"Az emberek okos szóra várnak."

Lázár Ervin

Optimális és közel-optimális online és félig online algoritmusok ütemezési feladatokra

PhD értekezés

készítette: Dósa György

témavezető: Vízvári Béla
egyetemi docens, kandidátus, ELTE Operációkutatási Tanszék

Szegedi Tudományegyetem, Természettudományi Kar
Matematika és Számítástudomány Doktori Iskola,
Vezetője: Dr. Hatvani László, tanszékvezető egyetemi tanár, akadémikus

Informatika Doktori Program,
Operációkutatás és kombinatorikus optimalizálás alprogram

Veszprém, 2007 február

Köszönetnyilvánítás

Nagy hálával tartozom közvetlen főnökömnek, Győri István egyetemi tanárnak, a Pannon Egyetem Matematikai és Számítástechnikai Tanszéke vezetőjének, hogy támogatott, buzdított e dolgozat megírására, és segített ennek a Szegedi Tudományegyetem Matematika és Számítástudományok Doktori Iskolájába történő beadásában.

Sokat segített az értekezés szerkesztésében témavezetőm, Vízvári Béla, akinek már egyetemi tanulmányaim során, később az ELTE Operációkutatási Tanszékén töltött néhány év alatt, majd aztán is jelentős szerepe volt abban, hogy az Operációkutatás tudományág művelését megszerettem, és akitől sokat tanultam.

Nem maradhat ki a sorból kínai Zhejiang Egyetem professzora, Yong He, akit az Interneten keresztül ismertem meg, és akivel három év alatt hét közös dolgozatot jelentettünk meg. Mikor ezek után (az ő meghívására) találkozhattam volna vele, korán, 36 évesen meghalt. Sokat köszönhetek e közös munkának, e dolgozat jelentős része a közösen publikált cikkek eredményeit tartalmazza.

Köszönöm kollégáim segítségét a dolgozat végső formába öntését illetően.

Családom tagjainak köszönöm türelmüket.

Végezetül: Hála Istennek.

Veszprém, 2007 február

TARTALOM

Optimális és közel-optimális online és félig online algoritmusok ütemezési feladatokra

1. Bevezetés

Ezen értekezés néhány ütemezéselméleti feladattal foglalkozik. A terület rendkívül szerteágazó, emiatt az első fejezetben egy általános áttekintést adunk azokról a fogalmakról, feladatokról, módszerekről, kutatási irányokról, amelyek az értekezés további részében előfordulnak majd. A következő négy fejezet lényegében új eredményeket tartalmaz. Az értekezésben leírtak gerincét a [12, 13, 14, 15, 38] dolgozatokban leírtak adják, amelyeket az értekezés szerzője, és az azóta (2005-ben) elhunyt Yong He publikálta. Az értekezésben szereplő új eredmények elsősorban (körülbelül 80 %-ban) az értekezés szerzőjének alkotásai. Az eredmények 70 %-ában az értekezés szerzőjének meghatározó a hozzájárulása, a maradék 30 % esetén a szerzők hozzájárulása oszthatatlan. Yong He Internetes kapcsolaton keresztül, folyamatos együttműködésben jelentős segítséget adott a fent nevezett publikációk elkészítésében, egzakt megfogalmazásában, a nemzetközi kutatások eredményeivel való összevetésében.

A feladatok mindegyike az NP-teljes feladatosztályba tartozik, és egyben olyan feladatok, amelyeket az utóbbi időkben sokan, intenzíven kutatnak, és jelentős nemzetközi folyóiratokban publikálnak.

1.1. Az ütemezéselmélet fogalmai, feladatai, módszerei

Bevezetjük az értekezésben található legfontosabb fogalmakat, áttekintünk néhány feladattípust, és az azokra kifejlesztett megoldási módszereket. Egy áttekintés magyar nyelven a [51, 66] munkákból nyerhető, angol nyelven pedig ajánljuk a a [9, 58] műveket. Az ütemezéselmélet tárgykörében általában feltesszük hogy adott valahány, valamilyen tulajdonságokkal rendelkező gép, amelyekkel valahány munkát kell elvégezni úgy, hogy valamilyen célfüggvény minimális vagy maximális értékét keressük. A feladattípust az

$$\alpha \,|\beta|\, \gamma$$

konvenció szerint szokás megadni, itt az α mezőben a gépekről, a β mezőben a feladatokról, a γ mezőben pedig az ütemezési feladat célfüggvényéről közlünk információkat. (Részletesen lásd: [66]). Tekintsük például a $P_m \,||\, C_{\max}$ feladatot. A jelölések jelentése itt a következő: A gépek száma m, ami pozitív egész szám, és a gépek egyforma párhuzamos (parallel) gépek. Ezen azt értjük, hogy a munkák végrehajtása minden gépen egységesen ugyanannyi időbe kerül. A feladatok vagy munkák számát általában n-nel jelöljük. A munkák *végrehajtási ideje*, vagy hossza legyen p_i, ahol tehát $1 \leq i \leq n$. Ha egy gép elkezd egy munkát végrehajtani, akkor a munka végrehajtását be is kell fejeznie, ezt úgy mondjuk, hogy a munkák megszakítása nem megengedett. A munkák *ütemezés*én azt értjük, hogy a munkákat szétosztjuk a gépek között, minden gép a hozzá rendelt munkákat kell hogy elvégezze. A munkák ütemezése itt tehát egyszerűen a munkák $\mathcal{T}$ halmaza egy partíciójának felel meg.

Nincs várakozási idő két munka elvégzése között, vagyis amint egy gép elvégez egy munkát, rögtön elkezdheti a következőt. Jelen probléma esetén az egy géphez rendelt munkák végrehajtási sorrendje közömbös. Valamely géphez hozzárendelt munkák végrehajtási idejeinek összegét a gép *terhelés*ének nevezzük. Minden munkának van egy *kezdési időpont*ja, és egy *befejezési időpont*ja. A gép terhelése tehát a gép által végzett utolsó munka befejezésének időpontja, (feltételezve, hogy várakozási idők nincsenek). Ekkor az utolsó munka befejezésének időpontját a gép *átfutási idejének* is mondjuk. A gépek átfutási idejeinek maximumát az ütemezés *teljes átfutási idejének* nevezzük. A $C_{\max}$ célfüggvény esetén a teljes átfutási időt minimalizáljuk.

Az előbbiekben meghatároztuk tehát a $P_m \,||\, C_{\max}$ feladatot. Ez a feladat az egyik legegyszerűbben leírható ütemezéselméleti feladat. A feladat az NP-teljes feladatosztályba tartozik, vagyis nem ismert a feladatra polinomiális idejű, optimális megoldást adó algoritmus.

Egy másik gyakori eset a *hasonló gépek* (uniform machines) esete, ezeket Q-val jelöljük. A gépek ebben az esetben is párhuzamosan működnek, de minden gépnek van egy sebessége: legyen az i-edik gép sebessége s_i. Ekkor a j-edik munka elvégzéséhez szükséges idő az i-edik gépen $\frac{p_j}{s_i}$. Ezen értekezés keretei között csak az előbb említett két esettel foglalkozunk.

A munkákra vonatkozó egyéb lehetséges feltételek: Azt mondjuk, hogy a munkák *megszakítás*a (preemption) megengedett, ha a munkák végrehajtását meg szabad szakítani, és a hátralévő munka akár ugyanazon a gépen, akár más gépen folytatható. Egy-egy munka tetszőlegesen sokszor, de véges sokszor megszakítható. Ha valamely munka végrehajtási ideje nemcsak hogy megszakítható, hanem a különböző részek végrehajtása akár ugyannabban a pillanatban egyszerre több gépen is lehetséges, akkor *átlapolás*ról (overlap) beszélünk, ez az eset leginkább sorozatok gyártásakor fordul elő.

A munkák között lehetnek még megelőzési feltételek, megengedett kezdési, és

előírt befejezési idők, és más feltételek is, ezekkel az értekezésben nem foglalkozunk.

A célfüggvény is lehet még például a munkák befejezési idejeinek (súlyozott) összege, amit minimalizálunk, a minimális gépterhelés, amit maximalizálunk, a befejezési idők (súlyozott) összege, a késve befejezett munkák száma, a késések összege, és sokfajta egyéb célfüggvény. Az értekezésben általában csak a $C_{\max}$ célfüggvény vizsgálatával foglalkozunk majd.

Offline és online feladatok, egzakt és heurisztikus algoritmusok

Az ütemezéselméleti feladatok osztályozásánál a feladatok két nagy csoportját szokták megkülönböztetni: ezek az úgynevezett offline, illetve online feladatok [9]. Amikor az ütemezendő munkákról minden információ előre, az ütemezés megkezdése előtt rendelkezésünkre áll, az ütemezési feladatot *offline*-nak nevezzük. A másik esetben a munkák egymás után, egyenként érkeznek, és amikor egy munka megérkezik, akkor még semmit nem tudunk arról, hogy jönnek-e még a későbbiekben munkák, és ha igen, akkor milyenek, és minden egyes munkát közvetlenül akkor kell (később meg nem változtatható módon) valamely gépre ütemezni, amikor megérkezik. Ez esetben az ütemezést *online*-nak nevezzük. Online esetben mondják azt is hogy a munkák valamely L lista szerinti sorrendben érkeznek, és az adott sorrendben érkező munkákat szokás *sorozat*nak (sequence) is nevezni. Hangsúlyozzuk viszont, hogy sorozaton itt a matematikai analízistől eltérően a munkáknak mindig véges sorozatát értjük. Az offline illetve online feladatokat megoldó algoritmusokat természetszerűleg offline-, illetve online algoritmusoknak nevezzük. Egy algoritmust egzakt módszernek nevezünk, ha optimális megoldást határoz meg, ha megelégszik valamilyen közel-optimális megoldás keresésével, akkor pedig heurisztikus módszernek.

Az algoritmusok hatékonyságvizsgálata

Az offline ütemezési algoritmusok hatékonyságát kétféleképpen szokták mérni. Legyen a gépek száma m, legyen $C_{OPT}(\mathcal{T})$ az optimális megoldás értéke, és jelölje $C_A(\mathcal{T})$ az A algoritmus által kapott ütemezés célfüggvényértékét valamely $\mathcal{T}$ feladathalmaz esetén. Az egyik lehetőség annak a vizsgálata, hogy az úgynevezett legrosszabb esetben mennyiszer rosszabb megoldást ad az algoritmus az optimális megoldásnál (angolul: worst case analysis). Ez esetben tehát a $\frac{C_A}{C_{OPT}}$ hányados szuprémumát keressük, vagyis legyen

$$R_m(A) = \sup_{\mathcal{T}} \left\{ \frac{C_A(\mathcal{T})}{C_{OPT}(\mathcal{T})} \right\},$$

az előbbi (pozitív) értéket az A algoritmus *legrosszabb eset arány*ának nevezzük. Egy másik lehetőség az, hogy azt vizsgáljuk, hogy az algoritmus átlagosan

mennyivel ad rosszabb eredményt a lehető legjobb megoldásnál, (angolul: probability analysis). Mi ez utóbbi vizsgálattal jelen értekezés keretei között nem foglalkozunk.

Az online problémákra alkalmazott algoritmusokat természetes módon *online algoritmusoknak* nevezzük. Ezek hatékonyságát az úgynevezett *versenyképességi analízis* segítségével vizsgáljuk, ez egyfajta legrosszabb eset vizsgálatnak felel meg, ahol egy online algoritmus által kapott ütemezés eredményét az offline eset optimális megoldásával hasonlítjuk össze, a következőképpen: Legyen A egy online algoritmus, jelölje C_A az A algoritmus által kapott ütemezés teljes átfutási idejét, illetőleg legyen C_{OPT} az optimum értéke (az offline ütemezés esetén). Ekkor A versenyképességi aránya a legkisebb olyan ρ valós szám, amelyre $C_A \leq \rho C_{OPT}$ teljesül a munkák tetszőleges sorozata esetén. Egy online feladatnak a ρ konstans *alsó korlátja*, ha nem létezik olyan online algoritmus, amelynek a versenyképességi aránya ennél kisebb lenne. Ezek után egy online algoritmust *optimálisnak* nevezünk, ha a versenyképességi aránya megegyezik a feladat alsó korlátjával.

Az LPT, és LS algoritmusok

R.L. Graham [35, 36] cikkeiben közölte az LPT (=Longest Processing Time) algoritmust: Először a munkákat a műveleti idő szerint csökkenő sorrendbe rendezzük (ez az úgynevezett LPT sorrend), majd ebben a sorrendben ütemezzük őket. A soron következő munka végrehajtását mindig a lehetséges legkorábbi időpontban kezdjük, vagyis a soron következő munka mindig valamelyik minimális terhelésű gépre kerül. Graham algoritmusára teljesül a következő tétel:

1.1 Tétel ([35]) $R_m\left(LPT\right) = \frac{4}{3} - \frac{1}{3m}$. $\square$

A legrosszabbhoz közeli esetek az összes esetnek csak igen kis százalékában fordulnak elő. Sőt, Dósa [16] bebizonyította, hogy a legrosszabb eset csak egyetlen feladathalmaz esetén fordulhat elő, amely (rögzített m gépszám esetén) a következő:

$$\mathcal{T} = \left\{2m-1, 2m-1, 2m-2, 2m-2, ..., m+1, m+1, m, m, m\right\}.$$

Ha az előbbi algoritmust úgy hajtjuk végre, hogy a munkák egy adott L lista szerinti sorrendben érkeznek, és ebben a sorrendben kell őket ütemezni, (vagyis a kezdeti LPT sorrendbe rendezésre nincs lehetőség), kapjuk az LS (List Scheduling, vagyis lista szerinti ütemezés) algoritmust. Az LS algoritmus felfogható tehát egy online algoritmusnak is. Könnyen látható [35], hogy az LS algoritmus versenyképességi aránya m gép esetén $2 - \frac{1}{m}$, (ami nyilván nagyobb mint az LPT algoritmus legrosszabb esetbeli aránya.) Faigle, Kern és Turán [31] cikke bebizonyította hogy LS optimális online algoritmus $m = 2, 3$ gépszámok esetén,

nagyobb gépszámok esetén léteznek némileg jobb versenyképességi aránnyal rendelkező algoritmusok ([1, 2, 6, 32, 57, 58]).

Gyakori az, hogy valamely algoritmusnak LPT vagy LS segédalgoritmusa, ilyenekkel az értekezés későbbi fejezeteiben is találkozhatunk majd.

1.2. Félig online ütemezési feladatok

Az elmúlt évek során megkülönböztetett figyelmet szenteltek az úgynevezett félig online (semi online) ütemezéseknek. Ezek a feladatok az offline és az online esetek "között" helyezkednek el: Előre tudunk a munkákról valamit, vagy valamilyen algoritmikus "könnyítés" lehetséges az ütemezés során. A gyakorlatban előforduló ütemezési feladatok jelentős részénél feltehetőek ilyen feltételek.

A félig online problémákra alkalmazott algoritmusokat természetes módon *félig online algoritmusoknak* nevezzük. Ezek hatékonyságát, az online esethez hasonlóan a *versenyképességi analízis* segítségével vizsgáljuk. Egy félig online algoritmust *optimálisnak* nevezünk, ha a versenyképességi aránya megegyezik a feladat alsó korlátjával.

Érdekes kérdés, hogy valamely félig online feltétel a feladatot "könnyebbé teszi-e" abban az értelemben, hogy feltétel teljesülése esetén van-e olyan algoritmus, amelynek versenyképességi aránya kisebb mint valamely optimális online algoritmusé. Tekintsük illusztráció céljából a $P_2||C_{\max}$ ütemezési feladat félig online változatait. Ismert [31, 35], hogy két egyforma párhuzamos gép esetén LS optimális online algoritmus, $3/2$ versenyképességi aránnyal. Ha előre ismerjük a munkák összhosszát [54], vagy előre ismert a maximális munkaméret [45], vagy pedig előre ismert a teljes átfutási idő optimális értéke [4], akkor egy optimális algoritmus versenyképességi aránya $4/3$, ha a munkák a méreteik nemnövekvő sorrendjében érkeznek [59], akkor pedig $7/6$.

Ha a munkák mindegyike a $[p, rp]$ intervallumban helyezkedik el valamilyen $p > 0$ és $1 \leq r \leq 2$ konstansokkal [45], akkor LS versenyképességi aránya $(1 + r)/2$ ha $1 \leq r \leq 2$, és $3/2$ tetszőleges $r \geq 2$ esetén, így $r \geq 2$ esetén LS optimális. A kétgépes esetben tehát $r < 2$ esetén van olyan félig online algoritmus, melynek versenyképességi aránya kisebb, mint az online eset optimális algoritmusáé. Továbbá két gép esetén LS optimális online, és egyben optimális félig-online algoritmus. A háromgépes esettel korábban még nem foglalkoztak. Az előbbi eredmények a következő kérdéseket vetik fel: Igaz-e, hogy három gép esetén is minden r méretarány esetén LS optimális marad, és milyen r esetén kapunk kisebb versenyképességi arányt valamely optimális algoritmus esetén.

Az értekezésben megmutatjuk a He - Dósa [38] dolgozatot követve, hogy három gép esetén az LS algoritmus versenyképességi aránya az r paraméter füg-

gvényében a következőképpen adható meg:

$$R_{LS} = \begin{cases} 1 + \frac{2(r-1)}{3}, & ha \quad 1 \leq r \leq \frac{3}{2}, \\ 2 - \frac{3}{r+3}, & ha \quad \frac{3}{2} < r \leq \sqrt{3}, \\ \frac{r+1}{2}, & ha \quad \sqrt{3} < r \leq 2, \\ 2 - \frac{1}{r}, & ha \quad 2 < r \leq 3, \\ \frac{5}{3}, & ha \quad 3 < r. \end{cases}$$

Észrevehetjük, hogy a versenyképességi arány értéke az r paraméternek folytonos, szakaszos, és nem minden szakaszon lineáris függvénye. Az LS algoritmusról belátjuk, hogy $r \in [1, 3/2] \cup [\sqrt{3}, 2] \cup [6, +\infty)$ esetén optimális. $2 < r < 6$ esetén pedig megadunk javított algoritmusokat, a következők szerint:

a, Az $r \in (2, 5/2]$ esetben megadunk egy optimális algoritmust, melynek versenyképességi aránya $3/2$;

b, $r \in (5/2, 3]$ esetén egy közel optimális algoritmust $\frac{4r+2}{2r+3}$ versenyképességi aránnyal, amikor az alsó korlát legalább $\frac{7r+4+\sqrt{r^2+8r+4}}{2r+2+2\sqrt{r^2+8r+4}}$. A két érték közötti eltérés legfeljebb 0.01417.

c, Az $r \in (3, 6)$ esetben megelégszünk azzal, hogy bemutatunk egy olyan algoritmust, amelynek a versenyképességi aránya minden rögzített r paraméterérték mellett jobb mint az LS algoritmusé. A versenyképességi arány értéke $\frac{5}{3} - \frac{\delta}{18}$, ahol $\delta = \min\{\frac{6-r}{18}, \frac{1}{37}\}$, (az LS algoritmusé $\frac{5}{3}$).

Az előbbiek konklúziója, hogy három gép, és közel egyforma végrehajtási idők esetén lehetséges az LS-nél hatékonyabb algoritmust megadni, ha a méretarány kettő és hat közötti szám, vagyis LS nem minden esetben optimális. Továbbá az optimális félig-online algoritmus versenyképességi aránya erősen függ az r paraméter értékétől.

Kellerer, Kotov, Speranza és Tuza már említett [54] cikke három félig online változatot vizsgált: Ismerjük a munkák méreteinek összhosszát; felhasználható egy puffer valahány munka tárolására, (vagyis néhány munkát félretehetünk, később dönthetünk az ütemezésükről); illetve két ütemezést is készíthetünk, és végül a jobbikat választhatjuk. Mindhárom esetben egy optimális algoritmus versenyképességi aránya $4/3$. Arra irányuló vizsgálatok is történtek, hogy két különböző félig online feltétel együttes megléte tovább csökkenti-e a feladatra adható optimális algoritmus versenyképességi arányát. Tan és He [62] közöl néhány olyan feltétel-kombinációt, amelyek "haszontalanok" abban az értelemben, hogy ezek együtt se biztosítanak jobb versenyképességi arányt, mint az egyszerű online feltétel. Ugyanebben a cikkben megmutatták, hogy ha előre ismerjük a munkák összhosszát, valamint a legnagyobb munkaméretet is, akkor egy optimális algoritmus versenyképességi aránya $6/5$. Ha pedig előre tudjuk a munkák összhosszát, és a munkák a méreteik csökkenő sorrendjében jönnek, akkor egy optimális algoritmus versenyképességi aránya $10/9$. Ez utóbbi eredményt közli Epstein [26] cikke is. Zhang és Ye [68] a "*suggestive*" és "*Largest Last*" informá-

ciók kombinációját vizsgálta. Ezeken azt értjük, hogy amikor az utolsó munka megérkezik, ennek érkezésekor megtudjuk, hogy ez az utolsó munka, illetve előre tudjuk, hogy az utolsó munka hossza a legnagyobb. Megadtak egy optimális $\sqrt{2}$-versenyképességű algoritmust, másrészt pedig megmutatták, hogy külön-külön mindkét feltétel "haszontalan", az előbb magadott értelemben [62].

Mi, az értekezés második fejezetében az előbbi [54] cikk három változatából kettőt-kettőt kombinálunk, a Dósa - He [13] dolgozatban megjelentek szerint, mindkét esetben igaz, hogy a második félig online feltétel miatt a versenyképességi arány tovább csökken. Először feltesszük hogy előre ismert az ütemezendő feladatok méretének összege, és rendelkezésünkre áll egy 1 pozícióval rendelkező puffer az ütemezés során, vagyis egy munka ütemezését mindig tartalékolhatjuk. Erre a félig online változatra megadunk egy optimális 5/4-versenyképességű algoritmust. Megmutatjuk, hogy a puffer méretének növelésével a versenyképességi arány nem javítható. A másik esetben is előre ismert az ütemezendő feladatok méretének összege, valamint egyszerre két ütemezést készíthetünk, és az ütemezés végén a jobbikat választhatjuk. Ebben az esetben megadunk egy optimális, 6/5-versenyképességű algoritmust. Mindkét esetben igaz tehát az, hogy a további félig online feltétel által csökkent az elérhető versenyképességi arány.

Sok egyéb félig online változatot is vizsgáltak, a [3, 44] művek esetén a célfüggvény a minimális gépterhelés maximalizálása, a [3, 24, 27, 28, 30, 43, 63, 64] cikkek a $Q||C_{\max}$ hasonló gépes feladat valamilyen félig online változatát tárgyalják. Egy különleges feladat szerepel Imreh Csanád [49] cikkében, itt adott gépek két egymástól független csoportja, és minden munkának két végrehajtási ideje van aszerint, hogy az egyikfajta, vagy a másikfajta gépre kerül. A cikk a feladat többféle (online vagy félig online) változatát vizsgálja. Mint látható, a félig online változatok száma elég nagy, népszerű kutatási terület az utóbbi években. Az értekezés második fejezetében tehát félig online ütemezési feladatokkal foglalkozunk.

1.3. Elutasításos modellek

Az ütemezéselmélet elutasításos modelljei esetén minden munkához adott egy új paraméter is, ami a munka ütemezésének elutasítása esetén fizetendő büntetés összege. Ha egy munkát elutasítunk, akkor meg kell fizetnünk a munkához hozzárendelt büntetést, vagy pedig a munkát ütemezzük valamelyik gépre. A célfüggvény lehet bármely korábbi célfüggvénynek, és az elutasított munkákért fizetett büntetéseknek valamilyen kombinációja.

Az ütemezési feladat elutasításos modelljével először Bartal és társai [7] foglalkoztak. Cikkükben párhuzamos, egyforma gépek ütemezésének elutasításos modelljét (MSR) vizsgálták. Az offline esetben minden rögzített m gépszám esetén teljesen polinomiális approximációs sémát adnak meg. Bemutatnak továbbá egy $O(n \log n)$ lépésszámú approximációs algoritmust, amelynek approximációs

aránya minden m esetén $2 - 1/m$. A cikk fő eredménye, hogy az MSR feladat online változatára megadtak egy algoritmust $RTP(\alpha)$ néven, amelynek versenyképességi aránya tetszőleges gépszám esetén legfeljebb $1+\phi \approx 2.618$, ahol $\phi = (1+\sqrt{5})/2$ az aranymetszés aránya. Egy másik algoritmus ($RP(\alpha)$) is szerepel a cikkben ϕ versenyképességi aránnyal, ami két gép esetén optimális, valamint még egy on-line algoritmus 2 versenyképességi aránnyal $m = 3$ gép esetére, ahol az alsó becslés 1.839. Amikor a munkák megszakítása megengedett, található az előzőeknél hatékonyabb algoritmus $(4+\sqrt{10})/3 < 2.38743$ versenyképességi aránnyal, míg az alsó korlát 2.12457 (Seiden [60]).

Epstein és társai [29] cikke egységnyi idejű munkáknak egy gépre történő online ütemezésével foglalkozik. A célfüggvény az ütemezett munkák befejezési idejeinek összege plusz az összes büntetés, ezt kell minimalizálni. A cikk közöl egy $\frac{1}{2}\left(2+\sqrt{3}\right) \approx 1.866$ versenyképességű online algoritmust, az alsó korlát 1.6378.

A hasonló gépek ütemezésének elutasításos modelljére (lásd például [25, 47, 61]) röviden USR feladatként hivatkozunk. He és Min [41] az online USR feladattal foglalkozik. A cikk közli az $LSR(\alpha)$ on-line algoritmust. Két gép esetén, ahol $s_1 = 1, s_2 = s \geq 1$ teljesül, az algoritmus versenyképességi aránya

$$\begin{cases} s + \alpha(1 + s - s^2), & ha\ 1 \leq s < \phi, \\ \frac{s+1}{s}, & ha\ s \geq \phi, \end{cases}$$

ahol $s \geq \phi$ esetén $\alpha = 1/s$, $1 \leq s < \phi$ esetén pedig α egyenlő az

$$\frac{s+1}{s + x(s+1) - 1} = s + x(1 + s - s^2)$$

egyenlet pozitív gyökével. Az algoritmus minden $s \geq \phi$ sebesség esetén optimális.

Mi, az értekezés harmadik fejezetében a Dósa - He [14] dolgozatot követve megmutatjuk, hogy az előző algoritmus az α paraméter megfelelőbb választásával javítható, sőt, a javított algoritmus bizonyos esetekben optimális is, (amikor a korábbi nem az). Pontosabban, bevezetjük az $LSRM(\alpha)$ algoritmust, amelynek versenyképességi aránya $\frac{1}{2s}\left(s^2 + \sqrt{s^4 - 4s^3 + 4s^2 + 4s}\right)$, ami minden $1 \leq s < \phi$ sebesség esetén jobb, mint az $LSR(\alpha)$ algoritmusé. Jegyezzük meg, hogy a [41] cikk alsó becslései triviálisak, hiszen azok az elutasítás nélküli esetben is alsó korlátok. Emiatt új, nemtriviális alsó korlátokat is közlünk, ezek által következik, hogy $LSRM(\alpha)$ algoritmusunk optimális minden $1.3852 \leq s < \phi$ esetén.

Az USR feladat megszakításos esetével tudomásunk szerint korábban még nem foglalkoztak. Mi itt közlünk egy olyan algoritmust az online megszakításos esetre, amely minden $s \geq 1$ sebesség esetén optimális, az algoritmus versenyképességi aránya $\frac{s+\sqrt{s^2+4s}}{2s} = \frac{1}{2} + \sqrt{\frac{1}{4} + \frac{1}{s}}$.

1.4. A gépköltséges ütemezési feladat

A [48] cikkben Imreh Csanád és John Noga felvetette a klasszikus (párhuzamos gépes) ütemezési feladat egy újfajta megközelítését. Az eredeti változattól való különbségek a következők: 1, Nincs rögzítve kezdetben a gépek száma, 2, bármikor lehetőségünk van egy újabb gép vásárlására, amikor egy újabb munka megérkezik, 3, a célfüggvény, amit minimalizálunk, a gépek vásárlására fordított összeg és a teljes átfutási idő összege. A feladatra *Lista Modell*-ként fogunk hivatkozni. (A cikk egy másik modellt is közöl, azzal itt nem foglalkozunk.)

Ez a feladat eléggé különböző a klasszikus ütemezési feladatoktól, amikor is a gépek m száma rögzített, és az algoritmus nem változtathatja meg a gépek számát, másrészt a gépek (illetve azok beszerzése) nem kerülnek külön pénzbe, tehát ez nem jelenik meg a feladat célfüggvényében. A gépköltséges eset felvetését nyilván az motiválja, hogy a valós életben a gépek beszerzése pénzbe kerül, másrészt pedig ha lehetőségünk ven a gépek számát befolyásolni (növelni), az lényegesen befolyásolja az ütemezés eredményét, illetőleg hatékonyságát. Jegyezzük meg, hogy Graham pontosan ezzel a kérdéssel is foglalkozott korai [35] cikkében, ("Bizonyos multiprocesszor anomáliákra vonatkozó korlátok"), tehát azt vette észre, hogy az online esetben nem feltétlenül lesz kisebb a teljes átfutási idő, ha valamely algoritmust több gépre alkalmazunk.

A Lista Modell feladat esetén, Imreh és Noga [48] közölte az A_ρ on-line algoritmust, amely $(1+\sqrt{5})/2 \approx 1.618$-versenyképes, míg az alsó korlát $4/3$. Amikor ismerjük a legnagyobb munka hosszúságot, He és Cai [42] közöl egy legfeljebb 1.5309 versenyképességi aránnyal rendelkező algoritmust. Amikor a munkák összhossza ismert, ugyanezen cikk közöl olyan algoritmust, amelynek versenyképességi aránya legfeljebb 1.414, míg a feladatnak 1.161 alsó korlátja. Csökkenő végrehajtási idők esetére Cai and He [10] közöl egy legfeljebb 3/2-versenyképes algoritmust, az alsó korlát $4/3$. Mindezen algoritmusok lényegében az A_ρ algoritmus valamilyen változatai. Egy rokon feladat a "paging". Ezen a területen is van erőforrás vásárlásos cikk: [11], ahol extra memóriát lehet venni.

Az értekezés negyedik fejezetében a Dósa - He [12] dolgozatban megjelentek alapján először megadunk egy javított, $(2\sqrt{6}+3)/5 \approx 1.5798$-versenyképes algoritmust a tiszta online esetre. Tudomásunk szerint ennél hatékonyabbat azóta nem közöltek. Ezután azzal a speciális esettel foglalkozunk, amikor a munkák mérete nem hosszabb mint a gépek vásárlásának költsége, vagyis minden munka hossza legfeljebb 1. E félig online feltétel először a [12] dolgozatban szerepelt. Erre az esetre közlünk egy $4/3$-versenyképes optimális félig online algoritmust.

Harmadszor, közlünk egy legfeljebb 3/2-versenyképes kétfázisos algoritmust arra az esetre, amikor előre ismert a legnagyobb munka hosszúság. Ez az eredmény szintén megjavítja a korábbi hasonló esetre vonatkozó ismert eredményt.

1.5. A gépköltséges feladat elutasításos változata

Az ötödik fejezetben definiáljuk a gépköltséges feladat elutasításos változatát, a feladattal korábban még nem foglalkoztak (kivételt képez Imreh Csanád és Nagy-György János most megjelenendő [50] cikke). Ez esetben tehát két különleges feltétel is van, az egyik, hogy lehetőségünk van új gépek vásárlására, másrészt pedig a munkák ütemezését elutasíthatjuk, ez esetben bizonyos nagyságú büntetést kell fizetnünk. Tehát ebben a fejezetben az Imreh Csanád és John Noga által bevezetett gépköltséges feladatot [48, 12], és a munkák elutasításának lehetőségét [7] ötvözzük.

A tiszta online esetre Imreh Csanád és Nagy-György János [50] cikke megad egy 2.618-versenyképes algoritmust, az általános esetben optimális algoritmus nem ismert. Mi csak azzal a félig online esettel foglalkozunk, amikor a munkák hossza legfeljebb 1. A feladat ezen megszorítás mellett is általánosítása az úgynevezett sí-kölcsönzési feladatnak (a felismerés Leah Epsteintől származik). A feladatra megadunk [15] egy egyszerű, optimális, 2-versenyképes, kétfázisú félig online algoritmust. Az algoritmus vizsgálatakor értekezésünkben egy újszerű módszerrel élünk: Ha az algoritmus nem lenne 2-versenyképes, akkor lennie kell (elemszám tekintetében) minimális ellenpéldának. Ez azonban rendkívül bonyolult szerkezetű feladathalmaz is lehet, (miként az algoritmus sem teljesen triviális). Emiatt az ellenpéldát fokozatosan kicseréljük újabb és újabb, de szintén minimális ellenpéldákkal, az utoljára kapott feladathatmazról pedig már viszonylag könnyen látjuk, hogy nem is ellenpélda, és így indirekt bizonyításunk véget ér.

Jegyezzük meg, hogy legjobb tudomásunk szerint a gépköltséges feladatra ezidáig csak három optimális algoritmus született, (a szerző három [12, 15, 19] dolgozatában szereplő), ez is jelzi a feladat bonyolultságát.

2. Félig online ütemezési feladatok

Ebben a fejezetben a $P_m|online|C_{\max}$ feladatnak bizonyos félig online változatával foglalkozunk, és megadunk néhány egyszerű optimális, vagy közel-optimális algoritmust. A fejezet tartalma a következő dolgozatokban jelent meg: [13, 38]. Az $ALG2.1$ és $ALG2.2$ algoritmusok, és azok vizsgálata lényegében az értekezés szerzőjének alkotásai. A fejezet második részében, az LS algoritmus vizsgálatát szintén az értekezés szerzője végezte. Az $ALG(\gamma)$, $ALG2.3$ és $ALG2.4$ algoritmusok, és azok vizsgálata esetén a [38] cikk szerzőinek hozzájárulása oszthatatlan.

Először legyen a gépek szám kettő, vagyis tekintsük a $P_2|online|C_{\max}$ ütemezési feladatot, ismert, hogy ez esetben LS optimális online algoritmus, és versenyképes-ségi aránya 3/2 [31, 35]. A félig online feladatok esetén mindig érdekes felmerülő kérdés az, hogy valamely félig online feltétel a feladatot "könnyebbé teszi-e" abban az értelemben, hogy a félig online feladatra van-e olyan algoritmus, amely-nek versenyképességi aránya kisebb mint valamely optimális online algoritmus versenyképességi aránya, illetve ha egy félig online feltételhez még egy továbbit csatolunk, tovább csökken-e ezáltal egy optimális algoritmus versenyképességi aránya.

Kellerer, Kotov, Speranza és Tuza [54] cikke a feladat három félig online változatával foglalkozik. Az első esetben előre ismerjük a munkák méreteinek összhosszát, a második esetben pedig felhasználható egy puffer valahány munka tárolására, vagyis néhány munkát félretehetünk, és később dönthetünk az ütemezésükről. A harmadik esetben egyszerre két, egymástól független ütemezést készíthetünk, és végül a jobbikat választhatjuk. A második változatot Zhang [67] is vizsgálta. Kiderült, hogy mindegyik előbb felsorolt esetben egy-egy optimális félig online algoritmus versenyképességi aránya 4/3.

Most az előbbi három feltételből párosítjuk valamelyik kettőt, kétféleképpen. Az első esetben feltesszük hogy előre ismert az ütemezendő feladatok méretének összege, és egy 1 pozícióval rendelkező puffer is rendelkezésünkre áll az ütemezés során, vagyis egy munka (amelynek mérete tetszőlegesen nagy lehet) ütemezését mindig tartalékolhatjuk. Erre a félig online változatra megadunk egy optimális 5/4-versenyképességű algoritmust. Azt is megmutatjuk, hogy a puffer méretének növelésével a versenyképességi arány nem javítható. A másik esetben is előre ismert az ütemezendő feladatok méretének összege, itt viszont egyszerre két ütemezést készíthetünk, és az ütemezés végén a jobbikat választhatjuk. Ebben az esetben megadunk egy optimális, 6/5-versenyképességű algoritmust. Mindkét esetben igaz tehát az, hogy a további félig online feltétel által csökkent az elérhető versenyképességi arány. Nyitott kérdés marad annak a feladatnak a vizsgálata, amikor a harmadikféle módon párosítunk kettőt az előbbi három közül. Szintén nyitott kérdés, hogy ha mindhárom feltétel teljesül, akkor ezáltal még tovább csökken-e a versenyképességi arány.

A fejezet második részében a $P_3|online|C_{\max}$ feladatnak azzal a félig online

változatával foglalkozunk, amikor előre tudjuk, hogy a munkák végrehajtási ideje közel egyforma, ez a félig online feltétel a gyakorlatban sok esetben feltételezhető. Pontosabban fogalmazva, előre tudjuk, hogy a munkák hossza a p és rp konstansok között van, (ahol $p > 0$, $r \geq 1$). Ha az r konstans túl nagy, akkor az előbbi információ nem sokat ér a következő értelemben: megmutatjuk, hogy ha $r \geq 6$, akkor nincs olyan félig online algoritmus, amelynek versenyképességi aránya jobb lenne, mint valamely optimális online algoritmusé. Érdekes kérdés tehát az, hogy mekkora az a maximális r méretarány, amely biztosítja azt, hogy egy optimális félig-online algoritmus versenyképességi aránya jobb legyen, mint az egyszerű online esetben.

A $P_m|online|C_{\max}$ feladat esetén az LS algoritmusa versenyképességi aránya $R_{LS} = 2 - 1/m$. Faigle, Kern és Turán [31] cikke szerint LS optimális algoritmus 2 és 3 gép esetén. (Nagyobb gépszámra több algoritmus is létezik, amelyek valamivel jobbak versenyképességi arány szempontjából, lásd [1, 2, 32, 58]).

Az az egymáshoz közeli végrehajtási idők félig online feltétele először a [45] cikkben szerepelt. A cikk megmutatta, hogy két gép esetén LS optimális félig online algoritmus, melynek versenyképességi aránya $(1 + r)/2$, ha $1 \leq r \leq 2$, és $3/2$ tetszőleges $r \geq 2$ esetén. Emiatt a kétgépes esetben $r < 2$ esetén van olyan félig online algoritmus, amelynek versenyképességi aránya kisebb, mint az online eset optimális algoritmusáé. Továbbá két gép esetén az optimális félig online algoritmus ugyanaz, mint ami optimális az online esetben. (Pontosaban fogalmazva, LS *egy* optimális online, és egyben optimális félig-online algoritmus.) Ezek az eredmények a következő kérdéseket vetik fel: Igaz-e, hogy három gép esetén is minden r méretarány esetén LS optimális marad, és milyen r esetén kapunk kisebb versenyképességi arányt valamely optimális algoritmus esetén.

Megmutatjuk, hogy három gép esetén az LS algoritmus versenyképességi aránya az r paraméter függvényében a következőképpen adható meg:

$$R_{LS} = \begin{cases} 1 + \frac{2(r-1)}{3}, & ha \quad 1 \leq r \leq \frac{3}{2}, \\ 2 - \frac{3}{r+3}, & ha \quad \frac{3}{2} < r \leq \sqrt{3}, \\ \frac{r+1}{2}, & ha \quad \sqrt{3} < r \leq 2, \\ 2 - \frac{1}{r}, & ha \quad 2 < r \leq 3, \\ \frac{5}{3}, & ha \quad 3 < r. \end{cases}$$

Észrevehetjük, hogy a versenyképességi arány értéke az r paraméternek folytonos, szakaszos, és nem minden szakaszon lineáris függvénye. Az LS algoritmusról belátjuk, hogy $r \in [1, 3/2] \cup [\sqrt{3}, 2] \cup [6, +\infty)$ esetén optimális. $2 < r < 6$ esetén pedig megadunk optimális, vagy majdnem optimális (de LS-nél mindenesetre, a versenyképességi arány tekintetében jobb) algoritmusokat, a következők szerint:

a, Az $r \in (2, 5/2]$ esetben megadunk egy optimális algoritmust, melynek versenyképességi aránya $3/2$;

b, $r \in (5/2, 3]$ esetén egy közel optimális algoritmust $\frac{4r+2}{2r+3}$ versenyképességi

aránnyal, amikor az alsó korlát legalább $\frac{7r+4+\sqrt{r^2+8r+4}}{2r+2+2\sqrt{r^2+8r+4}}$. A két érték közötti eltérés legfeljebb 0.01417.

c, Az $r \in (3,6)$ esetben megelégszünk azzal, hogy bemutatunk egy olyan algoritmust, amelynek a versenyképességi aránya minden rögzített r paraméterérték mellett jobb mint az LS algoritmusé. A versenyképességi arány értéke $\frac{5}{3} - \frac{\delta}{18}$, ahol $\delta = \min\{\frac{6-r}{18}, \frac{1}{37}\}$, (az LS algoritmusé pedig $\frac{5}{3}$).

Mindhárom javított algoritmus futásideje $O(n)$. Továbbra is nyitott kérdés marad optimális (félig) online algoritmus keresése az $r \in (3/2, \sqrt{3})$ illetve $r \in (5/2, 6)$ intervallumokban. Az előbbiek konklúziója, hogy három gép, és közel egyforma végrehajtási idők esetén lehetséges az LS-nél hatékonyabb algoritmust megadni, ha a méretarány kettő és hat közötti szám, vagyis LS nem minden esetben optimális. Továbbá az optimális félig-online algoritmus versenyképességi aránya erősen függ az r paraméter értékétől.

2.1. A $P_2|online|C_{\max}$ feladat azon feltételek mellett, amikor adott egy puffer, és ismerjük a munkák összméretét

Ebben a fejezetben feltesszük, hogy a munkák egyesével érkeznek, és előre tudjuk a munkák méreteinek összegét. Valamint azt is feltesszük, hogy egy $l \geq 1$ méretű pufferben tárolhatunk l számú munkát. Ez azt jelenti, hogy amennyiben a puffer nincs teljesen megtöltve, akkor egy beérkező munka esetén lehetőségünk van arra, hogy ideiglenesen eltároljuk a pufferben, vagy ha akarjuk, ütemezhetjük is valamelyik gépre. Ha már pontosan l munka van tárolva a pufferben, akkor egy beérkező munka esetén vagy rögtön ütemezzük valamely gépre, vagy betehetjük a pufferbe, de ez utóbbi esetben előbb valamelyik a pufferben tárolt munkát ütemeznünk kell valamelyik gépre. Bevezetjük az $ALG2.1$ algoritmust, amelynek esetén a puffer mérete $l = 1$, az algoritmus versenyképességi aránya 5/4. Azt is megmutatjuk, hogy kisebb versenyképességi arányt nagyobb pufferméret esetén sem kaphatunk. Ezek szerint $ALG2.1$ optimális félig online algoritmus a feladatra.

Jelölje T a munkák összméretét. A munkák méretének normalizálásával elérhető, hogy $T = 8$, ezért ezt feltesszük a következőkben. Jelentse továbbá az M_1 és M_2 gépek aktuális terhelését az algoritmus futása során s_1 és s_2. Az algoritmus működésének lényege a következő: Az első munka a pufferbe kerül. Jelentse ezután X azt a munkát, ami éppen a pufferben van. ($ALG2.1$ esetén mindig lesz egy munka a pufferben, amíg az utolsó munka meg nem érkezik. Másrészt nyilvánvaló, hogy tetszőleges optimális algoritmusnál is feltehető hogy mindvégig tárol egy-egy munkát a pufferben, mert ha a puffer kiürülne, ennél nem rosszabb az, ha megvárjuk a következő munkát, és csak akkor ütemezzük valamelyiket.) Legyen p_k a soron következő munka, ekkor X és p_k közül a nagyobbikat tesszük a pufferbe, és a másikat (vagyis a $\min\{p_k, X\}$ munkát) ütemezzük, ez utóbbit Y-nal jelöljük. Ezek szerint minden pillanatban X a legnagyobb méretű munka az első k munka

között. Az algoritmus futása során arra törekszünk, hogy az $s_2 \leq 1$, valamint az $s_2 \leq s_1 \leq s_2 + 2$ feltételek ameddig lehet, érvényben maradjanak. Idővel arra fogunk kényszerülni, hogy valamelyik feltételt megsértsük, (hiszen az algoritmus futása végén $s_1 + s_2 = 8$), belátjuk azonban, hogy a további munkák már "megfelelő módon" elhelyezhetők úgy, hogy az algoritmus 5/4-versenyképességű marad.

*ALG*2.1 **Algoritmus**

1. Legyen $s_1 = s_2 = 0, X = p_1, k = 1$.
2. $k = k + 1$. Ha nincs további munka, kerüljön az X munka az M_2 gépre, és stop.
3. Legyen $Y = \min\{p_k, X\}, X = \max\{p_k, X\}$.
4. Amennyiben $s_1 + Y \leq s_2 + 2$ teljesül, ütemezzük az Y munkát az M_1 gépre, legyen $s_1 = s_1 + Y$, és goto 2.
5. Amennyiben $s_2 + Y \leq 1$ teljesül, ütemezzük az Y munkát az M_2 gépre, legyen $s_2 = s_2 + Y$, és goto 2.
6. Ha van olyan $Z \in \{X, Y\}$ munka és $i \in \{1, 2\}$ index, amelyekre $s_i + Z \in [3, 5]$ teljesül, akkor ütemezzük a Z munkát az M_i gépre, a másik $\{X, Y\}$ halmazbeli munkát, és az összes további munkát a másik gépre, és vége. Ha van olyan $i \in \{1, 2\}$ index, amelyre $s_i + X + Y \in [3, 5]$ teljesül, akkor ütemezzük az X és Y munkákat az M_i gépre, az összes további munkát a másik gépre, és vége.
7. Ha $s_1 + X > 5$ teljesül, ütemezzük az X munkát az M_2 gépre, az összes további munkát az M_1 gépre, és vége.
8. Ha $s_1 + X < 3$, ütemezzük az X munkát az M_1 gépre, az Y munkát az M_2 gépre. A további munkákat pedig a következő szabály szerint ütemezzük: (a), ha valamelyik érkező munka hossza nagyobb mint 2, ütemezzük az M_2 gépre, ellenkező esetben ütemezzük az M_1 gépre. (b), ha M_1 aktuális terhelése a $[3, 5]$ intervallumba esik, ütemezzük az összes további munkát az M_2 gépre, és vége, ellenkező esetben menjünk vissza az (a), pontra.

2.1. Tétel *Az ALG*2.1 *algoritmus versenyképességi aránya legfeljebb* 5/4.

Bizonyítás: Nyilvánvaló, hogy *ALG*2.1 megáll a 2., vagy a 6-8. lépések valamelyikében. Ha a 2. során áll meg, akkor X az utolsó ütemezésre váró munka, valamint teljesülnek az $s_2 \leq 1$ és $s_1 \leq s_2 + 2$ feltételek, amikor az algoritmus végrehajtja a 2. lépést. Emiatt $s_1 + s_2 \leq 4$ és $X = 8 - (s_1 + s_2) \geq 4$. Következik, hogy

az optimumérték legalább $C_{OPT} \geq X \geq 4$, valamint $C_{ALG2.1} = \max\{s_1, s_2+X\} = s_2 + X$. Emiatt $C_{ALG2.1} = X + s_2 \leq X + 1 \leq X + \frac{X}{4} \leq \frac{5}{4} C_{OPT}$.

Amikor az algoritmus elkezdi végrehajtani a 6-8. lépések valamelyikét, a két gép aktuális terhelésére teljesülnek az $s_2 \leq s_1 \leq s_2 + 2, s_2 \leq 1, s_1 + Y > s_2 + 2$ valamint $s_2 + Y > 1$ feltételek.

Ha megáll az algoritmus a 6. lépésben $C_{ALG2.1}/C_{OPT} \leq 5/4$ nyilvánvalóan teljesül. Emiatt tegyük fel, hogy az algoritmus nem áll meg a 6. lépésben. Megmutatjuk, hogy ebben az esetben teljesül az $s_1 + Y < 3$ feltétel. Ugyanis, az algoritmus csak úgy tudja elkerülni a 6. lépést, ha vagy $s_1+Y < 3$, vagy $s_1+Y > 5$ teljesül. Amennyiben $s_1 + Y > 5$ lenne, akkor $s_2 + 2 + X \geq s_1 + X > 5$, és emiatt $s_1 + s_2 + X + Y > 8$, ami ellentmondás. Feltehetjük tehát, hogy $s_1 + Y < 3$.

