Versenyképes algoritmusok a diszkrét optimalizálásban

Doktori értekezés
tézisei

Galambos Gábor

Alkalmazott Természettudományi Intézet,
Szegedi Tudományegyetem
Szeged

2016

1. Bevezetés

A disszertációban tárgyalt diszkrét optimalizálási feladatok bonyolultságelméleti szempontból két csoportra oszthatók. Az első csoportban két feladattal foglalkozunk.

1.1. Feladat. *Egydimenziós ládapakolási feladatnál adott egy n elemű $L = \{a_1, a_2, \ldots, a_n\}$ lista, ahol a_i, $i = 1, \ldots, n$, jelöli az elem méretét, és $a_i \in (0, 1]$. Adva van legalább n darab egységnyi kapacitású láda. A feladat az, hogy rendeljük hozzá az elemeket a* legkevesebb *számú ládához úgy, hogy az azonos ládához rendelt tárgyak összmérete* legfeljebb 1 *lehet.*

1.2. Feladat. *Párhuzamos gépek ütemezése során adott n munka, $J_1, \ldots, J_n$, és m azonos (sebességű) gép, $M_1, \ldots, M_m$. A J_i munkát pontosan p_i idő alatt lehet elvégezni. A munkák bármilyen sorrendben végrehajthatók, de egy megkezdett munka nem szakítható meg. Rendeljük a munkákat a gépekhez úgy, hogy minimalizáljuk a maximális befejezési időt az ütemezett munkák halmazán.*

Mindkét probléma $\mathcal{NP}$-nehéz [36]. A számítógépek kapacitása az elmúlt évtizedekben rohamosan bővült, sebességük is nagyságrendekkel javult, ezért ma már viszonylag nagyméretű feladatok is megoldhatók egzakt algoritmusok segítségével. A polinomiális időben futó közelítő algoritmusok vizsgálata azonban továbbra is a kutatás középpontjában maradt. Ennek egyik oka az, hogy a közelítő algoritmusok vizsgálata hozzájárul ahhoz, hogy egy probléma elméleti hátterét jobban megérthessük.

A vizsgált problémák másik csoportjába tartozik a szöveg-tömörítési feladat. A disszertációban a szótárakkal történő szövegtömörítési problémára keresünk hatékony algoritmusokat.

1.3. Feladat. *Legyen adott egy szótár, amely (forrás-szó, kód-szó) adatpárokból áll. Az adatpárok véges ABC-ből kiválasztott karaktersorozatokat tartalmaznak. A szótár alapú tömörítési eljárás során a forrás-szöveget olyan karaktersorozatokra bontjuk, amelyek mindegyike megfelel egy forrás-szónak, és ezeket helyettesítjük a szótár megfelelő kód-szavával. A cél az, hogy a forrás-szöveget a könyvtár által meghatározható minimális hosszúságú kódszöveggé kódoljuk.*

Ez a probléma ekvivalens egy megfelelően konstruált súlyozott gráf élein alkalmazott legrövidebb út megkeresésével, és így a probléma polinomiális időben megoldható. Mégis, sok esetben a kezelendő probléma mérete megkívánja a (online) közelítő megoldások alkalmazását.

Számos olyan gyakorlati probléma van, amely modellezhető a disszertációban tárgyalt feladattal vagy annak valamely változatával, ezért a fenti feladatok vizsgálata gyakorlati szempontokból is jelentőséggel bír. Fontos tudni, hogy a feladat polinomiális időben megoldható-e, mert ez alapvetően befolyásolja azt, hogy egzakt vagy közelítő algoritmusokat használunk a megoldáshoz. Általában, ha egy probléma bonyolultságát gyakorlati szempontból vizsgáljuk, akkor pusztán az a tény, hogy az polinomiális időben megoldható, nem feltétlen garancia arra, hogy egy egzakt algoritmus elfogadható időn belül szolgáltat optimális megoldást. Az általunk vizsgált feladatok azonban ebből a szempontból jól viselkednek.

A közelítő algoritmusok alkalmazhatóságát két tényező befolyásolja: ismernünk kell a feladat megoldásához szükséges időt, és rendelkeznünk kell egy hatékony mérőszámmal, amely megmondja, hogy a gyorsaság érdekében milyen mértékben távolodunk el az optimális megoldástól. Egy gyorsabb, hatékonyabb algoritmus általában jobban alkalmazható a gyakorlatban, de azt is érdemes leszögezni, hogy egy algoritmus elméleti gyorsasága semmiképpen

nem biztosíték arra, hogy a valós életből vett feladatokon egy elméletileg lassabb algoritmus ne lenne jobban használható. Ha vannak hatékony és gyors algoritmusaink, akkor egy adott algoritmus-osztályon belül megtalált optimális algoritmus lezárhat kutatási irányokat, és biztosítékokat nyújthat az alkalmazóknak.

A disszertációban azokat az – általunk legfontosabbnak tartott – eredményeket mutatjuk be, amelyeket a fenti feladatok vizsgálata során elértünk.

2. Algoritmusok versenyképessége

Egy közelítő algoritmus versenyképessége az algoritmus-elmélet egyik központi kérdése. Különböző módszereket ismerünk a versenyképesség mérésére. Ezek közül a versenyképességi analízisekkel és a valószínűségszámítási módszerek alkalmazásával végzett elemzésekkel foglalkozunk.

Ha a *versenyképességi elemzés* mellett döntünk, akkor olyan hatékonysági garanciákat várunk el, amelyek érvényesek minden input esetén. Ehhez szükséges az is, hogy olyan szélsőséges példákra is elemzzük az algoritmus viselkedését, amelyek a gyakorlati életben nagyon ritkán – vagy egyáltalán nem – fordulnak elő. Az ilyen "patológikus" példák vizsgálata mégis nagyon fontos, mert ezek felfedhetik az adott algoritmus leggyöngébb pontjait. Legyen A egy tetszőleges közelítő algoritmus, és legyen I egy, az adott problémához tartozó input. $A(I)$ és $\mathrm{OPT}(I)$ rendre jelölje az A algoritmushoz, és az optimumhoz tartozó megoldásokat. A leggyakrabban használt mérőszámok ezen a területen a következők.

2.1. Definíció. Az *abszolút versenyképességi hányados* minimum feladatra a következő.

$$R_A = \sup_I \frac{A(I)}{\mathrm{OPT}(I)}.$$

Legyen C egy olyan valós konstans, amelyre $R_A \leq C$. Ekkor azt mondjuk, hogy az A algoritmus abszolút C-versenyképes.

2.2. Definíció. Az *aszimptotikus versenyképességi hányados* Minimum feladatra a következő.

$$R_A^\infty = \limsup_{k\to\infty} \left\{ \max_I \left\{ \frac{A(I)}{k} \,\middle|\, \mathrm{OPT}(I) = k \right\} \right\}.$$

Maximum feladatra a következőképpen adhatjuk meg a definíciót:

$$R_A^\infty = \liminf_{k\to\infty} \left\{ \min_I \left\{ \frac{A(I)}{k} \,\middle|\, \mathrm{OPT}(I) = k \right\} \right\}.$$

Az aszimptotikus versenyképességi hányadosra használják a következő definíciót is.

2.3. Definíció. Egy minimum feladatra az A algoritmus aszimptotikusan C-versenyképes, ha létezik olyan B valós konstans, amelyre $A(I) \leq C \cdot \mathrm{OPT}(I) + B$ teljesül bármely I inputra.

Valószínűségszámítási elemzéseket akkor tudunk végezni, ha a példáinkhoz tartozó adatok valószínűségi eloszlását ismerjük. Válasszuk a feladathoz tartozó példa inputjait egymástól függetlenül az adott valószínűségi eloszlásból. Ekkor a változók viselkedése elemezhető, és mind az optimum, mind a feladat megoldására használt közelítő algoritmus által szolgáltatott megoldás várható értéke létezik. Ilyenkor az aszimptotikus versenyképességi hányadosok definíciójában a valószínűségi változók várható értékeinek határértékeivel számolunk.

3. A versenyképesség kiszámítása

Az algoritmusok versenyképességének vizsgálatakor felmerül az a kérdés, hogy egy adott algoritmus-osztályhoz tartozó algoritmusok versenyképessége meddig javítható? A következőkben röviden bemutatjuk a különböző ládapakolási feladatok közelítő megoldására használt algoritmusok versenyképességének vizsgálatánál használt eszközöket.

3.1. Konkatenált listák

3.1. Definíció. Egy *online algoritmus* az input elemeit az érkezés sorrendjében pakolja el úgy, hogy az elpakolt elemek többé nem mozdíthatók. Az aktuális elem elpakolásakor az algoritmus semmit nem tud az utána következő elemekről, sem azok száma, sem a méreteik nem ismertek.

3.2. Definíció. Az $L_1, \ldots, L_k$ listák *konkatenációján* értjük azt a listát, amelyben az L_i lista elemeit az L_{i+1} lista elemei követik, $1 \leq i \leq k-1$. A konkatenált listát $(L_1 \ldots L_k)$-val jelöljük.

Online ládapakolási algoritmusok esetén az alsó korlátok keresése során a leggyakrabban alkalmazott módszer a következő: Válasszuk az $L_1, \ldots, L_k$ listákat úgy, hogy egy listán belül az elemek méretei legyenek azonosak. Határozzuk meg a minimálisan szükséges ládák számát az $(L_1 \ldots L_j)$, $1 \leq j \leq k$, konkatenált listák elpakolása során, és számítsuk ki ugyanezen listákra az A algoritmus által elhasznált ládák számát is. Ekkor a

$$\min_A \max_j \left(\frac{A(L_1 \ldots L_j)}{\mathrm{OPT}(L_1 \ldots L_j)} \mid j = 1, \ldots, k \right) \tag{1}$$

egy alsó korlát az adott algoritmus-osztályon belül.

3.2. Sylvester sorozat

Ládapakolási algoritmusok esetén alsó korlát meghatározására használt konstrukciók hosszú időn keresztül egy speciális sorozatot használtak. A sorozatot az $r = 1$ esetre először Sylvester alkalmazta 1880-ban egy számelméleti probléma megoldására [57]. Ezt a sorozatot általánosítottam [28]-ban tetszőleges $r \geq 1$ egész értékekre.

