BALOGH JÁNOS – BÉKÉSI JÓZSEF – GALAMBOS GÁBOR

Ládapakolások átpakolással

ALKALMAZOTT TERMÉSZETTUDOMÁNYI INTÉZET

ládapakolás, szemi-on-line algoritmus, legrosszabb eset viselkedés, versenyképességi elemzés.

1. Bevezetés

A ládapakolási feladat több mint 35 éve kutatott feladata a diszkrét optimalizálás területének. Az alapfeladat hétköznapi szavakkal is könnyen megfogalmazható: tárgyak sorozatát kell elpakolni a lehető legkevesebb számú, azonos méretű (kapacitású) ládába. Ennek egydimenziós változatában a *tárgy méreteként* csak egyetlen paramétert (a példa kedvéért mondjuk csak a térfogatot) vesszük figyelembe, a többit elhanyagoljuk. Nyilvánvalóan egy ládába nem rakhatunk nagyobb méretösszegű elemeket, mint a láda kapacitása, valamint a tárgyak nem „fedhetik át" egymást. A ládák kapacitását az irodalomban szokásos módon egységnyinek definiáljuk, az érkező tárgyak sorozatát pedig a *(0,1]* intervallumba eső számok egy listájával adjuk meg, ahogy ezt a következőkben pontosan formalizáljuk.

A feladat on-line változatában minden egyes tárgy érkezésekor azonnal dönteni kell annak végleges elhelyezéséről. Dönthetünk úgy, hogy egy olyan ládába rakjuk, amelyikben már vannak elemek (már megnyitott a láda), és ebbe az érkező tárgy még „belefér", vagy új ládát nyitunk, és abban helyezzük el a tárgyat. Az, hogy melyik ládába rakjuk, már nem változtatható meg később. Ezt a döntést anélkül kell meghozni, hogy bármit tudnánk az ezután érkező tárgyakról. (Még azt sem tudjuk, hogy hány tárgy jön még, vagy jön-e még egyáltalán.) Az általunk vizsgált ún. *c*-átpakolásos szemi-on-line ládapakolási feladat az on-line feladattól abban különbözik, hogy minden tárgy érkezésekor a korábban elpakolt tárgyak közül legfeljebb *c* darab tárgy újrapakolása (már nyitott ládák közötti mozgatása, vagy új láda nyitása és abba áthelyezése) is megengedett a pakoló, azaz az algoritmus számára.

Itt az [1–4] munkák alapján adunk alsó és felső korlátokat az így definiált *c*-átpakolásos szemi-on-line ládapakolási feladatra.

2. Definíciók

A klasszikus egydimenziós ládapakolási feladat az egyik legismertebb, intenzíven kutatott kombinatorikus optimalizálási probléma. A probléma a következő: adott olyan tárgyak (a továbbiakban tárgyak helyett az *elemek* szót fogjuk használni) egy $L=x_1, x_2, \ldots, x_n$ listája, ahol az elemek mérete a *(0,1]* intervallumba esik. Adott továbbá egység kapacitású ládáknak egy végtelen listája. Az L lista minden egyes x_i elemét hozzá kell rendelni pontosan egy ládához úgy, hogy a ládában lévő elemek méreteinek összege nem haladhatja meg a láda kapacitását. (Ahol a szövegkörnyezetben nem érthető félre, ott egy x_i elem méretét szintén x_i-vel jelöljük, különben $s(x_i)$-vel.) A cél a felhasznált (a pakolás befejezésekor nem üres) ládák számának minimalizálása. Az egy jól ismert tény, hogy optimális pakolást találni *NP–nehéz* feladat [13].

Ebből következően az utóbbi évtizedekben nagy számú, elfogadható közelítést nyújtó algoritmust publikáltak a téma kutatói.