Tegyük fel, hogy az algoritmus a 7. lépésben áll meg. Ekkor $s_1 + Y < 3$ és $s_1 + X > 5$, és ezek miatt $X > Y + 2$. Továbbá az $s_2 + Y > 1$ feltétel miatt $s_2+X > 3$. Mivel azonban s_2+X értékével nem léptünk be a $[3, 5]$ intervallumba, adódik hogy $s_2+X > 5$, másként az algoritmus megállna a 6. lépésben. Az $s_2 \leq 1$ feltétel következtében $X > 4$, amiből kapjuk, hogy $C_{OPT} \geq X > 4$. Újra adódik hogy $C_{ALG2.1} = X + s_2 \leq X + 1 < \frac{5}{4} X \leq \frac{5}{4} C_{OPT}$.

Utoljára nézzük azt az esetet, amikor az algoritmus a 8. lépésben áll meg. Ekkor teljesül $s_1 + X < 3$. Következik, hogy $Y \leq X < 3$. Jegyezzük meg, hogy az $s_1 + X \geq s_1 + Y > s_2 + 2$ és $s_2 + Y > 1$ feltételek is teljesülnek, ezekből adódóan pedig $s_1 + X + Y > s_2 + 2 + Y > 3$. Az algoritmus nem áll meg a 6. lépésben, emiatt $s_1 + X + Y > 5$. $s_1 \leq s_2 + 2$ következtében $s_2 + X + Y > 3$, és így $s_2 + X + Y > 5$. Következik az $s_2 \leq 1$ feltételt felhasználva, hogy $X + Y > 4$, és így $X > 2$. Mivel $s_1 + X < 3$, kapjuk hogy $s_1 < 1$. Könnyen látható, hogy az algoritmus szabálya folytán ekkor $s_2 = 0$ teljesül, és így $X + Y > 5$.

Ha az összes továbbbi munka mérete nem nagyobb mint 2, akkor lennie kell egy olyan munkának ezek között, hogy amikor ezt az M_1 gépre ütemezzük, a gép megnövekedett terhelése a $[3, 5]$ intervallumba esik, és emiatt $C_{ALG2.1}/C_{OPT} \leq 5/4$. Tegyük fel tehát, hogy van olyan munka, jelöljük J-vel, amelynek mérete $J > 2$. Ekkor teljesül hogy $J \leq 8 - X - Y < 3$. Vegyük észre, hogy közvetlenül a J munka érkezte előtt, egyedül Y lett az M_2 gépre ütemezve, ezért az ütemezés végén csak Y és J lesznek az M_2 gépen. Ha $Y + J \leq 5$, akkor a tétel állítása triviálisan adódik, mivel $Y + J > 4$. Tegyük ezért fel, hogy $Y + J > 5$. Ha $J \leq X$, akkor az X, Y és J munkák mindegyikének mérete nagyobb mint 2, és $\max\{X, Y, J\} = X$. Ez azt jelenti, hogy optimális ütemezés esetén az Y és J munkákat egy gépre kell ütemezni, és az összes többi munkát pedig a másik gépre. Ekkor $ALG2.1$ optimális ütemezést készít el. Ellenkező esetben pedig tudjuk, hogy $J > X$, és J a legnagyobb méretű munka. Emiatt az optimumérték $C_{OPT} = Y + X > 5$, és így $C_{ALG2.1} = Y + J < 6 < \frac{5}{4} C_{OPT}$. Ezzel a bizonyítást befejeztük. □

2.2. Tétel *A feladat bármely A megoldó algoritmusának a versenyképességi aránya legalább* $5/4$*, akkor is, ha a puffer mérete* 1*-nél nagyobb.*

Bizonyítás: Legyen a puffer mérete $l \geq 1$. Az állítást oly módon bizonyítjuk, hogy megadjuk a munkáknak olyan sorozatát, amelyek esetén az optimum értéke $4N + 1$, az A algoritmus által kapott teljes átfutási idő értéke pedig legalább $5N + 2 - l$. így $C_A/C_{OPT} \geq (5N + 2 - l)/(4N + 1) \to 5/4$ $(N \to \infty)$. Legyen a munkák összmérete $8N + 2 = 2\,(4N + 1)$, ahol $N = 2k$, és $k \geq 4$ egész szám. Feltehető, hogy az első l számú munka a pufferbe kerül, és amíg érkezik még munka, addig mindig pontosan l munka marad a pufferben. A következőkben az (M_1, M_2) gépeknek az aktuális terhelését jelöljük (s_1, s_2)-vel az A algoritmus végrehajtása során, amikor az összes beérkezett munkák már ütemezésre kerültek, kivéve azt az l számú munkát, amelyek aktuálisan a pufferben vannak.

Érkezzen először $3N$ darab egységnyi méretű munka. A $(3N + 1)$-edik munka érkezése előtt, l egységnyi méretű munka van a pufferben. Emiatt $s_1 + s_2 = 3N - l$. Az általánosság megszorítása nélkül feltehetjük, hogy $s_1 \geq s_2$.

Ha a $(3N + 1)$-edik munka érkezése előtti pillanatban $(s_1, s_2) = (3n - l, 0)$ teljesül, akkor a következő, és egyben utolsó két munka hossza $2.5N + 1$. Ha ezek egy gépre kerülnek, akkor ennek a gépnek a terhelése legalább $5N + 1$. Ellenkező esetben, ha két különböző gépre kerülnek, az M_1 gép végső terhelése $3N - l + 2.5N + 1 = 5.5N - l + 1 \geq 5N - l + 2$, mivel $N \geq 2$, és ezzel készen vagyunk.

Feltehetjük tehát, hogy a $(3N + 1)$-edik munka érkezése előtti pillanatban $s_2 \geq 1$ teljesül. Ekkor a következő két munka, tehát a $(3N + 1)$-edik és $(3N + 2)$-edik munkák hossza legyen megint 1. Ha a $(3N + 3)$-adik munka érkezése előtt $(s_1, s_2) = (3N + 1 - l, 1)$ vagy $(s_1, s_2) = (3N - l, 2)$ teljesül, akkor a következő, és egyben utolsó két munka hossza $2.5N$. Ha ezek egy gépre kerülnek, ennek a gépnek a terhelése legalább $5N + 1$ lesz. Ellenkező esetben M_1 végső terhelése legalább $3N - l + 2.5N = 5.5N - l \geq 5N + 2 - l$ hiszen most $N \geq 4$. Az előző esethez hasonlóan most is következik a tétel állítása, feltehetjük ezért, hogy $s_2 \geq 3$ a $(3N + 3)$-adik munka érkezése előtt. (A gondolatmenet az előzőek szerint folytatható, mindig két eset adódik, az egyik esetben készen vagyunk, a másik esetben pedig egy további, erősebb feltétel adódik.)

Lássuk az általános esetet. Tegyük fel tehát, hogy a $(3N + 2j + 1)$-edik munka érkezése előtti pillanatban $s_2 \geq 2j + 1$ teljesül, ahol $j = 0, 1, \cdots, k - 2$. Ekkor két egységnyi méretű munka érkezik. (Ezek tehát a $(3N + 2j + 1)$-edik és $(3N + 2j + 2)$-edik munkák). A két gép megnövelt terhelésére teljesül az $s_1 + s_2 = 3N + 2j + 2 - l$ egyenlőség. Ha $s_2 \leq 2j + 2$, a következő, és egyben utolsó két munka hossza $2.5N - j$. Amennyiben ezek egy gépre kerülnek, e gép végső terhelése legalább $2\,(2.5N - j) + 2j + 1 = 5N + 1$ hiszen $s_2 \geq 2j + 1$. Ellenkező esetben az M_1 gép terhelése legalább $3N - l + 2.5N - j = 5.5N - j - l \geq 5N + 2 - l$ hiszen $N \geq 2j + 4$.

Feltehetjük tehát, hogy a $(3N + 2j + 1)$-edik és $(3N + 2j + 2)$-edik munkák érkezése után teljesül az $s_2 \geq 2j + 3$ egyenlőtlenség, minden $j = 0, 1, \cdots, k - 2$ esetén. Tekintsük azt az esetet, amikor már $j = k - 2$.

Ekkor összesen $3N+2(k-2)+2 = 3N+N-2$ munka érkezett már, mindegyik hossza egy egység. Az előbbi rész szerint tudjuk hogy az A algoritmus olyan módon ütemezte a munkákat, hogy $s_2 \geq 2\,(k-2) + 3 = 2k - 1 = N - 1$ teljesül. Most újra két egységnyi méretű munka érkezik. Ha legalább egyikük az M_2 gépre kerül, akkor $s_2 > N - 1$. Ebben az esetben jöjjön az utolsó két munka, egyikük mérete $4N + 1$, a másiké 1, és ezáltal az egyik gép terhelése legalább $5N + 1$ lesz. Ellenkező esetben pedig teljesül $(s_1, s_2) = (3N - l + 1, N - 1)$. Ekkor szintén jöjjön a két utolsó munka, de ezek mindegyikének mérete legyen $2N + 1$, most egyik gép terhelése legalább $5N - l + 2$, és ezzel a tétel bizonyítását beláttuk. □

2.1. Megjegyzés. Tekintsük a következő, az előbbihez hasonló(nak tűnő) félig online változatot: Előre tudjuk a munkák összhosszát, valamint megengedett az "előre tekintés", vagyis előre tudjuk mindig a soron következő $l \geq 1$ számú munka méretét. Erre a feladatra könnyen látható, hogy nem kaphatunk 4/3-nál kisebb versenyképességi aránnyal bíró algoritmust, (vagyis ez a plusz feltétel nem javít 4/3-os arányon, ami elérhető akkor is, ha csak a munkák összhosszát ismerjük előre). Tekintsük a munkák következő sorozatát: $p_1 = p_2 = T/6$, $p_3 = \cdots = p_{l+2} = \epsilon$, $p_{l+3} = p_{l+4} = 2T/6 - l\epsilon/2$, valamint $p_1 = p_2 = T/6$, $p_3 = \cdots = p_{l+2} = \epsilon$, $p_{l+3} = T/2$, $p_{l+4} = T/6 - l\epsilon$, ahol T-vel jelöltük a munkák (előre ismert) összhosszát, valamint ϵ elegendően kicsiny pozitív szám. Arra következtethetünk tehát, hogy a "lookahead", "előre tekintés" nem segít versenyképességi arány tekintetében, amikor a munkák összhosszát előre ismerjük.

2.2. A $P_2|online|C_{\max}$ feladat egy másik félig online változata: előre ismerjük a munkák összméretét, és két egymástól független ütemezést készíthetünk

E fejezetben ismét feltesszük, hogy a munkák egymás után, egyenként érkeznek, és előre tudjuk a munkák méreteinek összegét, továbbá egyszerre két ütemezést is készíthetünk. (Mindegyik munkát megkettőzzük, és egyiket az egyik ütemezésnél rendeljük hozzá valamelyik géphez, a másikat pedig a másik ütemezésnél. A következő munka csak ezután érkezik.) Végül, az utolsó munka megérkezte után a jobbik ütemezést választjuk. Erre a félig online esetre megkonstruáljuk az $ALG2.2$ optimális algoritmust, amelynek versenyképességi aránya 6/5.

A két ütemezést α és β ütemezésnek fogjuk nevezni. A munkák hosszainak normalizálásával elérhetjük, hogy az összhosszúság $T = 10$ legyen, ezt a következőkben feltesszük.

Jelölje most s_1 és s_2 az M_1, M_2 gépek aktuális terhelését az α ütemezés esetén, míg ugyanez a β ütemezés esetén, vagyis az M_1, M_2 gépek aktuális terhelése legyen

t_1 és t_2. Bevezetjük a következő jelölést: $k = \min\{j | \sum_{i=1}^{j} p_i \geq 4\}$, és legyen $p_k = Y$. Ezek szerint p_k a legelső olyan munka, amelyikre az első k számú munka összhossza legalább 4. Jegyezzük meg, hogy lefeljebb egy olyan munka lehet, amelyik p_k előtt érkezik, és hossza több mint 2. Nem biztos hogy van ilyen munka, ha van ilyen munka, jelöljük X-szel. Hasonlóképpen jelöljük Z-vel a legelső olyan munkát, amelyik p_k után érkezik és hossza nagyobb mint 2, ha van ilyen munka. Ezek után lássuk az $ALG2.2$ által készített α és β ütemezéseket:

Az $ALG2.2$ algoritmus által készített α ütemezés

1. Legyen $s_1 = s_2 = 0, l = 1$.
2. Amennyiben $s_1 + p_l < 4$, ütemezzük a p_l munkát az M_1 gépre, legyen $s_1 = s_1 + p_l$, és goto 5.
3. Ha $4 \leq s_1 + p_l \leq 6$, ütemezzük a p_l munkát az M_1 gépre, az összes többi munkát az M_2 gépre, és vége.
4. Ha $s_1 + p_l > 6$, ütemezzük a p_l munkát az M_2 gépre, legyen $s_2 = s_2 + p_l$, és goto 5.
5. $l = l + 1$. Ha nincs több munka, vége. Ellenkező esetben goto 2.

Az $ALG2.2$ algoritmus által készített β ütemezés

1. Legyen $t_1 = t_2 = 0, l = 1$.
2. Amíg $\sum_{i=1}^{l} p_i \leq 2$ teljesül, ütemezzük a p_l munkát M_1 -re, legyen $t_1 = t_1 + p_l$ és $l = l + 1$.
3. Amíg $2 < \sum_{i=1}^{l} p_i < 4$, ütemezzük a p_l munkát az LS algoritmus szerint, módosítsuk ennek megfelelően t_1 és t_2 értékét, legyen $l = l + 1$.
4. Ha $4 \leq Y + t_i \leq 6$ valamely $i \in \{1, 2\}$ esetén, ütemezzük az Y munkát az M_i gépre, az összes további munkákat pedig a másik gépre, és vége.
5. Ütemezzük az Y munkát M_1-re.
6. A Z kivételével az összes munkát ütemezzük az M_1 gépre. Ha létezik a Z munka, ütemezzük az M_2 gépre, és vége.

2.2 Megjegyzés Tekintsük a β ütemezést közvetlenül a 3.lépés után. Két lehetőség fordulhat elő: (i). Ha nem létezik az X munka, akkor ebben a pillanatban $|t_1 - t_2| \leq 2$ teljesül. (ii). Ellenkező esetben, vagyis ha létezik az X munka, akkor egyedül az X munka került még csak az M_2 gépre.

A következőkben az egyszerűség kedvéért azt mondjuk, hogy egy ütemezés *megfelelő*, ha az összes munka ütemezése után a teljes átfutási ideje nem több mint 6. Ugyanis nyilvánvaló, hogy ha az α vagy β ütemezés megfelelő, akkor teljesül a belátandó $C_{ALG2.2}/C_{OPT} \leq 6/5$ egyenlőtlenség.

2.3. Tétel *Az ALG*2.2 *algoritmus versenyképességi aránya* 6/5.

Bizonyítás: Tekintsük az Y munka érkezésének pillanatát. Feltehetjük, hogy $s_1 + Y > 6$, ellenkező esetben ugyanis az α ütemezés megfelelő.

Először tegyük fel, hogy X munka nem létezik. Ebben a pillanatban, az összes korábbi munka az α ütemezésnél az M_1 gépre lett ütemezve, emiatt teljesülnek a következők: $s_2 = 0$, $t_1 + t_2 = s_1 < 4$. Mivel $s_1 + Y > 6$, következik, hogy $Y > 2$. A 2. Megjegyzés folytán pedig teljesül $|t_1 - t_2| \leq 2$. Három egymást kizáró, és az összes lehetőséget kimerítő esetet vizsgálunk meg, az alábbiak szerint:

1. eset $t_i + Y < 4, i = 1, 2$. Ekkor $t_1 + t_2 + 2Y = s_1 + 2Y < 8$. Felhasználva az $s_1 + Y > 6$ feltételt $Y < 2$ adódik, ami ellentmondás.

2. eset $t_i + Y \geq 4, i = 1, 2$. Ekkor teljesülnek a $t_i + Y > 6$ feltételek, ellenkező esetben a β ütemezés megfelelő. Következik, hogy $t_1 + t_2 + 2Y = s_1 + 2Y > 12$. Mivel $s_1 < 4$, adódik hogy $Y > 4$. Ebből következik, hogy az α ütemezésnél ekkor egyedül csak Y van az M_2 gépre ütemezve. Ha $Y \leq 6$, akkor az α ütemezés megfelelő. Ha viszont $Y > 6$, akkor pedig az α ütemezés optimális.

3. eset Valamely $i, j \in \{1, 2\}, i \neq j$, indexekre teljesül, hogy $t_i + Y < 4$ és $t_j + Y \geq 4$. Ekkor $t_j + Y > 6$ és emiatt $|t_1 - t_2| > 2$, ami ismét ellentmondás.

Most tegyük fel, hogy létezik az X-szel jelölt munka. Amikor az Y munka érkezik, teljesül az $s_1 + Y > 6$ feltétel, továbbá a β ütemezésnél egyedül X lett ütemezve az M_2 gépre. Ha $X + Y \leq 6$, akkor a β ütemezés megáll a 4. lépésben, és megfelelő ütemezés. Ezért tegyük fel, hogy $X + Y > 6$. Közvetlenül Y ütemezése után, az α ütemezésben csak az Y munka, míg a β ütemezésben csak az X munka került az M_2 gépre. Ha $Y \geq 4$, akkor az előbbi 2. eset vizsgálatához hasonló módon látható, hogy az α ütemezés optimális, vagy legalábbis megfelelő. Emiatt tegyük fel, hogy $Y < 4$. Ha nem létezik a Z munka, vagyis más szóval az összes további munka mérete kisebb mint 2, akkor az α ütemezés megáll a 3. lépésben és megfelelő. Tegyük fel tehát, hogy létezik a Z munka. Emlékezzünk vissza, hogy az Y munka érkeztekor $s_1 + Y > 6$ teljesül, emiatt az összes további munkák összhossza kevesebb mint 4. így csak egy olyan munka van, amelyik Y után érkezik, és a hossza nagyobb mint 2. Ha az α ütemezésnél a Z munka az M_1 gépre került, akkor ez az ütemezés megfelelő. Ellenkező esetben Z az M_2 gépre lett

ütemezve mindkét ütemezésnél, tehát az α és β ütemezések mindegyike esetében. Az ütemezések elkészülte után az α ütemezésnél csak Y és Z, a β ütemezésnél pedig csak X és Z kerültek az M_2 gépre.

Ha $X+Z \leq 6$ vagy $Y+Z \leq 6$ teljesül, akkor egyik ütemezés megfelelő. Emiatt vizsgáljuk azt az esetet, amikor $X+Z > 6$ és $Y+Z > 6$. Ha $Z < \max\{X, Y\}$, akkor egyik ütemzés optimális. Ellenkező esetben Z a leghosszabb idejű munka, és az optimális teljes átfutási idő pedig $C_{OPT} = Y+X > 6$. Ha még az is teljesül, hogy $X+Z \leq \frac{6}{5}(Y+X)$ vagy $Y+Z \leq \frac{6}{5}(Y+X)$, akkor $C_{ALG2.2}/C_{OPT} \leq 6/5$. Emiatt tegyük fel, hogy $Y+Z > \frac{6}{5}(Y+X)$ és $X+Z > \frac{6}{5}(Y+X)$. Ekkor egyszerű számolással kapjuk, hogy $Z > 4$ és ennek következtében $X+Y+Z > 10 = T$, ami ellentmondás, és ezzel minden esetet megvizsgáltunk, a bizonyítás kész. □

2.4. Tétel *Tetszőleges A algoritmus versenyképességi aránya legalább* $6/5$.

Bizonyítás: Legyen a munkák összmérete 10. Az első négy munka hossza legyen egységnyi. Ha az ezek ütemezése utáni állapotban $\min\{s_1, s_2, t_1, t_2\} \geq 1$ teljesül, vagyis mindkét ütemezés esetén kerül mindkét gépre legalább egy munka, akkor a következő, és egyben utolsó két munka jön, egyiknek a mérete 5, a másiké pedig 1 legyen. Nyilvánvaló, hogy $C_A/C_{OPT} \geq 6/5$. Ellenkező esetben viszont legalább az egyik ütemezés esetén valamelyik gépre nem lett ütemezve még munka, legyen például az α ütemezésnél a gépek aktuális terhelése $(s_1, s_2) = (4, 0)$. Ha $(t_1, t_2) \in \{(4,0), (3,1)\}$, akkor a következő két munka legyen az utolsó, mindkettő mérete legyen 3. Ellenkező esetben, ha $(t_1, t_2) = (2, 2)$, a két utolsó munka mérete legyen 4 és 2. Mindegyik esetben könnyen igazolhatóan $C_A/C_{OPT} \geq 6/5$ adódik. □

2.3. Hasonló méretű munkák félig online ütemezése három gépen

A fejezet további részében a $P_3|online|C_{\max}$ feladatnak azzal a félig online változatával foglalkozunk, amikor előre tudjuk, hogy a munkák végrehajtási ideje közel egyforma. Először néhány később szükséges eredményt közlünk, ezután az LS algoritmus versenyképességi arányát határozzuk meg az összes r paraméter esetén, majd pedig a $2 < r < 6$ intervallumba eső méretarányokra, (itt LS nem optimális), közlünk hatékonyabb algoritmusokat.

Előre tudjuk tehát, hogy a munkák hossza a p és rp konstansok között van, (ahol $p > 0$, $r \geq 1$). Az általánosság megszorítása nélkül feltehetjük, hogy $p = 1$. Nem követeljük meg, hogy legyenek pontosan 1 és r méretű munkák, vagyis az előbbi 1 és r csak alsó illetve felső korlátai a feladatok méreteinek. Feltesszük, hogy a munkák a $p_1, p_2, \ldots, p_n$ sorrendben érkeznek. Mivel megengedjük, hogy

1 és/vagy r méretű munka nem fordul elő, $p_{\min} = \min_{i=1,\cdots,n} p_i$ jelöli az aktuális minimális munkaméretet. Eztán némi előkészítésre van szükségünk:

2.5. Lemma *Tetszőleges $1 \leq r \leq 2$ paraméterérték esetén minden, legfeljebb hat elemből álló J feladatsorozat esetén teljesül a $\frac{C_{LS}}{C^*} \leq 1 + \frac{r-1}{2}$ egyenlőtlenség.*

Bizonyítás: Ha $n = 1, 2, 3$, akkor LS optimális megoldást határoz meg. Ha $n = 4, 5, 6$, akkor LS az első három munkát különböző gépekre teszi. Az általánosság megszorítása nélkül feltehetjük, hogy $p_1 \leq p_2 \leq p_3$, az LS algoritmus a p_i munkát a $P_i, i = 1, 2, 3$ gépre ütemezési. Hasonlóképpen feltehető, hogy a C_{LS} teljes átfutási időt p_n határozza meg, különben az utolsó munka elhagyható. Következik, hogy $C_{LS} = p_{n-3} + p_n$ teljesül, minden $n = 4, 5, 6$ esetén.

Tegyük fel, hogy létezik olyan optimális ütemezés, amikor p_{n-3} vagy p_n nem egyedüli munka a saját gépén, például p_{n-3} és p_j egy optimális gépre van ütemezve. Ekkor $\frac{C_{LS}}{C^*} \leq \frac{p_{n-3}+p_n}{p_{n-3}+p_j} \leq \frac{p_{n-3}+r}{p_{n-3}+1} = 1 + \frac{r-1}{p_{n-3}+1} \leq 1 + \frac{r-1}{2}$. Emiatt feltehető, hogy p_{n-3} is és p_n is egyedüli munkák egy-egy optimális gépen, *bármely* optimális ütemezés esetén, emiatt csak $n = 4$ lehet. Valóban, ha a harmadik optimális gép legalább három munkát végez, akkor $1 \leq r \leq 2$ miatt az ütemezés nem romlik úgy, hogy valamelyik munkát áttesszük valamelyik másik gépre, (amelyik csak egy munkát végez). Emiatt a harmadik optimális gép pontosan két munkát végez, és így $n = 4$. Emiatt viszont $p_{n-3} = p_1$, továbbá p_1 és p_4 bármelyike nagyobbak mint a másik két munka, hiszen optimális ütemezés esetén egyedül vannak a gépeiken. Ez viszont ellentmond a $p_1 \leq p_2 \leq p_3$ feltételnek, így készen vagyunk. □

A következő, könnyen igazolható lemma a későbbiekben rendkívül hasznosnak bizonyul majd.

2.6. Lemma *Legyen A egy olyan algoritmus, amely az egymást követő munkákat közbeiktatott várakozási idők nélkül ütemezi, továbbá az utolsó munkát, T-t olyan gépre ütemezi, amelynek az aktuális terhelése minimális. Tegyük fel továbbá azt is, hogy a T munka határozza meg az algoritmus teljes átfutás idejét. Ekkor $T \geq \frac{3}{2}(C_A - C^*)$.*

Bizonyítás: Legyen s a T munka kezdési ideje az A algoritmusnál, továbbá legyen a a munkák hosszainak összege. Ekkor teljesül $s \leq \frac{a-T}{3}$, és $C^* \geq \frac{a}{3}$. Ezeket az egyenlőtlenségeket az $C_A = s + T$ egyenletbe helyettesítve az állítás adódik.□

A fejezet további részében jelölje P_i az i-edik gépet, és azokat a munkákat is így jelöljük, amelyek a heurisztikus algoritmus által erre a gépre lettek ütemezve, a p_n munka megérkezése előtti pillanatban. Jelölje $l(P_i)$ a P_i gép aktuális terhelését a p_n munka ütemezése előtti pillanatban, és legyen ekkor a minimális aktuális terhelés $s = \min_{i=1,\cdots,m} l(P_i)$. Jelölje hasonlóképpen P_i^* mind az i-edik optimális gépet, mind pedig az erre az optimális gépre ütemezett munkákat (az összes munka optimális ütemezése esetén).

2.4. Az LS algoritmus versenyképességi aránya

E fejezetet arra szenteljük, hogy megállapítsuk az LS algoritmus versenyképességi arányát, tetszőleges $r \geq 1$ méretarány esetén. Ez a következő tételben foglalható össze:

2.7. Tétel *Az LS algoritmus versenyképességi aránya*

$$R_{LS} = \begin{cases} 1 + \frac{2(r-1)}{3}, & ha \quad 1 \leq r \leq \frac{3}{2}, \\ 2 - \frac{3}{r+3}, & ha \quad \frac{3}{2} < r \leq \sqrt{3}, \\ \frac{r+1}{2}, & ha \quad \sqrt{3} < r \leq 2, \\ 2 - \frac{1}{r}, & ha \quad 2 < r \leq 3, \\ \frac{5}{3}, & ha \quad 3 < r, \end{cases}$$

továbbá LS optimális félig-online algoritmus az $1 \leq r \leq 3/2$, $\sqrt{3} \leq r \leq 2$ és $r \geq 6$ esetekben.

Bizonyítás: A tételt az r paraméter intervallumai szerint egymás után bizonyítjuk. Először is, lássunk két egyszerű esetet:

2.8. Tétel *([45]) Ha $1 \leq r \leq 3/2$, LS versenyképességi aránya $1 + \frac{2(r-1)}{3}$, emiatt optimális.*

2.9. Tétel *Ha $r \geq 3$, LS versenyképességi aránya $\frac{5}{3}$, emiatt optimális minden $r \geq 6$ esetén.*

Bizonyítás: Graham klasszikus [35] munkája szerint, tudjuk, hogy $\frac{C_{LS}}{C^*} \leq \frac{5}{3}$ teljesül tetszőleges $r \geq 1$ esetén, a $\{p_1 = \cdots = p_6 = 1, p_7 = 3\}$ feladathalmaz pedig mutatja, hogy $r \geq 3$ esetén LS versenyképességi aránya nem lehet jobb, mint $\frac{5}{3}$. Másrészt ha $r \geq 6$, tetszőleges félig-online algoritmus hatékonysága a $\{p_1 = p_2 = p_3 = 1, p_4 = p_5 = p_6 = 3, p_7 = 6\}$ [31] sorozatra nem jobb mint $\frac{5}{3}$. Ugyanis, ha egy A az első három munkát legfeljebb két gépre ütemezi, akkor nem érkezik több munka, és $\frac{C_A}{C^*} \geq 2$. Ellenkező esetben A három különböző gépre ütemezi őket, ekkor érkezik a következő három munka. Ha ezek nem három különböző gépre kerülnek, akkor újra $\frac{C_A}{C^*} \geq \frac{7}{4}$ lesz. Végül ha a második három munka is három különböző gépre lesz ütemezve, akkor jön az utolsó munka, p_7, és $\frac{C_A}{C^*} = \frac{10}{6}$. □

A következőkben elég tehát csak a $3/2 \leq r \leq 3$ esettel foglalkoznunk.

2.4.1. Az LS algoritmus versenyképességi aránya $3/2 < r \leq 2$ esetén

Ebben a fejezetben belátjuk az alábbi tételt:

2.10. Tétel *$3/2 < r \leq 2$ esetén az LS algoritmus versenyképességi aránya*

$$R_{LS} = \max\{\frac{r+1}{2}, 2 - \frac{3}{r+3}\} = \begin{cases} 2 - \frac{3}{r+3}, & ha \ \ \frac{3}{2} < r < \sqrt{3}, \\ \frac{r+1}{2}, & ha \ \ \sqrt{3} \leq r \leq 2, \end{cases}$$

és így LS optimális minden $\sqrt{3} \leq r \leq 2$ esetén.

Bizonyítás: A 2.5. Lemma miatt elég csak a $|\mathcal{I}| > 6$ esetet vizsgálni. Az LS algoritmus legrosszabb eset vizsgálatához $3/2 < r \leq 2$ és $|\mathcal{I}| > 6$ esetén, tekintsük a következő feladat sorozatot: $\mathcal{I}_1 = \{1, 1, \frac{2r}{3}, \frac{r+3}{3}, \frac{r+3}{3}, \frac{6-r}{3}, r\}$. Erre $C_{LS} = \frac{4r+6}{3}, C^* = \frac{2r+6}{3}$, és így $\frac{C_{LS}}{C^*} = \frac{2r+3}{r+3} = 2 - \frac{3}{r+3}$.

2.11. Lemma *Legyen I olyan sorozat, amelyre $|\mathcal{I}| > 6$, ekkor minden $\frac{3}{2} < r \leq 2$ esetén teljesül $\frac{C_{LS}}{C^*} \leq \frac{2r+3}{r+3}$.*

Bizonyítás: Tegyük fel, hogy $\mathcal{I}$ olyan ellenpélda, amelyre $|\mathcal{I}| > 6$, továbbá $C^* = \frac{2r+6}{3}t$ és $C_{LS} > \frac{4r+6}{3}t$ valamely $t > 0$ esetén. Először is a 2.6 Lemma miatt $p_n > rt$. Mivel $p_n/p_{\min} \leq r$ teljesül, tudjuk hogy $p_{\min} > t$. A kívánt ellentmondáshoz a következő lemmák segítségével fogunk eljutni.

2.12. Lemma $p_n < \frac{2r+3}{3}t$.

Bizonyítás: Ha $p_{\min} > \frac{2r+6}{9}t$, akkor $3p_{\min} > \frac{2r+6}{3}t = C^*$, és emiatt egy optimális gép sem végezhet kettőnél több munkát, ami ellentmond a $|\mathcal{I}| > 6$ feltételnek, és így ellentmondáshoz jutunk. Emiatt $p_{\min} \leq \frac{2r+6}{9}t$. Ebből következik viszont, hogy $p_n \leq rp_{\min} \leq \frac{2r^2+6r}{9}t < \frac{2r+3}{3}t$, (az utolsó egyenlőtlenség akkor áll fenn, ha $r < 3/\sqrt{2}$, ami most teljesül.) □

2.13. Lemma *Ha $p_n \in P_1^*$, akkor $|P_1^*| = 2$.*

Bizonyítás: Mivel $p_n + 2p_{\min} > rt + 2t > \frac{2r+6}{3}t = C^*$, emiatt $|P_1^*| \leq 2$. Ha $|P_1^*| = 1$, akkor $\sum_{i=1}^{n-1} p_i \leq 2C^* = \frac{4r+12}{3}t$. Emiatt kapjuk, hogy az LS által készített ütemezésnél a p_n munka kezdési időpontjára teljesül, hogy $s \leq (\sum_{i=1}^{n-1} p_i)/3 \leq \frac{4r+12}{9}t$ és így, az előző lemmát is felhasználva adódik, hogy $C_{LS} \leq \frac{4r+12}{9}t + p_n < \frac{4r+12}{9}t + \frac{2r+3}{3}t \leq \frac{4r+6}{3}t$, ami ellentmondás. Emiatt $|P_1^*| = 2$. □

2.14. Lemma $|P_i| = 2$ *minden $i = 1, 2, 3$ esetén, és így $|\mathcal{I}| = 7$.*

Bizonyítás: Az általánosság megszorítása nélkül feltehető, hogy csak az utolsóként ütemezett munka befejezésének időpontja egyenlő a teljes átfutási idővel.

Ha van egy olyan $i \in \{1,2,3\}$ index, amelyre $|P_i| \geq 3$, akkor ennek a gépnek a terhelése legalább $3p_{\min} > 3t$ az utolsó munka, vagyis p_n érkezése előtt. Mivel $s \leq \frac{3C^*-p_n}{3} < \frac{2r+6}{3}t - \frac{rt}{3} = \frac{r+6}{3}t < 3t$, az LS nem erre a gépre ütemezi a p_n munkát. így az összes munka (LS által történő) ütemezése után van két gép, amelyikek közül az egyiknek a terhelése legalább $C_{LS} > \frac{4r+6}{3}t$, és a másik terhelése legalább $3t$. Mivel a munkák összmérete legfeljebb $3 \cdot \frac{2r+6}{3}t = (2r+6)t$, a harmadik gép terhelése legfeljebb $\frac{2r+3}{3}t$. Emiatt a p_n munka kezdési időpontja legfeljebb $\frac{2r+3}{3}t$. Tudjuk, hogy $C_{LS} > \frac{4r+6}{3}t$, emiatt kapjuk, hogy $p_n > \frac{2r+3}{3}t$, ami ellentmond a 2.12 Lemmának. Ezek szerint kaptuk, hogy $|P_i| \leq 2$ minden $i = 1,2,3$ esetén. Mivel $|\mathcal{I} \setminus \{p_n\}| \geq 6$, következik, hogy $|P_i| = 2$ minden i-re, és $|\mathcal{I}| = 7$. □

Most már készen vagyunk arra, hogy befejezzük a 2.11. Lemma bizonyítását. Mivel $4p_{\min} > 4t \geq \frac{2r+6}{3}t$, következik, hogy $|P_i^*| \leq 3, i = 1,2,3$. Ezt összevetve a 2.13 és 2.14 Lemmákkal, feltehetjük, hogy $P_1^* = \{p_n, X\}$, $P_2^* = \{Y, U\}$, és $P_3^* = \{A, B, C\}$. Ha van olyan P_k gép, amelyik az $\{A, B, C\}$ munkák közül kettőt végez a p_n munka megérkezése előtt az LS által készített ütemezésnél, akkor a P_k gép terhelése legfeljebb $C^* - p_{\min} < \frac{2r+6}{3}t - t = \frac{2r+3}{3}t$. Erre a gépre ütemezve a p_n munkát kapjuk, hogy $C_{LS} \leq \frac{2r+3}{3}t + rt < \frac{4r+6}{3}t$, ami ellentmondás. Ebből következik, hogy az A, B, és C munkákat az LS különböző gépekre ütemezi. Tekintsük azt a gépet, amelyik az X munkát végzi. Az általánosság megszorítása nélkül feltehető, hogy a másik munka, ami erre a gépre lett ütemezve, az A munka. Emiatt $X + A + p_n > \frac{4r+6}{3}t$. Mivel $X + p_n \leq C^* = \frac{2r+6}{3}t$, kapjuk, hogy $A > \frac{2r}{3}t$, emiatt $B + C \leq C^* - A < \frac{2r+6}{3}t - \frac{2r}{3}t = 2t$. Ebből következik, hogy $\min\{B, C\} < t < p_{\min}$, ellentmondás, és ezzel a bizonyítás teljes. □

Végül az alábbiakban belátjuk a 2.10. Tételt.

Tetszőleges olyan $\mathcal{I}$ sorozatra, amelyre $|\mathcal{I}| \leq 6$, tudjuk hogy $\frac{C_{LS}}{C^*} \leq \frac{r+1}{2}$ teljesül, ha pedig a $\mathcal{I}$ sorozatra $|\mathcal{I}| > 6$ áll, akkor $\frac{C_{LS}}{C^*} \leq \frac{2r+3}{r+3}$. Emiatt $\frac{C_{LS}}{C^*} \leq \max\{\frac{2r+3}{r+3}, \frac{r+1}{2}\}$ minden esetben teljesül. Jegyezzük meg, hogy $\frac{2r+3}{r+3}$ illetve $\frac{r+1}{2}$ folytonos, szigorúan növekvő függvénye r -nek, továbbá $\frac{2r+3}{r+3} \leq \frac{r+1}{2}$ pontosan akkor, ha $r \geq \sqrt{3}$, a felső becslést bebizonyítottuk. Másrészt a $\mathcal{I}_0 = \{1,1,1,r\}$ feladatsorozat bizonyítja az LS algoritmus optimalitását $\sqrt{3} \leq r \leq 2$ esetén. □

2.4.2. Az LS algoritmus versenyképességi aránya $2 < r < 3$ esetén

Az LS algoritmus legrosszabb eset vizsgálatához a $2 < r < 3$ esetben, tekintsük a $\mathcal{I}_2 = \left\{1, 1, \frac{1}{r-1}, \frac{1}{r-1}, \frac{r}{r-1}\right\}$ feladat sorozatot. Erre $C_{LS} = \frac{2r-1}{r-1}$, $C^* = \frac{r}{r-1}$, és így $\frac{C_{LS}}{C^*} = \frac{2r-1}{r}$.

2.15. Tétel *Minden $2 < r < 3$ paraméter esetén teljesül, hogy $\frac{C_{LS}}{C^*} \leq \frac{2r-1}{r}$.*

Bizonyítás: Tegyük fel, hogy a $\mathcal{I}$ feladat sorozatra $C^* = rt$ és $C_{LS} > (2r-1)t$ valamely $t > 0$-re. Az előző bizonyításhoz hasonlóan itt is feltehetjük, hogy az

LS teljes átfutási ideje az utolsó munka, p_n befejezésének idejével egyenlő. (A többi munka vagy korábban, vagy legfeljebb ebben az időpontban fejeződik be.) A tétel bizonyításához ezután először belátjuk a következő négy lemmát:

2.16. Lemma $p_{\min} > \frac{9r-9}{2r^2}t$.

Bizonyítás: A 2.6 Lemma miatt $p_n > \frac{3(r-1)}{2}t$ teljesül. Emiatt $p_{\min} > \frac{3r-3}{2}t \cdot \frac{1}{r} \cdot \frac{3}{r} = \frac{9r-9}{2r^2}t$. □

2.17. Lemma *Tegyük fel, hogy $p_n \in P_1^*$. Ekkor $|P_1^*| = 1$.*

Bizonyítás: Ellenkező esetben van olyan $X \neq p_n$ munka, amelyre $X \in P_1^*$, ekkor $X < rt - \frac{3r-3}{2}t = \frac{3-r}{2}t$. Emiatt $\frac{p_n}{X} > \frac{3r-3}{3-r} > r$, (ahol a második egyenlőtlenség $r > \sqrt{3}$ miatt teljesül), ellentmondást kaptunk. □

2.18. Lemma *Tegyük fel, hogy p_n a P_i gépre van ütemezve az LS által meghatározott ütemezésben. Ekkor $|P_i| = 1$.*

Bizonyítás: A 2.17 Lemma miatt $\sum_{i=1}^{n-1} p_i \leq 2C^* = 2rt$ teljesül. Ebből következik, hogy a p_n munka kezdési időpontjára teljesül, hogy $s \leq \frac{\sum_{i=1}^{n-1} p_i}{3} \leq \frac{2rt}{3}$. Ha $|P_i| > 1$, akkor van egy olyan munka, mondjuk az Y, amelyre $Y \in P_i$ és $Y \leq \frac{2rt}{3}/2 = \frac{rt}{3}$. Ekkor $\frac{p_n}{Y} > \frac{9r-9}{2r} \geq r$, (ahol az utolsó egyenlőtlenség $3/2 \leq r \leq 3$ esetén teljesül), ezáltal ellentmondást kaptunk. □

2.19. Lemma *Jelöljük Z-vel azt a munkát, amely az LS-ütemezésnél ugyanarra gépre kerül, mint p_n, és tegyük fel, hogy $Z \in P_2^*$. Ekkor $|P_2^*| > 1$.*

Bizonyítás: Tegyük fel, hogy Z egyedüli munka a P_2^* optimális gépen. Jegyezzük meg, hogy p_n egyedüli munka a P_1^* optimális gépen. Emiatt minden munka a Z és p_n kivételével a P_3^* optimális gépen van, és ezek a munkák az LS-ütemezésnél két gépre (a P_i-től különböző két gépre) vannak ütemezve. Ekkor viszont van olyan gép, amelynek a terhelése a p_n munka ütemezése előtt legfeljebb $\frac{C^*}{2} = \frac{rt}{2}$. Erre a gépre ütemezve p_n-t, kapjuk, hogy $C_{LS} \leq \frac{rt}{2} + rt \leq (2r-1)t$, ami ellentmondás. □

Most visszatérünk a 2.15 Tétel bizonyításához. Tegyük fel, hogy U egy másik munka (Z-től különböző), és a P_2^* optimális gépre lett ütemezve. Ekkor $U \leq C^* - Z = rt - Z$. Mivel $Z = C_{LS} - p_n > (2r-1)t - p_n$ továbbá $\frac{3r-3}{2}t < p_n \leq rt$, kapjuk, hogy $\frac{p_n}{U} \geq \frac{p_n}{rt-Z} > \frac{p_n}{rt-((2r-1)t-p_n)} = \frac{p_n}{t-rt+p_n} \geq \frac{rt}{t-rt+rt} = r$, ami a kívánt ellentmondás. □

2.5. Javított félig-online algoritmusok $2 < r < 6$ esetén

A következőkben, C_k^*-gal jelöljük az első k munkára, $p_1, ..., p_k$-ra vonatkozó optimális teljes átfutási időt, továbbá legyen $C_{1,k} = (\sum_{i=1}^{k} p_i)/3$, ami egy triviális alsó korlátja C_k^*-nak. Jelölje $L(P_i)$ a P_i gép aktuális terhelését, a heurisztikus algoritmus futása közben. Először bevezetünk egy új algoritmust, az $ALG(\gamma)$ algoritmust, amely a következő munkát a lehető legnagyobb terhelésű gépre ütemezi, de úgy, hogy az ezáltal megnövekedett terhelés ne haladja meg az aktuális optimum érték vagy ennek alsó korlátjának a γ-szorosát, ahol γ az algoritmus (tervezett) versenyképességi aránya. Az algoritmus formális leírása a következő:

$ALG(\gamma)$ **algoritmus**

1. Az első három munkát ütemezzük különböző gépekre, legyen $k = 4$.
2. $k \leq 7$ esetén számoljuk ki C_k^*-ot, és legyen $C = \gamma C_k^*$. Ha $k > 7$, számoljuk ki $C_{1,k}$-t, és legyen $C = \gamma C_{1,k}$.
3. Legyen I azon gépek halmaza, amelyek aktuális terhelése legfeljebb $C - p_k$. Ha $I = \emptyset$, az algoritmus végetér.
4. Legyen P_{i_0} olyan I halmazbeli gép, amelynek maximális a terhelése. Ütemezzük a p_k munkát a P_{i_0} gépre.
5. $k = k + 1$. Ha nem jön több munka, vége, ellenkező esetben menjünk a 2. lépésre.

2.5.1. Az $ALG(3/2)$ félig-online algoritmus optimális $2 < r \leq 5/2$ esetén

Belátjuk, hogy $ALG(3/2)$ optimális algoritmus tetszőleges $2 < r \leq 5/2$ esetén.

2.20. Tétel *Tetszőleges* $2 < r \leq 5/2$ *esetén* $ALG(3/2)$ *versenyképességi aránya* $3/2$*, ennek következtében optimális algoritmus.*

Bizonyítás: Nyilvánvaló, hogy tetszőleges $r > 2$ esetén az $\{1, 1, 1, 2\}$ feladat sorozat miatt, a feladatnak $3/2$ alsó korlátja. Belátjuk, hogy $ALG(3/2)$ hatékonysága nem rosszabb, mint $3/2$. Elég belátnunk, hogy $ALG(3/2)$ mindegyik munkát képes ütemezni. Tegyük fel tehát, hogy létezik olyan $\mathcal{I}$ sorozat, amelyre az $ALG(3/2)$ megállni kényszerül a 3. lépésben, vagyis a soron következő munkát nem képes ütemezni. Az általánosság megszorítása nélkül feltehetjük, hogy az utolsó munka, p_n az, amelyet nem tudott az algoritmus ütemezni. Először megmutatjuk néhány tulajdonság teljesülését a $\mathcal{I}$ sorozatra.

2.21. Lemma $|\mathcal{I}| \leq 7$.

Bizonyítás: Jelöljük a feladatok összméretét a következőképpen: $A = \sum_{i=1}^{n} p_i$. Ekkor a C konstansnak az aktuális értéke, éppen abban a pillanatban, amikor a p_n munkát kellene ütemeznünk legalább $C \geq \frac{3}{2} \cdot \frac{A}{3} = \frac{A}{2}$. Emiatt teljesül, hogy $s + p_n > \frac{A}{2}$. Másrészt nyilvánvalóan teljesül $3s + p_n \leq A$. E kettő egyenlőtlenség összevetéséből adódik $p_n > \frac{A}{4}$ és $s < \frac{A}{4}$. Az $r \leq \frac{5}{2}$ feltétel következménye $p_{\min} \geq \frac{p_n}{r} > \frac{A}{10}$. Az általánosság megszorítása nélkül feltehetjük, hogy $l(P_3) = s$. Ekkor viszont $s + p_n > \frac{A}{2}$ következményeképpen $l(P_1) + l(P_2) < \frac{A}{2}$ adódik. Emiatt a P_1, P_2 gépekre összesen legföljebb négy munka került a p_n munka érkezése előtti pillanatban. Továbbá $l(P_3) = s < \frac{A}{4}$ miatt ugyanekkor a P_3 gépre legföljebb két munka került. Ezekből kapjuk hogy valóban, $|\mathcal{I}| \leq 7$. □

Az előző lemmából következik, hogy a p_n munka érkezésekor, amikor őt kell ütemeznünk, a C konstans értéke $\frac{3}{2}C^*$.

2.22. Lemma *Minden optimális ütemezés esetén van olyan gép, amely egyedül a p_n munkát végzi, továbbá $C^* < 3$, és $|I| \leq 5$.*

Bizonyítás: Jelölje C' a teljes átfutási időt, ami úgy adódik, hogy az első $n-1$ számú munkát az $ALG(3/2)$ algoritmussal ütemezzük, (és mivel az nem képes ütemezni p_n-et), az utolsó munkát az LS algoritmus által helyezzük el. Ekkor $C' = s + p_n > C = \frac{3}{2}C^*$ hiszen $ALG(3/2)$ nem képes ütemezni p_n-et. Emiatt a 2.6 Lemma miatt kapjuk, hogy $p_n \geq \frac{3}{2}(C' - C^*) > \frac{3}{2} \cdot \frac{1}{2}C^* = \frac{3}{4}C^*$. Másrészt $p_n \leq r \leq \frac{5}{2}$ következtében tudjuk, hogy $C^* \leq \frac{10}{3}$. Emiatt $C^* - p_n < 1$. Következik, hogy a p_n munka egyedül van az ő optimális gépén. Emiatt az is teljesül viszont, hogy $s \leq \frac{1}{3}(\sum_{i=1}^{n-1} p_i) \leq \frac{2}{3}C^*$. Ezt összevetve az $s + p_n > \frac{3}{2}C^*$ egyenlőtlenséggel, kapjuk, hogy $\frac{5}{2} \geq r \geq p_n > \frac{5}{6}C^*$, vagyis $C^* < 3$.

A $C^* < 3$ feltétel miatt, mivel a munkák hossza legalább 1, mindegyik optimális gép legföljebb két munkát végez, és a p_n munka pedig egyedül szerepel a gépén, következik hogy $|\mathcal{I}| \leq 5$. □

Ezek után már beláthatjuk a 2.20 Tételt. A 2.22 Lemma miatt feltehetjük, hogy $P_3^* = \{p_n\}$. Ha még $|\mathcal{I}| \leq 3$ is teljesül, $ALG(3/2)$ optimális megoldást határoz meg. Legyen $|\mathcal{I}| = 4$. Ekkor van olyan optimális gép, például legyen ez a P_1^* gép, amely legalább két munkát végez. Emiatt van olyan munka, legyen ez X, amelyiket P_1^* végzi, és így teljesül hogy $X \leq \frac{C^*}{2}$. $X \neq p_n$ hiszen a p_n munkát a P_3^* gép végzi. Emiatt az $ALG(3/2)$ szabályából következően $s \leq X \leq \frac{C^*}{2}$ teljesül. Ennek következtében viszont $C' = s + p_n \leq \frac{C^*}{2} + C^* = \frac{3}{2}C^*$, ami ellentmondás.