3.3. Definíció. Legyenek adva $k > 1$ és $r \geq 1$ egészek. Ekkor az általánosított $m_1^r, \ldots, m_k^r$ Sylvester sorozatot a következő rekurzió adja meg.

$$m_1^r = r + 1, \quad m_2^r = r + 2, \quad m_j^r = m_{j-1}^r (m_{j-1}^r - 1) + 1, \quad j = 3, \ldots, k.$$

A fenti rekurzió az $r = 1$ esetben a Sylvester sorozat elemeit definiálja.A disszertációban gyakran használjuk a következő definíciót.

$$h_\infty(r) = 1 + \sum_{i=2}^{\infty} \frac{1}{m_i^r - 1}.$$

$h_\infty(r)$ első néhány értéke: $h_\infty(1) \approx 1.6910$, $h_\infty(2) \approx 1.4231$, $h_\infty(3) \approx 1.3023$.

3.3. Pakolási minták elemzése

A (1) meghatározásához többször használjuk a *pakolási mintákat.*

3.4. Definíció. Tegyük fel, hogy az $L = (L_1 \dots L_k)$ konkatenált lista elemeit az A algoritmus úgy pakolja ládákba, hogy az L_j listából p_j elemet helyez el egy ládában $(1 \le j \le k)$. Ekkor a $p = (p_1, p_2, \dots, p_k)$ vektort *pakolási mintának* nevezzük, ha $\sum_{j=1}^{k} p_j a_j \le 1$.

Legyenek adva a $c_j > 0$, $j = 1, 2, \dots, n$, egészek. Egy adott n egészhez legyen $n_j = c_j n$, ahol $j = 1, \dots, k$. Az L_j^n részlista tartalmazzon n_j azonos méretű tárgyat. Jelöljük P-vel az összes pakolási minta halmazát. Legyen

$$P_i = \{p \in P \mid p_i > 0,\ p_j = 0,\ \text{ha } j < i\},\ i = 1, \dots, k. \tag{2}$$

Egy alsó korlát értékének kiszámításához több helyen is használjuk a következő tételt.

3.3.1. Tétel (J. Balogh, J. Békési, G. Galambos, [4]). *Legyenek α_j és β_j $1 \le j \le k$-re olyan pozitív egészek, amelyekre teljesül, hogy bármely $p \in P_i$, $i = 1, 2, \dots, k$, esetén*

$$\sum_{j=i}^{k} \beta_j p_j \le \sum_{j=i}^{k} \alpha_j. \tag{3}$$

Ekkor bármely A online algoritmusra

$$R_A^\infty \ge \max_{1 \le j \le k} \limsup_{n \to \infty} \frac{A(L_1^n L_2^n \dots L_j^n)}{\mathrm{OPT}(L_1^n L_2^n \dots L_j^n)} \ge \frac{\sum_{j=1}^{k} \beta_j c_j}{\sum_{j=1}^{k} \alpha_j U_j}. \tag{4}$$

3.4. Súlyfüggvények alkalmazása

Az A algoritmus által felhasznált ládák számának felső becslésére a súlyfüggvényt Johnson [41] alkalmazta először. Legyen $L = \{a_1, a_2, \dots, a_n\}$ egy n elemű lista. Pakoljuk el az L lista elemeit az A algoritmussal. Legyen $W(x) : (0, 1] \to R^+$ egy olyan súlyfüggvény, amelyre bármely $S \subseteq L$ esetén $W(S) = \sum_{a_i \in S} W(a_i)$. Ekkor, ha létezik olyan C konstans, amelyre

(i) $W(L) \le C \cdot \mathrm{OPT}(L)$, és

(ii) $A(L) \le W(L) + B$

teljesül bármely L listára, és $B < \infty$, akkor $R_A^\infty \le C$.

4. A disszertáció eredményei

4.1. Egydimenziós online ládapakolás

A gyakorlati életben sűrün előfordul, hogy az elhelyezendő tárgyak méretei sokkal kisebbek, mint a ládák méretei. Az a tapasztalat, hogy az ilyen esetekben az algoritmusok versenyképessége szignifikánsan javul. Érdemes tehát megvizsgálni az algoritmusokat a tárgyak méretére tett korlátozások mellett is.

4.1. Definíció. Az egydimenziós (1D) ládapakolási problémát *r-parametrikusnak* nevezünk, ha az elemek méretei a $(0, \frac{1}{r}]$ intervallumba esnek, valamely $r \ge 1$ pozitív egész értékre.

4.1.1. Alkalmazás. *Egy olyan egydimenziós darabolási feladat, amelyben nagyobb rudakból megrendeléseket kell kivágni úgy, hogy a nem felhasználható anyagmennyiség (selejt) minimális legyen, átfogalmazható egydimenziós ládapakolási feladattá. A hirdetési idők elhelyezése a rendelkezésre álló időkeretben szintén egy ládapakolási feladattal írható le. Ez a modell használható a banki átutalások ütemezésekor is, ha az egy napra ütemezhető átutalások összege korlátozott.*

Online algoritmusokra Yao [60] 3/2 alsó korlátot bizonyított, majd Liang [46] javított ezen úgy, hogy $k \geq 4$ listát vizsgált. Az általa bizonyított alsó korlát $1.5364\ldots$. Felhasználva a Sylvester sorozat általánosított definícióját a parametrikus esetet vizsgáltam [29]-ben. Lineáris programozási modellre alapozva [58]-ben van Vliet bebizonyította, hogy minden A online algoritmusra $R_A^\infty \geq 1.5401\ldots$.

Sokáig úgy tűnt, hogy van Vliet javítása azért lehetséges, mert az LP modell pontosabb, mint a tisztán kombinatorikus módszereken alapuló bizonyítás. A disszertáció 2. fejezetében először megmutatjuk, hogy a súlyok helyes választásával az [58]-ban megadott alsó korlátok kombinatorikus eszközökkel is elérhetők.

Az a kérdés közel 20 évig nyitva maradt, hogy adható-e élesebb alsó korlát az online algoritmusokra más sorozatok alkalmazásával. A parametrikus esetet vizsgálva [3]-ban a következő sorozatot definiáltuk. Bármely $r \geq 1$ egész értékre legyenek

$$\begin{aligned} b_{1,r} &= r+1, & b_{2,r} &= r+2, \\ b_{3,r} &= b_{1,r}b_{2,r}+1, & b_{j,r} &= b_{3,r}^{j-2}, \quad 4 \leq j \leq k-1, \\ b_{k,r} &= b_{1,r}b_{2,r}b_{3,r}^{k-3}+1. \end{aligned}$$

A disszertációban megmutatjuk, hogy az α_j és a β_j értékek választhatók úgy, hogy a 3.3.1 Tétel feltételei teljesüljenek, és a megadott értékek segítségével bebizonyítható a következő állítás.

4.1.1. Tétel (J. Balogh, J. Békési, G. Galambos, [3]). *Legyen r egy pozitív egész, és tekintsük a parametrikus feladatot. Ekkor bármely A online egy-dimenziós ládapakolási algoritmusra*

$$R_A^\infty \geq \frac{r^6+8r^5+29r^4+60r^3+75r^2+55r+20}{r^6+7r^5+22r^4+40r^3+45r^2+33r+13}. \tag{5}$$

Ez a korábbi korlátok javítását eredményezi, és pl. az $r=1$ esetben $R_A^\infty \geq 1.5403\ldots$.

4.2. Egydimenziós félig-online ládapakolás

4.2. Definíció. Ha egy online algoritmus alkalmazása során azt feltételezzük, hogy az elemek méreteik szerint rendezve érkeznek, vagy az elemek elpakolása során megengedett korlátos számú láda tartalmának átrendezése, vagy egy elem elpakolása előtt korlátozott számú elem átpakolható a nyitott ládák között, akkor *félig-online algoritmusokról* beszélünk.

4.2.1. Rendezett listák pakolása

Az már elég korán kiderült, hogy az online algoritmusok aszimptotikus versenyképessége szignifikánsan javul, ha azt feltételezzük, hogy az elemek csökkenő sorrendben érkeznek.

Ha az input lista elemeinek sorrendjére nem teszünk feltételeket, akkor a legjobb ismert algoritmust Seiden publikálta ([53]). Az általa adott algoritmus aszimptotikus versenyképességi hányadosa $R_A^\infty \leq 1.5889$. Rendezett listákra a legjobb ismert online algoritmus a

Modified First Fit Decreasing [41], amelyre $R^{\infty}_{\mathrm{MFFD}} = \frac{71}{60} = 1.1833\ldots$. Az ilyen, előre rendezett listákat pakoló algoritmusokra hosszú ideig a legjobb alsó korlát $\frac{8}{7}$ volt, amelyet [23]-ban bizonyítottunk. A disszertációban – a [3]-ban alkalmazott technika segítségével – egy javítást adunk erre az alsó korlátra.

4.2.1. Tétel (J. Balogh, J. Békési, G. Galambos, [4]). *Érkezzenek egy online pakolási feladatban a listák elemei monoton csökkenő sorrendben. Ekkor bármely A online algoritmusra $R^{\infty}_{A} \geq \frac{54}{47} = 1.1489\ldots$.*

4.2.2. Átpakolás korlátozott számú ládák között

4.3. Definíció. Amikor egy algoritmus egy ládába először pakol elemet, akkor azt mondjuk, hogy a láda *aktívvá* (nyitottá) válik. A láda mindaddig aktív marad, amíg az algoritmus nem dönt annak bezárásáról. Az egyszer bezárt láda az adott lista elemeinek elpakolása során nem aktíválható többé.

4.4. Definíció. Egy ládapakolási algoritmust *k-tárkorlátosnak* nevezünk, ha minden a_i elem elpakolásakor maximum k aktív láda közül választhatunk.