A különböző algoritmusok hatékonyságának mérésére két általános módszer használt: a *legrosszabb eset viselkedés* vizsgálata, vagy – feltéve az elemek valamely valószínűség-eloszlását – az algoritmusok átlagos eset (valószínűségi) vizsgálata. Jelen dolgozatban csak az algoritmusok *legrosszabb eset viselkedési* vizsgálatára szorítkozunk, ezen belül is ennek aszimptotikus vizsgálatára. Ezt jól jellemzi az irodalomban szokásos, bevett módon használt *aszimptotikus legrosszabb eset hányados*, vagy másképpen *aszimptotikus versenyképességi hányados, AVK*. Ez a következő módon definiálható egy A algoritmusra. Egy adott L lista esetén jelöljük rendre $A(L)$-lel és $OPT(L)$-lel az A algoritmus által használt, illetve egy optimális pakolás által használt ládák számát. Ekkor

$$AVK\ (A) := \lim_{k\to\infty} \sup \left\{ \max_{L} \left\{ \frac{A(L)}{k} \middle| OPT\ (L) = k \right\} \right\}. \qquad (1)$$

AVK(A) az A algoritmus aszimptotikus versenyképességi hányadosa.

Az algoritmusok egy osztályát alkotják az *on-line algoritmusok*, amelyek végrehajtásakor az elemeket érkezésük sorrendjében pakoljuk el úgy, hogy semmit nem tudunk a lista további elemeiről. (Sem az elemek száma, sem a később érkező elemek mérete nem ismert.) Az egyszer elpakolt elemek az algoritmus során többet nem mozgathatók. Az *off-line algoritmusok* teljes információval rendelkeznek a listáról, ezt a stratégiájuk kialakításánál figyelembe is veszik. Sokszor rendezik a listát (legtöbbször az elemek mérete szerint csökkenő sorrendbe), majd ezután valamilyen on-line technikát alkalmaznak.

Off-line típusú algoritmusokra a legjobb eredmény [8]-ban és [19]-ben találhatók, amelyek szerzői bizonyították, hogy tetszőleges $\varepsilon>0$-hoz vannak olyan lineáris idejű algoritmusok, amelyek AVK-ja $1+\varepsilon$.

A legjobb ismert on-line algoritmust, a Harmonic++ nevűt nem régen adták meg, ez Seiden nevéhez fűződik [23] és *1,58889*-versenyképes. Pontosabban: Seiden azt bizonyította, hogy algoritmusának AVK-ja legfeljebb *1,58889*. Mivel a Harmonic++ Seiden által ugyanebben a cikkben definiált Super Harmonic algoritmusosztályba tartozik, és a Super Harmonic osztály minden algoritmusának aszimptotikus versenyképességére érvényes a [22]-ben bizonyított *1,58333* alsó korlát, így $1{,}58333 \leq AVK(Harmonic{+}{+}) \leq 1{,}58889$ teljesül. Megjegyezzük, hogy 1985 óta minden, aktuálisan legjobb algoritmus Super Harmonic típusú. Az on-line feladatra, azaz bármely on-line algoritmus aszimptotikus versenyképességére ismert legjobb ismert alsó korlát *1,5401*, amelyet van Vliet bizonyított [24]. Mindezek azt is jelentik, hogy egy *1,58333*-nál versenyképesebb algoritmus megadásához szakítani kell az utóbbi húsz évben használt Super Harmonic technikával, ha on-line algoritmust szeretnénk adni. Nyitott kérdés, hogy egyáltalán lehet-e ilyet adni. Egy másik lehetőség, amelyre *1,5401*-nél versenyképesebb algoritmus megadásához már biztosan szükség van: lazítani az on-line feladat előírásán.

3. Szemi-on-line ládapakolási algoritmusok

Az úgynevezett *szemi-on-line algoritmusok* az on-line és az off-line algoritmusok között helyezkednek el [5]. Az on-line algoritmusokhoz hasonlóan, ezek az algoritmusok sem rendelkeznek minden információval a listáról. Az on-line feladat definíciójától eltérően azonban valamilyen részleges információval igen. A szemi-on-line algoritmusoknál a következő műveletek legalább egyike megengedett pluszban az on-line esethez képest:

- elemek átpakolása [10, 12, 17, 18],
- néhány következő elem megvizsgálása („előrenézése", [14, 15]), vagy
- néhány elem (elő)rendezése [9].

Ezért tekinthetünk egy ilyen feladatot az on-line feladat egy meglazításának.