Tegyük fel végül, hogy $|\mathcal{I}| = 5$. Ekkor van két olyan optimális gép, amelyek két-két munkát végeznek, a harmadik optimális gép pedig egyedül az utolsó munkát, p_5-öt végzi. Legyen ennek megfelelően $P_1^* = \{X, Y\}$, $P_2^* = \{U, V\}$ ahol $Y \leq X$ és $V \leq U$ is teljesül. Ekkor $Y \leq \frac{C^*}{2}$, $V \leq \frac{C^*}{2}$ és $C_4^* \geq Y + V \geq 2$.

Emiatt a C konstans értéke a negyedik munka ütemezésekor $C = \frac{3}{2}C_4^* \geq 3 > C^*$. Most megmutatjuk, hogy $s \leq \frac{C^*}{2}$. Valóban, ha a negyedik munka X (vagy U), akkor van legalább két munka az első négy között, amelyek hossza legfeljebb $\frac{C^*}{2}$. Így, közvetlenül a negyedik munka ütemezése után van olyan gép, amelynek az aktuális terhelése legföljebb $\frac{C^*}{2}$. Ellenkező esetben a negyedik munka Y (vagy V). Ekkor viszont ez a munka ütemezhető arra a gépre, amelyik ekkor egyedül X-et (vagy U-t) végzi, az $ALG(3/2)$ szabálya, és amiatt, hogy $C > C^* \geq X + Y$ (vagy $C > C^* \geq U + V$). Vagyis most is kell lennie olyan gépnek, amelyikre egyedül a V (vagy Y) munka lett kiosztva még csak ebben a pillanatban, és így az aktuális terhelése legfeljebb $\frac{C^*}{2}$. Vagyis mindkét esetben teljesül, hogy $s \leq \frac{C^*}{2}$. Ebből következik, hogy $p_5 + s \leq \frac{3}{2}C^*$, ellentmondás. Ezzel a 2.20 Tétel bizonyításával készen vagyunk. □

2.5.2. Az $ALG2.3$ javított félig-online algoritmus $5/2 < r \leq 3$ esetére

Ebben a fejezetben bevezetjük az $ALG2.3$ algoritmust, melynek versenyképességi aránya $R_{ALG2.3}(r) = \frac{4r+2}{2r+3}$ tetszőleges $5/2 < r \leq 3$ esetén, míg a feladat alsó korlátja legalább $LB(r) = \frac{7r+4+\sqrt{r^2+8r+4}}{2r+2+2\sqrt{r^2+8r+4}}$.

Először lássuk az alsó korlátot. Tekintsük a következő sorozatot: $\mathcal{I}_3 = \{1, 1, 1, X, X, X, r\}$, ahol $1 \leq X \leq r$. Könnyen látható, hogy ha $\frac{2X+1}{X+1} \geq \frac{X+1+r}{2X+1}$ teljesül, akkor bármely félig-online algoritmus versenyképességi aránya a sorozatra legalább $\frac{X+1+r}{2X+1}$. (Ugyanis jöjjön először az első három munka. Ha ezek nem három különböző gépre kerülnek, a versenyképességi arány legalább 2. Tegyük fel tehát, hogy különböző gépre kerülnek. Most következzen a következő három munka. Optimális esetben egyenlően vannak szétosztva a munkák a három gép között, ha most nem így ütemezi őket az algoritmus, akkor a versenyképességi arány a feltevés szerint legalább $\frac{2X+1}{X+1} \geq \frac{X+1+r}{2X+1}$. Tegyük fel tehát, hogy ezek megint különböző gépre kerülnek. Az utolsó munka ütemezése után az algoritmus által kapott teljes átfutási idő $X + 1 + r$. Amennyiben még az is teljesül, hogy $X \leq 1.5$, akkor az optimális megoldás esetén r és X nem lehet egy gépen, ekkor az optimum értéke $2X + 1$.) Legyen $X = \frac{r-2+\sqrt{r^2+8r+4}}{6}$, ilyen választással az előbbi két X-re vonatkozó feltétel teljesül, továbbá ekkor $\frac{X+1+r}{2X+1} = \frac{7r+4+\sqrt{r^2+8r+4}}{2r+2+2\sqrt{r^2+8r+4}}$. Jelöljük az előbbi számot $LB(r)$-rel. Emiatt tetszőleges $5/2 \leq r \leq 3$ esetén, bármely félig-online algoritmus versenyképességi arányának értéke legalább $LB(r)$.

$ALG2.3$ **Algoritmus**: Válasszuk a γ paramétert a következőképpen: legyen $\gamma = \frac{4r+2}{2r+3}$, eztán futtassuk az $ALG(\gamma)$ algoritmust.

2.23. Megjegyzés Az $R_{ALG2.3}(r)$ és $LB(r)$ függvények monoton növekvőek az r függvényében, teljesülnek $R_{ALG2.3}(2.5) = 1.5$, $R_{ALG2.3}(3) = 14/9 \approx 1.5555$, $LB(2.5) = 1.5$ valamint $LB(3) \approx 1.5414$. Továbbá $\min\{R_{LS}(r) - R_{ALG2.3}(r) |\ 2.5 \leq r \leq 3\} = 1/10$, és $\max\{R_{ALG2.3}(r) - LB(r) |\ 2.5 \leq r \leq 3\} \approx 0.01417$, vagyis az $ALG2.3$ jelentősen jobb mint LS és majdnem optimális minden $2.5 < r \leq 3$ esetén.

Annak megmutatásához, hogy az $ALG2.3$ algoritmus versenyképességi aránya nem jobb mint $\frac{4r+2}{2r+3}$, elég a következő sorozatot tekintenünk: Legyen

$$\mathcal{I}_4 = \{1, 1, 1+\varepsilon, 1+\varepsilon, 1+2\varepsilon, 1+2\varepsilon, r\},$$

ahol $\varepsilon = \frac{2r-5}{8}$, és $2.5 < r \leq 3$. Ekkor $P_1 = P_2 = \{1, 1+2\varepsilon\}, P_3 = \{1+\varepsilon, 1+\varepsilon, r\}$, és $P_1^* = P_2^* = \{1, 1+\varepsilon, 1+2\varepsilon\}, P_3^* = \{r\}$. Emiatt $\frac{C_{ALG3}}{C^*} = \frac{4r+2}{2r+3}$. A $\mathcal{I}_4$ sorozat alapján látható, hogy a γ paraméter értékének csökkentése nem tud javítani az algoritmus versenyképességi arányán semmilyen $2.5 < r \leq 3$ esetén sem.

2.24. Tétel *Az $ALG2.3$ algoritmus versenyképességi aránya tetszőleges $\frac{5}{2} < r \leq 3$ esetén $\frac{4r+2}{2r+3}$.*

Bizonyítás: Annak belátásához, hogy az algoritmus versenyképességi aránya az előbbi arány, elég belátnunk azt, hogy $ALG2.3$ az összes munkát képes ütemezni. Indirekt bizonyítási módot választunk. Tegyük fel tehát, hogy létezik olyan $\mathcal{I}$ feladat sorozat, amelyre $ALG2.3$ nem tudja ütemezni az utolsó munkát, p_n-et. Legyen w az a teljes átfutási idő, amit úgy kapunk, hogy az első $n-1$ munkát az $ALG2.3$ algoritmus által ütemezzük, az utolsó munkát, vagyis a p_n munkát pedig az LS szabály szerint. Ekkor teljesül, hogy $w = s + p_n$. Mivel $ALG2.3$ nem képes ütemezni a p_n munkát, teljesül a $w > \frac{4r+2}{2r+3}C^* \geq \frac{3}{2}C^*$ feltétel. A 2.6 Lemma alapján azt kapjuk az utolsó munka hoszára, hogy $p_n > \frac{3}{4}C^*$. Ezt összevetve a $p_n \leq r \leq 3$ feltétellel, kapjuk hogy $C^* < 4$. Ezután a tétel bizonyítása a következő három lemma által adódik.

2.25. Lemma $|\mathcal{I}| \leq 7$, *továbbá* $s \leq \frac{2}{3}C^*$.

Bizonyítás: $C^* < 4$ miatt kapjuk, hogy mindegyik optimális gép legfeljebb három munkát végezhet, hiszen az egyes munkák hossza legalább 1. Továbbá a $p_n > \frac{3}{4}C^*$ feltétel miatt tudjuk azt is, hogy $p_n + 1 > C^*$, vagyis másképp szólva, a p_n munka egyedül szerepel a neki megfelelő optimális gépen. Ennek következtében teljesül, hogy $|\mathcal{I}| \leq 7$ továbbá

$$s \leq \frac{1}{3}\sum_{i=1}^{n-1} p_i \leq \frac{2}{3}C^*. \tag{1}$$

□

2.26. Lemma $|\mathcal{I}| \leq 5$ *esetén* $ALG2.3$ *minden munkát ütemez.*

Bizonyítás: A bizonyítás $|\mathcal{I}| \leq 4$ esetén a 2.20 Tételben szereplő módon történhet, ezért azt itt nem részletezzük. Tegyük fel tehát, hogy $|\mathcal{I}| = 5$. Tekintsük a p_5 munkának az $ALG2.3$ algoritmus által történő ütemezését. Az $ALG2.3$ szabálya miatt eddig az időpontig két gépre egy-egy munka, egy gépre pedig két munka lett kiosztva. Az általánosság megszorítása nélkül feltehető, hogy X, U, V, Y az első négy munka, ahol X és U lett ütemezve eddig az első, illetve második gépre, a V, Y munkák pedig a harmadik gépre kerültek, valamint a negyedik munka az Y.

Először megbecsüljük e négy munka méretét. Azt állítjuk, hogy $X > \frac{C^*}{2}$ és $U > \frac{C^*}{2}$ teljesül. Ellenkező esetben ugyanis a p_5 munka ütemezhető arra a gépre, amelyik X-et vagy arra, amelyik U- végzi ekkor, ezek hossza ugyanis legfeljebb $\frac{C^*}{2}$, és emiatt az adódó teljes átfutási idő nem több mint $w \leq \frac{3C^*}{2}$, ami ellentmondás, adódik tehát $X > \frac{C^*}{2}$ továbbá $U > \frac{C^*}{2}$. Abban az esetben, ha V (vagy Y) szintén több lenne mint $\frac{C^*}{2}$, figyelembe véve hogy a p_5 munka egyedül szerepel az egyik optimális gépen, az X, U, és V (vagy Y) munkákat nem tudjuk két optimális gépre ütemezni, hiszen mindegyik hossza nagyobb mint $\frac{C^*}{2}$, újra ellentmondás. Vagyis kapjuk ezen munkák hosszára, hogy $V \leq \frac{C^*}{2}$ és $Y \leq \frac{C^*}{2}$.

Ezek után tekintsünk egy optimális ütemezést. Tegyük fel, hogy van olyan optimális gép, amely legalább három munkát végez. Ekkor mivel $\frac{C^*}{2}+2 > C^*$ (hiszen $C^* < 4$), az erre a gépre ütemezett munkák mindegyikének hossza legfeljebb $\frac{C^*}{2}$, emiatt van három olyan munka X, U, V, Y között, amelyek mindegyikének hossza legfeljebb $\frac{C^*}{2}$, ez ellentmond az előbbi megállapításainknak. Vagyis minden optimális gépre legfeljebb két munka van ütemezve. Mivel p_5 egyedül szerepel az ő optimális gépén, a másik két optimális gépen két-két munka van. Következik, hogy Y optimális gépére van kiosztva rajta kívül még egy munka. Ez nem lehet V, mert akkor $X > \frac{C^*}{2}$ és $U > \frac{C^*}{2}$ egy optimális gépen lenne. Emiatt az Y optimális gépén levő másik munka X vagy U. Az általánosság megszorítása nélkül feltehető, hogy ez a munka X. Ekkor nyilvánvalóan teljesül a $X+Y < C^*$ feltétel.

Nyilván teljesülnek a $C^* \geq V+Y$ és $C^* \geq X$ feltételek, (az előbbi azért, mert e két munka a két legkisebb az első négy munka között). E két egyenlőtlenség következményeképpen

$$C^* \geq \frac{V+Y}{2} + \frac{X}{2}. \tag{2}$$

Mivel az $ALG2.3$ algoritmus Y-t nem azokra a gépekre ütemezi, amelyekre X vagy U került, továbbá $X, U \geq V$, az algoritmus szabályából következik, hogy

$$\frac{4r+2}{2r+3}C^* < X+Y. \tag{3}$$

Összevetve a (2) és (3) feltételeket, kapjuk, hogy $X+Y > (r+\frac{1}{2})V$. Ezt behelyettesítve az $X+Y < C^*$ feltételbe kapjuk hogy $C^* > (r+\frac{1}{2})V \geq r+\frac{1}{2}$. Eztán

ennek, és a (1) egyenlőtlenségnek a felhasználásával kapjuk, hogy

$$\frac{w}{C^*} = \frac{s+p_5}{C^*} \leq \frac{\frac{2}{3}C^*+r}{C^*} < \frac{2}{3} + \frac{r}{r+\frac{1}{2}} = \frac{10r+2}{6r+3} \leq \frac{4r+2}{2r+3},$$

ahol az utolsó egyenlőtlenség könnyen igazolhatóan teljesül $r \geq 5/2$ esetén. Ezzel a lemmát beláttuk. □

2.27. Lemma *Ha a sorozat elemszáma $|I| = 7$ vagy $|I| = 6$, akkor ALG*2.3 *szintén ütemezi az összes munkát.*

Bizonyítás: Elsőként belátjuk, hogy akár $|\mathcal{I}| = 7$, akár $|\mathcal{I}| = 6$ esetén az ALG2.3 algoritmus az utolsó előtti munkát, vagyis p_{n-1}-et minimális terhelésű gépre ütemezi, továbbá teljesül még a $C^* \geq p_{n-1} + 1$ feltétel. Tekintsük először a $|\mathcal{I}| = 7$ esetet. Az általánosság megszorítása nélkül feltehetjük, hogy $P_3^* = \{p_7\}$. Ekkor a P_1^* és P_2^* optimális gépek mindegyikére $3-3$ munka van ütemezve, és így $C^* \geq 3$. Mivel egyszerre $C^* < 4$ is teljesül, tudjuk hogy az első hat munka hossza kisebb 2-nél: $p_i < 2, i = 1, \cdots, 6$.

Tekintsük az ALG2.3 által kapott ütemezést az utolsó munka, p_7 ütemezése előtti pillanatban. Emlékezzünk vissza, hogy azelső három munkát az algoritmus három különböző gépre ütemezi. Ha van olyan gép amelyikre legalább három munka került, akkor olyan gép is van, amelyikre csak egy munka került, jelöljük ezt a munkát X-szel. Következik, hogy $s \leq X$. Nyilván teljesül, hogy $X \leq C^* - 2$ hiszen az X munka két másikkal együtt szerepel az ő optimális gépén. így kapjuk, hogy

$$w = s + p_7 \leq X + r \leq C^* - 2 + 3 \leq C^* + 1 < \frac{3}{2}C^*,$$

ahol az utolsó egyenlőtlenség a $C^* \geq 3$ feltétel miatt teljesül, ellentmondást kaptunk. Ezek szerint feltehetjük, hogy a p_7 munka ütemezése előtti pillanatban minden gépre éppen két munka került.

Most tekintsük az utolsó előtti munka, (vagyis p_6) ALG2.3 általi ütemezését. Feltehetjük (az általánosság megszorítása nélkül), hogy p_6 az első gépre, P_1-re került. Ekkor nyilván teljesül hogy $l(P_1) \geq 1 + p_6$. Bevezetjük a $L = \min\{L(P_2), L(P_3)\}$ jelölést, ekkor $L \geq 2$ hiszen a P_2 és P_3 gépekre ekkor már két-két munka lett kiosztva. Azonban ekkor amiatt csak egy munka lett ütemezve a P_1 gépre a p_6 munka előtt, továbbá $p_i < 2, i = 1, \cdots, 5$, így $L(P_1) < 2$. Vagyis más szóval $L(P_1) < L$, vagyis p_6 olyan gépre került, amelynek a terhelése ekkor minimális volt.

Másrészt, mivel bármely két munka együttes hossza legalább $2 > p_6$, ebből következik, hogy a $\{p_1, p_2, \cdots, p_6\}$ feladathalmaz bármely optimális ütemezése esetén a p_6 munka gépén másik munka is lesz. Ebből pedig következik, hogy $C^* \geq p_6 + 1$.

Most lássuk a $|\mathcal{I}| = 6$ esetet. Feltehetjük, hogy az első optimális gép, P_1^* három munkát végez, P_2^* két munkát, és a harmadik optimális gép, P_3^* pedig egyedül a p_6 munkát végzi. Nevezzük a munkákat a következőképpen: Legyen $P_1^* = \{A, B, C\}$ és $P_2^* = \{D, E\}$, ahol $D \leq E$. Mivel $C^* < 4$ teljesül, és minden munka mérete legalább 1, tudjuk hogy A, B és C mérete legfeljebb $C^* - 2 < \frac{C^*}{2}$. Triviálisan teljesül a $D \leq \frac{C^*}{2}$ feltétel is.

Tekintsük az $ALG2.3$ által létrejött ütemezést, a p_6 munka ütemezése előtti pillanatban. Ha van olyan gép, amelynek terhelése legfeljebb $\frac{C^*}{2}$, akkor $s \leq \frac{C^*}{2}$ és emiatt $w \leq \frac{C^*}{2} + p_6 \leq \frac{3}{2}C^*$, ami ellentmondás. Szintén feltehető az $E > \frac{C^*}{2}$ egyenlőtlenség (ellenkező esetben a p_n munka kivételével minden munka hossza legfeljebb $\frac{C^*}{2}$, és akkor lenne olyan gép amelyiknek a terhelése ebben a pillanatban legfeljebb $\frac{C^*}{2}$). Ugyanezen ok miatt teljesül, hogy van olyan gép, amelyikre ekkor egyedül az E munka lett kiosztva. Ekkor $E \neq p_5$ hiszen az $ALG2.3$ algoritmus az első három munkát különböző gépekre ütemezi, ennek következtében p_5 az $\{A, B, C, D\}$ halmaz egyik eleme, és így $p_5 \leq \frac{C^*}{2}$.

Most tekintsük az $\{A, B, C, D, E\}$ feladathalmaz egy optimális ütemezését. Tegyük fel, hogy p_5 minden optimális ütemezés esetén egyedül van az ő optimális gépén. Ekkor szükségképpen az E munka is, hiszen $E > \frac{C^*}{2} \geq p_5$. A többi három munka emiatt egy optimális gépre kerül. Ebből azonban ellentmondást kapunk, hiszen mivel bármely két munka együttes hossza legalább $2 > \frac{C^*}{2} \geq p_5$, nem rosszabb ütemezés adódik, ha az egyik munkát a három közül áttesszük arra a gépre, amelyik egyedül a p_5 munkát végzi. Ellentmondáshoz jutottunk, tehát feltehető, hogy arra az optimális gépre, amelyikre p_5 van ütemezve, kerül másik munka is. Következik ennek apalján, hogy $C^* \geq p_5 + 1$.

Tegyük fel, (az általánosság megszorítása nélkül), hogy $ALG2.3$ a p_5 munkát az első gépre, P_1-re ütemezi. Ekkor teljesül az $l(P_1) \geq 1 + p_5$ egyenlőtlenség. Továbbá, a másik két gép egyikére került az E munka, és a harmadik gépre további két munka lett kiosztva a p_5 munka ütemezése előtti pillanatban. Összevetve ekkor a gépek terhelését adódik, hogy a P_1 gép terhelése a legkisebb a három közül.

Ezzel beláttuk, hogy mindkét esetben ($|\mathcal{I}| = 7$ vagy $|\mathcal{I}| = 6$), teljesül hogy $C^* \geq p_{n-1} + 1$, és az is, hogy $ALG2.3$ az utolsó előtti munkát minimális terhelésű gépre ütemezi. Most folytatjuk a lemma bizonyítását. Mivel p_{n-1} nem a másik két gép valamelyikére, tehát nem is a P_2, és nem a P_3 gépre lett ütemezve, az algoritmus szabályából következik, hogy

$$\frac{4r+2}{2r+3}C^* < L + p_{n-1}, \tag{4}$$

ahol $L = \min\{L(P_2), L(P_3)\}$. Helyettesítsük be a $C^* \geq p_{n-1} + 1$ egyenlőtlenséget (4)-be, ekkor kapjuk hogy

$$L > \frac{2r-1}{2r+3}p_{n-1} + \frac{4r+2}{2r+3} \tag{5}$$

Legyen $p_{n-1} = \frac{2r-1}{4} + x$ valamely (nem feltétlenül nemnegatív) valós x-re. Ekkor teljesül

$$l(P_1) \geq 1 + p_{n-1} = \frac{2r+3}{4} + x, \tag{6}$$

továbbá (5) miatt,

$$L > \frac{2r-1}{2r+3}\left(\frac{2r-1}{4} + x\right) + \frac{4r+2}{2r+3} = \frac{2r+3}{4} + \frac{2r-1}{2r+3}x. \tag{7}$$

A következő két eset fordulhat elő:
1. eset $x \geq 0$. Ekkor felhasználva (6) és (7)-t, kapjuk, hogy $\sum_{i=1}^{n-1} p_i \geq 2L + l(P_1) > 3 \cdot \frac{2r+3}{4} = \frac{6r+9}{4}$. Továbbá mivel $P_3^* = \{p_n\}$, következik, hogy $C^* \geq \frac{\sum_{i=1}^{n-1} p_i}{2} > \frac{6r+9}{8}$. Ezt és (1)-t felhasználva adódik

$$\frac{w}{C^*} = \frac{s+p_n}{C^*} \leq \frac{\frac{2}{3}C^* + r}{C^*} = \frac{2}{3} + \frac{r}{C^*} < \frac{2}{3} + \frac{r}{\frac{6r+9}{8}} = \frac{2}{3} + \frac{8r}{6r+9} = \frac{4r+2}{2r+3},$$

ami ellentmondás.
2. eset $x < 0$. Ekkor a (6) és (7) feltételek miatt $C^* \geq \frac{1}{3}\sum_{i=1}^{3} l(P_i) \geq \frac{1}{3}\left(\frac{2r+3}{4} + x + 2L\right)$, és így

$$\frac{4r+2}{2r+3}C^* \geq \frac{4r+2}{2r+3} \cdot \frac{1}{3}\left(\frac{2r+3}{4} + x + 2L\right) = \frac{2r+1}{6} + \frac{4r+2}{6r+9}x + \frac{8r+4}{6r+9}L. \tag{8}$$

Felhasználva a (4), (8) és $p_{n-1} = \frac{2r-1}{4} + x$ feltételeket kapjuk hogy

$$L < \frac{2r+3}{4} + \frac{2r+7}{2r-5}x < \frac{2r+3}{4} + \frac{2r-1}{2r+3}x,$$

ahol az utolsó egyenlőtlenség amiatt teljesül, mert $\frac{2r+7}{2r-5} > \frac{2r-1}{2r+3}$ és $x < 0$. Ez pedig ellentmond a (7) feltételnek. A bizonyítás ezzel teljes. □

2.5.3. Az $ALG2.4$ félig-online algoritmus $3 < r < 6$ esetére

Ebben a fejezetben bemutatunk egy algoritmust, amelynek versenyképességi aránya minden rögzített $3 < r < 6$ esetén jobb, mint az LS algoritmusé. Látni fogjuk, hogy elég az első tíz munkát megfelelően ütemezni, az ezután érkező munkák ütemezését már elég egyszerűen az LS szabály szerint végezni. Az $ALG2.4$ algoritmus az első három munkát külön gépre ütemezi. A következő hét munka esetén pedig a következőképpen jár el: Az éppen érkező munkát olyan gépre ütemezi, amelynek az átfutási ideje ezáltal nagyobb lesz mint egy előre meghatározott alsó korlát, de egyidejűleg kisebb is, mint egy szintén előre meghatározott felső korlát. Ha van ilyen gép, akkor ezek közül minimális terhelésű gépre ütemezi a munkát, ellenkező esetben pedig egyszerűen az LS szabályt használjuk. Tetszőleges $3 < r < 6$ esetén legyen $\delta = \min\left\{\frac{6-r}{18}, \frac{3}{103}\right\} > 0$ és

$\Delta = \frac{5}{3} - \frac{\delta}{18} < \frac{5}{3}$. Azt fogjuk belátni, hogy az $ALG2.4$ algoritmus versenyképességi aránya nem nagyobb mint Δ. Használni fogjuk a következő jelölést, mint korábban is: Jelöljük C_k^*-gal az első k számú munka optimális ütemezése esetén adódó teljes átfutási időt. A P_j gép aktuális terhelését $l\left(P_j\right)$-vel jelöljük, ahol $j = 1, 2, 3$.

$ALG2.4$ **Algoritmus**:

1. Az első három munkát ütemezzük különböző gépekre. Legyen $k = 4$.
2. Legyen C_1 az aktuális terhelések maximuma plusz δ, valamint legyen $C_2 = \left(\frac{5}{3} - \delta\right) C_k^*$.
3. Legyen I azon gépek indexeinek halmaza, amelyeknek a terhelése a $(C_1 - p_k, C_2 - p_k)$ intervallumba eseik.
4. $I \neq \emptyset$ esetén legyen $i = \arg\min \left\{l\left(P_j\right), j \in I\right\}$, és ütemezzük a p_k munkát a P_i gépre.
5. $I = \emptyset$ esetén legyen P_i olyan gép, amelynek a terhelése minimális, és ütemezzük a p_k munkát a P_i gépre.
6. Legyen $k = k + 1$. Ha nem érkezik újabb munka, stop; ellenkező esetben $k \leq 10$ esetén menjünk vissza a 2. lépésre.
7. Ütemezzük a maradék munkákat az LS szabálynak megfelelően, és vége.

Jegyezzük meg, hogy az algoritmus lineáris futásidejű, mert a tizedik munkától kezdve az ütemezésre az LS szabályt használjuk.

2.28. Tétel *Az* $ALG2.4$ *algoritmus versenyképességi aránya tetszőleges* $3 < r < 6$ *esetén* $R_{ALG2.4} \leq \Delta = \frac{5}{3} - \frac{\delta}{18} < \frac{5}{3}$, *ahol* $\delta = \min\left\{\frac{6-r}{18}, \frac{3}{103}\right\} > 0$.

Bizonyítás: A tétel belátása egy kissé körülményes feladat, ezért szükségünk van némi előkészítésre. Újra indirekt módon bizonyítunk. Legyen megint I olyan minimális elemszámú sorozat, amelyre az állítás nem teljesül. Az általánosság megszorítása nélkül feltehető, hogy az $ALG2.4$ által elkészített ütemezés teljes átfutási ideje az utolsó munka, p_n által adódik. Vegyük észre, hogy abban az esetben, ha a p_n munkát az algoritmus 4. lépése által ütemezzük, akkor $\frac{C_{ALG2.4}}{C^*} \leq \frac{5}{3} - \delta < \Delta$ nyilvánvalóan teljesül. Emiatt az utolsó munkát az 5. vagy 7. lépés által ütemezzük, vagyis az LS szabály szerint. Legyen s a minimális terhelés értéke az utolsó munka, vagyis p_n ütemezése előtti pillanatban.

2.29. Lemma *Feltehető, hogy* $C^* \leq \frac{r+6}{2} < 6$, *az utolsó munka egyedül van valamely optimális gépen, a munkák száma legfeljebb tíz, valamint teljesül az* $\frac{2C^*-\delta}{3} \leq s \leq \frac{2}{3}C^* < 4$ *egyenlőtlenség. Feltehető továbbá az is, hogy közvetlenül*

az utolsó munka, p_n ütemezése előtti pillanatban a gépek terhelésére teljesül $s \leq l(P_i) \leq s + \delta$, és mindegyik optimális gép terhelése legalább $l(P_i^*) \geq C^* - \delta$.

Bizonyítás: A 2.6 Lemma alapján kapjuk, hogy

$$\frac{C_{ALG2.4}}{C^*} \leq 1 + \frac{2p_i}{3C^*}. \tag{9}$$

Tegyük fel, hogy $C^* > \frac{r+6}{2}$. Ekkor a (9) feltételt felhasználva kapjuk, hogy

$$\frac{C_{ALG2.4}}{C^*} \leq 1 + \frac{2p_i}{3C^*} \leq 1 + \frac{4r}{3(r+6)} \leq \Delta.$$

Emiatt a továbbiakban feltesszük, hogy $C^* < \frac{r+6}{2} < 6$. Most tegyük fel, hogy $p_n \leq C^* - \delta$, ekkor megint (9) alapján adódik, hogy

$$\frac{C_{ALG2.4}}{C^*} \leq 1 + \frac{2(C^* - \delta)}{3C^*} = \frac{5}{3} - \frac{2\delta}{3C^*} < \frac{5}{3} - \frac{\delta}{9} \leq \Delta.$$

Ezek szerint teljesül $p_n > C^* - \delta$, ami azt is jelenti, hogy hogy bármely optimális megoldás esetén valamelyik optimális gép csak egyedül a p_n munkát végzi, a maradék munkák összhossza ezek alapján legfeljebb $2C^*$. Nyilvánvalóan teljesül az $s \leq \frac{2}{3}C^* < 4$ egyenlőtlenség is, hiszen az utolsó munka kivételével a többi munkát két optimális gép végzi. Tegyük fel, hogy a munkák száma több mint tíz. Ez esetben van olyan gép, amelyikre már legalább 4 munka lett ütemezve a p_n ütemezése előtti pillanatban, emiatt e gép terhelése ekkor már legalább 4. Emiatt ilyenkor s-re az előbbinél erősebb alsó korlát adódik, nevezetesen $s \leq \frac{2C^*-4}{2} = C^* - 2$. Ennek következtében pedig

$$\frac{C_{ALG2.4}}{C^*} = \frac{s + p_n}{C^*} \leq \frac{C^* - 2 + p_n}{C^*} \leq 1 + \frac{p_n - 2}{p_n} \leq 1 + \frac{r-2}{r} \leq \Delta,$$

vagyis látjuk, hogy elég csak az $n \leq 10$ esettel foglalkoznunk. Amennyiben $s < \frac{2C^*-\delta}{3}$, kapjuk, hogy

$$\frac{C_{ALG2.4}}{C^*} = \frac{s + p_n}{C^*} \leq \frac{\frac{2C^*-\delta}{3} + p_n}{C^*} = \frac{5}{3} - \frac{\delta}{3C^*} < \frac{5}{3} - \frac{\delta}{18} \leq \Delta.$$

Ha van olyan P_i gép, amelyre az állítással ellentétben $l(P_i) > s + \delta$ teljesül, akkor, mivel a p_n munkát nem számolva, a többi munka összmérete legfeljebb $2C^*$, adódik, hogy $2C^* \geq \sum_{i=1}^{3} l(P_i) > 3s + \delta$, vagyis átrendezve $s < \frac{2C^*-\delta}{3}$ adódik, ami ellentmond annak, amit már előzőleg beláttunk. Kapjuk tehát, hogy $s \leq l(P_i) \leq s + \delta$. Most tegyük fel, hogy valamely optimális gépre $l(P_i^*) < C^* - \delta$ teljesül. Ez nem lehet az az optimális gép, amelyik egyedül a p_n munkát végzi, az előzőek miatt. Ez esetben újra, a p_n munkát nem számolva, a többi munka összmérete legfeljebb $2C^* - \delta$, és újra az $s < \frac{2C^*-\delta}{3}$ ellentmondás adódik. □

Folytatjuk a tétel bizonyítását. Tekintsük az $ALG2.4$ algoritmus által készített ütemezést az utolsó munka, p_n érkezte előtti időpontban. Ekkor minden gépre került már legalább egy munka, (hiszen $ALG2.4$ az első három munkát külön gépekre teszi), ezért minden gépen van olyan munka, amelyik utoljára került arra a gépre. Jelöljük X_i-vel $(i = 1, 2, 3)$ azt a három munkát, amelyek utolsóként kerültek valamely gépre a p_n munka érkezte előtti pillanatig. Feltesszük, hogy ezek a munkák ebben a sorrendben érkeztek. (Ennek következtében viszont nem tudjuk, hogy az X_i melyik gépre került, nem biztos, hogy az i-edikre). Jelöljük emiatt $P(X_i)$-vel mindhárom i-re azt a gépet, amelyikre lett ütemezve az X_i munka az $ALG2.4$ algoritmus által. ($P(X_i)$ tehát nem feltétlenül az i-edik gép.)

Belátjuk, hogy X_2 és X_3 ütemezése csak az LS szabály szerint történhetett. Valóban, az előző lemma feltevései szerint tudjuk, hogy az X_3 ütemezésekor a következők teljesülnek: $C_1 \geq l\left(P(X_1)\right)+\delta$, felhasználva X_1, X_2 és X_3 értelmezését, és mivel X_1 a sorrendben megelőzte az X_2 és X_3 munkákat. Másrészt a lemma feltevései szerint $l\left(P(X_2)\right) \leq s+\delta$ és $l\left(P(X_1)\right) \geq s$. Emiatt $l\left(P(X_2)\right) \leq s+\delta \leq l\left(P(X_1)\right)+\delta \leq C_1$. Ez azt jelenti, hogy az X_2 munkát nem ütemezhette az algoritmus a 4. lépésben, tehát az az LS szabály szerint történt. Hasonlóképpen látható, hogy X_3 ütemezése is az LS szabály szerint történt.

2.30. Lemma *A p_n munka ütemezése előtti pillanatban a $P(X_3)$ és $P(X_2)$ gépekre az X_3 és X_2 munkákon kívül pontosan egy-egy további munka lett ütemezve. Jelöljük ezeket a munkákat Z-vel illetve U-val. Ekkor teljesülnek a következő feltételek: (a) $\frac{2s}{3}-5\delta \leq X_3 \leq s-1+\delta$ és $Z \leq \frac{s}{3}+6\delta$; (b) $U \leq Z$ és $\frac{2s}{3}-6\delta \leq X_2 \leq s-1+\delta$; (c) $U \geq Z-2\delta$.*

Bizonyítás: Tekintsük az X_3 munka ütemezését. Rögtön X_3 ütemezése után mindegyik gép terhelése legalább s, emiatt az aktuális optimális teljes átfutási idő is legalább $C_k^* \geq s$. Emiatt az algoritmus 2. lépésében a C_2 konstans értékére a következőt kapjuk: $C_2 = \left(\frac{5}{3}-\delta\right) C_k^* \geq \left(\frac{5}{3}-\delta\right) s$. Az $s < 4$ feltétel miatt teljesül a $C_2 > \frac{5}{3}s-4\delta$ egyenlőtlenség. Másrészt, egyik gép terhelése sem több mint $s+\delta$. Mivel a $P(X_1)$ és $P(X_2)$ gépek terhelése legalább s, emiatt az X_3 munkát ezen gépek valamelyikére ütemezve a megnövelt terhelés legalább $s+1 > (s+\delta)+\delta > C_1$. Amennyiben $X_3 \leq \frac{5}{3}s-4\delta-(s+\delta) = \frac{2s}{3}-5\delta$ lenne, akkor az X_3 munkát az algoritmus 4. lépése szerint (vagyis a másik két gép valamelyikére) kellene ütemezni, hiszen így a megnövelt terhelés nem lenne több mint C_2, ellentmondást kapunk. Kapjuk tehát, hogy $X_3 \geq \frac{2s}{3}-5\delta$.

Tegyük fel, hogy $X_3 > s-1+\delta$. Ekkor ez egy egyedüli munka a $P\left(X_3\right)$ gépen (ellenkező esetben a $P\left(X_3\right)$ gép terhelése nagyobb lenne mint $(s-1+\delta)+1 = s+\delta$, ami ellentmond a 2.29 Lemmának). Továbbá, állítjuk, hogy X_2 szintén egyedüli munka a $P\left(X_2\right)$ gépen. Ellenkező esetben ugyanis az X_2 munka a nulla időpontra lenne ütemezve a $P\left(X_3\right)$ gépre, hiszen láttuk, hogy X_2-őt az LS szabály szerint ütemeztük, továbbá a $P\left(X_3\right)$ gép ekkor nem végez még másik munkát. Mivel az

X_2 és X_3 egyedüli munkák a nekik megfelelő gépeken, és X_1 ezek előtt érkezik, emiatt az X_1 munkát az algoritmus első lépése szerint ütemeztük, és emiatt szintén egyedüli munka. Kapjuk, hogy X_1, X_2 és X_3 mindegyikének legalább s a mérete. Másrészt, mivel p_n egyedüli munka az egyik gépen az optimális ütemezés esetén, az X_1, X_2, és X_3 munkákat két optimális gépre kell tenni, vagyis következik, hogy $C^* \geq 2s$, ez megint ellentmond a 2.29 Lemmának. Beláttuk tehát, hogy $X_3 \leq s - 1 + \delta$.

Mivel $X_3 < s - 1 + \delta$ valamint $l(P(X_3)) \geq s$, az X_3 munkán kívül legfeljebb egy további munka lehet a $P(X_3)$ gépre ütemezve. Továbbá, mivel $X_3 > \frac{2s}{3} - 5\delta$ és $l\left(P(X_3)\right) \leq s + \delta$, a p_n munka ütemezése előtti pillanatban a $P\left(X_3\right)$ gépen levő további munkák összmérete legfeljebb $s + \delta - X_3 < \frac{s}{3} + 6\delta \leq \frac{4}{3} + 6\delta < 2$ (mivel $\delta < \frac{1}{9}$), emiatt pontosan egy további munka, legyen Z, van erre a gépre ütemezve, és erre teljesül, hogy $Z \leq \frac{s}{3} + 6\delta$.

Az U létezését, és a (b) egyenlőtlenséget igazolandó, tegyük fel, hogy X_2 egyedüli munka a $P\left(X_2\right)$ gépen a p_n munka ütemezése előtti pillanatban. Két eset lehetséges: (i) Amennyiben X_2 megelőzi a Z munkát, akkor tekintsük a Z munka $ALG2.4$ algoritmus általi ütemezését. Az aktuális optimális teljes átfutási idő $C_k^* \geq X_2 \geq s$. Emiatt az előbbi gondolatmenet szerint $C_2 \geq \frac{5}{3}s - 4\delta$. Mivel $\mathcal{I}$-nek legalább négy eleme van p_n-en kívül, amelyeket két optimális gépre kell ütemezni, tudjuk hogy $C^* \geq 2$ és így $s > \frac{2C^* - \delta}{3} \geq \frac{4-\delta}{3}$ az előző Lemma miatt. Következik, hogy

$$\begin{aligned} \max\left\{l\left(P\left(X_1\right)\right), l\left(P\left(X_2\right)\right)\right\} + Z &\leq \\ (s+\delta) + \frac{s}{3} + 6\delta &= \frac{4}{3}s + 7\delta \leq \frac{5}{3}s - 4\delta \leq C_2 \end{aligned} \tag{10}$$

ahol felhasználtuk, hogy $s > \frac{4-\delta}{3}$ és $\delta < \frac{1}{25}$. Ugyanakkor, mivel $Z \geq 1 > 2\delta$, kapjuk, hogy

$$\begin{aligned} \max\{l(P(X_1)), l(P(X_2))\} + Z &\geq s + 2\delta \geq \\ \max\{l(P(X_1)), l(P(X_2)), l(P(X_3))\} + \delta &\geq C_1. \end{aligned} \tag{11}$$

Összevetve a (10) és (11) feltételeket, adódik, hogy a Z munkát a $P\left(X_2\right)$ vagy $P\left(X_1\right)$ gépek valamelyikére kellene ütemezni, ellentmondás. (ii) A másik esetben, ha Z megelőzi a sorrendben az X_2 munkát, az előzőhöz hasonlóan kapjuk, hogy $C_1 < L(P(X_3)) + X_2 = Z + X_2 < C_2$ teljesül az X_2 munka ütemezésekor, ami maga után vonja azt, hogy X_2 az algoritmus 4. lépésével ütemezendő, nem pedig az LS szabály által, ellentmondás. Következik tehát, hogy a $P(X_2)$ gépre szintén van másik munka is ütemezve, és így $X_2 \leq s - 1 + \delta$.

Továbbá, mivel X_2 az LS szabály által lett ütemezve, ebből következik, hogy Z megelőzte X_2-őt, (különben X_2 ütemezhető lenne a $P(X_3)$ gépre, amely ebben az esetben még ekkor üres lenne). Jegyezzük meg azt is, hogy $Z \leq \frac{s}{3} + 6\delta < 2$ miatt a Z munka mérete kisebb, mint bármely két munka együttes hossza. Mivel csak a

Z munka lett ütemezve a $P(X_3)$ gépre az X_2 érkezésekor, és X_2 ütemezése az LS szabály által történt, csak egy további munka szerepel a $P(X_2)$ gépen. Jelöljük ezt a munkát U-val. Újra, mivel X_2 megelőzi az X_3 munkát, és X_2 ütemezése az LS szabály által történik, következik, hogy $U \leq Z$. Az előző lemma miatt teljesülnek az $s \leq X_2 + U \leq s + \delta$ és $s \leq X_3 + Z \leq s + \delta$ feltételek. Ezekből kapjuk viszont, hogy $X_2 \geq X_3 - \delta$ aminek következtében pedig $X_2 \geq \frac{2s}{3} - 6\delta$.

Ezek után csak a (c) egyenlőtlenség bizonyítása maradt hátra. Tekintsük az X_2 munkának az algoritmus általi ütemezését. Tegyük fel indirekt módon, hogy $U < Z - 2\delta$. Ekkor az előző lemmából fakadóan

$$Z + X_2 > (U + 2\delta) + X_2 = l(P(X_2)) + 2\delta \geq s + 2\delta \geq C_1 \tag{12}$$

teljesül az $ALG2.4$ algoritmus 4. lépésénél, X_2 ütemezésekor. Másrészt pedig könnyen beláthatjuk a $Z + X_2 < C_2$ egyenlőtlenség teljesülését is: X_2 ütemezésekor $C_k^* \geq \frac{1}{3}\sum_{i=1}^{k} p_i \geq \frac{2s+1}{3}$ teljesül, (ahol $p_k = X_2$). Emiatt kapjuk hogy

$$C_2 = \left(\frac{5}{3} - \delta\right) C_k^* \geq \left(\frac{5}{3} - \delta\right)\left(\frac{2s+1}{3}\right) = \frac{10}{9}s + \frac{5}{9} - (\frac{2s+1}{3})\delta, \tag{13}$$

valamint

$$Z + X_2 \leq \frac{s}{3} + 6\delta + s - 1 + \delta = \frac{4}{3}s - 1 + 7\delta. \tag{14}$$

Összevetve (13) és (14)-t, kapjuk, hogy

$$C_2 - (Z + X_2) \geq \frac{14}{9} - \frac{2}{9}s - (\frac{2}{3}s + \frac{22}{3})\delta > \frac{6}{9} - 10\delta > 0 \tag{15}$$

hiszen $s < 4$ és $\delta < \frac{1}{15}$. (12) és (15) következtében pedig $I \neq \emptyset$, és így az X_2 munkát nem az LS szabály által, hanem az algoritmus 4. lépése szerint kellene ütemezni, ellentmondás. Ezzel a lemmát beláttuk. □

Ezek után visszatérünk a 2.28 Tétel bizonyításához. Mivel a $\mathcal{I}$ feladathalmaz legalább öt munkát tartalmaz (melyek X_1, X_2, X_3, U, Z) a p_n-en kívül, és ezeket két optimális gépre kell ütemezni, kapjuk, hogy $C^* \geq 3$. Tekintsük az X_2, X_3, U, Z munkák optimális ütemezését. Emlékezzünk vissza, hogy p_n egyedüli munka az ő saját optimális gépén, ezek után három eset fordulhat elő:

1. eset X_2 és X_3 ugyanazon optimális gépen szerepel. Ekkor

$$X_2 + X_3 \geq \frac{4}{3}s - 11\delta > \frac{4}{3}\left(\frac{2C^* - \delta}{3}\right) - 11\delta > \frac{8}{9}C^* - \frac{103}{9}\delta > C^* - 1,$$

ahol az első két egyenlőtlenség az előző két lemma következménye, a harmadik pedig azért teljesül, mert $C^* < 6$ és $\delta \leq \frac{3}{103}$. Következik, hogy nincs már több

munka ütemezve erre az optimális gépre, hiszen akkor a három munka összmérete túl sok lenne. Így ezen optimális gép terhelése

$$X_2 + X_3 \leq 2(s - 1 + \delta) \leq \frac{4}{3}C^* - 2 + 2\delta < C^* - \delta,$$

az első két egyenlőtlenség megint az előző két lemmából következik, az utolsó pedig $C^* < \frac{r+6}{2}$ és $\delta \leq \frac{6-r}{18}$ miatt teljesül. így viszont ellentmondásra jutottunk a 2.29 Lemmával, ezért ez az eset nem teljesülhet.

2. eset X_2 és X_3 két különböző optimális gépre van ütemezve, és e két optimális gép közül az egyiken szerepel még az U és Z munkák mindegyike is. Tegyük fel az általánosság megszorítása nélkül, hogy $X_2, U, Z \in P_1^*$ és $X_3 \in P_2^*$. Ekkor

$$X_2 + U + Z = l(P(X_2)) + Z \geq s + Z > \frac{2C^* - \delta}{3} + 1,$$

ahol az első két egyenlőtlenség a 2.29 Lemmából következik. Emiatt nem tartozhat ezeken kívül több munka a P_1^* optimális géphez, hiszen $\left(\frac{2C^*-\delta}{3} + 1\right) + 1 > C^*$ (mert $C^* < \frac{r+6}{2}$ és $\delta < \frac{6-r}{6}$). Másrészt, az 1. esethez hasonlóan kapjuk, hogy

$$\begin{aligned} l(P_1^*) &= X_2 + U + Z \\ &= l(P(X_2)) + Z \leq (s + \delta) + \left(\frac{s}{3} + 6\delta\right) \\ &= \frac{4}{3}s + 7\delta \leq \frac{8}{9}C^* + 7\delta < C^* - \delta, \end{aligned}$$

ahol az egyenlőtlenségek az előző lemmákból, valamint a $\delta < \frac{1}{24}$ és $C^* \geq 3$ feltételek teljesüléséből következnek, és ezzel újra ellentmondáshoz jutottunk.

3. eset Az X_2, X_3 munkák is és hasonlóan az U, Z munkák is különböző optimális gépekre vannak ütemezve. Más szóval, az X_2, X_3 munkák egyike a P_1^*, a másik pedig a P_2^* gépre került, és hasonlóan, az U, Z munkák egyike a P_1^*, a másik a P_2^* gépre került. Két alesetet különböztetünk meg, amelyekben megbecsüljük a P_1^* és P_2^* gépek tehelését.

(i) X_2 és U egyazon optimális gépeken vannak, mondjuk legyen ez a P_1^* gép, az X_3 és Z munkák kerültek ehát a P_2^* optimális gépre. Ekkor mivel $\{X_2, U\} \in P(X_2)$ és $\{X_3, Z\} \in P(X_3)$, tudjuk, hogy $s \leq X_2 + U \leq s + \delta$ és $s \leq X_3 + Z \leq s + \delta$ teljesül.

(ii) Az X_2 és Z munkák vannak a P_1^*, és az X_3 és U munkák vannak a P_2^* optimális gépre ütemezve. Ekkor felhasználva a $Z - 2\delta \leq U \leq Z$ egyenlőtlenségeket, valamint a 2.29 Lemma szerint teljesülő $s \leq X_2 + U \leq s + \delta$ és $s \leq X_3 + Z \leq s + \delta$ egyenlőtlenségeket is, tudjuk, hogy $s \leq X_2 + Z \leq s + 3\delta$ és $s - 2\delta \leq X_3 + U \leq s + \delta$.

Az előbbi becslések szerint, mindkét esetben igaz az, hogy az X_2, X_3, U, Z munkák az előbbi két optimális gép terheléseihez olyan értékekkel járulnak hozzá,

amelyek az $[s-2\delta, s+3\delta]$ intervallumban vannak. A 2.29 Lemma szerint teljesül hogy $C^* - \delta \leq l(P_i^*) \leq C^*$, emiatt a P_1^* illetve P_2^* optimális gépekre ütemezett összes további munkák együttes hossza legfeljebb $C^* - (s-2\delta) < C^* - \frac{2C^*-\delta}{3} + 2\delta = \frac{C^*}{3} + \frac{7}{3}\delta < 2$ (felhasználva hogy $\delta \leq \frac{6-r}{14}$); illetve legalább $C^* - \delta - (s+3\delta) \geq C^* - 4\delta - \frac{2}{3}C^* = \frac{C^*}{3} - 4\delta$. Ezek alapján pontosan egy további munka van ütemezve a P_1^* optimális gépre, jelöljük ezt A-val, és hasonlóképpen pontosan egy további munka van a P_2^* gépen, ezt pedig jelöljük B-vel. Következik, hogy $\mathcal{I} = \{A, B, U, Z, X_2, X_3, p_n\}$, továbbá az A és B munkák hossza a $(\frac{C^*}{3} - 4\delta, \frac{C^*}{3} + \frac{7}{3}\delta)$ intervallumban van.