4.2.1. Alkalmazás. *Tárkorlátos pakolással modellezhető a kamionok rakodása egy olyan depó esetén, ahol több berakodó hely van, és a kamionok közötti átpakolás a rakodás során megengedett.*

A k-tárkorlátos algoritmusokat széles körben elemezték. Lee és Lee [45] bebizonyították, hogy bármely k-tárkorlátos online algoritmusra $R^{\infty}_{A} \geq h_{\infty}(1)$. [28]-ban bebizonyítottam, hogy az állítás igaz az r-parametrikus esetre is, $R^{\infty}_{A}(r) \geq h_{\infty}(r)$. A cikkben megmutattam, hogy létezik olyan k-korlátos algoritmus, amelyre $h_{\infty}(r)$, ha $k \to \infty$.

Az nyitott kérdés volt, hogy véges sok aktív ládával el lehet-e érni a $h_{\infty}(r)$ értékét. A disszertáció 3.2. fejezetében megadunk egy három aktív ládát használó algoritmust, amellyel a fenti korlát az $r = 1$ esetben elérhető. Az algoritmust Repacking-3 algoritmusnak (REP_3) neveztük. Az algoritmus elemzésekor súlyfüggvény technikát alkalmazunk. Legyen $x \in L$. Ekkor

$$W(x) = \begin{cases} x + \frac{1}{m_{i+1}-1}, & \text{for } \frac{1}{m_i} < x \leq \frac{1}{m_i - 1}, \text{ and } 1 \leq i \\ \frac{m_i+1}{m_i} \cdot x, & \text{for } \frac{1}{m_{i+1}-1} < x \leq \frac{1}{m_i}, \text{ and } 1 \leq i. \end{cases}$$

A súlyfüggvényről belátható, hogy egy L lista elpakolásakor bármilyen pakolás esetén az egy ládába kerülő elemek súlyának összege nem lehet nagyobb, mint $h_{\infty}(1)$. Így $W(L) \leq h_{\infty}(1)\,\mathrm{OPT}(L)$. A REP_3 algoritmus a következő.

(1) Vegyük az input lista következő x elemét, és rakjuk x-et egy üres aktív ládába.

(2) Pakoljuk át a három aktív láda tartalmát úgy, hogy vagy keletkezzen egy üres láda, vagy legyen legalább egy olyan láda, amelyben az elemekhez tartozó súlyok összege legalább 1.

(3) Zárjunk be minden olyan ládát, amelyben az elemek súlyának az összege legalább 1, és nyissunk minden bezárt láda helyett egy üres ládát. Goto (1).

Az algoritmus aszimptotikus versenyképességi hányadosának kiszámításához a következő tétel bizonyítása szükséges.

4.2.2. Tétel (G. Galambos, G. J. Woeginger, [33]). *Tegyük fel, hogy egy L lista elemeit a* REP_3 *algoritmussal pakoljuk el. Legyen* BIN_1, BIN_2, BIN_3 *a három aktív láda. Ekkor lehetséges a három aktív láda tartalmát átpakolni úgy, hogy vagy keletkezzen egy üres láda, vagy legyen benne legalább egy láda, amelyben az elemek súlyának összege legalább* 1.

A 4.2.2 Tétel állításából következik, hogy bármely L listára teljesül, hogy $\mathrm{REP}_3(L) \leq W(L) + 3$. Mivel $W(L) \leq h_\infty(1)\,\mathrm{OPT}(L)$, így

$$\mathrm{REP}_3(L) - 3 \leq W(L) \leq h_\infty(1)\,\mathrm{OPT}(L),$$

ami a [45]-ben adott alsó korláttal kombinálva szolgáltatja, hogy $R^\infty_{\mathrm{REP}_3} = h_\infty(1)$.

Mivel $k = 3$-ra a REP_3 optimális, ezért kézenfekvő volt a kérdés, hogy $k = 2$-re is létezik-e optimális algoritmus. A cikkben beláttuk, hogy a REP_3 algoritmusnál alkalmazott technika a $k = 2$ esetre nem használható. Mivel Csirik és Johnson [25]-ben vizsgált egy olyan 2-tárkorlátos algoritmust, amely nem használ átpakolást és az aszimptotikus versenyképessége 17/10, a megmaradt eltérés nagyon kicsi. Ennek ellenére a probléma a mai napig nyitott.

4.2.3. Korlátozott számú elem átpakolása

4.5. Definíció. Ha egy A félig-online algoritmus $k < \infty$ elem átpakolását engedi meg minden lépésben, akkor A-t *k-elemkorlátos algoritmusnak* nevezzük.

A disszertáció 3.3. fejezetében egy HR-k-val jelölt algoritmust definiálunk. Az algoritmusról először azt bizonyítjuk be, hogy egy lépésben maximum k elemet pakol át, majd belátjuk azt, hogy az algoritmus idő-bonyolultsága $O(kn)$. Végezetül bebizonyítjuk a következő tételt.

4.2.3. Tétel. (J. Balogh, J. Békési, G. Galambos, G. Reinelt, [6]). *Bármely rögzített $k \in N^+$ egészre*

$$\left(\frac{3}{2} + \frac{b_k}{1-b_k}\right)\left(\frac{1-2b_k}{(1-b_k)^2}\right) \leq R^\infty_{\text{HR-k}} \leq \frac{3}{2} + \frac{b_k}{1-b_k}.$$

A felső korlát bizonyításához súlyfüggvényes technikát használunk, míg az alsó korlátot egy konstrukció segítségével látjuk be.

A k-elemkorlátos algoritmusok versenyképességére egy $\frac{4}{3}$ alsó korlátot adott Ivkovič és Lloyd [40]-ben. Ezt javítottuk meg az [5] cikkben. Az ott publikált eredmények a következő tételben foglalhatók össze.

4.2.4. Tétel. (J. Balogh, J. Békési, G. Galambos, G. Reinelt, [5]). *Bármely k-elemkorlátos online algoritmusra*

$$R^\infty_A \geq 1 - \frac{1}{W_{-1}\left(\frac{-2}{e^3}\right) + 1} \approx 1.3871,$$

ahol $W_{-1}(x)$ a Lambert W függvény $W(x) \leq -1$ ágát jelöli.

A tétel bizonyítása során különböző eszközöket alkalmazunk: először egy LP modellt állítunk fel, amelynek megoldásához a lineáris algebra tételeit használuk, és végül egy nemlineáris optimalizációs feladat megoldásával kapjuk a keresett alsó korlátot.

4.3. Kétdimenziós téglalap-pakolás

4.3.1. Feladat. *Adott egy $L = \{a_1, a_2, \ldots, a_n\}$ n elemű lista, amelyben az elemek méreteit a $(w(a_i), h(a_i))$ rendezett párok definiálják. Adva vannak téglalap alakú ládák W és H méretekkel. (Feltehetjük, hogy $W = H = 1$, és $w(a_i) \leq 1$, $h(a_i) \leq 1$.) A cél az, hogy pakoljuk el a kis téglalapokat minimális számú ládába oly módon, hogy a kis téglalapok oldalai maradjanak párhuzamosak a ládák megfelelő oldalaival (a 90 elforgatás nem megengedett), és az elemek a pakolás során nem fedhetik egymást.*

4.3.1. Alkalmazás. *Ehhez a feladathoz a bútorlapok és az üvegtáblák szabását említik gyakorlati alkalmazásként. Érdemes azonban megjegyezni, hogy üvegtáblák esetében a – derékszögű – forgatás megengedett, ami egy kissé bonyolítja a modellt.*

A fenti feladat közelítő megoldására az első – offline – algoritmust Chung, Garey és Johnson elemezték [15]. Az általuk vizsgált *Hybrid-First Fit* (HFF) algoritmusra bebizonyították, hogy $\frac{182}{90} \leq R^{\infty}_{\mathrm{HFF}} \leq \frac{17}{8}$. A *Hybrid-Next Fit* (HNF) tárkorlátos algoritmust [27]-ben elemezve azt kaptuk, hogy $R^{\infty}_{\mathrm{HNF}} = 3.38$.

Az online algoritmusokra az első eredményt Coppersmith és Raghavan [18]-ban publikálták. Az általuk vizsgált algoritmusokra $R^{\infty}_{A} \leq 3.25$. Később Csirik, Frenk és Labbé bebizonyította ([22]), hogy a versenyképesség a kétdimenziós online algoritmusokra 3.1666-ra javítható. A *Harmonic Fit* algoritmus kétdimenziós kiterjesztésével kapott tárkorlátos algoritmusokat [47]-ben elemeztek, és $R^{\infty}_{A} = 2.86$ aszimptotikus versenyképességet bizonyítottak. A ma ismert legjobb felső korlát 2.66013, lásd [54].

Az első – nem triviális – alsó korlátot egy 4-listás konstrukcióval [30]-ban bizonyítottam.

4.3.1. Tétel (G. Galambos ([30])). *Bármely A 2D online algoritmusra $R^{\infty}_{A} \geq 1.6$.*

A tétel azt mutatta meg, hogy a 2D online algoritmusok nem lehetnek olyan jók, mint az 1D online algoritmusok. További javítást sikerült bizonyítani [32]-ben.

4.3.2. Tétel (G. Galambos, A. van Vliet [32]). *Bármely 2D online algoritmusra*

$$R^{\infty}_{A} \geq \lim_{k \to \infty} \frac{4 \sum_{j=1}^{k} \frac{j}{m_j - 1}}{4 \sum_{j=1}^{k} \frac{1}{m_j - 1} - 1} = 1,8028\ldots.$$

Később, az LP-technika alkalmazásával van Vliet bizonyította [59], hogy minden kétdimenziós online algoritmusra $R^{\infty}_{A} \geq 1.851\ldots$.

4.4. Valószínűségszámítási módszerek alkalmazása

A disszertáció 5. fejezetében azokat az eredményeket mutatjuk be, amelyeket a különböző ládapakolási algoritmusok valószínűségszámítási módszerekkel történő elemzése során értünk el. A vizsgálataink során azt feltételezzük, hogy az elemeket egymástól függetlenül választjuk, és a méretek eloszlásfüggvénye ismert. A disszertációban mindig azt feltételezzük, hogy a méretek egyenletes eloszlást követnek a $(0, 1]$ intervallumon. A korábbiaktól eltérően, ebben a fejezetben egy A algoritmus ill. egy optimális pakolás által felhasznált ládák számát $A(n)$ ill. $\mathrm{OPT}(n)$ jelöli. Az ismert tény, hogy egyenletes eloszlás esetén $\lim_{n \to \infty} \frac{\mathrm{E}(\mathrm{OPT}(n))}{n/2} = 1$.