Időrendi sorrendben az első szemi-on-line ládapakolási algoritmust Galambos adta meg [9] az on-line ládapakolási feladat azon megszorítására, amikor csak korlátos számú láda lehet egyszerre nyitva (vagy más szóval *aktív*, a szintén a szakirodalomban szokásosan használt terminológiában). Egy láda lezárt, ha már nem pakolhatunk bele később. Ez az algoritmus két bufferládát használ az elemek átmeneti tárolására. Ezt javítva Galambos és Woeginger [10] definiált egy, *3* bufferládát és szintén átpakolást használó, szemi-on-line algoritmust. Ennek AVK-ja *1,69103...*, amely bizonyítottan optimális a korlátos számú nyitott ládát használó ládapakolási algoritmusok között.

Néhány évvel később Gambosi és szerzőtársai [11, 12] visszatértek bizonyos, szemi-on-line típusú algoritmusok elemzéséhez. Az általuk megadott algoritmusokban átpakolást használtak, de alkalmazását és költségét speciális módon definiálták. Az általuk adott algoritmus az „elég nagy" elemeket egyesével pakolja át (mozgatja ládák között), míg a „kis" elemekből csoportokat, kötegeket képez. Ezután egy-egy köteget egyszerre, egy lépésben pakolhatja át, és egy ilyen köteg ládák közötti mozgatása is egységnyi költségűnek definiált. Azaz egy ilyen mozgatás, hasonlóan, mint egy nagy elem átpakolása, szintén 1 átpakolásnak számít. Ebben az értelemben ezek az algoritmusok egy lépésben akár $O(n)$ darab elemet is mozgathatnak. A [12] cikkben két algoritmust elemeztek. Ezek közül a gyorsabb lineáris idejű és *3/2*-versenyképes, míg a másik algoritmus $O(n \log n)$ futási idejű és *4/3*-versenyképes.

Kapcsolat állítható fel a ládapakolás, valamint a számítógépes programok (jobok) memóriaterület-foglalásának ütemezési feladata között. Egy job (byte-okban adott) mérete egy elem méretének felel meg, míg egy memória partíció mérete megfelel a láda kapacitásának. A legfontosabb különbség azonban, hogy egy jobhoz hozzárendelt memóriaterület a job futásának befejezésekor felszabadítható a memóriában. Ezt (is) modellezve, a ládapakolási algoritmusok új osztálya adható meg, a *dinamikus ládapakolási feladatoké*, ahol a listabeli elemekre „Törlés" művelet is előírható, azaz az elemek távozása is megengedett, azok érkezése (*Beszúrás* művelet) mellett. Ez az előírás az input része, azaz egy input lista nem más, mint Beszúrás és Tör-

lés műveletek véges sorozata; azaz amíg el nem érjük a lista végét, addig a következő input művelet (lépés) mindig vagy egy új elem beszúrása, vagy egy korábban beszúrt (érkezett) és még nem törölt elem törlésének egyike lehet. Hangsúlyozzuk, hogy törlés csak az input része lehet, azaz az algoritmus nem törölhet elemet ennél a feladatnál (sem), csak az inputszolgáltató – nevezzük *Ütemezőnek* ezt a szereplőt – teheti ezt meg. Egy *A dinamikus ládapakolási algoritmus* által felhasznált ládák számát úgy definiáljuk, mint az *összes lépés során felhasznált (nemüres) ládák számának a maximuma*. Ekkor egy *A* dinamikus ládapakolási algoritmus versenyképességi hányadosa, *AVK(A)* is a korábbiakhoz teljesen hasonló módon, az (1) formulával definiálható. Könnyen látható, hogy az eredeti ládapakolási feladat a dinamikus ládapakolási feladat azon speciális esete, amikor a lista csak Beszúrás műveletekből áll, az itteni terminológiában fogalmazva. A dinamikus ládapakolási feladatot Coffman, Garey és Johnson definiálták és vizsgálták is [6]-ban, a feladatra közelítő algoritmust adva és többféleképpen elemezve azt.

Ivkovič és Lloyd a ládapakolási feladatok fentebbi két változatát kombinálták.