Foglaljuk összze az eddigieket: A $\mathcal{I}$ feladathalmazban az első négy munka a következők: A, B, U és Z (nem feltétlenül ebben a sorrendben), valamint A és B mérete nagyobb mint az U és Z munkáké, hiszen $\frac{C^*}{3} - 4\delta > \frac{s}{3} + 6\delta$ (mert $s \leq \frac{2}{3}C^*$, $C^* \geq 3$, és $\delta < \frac{1}{30}$).

Az $ALG2.4$ algoritmus az A és B munkákat egy gépre ütemezi, mármint a $P(X_1)$ gépre, és e két munka egyike megegyezik az X_1 munkával. Az általánosság megszorítása nélkül feltehetjük, hogy például $B = X_1$. Tekintsük az X_1 munkának az $ALG2.4$ algoritmus által történő ütemezését. Ebben a pillanatban, minden gépre került már pontosan egy munka, a legnagyobb terhelés A. Emlékezzünk vissza, hogy a $B = X_1$ munka az A munka gépére kerül, A után, és X_1 ütemezése az X_2 és X_3 előtt történik. Felhasználva hogy $\delta < \frac{3}{22}$, kapjuk, hogy

$$\min\{U, Z\} + X_1 \geq 1 + \left(\frac{C^*}{3} - 4\delta\right) > \left(\frac{C^*}{3} + \frac{7}{3}\delta\right) + \delta \geq A + \delta = C_1.$$

Ez azt jelenti, hogy ha bármelyik gépre kerülne is X_1, a gép megnövelt terhelése nagyobb lesz így mint a C_1 konstans aktuális értéke. Jelen pillanatban nem tudjuk, hogy az algoritmusbeli I halmaz éppen üres-e vagy sem, erre azonban nincs is szükségünk a következők miatt: Tegyük fel, hogy $I \neq \emptyset$. Ekkor mivel A nagyobb mint U és Z, az I szükségképpen tartalmazza azt a gépet, amelyik az U, Z munkák közül a kisebbiket (vagyis U-t) végzi, emiatt X_1 erre a gépre kerül, és ellentmondást kapunk. Ha pedig $I = \emptyset$, megint azt kapjuk, hogy X_1-nek arra a gépre kell kerülnie, amelyik az U, Z munkák kisebbikét végzi, és újra ellentmondást kapunk. A 2.28 Tétel bizonyítása ezzel készen van. □

Feltételezhetően ha az $3 < r < 6$ eset vizsgálatát több esetre bontanánk, egyenként hatékonyabb algoritmus is konstruálható lenne, ez azonban most nem volt célunk, csupán annak megmutatása, hogy $3 < r < 6$ esetén LS nem optimális.

3. Hasonló gépek elutasításos modelljei

E fejezetben hasonló gépek online ütemezésének elutasításos modelljével (USR) foglalkozunk, pontosabban ennek megszakítás nélküli, és megszakításos változatával. Tudomásunk szerint az USR feladat megszakításos esetével korábban még nem foglalkoztak, mi itt megadunk a feladat kétgépes változatára egy tetszőleges gépsebességek mellett optimális algoritmust. A megszakítás nélküli esetben pedig egy korábbi algoritmusnak megadjuk egy javított változatát, amely bizonyos gépsebességek mellett optimális is egyben.

A fejezet tartalma megjelent a [14] cikkben. A fejezetbeli algoritmusok és azok versenyképességi analízise az értekezés szerzőjének munkája.

Itt csak azzal a speciális esettel foglalkozunk, amikor a gépek száma kettő. Adott tehát n munka: $\mathcal{J} = \{J_1, \cdots, J_n\}$, mindegyik J_i munka esetén **(ebben a fejezetben az értekezés többi fejezetétől eltérően)** a munka hossza t_i, a munka ütemezésének elutasítása esetén fizetendő büntetés összege pedig $p_i, i = 1, \cdots, n$. Az M_1 és M_2 gépek működési sebessége $s_1 = 1$, és $s_2 = s \geq 1$.

Először megengedjük a munkák megszakítását, a második modellben pedig nem. A cél mindkét esetben a teljes átfutási időnek és a kifizetett büntetések összegének minimalizálása.

Az e fejezetben tárgyaltak közvetlen előzménye He és Min [41] cikke, amely az online megszakitás nélküli USR feladattal foglalkozik. A cikk közli az $LSR(\alpha)$ online algoritmust. Az algoritmus versenyképességi aránya

$$\begin{cases} s + \alpha(1 + s - s^2), & ha\ 1 \leq s < \phi, \\ \frac{s+1}{s}, & ha\ s \geq \phi, \end{cases}$$

ahol ahol $s \geq \phi$ esetén $\alpha = 1/s$, $1 \leq s < \phi$ esetén pedig α egyenlő az

$$\frac{s+1}{s + x(s+1) - 1} = s + x(1 + s - s^2)$$

egyenlet pozitív gyökével. Az algoritmus minden $s \geq \phi$ sebesség esetén optimális, (ahol ϕ az aranymetszés aránya).

(i) Közlünk egy olyan algoritmust az online megszakításos esetre, amely minden $s \geq 1$ sebesség esetén optimális, az algoritmus versenyképességi aránya $\frac{s+\sqrt{s^2+4s}}{2s} = \frac{1}{2} + \sqrt{\frac{1}{4} + \frac{1}{s}}$.

(ii) Mivel ismerünk optimális algoritmust a megszakítás nélküli esetre $s \geq \phi$ esetén, csak az $1 \leq s < \phi$ esettel foglalkozunk. Bevezetjük az $LSRM(\alpha)$ algoritmust, amely az $LSR(\alpha)$ algoritmusnak egy módosítása, attól annyiban különbözik, hogy az α paramétert más módon választjuk meg. Az algoritmus versenyképességi aránya $\frac{1}{2s}\left(s^2 + \sqrt{s^4 - 4s^3 + 4s^2 + 4s}\right)$, ami minden $1 \leq s < \phi$ sebesség esetén jobb, mint az $LSR(\alpha)$ algoritmusé. Jegyezzük meg, hogy a [41] cikk alsó becslései triviálisak, hiszen azok az elutasítás nélküli esetben is

alsó korlátok. Emiatt új, nemtriviális alsó korlátokat is közlünk, amelyek által következik, hogy $LSRM(\alpha)$ algoritmusunk optimális minden $1.3852 \leq s < \phi$ esetén. A maximális távolság az alsó és felső korlát között 0.0534.

Emlékeztetünk arra, hogy az első gépnek, M_1-nek a sebessége 1, míg a második gép, M_2, sebessége $s \geq 1$. Vezessük be a következő jelöléseket:

$$A = \{J_i \in \mathcal{J} \,|\alpha t_i < p_i\}, \; R = \left\{ J_i \in \mathcal{J} \,\middle|\, \frac{t_i}{s+1} \leq p_i \leq \alpha t_i \right\},$$

$$S = \left\{ J_i \in \mathcal{J} \,\middle|\, p_i < \frac{t_i}{1+s} \right\}.$$

A-val jelöljük ezek szerint a drága munkákat, ezek esetén a legnagyobb a büntetés nagysága a méretükhöz képest, ezek elutasítása tehát drága lenne, ezeket mindenképpen elfogadjuk, vagyis "akceptáljuk". Az S halmazban (s=small) olyan munkák vannak amelyeknek a fajlagos büntetése kicsi, ezeket elutasítjuk. Az R-rel jelöltek fajlagos büntetése kicsivel nagyobb, de ezeket is elutasítjuk (r= reject). (Az optimális megoldás esetén persze ettől különbözhet, hogy mely munkákat kell elfogadni, illetve elutasítani.)

Használjuk még a $T(\mathcal{J}') = \sum_{j_i \in \mathcal{J}'} t_i$ és $P(\mathcal{J}') = \sum_{j_i \in \mathcal{J}'} p_i$ jelöléseket, ahol $\mathcal{J}' \subset \mathcal{J}$ tetszőleges részhalmaz. Valamely $\mathcal{J}$ sorozathoz tartozó optimumérték jelölésére a $w^*(\mathcal{J})$ jelölést használjuk, (ami tehát a teljes átfutási idő plusz az elutasított munkákért fizetett büntetés összege), egy A algoritmus alkalmazásával kapott célfüggvényértékre pedig a $w_A(\mathcal{J})$ jelölést.

3.1. A megszakításos eset

A megszakításos online algoritmus esetén a soron következő munka kezelése két lépésből áll. Az első lépésben egy *elutasítási stratégiát* alkalmazunk, vagyis eldöntjük hogy a feladatot elutasítjuk-e vagy ütemezzük. Amennyiben a munkát elfogadtuk, második lépésként a munkát egy *ütemezési algoritmussal* ütemezzük. Jegyezzük meg, hogy az elutasítás nélküli online megszakításos esetre ismert optimális algoritmus két hasonló gép esetére, természetesen adódik az ötlet, hogy ezt az optimális ütemezési algoritmust kombináljuk valamely megfelelő elutasítási stratégiával. Mégsem ezt fogjuk tenni, ugyanis egy sokkal egyszerűbb ütemezési algoritmust alkalmazva is optimális algoritmust kapunk az USR feladatra.

Jelöljük L_i-vel az M_i gép pillanatnyi átfutási idejét. ahol $i = 1, 2$. Eztán az algoritmus formális leírása a következő:

$PMPT(\alpha)$ **algoritmus**

Az A halmaz elemeit elfogadjuk, a többit elutasítjuk. Az elfogadott munkákat a következő szabályok alkalmazásával ütemezzük, (a munkák, illetve azok egyes részei között nem hagyunk várakozási időket):

1. Az elsőként elfogadott munkát teljes egészében a második gépre ütemezzük.

2. Ha az éppen elfogadott munka méretére teljesül $t_i \leq L_2 - L_1$, akkor teljes egészében az első gépre ütemezzük.

3. Ha az éppen elfogadott munka méretére $t_i > L_2 - L_1$ teljesül, akkor annak $L_2 - L_1$ méretű darabját az első gépre, a megmaradó részt a második gépre ütemezzük.

Az α paramétert a következőképpen választjuk: Legyen $\alpha = -\frac{1}{2} + \sqrt{\frac{1}{4} + \frac{1}{s}}$ tetszőleges $s \geq 1$ esetén, ami az egyetlen pozitív gyöke az

$$1 + x = \frac{1}{xs}. \tag{16}$$

egyenletnek. Könnyen látható, hogy ekkor tetszőleges $s \geq 1$ esetén teljesül a következő egyenlőtlenség-lánc: $\frac{1}{s+1} < \alpha < \frac{1}{s}$. Az utóbbi egyenlőtlenségből következik, hogy $\alpha < \frac{1+\alpha}{s+1}$. Nyilvánvaló, hogy a $PMPT(\alpha)$ algoritmus által készített ütemezés megengedett abban az értelemben, hogy nincs átlapolás semelyik munka egyes részei között.

3.1. Tétel *A $PMPT(\alpha)$ algoritmus versenyképességi aránya $1+\alpha = \frac{s+\sqrt{s^2+4s}}{2s}$, és optimális tetszőleges $s \geq 1$ esetén.*

Bizonyítás: Az állítást indirekt módon bizonyítjuk, tegyük fel hogy a tétel állítása nem teljesül, ekkor léteznie kell egy (elemszám tekintetében) minimális $\mathcal{J}$ ellenpéldának. Ezután ezzel az ellenpéldával kapcsolatban fogalmazunk meg néhány állítást.

3.2. Lemma *A minimális ellenpélda optimális megoldása esetén minden S-beli munka elfogadásra kerül.*

Bizonyítás: Emlékezzünk vissza, a $PMPT(\alpha)$ algoritmus elfogadja az A beli munkákat, az R-, és S-belieket pedig elutasítja. Tegyük fel, hogy valamely $J_i \in S$ munka elutasítva van optimális megoldás esetén. Töröljük ezt a munkát törölve a sorozatból. A munka elhagyása nem befolyásolja azt, hogy a heurisztikus algoritmus esetén mi történik a többi munkával, emiatt a heurisztikus megoldás értéke pontosan a $P(J_i)$ értékkel csökken, ugyanígy az optimális megoldás értéke is. Ez viszont ellentmond annak hogy, az ellenpéldánk minimális. □

3.3. Lemma $w^*(\mathcal{J}) \geq \frac{T(A \cup R \cup S)}{s+1}$.

Bizonyítás: Legyen a feladat valamely optimális megoldása esetén S' az $A \cup R$-beli elfogadott munkák halmaza. Ekkor $A \cup R \setminus S'$ az elutasított munkák halmaza. Ekkor az előző lemmát is felhasználva kapjuk hogy

$$w^*(\mathcal{J}) \geq \frac{T(S)}{1+s} + L \doteq \frac{T(S)}{1+s} + \frac{T(S')}{1+s} + P(A \cup R \setminus S').$$

Vegyük észre, hogy a J_i munka az L alsó becslés értékéhez $\frac{t_i}{1+s}$-vel járul hozzá elfogadása esetén, elutasítása esetén pedig p_i-vel, emiatt kapjuk hogy

$$\begin{aligned} w^*(\mathcal{J}) &\geq \frac{T(S)}{1+s} + L \geq \frac{T(S)}{1+s} + \sum_{J_i \in A \cup R} \min\left\{\frac{t_i}{1+s}, p_i\right\} \\ &= \frac{T(S)}{1+s} + \sum_{J_i \in A \cup R} \frac{t_i}{1+s} = \frac{T(A \cup R \cup S)}{s+1}. \end{aligned}$$

□

3.4. Lemma *Tegyük fel, hogy az $X \in A$ feladatra teljesül az $\frac{T(X)}{s} > T(A \cup R \cup S \setminus \{X\})$ egyenlőtlenség, és az X feladat elfogadásra kerül valamely optimális megoldás esetén. Ekkor az optimum értéke $\frac{T(X)}{s}$.*

Bizonyítás: Optimális megoldás esetén minden munkát elfogadunk, az X munkát a második gépre ütemezzük, az összes többi munkát pedig az első gépre, amiből az állítás nyilvánvalóan következik. □

Ezek után lássuk a versenyképességi arány alsó becslését.

3.5. Tétel *Tetszőleges online algoritmus versenyképességi aránya az online megszakításos USR feladat esetén legalább $1 + \alpha = \frac{1}{2} + \sqrt{\frac{1}{4} + \frac{1}{s}}$, tetszőleges $s \geq 1$ esetén.*

Bizonyítás: Legyen az első munka J_1 ahol $t_1 = s$ és $p_1 = \alpha s$. Ha egy H algoritmus elfogadja a munkát, akkor nincs több munka, és $\frac{w_H(\{J_1\})}{w^*(\{J_1\})} \geq \frac{1}{p_1} = \frac{1}{\alpha s} = 1 + \alpha$ (a (16) feltétel miatt). Ha az algoritmus elutasítja a munkát, legyen a második munka J_2 ahol $t_2 = s^2$ és $p_2 = \infty$. Ekkor nincs több munka, és $w^*(\{J_1, J_2\}) = s$ amíg $w_H(\{J_1, J_2\}) \geq s + p_1$. Következik tehát hogy tetszőleges H algoritmus esetén $\frac{w_H(\{J_1,J_2\})}{w^*(\{J_1,J_2\}} \geq 1 + \alpha$. □

Most már be tudjuk fejezni a 3.1 Tétel bizonyítását, vagyis annak bizonyítását, hogy a $PMPT(\alpha)$ algoritmus versenyképességi aránya $1 + \alpha = \frac{s+\sqrt{s^2+4s}}{2s}$. (Ennek következtében optimális algoritmus tetszőleges $s \geq 1$ esetén.)

Az állítást tehát indirekt módon bizonyítjuk, feltettük, hogy $\mathcal{J}$ minimális ellenpélda. A $PMPT(\alpha)$ algoritmus elutasítja az $R \cup S$ halmazbeli munkákat, és csak az A-belieket fogadja el. Abban az esetben, ha $A = \emptyset$, vagyis minden munkát elutasítunk, akkor teljesül hogy $w_{PMPT(\alpha)}(\mathcal{J}) = P(R) + P(S)$. Másrészt a 3.3 Lemma miatt, tudjuk, hogy

$$w^*(\mathcal{J}) \geq \frac{T(A \cup R \cup S)}{1+s} = \frac{T(R \cup S)}{1+s} \geq \frac{P(R) + P(S)}{\alpha(1+s)}.$$

Emiatt

$$\frac{w_{PMPT(\alpha)}(\mathcal{J})}{w^*(\mathcal{J})} \leq \frac{P(R) + P(S)}{\frac{P(R)+P(S)}{\alpha(1+s)}} \leq \alpha(1+s) < 1 + \alpha.$$

Emiatt a következőkben feltesszük, hogy $A \neq \emptyset$.

Az ütemezési algoritmus szabályából következik, hogy $L_1 \leq L_2$ bármely pillanatban teljesül, és az is, hogy a gépek átfutási idejének a különbsége nem nagyobb, mint az utolsóként elfogadott munka hossza. Jelöljük Y-nal az utolsóként elfogadott munkát, valamint X-szel a leghosszabb elfogadott munkát. Ekkor a $PMPT(\alpha)$ algoritmus által készített ütemezés teljes átfutási ideje legfeljebb

$$\frac{T(A \setminus \{Y\})}{s+1} + \frac{T(Y)}{s} \leq \frac{T(A \setminus \{X\})}{s+1} + \frac{T(X)}{s}.$$

Kapjuk, hogy

$$w_{PMPT(\alpha)}(\mathcal{J}) \leq \frac{T(A \setminus \{X\})}{s+1} + \frac{T(X)}{s} + P(R) + P(S). \tag{17}$$

Abban az esetben, ha az X munka optimális megoldás esetén elutasításra kerül, a 3.3 Lemma bizonyításához hasonló módon kapható a következő alsó korlát:

$$w^*(\mathcal{J}) \geq P(X) + \frac{T(A \setminus \{X\}) + T(R) + T(S)}{s+1}. \tag{18}$$

Összevetve (17) és (18)-t, kapjuk, hogy

$$\begin{aligned}
w_{PMPT(\alpha)}(\mathcal{J}) &\leq \frac{T(A \setminus \{X\})}{s+1} + \frac{T(X)}{s} + P(R) + P(S) \\
&\leq \frac{T(A \setminus \{X\})}{s+1} + \frac{P(X)}{\alpha s} + \alpha(T(R) + T(S)) \\
&\leq \frac{T(A \setminus \{X\})}{s+1} + (1+\alpha)\left[P(X) + \frac{T(R) + T(S)}{s+1}\right] \\
&\leq (1+\alpha)\left[\frac{T(A \setminus \{X\})}{s+1} + P(X) + \frac{T(R) + T(S)}{s+1}\right] \\
&\leq (1+\alpha) w^*(\mathcal{J}).
\end{aligned}$$

ahol a harmadik sorban felhasználtuk, hogy $\frac{1}{\alpha s} = 1+\alpha$, és $\alpha < \frac{1+\alpha}{s+1}$. Emiatt feltehetjük a továbbiakban, hogy az X munka optimális megoldás esetén elfogadásra kerül. A következő két esetet megvizsgálva jutunk a kívánt ellentmondáshoz:

1. eset $\frac{T(X)}{s} \leq T(A \cup R \cup S \setminus \{X\})$. (17) miatt tuduk, hogy

$$w_{PMPT(\alpha)}(\mathcal{J}) \leq \frac{T(A \setminus \{X\})}{s+1} + \frac{T(X)}{s} + \alpha T(R) + P(S).$$

Ezt összevetve a 3.3 Lemmával, elég megmutatnunk azt, hogy

$$\frac{T(A \setminus \{X\})}{s+1} + \frac{T(X)}{s} + \alpha T(R) + \alpha T(S) \leq (1+\alpha)\frac{T(A) + T(R) + T(S)}{s+1}. \tag{19}$$

Legyen ennek érdekében $\frac{T(X)}{s} = \lambda\left(T(A \setminus \{X\}) + T(R) + T(S)\right)$ valamely $0 \leq \lambda \leq 1$ esetén. Ezt, valamint a $T(A) = T(A \setminus \{X\}) + T(X)$ egyenlőséget (19)-be helyettesítve kapjuk:

$$\begin{aligned} & \left(\frac{1}{s+1} + \lambda\right) T(A \setminus \{X\}) + (\lambda + \alpha)\left(T(R) + T\left(S\right)\right) \\ \leq \; & \frac{1+\alpha}{s+1}\left(1 + \lambda s\right)\left(T(A \setminus \{X\}) + T(R) + T\left(S\right)\right). \end{aligned}$$

Emiatt elég megmutatnunk a következő két egyenlőtlenség teljesülését:

$$\begin{aligned} \frac{1}{s+1} + \lambda &\leq \frac{(1+\alpha)\left(1+\lambda s\right)}{s+1}, \\ \lambda + \alpha &\leq \frac{(1+\alpha)\left(1+\lambda s\right)}{s+1}. \end{aligned}$$

Mivel $\frac{1}{s+1} < \alpha \leq \frac{1}{s}$ és $\lambda \leq 1$, mindkét egyenlőtlenség könnyen belátható. Emiatt kapjuk, hogy $w_{PMPT(\alpha)}(\mathcal{J}) \leq (1+\alpha)w^*(\mathcal{J})$.

2. eset $\frac{T(X)}{s} > T(A \cup R \cup S \setminus \{X\})$. Mivel az optimális megoldás esetén X elfogadásra kerül, a 3.4 Lemmából kapjuk, hogy az optimum értéke $\frac{T(X)}{s}$. Ezt összevetve (17)-el, az $\frac{1}{s+1} < \alpha$ feltételt, és a 2. esetet definiáló feltételt is felhasználva, kapjuk:

$$\begin{aligned} w_{PMPT(\alpha)}(\mathcal{J}) &\leq \frac{T\left(A \setminus \{X\}\right)}{s+1} + \frac{T\left(X\right)}{s} + P\left(R\right) + P\left(S\right) \\ &\leq \frac{T\left(A \setminus \{X\}\right)}{s+1} + \frac{T\left(X\right)}{s} + \alpha\left(T\left(R\right) + T\left(S\right)\right) \\ &\leq \alpha\left(T\left(A \setminus \{X\}\right) + T(R) + T\left(S\right)\right) + \frac{T\left(X\right)}{s} \\ &\leq (1+\alpha)\frac{T\left(X\right)}{s} = (1+\alpha)w^*(\mathcal{J}). \end{aligned}$$

Ezáltal beláttuk, hogy $w_{PMPT(\alpha)}(\mathcal{J}) \leq (1+\alpha)w^*(\mathcal{J})$ minden lehetséges esetben teljesül. A 3.5 Tétellel összevetve ezt kapjuk hogy a $PMPT(\alpha)$ algoritmus minden $s \geq 1$ esetben optimális. □

3.2. A megszakítás nélküli eset

Mivel a megszakítás nélküli esetben az $LSR(\alpha)$ algoritmus optimális minden $s \geq \phi$ esetén, csak az $1 \leq s < \phi$ esettel foglalkozunk. Az $LSRM(\alpha)$ algoritmus is úgy működik, hogy pontosan akkor utasít el egy J_i feladatot, ha a fajlagos büntetése, vagyis a p_i/t_i hányados, kisebb mint az előre rögzített α konstans, az összes többi (vagyis elfogadott) tárgyakat pedig az LS algoritmus megfelelő

változatával ütemezzük: a munka arra a gépre kerül, amelyik gép így a lehető legkorábbi időpontban fejezi be ezt a munkát. Az $LSRM(\alpha)$ algoritmus a korábbi $LSR(\alpha)$ algoritmustól így csak az α konstans megválasztásában különbözik.

Az $LSRM(\alpha)$ algoritmus (List Scheduling with Rejection, Modified (α))

A $J_i = (t_i, p_i)$ munkát elutasítjuk, ha $p_i \leq \alpha t_i$; ellenkező esetben az LS algoritmussal ütemezzük a munkát valamely gépre.

Az α konstanst a következő módon választjuk: Legyen tetszőleges $1 \leq s < \phi$ esetén $x_0 = \frac{1}{2s}\left(-s^2 + \sqrt{s^4 - 4s^3 + 4s^2 + 4s}\right)$, amely szám az egyedüli pozitív gyöke a következő egyenletnek

$$\frac{1 + s - s^2}{xs} = s + x. \tag{20}$$

Legyen továbbá

$$\beta = \frac{x_0\,(s+1)}{1 + s - s^2} = \frac{(s+1)}{s\,(s + x_0)} \quad \text{és} \quad \alpha = \frac{\beta}{1+s}.$$

Bevezetjük a $c(s) = s + x_0$ jelölést. Könnyen látható, hogy $c\,(s) > \frac{2s+1}{s+1}$ teljesül. Valóban, az (20) egyenlőtlenség mindkét oldala folytonos függvénye x-nek, továbbá a bal oldala szigorúan monoton csökkenő, míg a jobb oldal szigorúan monoton növekvő az x változóban. Nyilván pontosan egy pozitív megoldása van az egyenletnek. Helyettesítsük be az $x_1 = \frac{1+s-s^2}{s+1}$ számot x helyére, ekkor a bal oldal értéke $\frac{s+1}{s}$, a jobb oldalé pedig $\frac{2s+1}{s+1}$. Mivel $\frac{s+1}{s} > \frac{2s+1}{s+1}$ teljesül minden $1 \leq s < \phi$ esetén, következik hogy az egyenlőtlenség $x > x_1$ esetén teljesül. Újra felhasználva hogy a jobb oldal növekvő, rögtön kapjuk, hogy $c\,(s) > \frac{2s+1}{s+1}$. Felhasználjuk még a későbbiekben az $1 < \beta < c\,(s)$ egyenlőtlenségek teljesülését is, ezek az előzőhöz hasonlóan bizonyíthatóak. Továbbá most is teljesül, (mint a preemptív esetben) hogy $\frac{1}{s+1} < \alpha < \frac{1}{s}$.

Jegyezzük meg még, hogy az $LSRM(\alpha)$ algoritmus is (mint a megszakításos eset optimális $PMPT(\alpha)$ algoritmusa is) elutasít minden $S \cup R$ halmazbeli munkát, az A halmaz elemeit pedig az LS szabállyal ütemezi, emiatt a 3.2 és 3.3 Lemmák a megszakításos esetre is érvényesek.

3.6. Lemma *([41]) Tegyük fel, hogy az $LSRM(\alpha)$ algoritmus elvégzése után a teljes átfutás idő az $X \in A$ munka befejezésének időpontja által adódik. Ekkor a teljes átfutási idő legfeljebb*

$$\min\left\{\frac{T(A) + T(X)}{1+s}, \frac{T(A)}{s}\right\}$$

□

Ezek után rátérünk a versenyképességi arány bizonyítására.

3.7. Tétel *Tetszőleges $1 \leq s < \phi$ sebesség és tetszőleges J feladat sorozat esetén teljesül az alábbi egyenlőtlenség:*

$$\frac{w_{LSRM(\alpha)}(\mathcal{J})}{w^*(\mathcal{J})} \leq c(s) = s + x_0 = \frac{1}{2s}\left(s^2 + \sqrt{s^4 - 4s^3 + 4s^2 + 4s}\right).$$

Bizonyítás: Az állítást most is indirekt módon bizonyítjuk. Tegyük fel, hogy van olyan ellenpélda, amelyre az állítás nem teljesül, és tekintsünk egy $\mathcal{J}$ ellenpéldát, amely minimális számú munkából áll. Ha $A = \emptyset$, vagyis minden munkát alutasítunk, akkor $w_{LSRM(\alpha)} = P(R) + P(S)$. Másrészt a 3.3 Lemma miatt tudjuk, hogy

$$w^*(\mathcal{J}) \geq \frac{T(A \cup R \cup S)}{1+s} = \frac{T(R \cup S)}{1+s} \geq \frac{P(R) + P(S)}{\alpha(1+s)} = \frac{P(R) + P(S)}{\beta}.$$

Ezért kapjuk, hogy

$$\frac{w_{LSRM(\alpha)}(\mathcal{J})}{w^*(\mathcal{J})} \leq \frac{P(R) + P(S)}{\frac{P(R)+P(S)}{\beta}} \leq \beta < c(s).$$

A következőkben tehát feltehetjük, hogy $A \neq \emptyset$. Legyen $X \in A$ az a munka, amelyik meghatározza a teljes átfutási időt. (A teljes átfutási idő ennek a munkának a befejezési ideje.) Ekkor feltehetjük, hogy X az utolsóként érkező munka, és az $LSRM(\alpha)$ algoritmus elfogadja. Ha ugyanis érkezne még később, az $LSRM(\alpha)$ által elfogadott munka, mivel ezek a teljes átfutási időt nem befolyásolják, elhagyhatóak, kisebb elemszámú ellenpéldát kapunk, ami ellentmondás. Továbbá ha érkezik az X után még olyan munka, amelyeket az algoritmus elutasít, egy másik ellenpéldát készíthetünk úgy, hogy ezeket a munkákat az X érkezése elé tesszük. A feladathalmaz elemszáma nem változik, és nem változik sem az optimális, sem az $LSRM(\alpha)$ algoritmus által készített megoldás. (Ugyanazokat a munkákat utasítja el, és amelyeket pedig elfogadja, azokaz ugyanúgy ütemezi.) Tehát feltehetjük, hogy X az utolsó munka. Ezek után a bizonyítás úgy történik, hogy négy, minden esete kimerítő esetet vizsgálunk meg.

1 eset. Van olyan optimális megoldása a feladatnak, amikor az X munkát elutasítjuk. A $\mathcal{J}$ ellenpélda minimalitása miatt tudjuk, hogy $\frac{w_{LSRM(\alpha)}(\mathcal{J}\setminus\{X\})}{w^*(\mathcal{J}\setminus\{X\})} \leq c(s)$ teljesül. Másrészt az X munka a teljes átfutási időt legfeljebb $\frac{T(X)}{s}$-sel növelheti, az optimális megoldás értékét pedig pontosan $P(X)$-szel növeli (A $\mathcal{J} \setminus \{X\}$ feladathalmazra kapott heurisztikus illetve optimális megoldás értékekhez képest). Mivel $P(X) > \alpha T(X)$ teljesül, hiszen $X \in A$, valamint

$$\frac{T(X)/s}{\alpha T(X)} = \frac{1}{\alpha s} = \frac{1+s}{\beta s} = s + x_0 = c(s),$$

következik, hogy

$$\frac{w_{LSRM(\alpha)}(\mathcal{J})}{w^*(\mathcal{J})} \leq \frac{w(\mathcal{J}\setminus\{X\})+\frac{T(X)}{s}}{w^*(\mathcal{J}\setminus X)+P(X)} \leq \frac{w(\mathcal{J}\setminus\{X\})+\frac{T(X)}{s}}{w^*(\mathcal{J}\setminus\{X\})+\alpha T(X)} < s+x_0.$$

2 eset. Az optimális megoldás esetén az X munkát elfogadjuk, azonkívül teljesül a következő két feltétel: $T(X) \leq \frac{s}{s+1}T(A\cup R\cup S)$, és $T(R\cup S) \leq \frac{s+1-s^2}{s+1}T(A\cup R\cup S)$.

Alkalmazva a 3.6 Lemmát, kapjuk hogy

$$\begin{aligned}
w_{LSRM(\alpha)}(\mathcal{J}) &\leq \frac{T(A)+T(X)}{s+1}+P(R)+P(S)\\
&\leq \frac{T(A)}{s+1}+\frac{T(X)}{s+1}+\beta\frac{T(R)+T(S)}{s+1}\\
&\leq \frac{T(A)+T(R)+T(S)}{s+1}+\frac{s}{s+1}\frac{T(A\cup R\cup S)}{s+1}\\
&\qquad\qquad +(\beta-1)\frac{T(R)+T(S)}{s+1}\\
&\leq \left(1+\frac{s}{s+1}\right)\frac{T(A\cup R\cup S)}{s+1}+(\beta-1)\frac{s+1-s^2}{s+1}\frac{T(A\cup R\cup S)}{s+1}\\
&= \left(\frac{2s+1}{s+1}+(\beta-1)\frac{s+1-s^2}{s+1}\right)\frac{T(A\cup R\cup S)}{s+1}.
\end{aligned}$$

Mivel $\frac{2s+1}{s+1}+(\beta-1)\frac{s+1-s^2}{s+1} = s+\beta\frac{s+1-s^2}{s+1} = s+x_0 = c(s)$, felhasználva még a 3.3 Lemmát is, kapjuk:

$$w_{LSRM(\alpha)}(\mathcal{J}) \leq (s+x_0)\frac{T(A\cup R\cup S)}{1+s} \leq c(s)\,w^*.$$

3 eset. Az optimális megoldás esetén az X munkát elfogadjuk, továbbá $T(X) \leq \frac{s}{s+1}T(A\cup R\cup S)$, és $T(R\cup S) > \frac{s+1-s^2}{s+1}T(A\cup R\cup S)$. Az utóbbi feltételből kapjuk, hogy

$$T(A) < \frac{s^2}{s+1}T(A\cup R\cup S). \tag{21}$$

A 3.6 Lemma miatt a teljes átfutási idő legfeljebb $\frac{1}{s}T(A)$. Felhasználjuk még, hogy $\frac{1}{s} > \frac{1}{s(s+x_0)} = \frac{x_0}{1+s-s^2}$, ami nyilvánvalóan teljesül, valamint a (21) feltételt is,

kapjuk:

$$\begin{aligned} w_{LSRM(\alpha)}(\mathcal{J}) &\leq \frac{1}{s}T(A) + P(R) + P(S) \\ &\leq \frac{1}{s}T(A) + \beta\frac{T(R)+T(S)}{s+1} \\ &= \frac{1}{s}T(A) + \frac{x_0}{1+s-s^2}(T(R)+T(S)) \\ &= \frac{x_0}{1+s-s^2}T(A\cup R\cup S) + \left(\frac{1}{s} - \frac{x_0}{1+s-s^2}\right)T(A) \\ &< \left[\frac{x_0}{1+s-s^2} + \left(\frac{1}{s} - \frac{x_0}{1+s-s^2}\right)\frac{s^2}{s+1}\right]T(A\cup R\cup S) \\ &= (s+x_0)\,T(A\cup R\cup S) \leq c(s)\,w^*(\mathcal{J}). \end{aligned}$$

4. eset Az optimális megoldás esetén az X munkát elfogadjuk, továbbá teljesül $T(X) > \frac{s}{s+1}T(A\cup R\cup S)$. Az esetet definiáló feltételből kapjuk, hogy $\frac{T(X)}{s} \geq T(A\cup R\cup S\setminus\{X\})$, vagy ezzel ekvivalens alakban

$$T(R)+T(S) \leq \frac{T(X)}{s} - T(A\setminus\{X\}). \tag{22}$$

Mivel az X munkát elfogadjuk az optimális megoldásnál, valamint $\frac{T(X)}{s} \geq T(A\cup R\cup S\setminus\{X\})$, a 3.4 Lemma miatt az optimum értéke $\frac{T(X)}{s}$. Most már készen vagyunk a 4. eset vizsgálatára. Két alesetet vizsgálunk meg:

4.1. eset $T(A\setminus\{X\}) \geq (s-1)\,T(X)$. Összevetve ezt a (22) feltétellel, kapjuk, hogy

$$T(R)+T(S) \leq \frac{T(X)}{s} - (s-1)\,T(X) = \frac{s+1-s^2}{s}T(X),$$

és emiatt

$$T(R) \leq \frac{s+1-s^2}{s}T(X). \tag{23}$$

Ezután a 3.6 Lemma, $\beta > 1$, valamint a (22), (23) feltételek alapján, kapjuk,

hogy

$$\begin{aligned}
w_{LSRM(\alpha)}(\mathcal{J}) &\leq \frac{T(A)+T(X)}{s+1} + P(R) + P(S) \\
&\leq \frac{T(A\setminus\{X\})}{s+1} + \frac{2T(X)}{s+1} + \beta\frac{T(R)+T(S)}{s+1} \\
&\leq \frac{T(X)}{s(s+1)} + \frac{2T(X)}{s+1} + (\beta-1)\frac{T(R)+T(S)}{s+1} \\
&\leq \frac{T(X)}{s+1}\left[\frac{1}{s} + 2 + (\beta-1)\,\frac{s+1-s^2}{s}\right] \\
&= \frac{T(X)}{s+1}\left[1+s+\beta\frac{s+1-s^2}{s}\right] = \frac{T(X)}{s+1}\left[1+s+\frac{x_0(s+1)}{s}\right] \\
&= \frac{s+x_0}{s}T(X) = c(s)\,w^*(\mathcal{J}).
\end{aligned}$$

4.2. eset $T(A\setminus\{X\}) < (s-1)\,T(X)$. Az előzőhöz hasonlóan, (22), és $\frac{1}{s} > \frac{\beta}{s+1}$ felhasználásával

$$\begin{aligned}
w_{LSRM(\alpha)}(\mathcal{J}) &\leq \frac{1}{s}T(A) + P(R) + P(S) \\
&\leq \frac{1}{s}T(A) + \beta\frac{T(R)+T(S)}{s+1} \\
&\leq \frac{1}{s}\left(T(A\setminus\{X\}) + T(X)\right) + \frac{\beta}{s+1}\left(\frac{T(X)}{s} - T(A\setminus\{X\})\right) \\
&= \left(\frac{1}{s} + \frac{\beta}{s(s+1)}\right)T(X) + \left(\frac{1}{s} - \frac{\beta}{s+1}\right)T(A\setminus\{X\}) \\
&\leq \left(\frac{1}{s} + \frac{\beta}{s(s+1)}\right)T(X) + \left(\frac{1}{s} - \frac{\beta}{s+1}\right)(s-1)\,T(X) \\
&= \left(1+\beta\frac{1+s-s^2}{s(s+1)}\right)T(X) = \frac{s+x_0}{s}T(X) = c(s)\,w^*(\mathcal{J}).
\end{aligned}$$

Mindegyik esetben beláttuk tehát, hogy $w_{LSRM(\alpha)}(\mathcal{J}) \leq c(s)\,w^*(\mathcal{J})$.

Az algoritmus versenyképességi aránya nem lehet jobb, mint $c(s)$. Tekintsük ugyanis azt a $\mathcal{J}$ sorozatot, amely egy darab munkából áll, amelynek a mérete $t_1 = \lambda(1+s-s^2)$ és a büntetése $p_1 = x_0$, ahol $0 < \lambda < 1$. Ekkor az algoritmus ezt a munkát a második gépre ütemezi, és teljesül $\frac{w_{LSRM(\alpha)(\mathcal{J})}}{w^*(\mathcal{J})} = \lambda\frac{1+s-s^2}{sx_0} = \lambda c(s)$ $\longrightarrow c(s)$ midőn $\lambda \longrightarrow 1$. □

Alsó korlátok. Bevezetjük az alábbi konstansokat: Legyen $1 < s_1 \approx 1.1915$, $s_2 \approx 1.3831$, valamint $s_3 \approx 1.3852 < \phi$, amely valós számok rendre a következő

egyenletek megoldásaiként adódnak:

$$\begin{aligned}
\frac{1}{2}+\sqrt{\frac{1}{4}+\frac{1}{s}} &= \frac{2s+1}{s+1}, \\
\frac{2s+1}{s+1} &= \frac{s\left(s-1\right)+x_0}{1+s-s^2}, \\
\frac{s\left(s-1\right)+x_0}{1+s-s^2} &= s+x_0.
\end{aligned}$$

3.8. Tétel *Tetszőleges, az online megszakítás nélküli kétgépes USR feladatot megoldó algoritmus versenyképességi aránya legalább*

$$\begin{cases}
\frac{1}{2}+\sqrt{\frac{1}{4}+\frac{1}{s}}, \text{ ha } 1 \leq s \leq s_1 \\
\frac{2s+1}{s+1}, \text{ ha } s_1 \leq s \leq s_2 \\
\frac{s(s-1)+x_0}{1+s-s^2}, \text{ ha } s_2 \leq s \leq s_3 \\
c\left(s\right)=s+x_0, \text{ ha } s_3 \leq s < \phi \\
\frac{s+1}{s}, \text{ ha } \phi \leq s.
\end{cases}$$

Bizonyítás: A bizonyítás egy kicsit körülményes, emiatt itt nem közöljük, [14]-ben megtalálható. □

3.9. Következmény *Tetszőleges $s_3 \leq s < \phi$ esetén az $LSRM(\alpha)$ algoritmus optimális.* □

Jegyezzük meg, hogy az itt leírt $LSRM(\alpha)$ algoritmus versenyképességi aránya minden $1 \leq s < \phi$ esetén jobb, mint a korábbi $LSR(\alpha)$ algoritmusé, továbbá az $LSRM(\alpha)$ algoritmus versenyképességi aránya és az alsó korlát közötti maximális rés 0.0534.

4. Ütemezési feladatok gépköltséggel

4.1. A gépköltséges feladat

A fejezet tartalma a [12] cikkben jelent meg:. Ennek, és a következő fejezetnek egy részben kibővített változata [23] a Veszprémi Akadémiai Bizottság 2005 évi pályázatán I. díjat kapott. A fejezetben szereplő algoritmusok jelen értekezés szerzőjének alkotásai, a $H1$ algoritmus vizsgálata a [12] cikk két szerzőjének közös munkája.

A [48] cikkben Imreh Csanád és John Noga felvetette klasszikus ütemezési feladatok egy újfajta megközelítését. Az eredeti változattól való különbségek a következők:

1, Nincs rögzítve kezdetben a gépek száma,

2, bármikor lehetőségünk van egy újabb gép vásárlására, amikor egy újabb munka megérkezik,

3, a célfüggvény, amit minimalizálunk, a gépek vásárlására fordított összeg és a teljes átfutási idő összege.

A feladat (amire Lista Modellként hivatkozunk) esetén, Imreh és Noga [48] közölte az A_ρ on-line algoritmust, amely $(1+\sqrt{5})/2 \approx 1.618$-versenyképes. Arra a félig online esetre amikor ismerjük a legnagyobb munka hosszúságot, He és Cai [42] közöl egy olyan algoritmust, amelynek a versenyképességi aránya nem nagyobb mint 1.5309, mindkét esetben az alsó korlát 4/3.

Mi ebben a fejezetben először megadunk egy 1.5798-versenyképes algoritmust a tiszta online esetre, tudomásunk szerint ezt a versenyképességi arányt azóta nem sikerült javítani. Ezután azzal a speciális esettel foglalkozunk, amikor a munkák mérete nem hosszabb mint a gépek vásárlásának költsége, vagyis minden munka hossza legfeljebb 1. Erre az esetre közlünk egy 4/3-versenyképes, ezáltal optimális félig online algoritmust. Harmadszor, közlünk egy legfeljebb 3/2-versenyképes kétfázisos algoritmust arra az esetre, amikor előre ismert a legnagyobb munka hosszúság. Ez az eredmény szintén megjavítja a korábbi hasonló esetre vonatkozó ismert eredményt.

Az eredeti A_ρ algoritmustól eltérően, más stratégiákat alkalmazunk arra, hogy mikor vásároljunk új gépet. (A_ρ egyszerűen a következőt végzi: Amikor a beérkezett munkák összhossza meghaladja az m^2 értéket, ahol m a gépek aktuális száma, vásárol egy új gépet, és a munkákat mindig a már létező gépekre, az LS algoritmus segítségével ütemezi.) Most viszont az algoritmusaink végrehajtása során amikor valamely pillanatban arról döntünk, hogy vegyünk-e új gépet, ezt a döntést két dolog befolyásolja: Az egyik a teljes átfutási idő aktuális értéke, a másik pedig az aktuális célfüggvényérték és annak alsó becslésének hányadosa.

4.2. Néhány előzetes megállapítás, jelölések

A következő oldalakon élünk a következő jelölésekkel: Jelölje $L = \max_{1,2,\dots,n}\{p_i\}$ a legnagyobb munka méretét, legyen $P = \sum_{i=1}^{n} p_i$ a munkák összmérete, valamint $P_k = \sum_{j=1}^{k} p_j$ az első k számú munka összmérete. Feltehető a munkák hosszának normalizálásával, hogy a gépek vásárlásának költsége 1. (Ezen azt értjük, hogy ha például a gépek vásárlásának költsége K, és valamely A online vagy félig online algoritmus c-versenyképes, akkor az algoritmusnak megadható egy olyan A' változata, amelyik pedig c-versenyképes akkor, ha a gépvásárlás költsége 1. Vagy hasonlóan, ha ρ egy alsó korlát az online vagy valamely félig online esetre vonatkozóan, vagyis valamely munka sorozat ütemezése esetén a heurisztikus és optimális megoldások aránya legalább ρ, akkor a munkák méretét K-val osztva olyan sorozatot kapunk, amelynél a heurisztikus és optimális megoldás aránya szintén legalább ρ abban az esetben, amikor a gépvásárlás költsége 1.) További jelölésként amennyiben egy A online vagy félig online algoritmus az összes munka megérkezte és ütemezése után m gépet használ, és $C_{\max}$ a teljes átfutási idő, akkor az A által kapott célfüggvény értékére az $A(I) = C_{\max} + m$ jelölést használjuk.

4.1. Lemma *([48]) Az online feladat esetén az optimum értéke legalább* $2\sqrt{P}$. *Továbbá, ha a maximális munka méretére teljesül az* $L \geq \sqrt{P}$ *egyenlőtlenség, akkor az optimum értéke legalább* $L + P/L$.

4.2. Megjegyzés. A lemma második állításával ekvivalens a következő állítás: Ha van olyan p_k munka, amelynek mérete legalább $p_k \geq \sqrt{P}$, akkor az optimum értéke legalább $p_k + P/p_k$.

Itt közöljük Imreh és Noga [48] A_ρ elnevezésű online algoritmusát, amely a "Lista Modell" feladatnak egy heurisztikus algoritmusa.

A_ρ **Algoritmus**

Legyen $\rho = \{\rho_1, \rho_2, \rho_3, \cdots, \rho_i, \cdots\} = \{0, 4, 9, 16, \cdots, i^2, \cdots\}$. Ekkor, bármely p_i munka megérkezésekor, az A_ρ algoritmus során úgy vásárolunk (vagy nem vásárolunk) gépeket, hogy a gépek aktuális j számára teljesüljenek a $\rho_j \leq P_i < \rho_{j+1}$ feltételek. (Vagyis amikor az első i számú munka összmérete éppen eléri vagy átlépi a ρ_j számot valamely j-re, akkor veszünk egy gépet.) Ezek után a p_i munkát az A_ρ algoritmus (valamelyik) minimális terhelésű gépre ütemezi, vagyis az LS szabály szerint.

A [48] cikkben Imreh és Noga megmutatta, hogy az A_ρ algoritmus $(1 + \sqrt{5})/2 \approx 1.618$-versenyképes. A következő lemma szerint a felső korlát élesíthető, ha a munkák hossza legfeljebb 3.

4.3. Lemma *([42])* $L \leq 3$ *esetén az* A_ρ *algoritmus (megfelelő félig online változata)* $3/2$*-versenyképes.*

Később megadunk egy olyan algoritmust amely abban az esetben is 3/2-versenyképes, ha a legnagyobb munkaméret, vagyis L,háromnál nagyobb. Elegendő ehhez tehát az a félig online feltétel is, hogy L értékét előre ismerjük.

4.4. Lemma *([48, 42]) Bármely félig online algoritmusnak* $4/3$ *alsó korlátja, akkor is, ha a munkák hosszúsága legfeljebb* 1*, és akkor is, ha ismert a maximális munka hosszúság.*

Bizonyítás: Tekintsük a munkáknak egy elég hosszú sorozatát, mindegyik munka mérete legyen $L = 1/N$, ahol N egy megfelelően nagy pozitív egész. Ha egy A algoritmus nem vásárol második gépet, ebben az esetben a munkák száma legyen $8N$, (vagyis a $8N$-edik után nem jön több munka). Következik, hogy $\frac{A(I)}{OPT(I)} \geq \frac{8N/N+1}{8N/(2N)+2} = \frac{3}{2}$. Most tegyük fel, hogy A akkor vásárolja a második gépet, amikor a p_i munka éppen megérkezett, ekkor szintén nem érkezik több munka. Ha $P_i \leq 2$, akkor az az optimális megoldás, amikor minden munka egy gépre kerül, és $\frac{A(I)}{OPT(I)} \geq \frac{P_i-1/N+2}{P_i+1} \geq \frac{4-1/N}{3}$. Ellenkező esetben, ha $P_i > 2$, akkor optimális ütemezés esetén a munkák körülbelül egyenlően vannak szétoszva két gépre, és $\frac{A(I)}{OPT(I)} \geq \frac{P_i-1/N+2}{P_i/2+2} \geq \frac{4-1/N}{3+1/N}$. Mivel N tetszőlegesen nagy lehet, a lemma állítása adódik. □

Mielőtt továbbmennénk, bevezetünk néhány jelölést, amelyekre szükségünk lesz. Legyen tehát valamely heurisztikus algoritmus végrehajtása során

m= a vásárolt gépek aktuális száma;

$m_0 = \lceil L \rceil$;

p_{i_j}= az első olyan munka, amelyik a j-edik gépre lett ütemezve, $j = 1, ..., m$.