4.4.1. Az egydimenziós ládapakolási feladat

4.4.1. Algoritmus. *A Next Fit Decreasing* (NFD) *algoritmus az L listához tartozó elemeket először átindexeli úgy, hogy $a_1 \geq a_2 \geq \ldots \geq a_n$. Az így kapott listát ezután ebben a sorrendben helyezi el ládákban. Új ládát akkor nyit, ha az éppen nyitott ládában nincs elég hely arra, hogy elhelyezze benne az aktuális elemet. Egy új láda megnyitásakor az addig nyitott ládát lezárja.*

Az NFD algoritmussal elpakolt listák által felhasznált ládák várható értékének becsléséhez az r-paraméterű *Sliced* NFD algoritmust használjuk (SNFD_r). Legyen $r \geq 2$ egész. Az algoritmus az NFD szabályai szerint pakolja az elemeket mindaddig, amíg $a_i > 1/r$. A maradék elemeket pedig r-es csoportonként rakja külön ládába. Az világos, hogy

$$\mathrm{SNFD}_r(n) \geq \mathrm{NFD}(n), \qquad \lim_{r\to\infty} \mathrm{SNFD}_r(n) = \mathrm{NFD}(n),$$

ahol $r \geq 2$ és $n \geq 1$. Felhasználva azt, hogy az SNFD_r algoritmus valószínűségszámítási eszközökkel jobban elemezhető, mint az NFD, a következő tételt bizonyítjuk be.

4.4.1. Tétel. (J. Csirik, J.B.G. Frenk, A. Frieze, G. Galambos, A.H.G. Rinnooy Kan, [20]). *Tekintsünk egy L listát, és legyenek az a_i elemek független, a $(0,1]$ intervallumon egyenletes eloszlásúak. Pakoljuk el az L lista elemeit az* NFD *algoritmussal. Ekkor*

$$\lim_{n\to\infty} \frac{\mathrm{E}(\mathrm{NFD}(n))}{n/2} = 2\left(\frac{\pi^2}{6} - 1\right) = 1.289\ldots.$$

Az NFD pakolás által felhasznált ládák számának a várható értéktől való eltérésére érvényes a következő becslés.

$$\Pr\left\{\left|\mathrm{NFD}(n) - \mathrm{E}\Big(\mathrm{NFD}(n)\Big)\right| \geq nt\right\} \leq 2\exp\left(-2n\Big(t - \frac{3}{n^{2/3}}\Big)\right).$$

Az NFD algoritmus elemzését a következő a következő centrális határeloszlás tétellel zárjuk.

4.4.2. Tétel ([20]). *Bármely x valós értékre*

$$\lim_{n\to\infty} \Pr\left\{\frac{\mathrm{NFD}(n) - n(\pi^2/6 - 1)}{\sqrt{n}\sigma_\infty} \leq x\right\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-y^2/2} dy,$$

ahol $\sigma_\infty^2 = \sum_{i=1}^{\infty} i^{-3} + \frac{\pi^2}{6} - \frac{\pi^4}{36} = 0.14118\ldots$.

4.4.2. Kétdimenziós ládapakolási feladat

A kétdimenziós téglalalap pakolási feladatot fentebb definiáltuk. [27]-ben elemeztük a *Hybrid Next Fit* (HNF) algoritmust. Az algoritmust a következőképpen írhatjuk le.

(1) Legyen p az L lista egy eleme. Rendezzük az elemeket a $h(p)$ magasságuk szerint nemnövekvő sorrendbe. Vegyük az első elemet a rendezett listából, és helyezzük el az első ládában a bal alsó sarokban. Azt a $h(p)$ magasság által kialakított téglalap alakú területet a ládán belül, amelynek baloldali részét $w(p)$ lefed, nevezzük a p által megnyitott *blokknak.*

(2) Vegyük az L következő elemét, és az aktuálisan nyitott láda utoljára megnyitott blokkjába próbáljuk elhelyezni az elemet. Ha nem lehet elhelyezni, akkor nyissunk egy új blokkot a ládán belül. Ha ez sem lehetséges, akkor zárjuk le az aktuális ládát, és nyissunk egy új, üres ládát egy új, első blokkal.

(3) Ha van még elhelyezetlen elemünk, akkor ugorjunk a (2) pontra, különben vége az eljárásnak.

A [27] cikkben az algoritmus versenyképességi elemzése mellett vizsgáltuk annak viselkedését véletlen inputra is. Igaz a következő tétel.

4.4.3. Tétel (G. Galambos, J.G.B. Frenk, [27]).

$$\lim_{n\to\infty}\frac{\mathrm{E}(\mathrm{HNF}(n))}{n}=\lim_{n\to\infty}\frac{\mathrm{E}(\mathrm{NF}(n))}{n}\lim_{n\to\infty}\frac{\mathrm{E}(\mathrm{NFD}(n))}{n}.$$

Mivel

$$\lim_{n\to\infty}\mathrm{E}(\mathrm{OPT}(n))=\frac{1}{4}\quad\text{és}\quad\lim_{n\to\infty}\frac{\mathrm{E}(\mathrm{NF}(n))}{n}=\frac{2}{3}$$

(lásd [50]), ezért

$$\lim_{n\to\infty}\frac{\mathrm{E}(\mathrm{HNF}(n))}{\mathrm{E}(\mathrm{OPT}(n))}=\frac{2n}{3}\left(\frac{\pi^2}{6}-1\right)\frac{4}{n}=\frac{8}{3}\left(\frac{\pi^2}{6}-1\right)=1.7153\ldots.$$

4.4.3. Az egydimenziós ládalefedési feladat

4.4.1. Feladat. *A ládalefedési feladatban adott egy $L=\{a_1,\ldots,a_n\}$ n elemű lista, ahol $(a_i\in(0,1),\quad i=1,\ldots,n)$. Rendeljük az elemeket maximális számú egységnyi kapacitású ládához úgy, hogy az azonos ládához rendelt elemek összmérete legalább* 1.

4.4.1. Alkalmazás. *Ezzel a feladattal modellezhető a mélyhűtött áruk csomagolása, amennyiben előírt minimális súly tartozik a csomagolt termékhez. Ugyancsak ez a modell alkalmazható abban a gazdasági modellben, amely egy recessziós időszakban úgy allokál feladatokat maximális számú üzemhez, hogy a feladatok kumulált intenzitásának üzemenként el kell érnie egy minimális szintet.*

A problémát először Assmann és munkatársai vizsgálták [2]. Az elemzéseik elsősorban a versenyképességi vizsgálatokra vonatkoztak. A disszertáció 5.3. fejezetében azokat az eredményeket mutatjuk be, amelyeket a [19] és [21] cikkekben ismertettünk. Az első tétel az optimális pakolás várható értékére egy – a triviálisnál élesebb – felső korlátot ad.

4.4.4. Tétel. (J. Csirik, J.B.G. Frenk, G. Galambos, A.H.G. Rinnooy Kan, [21].) *Tegyük fel, hogy a lista elemei egymástól függetlenül vannak választva, és méreteik a $(0,1]$ intervallumon egyenletes eloszlást követnek. Ekkor*

$$\mathrm{E}(\mathrm{OPT}(n))\le\frac{n}{2}-\sqrt{\frac{n}{32\pi}}.$$

A disszertáció következő részében megmutatjuk, hogy a megadott felső korlát éles. Ennek bizonyításra definiáljuk a *Párosítási Algoritmust* (PA).

4.4.2. Algoritmus. *A Párosítási Algoritmus először rendezi az elemeket méretük szerint nemnövekvő sorrendbe. Ezután a legnagyobb elemhez keresi meg azt a legkisebb elemet, amellyel együtt lefednek egy ládát. Ha nincs ilyen elem, akkor a NF szabállyal pakolja el a lista maradék részét.*

4.4.5. Tétel ([21]). *Tegyük fel, hogy a lista elemei egymástól függetlenül vannak választva, és a méreteik a* $(0, 1]$ *intervallumon egyenletes eloszlást követnek. Ekkor*

$$\mathrm{E}(\mathrm{PA}(n)) \geq \frac{n}{2} - \left(\frac{n}{2\pi}\right)^{1/2} - \alpha,$$

ahol α *egy valós konstans.*

A disszertációban további algoritmusokat is vizsgálunk. Az NF algoritmus elemzése során a következő állítást bizonyítjuk be.

$$\lim_{n\to\infty} \frac{\mathrm{E}(\mathrm{NF}(n))}{n} = \frac{1}{e} = 0.3690\ldots. \tag{6}$$

A NFD algoritmusra belátjuk, hogy

$$\lim_{n\to\infty} \frac{\mathrm{E}(\mathrm{NFD}(n))}{n} = 2 - \frac{\pi^2}{6} = 0.3551\ldots.$$

Ez azt jelenti, hogy a ládalefedési feladat esetén az NF algoritmus hatékonyságának várható értéke – egyenletes valószínűségi eloszlás esetén – jobb, mint az NFD algoritmusé.

4.5. Párhuzamos gépek ütemezése

4.5.1. Feladat. *Párhuzamos gépek ütemezése során adott* n *munka,* $J_1, \ldots, J_n$*, és* m *azonos (sebességű) gép* $M_1, \ldots, M_m$*. A* J_i *munkát pontosan* p_i *idő alatt lehet elvégezni. A munkák bármilyen sorrendben végrehajthatók, de egy megkezdett munka nem szakítható meg. Rendeljük a munkákat a gépekhez úgy, hogy minimalizáljuk a maximális befejezési időt az ütemezett munkák halmazán.*

A disszertáció 6. fejezetében az online problémát vizsgáljuk: a munkák egyesével érkeznek, és azonnal el kell dönteni, hogy melyik gépen hajtjuk azokat végre. Ha egy munkát egy géphez rendeltünk, az többé nem "mozgatható" át egy másik gépre.