A dinamikus ládapakolási feladatban az algoritmus számára átpakolás használatát megengedve definiálták a *teljesen dinamikus ládapakolási feladatot (fully-dynamic bin packing, FDBP)*. Hasonlóan a [12]-ben használt technikához, az általuk az így kapott problémára adott algoritmusuk szintén felhasználta a kis elemek „kötegelésének" technikáját. Az általuk elért aszimptotikus versenyképességi hányados *5/4* volt.

Az utóbbi évtizedig nem publikáltak (a triviálison kívüli) alsó korlátokat szemi-on-line algoritmusok hatékonyságára. Az ilyen szempontból első, alsó korlátokat tanulmányozó dolgozatot szintén Ivkovič és Lloyd jegyzi [17]. A teljesen dinamikus ládapakolási feladat azon változatára bizonyítottak alsó korlátot, amikor az átpakolás lehetőségét úgy szorítjuk meg, hogy lépésenként (műveletenként) csak konstans számú elem pakolható át. Nevezzük az így kapott feladatot *c-átpakolásos teljesen dinamikus ládapakolási feladatnak* (*c-átpakolásos FDBP feladat*). Bizonyították, hogy nincs olyan *c*-átpakolásos FDBP algoritmus, amelynek aszimptotikus versenyképességi hányadosa jobb lenne, mint *4/3*. A konstrukció kis módosítással átvihető, és az adott *4/3*-os alsó korlát érvényes a *c*-átpakolásos szemi-on-line feladatra is, ahogy ez a tény megtalálható Csirik és Woeginger [7] tanulmányában is. Fontos megjegyezni, hogy a korlát minden *c*-re, és mindkét feladatra érvényes, ahogy az általunk megadott *1,3871*-es javításra is teljesül.

Gutin és szerzőtársai 2005-ben megjelent cikkükben [15] adtak alsó korlátot az általuk (ugyanitt) bevezetett és tanulmányozott ún. *batched ládapakolási feladatra* (rövidítve *BBPP* az angol *batched bin packing problem* névből). Ez, a szintén a szemi-on-line ládapakolási feladatok körébe sorolható feladat a klasszikus ládapakolási feladat azon módosítása, amikor az input lista részekre, úgynevezett batchekre osztott. Minden batch csak az előző batch feldolgozása után elérhető. Minden batch önmagában off-line módon pakolható, azonban a korábbi batchek elemei egy későbbi batch feldolgozása során már nem mozgathatók. Egy batch több elemből állhat, de lehet üres is. Ha minden batch pontosan egy elemből áll, akkor a probléma speciális eseteként visszakapjuk a klasszikus on-line ládapakolási feladatot. Ha az input pontosan *m* batch-ből áll, akkor a BBPP feladat ezen megszorítását *m-BBPP feladatnak* nevezzük. Az ilyen típusú feladatokra adott algoritmusok aszimptotikus versenyképességi hányadosa hasonló módon definiálható, mint a fentiekben.

A [15] cikkben a 2-BBPP probléma és annak néhány variánsa tárgyalt részletesebben. A 2-BBPP problémára itt *1,3871*...-es alsó korlátot bizonyítottak, továbbá korlátokat adtak a 2-BBPP probléma azon speciális eseteire, amikor a listában szereplő *különböző (méretű)* elemek maximális száma felülről korlátozott (egy rögzített $p \geq 2$ pozitív egész számmal). Bizonyították (ugyanezen szerzők egy másik, [16] közleményének eredményét felhasználva), hogy ha $p=2$ akkor $r(p)$ korlátjuk optimális, és nyitott kérdésként felvetették, hogy az ettől különböző, $p(>2)$ értékekre általuk adott korlát optimális-e. [1]-ben bizonyítottuk, hogy az általunk adott konstrukcióval ezek javíthatóak; azaz nemleges választ adunk az előző, korlátaik optimális voltára vonatkozó kérdésre.