Továbbá tekintsük azt a pillanatot valamely algoritmus végrehajtása során, amikor a p_k munka ütemezése éppen megtörtént valamely $k = 1, 2, \cdots, n$ esetén. Legyen

$s_{j,k}$= a j-edik gép pillanatnyi terhelése, $j = 1, 2, \cdots, m$;

s_k= a pillanatnyi terhelés értékek minimuma: $s_k = \min\{s_{j,k} | j = 1, ..., m\}$;

C_k = a pillanatnyi teljes átfutási idő;

z_k = a pillanatnyi célfüggvény érték, vagyis $z_k = C_k + m$;

$C_{1,k} = 2\sqrt{P_k}$, a már beérkezett munkákra vonatkozó optimális célfüggvényértéknek egy alsó becslése;

$C_{2,k} = L + \frac{P_k}{L}$, a már beérkezett munkákra vonatkozó optimális célfüggvényértéknek egy másik alsó becslése, amikor $L \geq \sqrt{P_k}$, és van legalább L hosszúságú munka.

4.3. Egy hatékonyabb algoritmus az online esetre

A Lista Modell feladattal foglalkozunk. Közöljük a $H1$ algoritmust, amelynek versenyképességi hányadosa nem több mint $\frac{2\sqrt{6}+3}{5} \approx 1.5798$. Az A_ρ algoritmussal ellentétben, $H1$ új stratégiát követ arra vonatkozólag hogy mikor vásároljunk újabb gépet. Egy újólag érkező munkát mindig olyan gépre ütemez, amelyiknek az

aktuális (vagy pillanatnyi) terhelése minimális, (vagyis az LS algoritmus szerint), feltéve, hogy a gép megnövelkedett terhelése nem lesz nagyobb, mint $2m$, ahol m a gépek aktuális száma. Ha viszont bármely gépre ütemezve az új munkát a teljes átfutási idő ennél nagyobb lenne, akkor az új munka egy újólag vásárolt gépre kerül. Formálisan a következőleg lehet leírni az algoritmust:

$H1$ **Algoritmus:**

1. Ütemezzük a p_1 munkát az első gépre. Legyen $k = 2, m = 1$.
2. Ütemezzük a p_k munkát az LS ütemezési algoritmus szerint, feltéve, hogy az új teljes átfutási idő nem nagyobb mint $2m$, és menjünk a 4. lépésre. Ha bármelyik gépre ütemezve a munkát a gép megnövelkedett terhelése nagyobb lenne mint $2m$, akkor menjünk a 3. lépésre.
3. Ütemezzük a p_k munkát egy új gépre, legyen $m = m + 1$, menjünk a 4. lépésre.
4. Legyen $k = k + 1$, $k > n$ esetén stop. Ellenkező esetben menjünk újra a 2. lépésre.

4.5. Lemma *Legyen $m \geq 2$ a gépek aktuális száma. Ha a pillanatnyi minimális terhelésre teljesül az $s_k \leq m - 1$ egyenlőtlenség, akkor a pillanatnyi terhelések összege legalább*

$$\sum_{j=1}^{m} s_{j,k} \geq s_k^2 - (m-1)s_k + (m-1)m.$$

Bizonyítás: Nyilvánvaló, hogy tetszőleges $1 \leq j < l \leq m$ esetén teljesül az

$$s_{j,k} + s_{l,k} \geq s_k + p_{i_l} > 2\,(l-1)$$

egyenlőtlenség az algoritmus szabálya miatt. Tekintsük az alábbi paraméteres nemlineáris programot, ahol az s_k paraméter értéke a $[0, m-1]$ intervallumból való:

$$\begin{aligned} \min \quad & u(s_k) = \sum_{j=1}^{m} s_{j,k} \\ s.t. \quad & s_{j,k} + s_{l,k} \geq 2\,(l-1)\,, \quad 1 \leq j < l \leq m, \\ & \min\left\{s_{j,k} : 1 \leq j \leq m\right\} = s_k. \end{aligned}$$

Tegyük fel, hogy $i \leq s_k < i+1 \leq m-1$ teljesül valamely $i \geq 0$ egész esetén. Ekkor könnyen látható, hogy az optimális megoldás a következő:

$$s_{j,k} = \begin{cases} s_k, & ha \;\; 1 \leq j \leq i+1, \\ 2(j-1) - s_k, & ha \;\; i+1 < j \leq m, \end{cases}$$

és a minimum értéke pedig

$$u^*(s_k) = \begin{cases} s_k^2 - (m-1)s_k + (m-1)m, & ha \;\; s_k = i, \\ (m+i)(m-i-1) - ms_k + 2(i+1)s_k, & ha \;\; i < s_k < i+1. \end{cases}$$

Definiáljuk az $f_1(s_k) = s_k^2 - (m-1)s_k + (m-1)m$ függvényt a $[0, m-1]$ intervallumon, és az $f_2(s_k) = (m+i)(m-i-1) - ms_k + 2(i+1)s_k$ függvényt az $[i, i+1]$ intervallumon minden $i = 0, 1, \cdots, m-2$ esetén. Mivel $f_2(s_k)$ lineáris az $[i, i+1]$ intervallumon, továbbá $f_2(s_k) = f_1(s_k)$ minden $s_k = 0, 1, \cdots, m-1$ esetén, így az $u^*(s_k)$ függvény folytonos és szakaszonként lineáris a $[0, m-1]$ intervallumon. Továbbá, az $f_1(s_k)$ függvény konvex a $[0, m-1]$ intervallumon, következik az előzőkből, hogy $u^*(s_k) \geq f_1(s_k)$ minden $0 \leq s_k \leq m-1$ esetén. Emiatt $u(s_k) \geq u^*(s_k) \geq f_1(s_k) = s_k^2 - (m-1)s_k + (m-1)m$, minden $0 \leq s_k \leq m-1$ esetén. □

4.6. Következmény *Ha a 4.5 Lemma feltételei teljesülnek, akkor teljesül a következő egyenlőtlenség:*

$$\sum_{j=1}^{m} s_{j,k} \geq \frac{3}{4}(m-1)^2 + (m-1) > \frac{3}{4}m(m-1).$$

Bizonyítás: Az $s_k^2 - (m-1)s_k + (m-1)m$ függvény a minimális értékét $s_k = \frac{m-1}{2}$ esetén veszi fel. □

4.7. Megjegyzés *Legyen $m \geq 2$ a gépek pillanatnyi száma, és tegyük fel, hogy a minimális terhelésre teljesül az $s_k \geq m-1$ egyenlőtlenség. Ekkor $\sum_{j=1}^{m} s_{j,k} \geq (m-1)m$ triviálisan teljesül.*

A $H1$ algoritmus versenyképességi arányának levezetéséhez két további jelölésre van szükségünk az alábbiak szerint: Tetszőleges $l \geq 1$ esetén legyen $u_l = \max\{p_i | 1 \leq i \leq l\}$, valamint

$$C'_l = \begin{cases} C_{1,l}, & \text{ha} \quad u_l \leq \sqrt{P_l}, \\ C_{2,l}, & \text{ha} \quad u_l > \sqrt{P_l}. \end{cases}$$

Nyilvánvaló, hogy tetszőleges $l \geq 1$ esetén teljesül hogy $C'_l \geq C_{1,l}$.

4.8. Lemma *A C'_l konstans értéke növekvő l -ben, vagyis $C'_1 \leq C'_2 \leq \cdots \leq C'_n$.*

Bizonyítás: Ha $C'_{l-1} = C_{1,l-1}$, akkor triviálisan teljesül $C'_l \geq C_{1,l} > C_{1,l-1} = C'_{l-1}$. Emiatt tegyük fel, hogy $C'_{l-1} = C_{2,l-1}$. Ha még $C'_l = C_{2,l}$ is teljesül, akkor az $u_{l-1} \geq \sqrt{P_{l-1}}$ feltételből kapjuk, hogy $C'_l = C_{2,l} = \frac{P_l}{u_l} + u_l > \frac{P_{l-1}}{u_l} + u_l \geq \frac{P_{l-1}}{u_{l-1}} + u_{l-1} = C_{2,l-1} = C'_{l-1}$. Ellenkező esetben pedig $C'_l = C_{1,l}$, ekkor tudjuk hogy $u_{l-1}^2 > P_{l-1}$ és $u_l^2 \leq P_l$. Következik, hogy $C'_l = C_{1,l} = 2\sqrt{P_l} \geq 2u_l \geq u_{l-1} + u_{l-1} > u_{l-1} + \frac{P_{l-1}}{u_{l-1}} = C_{2,l-1} = C'_{l-1}$, és a bizonyítás kész. □

4.9. Tétel *A $H1$ algoritmus versenyképességi aránya nem több, mint $\alpha \doteq \frac{2\sqrt{6}+3}{5} \approx 1.5798$.*

Bizonyítás: Teljes indukciót használunk, belátjuk, hogy minden $l = 1, 2, \cdots, n$ esetén $z_l \leq \alpha C'_l$ teljesül. Ebből nyilván következik a bizonyítandó állítás. Az állítás teljesül $l = 1$ esetén. Tegyük fel, hogy teljesül az állítás $l = k - 1$-re, két esetet kell hogy megvizsgáljunk, aszerint, hogy p_k új gépre kerül-e, vagy sem.

1. eset p_k nem kerül új gépre, vagyis a már meglévő m gép valamelyikére kerül ütemezésre az algortitmus 2. lépése szerint. Amennyiben a teljes átfutási idő nem növekszik, (vagyis $C_k = C_{k-1}$), akkor $z_k = z_{k-1}$, és így az állításunk triviálisan teljesül $j = k$ esetén is, az indukciós feltevés, és a 4.8. Lemma miatt. Emiatt feltehető, hogy a teljes átfutási idő növekszik a p_k munka ütemezése által, emiatt $C_k = s_{k-1} + p_k$, továbbá

$$z_k = C_k + m = s_{k-1} + p_k + m. \tag{24}$$

Három alesetet különböztetünk meg, az m gépszám aktuális értéke szerint.

1.1. eset $m = 1$ vagy $m = 2$. Az $\alpha C'_k \geq \alpha C_{1,k} \geq 3\sqrt{P_k}$ egyenlőtlenség-lánc, valamint (24) miatt, elég belátnunk az

$$s_{k-1} + p_k + m \leq 3\sqrt{P_k}. \tag{25}$$

egyenlőtlenség teljesülését. $m = 1$ esetén az algoritmus szabálya miatt $s_{k-1}+p_k \leq 2m = 2$. Ha $s_{k-1}+p_k < 1$, akkor $z_k < 2$, emiatt optimális ütemezés esetén szintén pontosan egy gép van, és ennek következtében a $H1$ algoritmus által készített ütemezés optimális. Amennyiben $1 \leq s_{k-1} + p_k \leq 2$ teljesül, akkor triviálisan következik $s_{k-1} + p_k + 1 \leq 3\sqrt{s_{k-1} + p_k}$, és adódik (25).

Legyen $m = 2$, ekkor az algoritmus szabályából következően kapjuk, hogy $s_{k-1}+p_k \leq 4$. Másrészt, rögtön azután, hogy p_{i_2} a második gépre került, a munkák összhossza több mint 2, emiatt az $s_{k-1}+p_k$ teljes átfutási időnek nagyobbnak kell lennie mint 1. Továbbá $1 < s_{k-1} + p_k \leq 4$ következtében teljesül $s_{k-1} + p_k + 2 \leq 3\sqrt{s_{k-1} + p_k}$, és így $s_{k-1} + p_k + 2 \leq 3\sqrt{s_{k-1} + p_k} \leq 3\sqrt{P_k}$.

1.2. eset $2 < m \leq s_{k-1} + 1$. Ha még az is teljesül, hogy $m \leq s_{k-1}$, akkor triviálisan következik $P_k \geq s_{k-1}m + p_k \geq m^2$, és így $z_k = s_{k-1} + p_k + m \leq 2m + m \leq 3\sqrt{P_k} = \frac{3}{2}C_{1,k}$. Ezzel megkaptuk a kívánt felső becslést a versenyképességi arányra vonatkozólag, emiatt feltehetjük, hogy $m - 1 \leq s_{k-1} < m$. Amennyiben $p_k > s_{k-1} + 1$ teljesül, akkor $P_k \geq s_{k-1}m + p_k \geq (m-1)\,m + m = m^2$, és újra $z_k \leq \frac{3}{2}C_{1,k}$. Ha $p_k \leq s_{k-1}$, akkor a (24) egyenlőtlenséget és az $\frac{1}{4} \leq \frac{s_{k-1}}{m} \leq 1$ feltételt felhasználva kapjuk, hogy $z_k \leq 2s_{k-1} + m \leq 3\sqrt{s_{k-1}m} \leq 3\sqrt{P_k} = \frac{3}{2}C_{1,k}$. Utoljára, ha $s_{k-1} < p_k \leq s_{k-1} + 1$, akkor hasonlóképpen $z_k \leq 2s_{k-1} + 1 + m \leq 3\sqrt{s_{k-1}m + s_{k-1}} \leq 3\sqrt{P_k} = \frac{3}{2}C_{1,k}$ hiszen $\frac{1}{4} \leq \frac{s_{k-1}}{m+1} \leq 1$.

1.3. eset $m > s_{k-1} + 1$. Ha $p_k \leq m - s_{k-1}$, akkor $z_k \leq 2m$ a (24) feltételből következően. A 4.6. Következmény miatt kapjuk hogy $P_k \geq \frac{3}{4}m\,(m-1)$. Másrészt $m > 2$ következtében $2m \leq 3\sqrt{\frac{3}{4}m\,(m-1)}$, és $z_k \leq \frac{3}{2}C_{1,k}$ adódik. Emiatt feltehetjük a

$$p_k > m - s_{k-1}. \tag{26}$$

egyenlőtlenség teljesülését. Összevetve ezt a 4.5. Lemmával, kapjuk, hogy

$$\begin{aligned} P_k = \sum_{j=1}^{m} s_{j,k} + p_k &\geq s_{k-1}^2 - (m-1)\,s_{k-1} + (m-1)\,m + p_k \\ &> s_{k-1}^2 - ms_{k-1} + m^2. \end{aligned} \tag{27}$$

Amennyiben $p_k \leq \sqrt{P_k}$ teljesül, a (24) és (27) feltételek miatt kapjuk hogy

$$z_k \leq \sqrt{P_k} + s_{k-1} + m \leq \sqrt{P_k} + 2\sqrt{s_{k-1}^2 - ms_{k-1} + m^2} < 3\sqrt{P_k} = \frac{3}{2}C_{1,k}.$$

Emiatt feltehetjük, hogy $p_k \geq \sqrt{P_k}$. Következik a 4.1. Lemmából, hogy $p_k + \frac{P_k}{p_k}$ $(\leq C'_k)$ alsó korlátja az aktuális célfüggvény értéknek. Annak érdekében, hogy megkapjuk a felső becslést a versenyképességi arányra, a (24) és (27) feltételeket felhasználva elég belátnunk az

$$s_{k-1} + p_k + m - \alpha\left(p_k + \frac{s_{k-1}^2 - ms_{k-1} + m^2}{p_k}\right) \leq 0 \tag{28}$$

egyenlőtlenség teljesülését. Ennek bizonyítását is több esetre bontjuk. Először tegyük fel, hogy $\frac{s_{k-1}}{m} > \frac{3}{5}$. Mivel $\alpha \leq \frac{8}{5}$, $s_{k-1} + p_k \leq 2m$, továbbá ${p_k}^2 \geq P_k > s_{k-1}^2 - ms_{k-1} + m^2$, a bal oldali függvény p_k szerinti deriváltjára

$$1 - \alpha\left(1 - \frac{s_{k-1}^2 - ms_{k-1} + m^2}{{p_k}^2}\right) > 1 - \frac{8}{5}\left(1 - \frac{s_{k-1}^2 - ms_{k-1} + m^2}{(2m - s_{k-1})^2}\right) > 0,$$

adódik, ahol az utolsó egyenlőtlenség ekvivalens azzal hogy $5(\frac{s_{k-1}}{m})^2 + 4\frac{s_{k-1}}{m} - 4 > 0$, ami triviálisan teljesül $\frac{s_{k-1}}{m} > \frac{3}{5}$ miatt. Emiatt elég azt belátnunk, hogy a

(28) feltétel teljesül, ha $p_k = 2m - s_{k-1}$. Helyettesítsük be a $p_k = 2m - s_{k-1}$ egyenlőséget (28)-ba, kapjuk hogy $3m \leq \alpha\left(2m - s_{k-1} + \frac{s_{k-1}^2 - m s_{k-1} + m^2}{2m - s_{k-1}}\right)$, vagy ezt átrendezve

$$3 \leq \alpha\left(2 - \frac{s_{k-1}}{m} + \frac{(\frac{s_{k-1}}{m})^2 - \frac{s_{k-1}}{m} + 1}{2 - \frac{s_{k-1}}{m}}\right). \tag{29}$$

Az $f_3\left(\frac{s_{k-1}}{m}\right) \doteq 2 - \frac{s_{k-1}}{m} + \frac{(\frac{s_{k-1}}{m})^2 - \frac{s_{k-1}}{m} + 1}{2 - \frac{s_{k-1}}{m}}$ függvény minimális értékét akkor veszi fel, ha $\frac{s_{k-1}}{m} = 2 - \sqrt{\frac{3}{2}}$, és ekkor $f_3\left(2 - \sqrt{\frac{3}{2}}\right) = 4\sqrt{\frac{3}{2}} - 3$. Mivel $\alpha \cdot (4\sqrt{\frac{3}{2}} - 3) = 3$, kész is vagyunk.

Most tegyük fel, hogy $\frac{s_{k-1}}{m} \leq \frac{3}{5}$. (28) bal oldalának s_{k-1} szerinti deriváltja $1 - \alpha\frac{2s_{k-1} - m}{p_k}$. Amennyiben $1 - \alpha\frac{2s_{k-1} - m}{p_k} \leq 0$ teljesül, akkor $\frac{s_{k-1}}{m} \leq \frac{3}{5}$ következtében $s_{k-1} + p_k \leq s_{k-1} + \alpha\,(2s_{k-1} - m) < \frac{3}{5}m + \frac{8}{5} \cdot \frac{1}{5}m < m$, ami ellentmond a (26) feltételnek. Ellenkező esetben a derivált értéke pozitív. Ekkor a (28) feltétel teljesülését elég belátnunk $s_{k-1} = \frac{3}{5}m$ esetén. Vagyis azt kell belátnunk, hogy

$$p_k + \frac{3}{5}m + m \leq \alpha\left(p_k + \frac{\left(\frac{9}{25} - \frac{3}{5} + 1\right)m^2}{p_k}\right). \tag{30}$$

A jobb oldal a következőképpen alakítható: $p_k + (\alpha - 1)p_k + \frac{19\alpha m^2}{25 p_k} \geq p_k + 2\sqrt{\frac{19(\alpha-1)\alpha}{25}}m > p_k + \frac{8}{5}m$, és így a bizonyításunk az 1. esetben készen is van.

2. eset. p_k az első munka amelyik az m-edik gépre kerül (vagyis p_k-t az algoritmus harmadik lépésével ütemezzük). Emiatt teljesül hogy $m \geq 2$ valamint $s_{j,k-1} + p_k > 2(m-1)$ minden $1 \leq j \leq m-1$ esetén. Az algoritmus szabályából következőleg tudjuk hogy $s_{j,k-1} \leq \max\{u_k, 2(m-1)\}$ teljesül minden $1 \leq j \leq m-1$ esetén. Most előkészítésképpen belátunk néhány olyan állítást, amelyekre a 2. eset vizsgálata során szükségünk lesz.

Először belátjuk, hogy $u_k \leq 2(m-1)$. Tegyük fel indirekt módon, hogy ez nem igaz, vagyis $u_k > 2(m-1)$. Mivel a teljes átfutási idő a p_k munka ütemezése előtti pillanatban nem több mint u_k, tudjuk hogy $P_k = \sum_{j=1}^{m-1} s_{j,k-1} + p_k \leq (m-1)u_k + u_k = u_k m \leq u_k \cdot 2(m-1) < u_k^2$. Következik, hogy $C'_k = C_{2,k}$, és az is, hogy $z_k \leq \max\{p_k, 2(m-1)\} + m \leq u_k + m < \frac{3}{2}(u_k + 1) \leq \frac{3}{2}\left(u_k + \frac{P_k}{u_k}\right)$, és ezzel kész is vagyunk. Beláttuk tehát, hogy $u_k \leq 2(m-1)$. Emiatt viszont teljesül, hogy

$$C_{k-1} = \max\{s_{j,k-1} \mid 1 \leq j \leq m-1\} \leq 2(m-1). \tag{31}$$

Ezek után belátjuk a következő két egyenlőtlenséget: $P_k < (m - \frac{2}{3})^2$ és $p_k > m - \frac{2}{3}$. Ha $P_k \geq (m - \frac{2}{3})^2$, akkor $z_k \leq \max\{u_k, 2(m-1)\} + m \leq 3m - 2 \leq 3\sqrt{P_k} = \frac{3}{2}C_{1,k}$, és a felső becslés a versenyképességi arányra teljesül. Emiatt a következőkben feltesszük a $P_k < (m - \frac{2}{3})^2$ egyenlőtlenség teljesülését. Most

tegyük fel, hogy $p_k \leq m - \frac{2}{3}$. Vegyük észre, hogy $s_{j,k-1} > 2(m-1) - p_k$ teljesül minden $1 \leq j \leq m-1$ esetén, emiatt $P_k > (m-1)\left(2(m-1) - p_k\right) + p_k \geq 2(m-1)^2 - (m-2)\left(m - \frac{2}{3}\right) > (m - \frac{2}{3})^2$, ami ellentmond az előbbi, már belátott egyenlőtlenségnek. Vagyis feltehetjük azt is hogy $p_k > m - \frac{2}{3}$.

A $P_k < (m - \frac{2}{3})^2$ és $p_k > m - \frac{2}{3}$ feltételek miatt $p_k \geq \sqrt{P_k}$ teljesül, és így következik hogy $C'_k = C_{2,k}$. Most újra három aleset következik, C_{k-1} és p_k lehetséges értékei szerint.

2.1. eset $C_{k-1} \leq p_k$. Ekkor p_k a leghosszabb idejű munka, és $z_k = p_k + m$. A 4.6. Következmény valamint a 4.7. Megjegyzés következtében tudjuk hogy $P_k > \frac{3}{4}(m-1)m$, így elég belátnunk a $p_k + m \leq \frac{3}{2}\left(p_k + \frac{\frac{3}{4}(m-1)m}{p_k}\right)$ egyenlőtlenség teljesülését, vagyis ezt átrendezve, azt kell megmutatni hogy

$$m \leq \frac{p_k}{2} + \frac{9(m-1)m}{8p_k}. \tag{32}$$

A számtani-mértani közepekre vonatkozó egyenlőtlenség szerint (32) jobboldala legalább $2\sqrt{\frac{9}{16}(m-1)m} = \frac{3}{2}\sqrt{(m-1)m} \geq m$, készen vagyunk.

2.2. eset $C_{k-1} > p_k$ és $u_k \geq \frac{3}{2}(m-1)$. Az előző esethez hasonlóan adódik, hogy $P_k > \frac{3}{4}(m-1)^2 + m - 1$ a 4.6. Következmény valamint a 4.7. Megjegyzés miatt. Összevetve ezt a (31) egyenlőtlenséggel, elég azt belátnunk, hogy

$$z_k = C_{k-1} + m \leq 2(m-1) + m \leq \frac{3}{2}\left(u_k + \frac{\frac{3}{4}(m-1)^2 + m - 1}{u_k}\right). \tag{33}$$

Ennek belátása ekvivalens a $0 \leq 12{u_k}^2 - 8(3m-2)u_k + 9m^2 - 6m - 3$ egyenlőtlenség belátásával, ami pedig a következővel:

$$(u_k - \frac{3(m-1)}{2})(u_k - \frac{3m+1}{6}) \geq 0. \tag{34}$$

Mivel $u_k \geq \frac{3}{2}(m-1)$, (34) nyilvánvalóan teljesül.

2.3. eset $C_{k-1} > p_k$ és $u_k < \frac{3}{2}(m-1)$. Két esetet különböztetünk meg az alábbiak szerint:

Ha $z_{k-1} \leq \alpha C_{1,k-1}$, vagyis $C_{k-1} + m - 1 \leq 2\alpha\sqrt{P_{k-1}}$ teljesül, belátjuk, hogy $z_k \leq \alpha C_{1,k}$ szintén teljesül, ami ekvivalens a következővel: $C_{k-1} + m \leq 2\alpha\sqrt{P_k} = 2\alpha\sqrt{P_{k-1} + p_k}$. Elég belátnunk, hogy $2\alpha\sqrt{P_{k-1} + p_k} \geq 2\alpha\sqrt{P_{k-1}} + 1$, vagyis a következő egyenlőtlenséget:

$$p_k \geq \frac{1}{\alpha}\sqrt{P_{k-1}} + \frac{1}{4\alpha^2}. \tag{35}$$

Most alkalmazva a $p_k > m - \frac{2}{3}$ és $P_k < (m - \frac{2}{3})^2$ egyenlőtlenségeket, kapjuk hogy $p_k > m - \frac{2}{3} = \frac{1}{\alpha}\left(m - \frac{2}{3}\right) + (1 - \frac{1}{\alpha})(m - \frac{2}{3}) > \frac{1}{\alpha}\sqrt{P_k} + \frac{1}{4\alpha^2} > \frac{1}{\alpha}\sqrt{P_{k-1}} + \frac{1}{4\alpha^2}$.

Amennyiben $z_{k-1} \leq \alpha C_{2,k-1}$, vagyis $C_{k-1} + m - 1 \leq \alpha \left(u_{k-1} + \frac{P_{k-1}}{u_{k-1}} \right)$ teljesül, akkor az indukciós feltevés miatt $u_{k-1} \geq \sqrt{P_{k-1}}$ adódik. Következik, hogy $\alpha \left(u_{k-1} + \frac{P_{k-1}}{u_{k-1}} \right) \leq \alpha \left(u_k + \frac{P_{k-1}}{u_k} \right)$. Elég belátnunk hogy

$$\alpha \frac{P_{k-1} + p_k}{u_k} \geq \alpha \frac{P_{k-1}}{u_k} + 1, \tag{36}$$

ebből ugyanis adódik hogy $z_k = C_{k-1} + m \leq \alpha \left(u_k + \frac{P_{k-1}+p_k}{u_k} \right) = \alpha C_{2,k}$. A (36) feltétel ekvivalens azzal hogy $p_k \geq \frac{1}{\alpha} u_k$, ami rögtön következik a $p_k > m - \frac{2}{3}$ és $u_k < \frac{3}{2}(m-1)$ feltételekből.

Összefoglalva az előzőket, beláttuk hogy az indukciós állításunk $l = k$ esetén is teljesül, és ezzel beláttuk a 4.9. Tételt. □

4.4. Optimális félig online algoritmus rövid munkák esetén

Ebben a fejezetben egy olyan félig online esettel foglalkozunk, amikor előre tudjuk, hogy a bármelyik munka mérete legfeljebb 1, vagyis $L \leq 1$. Megadunk egy optimális félig online algoritmust amelynek a versenyképességi aránya 4/3. Az algoritmus majdnem megegyezik a $H1$ algoritmussal, az egyedüli különbség csak a 2. lépésben van.

Jegyezzük meg, hogy az A_ρ algoritmus nem lehet jobb mint 3/2-versenyképes a kis méretű munkák esetén. Tekintsük ugyanis a $p_1 = \cdots = p_{4N} = 1/N$ munkák sorozatát, ahol N elegendően nagy pozitív egész. Erre a sorozatra $A_\rho(I)/OPT(I) = (6 - 1/N)/4 \longrightarrow 3/2$ teljesül.

$H2$ **Algoritmus**:

1. Ütemezzük a p_1 munkát az első gépre. Legyen $k = 2, m = 1$.

2. Ütemezzük a p_k munkát olyan gépre, amelynek aktuális terhelése mnimális (vagyis az LS ütemezési algoritmus szerint), feltéve, hogy az új teljes átfutási idő nem nagyobb mint $m + 1$, és menjünk a 4. lépésre. Ha bármely gépre ütemezve a munkát a gép megnövekedett terhelése nagyobb lenne mint $m + 1$, akkor menjünk a 3. lépésre.

3. Ütemezzük a p_k munkát egy új gépre, legyen $m = m + 1$, menjünk a 4. lépésre.

4. Legyen $k = k + 1$, $k > n$ esetén stop. Ellenkező esetben menjünk újra a 2. lépésre.

4.10. Lemma *Legyen $m \geq 2$ pozitív egész, és tegyük fel hogy b-re teljesül $m < b \leq m+1$.*

(1) Definiáljuk az $f_4(i)$ függvényt a következőképpen: $f_4(i) = \frac{(m-1)^2+1}{i} + i$, ahol i tetszőleges pozitív egész. Ekkor $f_4(i)$ minimális értéke $2(m-1) + \frac{1}{m-1}$.

(2) Definiáljuk az $f_5(i)$ függvényt úgy, hogy $f_5(i) = \frac{m(b-1)}{i} + i$, szintén tetszőleges pozitív egész i szám esetén. Ekkor $f_5(i)$ minimális értéke $b+m-1$.

Bizonyítás: (1) Legyen $g(x) = \frac{(m-1)^2+1}{x} + x$ tetszőleges $x \geq 1$ valós esetén. Ekkor

$$g'(x) = \frac{1}{x^2}\left(x^2 - \left((m-1)^2+1\right)\right) = \frac{x+\sqrt{(m-1)^2+1}}{x^2}\left(x - \sqrt{(m-1)^2+1}\right),$$

$$g''(x) = \frac{2((m-1)^2+1)}{x^3} > 0.$$

Emiatt $g(x)$ minimális értéke $x = \sqrt{(m-1)^2+1}$ esetén vétetik fel, továbbá ebből következően $f_4(i)$ minimális értéke $i = \left\lfloor \sqrt{(m-1)^2+1} \right\rfloor = m-1$ vagy $i = \left\lceil \sqrt{(m-1)^2+1} \right\rceil = m$ esetén vétetik fel. Másrészt könnyen látható, hogy $f_4(m-1) = \frac{(m-1)^2+1}{m-1} + m - 1 \leq f_4(m) = \frac{(m-1)^2+1}{m} + m$ tetszőleges $m \geq 2$ esetén. Mindezekből (1) már következik.

(2) Az előző állításhoz hasonlóan látható hogy a minimum $i = m$ esetén vétetik fel. □

4.11. Tétel *Amennyiben a munkák hossza legfeljebb 1, akkor a $H2$ algoritmus versenyképességi aránya $4/3$, emiatt optimális algoritmus.*

Bizonyítás: A 4.9. Tétel bizonyításához hasonlóan járunk el: néhány esetet különböztetünk meg aszerint, hogy $H2$ összesen hány gépet vásárol (ezek száma m'), illetve hogy az utolsó munka, p_n, melyik gépre kerül.

1. eset $m' = 1$. Ez esetben az algoritmus szabályából következően kapjuk hogy $C_{\max} = P = \sum_{i=1}^{n} p_i \leq 2$, valamint azt is hogy $H2(I) = C_{\max} + m' \leq 3$. Következik, hogy optimális esetben is legfeljebb két gép van. Amennyiben egyetlen optimális gép lenne, az esetben $H2$ optimális megoldást határoz meg. Ellenkező esetben, vagyis ha két optimális gép van, akkor pedig könnyen látható, hogy $OPT(I) \geq \frac{P}{2} + 2$ teljesül. Ebből nyilvánvaló, hogy $H2(I) = P + 1 \leq \frac{4}{3}\left(\frac{P}{2} + 2\right) \leq \frac{4}{3}OPT(I)$.

2. eset $m' \geq 2$ és p_n az első munka, amely az m'-edik gépre lett ütemezve (vagyis p_n az algoritmus 3. lépése szerint lett ütemezve). Ekkor a p_n ütemezése előtti pillanatban a gépek száma pontosan $m'-1$. Az algoritmus szabálya szerint $s_{j,n-1} + p_n > m'$ teljesül minden $j = 1, ..., m'-1$ esetén. Emiatt

$\sum_{j=1}^{m'-1}(s_{j,n-1}+p_n) = \sum_{j=1}^{m'-1} s_{j,n-1} + (m'-1)p_n > m'(m'-1)$. Ezt összevetve azzal hogy $p_n \leq 1$, adódik, hogy

$$\begin{aligned} P &= \sum_{j=1}^{m'-1} s_{j,n-1} + p_n > m'(m'-1) - (m'-2)\,p_n \\ &\geq m'(m'-1) - (m'-2) = (m'-1)^2 + 1. \end{aligned}$$

Ebből egyszerűen adódik, hogy $OPT(I) \geq \min_{i\geq 1}\left\{\frac{(m'-1)^2+1}{i} + i\right\}$, ahol az i szám azon gépszámokat jelenti, amelyek lehetségesek optimális ütemezés esetén. A 4.10 Lemma (1) része szerint $OPT(I) \geq 2(m'-1) + \frac{1}{m'-1}$. Másrészt nyilvánvaló, hogy $C_{\max} \leq m'$, emiatt $H2(I) = C_{\max} + m' \leq 2m'$. Minden $m' \geq 2$ esetén $\frac{2m'-3}{2m'^2-4m'+3} \leq \frac{4}{3}$ triviálisan teljesül. Ezek miatt végül kapjuk, hogy

$$\frac{H2(I)}{OPT(I)} \leq \frac{2m'}{2(m'-1)+\frac{1}{m'-1}} = \frac{2\,(m'-1)\,m'}{2(m'-1)^2+1} = 1 + \frac{2m'-3}{2m'^2-4m'+3} \leq \frac{4}{3}.$$

3. eset $m' \geq 2$ és p_n nem kerül új gépre, vagyis a már létező m' gép valamelyikére ütemezi az algoritmus a 2. lépésben. Nyilvánvaló, hogy a végső teljes átfutási idő értéke $C_{\max} \leq m'+1$, és $H2(I) = C_{\max} + m'$. Ha még $C_{\max} \leq m'$ is teljesül, akkor $H2(I) \leq 2m'$. Tegyük fel, hogy az m'-edik gépet a p_k munka ütemezésekor vásároljuk. Ugyanúgy, mint az előző pontban, kapjuk, hogy $P_n \geq P_k \geq (m'-1)^2+1$ és így $H2(I)/OPT(I) \leq 4/3$.

Emiatt feltehetjük, hogy $m' < C_{\max} \leq m'+1$. Mivel p_n olyan gépre került, amelynek az aktuális terhelése minimális, mindegyik gép terhelése legalább $C_{\max} - p_n \geq C_{\max} - 1$, így a terhelések összegére, ami egyenlő a munkák hosszainak összegével $P \geq (m'-1)\,(C_{\max}-1) + C_{\max} > m'\,(C_{\max}-1)$ adódik. Következik, hogy $OPT(I) \geq \min_{i\geq 1}\left\{\frac{m'(C_{\max}-1)}{i} + i\right\}$. A 4.10 Lemma (2) része szerint pedig $OPT(I) \geq C_{\max} + m' - 1$. Következik, hogy

$$\frac{H2(I)}{OPT(I)} \leq \frac{C_{\max}+m'}{C_{\max}+m'-1} = 1 + \frac{1}{C_{\max}+m'-1} \leq \frac{4}{3},$$

ahol az utolsó egyenlőtlenség amiatt teljesül, mert $m' \geq 2$ és $C_{\max} > m' \geq 2$.

Ezáltal beláttuk, hogy $H2$ 4/3-versenyképes, az optimalitás pedig ennek, és a 4.4. Lemmának a közvetlen következménye. □

4.12. Megjegyzés *$H2$ egy kis módosítással az $1 < L \leq 3$ esetben is működik, csak megint a 2. lépésben kell hogy kicseréljük az $m+1$ felső korlátot $m+L$ -lel. Ekkor az előzőekhez hasonlóan látható be, hogy a módosított $H2$ versenyképességi aránya nem nagyobb, mint $3/2$ (arra a félig online esetre nézve, amikor a munkák hosszának L felső korlátja előre ismert, és $1 < L \leq 3$).*

4.5. A korábbinál hatékonyabb félig online algoritmus ismert legnagyobb munkaméret esetére

Ebben a fejezetben továbbra is feltesszük hogy a munkák egyesével érkeznek, viszont előre tudjuk, hogy a munkák hosszának maximuma L (biztosan van L hosszú munka). Amint az előbbi fejezetben láttuk, $L \leq 1$ esetén a $H2$ algoritmus 4/3-versenyképes optimális algoritmus, $1 < L \leq 3$ esetén pedig A_ρ és a módosított $H2$ mindegyike legfeljebb 3/2 -versenyképes. Ezért ebben a fejezetben feltesszük, hogy $L > 3$, és erre az esetre megadunk egy kétfázisú algoritmust, amely szintén 3/2 -versenyképes. Ez az algoritmus az előzőekhez képest újfajta stratégiát követ e gépek vásárlására. Emlékeztetünk arra hogy korábban az $m_0 = \lceil L \rceil \geq 3$ jelöléssel éltünk, ezt most is használni fogjuk.

$H3$ **Algoritmus:**

1. fázis

1.1 Legyen $k = 1, m = 1$.

1.2 Ütemezzük a p_k munkát minimális terhelésű gépre, ha a megnövekedett terhelés nem több mint $\frac{3}{2}L$, és menjünk az 1.4. lépésre. Egyébként menjünk az 1.3. lépésre.

1.3 Ütemezzük a p_k munkát új gépre, legyen $m = m+1$. Amennyiben $m = m_0$, menjünk a második fázis 2.6 lépésére, egyébként menjünk az 1.4. lépésre.

1.4 Legyen $k = k + 1$, ha $k > n$, stop. Egyébként menjünk az 1.2. lépésre.

2. fázis

2.1 Számoljuk ki $C_{1,k} = 2\sqrt{P_k}$ aktuális értékét, valamint az s_{k-1} minimális aktuális terhelést.

2.2 Amennyiben $s_{k-1}+p_k$ nem nagyobb mint az aktuális teljes átfutási idő, ütemezzük a p_k munkát minimális terhelésű gépre, és menjünk a 2.6. lépésre.

2.3 Ha $P_k > m^2$, ütemezzük p_k-t új gépre, legyen $m = m+1$, és menjünk a 2.6. lépésre.

2.4 Ha $s_{k-1} + p_k + m > \frac{3}{2}C_{1,k}$, ütemezzük p_k-t új gépre, legyen $m = m + 1$, és menjünk a 2.6. lépésre.

2.5 Egyébként ütemezzük a p_k munkát minimális terhelésű gépre, és menjünk a 2.6. lépésre.

2.6 Legyen $k = k + 1$. Ha $k > n$, stop. Egyébként menjünk újra a 2.1. lépésre.

4.13. Tétel *A H3 algoritmus versenyképességi aránya legfeljebb* 3/2.

A tételt egy sor lemma bizonyításán keresztül látjuk be. Először az algoritmus első fázisára koncentrálunk.

4.14. Lemma Minden $m \leq m_0$ *esetén teljesül a* $P_{i_m} > \frac{3}{4}mL$ *egyenlőtlenség.*

Bizonyítás: Nyilvánvaló, hogy $P_{i_m} = \sum_{l=1}^{m} s_{l,i_m}$. Az első fázis szabálya folytán teljesül $s_{l,i_m} + s_{j,i_m} \geq s_{i_m} + p_{i_j} > \frac{3}{2}L$ minden $1 \leq l < j \leq m$ esetén. Ezen egyenlőtlenségeket összeadva adódik a kívánt $P_{i_m} > \frac{3}{4}mL$ egyenlőtlenség. □

4.15. Lemma *Ha H3 futása befejeződik az első fázis során, a versenyképességi arány nem nagyobb mint* 3/2.

Bizonyítás: Tegyük fel, hogy az első fázis során összesen $m' \leq m_0$ gépet vásárolunk a $H3$ algoritmus során. Ekkor a 4.14. Lemma alapján $P \geq P_{i_{m'}} > \frac{3}{4}m'L$ teljesül. Ha $P \leq L^2$, akkor a 4.1. Lemma miatt kapjuk, hogy

$$\begin{aligned} \frac{3}{2}OPT(I) \geq \frac{3}{2}C_{2,n} = \frac{3}{2}\left(L + \frac{P}{L}\right) > \frac{3}{2}\left(L + \frac{3}{4}m'\right) = \frac{3}{2}L + \frac{9}{8}m' \qquad (37) \\ > C_{\max} + m' = H3(I). \end{aligned}$$

Ellenkező esetben, ha $P > L^2$, akkor megint a 4.1. Lemma miatt adódik

$$\frac{3}{2}OPT(I) \geq \frac{3}{2}C_{1,n} \geq 3\sqrt{L^2} = \frac{3}{2}L + \frac{3}{2}L \geq C_{\max} + m' = H3(I), \qquad (38)$$

ahol az utolsó egyenlőtlenség $m' \leq m_0 = \lceil L \rceil$ és $L > 3$ következménye. □

A továbbiakban a második fázisra koncentrálunk. Belátjuk hogy minden $k = i_{m_0}, \cdots, n$ esetén teljesül $z_k \leq \frac{3}{2}C_{1,k}$, amiből következik, hogy $H3$ versenyképességi aránya nem rosszabb mint 3/2. Megjegyezzük, hogy a második fázis elején éppen m_0 a gépek aktuális száma, pontosan egy munka, mégpedig $p_{i_{m_0}}$ van ütemezve az m_0-adik gépre, az eddigi munkák összmérete $\frac{3}{4}m_0L$ és $\frac{3}{2}m_0L$ közötti érték, és a teljes átfutási idő pillanatnyi értéke nem több mint $\frac{3}{2}L$.

4.16. Lemma $z_{i_{m_0}} \leq \frac{3}{2}C_{1,i_{m_0}}$.

Bizonyítás: A 4.14. Lemma alapján teljesül $P_{i_{m_0}} \geq \frac{3}{4}m_0L$. Mivel $z_{i_{m_0}} \leq m_0 + \frac{3}{2}L$ továbbá $C_{1,i_{m_0}} = 2\sqrt{P_{i_{m_0}}}$, elegendő belátnunk azt hogy $3\sqrt{\frac{3}{4}m_0L} \geq \frac{3}{2}L + m_0$,

vagyis a $9\left(\frac{L}{m_0}\right)^2 - 15\left(\frac{L}{m_0}\right) + 4 \leq 0$ egyenlőtlenséget. Mivel $L > 3$, teljesül $\frac{1}{3} \leq \frac{L}{m_0} \leq \frac{4}{3}$, amiből következik az előbbi egyenlőtlenség. □

Vegyük észre, hogy a második fázis során a versenyképességi arány csak úgy válhatna 3/2-nél rosszabbá, ha éppen egy új gépet veszünk a 2.3 vagy 2.4 lépések egyike által, és az új munka erre az új gépre kerül. Ezért ha $z_{k-1} \leq \frac{3}{2}C_{1,k-1}$ teljesül és a p_k munkát a 2.2 és 2.5 lépések egyike által ütemezzük, akkor $z_k \leq \frac{3}{2}C_{1,k}$ is teljesül. Ezért a következőkben, főként a 2.3 és 2.4 lépésekkel foglalkozunk, más szóval minden $m \geq m_0 + 1$ gépszám esetén a p_{i_m} munka ütemezését vizsgáljuk, és megbecsüljük a z_{i_m} értéket.

4.17. Lemma *Ha a 2. fázis során $z_k \leq \frac{3}{2}C_{1,k}$ teljesül, akkor a pillanatnyi teljes átfutási idő legfeljebb $C_k \leq 2m$, ahol m az aktuális gépszám.*

Bizonyítás: Nyilvánvaló, hogy a második fázis során egy munka hossza sem lehet több mint $2m$. Emiatt ha a teljes átfutási idő $2m$-nél nagyobbra növekszik, ez csak a 2.5. lépés során történhet. Azonban ebben az esetben teljesül $P_k \leq m^2$, és így $z_k = C_k + m \leq \frac{3}{2}C_{1,k} \leq 3m$. Következik, hogy $C_k \leq 2m$. □

4.18. Lemma *Ha az m-edik gépet az algoritmus a 2.3 lépésben vásárolja, ahol $m \geq m_0 + 1$, továbbá $z_{i_m-1} \leq \frac{3}{2}C_{1,i_m-1}$, akkor $z_{i_m} \leq \frac{3}{2}C_{1,i_m}$ teljesül.*

Bizonyítás: Mivel az m-edik gép a 2.3. lépésben lett vásárolva, (vagyis a p_{i_m} munkát nem a 2.2. lépésben ütemezzük), emiatt van olyan $\beta > 1$ amelyre $P_{i_m} = \beta(m-1)^2$, és

$$C_{i_m-1} < s_{i_m-1} + p_{i_m} \leq s_{i_m-1} + L \leq s_{i_m-1} + m - 1. \tag{39}$$

A 4.17. Lemma miatt kapjuk hogy $P_{i_m} \leq (m-1)C_{i_m-1} + p_{i_m} \leq 2(m-1)^2 + (m-1) \leq 3(m-1)^2$, és így $\beta \leq 3$. Most megbecsüljük a C_{i_m-1} konstans értékét: Amennyiben $C_{i_m-1} > 3\sqrt{\beta}(m-1) - m$, akkor következik (39)-ből, hogy $s_{i_m-1} > 3\sqrt{\beta}(m-1) - m - (m-1)$. Emiatt

$$\begin{aligned} P_{i_m} &= \sum_{j=1}^{m-1} s_{j,i_m-1} + p_{i_m} \geq (m-2)\, s_{i_m-1} + C_{i_m-1} \\ &> (m-1)\left(3\sqrt{\beta}(m-1) - m\right) - (m-2)\,(m-1) \\ &= (m-1)^2(3\sqrt{\beta} - 2) > \beta(m-1)^2 = P_{i_m}, \end{aligned} \tag{40}$$

ahol az utolsó egyenlőtlenség $1 < \beta \leq 3$ miatt teljesül. Ellentmondást kaptunk, így viszont $C_{i_m-1} \leq 3\sqrt{\beta}(m-1) - m$ teljesül. Emiatt pedig kapjuk, hogy $z_{i_m} = C_{i_m-1} + m \leq 3\sqrt{\beta}(m-1) = 3\sqrt{P_{i_m}} = \frac{3}{2}C_{1,i_m}$, vagy pedig $z_{i_m} = p_{i_m} + m \leq 2m - 1 < 3(m-1) < 3\sqrt{\beta}(m-1) = \frac{3}{2}C_{1,i_m}$. □

4.19. Lemma *Ha az $m-1$-edik gép vásárlása a 2.3 lépésben történik, ahol $m > m_0+1$, akkor az m-edik gép vásárlása is. Következik, hogy ezután $H3$ nem hajtja végre a 2.4 lépést, más szóval az összes további gépek mindegyikének a vásárlása szintén a 2.3. lépésben történik.*

Bizonyítás: Ha $P_{i_m} > (m-1)^2$, vagyis az m-edik gép megvásárlása előtti pillanatban a munkák összhossza nagyobb mint $(m-1)^2$, akkor az algoritmusnak a 2.3. lépést kell végrehajtania, és az állításunk teljesül. Tegyük tehát fel, hogy $P_{i_m-1} \leq P_{i_m} \leq (m-1)^2$. Mivel az $m-1$-edik gép vásárlása a 2.3. lépés által történik, az algoritmus szabálya miatt teljesül $P_{i_m-1} \geq P_{i_{m-1}} > (m-2)^2$.

