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

Ph.D értekezés tézisei

Készítette: Dósa György

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

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

Informatika Doktori Program,
Operációkutatás és kombinatorikus optimalizálás alprogram

Veszprém, 2007 február

# 1. Az értekezés tárgya

Az értekezésben néhány ütemezéselméleti feladattal foglalkozunk, és mindegyikre bevezetünk egy vagy több algoritmust, és megvizsgáljuk azok versenyképességi arányát. Az értekezésben leírtak gerincét a [4, 5, 6, 7, 21] dolgozatokban megjelentek adják, amelyeket az értekezés szerzője, és az azóta (2005-ben) elhunyt Yong He publikálta. A bevezetés után következő négy fejezet új eredményt 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. Az értekezés első fejezetében egy rövid áttekintést közlünk az ütemezési feladatok néhány fajtájáról, és az utóbbi évek kutatási irányzatairól. Az ezután következő négy fejezet tartalmát az alábbiakban röviden ismertetjük. Mivel az értekezés tizenegy új algoritmust tartalmaz különböző feladatokra, ezek mindegyikének ismertetésére itt nincs elegendő hely. Megelégszünk ezért ízelítőként, csak egy érdekesebbnek ítélt algoritmus közlésével.

# 2. Félig online ütemezési feladatok

E a fejezetben a $P_m|online|C_{\max}$ feladatnak bizonyos félig online változatával foglalkoztunk. Ismert, hogy az LS algoritmus a $P_2|online|C_{\max}$ ütemezési feladat esetén optimális, és versenyképességi aránya 3/2 [16, 19]. 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. Kellerer, Kotov, Speranza és Tuza [32] 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.

Olyan vizsgálatok is történtek, hogy két különböző félig online feltétel együttes megléte tovább csökkenti-e a feladat optimális algoritmusának versenyképességi arányát. Tan és He [35] közöl olyan feltétel-kombinációkat, amelyek "haszontalanok" abban az értelemben, hogy ezek együtt sem adnak jobb versenyképességi arányt, mint az egyszerű online feltétel. Ugyanebben a cikkben megmutatták, hogy ha előre ismerjük a munkák összhosszát, és 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.

Az értekezés második fejezetében [32] előbbi három feltételéből párosítjuk valamelyik kettőt, kétféleképpen.

Először feltesszük hogy ismert az ütemezendő feladatok méretének összege, és rendelkezésünkre áll egy 1 pozícióval rendelkező puffer az ütemezés során. Erre a félig online változatra megkonstruáltuk az $ALG2.1$ algoritmust, és bebizonyítottuk annak optimalitását az alábbiak szerint:

**2.1. Tétel** *Az* $ALG2.1$ *algoritmus versenyképességi aránya legfeljebb* $5/4$.

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

A másik esetben is előre ismert az ütemezendő feladatok méretének összege, itt viszont egyszerre két egymástól független ütemezést készíthetünk, és az ütemezés végén a jobbikat választhatjuk. A feladatra megadtuk az $ALG2.2$ algoritmust.

**2.3. Tétel** *Az* $ALG2.2$ *algoritmus versenyképességi aránya* $6/5$.

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

Tehát ez az algoritmus is optimális. A következőt kaptuk: A $P_2|online|C_{\max}$ ütemezési feladat esetén LS optimális, 3/2 versenyképességi aránnyal; a [32] cikkbeli három félig online változat esetén egy-egy optimális félig online algoritmus versenyképességi aránya 4/3, ha pedig ezekből kettőt-kettőt kombinálunk az előbbiek szerint, tovább csökkent (de nem egyenlő mértékben) 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: a munkák hossza 1 és $r > 1$ között van. Az értekezésben leírtak közvetlen előzménye He és Zhang [26] cikke, amely a kétgépes feladatra vonatkozó eredményeket közöl. Kiderül, hogy két gép esetén $r < 2$ esetén képes csökkenteni a félig online feltétel az online eset versenyképességi arányát, és tetszőleges $r$ méretarány esetén $LS$ optimális algoritmus. A háromgépes feladattal korábban még nem foglalkoztak.