4. Eredményeink

A [2] dolgozatban a *c*-átpakolásos szemi-on-line feladatra javítottuk az irodalomban ismert korábbi legjobb alsó korlátot. Ez Ivkovič és Lloyd 1996-ban publikált [17] *4/3*-os korlátja volt, amelyet a [2]-ben nyert *1,3871* értékre javítottuk. Megjegyezzük, hogy Ivkovič és Lloyd eredményüket a *c*-átpakolásos teljesen dinamikus ládapakolási feladatra fogalmazták meg. Mi *1,3871*-es eredményünket a *c*-átpakolásos szemi-on-line feladata bizonyítottuk, azonban megmutattuk, hogy ez érvényes a *c*-átpakolásos teljesen dinamikus ládapakolási feladatra is. A modell felírása után konstrukciónk egy nemlineáris optimalizálási problémához vezetett, amelynek megoldását átalakítások után megadtuk. A modellünkkel megadható alsó korlát kiszámításánál a Lambert-féle

W függvény tulajdonságait használtuk fel. Megjegyezzük továbbá, hogy alsó korlát konstrukciónk általánosítja a [15]-beli és a [17]-beli konstrukciókat.

Az [1] cikkben ugyanezen probléma azon speciális esetével foglalkoztunk, amikor a feladatnál pluszban még egy feltételt kirovunk: a megengedett különböző elemméretek maximális számát is korlátozzuk, egy előre rögzített p ($p \geq 2$) konstanssal. Az így nyert alsó korlátok szintén érvényesek mindkét, fentebb említett szemi-on-line feladat ezen speciális esetére. Továbbá érvényesek egy, a föntebb említett, [15]-ben definiált, harmadik, szintén a szemi-on-line ládapakolási feladatok körébe sorolható, úgynevezett „batched" ládapakolási feladatra, ennek a problémának is a szintén [15]-ben vizsgált 2-BBPP esetére, amikor a megengedett batchek száma 2. A korlátok javítják a $p \geq 3$ esetekben a [15]-beli alsó korlátokat (amelyek ezen feladat akár c-átpakolásos akár átpakolás nélküli változatában is érvényesek). Ezzel megválaszoltuk a [15]-ben felvetett kérdést, a 2-batched ládapakolási feladat szintén legfeljebb p különböző elemméretet megengedő esetére. A konkrét p-khez ($p \geq 2$) a konstrukció működik, és az adott alsó korlátok is természetesen érvényesek a fentebb említett másik két, c-átpakolásos szemi-on-line ládapakolási feladatra is. Bizonyítottuk, hogy bár korlátainkat egy dimenzióban fogalmaztuk meg, a konstrukció és az eredmények minden d dimenzióban ($d \geq 2$) dimenzióban is érvényesek (mindhárom feladatra).

Az [1] és a [2] cikkekben megadott alsó korlátok érdekessége, hogy habár a feladat tisztán diszkrét optimalizálási, a megoldáshoz speciális (folytonos) nemlineáris optimalizálási problémák megoldása szükséges, és ehhez globális optimalizálási eszközöket használtunk. Így a megadott és elemzett alsó korlát konstrukció és az adódó feladat megoldása egy szép kapcsolatot mutat az eredeti diszkrét (kombinatorikus) optimalizálási feladat, és a (folytonos) globális optimalizálás között. Az [1] cikkben a bizonyítás, a modellünk vizsgálatával adódó nemlineáris optimalizálási probléma megoldása globális optimalizálási eszközökkel történt, intervallummatematikán alapuló, megbízható, Branch-and-Bound technikát használó módszerrel [20, 21]. A megbízhatóság azt jelenti, hogy bár bizonyításunk számítógépes eszközökkel történt, az eredmény mégis ellenőrzött, azaz kiküszöböli a kerekítési hibákból (vagy azok felhalmozódásából) származó esetleges pontatlanságokat.

Egy további dolgozatban [3] korábbi eredményünket továbbfejlesztve [4] felső korlátokat adtunk a c-átpakolásos szemi-on-line ládapakolási feladatra, amelynél már „tisztán" diszkrét optimalizálási, azaz a ládapakolás szakirodalmában klasszikusnak számító algoritmus-elemzési eszközökkel (pl. súlyfüggvény-technika) dolgoztunk. Egy algoritmussorozatot adtunk meg, egészen pontosan minden pozitív egész c értékre egy, HFR-c nevű algoritmust. Bizonyítottuk, hogy ha c tart a végtelenbe, akkor az algoritmus(ok) aszimptotikus versenyképessége, *AVK(HFR-c)* tart *3/2*-hez.