Annak belátásához, hogy az m-edik gép vásárlása is a 2.3. lépésben történik, igazoljuk, hogy $s_{i_m-1} + p_{i_m} + m - 1 \leq \frac{3}{2}C_{1,i_m}$. Mivel a terhelések átlagát véve adódik $s_{i_m-1} \leq \frac{P_{i_m-1}}{m-1}$, valamint $C_{1,i_m} = 2\sqrt{P_{i_m-1}+p_{i_m}}$ is teljesül, elég belátnunk azt hogy

$$\frac{P_{i_m-1}}{m-1} + p_{i_m} + m - 1 - 3\sqrt{P_{i_m-1}+p_{i_m}} \leq 0. \tag{41}$$

A bal oldal P_{i_m-1} szerinti deriváltja $\frac{1}{m-1} - \frac{3}{2\sqrt{P_{i_m-1}+p_{i_m}}}$. Mivel $P_{i_m-1} \leq (m-1)^2$ és $p_{i_m} \leq L \leq m-2$, következik, hogy ez a derivált negatív értékű. Ezért $P_{i_m-1} > (m-2)^2$ miatt elég belátnunk azt hogy

$$\frac{(m-2)^2}{m-1} + p_{i_m} + m - 1 - 3\sqrt{(m-2)^2+p_{i_m}} \leq 0, \tag{42}$$

vagyis ami ezzel ekvivalens, a következő egyenlőtlenséget:

$$2m - 4 + \frac{1}{m-1} + p_{i_m} - 3\sqrt{(m-2)^2+p_{i_m}} \leq 0. \tag{43}$$

A bal oldal p_{i_m} szerinti deriváltja $1 - \frac{3}{2\sqrt{(m-2)^2+p_{i_m}}} > 0$ mivel $m > m_0+1 \geq 5$. Felhasználva még, hogy $p_{i_m} \leq m-2$, $\frac{1}{m-1} < 1$, és $3m-5-3\sqrt{(m-2)^2+m-2} \leq 0$, következik, hogy (43) teljesül, és ezzel a lemmát beláttuk. □

4.20. Megjegyzés *Az előző két lemma alapján következik, hogy ha valamikor a második fázis során egy gép vásárlása a 2.3. lépésben történik, és eddig az időpontig $H3$ versenyképességi aránya nem nagyobb mint $3/2$, akkor ezután a versenyképességi arány már nem lehet rosszabb mint $3/2$. (Például, ha már az m_0+1-edik gép vásárlása is a 2.3. lépés által történik, akkor végrehajtva a $H3$ algoritmust annak versenyképességi aránya legfeljebb $3/2$ marad.) Ezért a következőkben feltehetjük, hogy minden $m \geq m_0+1$ esetén $P_{i_m} \leq (m-1)^2$ teljesül.*

4.21. Lemma $s_{i_{m_0+1}-1} > \frac{m_0}{2}$.

Bizonyítás: Ellenkező esetben $s_{i_{m_0+1}-1}+p_{i_{m_0+1}}+m_0 \leq \frac{3}{2}m_0+L$ teljesül. A 4.16. Lemma bizonyításához hasonlóan kapjuk, hogy $\frac{3}{2}m_0+L \leq 3\sqrt{\frac{3}{4}Lm_0} \leq \frac{3}{2}C_{1,s_{i_{m_0+1}}}$. Ekkor viszont az m_0+1-edik gép vásárlása a 2.3. lépésben történt, és a 4.20. Megjegyzést is felhasználva készen vagyunk. □

4.22. Lemma *Amennyiben $m_0 \leq 11$ és $P_{i_{m_0+1}} \leq m_0^2$ teljesül, akkor $s_{i_{m_0+1}-1}+p_{i_{m_0+1}}+m_0 \leq \frac{3}{2}C_{1,i_{m_0+1}}$. Következik, hogy az (m_0+1)-edik gép vásárlása a 2.3. lépésben történik.*

Bizonyítás: Ellenkező esetben ugyanis

$$s_{i_{m_0+1}-1}+p_{i_{m_0+1}}+m_0 > \frac{3}{2}C_{1,i_{m_0+1}} = 3\sqrt{P_{i_{m_0+1}}}, \tag{44}$$

teljesül, és az (m_0+1)-edik gép vásárlása a 2.4. lépésben történne. Két esetet különböztetünk meg $s_{i_{m_0+1}-1}$ lehetséges értéke szerint.

1. eset $s_{i_{m_0+1}-1} \geq \frac{3}{4}L$. Mivel $P_{i_{m_0+1}} \geq s_{i_{m_0+1}-1}m_0+p_{i_{m_0+1}}$, teljesül az

$$s_{i_{m_0+1}-1}+p_{i_{m_0+1}}+m_0-3\sqrt{s_{i_{m_0+1}-1}m_0+p_{i_{m_0+1}}} > 0 \tag{45}$$

egyenlőtlenség (44) következtében. A bal oldali függvény $p_{i_{m_0+1}}$ szerinti deriváltja $1-\frac{3}{2\sqrt{s_{i_{m_0+1}-1}m_0+p_{i_{m_0+1}}}} > 0$ mivel $m_0 \geq L > 3$ és $s_{i_{m_0+1}-1} \geq \frac{3}{4}L$. Emiatt $p_{i_{m_0+1}} \leq L$ következtében teljesül

$$s_{i_{m_0+1}-1}+L+m_0-3\sqrt{s_{i_{m_0+1}-1}m_0+L} > 0. \tag{46}$$

Most vegyük a bal oldal $s_{i_{m_0+1}-1}$ szerinti deriváltját, ami $1-\frac{3m_0}{2\sqrt{s_{i_{m_0+1}-1}m_0+L}}$. $P_{i_{m_0+1}} \leq m_0^2$ miatt $s_{i_{m_0+1}-1} \leq m_0$ teljesül. Ezt összevetve az $s_{i_{m_0+1}-1}m_0+L \leq m_0^2+m_0 < \frac{9}{4}m_0^2$ feltétellel, kapjuk hogy a derivált értéke negatív. Mivel $s_{i_{m_0+1}-1} \geq \frac{3}{4}L$ teljesül, a (46) egyenlőtlenség következtében kapjuk, hogy

$$\frac{3}{4}L+L+m_0 > 3\sqrt{\frac{3}{4}Lm_0+L}, \tag{47}$$

vagyis $(\frac{7}{4}L+m_0)^2 > (3\sqrt{\frac{3}{4}Lm_0+L})^2$. Ebből kapjuk, hogy

$$f_6(L) \doteq 49L^2-(52m_0+144)\,L+16m_0^2 > 0, \tag{48}$$

tetszőleges $L > 3$ és $m_0 = \lceil L \rceil \geq 4$ esetén.

Másrészt vizsgáljuk meg, hogy az $f_6(L)$ függvénynek milyen L esetén adódik és mennyi a maximális értéke. Könnyen látható, hogy a keresett maximum $L = m_0$ esetén adódik ha $m_0 > 4$, illetve $L = m_0-1$ esetén, amennyiben $m_0 = 4$. Emiatt

a maximum értéke $f_6(3) = 49 \cdot 3^2 - (52 \cdot 4 + 144)\, 3 + 16 \cdot 4^2 = -359 < 0$ ha $m_0 = 4$, illetve $f_6(m_0) = 49m_0^2 - (52m_0 + 144)\, m_0 + 16m_0^2 < 0$ ha $4 < m_0 \leq 11$. Ez viszont ellentmond a (48) egyenlőtlenségnek.

2. eset. $s_{i_{m_0+1}-1} < \frac{3}{4}L$. Ekkor a (44) feltételből és a 4.14. Lemmából kapjuk hogy

$$s_{i_{m_0+1}-1} + p_{i_{m_0+1}} + m_0 - 3\sqrt{\frac{3}{4}Lm_0 + p_{i_{m_0+1}}} > 0. \tag{49}$$

Megint vegyük a bal oldal $p_{i_{m_0}+1}$ szerinti deriváltját, ami most pozitív. Emiatt $p_{i_{m_0}+1} \leq L$ következtében kapjuk hogy

$$s_{i_{m_0+1}-1} + L + m_0 - 3\sqrt{\frac{3}{4}Lm_0 + L} > 0. \tag{50}$$

Helyettesítsük be az $s_{i_{m_0+1}-1} < \frac{3}{4}L$ egyenlőtlenséget az előbbibe, és újra (46) adódik. Ezek után a bizonyítás hasonlóképpen befejezhető. □

Az előző lemma alapján már tudjuk, hogy ha $H3$ valamikor a 2.4. lépést hajtja végre, az csak úgy lehet, ha $m_0 \geq 12$ teljesül, (valamint az is, hogy $P_{i_m} \leq (m-1)^2$ minden $m \geq m_0+1$ esetén, amint ezt már beláttuk korábban). Emiatt mostantól feltesszük, hogy $m_0 \geq 12$. Innentől bizonyításunkat indirekt módon folytatjuk. Tegyük fel, hogy $H3$ rosszabb mint 3/2-versenyképes, ekkor léteznie kell olyan m-nek, amelyre $m_0 + 1 \leq m$, és az m-edik gép vásárlása a 2.4. lépésben történik, továbbá $H3$ éppen ezáltal lesz rosszabb mint 3/2-versenyképes, hogy p_{i_m} ezen új gépre kerül. A következő lemmák segítségével belátjuk, hogy ilyen m nem létezik. A 4.23-4.26 Lemmákban az $m_0 \leq m \leq 2m_0$ esetttel, a 4.27-4.28. Lemmákban pedig az $m > 2m_0$ esettel foglalkozunk.

4.23. Lemma *Minden $m_0 \leq m \leq 2m_0$ esetén teljesül $p_{i_{m+1}} > s_{i_{m+1}-1}$.*

Bizonyítás: Először lássuk be az állítást $m = m_0$ esetén. Tegyük fel indirekt módon az állítás ellenkezőjét, vagyis azt hogy $p_{i_{m_0+1}} \leq s_{i_{m_0+1}-1}$. Belátjuk, hogy $s_{i_{m_0+1}-1} + p_{i_{m_0+1}} + m_0 \leq 3\sqrt{P_{i_{m_0+1}}}$, amiből következik, hogy az (m_0+1)-edik gép vásárlása a 2.3. lépésben történik. Nyilván teljesül $P_{i_{m_0+1}} \geq m_0 s_{i_{m_0+1}-1}$. Emiatt elég belátnunk hogy $2s_{i_{m_0+1}-1} + m_0 \leq 3\sqrt{m_0 s_{i_{m_0+1}-1}}$, ami ekvivalens a következővel: $\frac{1}{4} \leq \frac{s_{i_{m_0+1}-1}}{m_0} \leq 1$, ez viszont nyilvánvalóan teljesül, hiszen $s_{i_{m_0+1}-1} > \frac{m_0}{2}$ a 4.21. Lemma és $s_{i_{m_0+1}-1} \leq \frac{P_{i_{m_0+1}-1}}{m_0} \leq m_0$ folytán. Beláttuk tehát hogy $p_{i_{m_0+1}} > s_{i_{m_0+1}-1}$.

Mivel az aktuális minimális terhelés nem csökken valamely új gép vásárlását közvetlenül megelőzően, illetve közvetlenül ez után, következik hogy $s_{i_{m_0+2}-1} \geq \min\{p_{i_{m_0+1}}, s_{i_{m_0+1}-1}\} > \frac{m_0}{2}$. Másrészt pedig teljesül $s_{i_{m_0+2}-1} \leq \frac{P_{i_{m_0+2}-1}}{m_0+1} \leq m_0+1$, és így $\frac{1}{4} \leq \frac{s_{i_{m_0+2}-1}}{m_0+1} \leq 1$. Ezután az előbbihez hasonló gondolatmenettel látható,

hogy $p_{i_{m_0+2}} > s_{i_{m_0+2}-1}$. A további $m = m_0 + 2, \cdots, 2m_0$ gépszámok esetén hasonlóan látható be az állítás. □

4.24. Megjegyzés *Az előző lemma következtében tudjuk már, hogy amikor a gépek száma m_0 és $2m_0$ között van, az aktuális minimális terhelés értéke nem csökkent. Más szóval, $s_{i_{m+1}-1} \geq s_{i_{m_0+1}-1}$ teljesül minden $m_0 + 1 \leq m \leq 2m_0$ esetén. Továbbá amennyiben $s_{i_{m_0+1}-1} \geq \frac{3}{4}L$ teljesül, akkor $p_{i_{m+1}} \geq s_{i_{m_0+1}-1} \geq \frac{3}{4}L$ is teljesül minden $m_0 \leq m \leq 2m_0$ esetén, és így a 4.14. Lemma folytán adódik, hogy $P_{i_m} \geq P_{i_{m_0}} + p_{i_{m_0+1}} + \cdots + p_{i_m} \geq \frac{3}{4}Lm_0 + \frac{3}{4}L(m - m_0) = \frac{3}{4}Lm$ teljesül minden $m_0 \leq m \leq 2m_0$ esetén.*

4.25. Lemma *Minden $m_0 \leq m \leq 2m_0$ esetén teljesül $p_{i_{m+1}} \geq \frac{3}{4}L$, és így $P_{i_m} \geq \frac{3}{4}Lm$.*

Bizonyítás: Tegyük fel, hogy $p_{i_{m_0+1}} < \frac{3}{4}L$. A 4.24. Megjegyzés miatt elég csak az $s_{i_{m_0+1}-1} < \frac{3}{4}L$ esettel foglalkoznunk, és elég azt belátnunk, hogy $s_{i_{m_0+1}-1} + p_{i_{m_0+1}} + m_0 \leq 3\sqrt{P_{i_{m_0+1}}}$, mert ebből következik hogy az (m_0+1)-edik gép vásárlása a 2.3 lépés által történik, és így a 4.20. Megjegyzés miatt készen is vagyunk. Mivel $s_{i_{m_0+1}-1} < \frac{3}{4}L$, elég azt belátnunk, hogy $\frac{3}{2}L + m_0 \leq 3\sqrt{\frac{3}{4}Lm_0}$. Ez ekvivalens átalakítással $\frac{1}{3} \leq \frac{L}{m_0} \leq \frac{4}{3}$ alakra hozható, amely viszont nyilvánvalóan teljesül. Így tehát kapjuk hogy $p_{i_{m_0+1}} \geq \frac{3}{4}L$ és emiatt a 4.24. Megjegyzés okfejtését követve kapjuk hogy $P_{i_{m_0+1}} \geq \frac{3}{4}L(m_0 + 1)$. Az $m = m_0 + 1, \cdots, 2m_0$ gépszámokra az állítás hasonlóképpen látható be egyenként. □

4.26. Lemma *Ha valamely $m_0 \leq m \leq 2m_0$ esetén az $(m + 1)$-edik gép vásárlása a 2.4 lépésben történik, továbbá $z_{i_{m+1}-1} \leq \frac{3}{2}C_{1,i_{m+1}-1}$ teljesül, akkor $z_{i_{m+1}} \leq \frac{3}{2}C_{1,i_{m+1}}$.*

Bizonyítás: Ha $p_{i_{m+1}} \geq \max\{s_{j,i_{m+1}-1} |\ j = 1, \cdots, m\}$ teljesül, akkor könnyen láthatóan $z_{i_{m+1}} = p_{i_{m+1}} + m + 1 \leq L + m + 1 \leq 3\sqrt{\frac{3}{4}Lm} \leq \frac{3}{2}C_{1,i_{m+1}}$, ahol az utolsó egyenlőtlenség a 4.25. Lemma következménye. Ellenkező esetben a célfüggvény 1-gyel növekszik, hiszen a teljes átfutási idő változatlan marad. Mivel $z_{i_{m+1}-1} \leq \frac{3}{2}C_{1,i_{m+1}-1}$ teljesül, elég belátnunk azt hogy $3\sqrt{P_{i_{m+1}-1} + p_{i_{m+1}}} \geq 3\sqrt{P_{i_{m+1}-1}} + 1$, vagyis ekvivalens alakban

$$p_{i_{m+1}} \geq \frac{2}{3}\sqrt{P_{i_{m+1}-1}} + \frac{1}{9}. \tag{51}$$

Legyen $P_{i_{m+1}-1} = \beta m^2$ valamely $\beta \leq 1$ esetén. Mivel $\frac{L}{m} > \frac{m_0-1}{2m_0} \geq \frac{1}{2} - \frac{1}{24} > \frac{4}{9}$, felhasználva a 4.25. Lemmát is, adódik hogy $P_{i_{m+1}-1} \geq P_{i_m} \geq \frac{3}{4}Lm > \frac{1}{3}m^2$, és emiatt $\frac{1}{3} < \beta \leq 1$. Tudjuk hogy az $(m + 1)$-edik gép vásárlása a 2.4 lépésben történik, emiatt

$$s_{i_{m+1}-1} + p_{i_{m+1}} + m > 3\sqrt{P_{i_{m+1}-1}} = 3\sqrt{\beta}m. \tag{52}$$

Alkalmazva az $s_{i_{m+1}-1} \leq \frac{P_{i_{m+1}-1}}{m} = \beta m$ egyenlőtlenséget, adódik hogy $p_{i_{m+1}} > \left(3\sqrt{\beta} - \beta - 1\right) m$. Emiatt (51) teljesül, ha be tudjuk látni az alábbi egyenlőtlenséget:

$$\left(3\sqrt{\beta} - \beta - 1\right) m > \frac{2}{3}\sqrt{\beta}m + \frac{1}{9}, \tag{53}$$

ami viszont ekvivalens alakban a következő: $\left(\frac{7}{3}\sqrt{\beta} - \beta - 1\right) m > \frac{1}{9}$. Ez viszont nyilvánvalóan teljesül, mivel $\frac{1}{3} < \beta \leq 1$ és $m \geq 12$. □

A 4.23-4.26. Lemmák alapján már beláttuk, hogy $H3$ versenyképességi aránya nem lehet több mint 3/2 a $(2m_0 + 1)$-edik gép vásárlása előtt.

4.27. Lemma *Legyen $m_0 \leq m \leq 2m_0$, ekkor teljesül $p_{i_{m+1}} > y$, ahol*

$$y = 3\sqrt{\frac{3}{4}Lm_0} - m_0 - \frac{3}{4}L.$$

Bizonyítás: Az algoritmus szabálya szerint nyilvánvaló, hogy minden $m_0 \leq m \leq 2m_0$ esetén teljesül $s_{i_{m+1}-1} + p_{i_{m+1}} + m > \frac{3}{2}C_{1,i_{m+1}} \geq 3\sqrt{s_{i_{m+1}-1}m}$, hiszen egyébként az m-edik gép vásárlása a 2.3. lépésben történt volna. Emiatt

$$p_{i_{m+1}} > 3\sqrt{s_{i_{m+1}-1}m} - s_{i_{m+1}-1} - m.$$

Definiáljuk a következő függvényt: $f(s_t, m) = 3\sqrt{s_t m} - s_t - m$. Ez szimmetrikus, és mindkét változójában növekvő, hiszen most $s_t < m$ teljesül. Emlékezzünk vissza, hogy tetszőleges $m_0 \leq m_1 < m_2 \leq 2m_0$ esetén $s_{i_{m_1}} \leq s_{i_{m_2}}$ teljesül a 4.24. Megjegyzés alapján. Két esetet különböztetünk meg az alábbiak szerint:

a, Ha $s_{i_{m+1}-1} \geq \frac{3}{4}L$, akkor kapjuk hogy

$$\begin{aligned} p_{i_{m+1}} &> 3\sqrt{s_{i_{m+1}-1}m} - s_{i_{m+1}-1} - m \geq 3\sqrt{s_{i_{m+1}-1}m_0} - s_{i_{m+1}-1} - m_0 \\ &\geq 3\sqrt{s_{i_{m_0+1}-1}m_0} - s_{i_{m_0+1}-1} - m_0 \geq 3\sqrt{\frac{3}{4}Lm_0} - \frac{3}{4}L - m_0 = y. \end{aligned}$$

b, Ha pedig $s_{i_{m+1}-1} < \frac{3}{4}L$, akkor a 4.25. Lemma és az algoritmus szabályából következően $p_{i_{m+1}} + s_{i_{m+1}-1} + m > \frac{3}{2}C_{1,i_{m+1}} \geq 3\sqrt{\frac{3}{4}Lm}$. Emiatt pedig az előzőhöz hasonlóan kapjuk hogy

$$\begin{aligned} p_{i_{m+1}} &> 3\sqrt{\frac{3}{4}Lm} - s_{i_{m+1}-1} - m \\ &> 3\sqrt{\frac{3}{4}Lm} - \frac{3}{4}L - m \geq 3\sqrt{\frac{3}{4}Lm_0} - \frac{3}{4}L - m_0 = y. \end{aligned}$$

□

4.28. Lemma *A $(2m_0+1)$-edik gép vásárlása nem a 2.4.lépésben történik.*

Bizonyítás: Először tegyük fel hogy $s_{i_{2m_0+1}-1} \geq \frac{3}{4}L$. A 4.27. Lemma alapján kapjuk hogy $P_{i_{2m_0+1}} \geq m_0 s_{i_{2m_0+1}-1} + p_{i_{m_0+1}} + \cdots + p_{i_{2m_0}} > m_0 s_{i_{2m_0+1}-1} + m_0 y = m_0\left(s_{i_{2m_0+1}-1} + y\right)$, ahol az $y = 3\sqrt{\frac{3}{4}Lm_0} - \frac{3}{4}L - m_0$ jelöléssel éltünk. Ahhoz hogy a lemma állítását belássuk, felhasználva a $p_{i_{2m_0+1}} \leq L$ feltételt, elég megmutatnunk hogy

$$L + s_{i_{2m_0+1}-1} + 2m_0 < 3\sqrt{m_0\left(s_{i_{2m_0+1}-1} + y\right)}. \tag{54}$$

A korábbi okfejtésekhez hasonlóan elegendő megmutatnunk a

$$\frac{7}{4}L + 2m_0 < 3\sqrt{3m_0\sqrt{\frac{3}{4}Lm_0} - m_0^2}, \tag{55}$$

egyenlőtlenség teljesülését, ami könnyen igazolható, hiszen $L \leq m_0$ és $m_0 \geq 12$.

Most lássuk az ellenkező esetet: $s_{i_{2m_0+1}-1} < \frac{3}{4}L$, ekkor teljesül $P_{i_{2m_0+1}} \geq P_{i_{2m_0}} \geq \frac{3}{4}Lm_0 + m_0 y = 3m_0\sqrt{\frac{3}{4}Lm_0} - m_0^2$. Ezért azt kell belátnunk, hogy

$$L + s_{i_{2m_0+1}-1} + 2m_0 < 3\sqrt{3m_0\sqrt{\frac{3}{4}Lm_0} - m_0^2}. \tag{56}$$

Mivel $s_{i_{2m_0+1}-1} < \frac{3}{4}L$, elég, ha belátjuk az erősebb (55) egyenlőtlenséget, ami az előzőhöz hasonlóképpen elvégezhető. □

Ezek után megállapíthatjuk tehát, hogy a 4.28. és 4.26. Lemmák valamint a 4.20. Megjegyzés alapján az összes esetet megvizsgáltuk, és ezzel beláttuk, hogy a $H3$ algoritmus versenyképességi aránya nem nagyobb mint 3/2, vagyis a 4.13. Tétel bizonyítása befejeződött. □

5. A gépköltséges feladat elutasításos modellje

A fejezet tartalma a [15] cikkekben jelent meg. A fejezetbeli algoritmus, és annak vizsgálata jelen értekezés szerzőjének alkotásai.

A klasszikus párhuzamos gépek ütemezése feladatnak egy olyan variánsával foglalkozunk, amikor két különleges feltétel is van, az egyik, hogy lehetőségünk van új gépek vásárlására, másrészt pedig a munkák ütemezését elutasíthatjuk, ez esetben bizonyos nagyságú büntetést kell fizetnünk. Tehát ebben a fejezetben az Imreh Csanád és John Noga által bevezetett gépköltséges feladatot [48, 12], és a munkák elutasításának lehetőségét [7] ötvözzük.

Bármelyik J_i munkát két paraméter jellemzi, egyrészt $w_i \geq 0$ a mérete, és $p_i \geq 0$ az a büntetés, amit elutasítása esetén fizetni kell. Kezdetben nem állnak még rendelkezésünkre gépek, viszont az algoritmus végrehajtása során bármikor van lehetőségünk új gép vásárlására. Mivel a gépek egyformák, bármely gép vásárlásának költsége 1 egység, (ha nem így lenne, ez a munkák hosszának illetve a gépköltség normalizálásával elérhető). Bármelyik munkát elutasíthatjuk, ez esetben kifizetjük érte a büntetést, vagy ütemezhetjük a meglévő gépek valamelyikére, vagy pedig vásárolhatunk egy új gépet, és az új gépre is ütemezhetjük a munkát. A munkák végrehajtását nem szabad megszakítani, sem átlapolni: amelyik munka végrehajtsát megkezdi egy gép, azt be is kell hogy fejezze. A célfüggvény a következő: A megvásárolt gépek költségeinek összege, plusz a teljes átfutási idő, plusz az elutasított munkákért kifizetett büntetések összege, ennek a célfüggvénynek keressük a minimális értékét. A feladatra röviden MCR feladatként hivatkozunk (*scheduling with machine cost and rejection penalties*).

A tiszta online esetre Imreh Csanád és Nagy-György János [50] cikke megad egy 2.618-versenyképes algoritmust, az általános esetben optimális algoritmus nem ismert. Ebben a fejezetben csak azzal a speciális változattal foglalkozunk, (*kis munkák esete*) amikor feltesszük hogy minden munka végrehajtási ideje legfeljebb 1. Ez a feltétel gyakran teljesülhet a gyakorlatban: a munka végrehajtási ideje általában nem növelheti annyival a célfüggvényt mint egy gép vásárlási költsége. Ezzel a kis munkák esetével foglalkoztunk már az előző fejezetben, ez a félig online feltétel először a [12] cikkben szerepel.

Jegyezzük meg, hogy a feladat ezen megszorítás mellett is általánosítása az úgynevezett sí-kölcsönzési (SRP) feladatnak, e tény felismerése Leah Epsteintől származik. Az SRP feladat esetén ha megvesszük a síléceket, akkor 1 egység pénzt fizetünk ezért, ha kölcsönözzük, ez $p < 1$ forintba kerül. Nem tudjuk előre, hogy hányszor fogunk kölcsönözni, ha megvesszük, onnantól akárhányszor, újabb fizetés nélkül használhatjuk. A feladat: döntsük el, hogy mikor vásároljuk meg a síléceket úgy, hogy az összes költség (a sílécek ára plusz a már kifizetett kölcsönzési díjak összege minimális legyen. Tekintsük a munkák következő sorozatát. Minden munka esetén $w_i = 0,\ p_i = p$. Ekkor feladatunk ekvivalens az SRP feladattal. Ismert, hogy az SRP feladat optimális versenyképességi aránya 2. Most ebben a

fejezetben szintén megadunk egy optimális algoritmust, 2 versenyképességi aránnyal, az előbbiek szerint tehát ez egyben optimális algoritmus.

Az algoritmus vizsgálatakor egy újszerű módszerrel élünk: Ha az algoritmus nem lenne 2-versenyképes, akkor lennie kell (elemszám tekintetében) minimális ellenpéldának. Ez azonban rendkívül bonyolult szerkezetű feladathalmaz is lehet, (mint látni fogjuk, az algoritmus sem teljesen triviális). Emiatt az ellenpéldát fokozatosan kicseréljük újabb és újabb, de szintén minimális ellenpéldákkal, az utoljára kapott feladathatmazról pedig már viszonylag könnyen látjuk, hogy nem is ellenpélda, és így indirekt bizonyításunk véget ér.

Az alábbiakban bevezetünk egy egyszerű kétfázisú online algoritmust. Az első fázis során csak arra ügyelünk, hogy az első gépet ne vásároljuk meg túlságosan elhamarkodottan. A második fázis során pedig arra vigyázunk, hogy az elutasított munkák összbüntetése, vagy pedig a teljes átfutási idő és a vásárolt gépek számának összege ne lehessen túlságosan nagy. Ennek érdekében egy fokozatosan növelt, *par* elnevezésű paraméter segítségével döntjük el, hogy a következő munkát elfogadjuk-e, vagy vessük el: $\frac{w_k}{par} < p_k$ esetén elfogadjuk a következő munkát, $\frac{w_k}{par} > p_k$ esetén pedig elvetjük. Megjegyezzük, hogy az algoritmusunk nem determinisztikus abban az értelemben, hogy $\frac{w_k}{par} = p_k$ esetén a munkát tetszés szerint elutasíthatjuk, vagy elfogadhatjuk. Erre azért van szükség, hogy az algoritmus hatékonyságának vizsgálata egyszerűbb legyen. Ahogy az elfogadott munkák összmérete, illetve az elutasított munkák összbüntetése növekszik, *par* értékét bizonyos alkalmakkor eggyel megnöveljük. Az algoritmus formális leírása előtt bevezetünk néhány, a későbbiekben hasznos jelölést:

Legyen $W(I) = \sum_{J_i \in I} w_i$ és $P(I) = \sum_{J_i \in I} p_i$ tetszőleges I feladathalmaz esetén. Jelölje $\mathcal{J}_k = \{J_1, J_2, \cdots, J_k\}$ az első k számú munkát. Legyenek S_k illetve R_k azon részei a $\mathcal{J}_k$ halmaznak, amelyeket elfogad, illetve elutasít valamely online ütemezési algoritmus. Ekkor nyilván teljesülnek a $\mathcal{J}_k = S_k \cup R_k$ illetve $S_k \cap R_k = \emptyset$ feltételek. Az egyszerűség kedvéért legyen $W_k = W(S_k)$ (az első k munka közül az elfogadottak összmérete), és $P_k = P(R_k)$ (az első k munka közül az elutasítottak összbüntetése). Használjuk a $W_0 = P_0 = 0$ jelöléseket is. Egy online algoritmus által vásárolt gépek végső számát jelölje m, és legyen m' a vásárolt gépek aktuális száma az algoritmus végrehajtása során. Továbbá valamely J_i munka jelölésére fogjuk használni a (w_i, p_i) jelölést is.

Az Óvatos algoritmus

1. $k = 0$.
2. Ha nincs több munka, stop. Ellenkező esetben legyen $k = k + 1$. Ha $P_{k-1} + p_k \leq 1$, elutasítjuk a k-adik munkát, és visszatérünk a 2. lépésre.
3. Legyen $m' = 0$, és $par = 1$.

4. A J_k munkát $\frac{w_k}{par} < p_k$ esetén legyen elfogadjuk, $\frac{w_k}{par} > p_k$ esetén elutasítjuk; $\frac{w_k}{par} = p_k$ esetén J_k tetszőlegesen elutasítható, vagy elfogadható.

5. Ha J_k-t elfogadtuk, és $W_k < m'(m'+1)$ teljesül, ütemezzük a munkát a már meglévő gépek valamelyikére az LS algoritmus által, és menjünk a 8. lépésre.

6. Ha J_k elfogadásra került de $W_k \geq m'(m'+1)$ teljesül, vegyünk egy új gépet, ütemezzük az új munkát az újonnan vásárolt gépre. Legyen $m' = m'+1$ és $par = \max\{m', par\}$. Menjünk a 8. lépésre.

7. Ha J_k elutasításra került, és $P_k > 2par + 1 - \frac{1}{par}$, legyen $par = par + 1$ és menjünk a 8. lépésre.

8. Ha nincs több munka, stop. Egyébként legyen $k = k+1$, és menjünk a 4. lépésre.

Megjegyezzük, hogy az algoritmusnak többféle, még mindig optimális változata lehetséges. Például a második lépés a következő is lehetne:

2' Ha nincs több munka, stop. Ellenkező esetben legyen $k = k+1$. Ha teljesül $1 + w_k + P_{k-1} > 2(P_{k-1} + p_k)$, elvetjük a J_k munkát, és menjünk vissza a 2. lépésre.

A 7. lépésben is állhatna egyszerűen a $P_k > 2par$ feltétel is, ekkor azonban az algoritmus versenyképességi analízise (még) bonyolultabb lenne később. A további lehetséges változatok részletezésével itt nem foglalkozunk.

Megjegyzés *Az algoritmus "óvatos" abban az értelemben, hogy ellenkezőképpen viselkedik mint a szokásos "mohó" módszerek: Körültekintően igyekszik kizárni annak lehetőségét, hogy a versenyképességi arány túlságosan "rossz" legyen. Még egy érdekessége az algoritmusnak az, hogy nem determinisztikus, bizonyos esetekben a 4. lépésben többféleképpen is dönthetünk, és ez az algoritmus versenyképességi arányát nem befolyásolja. Az első fázis az 1. és 3. lépések közötti rész. Ebben a fázisban óvatosan elkerüljük, hogy az első gép vásárlása a versenyképességi arányt 2 fölé növelje. A második fázis a 4. és 8. lépések között van. E fázis során pedig azon igyekszünk, hogy az elutasított munkák összbüntetése, vagy pedig az elfogadott munkák miatt kapott teljes átfutási idő plusz a gépek száma ne lehessen "túl nagy szám".*

5.1. Néhány előzetes megállapítás, alsó korlátok

Nyilvánvaló, hogy optimális megoldás esetén nem lehet több a megvásárolt gépek száma mint n, (ahol n a tárgyak száma). Legyen

$$y_i = \begin{cases} 1, & \text{ha az } i\text{-edik gép megvásárlásra kerül,} \\ 0, & \text{egyébként,} \end{cases}$$

és

$$x_{ij} = \begin{cases} 1, & \text{ha a } J_j \text{ munkát az } i\text{-edik gépre ütemezzük,} \\ 0, & \text{egyébként.} \end{cases}$$

Ezekkel a jelölésekkel az MCR feladat matematikai modellje a a következő

$$\begin{array}{ll} \min & C_{\max} + \sum_{i=1}^{n} y_i + \sum_{j=1}^{n} p_j(1 - \sum_{i=1}^{n} x_{ij}) \\ \text{s.t.} & \sum_{j=1}^{n} w_j x_{ij} \leq y_i C_{\max} \quad i = 1, \cdots, n \\ & \sum_{i=1}^{n} x_{ij} \leq 1 \qquad j = 1, \cdots, n \\ & y_i = 0, 1 \qquad i = 1, \cdots, n \\ & x_{ij} = 0, 1 \qquad i, j = 1, \cdots, n \end{array}$$

Enyhítsük az $x_{ij} \in \{0, 1\}$ feltételt a szokásos módon a következővel: $0 \leq x_{ij} \leq 1$ minden $i, j = 1, \cdots, n$ esetén, ekkor kapunk egy relaxált feladatot. A relaxáció jelentése a következő: A munkák végrehajtása megszakítható, a munkák egyes részeit különböző gépekkel is szabad végezni, és az átlapolás is megengedett. (Sőt akár az is lehetséges, hogy valamely munkának valamely részét elvetjük, és kifizetjük a megfelelő arányú büntetést, a maradék részt pedig megszakítható és átlapolható módon ütemezzük, nevezzük ezt elszakításnak. Látni fogjuk, hogy elszakításra már nincs is szükség a másik két relaxáció (az elfogadott munkák megszakíthatóak, átlapolhatóak) megengedése mellett: a relaxált feladat optimális megoldása esetén minden munkáról feltételezhető, hogy teljes egészében ütemezve, vagy teljes egészében elutasítva van.) A relaxált feladatra MCRR feladatként (*problem MCR with relaxation*) hivatkozunk. Legyen I tetszőleges adathalmaz (instance), ennek esetén a relaxált feladat optimális megoldásának értékét jelölje $OPT_\rho(I)$, erre relaxált optimális megoldásként fogunk hivatkozni. Ekkor $OPT_\rho(I) \leq OPT(I)$ triviálisan teljesül, és $OPT_\rho(I) = C_\rho + m_\rho + P_\rho$, ahol C_ρ, m_ρ illetve P_ρ a relaxált optimális megoldás esetén a teljes átfutási idő, a gépek száma, illetve az elutasított munkákért kifizetett büntetés összege. Jelölje C az Óvatos algoritmus által kapott teljes átfutási időt, m pedig a vásárolt gépek számát az algoritmus végrehajtásának befejeztével.

5.1. Megjegyzés *Tegyük fel hogy a relaxált optimális gépek száma $m_\rho > 0$. Ekkor a relaxált optimális megoldás esetén minden olyan munkát amelyre $\frac{w_k}{p_k} < m_\rho$ teljesül el kell fogadni, minden olyan munkát amelyre $\frac{w_k}{p_k} > m_\rho$ teljesül el kell utasítani, azok a munkák pedig amelyekre $\frac{w_k}{p_k} = m_\rho$ teljesük szabadon elutasíthatók vagy elfogadhatók. Most látjuk tehát azt is, hogy elszakításra nincs szükség.*

5.1. Lemma *Tegyük fel, hogy a relaxált feladat optimális megoldása esetén a gépek számára teljesül* $m_\rho \geq 1$. *Ekkor* $m_\rho - 1 \leq C_\rho \leq m_\rho + 1$.

Bizonyítás: A relaxált optimális megoldás esetén a gépek terhelése egyenlő. Emiatt az elfogadott munkák összmérete pontosan $C_\rho m_\rho$. Tegyük fel, hogy $C_\rho < m_\rho - 1$, ekkor $m_\rho \geq 2$, továbbá

$$(C_\rho + 1)(m_\rho - 1) = C_\rho m_\rho + m_\rho - 1 - C_\rho > C_\rho m_\rho.$$

Ezek szerint, amennyiben az elfogadott munkák $m_\rho - 1$ gépre lennének ütemezve, akkor a teljes átfutási idő kevesebb lenne mint $C_\rho + 1$. Emiatt kevesebb gép vásárlásával a megoldás javul, ami ellentmondás. A másik egyenlőtlenség hasonlóképpen látható be. □

A következő lemmában egy egyszerűen kapható felső korlátot határozunk meg a heurisztikus megoldás teljes átfutási idejére.

5.2. Lemma *Legyen* m *a vásárolt gépek száma, valamint* C *a teljes átfutási idő az Óvatos algoritmus elvégzése után. Ha* $m = 2$, és az utolsó munka újonnan vásárolt gépre került, akkor $C \leq 2$, h*a* $m \geq 3$ *és az utolsó munka újonnan vásárolt gépre került, akkor*

$$C \leq m + 1 - \frac{1}{m-1}, \tag{57}$$

egyébként pedig

$$C \leq \frac{W_n - 1}{m} + 1. \tag{58}$$

Bizonyítás: Az elfogadott munkák száma szerinti teljes indukciót alkalmazunk. Mindegyik munka elutasítása esetén az állítás triviálisan teljesül. Tegyük fel ezért, hogy van legalább egy elfogadott munka, és a teljes átfutási idő egy olyan munka befejezési ideje, amelynek a hossza a. Ha az elfogadott munkák száma egy, (57) szintén triviálisan teljesül. Tegyük fel, hogy az állítás teljesül az első $k-1$ elfogadott munka esetén, és tekintsük a k-adik elfogadott munkát. Ha ez új gépre kerül, és $m \geq 2$, akkor a teljes átfutási idő az első $m-1$ gép valamelyikén vétetik föl, emiatt változatlan marad, másrészt az ezekre a gépekre ütemezett munkák összhossza az algoritmus szabálya szerint legfeljebb $m(m-1)$. Emiatt egyszerű átlagszámolással és az $a \leq 1$ feltételt felhasználva adódik, hogy

$$C \leq \frac{m(m-1) - a}{m-1} + a \leq \frac{m(m-1) - 1}{m-1} + 1 = m + 1 - \frac{1}{m-1}.$$

Tegyük fel, hogy a k-adik munka nem kerül új gépre. Ha a teljes átfutási idő nem növekszik, készen vagyunk, ezért tegyük fel, hogy az növekszik. Emiatt az utolsó

munka által vétetik fel a teljes átfutási idő, továbbá e munka ütemezése az LS szabály szerint történt, kapjuk ezáltal hogy

$$C \leq \frac{W_n - a}{m} + a \leq \frac{W_n - 1}{m} + 1.$$

Maradt az az eset, amikor a k-adik munka új gépre kerül, és $m = 1$. Ekkor ez az egyetlen elfogadott munka, de ezt az esetet a bizonyítás elején tisztáztuk. □

5.1. Következmény Tegyük fel, hogy $m \geq 1$. *Legyen*

$$C' = \begin{cases} m + 1 - \frac{1}{m-1}, \text{ ha } m \geq 2, \text{ és az utolsó munka új gépre kerül,} \\ \frac{W_n - 1}{m} + 1, \text{ egyébként.} \end{cases} \tag{59}$$

Ekkor C' a heurisztikus teljes átfutási időnek felső korlátja. Továbbá teljesül $C' \leq m + 2$.

Bizonyítás: A $C' \leq m + 2$ állítás azért teljesül, mert $W_n \leq m\,(m + 1)$. □

5.2. Megjegyzés: Vegyük észre, *hogy C' értéke csak a gépek számától és az elfogadott munkák W_n összméretétől függ, továbbá W_n meghatározza a vásárolt gépek számát is.*

5.2. Az Óvatos algoritmus versenyképességi analízise

5.1. Tétel *Az Óvatos algoritmus optimális.*

Bizonyítás: Legyen $UB(I) = P_n + m + C'$, ahol C' a teljes átfutási idő előbbi felső korlátja. Ekkor $UB(I)$ egy triviális felső korlát a heurisztikus algoritmus megoldásának értékére. Elég belátnunk az $UB(I) \leq 2OPT_\rho(I)$ egyenlőtlenség teljesülését. Tegyük fel az állítással ellentétben, hogy I olyan ellenpélda, amelyre $UB(I) > 2OPT_\rho(I)$ teljesül. Legyen $fpar$ a par paraméter végső értéke az algoritmus befejezése után. Ekkor nyilván teljesül $fpar \geq m$. Legyen F az a részhalmaza I-nek, amely munkákat az algoritmus a 2. lépés során utasít el. Bevezetjük az $r = P(F)$ jelölést. Jelölje I_0 az algoritmus második fázisában érkező munkákat. I_0 azon részét, amely munkákat az algoritmus elutasít, jelöljük G-vel, az elfogadottakat pedig H-val. Tegyük fel, hogy I olyan halmaz, amelynek esetén I_0 minimális számú munkát tartalmaz. I-t a későbbiekben az egyszerűség kedvéért minimális ellenpéldának fogjuk nevezni (bár a halmaz F részére nem követeltük meg a minimalitást).

Könnyen látható, hogy *az* algoritmus eléri el a 3. lépést. Ugyanis tegyük fel, hogy megáll a 2. lépésben. Ekkor $UB(I) = P_k \leq 1$. Ha optimális megoldás esetén van legalább egy gép, akkor a célfüggvény értéke legalább 1, emiatt ebben az

esetben a heurisztikus megoldás egyben optimális is. Ellenkező esetben optimális megoldás esetén minden munkát el kell utasítani, és a heurisztikus megoldás újra optimális. Emiatt a következőkben feltesszük, hogy az algoritmus eléri el a 3. lépést. Következik, hogy $G \cup H \neq \emptyset$. Ha $F = \emptyset$, F helyettesíthető egyetlen munkával, amelyre $w_1 = p_1 = 0$, ha pedig $W(F) > 0$, helyettesítsük az F-beli munkákat egyetlen olyan munkával, amelynek a mérete 0, míg a büntetése éppen $P(F) \leq 1$. Ezek után emiatt feltesszük hogy F egyetlen munkából álló halmaz, amelyre $w_1 = 0$, és $0 \leq p_1 = P(F) = r \leq 1$.

A következőkben belátjuk, hogy az I ellenpélda nem létezik. Azonban amiatt, hogy az ellenpélda szerkezete (ha létezne), vagyis hogy milyen munkák következnek egymás után, és mi lett velük, melyik lett elutasítva, melyik elfogadva, és mekkora lehet a méretük illetve a büntetésük, rendkívül bonyolult lehet, a következő konstrukcióval élünk: Az I sorozatot egy másik I' sorozattal helyettesítjük, amelyik szintén minimális ellenpélda, viszont egyszerűbb a szerkezete. Ezt a cserét több alkalommal elvégezzük, végül egy viszonylag már eléggé egyszerű szerkezetű ellenpéldához jutunk, amivel már könnyen elvégezhető a tétel bizonyítása, vagyis annak belátása hogy ellenpélda nem létezik.

5.3. Lemma *Feltehető, hogy az összes G-beli munka előbb érkezik mint bármely H-beli.*

Bizonyítás: Megmutatjuk, hogy a minimális ellenpélda G és H-beli elemei átrendezhetők úgy, hogy az algoritmus által kapott megoldás ugyanaz marad, de a kívánt tulajdonság teljesül. (A sorrend megváltoztatása nyilván nem változtatja meg az optimális megoldást, emiatt a módosított feladatsorozat is minimális ellenpélda marad.) Legyenek J_i és J_{i+1} egymás után következő olyan munkák, amelyekre $J_i \in H$ and $J_{i+1} \in G$. Legyen $par1, par2, par3$ a par paraméter aktuális értéke közvetlenül a J_i munka kezelése (ütemezése vagy elvetése) előtt, a J_{i+1} munka kezelése előtt, és közvetlenül a J_{i+1} munka kezelése után. Az algoritmus szabálya következtében teljesülnek ekkor $\frac{w_i}{par1} \leq p_i$ és $\frac{w_{i+1}}{par2} \geq p_{i+1}$. Három eset következhet be $par1$, $par2$, és $par3$ lehetséges értékei szerint, megmutatjuk, hogy mindhárom esetben igaz az, hogy a J_i és J_{i+1} munkák sorrendjének cseréje nem befolyásolja a heurisztikus megoldást.

1. eset. $par1 = par2 = par3$. Ekkor egyik munka érkezése sem okozza par növekedését. Ilyenkor a munkák sorrendjének cseréje nem befolyásolja a megoldást.

2. eset. $par1 = par2 < par3$. Mivel $par1 = par2$, a J_i, J_{i+1} munkák sorrendjének cseréje nem változtat azon, hogy az algoritmus továbbra is elveti az előbb érkező J_{i+1} munkát. A J_{i+1} munka elutasítása növeli par értékét. Mivel $\frac{w_i}{par3} < \frac{w_i}{par2} = \frac{w_i}{par1} \leq p_i$ teljesül, a J_i munkát a csere után is elfogadja az algoritmus.

3. eset. $par1 < par2$. $\frac{w_{i+1}}{par2} \geq p_{i+1}$ következtében teljesül, hogy $\frac{w_{i+1}}{par1} \geq p_{i+1}$.

Emiatt J_i és J_{i+1} sorrendjének cseréje esetén is el kell utasítani a J_{i+1} munkát. Függetlenül attól, hogy e munka elutasítása miatt *par* értéke növekszik-e vagy sem, J_i-t az algoritmus elfogadja a csere után, hiszen $\frac{w_i}{par1} \leq p_i$. □

A bizonyítás során ettől a ponttól feltehető, hogy a munkák sorrendje F, G, H.

5.3. Megjegyzés: *A bizonyítás során (a minimalitás miatt) feltehetjük, hogy az algoritmus által utolsóként elfogadott munka (ha elfogad az algoritmus legalább egy munkát) növeli a heurisztikus megoldás értékét. Ez azt jelenti, hogy az utolsó munka vagy új gépre kerül, vagy ha nem, akkor növeli, és ezáltal ő határozza meg a teljes átfutási időt.*

5.4. Lemma *Feltehető, hogy* $\frac{w_k}{par} = p_k$ *teljesül minden* $J_k \in G \cup H$ *munka esetén, a par paraméter aktuális értékével.*

Bizonyítás: Tegyük fel, hogy a J_k munkát az algoritmus elfogadja a 4. lépésben, ekkor $\frac{w_k}{par} \leq p_k$ teljesül. Ha még a szigorú $\frac{w_k}{par} < p_k$ egyenlőtlenség is teljesül, helyettesítsük a munkát egy másikkal, ahol a w_k értéket nem változtatjuk, azonban p_k értékét csökkentjük úgy, hogy egyenlő legyen $\frac{w_k}{par}$-ral. $\frac{w_k}{par} = p_k$ esetén az algoritmus esetén szabadon dönthetünk, hogy a munkát elutasítjuk vagy elfogadjuk, döntsünk úgy, hogy a megváltoztatott munkát ugyanúgy elfogadja az algoritmus. Ekkor a heurisztikus megoldás ugyanaz, az optimális megoldás értéke pedig nem növekszik, újra minimális ellenpéldát kaptunk. Most tegyük fel, hogy a J_k munka elutasításra kerül, és $\frac{w_k}{par} > p_k$ teljesül. Ekkor helyettesítsük a munkát egy olyan munkával, amelyik esetén a büntetés értéke nem változik, viszont a végrehajtási idő csökken, és egyenlő $par \cdot p_k$-val, és az új munka esetén is döntsünk úgy, hogy az algoritmus elutasítja a munkát.

Összegezve az előbbieket, a módosított feladatsorozat esetén minden munkára teljesül $\frac{w_k}{par} = p_k$ a paraméter aktuális értékével számolva, az algoritmus által kapott megoldás ugyanaz marad abban az értelemben, hogy az elutasított munkák büntetése ugyanaz, és az elfogadott munkák mérete is ugyanaz, mint a korábbi feladathalmaz esetén. Másrészt az optimális megoldás értéke nem növekedett, tehát kaptunk egy olyan minimális ellenpéldát, amelyikre a lemma feltétele teljesül. □

5.5. Lemma *Legyen* J_k *az utolsó* G-beli *munka (amennyiben létezik ilyen), és tegyük el, hogy par értéke ugyanannyi a munka elutasítása után közvetlenül, mint közvetlenül előtte. Jelöljük ezt a paraméter értéket* $par\,(J_k)$*-val. Ekkor ez a munka a relaxált optimális megoldás esetén elfogadásra kerül, és a relaxált optimális megoldás esetén a vásárolt gépek száma legalább* $m_\rho \geq 2par\,(J_k) + 1$.