1969-ben Graham [30] egy egyszerű – *List Scheduling* (LS)– algoritmust javasolt az online feladat megoldására: Az LS algoritmus az aktuális munkát ahhoz a géphez rendeli, amelyen a munka végrehajtása leghamarabb megkezdhető. Graham megmutatta, hogy az LS abszolút versenyképességi hányadosa $2 - 1/m$. Faigle, Kern és Turán [26]-ban bebizonyították, hogy az LS algoritmus az $m = 2$ és $m = 3$ esetre nem javítható, és az $m \geq 4$ esetre egy $1 + \sqrt{2}/2 \approx 1.707$ alsó korlátot szolgáltattak. Hosszú ideig tartotta magát az az elv, hogy az LS algoritmus alapgondolata – minden lépésben a lehető legegyenletesebben terheljük a gépeket – nem javítható, és az új algoritmusokat is ezen elv mentén konstruálták. A kísérletek eredménytelenek voltak, és több mint 20 évig nem publikáltak az LS-nél jobb algoritmust.

1993-ban a [34] cikkben egy új elv mentén kerestünk jobb algoritmust. Legyen $m \geq 4$, és legyen adva két valós szám, $\alpha(m)$ és $\beta(m)$, ahol $0 \leq \alpha(m) \leq 1/3$ és $1 \leq \beta(m) \leq 5/4$. Azt is feltételezzük, hogy $(\beta(m)+1)/\beta(m)^3 > \alpha(m)+1$. (A disszetációban és itt is $\alpha(m)$ és $\beta(m)$ helyett az egyszerűbb α és β jelölést követjük.) Az x, y nemnegatív valós számokra definiáljuk a következő szimmetrikus relációt.

4.6. Definíció. Azt mondjuk, hogy x *hasonló* y-hoz ($x \sim y$,) ha

$$\frac{y}{\beta} \leq x \leq \beta y.$$

Az S nemnegatív valós számok halmaza akkor és csak akkor *hasonló*, ha S bármely két eleme hasonló. A hasonló halmaz jelölése: $\sim S$.

Jelölje $L_1, L_2, \ldots, L_m$ a gépek terhelését, azaz azt az időpontot, ameddig a gép foglalt. Az általunk vizsgált – *Refined List Scheduling*, (RLS) – algoritmus a következő.

(1) Rendezzük át a gépeket úgy, hogy legyen $L_1 \leq L_2 \leq \ldots \leq L_m$, és legyen x a következő munka.

(2) Ha $\not\sim (L_1 + x, L_2, L_3, \ldots, L_m)$ akkor rendeljük x-t az M_1 géphez, és goto (1).

(3) Különben, ha $L_1 > \alpha L_m$ akkor ütemezzük x-t az M_2-höz, és goto (1).

(4) Különben, ha $L_1 \leq \alpha L_m$ akkor

  (4.1) Rendeljük x-t az M_1 géphez.

  (4.2) Mindaddig, amíg $L^{\min} \sim L^{\max}$ tegyünk minden új munkát az M_1-re.

  (4.3) Goto (1).

Az algoritmus attól függően, hogy az egyes gépek milyen mértékben vannak terhelve az aktuális munka elpakolásakor, más-más gépet választ a munka elhelyezésére. Jelölje C^* az adott munka elhelyezésekor az optimális befejezési időt. Az elemzés során α és β függvényében felső korlátokat adunk meg az $RLS(m)/C^*$ aktuális értékeire. Ezek minimuma szolgáltatja a keresett felső korlátot az R_{RLS} abszolút versenyképességi hányadosra. A disszertációban belátjuk, hogy a minimumot egy – a β-ban negyedfokú – függvény zérushelyeiben kaphatjuk meg. Ennek segítségével belátjuk, hogy ha $m \geq 4$, akkor $R_{RLS} < 2 - \frac{1}{m}$, és a különbséget ε_m-mel jelőljük. Így a következő tételt kapjuk.

4.5.1. Tétel (G. Galambos, G. Woeginger, [34]). *Legyen $m \geq 4$. Ekkor $R_{RLS} \leq 2 - \frac{1}{m} - \varepsilon_m$, ahol $\varepsilon_m > 0$.*

A fenti tételben ha $m \to \infty$, akkor $\varepsilon_m \to 0$, azaz határértékben az RLS és az LS algoritmusok abszolút versenyképessége megegyezik. A cikk eredménye több kutatót inspirált. Felhasználva a cikkben definiált "hasonlóság" elvét, rövid időn belül több javítást is publikáltak. Ezek közül itt megemlítjük a [7], [8], [14], [43] és [59] cikkeket. A legjobb algoritmusok versenyképessége ma már minden $m \geq 4$ értékre jobb az LS algoritmus versenyképességénél.

4.6. Egy open-shop feladat bonyolultsága

4.6.1. Feladat. *Legyen adva m gép, $M_1, \ldots, M_m$, és n munka, $J_1, \ldots J_n$. A J_i munkának m művelete van, $O_{i1}, \ldots, O_{im}$, $i = 1, 2, \ldots, n$. Az O_{ij} műveletet az M_j gépen kell végrehajtani, és p_{ij} ideig tart. A műveletek nem megszakíthatók. Egy gép egy időben legfeljebb egy*

munkát végezhet. Az azonos munkához tartozó műveletek bármilyen sorrendben végrehajthatók. A feladat az, hogy rendeljük a gépeket a munkákhoz úgy, hogy a legutoljára befejezett művelet a legkorábban érjen véget. (Más célfüggvény is lehetséges: ilyen például a maximális késések minimalizálása, a késő munkák számának a minimalizálása.)

A feladat gyakorlati alkalmazhatóságát a következő példával illusztrálhatjuk (lásd [37]).

4.6.1. Alkalmazás. *Tekintsünk egy autójavító műhelyt, ahol az egyes speciális feladatokat különböző épületekben végzik el. Egy beérkező autón javításokat kell végrehajtani. Legyenek ezek a következők: olajcsere, a gumik ellenőrzése, karosszéria-munkák, tuningolás. Ezek egymástól függetlenül elvégezhetők, de a műszerigényesség miatt más-más épületben kell elvégezni a műveleteket, ezért párhuzamosan nem hajthatók végre. Ütemezzük a munkákat úgy, hogy a javításra leadott gépkocsik a leggyorsabban elkészüljenek.*

Az egységnyi hosszúságú végrehajtási időkkel (UET) rendelkező open shop problémák a fenti feladatok egy speciális esetét alkotják. Ebben az esetben $p_{ij} = 1$ minden $1 \leq i \leq n$ és $1 \leq j \leq m$ értékre. Néhány kivételtől eltekintve a tetszőleges műveleti idővel rendelkező open shop problémák $\mathcal{NP}$-nehezek, ugyanakkor majdnem minden UET open shop probléma polinomiális időben megoldható. Egy egységes módszert ismertettek [12]-ben arra, hogy miként lehet polinomiális algoritmusokat készíteni az open shop feladatokra. Egy olyan eset volt, amelyet az általuk javasolt módszerrel nem lehetett kezelni:

4.6.2. Feladat. *Adott $m = 2$ gép. Minden J_i munkához tartozik egy r_i érkezési idő, egy d_i lejárati idő, és egy w_i súly, $i = 1, \ldots, n$. Egy munka akkor késik, ha az ütemezésünk olyan, hogy azt a d_i lejárati idő után fejezzük be. A cél az, hogy megtaláljuk azt az ütemezést, amely minimalizálja a késő munkák súlyozott összegét.*

Vezessük be a következő jelölést: legyen $U_i = 1$, ha az i. munka ütemezése késik, különben $U_i = 0$. Ez a feladat a [39]-ben bevezetett jelöléssel $O2|p_{ij} = 1, r_i|\sum w_i U_i$ formában írható fel. A probléma bonyolultsága nyitottként volt említve [12]-ben. (Egészen pontosan: már a $w_i = 1$ súlyozatlan probléma bonyolultsága is nyitott volt.)

A [35] cikkben a fenti nyitott problémára adtunk egy megoldást azzal, hogy bemutattunk egy polinomiális algoritmust. A megoldás arra alapul, hogy a problémát transzformáljuk egy megfelelően konstruált él-súlyozott gráfba, amelyben keresünk egy – kissé módosított – teljes, minimális súlyú 2-faktort. A polinom-időben történő megoldhatóságot a következő tétel biztosítja.

4.6.1. Propozíció. (Lovász és Plummer, [48]). *Egy $G = (V, E)$ élsúlyozott gráfra a minimális súlyú teljes 2-faktor kiszámítható $O(|E|^2 \log |V|)$ időben.*

A fejezet fő tételének bizonyításához először bevezetjük a következő definíciót.

4.7. Definíció. Tekintsünk egy $G = (V, E)$ gráfot, amelynek $V = V_2 \cup V_*$ legyen egy csúcspartíciója. A G gráf $G' = (V', E')$ részgráfja $(2, *)$-faktort alkot, ha $\deg'(v) = 2$ ha $v \in V_2$, és $\deg'(v) \leq 2$ ha $v \in V_*$, *ahol* $\deg'(v)$ *a G'-beli fok.*

A feladathoz tartozó G gráfhoz konstruálunk egy $G'' = (V'', E'')$ gráfot, amelyre $|V''| \leq |V_2| + 16\,|V_*|$ és $|E''| \leq |E| + 24\,|V_*|$. Ezután megmutatjuk, hogy miként lehet transzformálni egy G'' gráfban megtalált 2-faktort egy G-beli $(2, *)$-faktorrá, és leírunk egy eljárást az ellenkező irányba is. A G és G'' gráfok közötti összefüggések felhasználásával bebizonyítjuk a következő állítást.

4.6.2. Tétel (G. Galambos, G. Woeginger, [35]). *Tekintsük az*

$$O2|p_{ij}=1,r_i|\sum w_iU_i$$

munkát tartalmazó UET open shop problémát. Létezik olyan algoritmus, amely ezt a problémát megoldja $O(n^4\log n)$ időben.

