{\max}$ – maximum késés
   (e) $\sum U_j$ – késő munkák száma
   (f) $\sum T_j$ – a késő munkák összkésése
   (g) $\sum \omega_j U_j$ – késéssel elvégzett munkák súlyozott összege
   (h) $\sum \omega_j T_j$ – a késő munkák késéseinek súlyozott összege