Bizonyítás: Legyen $I' = I \backslash \{J_k\}$. Vegyük észre, hogy a heurisztikus algoritmus esetén a J_k munka hozzájárulása a célfüggvényhez p_k, másrészt ha a feladathalmazból elhagyjuk J_k-t, ez nem befolyásolja az algoritmus további menetét (hiszen

par értéke nem változott meg J_k elhagyása miatt), és így $UB(I') = UB(I) - p_k$. Tegyük fel először, hogy a relaxált optimális ütemezés esetén J_k elutasítjuk. Ekkor e munka elhagyása esetén a relaxált optimális megoldás értéke legalább p_k-val csökken, vagyis $OPT_\rho(I') \leq OPT_\rho(I) - p_k$. Ezekből pedig kapjuk, hogy

$$\frac{UB(I')}{OPT_\rho(I')} \geq \frac{UB(I) - p_k}{OPT_\rho(I) - p_k} \geq \frac{UB(I)}{OPT_\rho(I)} > 2,$$

ami ellentmond annak hogy az I ellenpéldánk minimális. Következik, hogy a J_k munkát az algoritmus elfogadja.

Tegyük fel most, hogy $m_\rho \leq 2par\,(J_k)$, és tekintsük újra az I' feladathalmazt. A J_k munka törlése a relaxált optimális megoldás értékét megint legalább $\frac{w_k}{m_\rho}$-val csökkenti, ami egyenlő $\frac{par(J_k)}{m_\rho}p_k$-val az előző lemma miatt. Emiatt $OPT_\rho(I') \leq OPT_\rho(I) - \frac{w_k}{m_\rho} = OPT_\rho(I) - \frac{par(J_k)}{m_\rho}p_k$. Ezt felhasználva

$$\frac{UB(I')}{OPT_\rho(I')} \geq \frac{UB(I) - p_k}{OPT_\rho(I) - \frac{par(J_k)}{m_\rho}p_k} \geq \frac{UB(I) - p_k}{OPT_\rho(I) - \frac{1}{2}p_k} > 2,$$

adódik, ami ismét ellentmondás. Kapjuk tehát, hogy $m_\rho > 2par\,(u_k)$. □

5.1. Definíció *Az I sorozat I'-re történő cseréjét szabályos helyettesítésnek nevezzük, ha $UB(I') \geq UB(I)$, és $OPT_\rho(I') \leq OPT_\rho(I)$.*

Megjegyzés: *Nyilvánvaló, hogy I' szintén minimális ellenpélda, ha ugyanannyi munkából áll mint I. Másrészt az $UB(I') \geq UB(I)$ feltétel biztosan teljesül, ha az elutasított munkák összbüntetése ugyanakkora az I és I' feladathalmaz esetén, és az elfogadott munkák összmérete is megegyezik.*

5.2. Definíció

a *Legyen J_i egy elutasított munka, amelynek elutasítása nem növeli a par paraméter értékét, és $p_i > 0$. Azt mondjuk, hogy a J_i munkát kettévágjuk, ha a munkát helyettesítjük a J_{i1} és J_{i2} munkákkal, ahol $w_i = w_{i1} + w_{i2}$, $p_i = p_{i1} + p_{i2}$, és $w_{i1}/p_{i1} = w_{i2}/p_{i2}$.*

b *Legyenek J_i és J_{i+1} két egymást követő elutasított munka, amelyekre $\frac{w_i}{p_i} = \frac{w_{i+1}}{p_{i+1}}$ teljesül, továbbá mendkettőt elutasítja, vagy mindkettőt elfogadja az algoritmus. Azt mondjuk, hogy a munkákat összeragasztjuk, ha helyettesítjük őket a J_l munkával, amelyre $w_l = w_i + w_{i+1} \leq 1$ és $p_l = p_i + p_{i+1}$.*

Megjegyezzük, hogy csak a definíciókban leírt feltételek teljesülése esetén definiáltuk valamely munkákra az előbbi két operációt.

5.6. Lemma *A következők szabályos helyettesítések:*
a, Egy munkát kettévágunk,
b, két munkát összeragasztunk,
c, valamely elutasított munka méretét csökkentjük, és a módosított munkát az algoritmus szintén elutasítja,
d, valamely elfogadott munka büntetését csökkentjük, és a módosított munkát az algoritmus szintén elfogadja.

Bizonyítás: Ha valamely munkát kettévágunk az előbb leírt módon, a relaxált optimális megoldás értéke nem változik, és a heurisztikus megoldás értéke is ugyanaz marad. Emiatt ez szabályos helyettesítés. A többi állítás is hasonló módon látható be. □

5.7. Lemma Ha $G \neq \emptyset$*; akkor feltehető, hogy* $r = P(F) = 1$.

Tudjuk már, hogy F egyetlen munkából álló halmaz, amelynek mérete $w_1 = 0$, valamint $0 \leq p_1 = P(F) = r \leq 1$. Legyen J_k az első G-beli munka. Ekkor $w_k = p_k$, és az algoritmus szabálya miatt $P_{k-1} \leq 1$, és $P_{k-1} + p_k > 1$. Vágjuk ketté a J_k munkát a $J_{k1}\,(1 - P_{k-1}, 1 - P_{k-1})$, és $J_{k2}\,(p_k - 1 - P_{k-1}, p_k - 1 - P_{k-1})$ részekre. Ekkor az algoritmus elutasítja a J_{k1} munkát a 2. lépésben, rögtön ezután az elutasított munkák összes büntetése éppen 1. Közvetlenül ezután az algoritmus elutasítja a J_{k2} munkát a 2. lépésben, hiszen $1 + p_k > 1$, az algoritmus további lefolyása pedig nem változott. Helyettesítsük eztán az F halmazbeli két munkát egy olyan munkával, amelynek mérete $w_1 = 0$ és büntetése $p_1 = 1$, ezek után a módosított feladatsorozat újra minimális ellenpéldát alkot, és F egyetlen munkából áll, amelyre $W(F) = 0$ és $P(F) = 1$. A szabályos helyettesítésekkel módosított feladatsorozat újra minimális ellenpéldát alkot. □

Most visszatérünk a tétel bizonyításához. (Az egyszerűség kedvéért a J_k munkát (w_k, p_k)-ként jelöljük.) Tekintsük azt a sorozatot, ahol a G halmaz tagjai:

$$G_0 = \{\text{két darab } (1,1)\,,\ \text{négy darab } (1,1/2)\,,\ \text{hat darab } (1,1/3)\,,...\}\,,\ \text{vagyis}$$

az első két munka $(1,1)$, következik négy $(1,1/2)$ munka, eztán jön hat darab $(1,1/3)$, és így tovább. (Az utolsó munka pedig $(x, x/par)$ valamilyen alkalmasan választott x számmal, ahol $0 < x \leq 1$, és a paraméter aktuális par értékével.) Belátjuk, hogy szabályos helyettesítésekkel a minimális ellenpéldánk átalakítható ilyen halmazzá.

Először is megjegyezzük, hogy a sorozat minimalitása folytán nem lehet két olyan egymást követő munka, amelyekre $\frac{w_i}{p_i} = \frac{w_{i+1}}{p_i+1}$ és $w_i + w_{i+1} \leq 1$, valamint

emlékezzünk vissza, hogy F egyetlen munkából áll, amely a következő: $(0, p)$, és ha van legalább egy elutasított munka, akkor $p = 1$. Legyen J_{last} az utolsó elutasított munka, vagyis G utolsó eleme, és legyen $j_{last} = \frac{w_{j_{last}}}{p_{j_{last}}}$, ekkor j_{last} a paraméter aktuális értéke az előbbi munka megérkeztekor. Nyilvánvalóan teljesül a $j_{last} \leq fpar$ egyenlőtlenség. Legyen $G_j = \left\{ J_k \in G \,\middle|\, \frac{w_k}{p_k} = j \right\}, 1 \leq j \leq j_{last}$, ekkor $G = G_1 \cup G_2 \cup \cdots \cup G_{j_{last}}$. ($G = \emptyset$ esetén $j_{last} = 1$, ekkor legyen definíció szerint $G_1 = \emptyset$.) A következő két típusú helyettesítést fogjuk végrehajtani néhány alkalommal.

A, Legyen J_α és J_β két egymást követő munka, amelyek mindegyike ugyanabban a G_j halmazba tartoznak valamely $1 \leq j \leq j_{last}$ esetén. Ekkor $\frac{w_\alpha}{p_\alpha} = \frac{w_\beta}{p_\beta} = j$ teljesül, valamint $w_\alpha, w_\beta \leq 1$, emiatt $p_\alpha, p_\beta \leq \frac{1}{j}$. Előbb láttuk, hogy $w_\alpha + w_\beta > 1$, és emiatt $p_\alpha + p_\beta > \frac{1}{j}$. Vágjuk szét a J_β munkát a $J_{\beta 1}$ és $J_{\beta 2}$ munkákra, ahol $p_{\beta 1} = \frac{1}{j} - p_\alpha$, és $p_{\beta 2} = p_\beta - p_{\beta 1} = p_\beta + p_\alpha - \frac{1}{j} > 0$, valamint a méretüket ennek megfelelően választjuk meg, vagyis $w_{\beta 1} = j p_{\beta 1}$ és $w_{\beta 2} = j p_{\beta 2}$. Ragasszuk eztán össze a J_α és $J_{\beta 1}$ munkákat, így kapjuk a J_γ munkát. Következik, hogy a J_γ munkához tartozó büntetés nagysága éppen $\frac{1}{j}$, emiatt a mérete pedig 1, az is látszik, hogy ez szabályos helyettesítés volt.

B, Legyen J_α és J_β két egymást követő munka, amelyekre $J_\alpha \in G_j$, de $J_\beta \in G_{j+1}$ valamely $1 \leq j < j_{last}$ esetén. Ekkor $\frac{w_\alpha}{p_\alpha} = j$, míg $\frac{w_\beta}{p_\beta} = j + 1$. Jegyezzük meg, hogy J_α az utolsó munka G_j-ben, emiatt e munka miatt növekszik a *par* paraméter értéke. Állítjuk, hogy $p_\alpha + p_\beta > \frac{1}{j}$ teljesül. Tegyük fel ezzel ellentétben hogy $p_\alpha + p_\beta \leq \frac{1}{j}$. Csökkentsük w_β értékét úgy hogy egyenlő legyen $j p_\beta$-val, és ragasszuk össze a J_α és J_β munkákat. Ennek a büntetése ekkor $p_\alpha + p_\beta \leq \frac{1}{j}$, ezért a mérete pedig legfeljebb 1. Ezáltal szabályos helyettesítés során kisebb elemszámú ellenpéldát kaptunk, ami ellentmondás. Adódik tehát, hogy valóban teljesül a $p_\alpha + p_\beta > \frac{1}{j}$ egyenlőtlenség. Vágjuk szét a J_β munkát a $J_{\beta 1}$ és $J_{\beta 2}$ munkákra úgy, hogy legyen $p_{\beta 1} = \frac{1}{j} - p_\alpha$, és $p_{\beta 2} = p_\beta - p_{\beta 1}$, a méretek pedig ennek megfelelően: $w_{\beta 1} = j p_{\beta 1}$és $w_{\beta 2} = (j+1)\, p_{\beta 2}$. Ragasszuk össze ezután a J_α és $J_{\beta 1}$ munkákat, az így kapott munka legyen ekkor J_γ Adódik, hogy J_γ büntetése éppen $\frac{1}{j}$, emiatt a mérete éppen 1, és ez is szabályos helyettesítés volt.

Ezek után már kész vagyunk arra, hogy szabályos helyettesítések során eljussunk a kívánt G halmazhoz a következőképpen: A G_1 halmazbeli első két munkára alkalmazzuk az A, helyettesítést, ezek után G_1 első munkájának a mérete éppen 1. Ha van még két további munka G_1-ben megint az A, helyettesítést alkalmazzuk, és ezután már a második G_1-beli munka mérete is éppen 1, és így tovább. Ha már csak egy további munka van G_1-ben, és G_2 nem üres, akkor G_1 utolsó, és G_2 első munkájára alkalmazzuk a B, típusú helyettesítést, és ezután G_1 utolsó munkájának a mérete is pontosan 1. Ha van legalább két munka G_2-ben megint

az A, helyettesítés következik az első két G_2-beli munkára, és így tovább. Végül megkapjuk a kívánt G halmazt. $G_{j_{last}}$ néhány $\left(1, \frac{1}{j_{last}}\right)$ típusú munkából fog állni, és az utolsó $G_{j_{last}}$-beli munka pedig $(x, x/j_{last})$ valamilyen $0 < x \leq 1$ számmal. Az előzőek miatt

$$\begin{aligned} 1 &< r + P(G_1) \leq 4, \text{ ha } j_{last} = 1, \text{ egyébként } r + P(G_1) = 4, && (60) \\ P(G_j) &= 2, \text{ ha } 2 \leq j \leq j_{last} - 1, && (61) \\ 0 &\leq P(G_{j_{last}}) \leq 2. && (62) \end{aligned}$$

Szintén teljesül, hogy

$$P_n = r + P(G) \leq 2j_{last} + 2. \tag{63}$$

Folytatjuk az ellenpélda módosítását, de most az elfogadott munkák alakítása kerül sorra. Legyen a *par* paraméter értéke rögtön az utolsó elutasított munka elutasítása után *spar*. Amennyiben az utolsó elutasított munka nem növelte a paraméter értékét, akkor persze $spar = j_{last}$ teljesül, ellenkező esetben pedig $spar = j_{last} + 1$. Más szóval, *spar* értéke egyenlő a paraméter kezdeti értékével az első elfogadott munka elfogadása előtti pillanatban. Az első néhány H-beli J_k munka esetén, vagyis amíg $W_k \leq (spar + 1)\, spar$ teljesül, tudjuk, hogy $\frac{w_k}{p_k} = spar$. Jegyezzük meg, hogy $fpar = \max\{spar, m\}$. Ha $m \leq spar$, akkor az összes H-beli J_k munka esetén teljesül hogy $\frac{w_k}{p_k} = spar$. Ha viszont $m > spar$, akkor azokra a munkákra, amelyekre már $W_k > (spar + 1)\, spar$, a paraméter aktuális értéke *spar*-nál nagyobb. Ebben az esetben $fpar = m > spar$. Legyen $H_j = \left\{ J_k \in H \,\middle|\, \frac{w_k}{p_k} = j \right\}$, $spar \leq j \leq fpar$, ekkor $H = H_{spar} \cup H_{spar+1} \cup \cdots \cup H_{fpar}$. Az előző esethez hasonlóan, ahol G elemeire hajtottunk végre megfelelő szabályos helyettesítéseket, most is feltehető, hogy a módosítás után H elemeire teljesülnek a következők:

$$\begin{aligned} W(H_{spar}) &= (spar + 1)\, spar, \text{ ha } spar < fpar, && (64) \\ W(H_j) &= 2j, \text{ ha } \quad spar < j < fpar, && (65) \\ 0 &\leq W(H_{fpar}) \leq 2fpar. && (66) \end{aligned}$$

Ez utóbbi helyettesítések lényegében ugyanúgy végezhetők, mint az előbbiek, ezért ezeknek részletezésétől itt eltekintünk.

5.8. Lemma *$G \neq \emptyset$ esetén $m_\rho = fpar$, míg $G = \emptyset$ esetén $m_\rho = 0$.*

Bizonyítás: Tegyük fel hogy $m_\rho > fpar$. Az 5.1. Lemma alapján teljesül a $C_\rho \geq m_\rho - 1$ egyenlőtlenség, emiatt $OPT_\rho(I) \geq m_\rho + (m_\rho - 1) = 2m_\rho - 1 \geq 2fpar + 1$. Felhasználva a (63) egyenlőtlenséget, tudjuk hogy $P_n \leq 2j_{last} + 2$. Amennyiben $j_{last} < fpar$ is teljesül, adódik hogy

$$\begin{aligned} UB(I) &= P_n + m + C' \\ &\leq 2j_{last} + 2 + fpar + fpar + 2 \leq 4fpar + 2 \leq 2OPT_\rho(I), \end{aligned}$$

ami ellentmondás. Következik, hogy $j_{last} = spar = fpar$. Ekkor az 5.5. Lemma miatt adódik, hogy $m_\rho \geq 2spar + 1 = 2fpar + 1$, emiatt $OPT_\rho(I) \geq 2m_\rho - 1 \geq 4fpar + 1$, valamint $UB(I) = P_n + m + C' \leq 4fpar + 4 \leq 2OPT_\rho(I)$, és ellentmondása kaptunk. Beláttuk tehát, hogy $m_\rho \leq fpar$.

Most tegyük fel, hogy G nem üres, belátjuk hogy ebben az esetben $m_\rho = fpar$. Az 5.7. Lemma alapján tudjuk már, hogy $r = P(F) = 1$. Amennyiben a relaxált feladat optimális megoldása esetén minden munka elutasításra kerül, a megoldás értéke $P(F) + P(G) + P(H) = 1 + P(G) + P(H)$. Ha a relaxált feladat optimális megoldása során a gépek száma egy, akkor pedig feltehető, hogy F kivételével minden munkát elutasítunk, és ugyanazt az értéket kapjuk mint az előbb: hiszen $W(F) = 0$, de a gépek száma eggyel nőtt. Feltehető tehát, hogy $m_\rho \geq 1$. Most tegyük fel, hogy $2 \leq m_\rho \leq j_{last} - 1$. A 2.1. Megjegyzés miatt kapjuk, hogy a relaxált optimális megoldás esetén minden F, $G_1, ..., G_{m_\rho - 1}$-beli munka elfogadásra kerül, a G_{m_ρ} elemei tetszés szerint elfogadhatóak vagy elutasíthatóak, a többi munkát, vagyis $G_{m_\rho+1}, \cdots, G_{j_{last}}$ és H elemeit pedig el kell utasítani. Feltehetjük tehát, hogy a G_{m_ρ} halmaz elemei elutasításra kerülnek. Tudjuk (60)–(61) alapján, hogy $P(G_j) = 2$ minden $j < j_{last}$ esetén, és $P(G_{j_{last}}) \leq 2$. Emiatt az elfogadott munkák összméretére teljesül a következő becslés:

$$\begin{aligned} &W(F) + W(G_1) + ... + W(G_{m_\rho - 1}) \\ &\leq 0 + 3 + 2 \cdot 2 + 2 \cdot 3 + ... + 2 \cdot m_\rho = m_\rho (m_\rho - 1) + 1, \end{aligned}$$

emiatt a relaxált optimum nem növekszik, ha a relaxált optimális gépek számát eggyel megnöveljük. Most tegyük fel, hogy $m_\rho = j_{last} < spar$. Emiatt az utolsó elutasított munka miatt növekszik a paraméter értéke, emiatt az algoritmikus szabály szerint $P_n = r + P(G) > 2 + 2par - \frac{1}{par}$ teljesül. A $j_{last} = 1$ esetben $W(G_1) = P(G_1) > 2$ adódik, ha $j_{last} > 1$, kapjuk, hogy $P(G_{j_{last}}) > 2 - \frac{1}{j_{last}}$, mindkét esetben teljesül $W(G_{j_{last}}) > 2j_{last} - 1$. Hasonlóan az előbbi levezetéshez, kapjuk, hogy

$$\begin{aligned} &W(F) + W(G_1) + ... + W(G_{m_\rho}) \\ &\leq 0 + 3 + 2 \cdot 2 + 2 \cdot 3 + ... + 2 \cdot (m_\rho - 1) + 2 \cdot m_\rho - 1 = m_\rho (m_\rho + 1), \end{aligned}$$

ami megint ellentmondás, mert a gépek száma megint növelhető, a relaxált optimum növekedése nélkül. Utoljára tegyük fel, hogy $spar \leq m_\rho < fpar$. Ekkor a relaxált optimális megoldás esetén azokat a munkákat, amelyek az F, G, illetve $H_{spar}, ..., H_{m_\rho}$ halmazokban vannak, el lehet fogadni, emiatt $W_n \geq W(H_{spar}) + ... + W(H_{m_\rho}) \geq m_\rho(m_\rho + 1)$, így a relaxált optimum nem növekszik, ha ugyanazok a munkák kerülnek elfogadásra illetve elutasításra, azonban a vásárolt gépek számát eggyel növeljük. Ezek szerint feltehető, hogy $m_\rho \geq fpar$, másrészt pedig $m_\rho \leq fpar$ biztosan teljesül, a $G \neq \emptyset$ esetre vonatkozó állítás bizonyításával készen vagyunk.

Most lássuk a $G = \emptyset$ esetet. Az előbbi esethez hasonlóan látható be, hogy ha m_ρ értéke legalább 1, akkor feltehető az is, hogy egyenlő $fpar$-ral. Azonban most, mivel $r = 1$ nem feltétlenül teljesül, az is előfordulhat, hogy $m_\rho = 0$. Más szóval, csak két esetet kell megnéznünk, mikor adódik a célfüggvénynek kisebb értéke, ha $m_\rho = 0$, vagy $m_\rho = fpar$. Az előbbi esetben minden munkát elutasítunk, az utóbbiban minden munkát elfogadunk. Jelölje a relaxált feladat célfüggvényértékét az előbbi esetben C_1, az utóbbiban C_2. Ekkor

$$\begin{aligned}
C_1 = P(I) &= P(F) + P(H) = r + P(H_1) + ... + P(G_{fpar}) \\
&= r + \frac{2 \cdot 1}{1} + \frac{2 \cdot 2}{2} + ... + \frac{2(fpar - 1)}{fpar - 1} + \frac{W(H_{fpar})}{fpar} \\
&= r + 2(fpar - 1) + \frac{W(H_{fpar})}{fpar} \\
&\leq 2fpar - 1 + \frac{W(H_{fpar})}{fpar} \\
&= fpar + \frac{fpar(fpar - 1) + W(H_{fpar})}{fpar} = fpar + \frac{W(H)}{fpar} = C_2,
\end{aligned}$$

vagyis beláttuk, hogy $G = \emptyset$ esetén feltehető, hogy $m_\rho = 0$. □

5.2. Következmény *A $G = \emptyset$ esetben $OPT_\rho(I) = r + 2(fpar - 1) + \frac{W(H_{fpar})}{fpar}$.*

5.9. Lemma *Ha $G \neq \emptyset$, és $spar > j_{last} > 1$, akkor $W(G) > j_{last}(j_{last} + 1)$.*

Bizonyítás: A lemma feltételeiből következik, hogy az utolsó elutasított munka növeli a paraméter értékét, emiatt $P(G_{j_{last}}) > 2 - \frac{1}{j_{last}}$.

Adódik, hogy

$$\begin{aligned} W(G) &= W(G_1) + \sum_{j=2}^{j_{last}-1} W(G_j) + W(G_{j_{last}}) \\ &= 0 + P(G_1) + \sum_{j=2}^{j_{last}-1} jP(G_j) + j_{last}P(G_{fpar}) \\ &> 3 + 2\sum_{j=2}^{j_{last}-1} j + j_{last}\left(2 - \frac{1}{j_{last}}\right) \\ &= 1 + j_{last}\left(j_{last} - 1 + 2 - \frac{1}{j_{last}}\right) = j_{last}\left(j_{last}+1\right). \end{aligned}$$

□

5.10. Lemma *$fpar \leq 2$ esetén $UB(I) \leq 2OPT_\rho(I)$ teljesül.*

Bizonyítás: Először vizsgáljuk meg a $G \neq \emptyset$ esetet. Ha G_2 nem üres, és az utolsó elutasított munka növeli a paramétert, akkor emiatt $fpar \geq 3$ lenne, ha pedig az utolsó elutasított munka nem növeli a paraméter értékét, akkor viszont az 5.5. Lemma miatt kapjuk, hogy $fpar \geq 3$, mindkét esetben ellentmondáshoz jutottunk. Emiatt $G = G_1$, és az előbb említett 5.5. Lemma miatt adódik, hogy az utolsó elutasított munka növeli a paramétert. Eszerint $j_{last} = 1$, $spar = fpar = 2$. Ekkor $r = 1$, továbbá $W(G_1) > 2$. Az 5.8. Lemma alapján a relaxált feladat optimumértéke $OPT_\rho(I) = \frac{W(G_1)+W(H_2)}{2}$. Ha $m = 0$, $UB(I) = r + W(G_1) \leq 2OPT_\rho(I)$. Ha $m = 1$, $UB(I) = r + W(G_1) + 1 + W(H_2) \leq 2OPT_\rho(I)$. Most tegyük fel, hogy $m = 2$, és az utolsó munka új gépre lett ütemezve. Ekkor $W(H_2) \geq 2$, emiatt $UB(I) = r + W(G_1) + 2 + 2 \leq 4 + W(G_1) + W(H_2) \leq 2OPT_\rho(I)$. Utoljára tegyük fel, hogy $m = 2$, és az utolsó munka növeli a teljes átfutási időt. Ekkor

$$\begin{aligned} UB(I) &= P_n + m + C' = 1 + W(G_1) + 2 + \frac{W(H_2) - 1}{2} + 1 \\ &= 3.5 + W(G_1) + \frac{W(H_2)}{2} \leq 4 + W(G_1) + W(H_2) = 2OPT_\rho(I). \end{aligned}$$

Most lássuk azt az esetet, amikor G üres. Ekkor az 5.8. Lemma alapján a relaxált feladat optimumértéke $OPT_\rho(I) = r + P(H)$. Ha $m = 1$, akkor $UB(I) = r+1+W(H_1) = r+1+P(H_1) \leq 2OPT_\rho(I)$, hiszen $r+P(H_1) > 1$ az algoritmus szabálya miatt. Tegyük fel, hogy $m = 2$, és az utolsó munka új gépre lett ütemezve. Ekkor $W(H_1) \geq 2$, emiatt $UB(I) = r + 2 + 2 \leq 2\left(r + W(H_1)\right) \leq 2OPT_\rho(I)$.

Utoljára tegyük fel megint, hogy $m = 2$, és az utolsó munka növeli a teljes átfutási időt. Ekkor $W(H_1) = P(H_1) = 2$, és

$$UB(I) = r + 2 + \frac{W(H_1) + W(H_2) - 1}{2} + 1 = r + 3.5 + \frac{W(H_2)}{2}$$
$$= r + 3.5 + P(H_2) \leq 2\left(r + P(H_1) + P(H_2)\right) = 2OPT_\rho(I).$$

□

A bizonyítás során mostantól feltesszük hogy $fpar \geq 3$. Már csak két lemma van hátra, az $spar = fpar$ illetve $spar < fpar$ esetek vizsgálata.

5.11. Lemma *$fpar = spar$ esetén $UB(I) \leq 2OPT_\rho(I)$ teljesül.*

Bizonyítás: Az $fpar = spar$ feltétel azt jelenti, hogy a par paraméter az algoritmus 7. lépésében kapja meg a végső értékét. Ebből következik, hogy $spar = fpar \geq 3$, valamint az is, hogy $G \neq \emptyset$. A vásárolt gépek számára teljesül a $0 \leq m \leq spar = fpar$ feltétel. Mivel $spar = fpar$, minden H-beli J_k munkára teljesül $\frac{w_k}{p_k} = fpar$. Ha $spar = j_{last}$ lenne, az 5.5 Lemma szerint $m_\rho \geq 2spar + 1$, ami ellentmondás, emiatt $spar = j_{last} + 1$, vagyis $G_{j_{last}}$ utolsó eleme növeli a paraméter értékét. Az 5.8. Lemma szerint $m_\rho = fpar$. Jegyezzük meg, hogy a munkák mérete $W(I) = W(F) + W(G) + W(H)$, ahol $W(F) = 0$, $0 \leq W(H) = W\left(H_{spar}\right) \leq 2fpar$, valamint az 5.9. Lemma miatt $W(G) > 1 + j_{last}\left(j_{last} + 1\right) - 1 = \left(fpar - 1\right) fpar$, ezek alapján a következőt kapjuk:

$$OPT_\rho(I) \geq fpar + \frac{\left(fpar - 1\right) fpar + W\left(H_{spar}\right)}{fpar} = 2fpar - 1 + \frac{W\left(H_{spar}\right)}{fpar}. \quad (67)$$

Ebből $m \leq fpar - 2$ esetén a következő adódik:

$$\begin{aligned} UB(I) &= P_n + m + C' \leq 2fpar + m + m + 2 \\ &\leq 4fpar - 2 \leq 2\left(2fpar - 1 + \frac{W_n}{fpar}\right) \leq 2OPT_\rho(I), \end{aligned}$$

emiatt feltehetjük, hogy $m = fpar - 1$ vagy $m = fpar$.

1. eset $m = fpar$. Ekkor $fpar \geq 3$ és $W\left(H_{spar}\right) \geq \left(fpar - 1\right) fpar$ miatt kapjuk, hogy

$$\begin{aligned} UB(I) &= P_n + m + C' \leq 2fpar + m + m + 2 = 2\left(fpar - 1\right) fpar \\ &\leq 2\left(3fpar - 2\right) = 2\left(2fpar - 1 + \frac{\left(fpar - 1\right) fpar}{fpar}\right) \leq 2OPT_\rho(I). \end{aligned}$$

2. eset $m = fpar - 1$, és az utolsó munka új gépre került. Ekkor felhasználva a $P_n \leq 2j_{last} + 2 \leq 2fpar$, az 5.1. Következmény, a $W_n \geq m\,(m-1) = (fpar-1)\,(fpar-2)$, és $fpar \geq 3$ feltételeket, adódik, hogy

$$\begin{aligned} UB(I) &= P_n + m + C' \leq 2fpar + m + \left(m + 1 - \frac{1}{m-1}\right) \\ &= 4fpar - 1 - \frac{1}{fpar-2} \\ &\leq 2\left(2fpar - 1 + \frac{(fpar-1)\,(fpar-2)}{fpar}\right) \leq 2OPT_\rho(I). \end{aligned}$$

3. eset $m = fpar - 1$, és az utolsó munka nem került új gépre. Ekkor

$$\begin{aligned} UB(I) &= P_n + m + C' \leq 2fpar + m + \frac{W_n - 1}{m} + 1 \\ &= 2fpar + fpar + \frac{W_n - 1}{fpar - 1} \leq 3fpar + \frac{W_n - 1}{fpar - 1} \\ &\leq 2\left(2fpar - 1 + \frac{W_n}{fpar}\right) \leq 2OPT_\rho(I). \end{aligned}$$

□

5.12. Lemma *Ha $fpar > spar$, akkor is teljesül $UB(I) \leq 2OPT_\rho(I)$.*

Bizonyítás: Ebben az esetben a *par* paraméter a 6. lépésben kapja meg a végső értékét. Jegyezzük meg, hogy ebben az esetben $m = fpar \geq 3$ teljesül.

1. eset $G \neq \emptyset$. Ekkor az 5.8. Lemma miatt $m_\rho = fpar$ teljesül. Tudjuk, hogy $r = P(F) = 1$, továbbá (63) miatt $P_n \leq 2j_{last} + 2$ teljesül. Tudjuk továbbá, hogy $j_{last} \leq spar \leq fpar - 1$. Ha $j_{last} = fpar - 1$ lenne, akkor $j_{last} = spar$ is teljesül, emiatt az 5.5. Lemma miatt $fpar = m_\rho \geq 2j_{last} + 1 = 2fpar - 1$, ami $fpar \geq 3$ miatt lehetetlen. Ha pedig $j_{last} \leq fpar - 3$, akkor

$$\begin{aligned} UB(I) = P_n + m + C' &\leq 2j_{last} + 2 + fpar + fpar + 2 \\ &\leq 2fpar - 6 + 2fpar + 4 = 2\,(2fpar - 1) \leq 2OPT_\rho(I), \end{aligned}$$

hiszen $OPT_\rho(I) \geq fpar\,(fpar-1)$ triviálisan teljesül az 5.1. Lemma miatt. Az előbbiek szerint csak az a lehetőség maradt, hogy $j_{last} = fpar - 2$. Tegyük fel, hogy $j_{last} = 1$. Ekkor $P(G1) \leq 3$, a (64)-(65) egyenlőtlenségek következtében pedig $W(H) \geq 2 \cdot 3 = 6$, emiatt

$$\begin{aligned} 2OPT_\rho(I) &= 2\left(3 + \frac{P(G_1) + W(H)}{3}\right) \geq 2\left(3 + \frac{P(G1) + 6}{3}\right) \\ &= 10 + \frac{2}{3}P(G1) \geq 9 + P(G1) = r + P(G1) + 3 + 5 \\ &\geq P_n + m + C' \geq UB(I), \end{aligned}$$

amiből következik, hogy $j_{last} > 1$, és ennek következtében $fpar = j_{last} + 2 > 3$. Amennyiben az utolsóként elutasított munka nem növeli a paraméter értékét, az 5.5. Lemma miatt $fpar = m_\rho \geq 2j_{last} + 1 = 2\,(fpar - 2) + 1$, vagyis $fpar \leq 3$, ellentmondás adódik. Emiatt az utolsóként elutasított munka növeli a paraméter értékét, így az 5.9. Lemma következtében kapjuk, hogy $W(G) > j_{last}\,(j_{last} + 1) = (fpar - 2)\,(fpar - 1) > fpar$, mivel $fpar > 3$. Szintén teljesül a $W(H) \geq fpar\,(fpar - 1)$ feltétel. Ezeket felhasználva kapjuk, hogy

$$\begin{aligned} 2OPT_\rho(I) &= 2\left(fpar + \frac{W(G) + W(H)}{fpar}\right) \\ &\geq 2\left(fpar + \frac{fpar + fpar\,(fpar - 1)}{fpar}\right) = 4fpar \\ &> 2j_{last} + 2 + fpar + fpar + 2 \geq P_n + m + C' \geq UB(I). \end{aligned}$$

2. eset $G = \emptyset$. Ekkor $m_\rho = 0$, az 5.1. Következményt, az 5.1. Lemmát és az $fpar \geq 3$ feltételt alkalmazva adódik, hogy

$$\begin{aligned} UB(I) = P_n + m + C' &\leq r + fpar + fpar + 2 = r + 2fpar + 2 \\ &\leq r + 4fpar - 4 \leq 2\,(r + 2\,(fpar - 1)) \leq 2OPT_\rho(I). \end{aligned}$$

□

A versenyképességi arányra vonatkozó tételt ezzel beláttuk.

5.3. Nyitott kérdések

Mivel még a klasszikus $P_m|online|C_{\max}$ on-line ütemezési feladatra, ahol a gépek m száma rögzített és $m > 3$, sem ismerünk optimális algoritmust, úgy tűnik, a Lista Modell feladat esetén sem könnyű optimális online algoritmust konstruálni. A negyedik fejezetben megadtunk egy online algoritmust amelynek versenyképességi aránya legfeljebb $(2\sqrt{6} + 3)/5 \approx 1.5798$. Ez megjavítja a korábbi leghatékonyabb algoritmus arányát, amely $(1 + \sqrt{5})/2 \approx 1.618$. Eztán azzal a félig online feladattal foglalkoztunk, amikor előre tudható, hogy a munkák hossza legfeljebb 1. Erre az esetre megadtunk egy optimiális algoritmust, 4/3 versenyképességi aránnyal. Ezután azzal a problémával foglalkoztunk, amikor előre tudjuk a leghosszabb munka méretét, és erre az esetre megadtunk egy legfeljebb 3/2-versenyképes algoritmust. A korábbi legjobb algoritmus erre az esetre 1.5309-versenyképes.

A korábbi algoritmusok javítása érdekében újfajta stratégiákat alkalmaztunk arra vonatkozóan, hogy mikor vásároljunk új gépet. A döntés függ a pillanatnyi teljes átfutási időtől, valamint a pillanatnyi célfüggvény értéknek és az alsó korlátnak az arányától is, emiatt az algoritmus eléggé különbözik az A_ρ algoritmusról, amelyik egyszerűen a már megérkezett munkák összhossza alapján

dönt új gép vásárlásáról. Másrészt viszont mindegyik algoritmusunkban erősen építettünk a klasszikus LS ütemezési szabályra. Ismert, hogy LS nem optimális a $P_m|on-line|C_{\max}$ feladatra $m > 3$ esetén. Emiatt érdekes lenne tudni, tovább javíthatóek-e az algoritmusok, ha a javított gépvásárlási stratégiákat az LS helyett annál hatékonyabb $P_m|on-line|C_{\max}$ algoritmussal kombináljuk. Másrészt szintén érdekes lehet egyéb félig online esetek vizsgálata is.

A gépköltséges feladat elutasításos modelljére Imreh Csanád és Nagy-György János [50] cikke megad egy 2.618-versenyképes online algoritmust, a tiszta online esetre egyelőre szintén nem ismert optimális algoritmus. Arra az esetre, amikor a munkák hossza nem nagyobb a gépek vásárlásának költségénél, és megadtunk egy egyszerű, optimális 2-versenyképes algoritmust. Érdekes lehet további optimális algoritmusokat konstruálni.

Hivatkozások

[1] S. Albers, Better bounds for on-line scheduling, *SIAM J. on Computing,* **29**, 459-473, 1999.

[2] S. Albers, On randomized online scheduling, *Proc. 34th ACM Symposium on Theory of Computing*, Montreal, 134-143, 2002.

[3] Y. Azar, L. Epstein, On-line machine covering, *J. of Scheduling*, **1**, 67-77, 1998.

[4] Y. Azar, O. Regev: Online bin stretching. Theoretical Computer Science 268 (2001), 17-41.

[5] S. Basse, Computer algorithms: Introduction to design and analysis, Reading, Addison-Wesley, 1994.

[6] Y. Bartal, A. Fiat, H. Karloff, R. Vohra, New algorithms for an ancient scheduling problem, *J. Comput. System Sci.*, **51**, 359-366, 1995.

[7] Bartal, Y., Leonardi, S., Marchetti-Spaccamela, A., Sgall, J., Stougie, L.: Multiprocessor scheduling with rejection, *SIAM J. Discrete Math.,* 13, 64-78(2000).

[8] B. Chen, A. van Vliet, G. Woeginger, New lower and upper bounds for online scheduling, *Oper. Res. Lett.*, 1994, 16, 221-230.

[9] B. Chen, C. N. Potts, G. Woeginger, A review of machine scheduling: Complexity, algorithms and approximability. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, Kluwer Academic Publishers, 1998.

[10] S.Y. Cai, Y. He, Semi-online algorithms for scheduling non-increasing processing jobs with processor cost, *Acta Automatica Sinica*, 2003, to appear. (in Chinese)

[11] J. Csirik, Cs. Imreh, J. Noga, S. S. Seiden, G. Woeginger, Buying a constant competitive ratio for paging, *Proceedings of ESA* 2001, 98–108.

[12] Dósa, Gy., He, Y., Better on-line algorithms for scheduling with machine cost, *SIAM J. on Computing*, **33,** 1035-1051(2004).

[13] Dósa, Gy., He, Y., Semi-online algorithms for parallel machine scheduling problems, *Computing* 72 (2004), no. 3-4, 355-363.

[14] Dósa, Gy., He, Y., Preemptive and non-preemptive on-line algorithms for scheduling with rejection on two uniform machines, Computing 76, 149-164, (2006).

[15] Dósa, Gy., He, Y., Scheduling with machine cost and rejection, *J. Comb. Optim.*, (2006) 12: 337-350 (online)

[16] Dósa, Gy., Graham's example is the only tight one for $P||C_{\max}$, *Annales Univ. Sci. Budapest.*, **47** (2004), 207-210.

[17] Dósa, Gy., Multifit típusú módszerek párhuzamos gépek ütemezésére, Alkalmazott Matematikai Lapok 19 (1999) 155-168.

[18] Dósa, Gy., Általánosított Multifit típusú módszerek II, Alkalmazott Matematikai Lapok 20 (2000) 91-111.

[19] Dósa, Gy., An optimal algorithm for scheduling jobs with nonincreasing sizes and machine cost, working paper

[20] Dósa, Gy., Vizvári, B., Az LPT(k)' algoritmus egyforma párhuzamos gépek ütemezésére, Alkalmazott Matematikai Lapok, (2004), 269-290.

[21] Dósa, Gy., Vizvári, B., Az általánosított LPT(k) algoritmuscsalád egyforma párhuzamos gépek ütemezésére, Alkalmazott Matematikai Lapok, 23 (2006), 17-37.

[22] Dósa, Gy., Generalized Multifit-type methods for scheduling parallel identical machines, *Pure Math. Appl.* **12** (2001), no 3, 287-296.

[23] Dósa, Gy., Optimális és javított algoritmusok a gépköltséges ütemezési feladatra, Pályamű az MTA Veszprémi Területi Bizottsága pályázatára, 2005.

[24] Du, D.L., Optimal preemptive semi-online scheduling on two uniform processors, *Information Processing Letters 92 (5):* 219-223, Dec 16 2004.

[25] D. W. Engels, D. R. Kargar, S.G. Kolliopoulos, S. Sengupta, R. N. Uma, J. Wien, Techniques for scheduling with rejection, *Journal of Algorithms*, 49 (2003) 175-191.

[26] L. Epstein, Bin stretching revisted. Acta Informatica, 39 (2003), 987-117.

[27] L. Epstein, J. Sgall, Approximation schemes for scheduling on uniformly related and identical parallel machines, *Algorithmica*, **39**, 43-57 (2004).

[28] L. Epstein, L. Favrholdt, Optimal non-preemptive semi-online scheduling to minimize makespan on two related machines, *Operations Research Letters 30 (4):* 269-275, Aug 2002.

[29] L. Epstein, J. Noga, G. J. Woeginger, online scheduling of unit time jobs with rejection: minimizing the total completion time, *Operations Research Letters*, 30(2002) 415-420.

[30] Epstein, L., Ganot, A., Optimal on-line algorithms to minimize makespan on two machines with resource augmentation, Approximation and online algorithms, Lecture Notes in Computer Science 2909: 109-122, 2004.

[31] U. Faigle, W. Kern, G. Turán, On the performance of on-line algorithm for particular problems. *Acta Cybernetica*, 9 (1989), 107-119.

[32] R. Fleischer, M. Wahl, On-line scheduling revisited, *J. of Scheduling*, 3, 343-353, 2000.

[33] G. Galambos, G. Woeginger, An on-line scheduling heuristic with better worst-case ratio than Graham's list scheduling, *SIAM J. Comput.*, 1993, 22, 349-355.

[34] M. R. Garey, D. S. Johnson, Computer and Intractability: A Guide to the theory of NP-Completeness, New York, Freeman, 1979.

[35] R. L. Graham, Bounds for certain multiprocessing anomalies, *The Bell System Technical J.*, 45, 1563-1581, 1966.

[36] Graham, R.L., Bounds on multiprocessor timing anomalies, *SIAM J. Appl. Math.* 17(1969) 416-429.

[37] L.A. Hall, "Approximation algorithms for scheduling", in: *Approximation Algorithms for NP-hard Problems*, Ed. D.S. Hochbaum, PWS Publishing Company, MA, 1977, pp. 1-45.

[38] He, Y., Dósa, Gy., Semi-online scheduling jobs with tightly-grouped processing times on three identical machines, Discrete Appl. Math. 150 (2005), no. 1-3, 140-159.

[39] He, Y., Dósa, Gy., Extension of algorithm LS for a semi-online scheduling problem, Central European Journal of Op. Res., (2006) online

[40] He, Y., Kellerer, H., Kotov, V., Linear Compound Algorithms for the Partitioning Problem, Naval Research Logistics, Vol. 47 (2000).

[41] He, Y., Min, X.: On-line machine scheduling with rejection, *Computing*, 65, 1-12 (2000).

[42] He, Y., Cai, S.Y., Semi-online scheduling with machine cost, Journal of Computer Science and Technology 17 (6): 781-787 NOV 2002.

[43] He, Y., Jiang, Y., Optimal semi-online preemptive algorithms for machine covering on two uniform machines, Theoretical Computer Science, (online változat, 2005 április 5.)

[44] Y. He, Z. Y. Tan, Ordinal online scheduling for maximizing the minimum machine completion time. Journal of Combinatorial Optimization, 6 (2002), 199-206.

[45] Y. He, G. C. Zhang, Semi on-line scheduling on two identical machines. Computing, 62 (1999), 179-187.

[46] He, Y., The optimal online parallel machine scheduling, *Computers & Mathemathics with applications*, 39, 117-121, 2000.

[47] Hoogeveen, H., Skutella, M., Woeginger, G.J.: Preemptive scheduling with rejection, *Mathematical Programming Ser. B*, 94, 361-374 (2003).

[48] C. Imreh, J. Noga, Scheduling with machine cost, In *Proc. of RANDOM-APPROX'99,* Lecture Notes in Computer Science 1671, Springer-Verlag, 1999, pp. 168-176.

[49] Imreh, Cs., Scheduling problems on two sets of identical machines, Computing 70 (4): 277-294, 2003.

[50] Nagy-György, J., Imreh, Cs., On-line scheduling with machine cost and rejection, working paper, 2006.

[51] Imreh, B., Imreh, Cs., Kombinatorikus optimalizálás, Novadat, 2006

[52] Y.W. Jiang, Y. He, Preemptive online algorithms for scheduling with machine cost, Acta Informatica, 41, 315-240, 2005.

[53] D. R. Karger, S. J. Phillips, E. Torng, A better algorithm for an ancient scheduling problem, *J. Algorithms*, **20,** 400-430, 1996.

[54] H. Kellerer, V. Kotov, M. G. Speranza, Z. Tuza, Semi on-line algorithms for the partition problem. Operations Research Letters, 21 (1997) 235-242.

[55] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, "Sequencing and scheduling: algorithms and complexity", in: *Handbooks in Operation Research and Management Science*, Vol. 4, North-Holland, Amsterdam, 1993, pp. 445-522.

[56] J. F. Rudin III, R. Chandrasekaran, Improved bounds for the online scheduling problem, *SIAM J. Comput.*, 2003, 32, 717-735.

[57] J. F. Rudin III, Improved bounds for the online scheduling problem, *Ph.D. thesis*, The University of Texas at Dallas, May 2001.

[58] J. Sgall, On-line scheduling, On-line algorithms: The state of art, *Lecture Notes in Computer Sciences* 1442, Springer Verlag, 196-231, 1998.

[59] S. Seiden, J. Sgall, G. Woeginger, Semi-online scheduling with decreasing job sizes. *Operations Research Letters,* 27 (2000), 215-227.

[60] S. S. Seiden, Preemptive multiprocessor scheduling with rejection, *Theoretical Computer Science*, 262 (2001) 437-458.

[61] S. Sengupta, Algorithms and approximation schemes for minimum lateness/tardiness with rejection, *Proceeding of WADS'2003*, Lecture Notes in Computer Science 2748, Springer, Berlin, 2000, 79-90.

[62] Z. Y. Tan, Y. He, Semi-on-line problems on two identical machines with combined partial information. *Operations Research Letters*, 30 (2002) 408-414.

[63] Z. Tan, Y. He, Semi online scheduling with ordinal data on two uniform machines. *Operations Research Letters*, **28,** 221-231, 2001.

[64] Z. Tan, Y. He, Epstein, L., Optimal on-line algorithms for the uniform machine scheduling problem with ordinal data, *Information and Computation* 196 (1): 57-70 JAN 10 2005.

[65] Vizvári, B., Demir, R., It is Difficult to Find a Difficult Problem for the Scheduling of Identical Parallel Machines, Department of Industrial Engineering of Bilkent University, *Research Report*, IEOR-9212, 1992.

[66] Vizvári, B., Bevezetés a termelésirányítás matematikai elméletébe, *Egyetemi jegyzet*, ELTE, 1992.

[67] G. C. Zhang, A simple semi-online algorithm for $P2||C_{\max}$ with a buffer. Information Processing Letters, 61 (1997), 145-148.

[68] G. C. Zhang, D. S. Ye, A note on on-line scheduling with partial information. Computers & Mathematics with Applications, 44 (2002), 539-543.

[69] Zhong, W., Dósa, Gy., Tan, Z., On the machine scheduling problem with job delivery coordination, *European Journal of Operatioanal Research,* 2006, online

Összegzés

1. Ütemezési feladatok

Az értekezésben bizonyos ütemezéselméleti feladatokkal foglalkozunk, áttekintünk néhány azokra kifejlesztett algoritmust, megvizsgáljuk azok hatékonyságát. Az értekezésben leírtak gerincét a [12, 13, 14, 15, 38] dolgozatokban megjelentek adják, amelyeket az értekezés szerzője, és az azóta (2005-ben) elhunyt Yong He publikálta. A bevezetés után következő négy fejezet új eredményeket tartalmaz. Az értekezésben szereplő új eredmények elsősorban (körülbelül 80 %-ban) az értekezés szerzőjének alkotásai. Az eredmények nagyjából 70 %-ában az értekezés szerzőjének meghatározó a hozzájárulása, a maradék 30 % esetén a szerzők hozzájárulása oszthatatlan.