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

**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.*

Az előbbi eredmények közül csak az $1 \leq r \leq 3/2$, és $r \geq 3$ esetek állításai azok, amelyek korábbról ismertek voltak, [16, 19, 26] alapján. Látjuk, hogy a versenyképességi arány értéke az $r$ paraméternek folytonos, szakaszos, és nem minden szakaszon lineáris függvénye. Megmutatjuk, hogy $LS$ nem optimális $2 < r < 6$ esetén, továbbá $(2; 6)$ három részintervallumára megadunk három javított algoritmust, a következők szerint:

**a**, Az $r \in (2, 5/2]$ esetben bevezetjük az $ALG(\gamma)$ algoritmust. Erre teljesül a következő

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

**b**, $r \in (5/2, 3]$ esetén bevezetjük az $ALG2.3$ algoritmust.

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

Azt is megmutatjuk, hogy $r \in (5/2, 3]$ esetén a feladat alsó korlátja legalább $\frac{7r+4+\sqrt{r^2+8r+4}}{2r+2+2\sqrt{r^2+8r+4}}$, az alsó korlát és a versenyképességi arány közötti eltérés legfeljebb 0.01417.

**c**, Az $r \in (3, 6)$ esetben megelégszünk azzal, hogy bemutatjuk az $ALG2.4$ algoritmust, amelynek a versenyképességi aránya minden rögzített $r$ paraméterérték mellett jobb mint az $LS$ algoritmusé, amelynek $\frac{5}{3}$:

**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* $\delta = \min\left\{\frac{6-r}{18}, \frac{3}{103}\right\} > 0$.

Az előzőek szerint lényeges különbségek adódnak, ha a gépek száma nem kettő hanem három, először is, $LS$ nem minden méretarány esetén optimális: Akkor lehetséges $LS$-nél hatékonyabb algoritmust megadni, ha a méretarány kettő és

hat közötti szám. Továbbá az optimális félig-online algoritmus versenyképességi aránya erősen függ az $r$ paraméter értékétől, és $r$ különböző intervallumaira különböző algoritmusokat kell magadni, ha azt akarjuk, hogy azok hatékonyan működjenek. Mindhárom bemutatott új algoritmus lineáris futásidejű.

# 3. Hasonló gépek elutasításos modelljei

Az e fejezetben tárgyaltak közvetlen előzménye He és Min [24] cikke, amely csak a megszakitás nélküli feladattal foglalkozik. A cikk közli az $LSR(\alpha)$ online algoritmust, amely minden $s \geq \phi$ sebesség esetén optimális, (ahol $\phi$ az aranymetszés aránya).

A fejezetben a Dósa - He [6] dolgozatot követve megmutatjuk, hogy az előző algoritmus az $\alpha$ paraméter megfelelőbb választásával javítható: A javított algoritmus versenyképességi aránya minden $1 \leq s < \phi$ sebesség esetén jobb, mint a korábbié, sőt, a javított algoritmus bizonyos esetekben optimális is, (amikor a korábbi nem az). Az értekezés harmadik fejezete a következő eredményeket tartalmazza:

(i) bevezetjük a $PMPT(\alpha)$ algoritmust az online megszakításos esetre. Erre teljesül a következő

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

Tudomásunk szerint a feladat megszakításos esetével korábban nem foglalkoztak. Jegyezzük meg, hogy az elutasítás nélküli online megszakításos esetre [15, 36] optimális algoritmust közöl két hasonló gép esetére, természetesen adódna az ötlet, hogy ezt az optimális ütemezési algoritmust kombináljuk egy megfelelő elutasítási stratégiával. Mégsem ezt tettük, ugyanis egy sokkal egyszerűbb ütemezési algoritmust alkalmazva is optimális algoritmust kaptunk a feladatra.