4.7. Páros műveletek ütemezése

4.7.1. Feladat. *A páros műveletek ütemezésének problémájánál adva van 1 gép és n munka, amelyek mindegyike két műveletet tartalmaz. A két művelet sorrendje kötött, és köztük pontosan meghatározott követési időnek kell eltelni. Az i. munka egy (a_i, L_i, b_i) számhármassal írható le, ahol a_i és b_i a két művelet időszükséglete, L_i a követési idő. A követési idő alatt a gép üresjáratban lenne, ebben az időszakban más munkához tartozó művelete(ke)t is végezhet. A megkezdett műveletek nem megszakíthatóak.*

A feladat az, hogy az n műveletpárt ütemezzük egy gépen úgy, hogy a műveletek nem fedhetik egymást, és a legutoljára ütemezett munka a lehető legkorábbban fejeződjön be. (Ezt az időpontot $C_{\max}$-al jelöljük.)

4.7.1. Alkalmazás. *Számos olyan gyakorlati feladat van, amelynek matematikai modellje a páros műveletek problémájával írható le. A repülőgép-anyahajók felszállópályáján a repülőgépek fel- és leszállásának ütemezése, és a kamionok ki- és berakodása egy központi depónál egyaránt erre a modellre vezethető vissza.*

Ezzel a problémával először Shapiro foglalkozott [56]. Az általa bemutatott algoritmusokat empírikusan vizsgálta, versenyképességüket nem elemezte. Orman és Potts kimerítően tanulmányozta a problémát bonyolultság-elméleti szempontból [51].

Egyetlen olyan probléma van, amelynek időbonyolultsága tisztázatlan maradt: az azonos műveletpárok ütemezésének a problémája. Ekkor $a_i = a$, $L_i = L$, $b_i = b$, $i = 1, \ldots, n$. A disszertáció 8. fejezetében [1] alapján ezt a feladatot vizsgáljuk.

4.8. Definíció. Egy L hosszúságú 0-1 sorozatot $P(a, L, b)$ mintának nevezünk, ha a minta 1-eseket csak b hosszúságú blokkokban tartalmaz, és minden ilyen blokkot egy legalább $a-b$ hosszúságú 0-kat tartalmazó blokk követ.

4.9. Definíció. Legyen p egy $P(a, L, b)$ minta, jelölje $p[i]$ a p minta i. elemét, és legyen i, $1 \le i \le L - a + 1$, olyan egész, amelyre teljesül, hogy

$$p[i] = p[i+1] = \ldots = p[i+a-1] = 0. \tag{7}$$

Ekkor $S(p, i)$ legyen egy olyan 0-1 sorozat, amelyre

$$p[i+a]\ p[i+a+1]\ \ldots\ p[L]\ 1^b\ 0^{i+a-b-1}$$

ahol 1^k (0^k) egy k darab 1-kből (0-kból) álló sorozatot jelöl. Legyen továbbá $w(p, S(p, i)) = i + a - 1$.

Először belátjuk, hogy a $P(a, L, b)$ minták száma $O(r^L)$ nagyságrendű, ahol $r \le \sqrt[a-1]{a}$, majd a fenti definíciók alapján megkonstruálunk egy olyan G súlyozott, irányított gráfot, amelynek csúcspontjai a lehetséges minták, él akkor vezet két csúcspont között, ha a végponthoz tartozó minta $(S(p, i))$ a kezdőponthoz tartozó minta (p) követője, és a $(p, S(p, i))$ élhez tartozó súly $w(p, S(p, i)) = i + a - 1$.

Így a páros műveletek problémáját egy G gráfban keresendő n hosszúságú, minimális összsúlyú út keresésére vezetjük vissza. A disszertációban megadunk egy olyan algoritmust, amely ezt az utat maximum $O(nr^{2L})$ idő alatt találja meg. Az algoritmus n-ben lineáris, L-ben azonban exponenciális. Ezért, ha a két művelet közötti előírt idő nagyon hosszú, akkor az algoritmus használhatósága kétséges lehet. Ugyanakkor az is igaz, hogy $\lim_{a\to\infty} \sqrt[a-1]{a} = 1$, ezért növekvő a értékekre az exponenciális tag egyre kevésbé lesz domináns. A feladat összesen négy adattal leírható, ezért a bonyolultságelméleti kérdés még nyitva maradt, de az algoritmusunk jelenleg is a legjobb az azonos műveletpárok ütemezésére.

4.8. Szövegtömörítési feladat

A disszertáció 9. fejezetében a szótárak segítségével történő szövegtömörítési eljárásokkal foglalkozunk.

4.8.1. Feladat. *Egy szótár (forrás-szó, kód-szó) párokat tartalmaz, ahol a kódok egy-egy ABC-ből választott karakter-sorozatok. A szótáron alapuló kódolás során a szótár elemeinek megfelelően a forrás-szöveget karaktersorozatokra bontva megfeleltetünk mindegyiknek egy kód-szót, és ezzel helyettesítjük a kódolt szövegben. Ha a szótárat a kódolás során nem változtathatjuk, akkor* statikus szótárról *beszélünk.*

Legyen $D = \{(w_i, c_i) : i = 1, \ldots, k\}$ egy statikus szótár. Legyen Bt az S forrás-szöveg egy karakterének hossza bitekben mérve, $|w_i|$ jelölje a w_i forrás-szó karaktereinek a számát, és $\|c_i\|$ a c_i kód-szó hosszát bitekben. Vezessük be a következő jelöléseket:

- $\mathrm{lmax}(D) = \max\{|w_i|\ i = 1, \ldots, k\}$,
- $\mathrm{cmin}(D) = \min\{\|c_i\|\ i = 1, \ldots, k\}$,
- $\mathrm{cmax}(D) = \max\{\|c_i\|\ i = 1, \ldots, k\}$.

Legyen A egy tetszőleges tömörítő algoritmus. A versenyképessége alapvetően függ a rendelkezésre álló szótár tulajdonságaitól. Jelölje az S forrás-szöveg tömörítése során kapott szövegeket $A(D, S)$ ill. $\mathrm{OPT}(D, S)$, és a tömörített szövegek hossza legyen rendre $\|A(D, S)\|$ és $\|OPT(D, S)\|$. Ekkor az aszimptotikus versenyképességi hányados

$$R_A^\infty(D) = \limsup_{n\to\infty} \left\{ \frac{\|A(D,S)\|}{\|OPT(D,S)\|} | S \in S(n) \right\}$$

lesz, ahol $S(n)$ az összes n karaktert tartalmazó forrás-szövegek halmazát jelöli. A ládapakolástól és az ütemezéstől eltérően, ahol legtöbbször egy konstans adható meg egy algoritmus aszimptotikus versenyképességére, az adattömörítés esetében ez az érték az lmax, cmin, és cmax értékek függvénye.

4.10. Definíció. Amennyiben sem a forrás-szavakra, sem a kód-szavakra nem teszünk kikötéseket, akkor *általános szótárról* beszélünk. Egy D általános szótár

- *egységes kódolású*, ha minden kód-szó egyforma hosszú,
- *nem-hosszabbító*, ha egyetlen kód-szó sem hosszabb a hozzá tartozó forrás-szónál,
- *suffix*, ha minden w forrás-szó esetén az összes hozzá tartozó suffix is a szótárhoz tartozik (ha $w = \omega_1\omega_2\cdots\omega_q \in D \Rightarrow \omega_h\omega_{h+1}\cdots\omega_q \in D\ 2 \le h \le q$),

- *prefix*, ha minden w forrás-szó esetén az összes hozzá tartozó prefix is a szótárhoz tartozik (ha $w = \omega_1\omega_2\cdots\omega_q \in D \Rightarrow \omega_1\omega_2\cdots\omega_h \in D$ $1 \leq h \leq q-1$).

A fejezet első tétele az általános szótárakra ad egy felső korlátot.

4.8.1. Tétel (J. Békési, G. Galambos, U. Pferschy, G. Woeginger, [9]). *Legyen D egy általános szótár. Akkor bármely A szövegtömörítési algoritmusra*

$$R_A^{\infty}(D) \leq \frac{(\mathrm{lmax}-1)\mathrm{cmax}}{\mathrm{cmin}}.$$

4.8.1. Longest Matching (LM) algoritmus

4.8.1. Algoritmus. *Az LM algoritmus a tömörítendő szöveget balról jobbra haladva ellenőrzi, és minden pozíciójában a leghosszabb forrás-kódot választja a tömörítéshez.*

Az LM algoritmust Katajainen és Raita [44]-ben elemezték suffix, egységes-kódolású és nem-hosszabbító szótárakra. A prefix algoritmusokat [10]-ben vizsgáltuk. Eredményeinket a következő tételekben foglalhatjuk össze.

4.8.2. Tétel ([10]). *Legyen D_1 egy prefix és D_2 egy prefix és egységes kódolású szótár. Ekkor*

$$R_{LM}^{\infty}(D_1) \leq \frac{(\mathrm{lmax}-1)\mathrm{cmax}}{\mathrm{cmin}}, \qquad R_{LM}^{\infty}(D_2) \leq \mathrm{lmax}-1,$$

és mindkét korlát éles.

4.8.3. Tétel (J. Békési, G. Galambos, U. Pferschy, G. Woeginger, [10]). *Legyen D_1 egy prefix, nem-hosszabbító és D_2 egy prefix, nem-hosszabbító és egységes kódolású szótár. Ekkor*

$$R_{LM}^{\infty}(D_1) \leq \frac{\mathrm{lmaxBt}}{\mathrm{cmin}}, \qquad R_{LM}^{\infty}(D_2) \leq \mathrm{lmax}-1,$$

és mindkét korlát éles.

4.8.2. Differential Greedy (DG) algoritmus

4.8.2. Algoritmus. *A DG algoritmus az aktuális pozícióban mindig azt a forrás-szót választja, amelyre a lokális tömörítés mértéke ($|w_i|\mathrm{Bt} - \|c_i\|$) maximális.*

Belátható, hogy a DG és az LM algoritmusok prefix szótárakra hasonlóan viselkednek. Suffix típusú szótárakat használó online algoritmusokat [10]-ben vizsgáltunk.