A megadott felső korlátok bizonyítják, hogy az on-line feladat lépésenként c elem átpakolásának megengedésével kapott meglazításánál (ami immár szemi-on-line feladat) jól kihasználható ez a plusz „engedmény". Ehhez két konkrét esetet érdemes kiemelni. A $c=2$ esetre megadott a HFR–2 algoritmusunk aszimptotikus versenyképessége kisebb, vagy egyenlő, mint *1,5728...*, ami alatta van a legjobb ismert on-line algoritmusra bizonyított aszimptotikus versenyképességnek [23]. A $c=2$ eset másik érdekessége, hogy annak AVK-ja alatta van az úgynevezett Harmonic Fit típusú on-line ládapakolási algoritmusokra bizonyított *1,58333*-as alsó korlátnak [22]. A $c=1$ esetre adott algoritmusunk eredménye irreleváns, abból a szempontból, hogy az aktuális legjobb on-line algoritmus [23] versenyképesebb. Megjegyezzük azonban, hogy a mi *1*-átpakolásos algoritmusunk jóval kevesebb ládaosztályt használ, mint az. A másik kiemelendő a $c=4$ esetre adott algoritmusunk, a HFR–4, ez az általunk megadott algoritmus-sorozatban az első olyan, amelynek az aszimptotikus versenyképességi hányadosa az on-line algoritmusokra bizonyított legjobb alsó korlát [24] értéke alatt van. Ez azt is jelenti, hogy biztosan versenyképesebb minden on-line algoritmusnál – nemcsak minden ma ismert on-line algoritmusnál, hanem minden, a jövőben tervezettnél is. Végezetül megjegyezzük, hogy ez természetesen csak úgy lehetséges, hogy az általunk adott algoritmusok nem on-line, hanem szemi-on-line algoritmusok, és a feladat meglazítása által rendelkezésre álló átpakolási lehetőség már kis c értékeknél is jól kihasználható. Ez egyben validálja a feladat definiálásának létjogosultságát, azaz azt jelenti, hogy van értelme ilyen feladatokat vizsgálni.

Számos nyitott kérdés felvethető, ezekből hármat említünk meg. Az első, hogy adjunk lépésenként egy elem átpakolását megengedő, *1,58333*-nál kisebb AVK-jú szemi-on-line ládapakolási algoritmust! A második, hogy adjunk lépésenként *4*-nél kevesebb elem átpakolását megengedő, *1,5401*-nél kisebb AVK-jú algoritmust! Továbbá, a harmadik nyitott probléma, amelyet megemlítünk a kis c értékekre ($c=1,2$) az alsó korlátok javítása.

Köszönetnyilvánítás. A kutatást támogatta az OTKA (T 048377 és T 046822 számú projektek), és a MÖB-DAAD Magyar-Német Kutatócsere Program (21. sz. projekt).

Hivatkozások

[1] Balogh, J., J. Békési, G. Galambos, and M.Cs. Markót, Analysis of a Nonlinear Optimization Problem Related to Semi-on-line Bin Packing Problems, Proceedings of the International Workshop on Global Optimization, pp. 29–34, San José, Spain, September 18–22, 2005.

[2] Balogh, J., J. Békési, G. Galambos and G. Reinelt, Lower bound for bin packing problem with restricted repacking. Közlésre benyújtva, 2005. Elérhető: http://www.jgytf.u-szeged.hu/~bekesi/crestbinp.ps.

[3] Balogh, J., J. Békési, G. Galambos, and G. Reinelt, On-line Bin Packing with Restricted Repacking, *közlésre benyújtva*, (2006).

[4] Balogh J. és Galambos G., Átpakolást használó szemi-on-line ládapakolási algoritmusok, *Alkalmazott Matematikai Lapok*, **24**(2007):117–130.

[5] Coffman, E. G., G. Galambos, S. Martello, and D. Vigo, Bin packing approximation algorithms: Combinatorial analysis, in *Handbook of Combinatorial Optimization Supplement Volume*, eds.: Du, D.-Z. and P.M. Pardalos, Kluwer Academic Publishers, 1999, 151–208.