A feladatok mindegyike az NP-teljes feladatosztályba tartozik, és egyben olyan feladatok, amelyeket az utóbbi időkben sokan, intenzíven kutatnak, és jelentős nemzetközi folyóiratokban publikálnak. A dolgozat első fejezetében egy rövid áttekintést közlünk az ütemezési feladatok fajtáiról, és az utóbbi évek kutatási irányzatairól. Az ezután következő négy fejezet tartalmát az alábbiakban röviden ismertetjük.

2. Félig online ütemezési feladatok

E a fejezetben a $P_m|online|C_{\max}$ feladatnak bizonyos félig online változatával foglalkoztunk, és megadunk néhány egyszerű optimális, vagy közel-optimális algoritmust. Ismert, hogy az LS algoritmus a $P_2|online|C_{\max}$ ütemezési feladat esetén optimális, és versenyképességi aránya 3/2 [31, 35]. A félig online feladatok esetén érdekes felmerülő kérdés az, hogy valamely félig online feltétel a feladatot "könnyebbé teszi-e" abban az értelemben, hogy a félig online feladatra van-e olyan algoritmus, amelynek versenyképességi aránya kisebb mint valamely optimális online algoritmus versenyképességi aránya, illetve ha egy félig online feltételhez még egy továbbit csatolunk, tovább csökken-e ezáltal a versenyképességi arány.

Kellerer, Kotov, Speranza és Tuza [54] cikke a feladat három félig online változatával foglalkozik. Az első esetben előre ismerjük a munkák méreteinek összhosszát, a másodikban felhasználható egy puffer valahány munka ideiglenes

tárolására. A harmadik esetben egyszerre két, egymástól független ütemezést készíthetünk, és végül a jobbikat választhatjuk. Kiderült, hogy az előbbi esetekben egy-egy optimális félig online algoritmus versenyképességi aránya 4/3.

A dolgozat második fejezetében az előbbi három feltételből párosítjuk valamelyik kettőt, kétféleképpen. Először feltesszük hogy előre ismert az ütemezendő feladatok méretének összege, és rendelkezésünkre áll egy 1 pozícióval rendelkező puffer az ütemezés során, vagyis egy munka ütemezését mindig tartalékolhatjuk. Erre a félig online változatra megadunk egy optimális 5/4-versenyképességű algoritmust. Megmutatjuk, hogy a puffer méretének növelésével a versenyképességi arány nem javítható. A másik esetben is előre ismert az ütemezendő feladatok méretének összege, itt viszont egyszerre két ütemezést készíthetünk, és az ütemezés végén a jobbikat választhatjuk. Ebben az esetben megadunk egy optimális, 6/5-versenyképességű algoritmust. Mindkét esetben igaz tehát az, hogy a további félig online feltétel által csökkent az elérhető versenyképességi arány.

A fejezet második részében a $P_3|online|C_{\max}$ feladatnak azzal a félig online változatával foglalkozunk, amikor előre tudjuk, hogy a munkák végrehajtási ideje közel egyforma: a munkák hossza a p és rp konstansok között van, (ahol $p > 0$, $r \geq 1$). Ha az r konstans túl nagy, akkor az előbbi információ nem sokat ér: megmutatjuk, hogy $r \geq 6$ esetén nincs olyan félig online algoritmus, amelynek versenyképességi aránya jobb lenne, mint valamely optimális online algoritmusé. Érdekes kérdés tehát az, hogy mekkora az a maximális r méretarány, amely biztosítja azt, hogy egy optimális félig-online algoritmus versenyképességi aránya jobb legyen, mint az egyszerű online esetben. (Az online feladatra három gép esetén LS optimális, versenyképességi aránya $R_{LS} = 2 - 1/3 = 5/3$ Faigle, Kern és Turán [31] cikke szerint).

Az egymáshoz közeli végrehajtási idők félig online feltétele először a [45] dolgozatban szerepelt. A cikk megmutatta, hogy két gép esetén LS optimális félig online algoritmus, melynek versenyképességi aránya $(1 + r)/2$, ha $1 \leq r \leq 2$, és $3/2$ tetszőleges $r \geq 2$ esetén. Emiatt a kétgépes esetben $r < 2$ esetén van olyan félig online algoritmus, amelynek versenyképességi aránya kisebb, mint az online eset optimális algoritmusáé. Továbbá két gép esetén LS optimális online, és egyben optimális félig-online algoritmus. Ezek az eredmények a következő kérdéseket vetik fel: Igaz-e, hogy három gép esetén is minden r méretarány esetén LS optimális marad, és milyen r esetén kapunk kisebb versenyképességi arányt valamely optimális algoritmus esetén.

Megmutatjuk, hogy három gép esetén az LS algoritmus versenyképességi

aránya az r paraméter függvényében a következőképpen adható meg:

$$R_{LS} = \begin{cases} 1 + \frac{2(r-1)}{3}, & ha \quad 1 \leq r \leq \frac{3}{2}, \\ 2 - \frac{3}{r+3}, & ha \quad \frac{3}{2} < r \leq \sqrt{3}, \\ \frac{r+1}{2}, & ha \quad \sqrt{3} < r \leq 2, \\ 2 - \frac{1}{r}, & ha \quad 2 < r \leq 3, \\ \frac{5}{3}, & ha \quad 3 < r. \end{cases}$$

Észrevehetjük, hogy a versenyképességi arány értéke az r paraméternek folytonos, szakaszos, és nem minden szakaszon lineáris függvénye. Az LS algoritmusról belátjuk, hogy $r \in [1, 3/2] \cup [\sqrt{3}, 2] \cup [6, +\infty)$ esetén optimális. $2 < r < 6$ esetén pedig megadunk javított algoritmusokat, a következők szerint:

a, Az $r \in (2, 5/2]$ esetben megadunk egy optimális algoritmust, melynek versenyképességi aránya $3/2$;

b, $r \in (5/2, 3]$ esetén egy közel optimális algoritmust $\frac{4r+2}{2r+3}$ versenyképességi aránnyal, amikor az alsó korlát legalább $\frac{7r+4+\sqrt{r^2+8r+4}}{2r+2+2\sqrt{r^2+8r+4}}$. A két érték közötti eltérés legfeljebb 0.01417.

c, Az $r \in (3, 6)$ esetben megelégszünk azzal, hogy bemutatunk egy olyan algoritmust, amelynek a versenyképességi aránya minden rögzített r paraméterérték mellett jobb mint az LS algoritmusé. A versenyképességi arány értéke $\frac{5}{3} - \frac{\delta}{18}$, ahol $\delta = \min\{\frac{6-r}{18}, \frac{1}{37}\}$, (az LS algoritmusé $\frac{5}{3}$).

Az előbbiek konklúziója, hogy három gép, és közel egyforma végrehajtási idők esetén lehetséges az LS-nél hatékonyabb algoritmust megadni, ha a méretarány kettő és hat közötti szám, vagyis LS nem minden esetben optimális. Továbbá az optimális félig-online algoritmus versenyképességi aránya erősen függ az r paraméter értékétől.

3. Hasonló gépek elutasításos modelljei

Az ütemezéselmélet elutasításos modelljei esetén minden munkához adott egy új paraméter is, ami a munka ütemezésének elutasítása esetén fizetendő büntetés összege. Az ütemezési feladat elutasításos modelljével (MSR) először Bartal és társai [7] foglalkoztak, ahol párhuzamos, egyforma gépek ütemezésének elutasításos modelljét vizsgálták. A megszakításos esettel foglalkozik [60].

A hasonló gépek ütemezésének elutasításos modelljére USR feladatként hivatkozunk, a harmadik fejezetben ennek kétgépes változatával foglalkozunk. Adott tehát n munka: $\mathcal{J} = \{J_1, \cdots, J_n\}$, mindegyik J_i munka esetén a munka hossza t_i, a munka ütemezésének elutasítása esetén fizetendő büntetés összege pedig $p_i, i = 1, \cdots, n$. Az M_1 és M_2 gépek működési sebessége $s_1 = 1$, és $s_2 = s \geq 1$. Először megengedjük a munkák megszakítását, a második modellben pedig nem. A cél mindkét esetben a teljes átfutási időnek és a kifizetett büntetések összegének minimalizálása.

Az e fejezetben tárgyaltak közvetlen előzménye He és Min [41] cikke, amely csak a megszakitás nélküli USR feladattal foglalkozik. A cikk közli az $LSR(\alpha)$ online algoritmust, melynek versenyképességi aránya

$$\begin{cases} s+\alpha(1+s-s^2), & ha\ 1 \le s < \phi, \\ \frac{s+1}{s}, & ha\ s \ge \phi, \end{cases}$$

ahol ahol $s \ge \phi$ esetén $\alpha = 1/s$, $1 \le s < \phi$ esetén pedig α egyenlő az

$$\frac{s+1}{s+x(s+1)-1} = s + x(1+s-s^2)$$

egyenlet pozitív gyökével. Az algoritmus minden $s \ge \phi$ sebesség esetén optimális, (ahol ϕ az aranymetszés aránya).

Az értekezés harmadik fejezetében a Dósa - He [14] dolgozatot követve megmutatjuk, hogy az előző algoritmus az α paraméter megfelelőbb választásával javítható, sőt, a javított algoritmus bizonyos esetekben optimális is, (amikor a korábbi nem az). A harmadik fejezetben

(i) közlünk egy olyan algoritmust az online megszakításos esetre, amely minden $s \ge 1$ sebesség esetén optimális, az algoritmus versenyképességi aránya $\frac{s+\sqrt{s^2+4s}}{2s} = \frac{1}{2} + \sqrt{\frac{1}{4}+\frac{1}{s}}$. (Tudomásunk szerint az USR feladat megszakításos esetével korábban még nem foglalkoztak.)

(ii) Mivel ismerünk optimális algoritmust a megszakítás nélküli esetre $s \ge \phi$ esetén, csak az $1 \le s < \phi$ esettel foglalkozunk. Bevezetjük az $LSRM(\alpha)$ algoritmust, amely az $LSR(\alpha)$ algoritmusnak egy módosítása, attól annyiban különbözik, hogy az α paramétert más módon választjuk meg. Az algoritmus versenyképességi aránya $\frac{1}{2s}\left(s^2+\sqrt{s^4-4s^3+4s^2+4s}\right)$, ami minden $1 \le s < \phi$ sebesség esetén jobb, mint az $LSR(\alpha)$ algoritmusé. Jegyezzük meg, hogy a [41] cikk alsó becslései triviálisak, hiszen azok az elutasítás nélküli esetben is alsó korlátok. Emiatt új, nemtriviális alsó korlátokat is közlünk, amelyek által következik, hogy $LSRM(\alpha)$ algoritmusunk egyben optimális is minden $1.3852 \le s < \phi$ esetén.

4. A gépköltséges ütemezési feladat

A [48] cikkben Imreh Csanád és John Noga felvetette a klasszikus (párhuzamos gépes) ütemezési feladat egy újfajta megközelítését. Az eredeti változattól való különbségek a következők: 1, Nincs rögzítve kezdetben a gépek száma, 2, bármikor lehetőségünk van egy újabb gép vásárlására, amikor egy újabb munka megérkezik, 3, a célfüggvény, amit minimalizálunk, a gépek vásárlására fordított összeg és a teljes átfutási idő összege.

A cikk közli az A_ρ $(1+\sqrt{5})/2 \approx 1.618$-versenyképes, online algoritmust, míg az alsó korlát 4/3. Arra az esetre amikor ismerjük a legnagyobb munka hosszúságot,

He és Cai [42] közöl egy olyan algoritmust, amelynek a versenyképességi aránya nem nagyobb mint 1.5309, miközben az alsó korlát 4/3. Arra az esetre, amikor a munkák összhossza ismert, ugyanezen cikk közöl olyan algoritmust, amelynek versenyképességi aránya legfeljebb 1.414, míg a feladatnak 1.161 alsó korlátja. Csökkenő végrehajtási idők esetére Cai and He [10] közöl egy legfeljebb 3/2-versenyképes algoritmust, az alsó korlát 4/3. Mindezen algoritmusok lényegében az A_ρ algoritmus valamilyen változatai.

A dolgozat negyedik fejezetében először megadunk egy javított, $(2\sqrt{6}+3)/5 \approx 1.5798$-versenyképes algoritmust a tiszta online esetre. Tudomásunk szerint ennél hatékonyabbat azóta nem közöltek. Ezután azzal a speciális esettel foglalkozunk, amikor a munkák mérete nem hosszabb mint a gépek vásárlásának költsége, vagyis minden munka hossza legfeljebb 1. E félig online feltétel először a [12] dolgozatban szerepelt. Erre az esetre közlünk egy 4/3-versenyképes optimális félig online algoritmust.

Harmadszor, közlünk egy legfeljebb 3/2-versenyképes kétfázisos algoritmust arra az esetre, amikor előre ismert a legnagyobb munka hosszúság. Ez az eredmény szintén megjavítja a korábbi hasonló esetre vonatkozó ismert eredményt.

5. A gépköltséges feladat elutasításos változata

Az ötödik fejezetben definiáljuk a gépköltséges feladat elutasításos változatát. Ez esetben tehát két különleges feltétel is van, az egyik, hogy lehetőségünk van új gépek vásárlására, másrészt pedig a munkák ütemezését elutasíthatjuk, ez esetben bizonyos nagyságú büntetést kell fizetnünk. Tehát ebben a fejezetben az Imreh Csanád és John Noga által bevezetett gépköltséges feladatot [48, 12], és a munkák elutasításának lehetőségét [7] ötvözzük.

A tiszta online esetre Imreh Csanád és Nagy-György János [50] cikke megad egy 2.618-versenyképes algoritmust, az általános esetben optimális algoritmus nem ismert. Mi csak azzal a félig online esettel foglalkozunk, amikor a munkák hossza legfeljebb 1. A feladat ezen megszorítás mellett is általánosítása a síkölcsönzési feladatnak (a felismerés Leah Epsteintől származik). A feladatra megadunk egy egyszerű, optimális, 2-versenyképes, kétfázisú félig online algoritmust. Az algoritmus vizsgálatakor egy újszerű módszerrel élünk: Ha az algoritmus nem lenne 2-versenyképes, akkor lennie kell (elemszám tekintetében) minimális ellenpéldának. Ez azonban rendkívül bonyolult szerkezetű feladathalmaz is lehet, (miként az algoritmus sem teljesen triviális). Emiatt az ellenpéldát fokozatosan kicseréljük újabb és újabb, de szintén minimális ellenpéldákkal, az utoljára kapott feladathatmazról pedig már viszonylag könnyen látjuk, hogy nem is ellenpélda, és így indirekt bizonyításunk véget ér.

Hivatkozások

[1] S. Albers, Better bounds for on-line scheduling, *SIAM J. on Computing,* **29**, 459-473, 1999.

[2] S. Albers, On randomized online scheduling, *Proc. 34th ACM Symposium on Theory of Computing*, Montreal, 134-143, 2002.

[3] Y. Azar, L. Epstein, On-line machine covering, *J. of Scheduling*, **1**, 67-77, 1998.

[4] Y. Azar, O. Regev: Online bin stretching. Theoretical Computer Science 268 (2001), 17-41.

[5] S. Basse, Computer algorithms: Introduction to design and analysis, Reading, Addison-Wesley, 1994.

[6] Y. Bartal, A. Fiat, H. Karloff, R. Vohra, New algorithms for an ancient scheduling problem, *J. Comput. System Sci.*, **51**, 359-366, 1995.

[7] Bartal, Y., Leonardi, S., Marchetti-Spaccamela, A., Sgall, J., Stougie, L.: Multiprocessor scheduling with rejection, *SIAM J. Discrete Math.,* 13, 64-78(2000).

[8] B. Chen, A. van Vliet, G. Woeginger, New lower and upper bounds for on-line scheduling, *Oper. Res. Lett.*, 1994, 16, 221-230.

[9] B. Chen, C. N. Potts, G. Woeginger, A review of machine scheduling: Complexity, algorithms and approximability. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, Kluwer Academic Publishers, 1998.

[10] S.Y. Cai, Y. He, Semi-online algorithms for scheduling non-increasing processing jobs with processor cost, *Acta Automatica Sinica*, 2003, to appear. (in Chinese)

[11] J. Csirik, Cs. Imreh, J. Noga, S. S. Seiden, G. Woeginger, Buying a constant competitive ratio for paging, *Proceedings of ESA* 2001, 98–108.

[12] Dósa, Gy., He, Y., Better on-line algorithms for scheduling with machine cost, *SIAM J. on Computing*, **33,** 1035-1051(2004).

[13] Dósa, Gy., He, Y., Semi-online algorithms for parallel machine scheduling problems, *Computing* 72 (2004), no. 3-4, 355-363.

[14] Dósa, Gy., He, Y., Preemptive and non-preemptive on-line algorithms for scheduling with rejection on two uniform machines, Computing 76, 149-164, (2006).

[15] Dósa, Gy., He, Y., Scheduling with machine cost and rejection, *J. Comb. Optim.*, (2006) 12: 337-350 (online)

[16] Dósa, Gy., Graham's example is the only tight one for $P||C_{\max}$, *Annales Univ. Sci. Budapest.*, **47** (2004), 207-210.

[17] Dósa, Gy., Multifit típusú módszerek párhuzamos gépek ütemezésére, Alkalmazott Matematikai Lapok 19 (1999) 155-168.

[18] Dósa, Gy., Általánosított Multifit típusú módszerek II, Alkalmazott Matematikai Lapok 20 (2000) 91-111.

[19] Dósa, Gy., An optimal algorithm for scheduling jobs with nonincreasing sizes and machine cost, working paper

[20] Dósa, Gy., Vizvári, B., Az LPT(k)' algoritmus egyforma párhuzamos gépek ütemezésére, Alkalmazott Matematikai Lapok, (2004), 269-290.

[21] Dósa, Gy., Vizvári, B., Az általánosított LPT(k) algoritmuscsalád egyforma párhuzamos gépek ütemezésére, Alkalmazott Matematikai Lapok, 23 (2006), 17-37.

[22] Dósa, Gy., Generalized Multifit-type methods for scheduling parallel identical machines, *Pure Math. Appl.* **12** (2001), no 3, 287-296.

[23] Dósa, Gy., Ütemezési feladatok gépköltséggel, Pályamű a VEAB (Veszprémi Akadémiai Bizottság) pályázatára, 2005.

[24] Du, D.L., Optimal preemptive semi-online scheduling on two uniform processors, *Information Processing Letters 92 (5):* 219-223, Dec 16 2004.

[25] D. W. Engels, D. R. Kargar, S.G. Kolliopoulos, S. Sengupta, R. N. Uma, J. Wien, Techniques for scheduling with rejection, *Journal of Algorithms*, 49 (2003) 175-191.

[26] L. Epstein, Bin stretching revisted. Acta Informatica, 39 (2003), 987-117.

[27] L. Epstein, J. Sgall, Approximation schemes for scheduling on uniformly related and identical parallel machines, *Algorithmica*, **39**, 43-57 (2004).

[28] L. Epstein, L. Favrholdt, Optimal non-preemptive semi-online scheduling to minimize makespan on two related machines, *Operations Research Letters 30 (4):* 269-275, Aug 2002.

[29] L. Epstein, J. Noga, G. J. Woeginger, online scheduling of unit time jobs with rejection: minimizing the total completion time, *Operations Research Letters*, 30(2002) 415-420.

[30] Epstein, L., Ganot, A., Optimal on-line algorithms to minimize makespan on two machines with resource augmentation, Approximation and online algorithms, Lecture Notes in Computer Science 2909: 109-122, 2004.

[31] U. Faigle, W. Kern, G. Turán, On the performance of on-line algorithm for particular problems. *Acta Cybernetica*, 9 (1989), 107-119.

[32] R. Fleischer, M. Wahl, On-line scheduling revisited, *J. of Scheduling*, 3, 343-353, 2000.

[33] G. Galambos, G. Woeginger, An on-line scheduling heuristic with better worst-case ratio than Graham's list scheduling, *SIAM J. Comput.*, 1993, 22, 349-355.

[34] M. R. Garey, D. S. Johnson, Computer and Intractability: A Guide to the theory of NP-Completeness, New York, Freeman, 1979.

[35] R. L. Graham, Bounds for certain multiprocessing anomalies, *The Bell System Technical J.*, 45, 1563-1581, 1966.

[36] Graham, R.L., Bounds on multiprocessor timing anomalies, *SIAM J. Appl. Math.* 17(1969) 416-429.

[37] L.A. Hall, "Approximation algorithms for scheduling", in: *Approximation Algorithms for NP-hard Problems*, Ed. D.S. Hochbaum, PWS Publishing Company, MA, 1977, pp. 1-45.

[38] He, Y., Dósa, Gy., Semi-online scheduling jobs with tightly-grouped processing times on three identical machines, Discrete Appl. Math. 150 (2005), no. 1-3, 140-159.

[39] He, Y., Dósa, Gy., Extension of algorithm LS for a semi-online scheduling problem, Central European Journal of Op. Res., (2006) online

[40] He, Y., Kellerer, H., Kotov, V., Linear Compound Algorithms for the Partitioning Problem, Naval Research Logistics, Vol. 47 (2000).

[41] He, Y., Min, X.: On-line machine scheduling with rejection, *Computing*, 65, 1-12 (2000).

[42] He, Y., Cai, S.Y., Semi-online scheduling with machine cost, Journal of Computer Science and Technology 17 (6): 781-787 NOV 2002.

[43] He, Y., Jiang, Y., Optimal semi-online preemptive algorithms for machine covering on two uniform machines, Theoretical Computer Science, (online változat, 2005 április 5.)

[44] Y. He, Z. Y. Tan, Ordinal online scheduling for maximizing the minimum machine completion time. Journal of Combinatorial Optimization, 6 (2002), 199-206.

[45] Y. He, G. C. Zhang, Semi on-line scheduling on two identical machines. Computing, 62 (1999), 179-187.

[46] He, Y., The optimal online parallel machine scheduling, *Computers & Mathemathics with applications*, 39, 117-121, 2000.

[47] Hoogeveen, H., Skutella, M., Woeginger, G.J.: Preemptive scheduling with rejection, *Mathematical Programming Ser. B*, 94, 361-374 (2003).

[48] C. Imreh, J. Noga, Scheduling with machine cost, In *Proc. of RANDOM-APPROX'99,* Lecture Notes in Computer Science 1671, Springer-Verlag, 1999, pp. 168-176.

[49] Imreh, Cs., Scheduling problems on two sets of identical machines, Computing 70 (4): 277-294, 2003.

[50] Nagy-György, J., Imreh, Cs., On-line scheduling with machine cost and rejection, working paper, 2006.

[51] Imreh, B., Imreh, Cs., Kombinatorikus optimalizálás, Novadat, 2006

[52] Y.W. Jiang, Y. He, Preemptive online algorithms for scheduling with machine cost, Acta Informatica, 41, 315-240, 2005.

[53] D. R. Karger, S. J. Phillips, E. Torng, A better algorithm for an ancient scheduling problem, ***J. Algorithms*, 20,** 400-430, 1996.

[54] H. Kellerer, V. Kotov, M. G. Speranza, Z. Tuza, Semi on-line algorithms for the partition problem. Operations Research Letters, 21 (1997) 235-242.

[55] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, "Sequencing and scheduling: algorithms and complexity", in: *Handbooks in Operation Research and Management Science*, Vol. 4, North-Holland, Amsterdam, 1993, pp. 445-522.

[56] J. F. Rudin III, R. Chandrasekaran, Improved bounds for the online scheduling problem, *SIAM J. Comput.*, 2003, 32, 717-735.

[57] J. F. Rudin III, Improved bounds for the online scheduling problem, *Ph.D. thesis*, The University of Texas at Dallas, May 2001.

[58] J. Sgall, On-line scheduling, On-line algorithms: The state of art, *Lecture Notes in Computer Sciences* 1442, Springer Verlag, 196-231, 1998.

[59] S. Seiden, J. Sgall, G. Woeginger, Semi-online scheduling with decreasing job sizes. *Operations Research Letters,* 27 (2000), 215-227.

[60] S. S. Seiden, Preemptive multiprocessor scheduling with rejection, *Theoretical Computer Science*, 262 (2001) 437-458.

[61] S. Sengupta, Algorithms and approximation schemes for minimum lateness/tardiness with rejection, *Proceeding of WADS'2003*, Lecture Notes in Computer Science 2748, Springer, Berlin, 2000, 79-90.

[62] Z. Y. Tan, Y. He, Semi-on-line problems on two identical machines with combined partial information. ***Operations Research Letters***, 30 (2002) 408-414.

[63] Z. Tan, Y. He, Semi online scheduling with ordinal data on two uniform machines. ***Operations Research Letters*, 28,** 221-231, 2001.

[64] Z. Tan, Y. He, Epstein, L., Optimal on-line algorithms for the uniform machine scheduling problem with ordinal data, *Information and Computation* 196 (1): 57-70 JAN 10 2005.

[65] Vizvári, B., Demir, R., It is Difficult to Find a Difficult Problem for the Scheduling of Identical Parallel Machines, Department of Industrial Engineering of Bilkent University, *Research Report*, IEOR-9212, 1992.

[66] Vizvári, B., Bevezetés a termelésirányítás matematikai elméletébe, *Egyetemi jegyzet*, ELTE, 1992.

[67] G. C. Zhang, A simple semi-online algorithm for $P2||C_{\max}$ with a buffer. Information Processing Letters, 61 (1997), 145-148.

[68] G. C. Zhang, D. S. Ye, A note on on-line scheduling with partial information. Computers & Mathematics with Applications, 44 (2002), 539-543.

[69] Zhong, W., Dósa, Gy., Tan, Z., On the machine scheduling problem with job delivery coordination, *European Journal of Operatioanal Research,* 2006, online

Abstract

1. Scheduling problems

In this thesis we deal with some scheduling problems, investigate some algorithms which are introduced to solve the problems, and analyze their efficiency. The most of the text is based on five common papers [12, 13, 14, 15, 38] of Dósa György, and Yong He, who unfortunately very early died in 2005. The algorithms being in these papers, and the analysis of them as well, are mainly (in 70 %) made by the author of this thesis. In the remained 30 % the results made by joint contribution of the authors.

All treated problems are NP-hard, and are intensively investigated in the last some years by leading researchers, and also are published in current international journals. In the first chapter of the thesis we give a short review of the classification of some scheduling problems, and the directions of the research of the last years. Now we expose the matter of the next four chapters, as follows.

2. Semi-online scheduling problems

In the second chapter we deal with some special semi online versions of problem $P_m|online|C_{\max}$, and give some optimal or nearly optimal algorithms. It is well-known, that algorithm LS is an optimal algorithm for solving the problem $P_2|online|C_{\max}$, and has competitive ratio 3/2 [31, 35].

In case of semi online problems it is an interesting question whether the semi online condition makes the problem to be easier in the next sense: Has the semi online problem an algorithm with smaller competitive ratio than the pure online problem? Furthermore, to further shed light on usefulness of different types of information, some papers considered whether the combination of two types of information can admit to construct a semi-online algorithm with much smaller competitive ratio than that of the case where only one type of information is available in advance.

Kellerer, Kotov, Speranza and Tuza [54] considered the versions where the sum of the sizes of the jobs is known in advance, or one type of additional algorithm extension is allowed. For example, a buffer is available where some of jobs can be

stored, two parallel processors are available which allows to yield two schedules simultaneously and best one is chosen at last. The first version was also studied by Zhang [67]. It has been shown that for all above versions, optimal semi-online algorithms have competitive ratio 4/3.

In the second chapter we study the semi-online versions where one type of partial information and one type of additional algorithmic extension are combined. Two versions are considered. For the semi-online version where a buffer of length 1 is available and the total size of all jobs is known in advance, we present an optimal algorithm with competitive ratio 5/4. We also show that it does not help that the buffer length is greater than 1. For the semi-online version where two parallel processors are available and the total size of all jobs is known in advance, we present an optimal algorithm with competitive ratio 6/5. Thus we can conclude that in both problem by the combination of two semi online conditions the problem can be solved more efficiently.

In the remained part of the second chapter we deal with the special version of problem $P_3|online|C_{\max}$, (called as *semi-online with tightly-grouped processing times*), where we know in advance that all jobs have their sizes between p and rp ($p > 0, r \geq 1$). The parameter r is called *size ratio*. It is allowed that the jobs with size p or/and rp may not come up in this semi-online problem, since p and rp are only the lower and upper bounds of job sizes. By normalization, we assume in this paper that $p = 1$. In fact, we will see that the knowledge of p is unnecessary for designing our algorithms. It is clear that the information is useless if r is sufficiently large, hence we are interested in the maximum r, denoted by $r_{\max}$, for which a semi-online algorithm can have a better performance than that for the pure online problem. Then the sequence satisfying $r < r_{\max}$ can be called tightly-grouped. We show that for the investigated problem $r_{\max} = 6$ holds. (In case of the pure online problem, the algorithm *List Scheduling* (LS in short) proposed by Graham [35] has competitive ratio $R_{LS} = 2 - 1/m = 5/3$, and Faigle, Kern and Turán [31] observed that LS is an optimal online algorithm for $m = 2, 3$.)

The semi-online scheduling problem with tightly-grouped processing times was proposed in [45]. For $m = 2$, it was shown that the optimal semi-online algorithm has a competitive ratio of $(1 + r)/2$ for any $1 \leq r \leq 2$ and 3/2 for any $r > 2$. It states that a job sequence is tightly-grouped for $m = 2$ iff $r < 2$. However, the optimal semi-online algorithm is just LS, the same as that for the pure online problem. Noting that LS is also an optimal algorithm for the pure online problem if $m = 3$, it is interesting to know whether it is still optimal for every $r \geq 1$, although the competitive ratio may become smaller than 5/3 for small r.

We present a comprehensive competitive ratio of LS, which is a continuous,

piecewise function on r and can be formulated as follows:

$$R_{LS} = \begin{cases} 1 + \frac{2(r-1)}{3}, & \text{if} \quad 1 \leq r \leq \frac{3}{2}, \\ 2 - \frac{3}{r+3}, & \text{if} \quad \frac{3}{2} < r \leq \sqrt{3}, \\ \frac{r+1}{2}, & \text{if} \quad \sqrt{3} < r \leq 2, \\ 2 - \frac{1}{r}, & \text{if} \quad 2 < r \leq 3, \\ \frac{5}{3}, & \text{if} \quad 3 < r. \end{cases}$$

For $r < 3$ we thus obtain competitive ratios below the ratio $5/3$, given by the pure on-line optimal algorithm. LS algorithm is optimal only for $r \in [1, 3/2] \cup [\sqrt{3}, 2] \cup [6, +\infty)$. In case $2 < r < 6$ we give improved algorithms as follows:

a, In case $r \in (2, 5/2]$ we give an optimal algorithm with competitive ratio $3/2$;

b, In case $r \in (5/2, 3]$ we introduce a near-optimal algorithm with competitive ratio of $\frac{4r+2}{2r+3}$, while the lower bound is at least $\frac{7r+4+\sqrt{r^2+8r+4}}{2r+2+2\sqrt{r^2+8r+4}}$. The largest gap between them is at most 0.01417.

c, In case $r \in (3, 6)$ we also present an improved algorithm with smaller competitive ratio than that of LS. The competitive ratio is $\frac{5}{3} - \frac{\delta}{18}$, where $\delta = \min\{\frac{6-r}{18}, \frac{3}{103}\}$.

All algorithms run in $O(n)$ time. The problem of finding optimal algorithms for $r \in (3/2, \sqrt{3})$ and $r \in (5/2, 6)$ is still open. On the base of the above results, we conclude that a job sequence is tightly-grouped for $m = 3$ iff $r < 6$, further the optimal semi-online algorithms strongly depend on the value of r, and LS is not always optimal.

3. Uniform machine scheduling with rejection

In the third chapter we consider uniform machine scheduling with rejection or USR for short. Each job is characterized by its size (processing time) and its penalty. A job can be either rejected, in which case we pay its penalty, or scheduled on machines, in which case it contributes its processing time to the completion time of that machine.

Multiprocessor scheduling with rejection was first introduced by Bartal et al. [7]. They considered the parallel and identical machine scheduling with rejection (MSR in short), where the speeds of all machines are the same.

We deal with the USR problem where the number of machines is two. There are n jobs: $\mathcal{J} = \{J_1, \cdots, J_n\}$, for every job J_i it has size t_i, and rejection penalty $p_i, i = 1, \cdots, n$. The speed of the machines M_1 and M_2 is $s_1 = 1$, $s_2 = s \geq 1$, respectively. We treat the preemptive, and also the non-preemptive cases, as well. The objective is to minimize the sum of the makespan of the schedule for all accepted jobs and the total penalty of all the rejected jobs.

As the straight antecedent of our work, for the on-line non-preemptive USR problem, He and Min [41] provided an on-line algorithm $LSR(\alpha)$. The algorithm has a competitive ratio

$$\begin{cases} s+\alpha(1+s-s^2), & \text{if } 1 \le s < \phi, \\ \frac{s+1}{s}, & \text{if } s \ge \phi, \end{cases}$$

where in case $s \ge \phi$ it holds that $\alpha = 1/s$, and in case $1 \le s < \phi$ the value of α equals to the positive root of equation

$$\frac{s+1}{s+x(s+1)-1} = s+x(1+s-s^2)$$

The algorithm is optimal for every $s \ge \phi$, where ϕ denotes the golden ratio.

Regarding our work, in the third chapter, we focus on the on-line USR problem on two uniform machines. Both preemptive and non-preemptive versions are considered.

(i) We present an optimal on-line preemptive algorithm with a competitive ratio $\frac{s+\sqrt{s^2+4s}}{2s} = \frac{1}{2} + \sqrt{\frac{1}{4}+\frac{1}{s}}$ for any $s \ge 1$. To our best knowledge, no paper ever considered on-line preemptive USR problem even for two machine case.

(ii) As algorithm $LSR(\alpha)$ is optimal for every $s \ge \phi$, we show that by properly choosing α for $1 \le s < \phi$, the algorithm can work better. Namely, the algorithm with new α has a competitive ratio $\frac{1}{2s}\left(s^2+\sqrt{s^4-4s^3+4s^2+4s}\right)$, which is smaller than that of the original α for every $1 \le s < \phi$. Furthermore, noticing that the lower bound in [41] is trivial in a sense that it is also the lower bound of the uniform machine scheduling without rejection, we further present a nontrivial lower bound of the problem under consideration. It implies that algorithm $LSR(\alpha)$ with the newly chosen α is optimal for any $1.3852 \le s < \phi$, and the maximum gap between its competitive ratio and the lower bound is 0.0534 for $1 \le s < 1.3852$.

4. Scheduling with machine cost

In [48], Imreh and Noga proposed a variant of the classical parallel machine scheduling problem. The differences are that 1) no machines are initially provided, 2) when a job is revealed the algorithm has the option to purchase new machines, and 3) the objective is to minimize the sum of the makespan and cost of machines. We refer to this problem as the *List Model*.

For the List Model problem, Imreh and Noga [48] presented an on-line $(1+\sqrt{5})/2 \approx 1.618$-competitive algorithm A_ρ. For the semi-online problem with known largest size, He and Cai [42] presented an algorithm with a competitive ratio at most 1.5309. For the semi-online problem with decreasing sizes, Cai and

He [10] presented an algorithm with a competitive ratio 3/2. For all previous cases the lower bound is 4/3. For the semi-online problem with known total size, He and Cai [42] presented an algorithm with a competitive ratio at most 1.414 while the lower bound is 1.161. These semi-online algorithms are essentially modified from A_ρ.

In this thesis, we first present a new on-line algorithm with a competitive ratio at most $(2\sqrt{6}+3)/5 \approx 1.5798$, which improves the known upper bound 1.618. Up to the author's best knowledge, there has not been published yet better algorithm for this problem. Then for a special case where every job has a size no greater than the machine cost (called *small job*), we present an optimal on-line algorithm with a competitive ratio 4/3. This semi online version is treated first by Dósa and He [12].

Last, we present a new two-phase algorithm with a competitive ratio at most 3/2 for the semi-online problem with known largest size, which also improves the known result.

5. Scheduling with machine cost and rejection

In Chapter 5 we define that variant of the classical parallel machine scheduling problem which has the special feature that we have a possibility to purchase new machines (introduced by Imreh Csanád and John Noga, [48, 12]), and jobs can be rejected at a certain cost ([7]).

Recently, Nagy-György and Imreh presented a 2.618-competitive online algorithm for the general problem ([50]), optimal algorithm is not known yet. We consider a special case of the problem, called *small job case*, where we assume that all jobs have sizes not greater than 1. The small job case was first proposed in Dósa - He, [12] in the case of the scheduling problem with machine cost. Note, that our problem MCR in the small job case is even a generalization of the Ski-Rental Problem (SRP for short). It is well known that problem SRP have optimal deterministic online algorithms with competitive ratio 2, thus 2 is also a lower bound for problem MCR even in the small job case.

In this paper we present a simple optimal two phase online algorithm for the small job case with a competitive ratio 2. In the first phase, we reject the first few jobs to avoid the situation that the first machine is purchased too early. In the second phase we avoid that the total penalty of all rejected jobs, or sum of the makespan (for the accepted jobs) and the number of purchased machines would be too large. Because of this carefully approach, opposite to the *Greedy* methods, we call the algorithm as *Carefully*. We hope that the idea of the algorithm design and analysis presented in this paper can be further applied to solve efficiently also the general problem. In the proof we use some new techniques.

Hivatkozások

[1] S. Albers, Better bounds for on-line scheduling, *SIAM J. on Computing,* **29**, 459-473, 1999.

[2] S. Albers, On randomized online scheduling, *Proc. 34th ACM Symposium on Theory of Computing*, Montreal, 134-143, 2002.

[3] Y. Azar, L. Epstein, On-line machine covering, *J. of Scheduling*, **1**, 67-77, 1998.

[4] Y. Azar, O. Regev: Online bin stretching. Theoretical Computer Science 268 (2001), 17-41.

[5] S. Basse, Computer algorithms: Introduction to design and analysis, Reading, Addison-Wesley, 1994.

[6] Y. Bartal, A. Fiat, H. Karloff, R. Vohra, New algorithms for an ancient scheduling problem, *J. Comput. System Sci.*, **51**, 359-366, 1995.

[7] Bartal, Y., Leonardi, S., Marchetti-Spaccamela, A., Sgall, J., Stougie, L.: Multiprocessor scheduling with rejection, *SIAM J. Discrete Math.,* 13, 64-78(2000).

[8] B. Chen, A. van Vliet, G. Woeginger, New lower and upper bounds for on-line scheduling, *Oper. Res. Lett.*, 1994, 16, 221-230.

[9] B. Chen, C. N. Potts, G. Woeginger, A review of machine scheduling: Complexity, algorithms and approximability. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, Kluwer Academic Publishers, 1998.

[10] S.Y. Cai, Y. He, Semi-online algorithms for scheduling non-increasing processing jobs with processor cost, *Acta Automatica Sinica*, 2003, to appear. (in Chinese)

[11] J. Csirik, Cs. Imreh, J. Noga, S. S. Seiden, G. Woeginger, Buying a constant competitive ratio for paging, *Proceedings of ESA* 2001, 98–108.

[12] Dósa, Gy., He, Y., Better on-line algorithms for scheduling with machine cost, *SIAM J. on Computing*, **33,** 1035-1051(2004).

[13] Dósa, Gy., He, Y., Semi-online algorithms for parallel machine scheduling problems, *Computing* 72 (2004), no. 3-4, 355-363.

[14] Dósa, Gy., He, Y., Preemptive and non-preemptive on-line algorithms for scheduling with rejection on two uniform machines, Computing 76, 149-164, (2006).

[15] Dósa, Gy., He, Y., Scheduling with machine cost and rejection, *J. Comb. Optim.*, (2006) 12: 337-350 (online)

[16] Dósa, Gy., Graham's example is the only tight one for $P||C_{\max}$, *Annales Univ. Sci. Budapest.*, **47** (2004), 207-210.

[17] Dósa, Gy., Multifit típusú módszerek párhuzamos gépek ütemezésére, Alkalmazott Matematikai Lapok 19 (1999) 155-168.

[18] Dósa, Gy., Általánosított Multifit típusú módszerek II, Alkalmazott Matematikai Lapok 20 (2000) 91-111.

[19] Dósa, Gy., An optimal algorithm for scheduling jobs with nonincreasing sizes and machine cost, working paper

[20] Dósa, Gy., Vizvári, B., Az LPT(k)' algoritmus egyforma párhuzamos gépek ütemezésére, Alkalmazott Matematikai Lapok, (2004), 269-290.

[21] Dósa, Gy., Vizvári, B., Az általánosított LPT(k) algoritmuscsalád egyforma párhuzamos gépek ütemezésére, Alkalmazott Matematikai Lapok, 23 (2006), 17-37.

[22] Dósa, Gy., Generalized Multifit-type methods for scheduling parallel identical machines, *Pure Math. Appl.* **12** (2001), no 3, 287-296.

[23] Dósa, Gy., Ütemezési feladatok gépköltséggel, Pályamű a VEAB (Veszprémi Akadémiai Bizottság) pályázatára, 2005.

[24] Du, D.L., Optimal preemptive semi-online scheduling on two uniform processors, *Information Processing Letters 92 (5):* 219-223, Dec 16 2004.

[25] D. W. Engels, D. R. Kargar, S.G. Kolliopoulos, S. Sengupta, R. N. Uma, J. Wien, Techniques for scheduling with rejection, *Journal of Algorithms*, 49 (2003) 175-191.

[26] L. Epstein, Bin stretching revisted. Acta Informatica, 39 (2003), 987-117.

[27] L. Epstein, J. Sgall, Approximation schemes for scheduling on uniformly related and identical parallel machines, *Algorithmica*, **39**, 43-57 (2004).

[28] L. Epstein, L. Favrholdt, Optimal non-preemptive semi-online scheduling to minimize makespan on two related machines, *Operations Research Letters 30 (4):* 269-275, Aug 2002.

[29] L. Epstein, J. Noga, G. J. Woeginger, online scheduling of unit time jobs with rejection: minimizing the total completion time, *Operations Research Letters*, 30(2002) 415-420.

[30] Epstein, L., Ganot, A., Optimal on-line algorithms to minimize makespan on two machines with resource augmentation, Approximation and online algorithms, Lecture Notes in Computer Science 2909: 109-122, 2004.

[31] U. Faigle, W. Kern, G. Turán, On the performance of on-line algorithm for particular problems. *Acta Cybernetica*, 9 (1989), 107-119.

[32] R. Fleischer, M. Wahl, On-line scheduling revisited, *J. of Scheduling*, 3, 343-353, 2000.

[33] G. Galambos, G. Woeginger, An on-line scheduling heuristic with better worst-case ratio than Graham's list scheduling, *SIAM J. Comput.*, 1993, 22, 349-355.

[34] M. R. Garey, D. S. Johnson, Computer and Intractability: A Guide to the theory of NP-Completeness, New York, Freeman, 1979.

[35] R. L. Graham, Bounds for certain multiprocessing anomalies, *The Bell System Technical J.*, 45, 1563-1581, 1966.

[36] Graham, R.L., Bounds on multiprocessor timing anomalies, *SIAM J. Appl. Math.* 17(1969) 416-429.

[37] L.A. Hall, "Approximation algorithms for scheduling", in: *Approximation Algorithms for NP-hard Problems*, Ed. D.S. Hochbaum, PWS Publishing Company, MA, 1977, pp. 1-45.

[38] He, Y., Dósa, Gy., Semi-online scheduling jobs with tightly-grouped processing times on three identical machines, Discrete Appl. Math. 150 (2005), no. 1-3, 140-159.

[39] He, Y., Dósa, Gy., Extension of algorithm LS for a semi-online scheduling problem, Central European Journal of Op. Res., (2006) online

[40] He, Y., Kellerer, H., Kotov, V., Linear Compound Algorithms for the Partitioning Problem, Naval Research Logistics, Vol. 47 (2000).

[41] He, Y., Min, X.: On-line machine scheduling with rejection, *Computing*, 65, 1-12 (2000).

[42] He, Y., Cai, S.Y., Semi-online scheduling with machine cost, Journal of Computer Science and Technology 17 (6): 781-787 NOV 2002.

[43] He, Y., Jiang, Y., Optimal semi-online preemptive algorithms for machine covering on two uniform machines, Theoretical Computer Science, (online változat, 2005 április 5.)

[44] Y. He, Z. Y. Tan, Ordinal online scheduling for maximizing the minimum machine completion time. Journal of Combinatorial Optimization, 6 (2002), 199-206.

[45] Y. He, G. C. Zhang, Semi on-line scheduling on two identical machines. Computing, 62 (1999), 179-187.

[46] He, Y., The optimal online parallel machine scheduling, *Computers & Mathemathics with applications*, 39, 117-121, 2000.

[47] Hoogeveen, H., Skutella, M., Woeginger, G.J.: Preemptive scheduling with rejection, *Mathematical Programming Ser. B*, 94, 361-374 (2003).

[48] C. Imreh, J. Noga, Scheduling with machine cost, In *Proc. of RANDOM-APPROX'99,* Lecture Notes in Computer Science 1671, Springer-Verlag, 1999, pp. 168-176.

[49] Imreh, Cs., Scheduling problems on two sets of identical machines, Computing 70 (4): 277-294, 2003.

[50] Nagy-György, J., Imreh, Cs., On-line scheduling with machine cost and rejection, working paper, 2006.

[51] Imreh, B., Imreh, Cs., Kombinatorikus optimalizálás, Novadat, 2006

[52] Y.W. Jiang, Y. He, Preemptive online algorithms for scheduling with machine cost, Acta Informatica, 41, 315-240, 2005.

[53] D. R. Karger, S. J. Phillips, E. Torng, A better algorithm for an ancient scheduling problem, ***J. Algorithms*, 20,** 400-430, 1996.

[54] H. Kellerer, V. Kotov, M. G. Speranza, Z. Tuza, Semi on-line algorithms for the partition problem. Operations Research Letters, 21 (1997) 235-242.

[55] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, "Sequencing and scheduling: algorithms and complexity", in: *Handbooks in Operation Research and Management Science*, Vol. 4, North-Holland, Amsterdam, 1993, pp. 445-522.

[56] J. F. Rudin III, R. Chandrasekaran, Improved bounds for the online scheduling problem, *SIAM J. Comput.*, 2003, 32, 717-735.

[57] J. F. Rudin III, Improved bounds for the online scheduling problem, *Ph.D. thesis*, The University of Texas at Dallas, May 2001.

[58] J. Sgall, On-line scheduling, On-line algorithms: The state of art, *Lecture Notes in Computer Sciences* 1442, Springer Verlag, 196-231, 1998.

[59] S. Seiden, J. Sgall, G. Woeginger, Semi-online scheduling with decreasing job sizes. *Operations Research Letters,* 27 (2000), 215-227.

[60] S. S. Seiden, Preemptive multiprocessor scheduling with rejection, *Theoretical Computer Science*, 262 (2001) 437-458.

[61] S. Sengupta, Algorithms and approximation schemes for minimum lateness/tardiness with rejection, *Proceeding of WADS'2003*, Lecture Notes in Computer Science 2748, Springer, Berlin, 2000, 79-90.

[62] Z. Y. Tan, Y. He, Semi-on-line problems on two identical machines with combined partial information. ***Operations Research Letters***, 30 (2002) 408-414.

[63] Z. Tan, Y. He, Semi online scheduling with ordinal data on two uniform machines. ***Operations Research Letters*, 28,** 221-231, 2001.

[64] Z. Tan, Y. He, Epstein, L., Optimal on-line algorithms for the uniform machine scheduling problem with ordinal data, *Information and Computation* 196 (1): 57-70 JAN 10 2005.

[65] Vizvári, B., Demir, R., It is Difficult to Find a Difficult Problem for the Scheduling of Identical Parallel Machines, Department of Industrial Engineering of Bilkent University, *Research Report*, IEOR-9212, 1992.

[66] Vizvári, B., Bevezetés a termelésirányítás matematikai elméletébe, *Egyetemi jegyzet*, ELTE, 1992.

[67] G. C. Zhang, A simple semi-online algorithm for $P2||C_{\max}$ with a buffer. Information Processing Letters, 61 (1997), 145-148.

[68] G. C. Zhang, D. S. Ye, A note on on-line scheduling with partial information. Computers & Mathematics with Applications, 44 (2002), 539-543.

[69] Zhong, W., Dósa, Gy., Tan, Z., On the machine scheduling problem with job delivery coordination, *European Journal of Operatioanal Research,* 2006, online