4.8.4. Tétel ([10]). *Legyen D egy suffix szótár, és legyen* $\mathrm{lmax} \geq 3$. *Ekkor*

$$R_{DG}^{\infty}(D) \leq \begin{cases} \dfrac{2\mathrm{cmax}-\mathrm{Bt}}{\mathrm{cmin}} & \text{ha } \mathrm{cmax} \leq \frac{3}{2}\,\mathrm{Bt} \\ \dfrac{(2\mathrm{cmax}+\mathrm{Bt})^2}{8\mathrm{Bt}\,\mathrm{cmin}} & \text{ha } \frac{3}{2}\,\mathrm{Bt} < \mathrm{cmax} \leq (\mathrm{lmax}-\frac{3}{2})\mathrm{Bt} \\ \dfrac{(\mathrm{lmax}-1)(2\mathrm{cmax}-(\mathrm{lmax}-2)\mathrm{Bt})}{2\mathrm{cmin}} & \text{ha } (\mathrm{lmax}-\frac{3}{2})\mathrm{Bt} < \mathrm{cmax} \end{cases}$$

és a megadott felső korlátok élesek.

Suffix, nem-hosszabbító szótárakra [44]-ben egy

$$R^{\infty}_{DG}(D) \leq \frac{\min\{\text{lmaxBt}, 2\text{cmax} - \text{Bt}\}}{\text{cmin}}.$$

felső korlátot adtak. A disszertációban bebizonyítjuk, hogy a felső korlát éles.

4.8.5. Tétel ([10]). *Végtelen sok* Bt, lmax, cmin *és* cmax *pozitív egészekből álló számnégyeshez* (cmin $\leq$ Bt, cmin $\leq$ cmax, cmax $\leq$ lmaxBt) *létezik olyan nem-hosszabbító, suffix szótár, amelyre*

$$R^{\infty}_{DG}(D) \geq \frac{\min\{\text{lmaxBt}, 2\text{cmax} - \text{Bt}\}}{\text{cmin}}.$$

4.8.3. Fractional Greedy (FG) algoritmus

Egy korábbi cikkben ([44]) azt kérdezték, hogy definiálható-e a fenti algoritmusoktól eltérő, jó aszimptotikus versenyképességgel rendelkező, új algoritmus. [9]-ben definiáltuk a következő algoritmust, amelyről bebizonyítottuk, hogy több szótártípus esetén jobb versenyképességgel rendelkezik, mint a korábban ismert algoritmusok.

4.8.3. Algoritmus. *Az FG algoritmus az adott pozícióban azt a forrás-kódot választja, amely arányaiban a legjobb tömörítést adja. Ha I jelöli a szöveg adott pontjában rendelkezésre álló kód-párok indexeinek a halmazát, akkor az FG algoritmus azt az i_0 indexet választja, amelyre*

$$i_0 = \min_{i \in I} \frac{\|c_i\|}{|w_i|\text{Bt}}.$$

A következő tételek azt mutatják, hogy *suffix szótárak* esetében az FG algoritmus viselkedése teljesen eltér a korábbi heurisztikák viselkedésétől.

4.8.6. Tétel ([9])). *Legyen D egy suffix szótár. Ekkor*

$$R^{\infty}_{FG}(D) \leq \frac{\text{cmax}(\ln(\text{lmax} - 1) + 1)}{\text{cmin}}$$

és megadható egy olyan D_0 suffix szótár, amelyre

$$R^{\infty}_{FG}(D_0) > \frac{\text{cmax}(\ln(\text{lmax} - 1) + 1 - \ln 2)}{\text{cmin}}.$$

4.8.7. Tétel ([9]). *Legyen D egy suffix és nem-hosszabbító szótár. Ekkor*

$$R^{\infty}_{FG}(D) \leq \frac{\min\{\text{lmax}\,\text{Bt}, \text{cmax}\left(\ln\left(\frac{\text{lmax}\,\text{Bt}}{\text{cmax}}\right) + 3\right) - \text{Bt}\}}{\text{cmin}}.$$

Az alsó korlátra csak egy nagyon közeli – de nem éles – állítást tudtunk bizonyítani.

4.8.8. Tétel ([9]). *Létezik olyan D_1 suffix, nem-hosszabbító szótár, amelyre*

$$R^{\infty}_{FG}(D_1) \geq \frac{\text{cmax}\left(\ln\left(\frac{\text{lmax}\,\text{Bt}}{\text{cmax}}\right) + 1\right)}{\text{cmin}}.$$

4.8.4. Iterated Longest Fragment (ILF) algoritmus

Az algoritmust Stauffer és Hirschberg ([55]) dogozta ki párhuzamos architektúrájú gépekre. Nagumo, Liu és Watson ([49]) bizonyította be, hogy az algoritmus adaptálható az egyprocesszoros környezetre is. Tekintsünk egy $\frac{1}{2}(\text{lmax}-1)(\text{lmax}-2)+\text{lmax}$ hosszúságú puffert.

(1) Mozdítsuk el a puffer ablakának kezdőpontját az első olyan csúcspontba, amelyhez tartozik – nem eliminált – (forrás-szó, kód-szó) pár a szótárban.

(2) Válasszuk ki az első leghosszabb élet kódolásra a pufferterületen belül, és hagyjuk el az összes őt átfedő élt a gráfból. Ugorjunk az (1) pontra.

Az algoritmus versenyképességét nem vizsgálták. A disszertációban a [11] cikk alapján teljeskörű elemzést adunk az ILF algoritmusról, és valamennyi szótártípus kombinációra éles aszimptotikus versenyképességi becslést bizonyítunk.

4.8.9. Tétel (J. Békési, G. Galambos, [11]). *Legyen D egy általános szótár, és legyen* $\text{lmax} \geq 3$. *Ekkor*

$$R_{ILF}^{\infty}(D) = \frac{2\,(\text{lmax}-1)}{3}\frac{\text{cmax}}{\text{cmin}}.$$

Egységes kódolású szótárra hasonló tétel mondható ki, ha figyelembe vesszük, hogy $\text{cmax} = \text{cmin}$. Nem hosszabbító kódolással rendelkező szótárra a következő állítás bizonyítható be.

4.8.10. Tétel ([11]). *Legyen D egy nem-hosszabbító általános szótár, amelyre* $\text{Bt} \leq \text{cmax}$. *Ekkor*

$$R_{ILF}^{\infty}(D) = \frac{(2\text{lmax}-3)\ \text{Bt}+\text{cmax}}{3\text{cmin}}$$

Egy korábbi cikkben ([44]) azt a sejtést fogalmazták meg, hogy a prefix típusú szótárak alkalmazása általában nem nem javítja az algoritmusok versenyképességét. A disszertációban megmutatjuk, hogy ez a sejtés az ILF algoritmus esetén nem igaz. Belátjuk, hogy az ILF algoritmus versenyképessége prefix típusú szótárakra közel kétszer olyan jó, mint az általános szótárak használatakor. Azt is megmutatjuk, hogy a prefix és a suffix típusú szótárak esetén az ILF algoritmus közel azonos aszimptotikus versenyképességet eredményez.

4.8.11. Tétel ([11]). *Legyen D egy prefix, általános szótár. Ekkor*

$$R_{ILF}^{\infty}(D) = \frac{\text{lmax}+1}{3}\frac{\text{cmax}}{\text{cmin}}.$$

4.8.12. Következmény. *Legyen D egy prefix, nem-hosszabbító általános szótár. Ekkor*

$$R_{ILF}^{\infty}(D) = \begin{cases} \frac{(\text{lmax}-1)\text{Bt}+2\text{cmax}}{3\text{cmin}}, & \textit{ha } \text{Bt} < \text{cmax} \leq (\text{lmax}-2)\,\text{Bt} \\ \\ \frac{(2\text{lmax}-3)\ \text{Bt}+\text{cmax}}{3\text{cmin}}, & \textit{ha } (\text{lmax}-2)\,\text{Bt} < \text{cmax} \end{cases}$$

4.8.13. Következmény. *Legyen D_1 prefix, egységes-kódolású és D_2 prefix, egységes- kódolású és nem-hosszabbító általános szótár. Ekkor*

$$R_{ILF}^{\infty}(D_i) = \frac{\text{lmax}+1}{3}, \qquad i = 1, 2.$$

A disszertáció utolsó tétele suffix szótárakra vonatkozik.

4.8.14. Tétel ([11]). *Legyen D suffix szótár, és legyen* $\mathrm{lmax} \geq 3$. *Ekkor*

$$R_{ILF}^{\infty}(D) = \frac{\mathrm{lmax}}{3}\frac{\mathrm{cmax}}{\mathrm{cmin}}.$$

4.8.15. Következmény. *Legyen D egy suffix, nem-hosszabbító általános szótár. Ekkor*

$$R_{ILF}^{\infty}(D) = \begin{cases} \frac{(\mathrm{lmax}-2)\mathrm{Bt}+2\mathrm{cmax}}{3\mathrm{cmin}}, & \textit{ha } \mathrm{Bt} < \mathrm{cmax} \leq (\mathrm{lmax}-1)\,\mathrm{Bt} \\ \frac{(2\mathrm{lmax}-3)\ \mathrm{Bt}+\mathrm{cmax}}{3\mathrm{cmin}}, & \textit{ha } (\mathrm{lmax}-1)\,\mathrm{Bt} < \mathrm{cmax} \end{cases}$$

Hivatkozások

[1] D. Ahr, J. Békési, G. Galambos, M. Oswald, és G. Reinelt, An Exact Algorithm for Scheduling Identical Coupled Tasks, *ZOR*, **598**, 2004, 193–203.

[2] S. F. Assmann, D. S. Johnson, D. J. Kleitman, és J. Y.-T. Leung, On a dual version of the one-dimensional binpacking problem, *J. Algorithms*, **5,** 1984, 502–525.

[3] J. Balogh, J. Békési, és G. Galambos, New Lower Bounds for Certain Classes of Bin Packing Algorithms, *In Klaus Jansen and Roberto Solis-Oba, editors, Approximation and Online Algorithms - 8th International Workshop, WAOA 2010, Lecture Notes in Computer Science, Springer,* **6534**, 2011, 25–36.

[4] J. Balogh, J. Békési, és G. Galambos, New Lower Bounds for Certain Classes of Bin Packing Algorithms, *Theoretical Comp. Sci.,* **440-441**, 2012, 1–13.