(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 a korábbi $LSR(\alpha)$ algoritmusnak egy módosítása, attól annyiban különbözik, hogy az $\alpha$ paramétert más módon választjuk meg: 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

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

Legyen továbbá

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

**3.7. Tétel** *Tetszőleges $1 \le s < \phi$ sebesség esetén az $LSRM(\alpha)$ 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.*

Mivel a [24] cikk alsó becslései triviálisak, új, nemtriviális alsó korlátokat is közlünk, az alábbiak szerint: 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:

$$\frac{1}{2}+\sqrt{\frac{1}{4}+\frac{1}{s}} = \frac{2s+1}{s+1}, \frac{2s+1}{s+1} = \frac{s\,(s-1)+x_0}{1+s-s^2}, \frac{s\,(s-1)+x_0}{1+s-s^2} = s+x_0.$$

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

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

**3.9. Következmény** *Tetszőleges $s_3 \approx 1.3852 \le s < \phi$ esetén az $LSRM(\alpha)$ algoritmus optimális.*

Továbbá az $LSRM(\alpha)$ algoritmus versenyképességi aránya minden $1 \le s < \phi$ esetén jobb, mint a korábbi $LSR(\alpha)$ algoritmusé, és az $LSRM(\alpha)$ algoritmus versenyképességi aránya és az alsó korlát közötti maximális rés 0.0534.

# 4. A gépköltséges ütemezési feladat

Az értekezés negyedik és ötödik fejezetében a gépköltséges feladattal foglalkozunk. E két fejezetnek egy részben kibővített változata [14] a Veszprémi Akadémiai Bizottság 2005 évi pályázatán I. díjat kapott. A [28] 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 $A_\rho$ $(1+\sqrt{5})/2 \approx 1.618$-versenyképes, online algoritmust, míg a feladat alsó korlátja $4/3$.

Az eredeti cikk nyomán, többen foglalkoztak a feladattal. Arra az esetre amikor előre ismert a legnagyobb munkahosszúság, He és Cai [25] közöl egy olyan algoritmust, amelynek a versenyképességi aránya nem nagyobb mint 1.5309, miközben az alsó korlát $4/3$.

Az értekezés negyedik fejezetében közöljük a $H1$ algoritmust a tiszta online esetre, amely a következő: ($k$ a munkák sorszámát, $m$ a vásárolt gépek számát jelenti.)

$H1$ **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.

Tudomásunk szerint ennél (versenyképességi arány tekintetében) jobbat azóta nem közöltek. Az $A_\rho$ algoritmussal ellentétben, $H1$ új stratégia szerint vásárol újabb gépet.

**4.9. Tétel** *A* $H1$ *algoritmus versenyképességi aránya legfeljebb* $(2\sqrt{6}+3)/5 \approx 1.5798$.

Jegyezzük meg, hogy Imreh Csanád [27] cikke visszatér az $A_\rho$ algoritmushoz: Általános gépköltség-függvény esetére (tehát a gépek költsége nem egyenlő) ennek bizonyos változataival, és azok hatékonyság-vizsgálatával foglalkozik. Amint kiderül, bizonyos esetekben $A_\rho$ is hatékonyan viselkedik.

Az értekezésben 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 [4] dolgozatban szerepelt. Erre az esetre bevezetjük a $H2$ algoritmust. Jegyezzük meg, hogy az eredeti $A_\rho$ algoritmus nem lehet jobb mint 3/2-versenyképes a kis méretű munkák esetén. A $H2$ algoritmusra teljesül a következő

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

Ez az első optimális algorimus, amit sikerült megadni a gépköltséges feladat valamilyen változatára. A már említett [27] cikkben, általános gépköltség esetén, szintén sikerült megadni optimális algoritmust kis munkák esetére: amikor bármelyik gép költsége legalább akkora, mint a legnagyobb munkahosszúság.

Harmadszor, közöljük a $H3$ kétfázisos algoritmust arra az esetre, amikor előre ismert a legnagyobb munka hosszúság. Ez az algoritmus az előzőekhez képest újfajta stratégiát követ e gépek vásárlására, és szintén némileg megjavítja a korábban ismert legjobb eredményt.

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

A tételt 14 lemma és két megjegyzés segítségével látjuk be.