[6] Coffmann, E. G., M. R. Garey, and D.S. Johnson, Dynamic bin packing, *SIAM Journal on Computing*, **12**(2):227–260, 1983.

[7] Csirik, J. and G.J. Woeginger, On-line packing and covering problems, in: *On-line algorithms, Lecture Notes in Computer Science*, eds.: Fiat, A. and G. Woeginger, Vol. 1442, Berlin, 1998, 147–177.

[8] Fernandez de la Vega, W. and G.S. Lueker, Bin packing can be solved within $1+\varepsilon$ in linear time, *Combinatorica*, **1**(1981):349–355.

[9] Galambos, G., *A new heurisic for the classical bin packing problem*, Technical Report 82, Institute fuer Mathematik, Augsburg, 1985.

[10] Galambos, G. and G.J. Woeginger, Repacking helps in bounded space on-line bin packing, *Computing*, **49**(1993):329–338.

[11] Gambosi, G., A. Postiglione, and M. Talamo, New algorithms for on-line bin packing, in *Algorithms and Complexity, Proceedings of the First Italian Conference*, eds.: Petreschi R., G. Aussiello, D.P. Bovet, World Scientific, Singapore, 1990, 44–59.

[12] Gambosi, G., A. Postiglione, and M. Talamo, Algorithms for the relaxed on-line bin-Packing model, *SIAM Journal on Computing*, **30**(5):1532–1551, 2000.

[13] Garey, M. R. and D. S. Johnson, *Computers and Intractability (A Guide to the theory of NP-Completeness)*. W.H. Freeman and Company, San Francisco, 1979.

[14] Grove, E. F., On-line bin packing with lookahead, *SODA*, 1995, 430–436.

[15] Gutin, G., T. Jensen, and A. Yeo, Batched bin packing, *Discrete Optimization*, **2**(1): 71–82, 2005.

[16] Gutin, G., T. Jensen, and A. Yeo, Optimal on-line bin packing with two item sizes, *Algorithmic Oper. Research*, **1**(2006):72–78.

[17] Ivkovič, Z. and E. L. Lloyd, A fundamental restriction on fully dynamic maintenance of bin packing, *Information Processing Letters*, **59**(4):229–232, 1996.

[18] Ivkovič, Z. and E. L. Lloyd, Fully dynamic algorithms for bin packing: being (mostly) myopic helps, *SIAM Journal on Computing*, **28**(2): 574–611, 1998.

[19] Karmarkar, N. and Karp, R., An efficient *approximation* scheme for the one-dimensional bin packing problem, Proc. 23rd Annual Symposium Foundation Computer Science (FOCS): 312–320, 1982.

[20] Markót, M. Cs. and T. Csendes, A *new* verified optimization technique for the „packing circles in a unit square" problems, *SIAM Journal. on Optimization*, **16**(2005):193–219.

[21] Markót, M. Cs., T. Csendes, and A.E. Csallner, Multisection in interval branch-and-bound methods for global optimization. II. Numerical tests. *Journal of Global Optimization*, **16**(3):219–228, 2000.

[22] Ramanan, P., D.J. Brown, C.C. Lee, and D.T. Lee, On-line bin packing in linear time, *Journal of Algorithms*, **10**(3):305–326, 1989.

[23] Seiden, S. S., On the on-line bin packing problem, *Journal of the ACM*, **49**(5):640–671, 2002.

[24] Van Vliet, A., An improved lower bound for on-line bin packing algorithms, *Information Processing Letters*, **43**(5): 277–284, 1992.

BALOGH JÁNOS – BÉKÉSI JÓZSEF – GALAMBOS GÁBOR

Bin-packing with repacking

In contrast to on-line bin packing, semi-on-line bin-packing allows the algorithm to carry out extra operations, in addition to the packing of the actual element, in each step of the process. These extra operations might include at least one of the following operations: repacking, reordering or buffering.

This paper defines and analyses a semi-on-line bin-packing problem, where repacking is allowed, but only for a restricted number of elements. We provide lower and upper bounds for the problem, and lower bounds for some special cases of the problem. The lower bounds also apply to some related problems.