[5] J. Balogh, J. Békési, G. Galambos, és G. Reinelt, Lower bound for the online bin packing problem with restricted repacking, *SIAM J. Comput.*, **38(1)** 2008, 398–410.

[6] J. Balogh, J. Békési, G. Galambos, és G. Reinelt, Online bin packing with restricted repacking, J. of Comb. Opt., **27(1)**, 2014, 115–131, 2014.

[7] Y. Bartal, H. Karloff, és Y. Rabani, A Better Lower Bound fo online Scheduling, *Inf. Proc. Letters,* **50**, 1994, 113–116.

[8] Y. Bartal, A. Fiat, H. Karloff, és R. Vohra, New Algorithms for an Ancient Scheduling Problem, *Proc. of 24th ACM STOC*, 1992, 51–58.

[9] J. Békési, G. Galambos, U. Pferschy, és G.J.Woeginger, The Fractional Greedy Algorithm for Data Compression, *Computing,* **56(1)**, 1996, 29-46.

[10] J. Békési, G. Galambos, U. Pferschy, G. Woeginger, Greedy Algorithms for online Data Compression, *Journal of Algorithms,* **25**, 1997, 274-289.

[11] J. Békési, G. Galambos, Worst-case analysis of the Iterated Longest Fragment algorithm, *Information Processing Letters,* **97**, 2001, 147–153.

[12] P. Brucker, J. Jurisch, és M. Jurisch, Open shop problems with unit time operations, *ZOR,* **37**, 1993, 57–73.

[13] R. Chandrasekaran, B. Chen, G. Galambos, P.R. Narayanan, A. van Vliet, és G.W. Woeginger. A note on "An online scheduling heuristic with better worst case ratio than Graham's list scheduling", *SIAM J. on Computing,* **26,(3)**, 1997, 870–872.

[14] B. Chen, és A. van Vliet, On the online Scheduling Algorithm RLS, *Report 9325/A, Economic Institute, Erasmus University, Rotterdam,* 1993.

[15] R. F. K. Chung, M. R. Garey, és D. S. Johnson, On packing two-dimensional bins, *SIAM Alg.Discr.Meth.,* **3,** 1982, 66–76.

[16] E. G. Coffman, G. Galambos, S. Martello, és D. Vigo, Bin Packing Approximation Algorithms: Combinatorial Analysis, *In "Handbook of Combinatorial Optimization" Supplement Volume. (Eds. D.-Z. Du and P.M. Pardalos Eds.). Kluwer Academic Publishers*, 2002, 151–208.

[17] E. G. Coffman, J. Csirik, G. Galambos, S. Martello, és D. Vigo, Bin Packing Approximation Algorithms: Survey and Classification, *In "Handbook of Combinatorial Optimization," New York, Springer Science+Business Media, Second Edition, (Eds. D.-Z. Du and P.M. Pardalos Eds.). Kluwer Academic Publishers*, 2013, 1455–531.

[18] P. R. Coppersmith, P. Raghavan, Multidimensional online bin-packig: algorithms and worst-case analysis, *Op. Res. Letters* **8**, 1989, 17–20.

[19] J. Csirik és G. Galambos, An O(n) bin packing algorithm for uniformly distributed data. *Computing,* **36**, 1986, 313–319.

[20] J. Csirik, J. B. G. Frenk, A. M. Frieze, G. Galambos, és A. H. G. Rinnooy Kan, A probabilistic analysis of the next fit decreasing bin packing heuristic, *Op. Res. Letters*, 1986, 233–236.

[21] J. Csirik, J. B. G. Frenk, G. Galambos, és A. H. G. Rinnooy Kan, A probabilistic analysis of algorithms for dual bin packing problems, *J. of Algorithms,* **12**, 1991, 189–203.

[22] J. Csirik, J. B. G. Frenk, és M. Labbe, Two dimensional rectangle packing: On line methods and results, *ARIDAM V, Rutgers University*, 1990.

[23] J. Csirik, G. Galambos, és Gy. Turán, Some Results on Bin Packing, *Proceedings of EURO VI.*, Vienna, 1983.

[24] J. Csirik, és G. J. Woeginger, Online Packing and Covering Problems, *in: online Algorithms, Lecture Notes in Computer Science, eds. A. Fiat and G. Woeginger, Vol. 1442,* Berlin, 1998, 147–177.

[25] J. Csirik, és D.S. Johnson, Bounded space online bin packing: Best is better than First, *Proc. 2nd Ann. ACM-SIAM SODA,San Francisco*, 1991, 309–319.

[26] U. Faigle, W. Kern és Gy. Turán, On the performance of online algorithms for partition problems, *Acta Cybernet.*, **9**, 1989, 107–119.

[27] J. B. Frenk, és G. Galambos, Hybrid next-fit algorithm for the two-dimensional rectangle bin packing problem, *Computing,* **39**, 1987, 201–217.

[28] G. Galambos, Parametric Lower Bound for online Bin Packing, *SIAM J. on Alg. and Discr. Meth.* **7**, 1986, 362–367.

[29] G. Galambos, Notes on the Lee's harmonic fit algorithm, Computatorica, *Annales Univ. Sci. Budapest, Sect. Comp.,* **9**, 1988, 121–126.

[30] G. Galambos, A 1.6 Lower Bound for the Two-Dimensional online Rectangle Bin Packing, *Acta Cybernetica,* **10**, 1991, 21–24.

[31] G. Galambos és J. B. G. Frenk, A simple proof of Liang's lower bound for online bin packing and the extension to the parametric case, *Discrete Applied Mathematics,* **41(2)**, 1993, 173–178.

[32] G. Galambos és A. van Vliet, *Lower bounds for 1,2 and 3-dimensional online bin packing algorithms, Computing,* **52**, 1994, 281–297.

[33] G. Galambos és G. J. Woeginger, Repacking helps in bounded space online bin packing,*Computing,* **49**, 1993, 329–338.

[34] G. Galambos és G. J. Woeginger, An online scheduling heuristic with better worst case ratio than Grahams list scheduling, *SIAM Journal on Computing,* **22**, 1993, 349–355.

[35] G. Galambos és G. J. Woeginger, Minimizing the Weighted Number of Late Jobs in UET Open Shops, *ZOR,* **41**, 1995, 109–114.

[36] M. R. Garey és D. S. Johnson, Computers and Intractibility: A Guide to the Theory of NP-completeness, *W.H. Freeman & Co, San Francisco*, 1979.

[37] T. Gonzales, és S. Sahni, Open Shop Scheduling to Minimize Finish Time, *Journal of ACM,* **23**, 1976, 665–679.

[38] R. L. Graham, Bounds on multiprocessing timing anomalies, *SIAM J. Appl. Math.,* **17**, 1969, 416–429.

[39] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey, *Annals of Discrete Mathematics,* **5,** 1979, 287–326.

[40] Z. Ivkovič, és E. L. Lloyd, A Fundamental Restriction on Fully Dynamic Maintenance of Bin Packing, *Information Processing Letters,* **59(4),** 1996, 229–232.

[41] D. S. Johnson, Fast algorithms for bin packing, *Computer Syst. Sciences,* **8**, 1974, 272–314.

[42] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, és R. L. Graham, Worst case performance bounds for simple one-dimensional packing algorithms, *SIAM J. on Computing,* **3**, 1974, 299–325.

[43] D. R. Karger, S. J. Philips, és E. Torng, A Better Algorithm for an Ancient Scheduling Problem, *Proc. 5th Ann. ACM-SIAM SODA*, 1994, 132–140.

[44] J Katajainen és T. Raita, An analysis of the longest matching and the greedy heuristic in text encoding, *Journal of the ACM,* **39**, 1992, 281–294.

[45] C. C. Lee, és D. T. Lee, A simple online bin packing algorithm, *J. Assoc. Comput. Mach.* **32**, 1985, 562–572.

[46] F. M. Liang, A Lower Bound for online Bin Packing, *Inf. Proc. Letters,* **10,** 1980, 76–79.

[47] K. Li, K. H. Chang, A generalized Harmonic algorithm for online multi-dimensional bin packing, *Technical Report TR UH-Cs-90-2, University of Houston, January*, 1990.

[48] L. Lovász, M. D. Plummer, Matching Theory, *Akadémiai Kiadó, North Holland*, 1986.

[49] H. Nagumo, M. Lu, and K. Watson, On–line longest fragment first algorithm. *Information Processing Letters* **59** (1996) 91-96.

[50] H. L. Ong, M. J. Magazine, and T. S. Wee, Probabilistic analysis of bin packing heuristics, *Operation Research,* **32,** 1984, 983–998.

[51] A. J. Orman, és C. N. Potts, On the Complexity of Coupled-Task Scheduling, *Discrete Applied Mathematics,* **72**, 1997, 141–154.

[52] M. B. Richey, Improved bounds for Refined Harmonic Bin Packing, *unpublished manuscript*, 1990.

[53] S. Seiden, On the online Bin Packing Problem, *Journal of ACM,* **49,** 2002, 640–671.

[54] Steve S. Seiden és Rob van Stee, New bounds for multi-dimensional packing, *Algorithmica,* **36(3),** 2003, 261–293.

[55] E. J. Schuegraf and H. S. Heaps, A comparison of algorithms for data base compression by use of fragments as language elements, *Inf. Stor. Ret.,* **10**, 1974, 309–319.

[56] R. D. Shapiro, Scheduling Coupled Tasks, *Naval Research Logistics Quarterly,* **20**, 1980, 489–498.

[57] J. Sylvester, On a Point in the Theory of Vulgar Fractions. *American Journal of Mathematics,* **3,** 1880, 332–335.

[58] A. van Vliet, An Improved Lower Bound for online Bin Packing Algorithms, *Inf. Proc. Letters,* **43** 1992, 274–284.

[59] A. van Vliet, Lower Bound and Upper Bounds for online Bin Packing and Scheduling Algorithms, PhD Thesis, *Tinbergen Institute Res. Ser.* no. 93.

[60] A. C. C. Yao, New Algorithms for Bin Packing. *J. Assoc. Comp. Mach.* **27**, 1980, 207–227.