# 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 [4, 28], és a munkák elutasításának lehetőségét [1] ötvözzük.

A tiszta online esetre Imreh Csanád és Nagy-György János [29] 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 fejezet tartalma a [7] cikkben jelent meg. A feladat ezen megszorítás mellett is általánosítása az úgynevezett sí-kölcsönzési feladatnak. A feladatra megadjuk az Óvatos algoritmust, amelynek nevét az indokolja, hogy némileg ellenkezőképpen viselkedik, mint a szokásos "mohó" módszerek. Az algoritmus egy egyszerű, kétfázisú félig online algoritmus. Az első fázis során csak

arra vigyázunk, nehogy túl korán vásároljuk meg az első gépet, a második fázisban pedig azon igyekezünk, hogy az elutasított munkák összbüntetése se, és az elfogadott munkák összhossza se lehessen túl nagy a másik rovására.

**5.1. Tétel** *Az Óvatos algoritmus optimális,* 2-versenyképes algoritmus.

A gépköltséges feladat elutasításos változatára más optimális algoritmus még nem született (a szerző ismeretei szerint). 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. A cserék során sikerül elérni, hogy végül egy szép, eléggé szabályos szerkezetű ellenpéldát kapunk.

Az utoljára kapott feladathatmazról pedig már viszonylag könnyen belátjuk, hogy nem is ellenpélda, és így indirekt bizonyításunk véget ér.

# Hivatkozások


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

[2] 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)

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

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

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

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

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

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

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

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

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

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

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

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

[15] Epstein, L., Noga, J., Seiden, S. S., Sgall, J., Woeginger, G. J.: Randomized online scheduling for two related machines. *J. of Scheduling 4*, 71-92 (2001).

[16] U. Faigle, W. Kern, G. Turán, On the performance of on-line algorithm for particular problems. *Acta Cybernetica*, 9 (1989), 107-119.

[17] 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.

[18] M. R. Garey, D. S. Johnson, Computer and Intractability: A Guide to the theory of NP-Completeness, New York, Freeman, 1979.

[19] R. L. Graham, Bounds for certain multiprocessing anomalies, *The Bell System Technical J.*, 45, 1563-1581, 1966.

[20] Graham, R.L., Bounds on multiprocessor timing anomalies, *SIAM J. Appl. Math.* 17(1969) 416-429.

[21] 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.

[22] He, Y., Dósa, Gy., Extension of algorithm LS for a semi-online scheduling problem, Central European Journal of Op. Res., (2006) online

[23] He, Y., Kellerer, H., Kotov, V., Linear Compound Algorithms for the Partitioning Problem, Naval Research Logistics, Vol. 47 (2000).

[24] He, Y., Min, X.: On-line machine scheduling with rejection, *Computing*, 65, 1-12 (2000).

[25] He, Y., Cai, S.Y., Semi-online scheduling with machine cost, Journal of Computer Science and Technology 17 (6): 781-787 NOV 2002.

[26] Y. He, G. C. Zhang, Semi on-line scheduling on two identical machines. *Computing*, 62 (1999), 179-187.

[27] Imreh, Cs., On-line scheduling with general machine cost functions, Electronic Notes in Discrete Mathematics, Vol 27, ODSA 2007 Conference, 49-50.

[28] 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.

[29] Nagy-György, J., Imreh, Cs., On-line scheduling with machine cost and rejection, working paper, 2006.

[30] Imreh, B., Imreh, Cs., Kombinatorikus optimalizálás, Novadat, 2006

[31] Y.W. Jiang, Y. He, Preemptive online algorithms for scheduling with machine cost, *Acta Informatica*, 41, 315-240, 2005.

[32] H. Kellerer, V. Kotov, M. G. Speranza, Z. Tuza, Semi on-line algorithms for the partition problem. *Operations Research Letters*, 21 (1997) 235-242.

[33] 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.

[34] J. Sgall, On-line scheduling, On-line algorithms: The state of art, *Lecture Notes in Computer Sciences* 1442, Springer Verlag, 196-231, 1998.

[35] Z. Y. Tan, Y. He, Semi-on-line problems on two identical machines with combined partial information. *Operations Research Letters,* 30 (2002) 408-414.

[36] Wen, J. J., Du, D. L.: Preemptive on-line scheduling for two uniform processors. *Operations Research Letters 23*, 113-116 (1998).

[37] G. C. Zhang, A simple semi-online algorithm for $P2||C_{\max}$ with a buffer. *Information Processing Letters*, 61 (1997), 145-148.