Online algoritmusok versenyképességi elemzése

MTA doktora disszertáció
tézisfüzete

Imreh Csanád

Informatika Intézet, Szegedi Tudományegyetem

Szeged

2016

1. Online algoritmusok

A gyakorlati problémákban gyakran fordulnak elő olyan optimalizálási feladatok, ahol az inputot csak részenként ismerjük meg, és a döntéseinket a már megkapott információk alapján, a további adatok ismerete nélkül kell meghoznunk. Ilyen feladatok esetén online problémáról beszélünk. Az online algoritmusok elméletének igen sok alkalmazása van a számítástudomány, az operációkutatás és a közgazdaságtan különböző területein.

Az első eredmények az online algoritmusok elméletének területéről az 1970-es évekből származnak, majd a 90-es évek elejétől kezdve egyre több kutató kezdett el az online algoritmusok területéhez kapcsolódó problémákkal foglalkozni. Számos részterület alakult ki és napjainkban is a legfontosabb, algoritmusokkal foglalkozó konferenciákon rendszeresen ismertetnek új eredményeket ezen témakörből.

Mivel egy online algoritmusnak részenként kell meghozni a döntéseit a teljes input ismerete nélkül, ezért egy ilyen algoritmustól nem várhatjuk el, hogy a teljes információval rendelkező algoritmusok által megkapható optimális megoldást szolgáltassa. Azon algoritmusokat, amelyek ismerik a teljes inputot offline algoritmusoknak nevezzük.

Az online algoritmusok hatékonyságának vizsgálatára két alapvető módszert használnak. Az egyik lehetőség az átlagos eset elemzése. Ebben az esetben fel kell tételeznünk valamilyen valószínűségi eloszlást a lehetséges inputok terén, és az erre az eloszlásra vonatkozó várható értékét vizsgáljuk a célfüggvénynek. Ezen megközelítés hátránya, hogy általában nincs információnk arról, hogy a lehetséges inputok milyen valószínűségi eloszlást követnek. Mi a disszertációban az átlagos eset elemzésének témakörével nem foglalkozunk, hanem az elterjedtebb versenyképességi analízis módszerét használjuk.

A másik megközelítés egy legrosszabb-eset korlát elemzés, amelyet versenyképességi elemzésnek nevezünk. Ebben az esetben az online algoritmus által kapott megoldás célfüggvényértékét hasonlítjuk össze az optimális offline célfüggvényértékkel.

Egy online minimalizálási probléma esetén egy online algoritmust C-versenyképesnek nevezünk, ha tetszőleges inputra teljesül, hogy az algoritmus által kapott megoldás költsége nem nagyobb, mint C-szer az optimális offline költség. Egy algoritmus versenyképességi hányadosa a legkisebb olyan C szám, amelyre az algoritmus C-

versenyképes.

Általában egy tetszőleges ALG online algoritmusra az I inputon felvett célfüggvényértéket $\mathrm{ALG}(I)$-vel jelöljük. Az I inputon felvett optimális offline célfüggvényértéket $\mathrm{OPT}(I)$-vel jelöljük. Használva ezt a jelölésrendszert a fent definiált versenyképességet minimalizálási problémákra a következőképpen adhatjuk meg.

Az ALG algoritmus C-versenyképes ha $\mathrm{ALG}(I) \leq C \cdot \mathrm{OPT}(I)$ teljesül minden I input esetén.

Szokásos használni a versenyképesség egy további változatát. Egy minimalizálási probléma esetén az ALG algoritmus aszimptotikusan C-versenyképes, ha van olyan B konstans, hogy $\mathrm{ALG}(I) \leq C \cdot \mathrm{OPT}(I) + B$ teljesül minden I input esetén. Egy algoritmus aszimptotikus versenyképességi hányadosa a legkisebb olyan C szám, amelyre az algoritmus aszimptotikusan C-versenyképes.

Fontosnak tartjuk megjegyezni, hogy a fenti fogalomra néha használják a gyenge versenyképesség kifejezést is, illetve sok esetben ezt nevezik versenyképességnek és az additív konstanst nem megengedő fogalmat pedig abszolút versenyképességnek. A aszimptotikus versenyképességi hányadost ($R^{\infty}_{\mathrm{ALG}}$) több ládapakolási dolgozatban a következő formulákkal definiálják.

$$R^{n}_{\mathrm{ALG}} = \max\{\mathrm{ALG}(I)/\mathrm{OPT}(I) \mid \mathrm{OPT}(I) = n\}$$

$$R^{\infty}_{\mathrm{ALG}} = \limsup_{n\to\infty} R^{n}_{\mathrm{ALG}}.$$

A disszertációban (és számos egyéb dolgozatban) használt additív konstanst megengedő definíció nem ekvivalens a lim sup függvény alapján kapott definícióval, ez utóbbi például az additív konstans helyett megenged tetszőleges $o(OPT(I))$ nagyságú additív függvényt is. Másrészt a legtöbb esetben, és a dolgozatban bemutatott eredményeknél is, az additív tag konstans, így mindkét definíció szerint ugyanazt az aszimptotikus versenyképességi hányadost kapjuk. Az aszimptotikus hányados fő tulajdonsága az, hogy azt vizsgálja, miként viselkedik az algoritmus akkor, ha az optimum értéke nagy, azaz ha az optimális költség végtelenhez tart. Ez általában azt jelenti, hogy az algoritmus szabadon dönthet az input kezdeti részeinél.

A fentiekben a minimalizálási problémákra definiáltuk a versenyképességi analízis fogalmait. A definíciók hasonlóan értelmezhetőek

maximalizálási problémák esetén is. Ekkor az ALG algoritmus C-versenyképes ha $C \cdot \mathrm{ALG}(I) \geq \mathrm{OPT}(I)$ teljesül minden I input esetén, illetve aszimptotikusan C-versenyképes ha valamely B konstans mellett $C \cdot \mathrm{ALG}(I) + B \geq \mathrm{OPT}(I)$ teljesül minden I inputra.

Számos tudományos dolgozat vizsgál véletlenített online algoritmusokat. Ebben az esetben az algoritmus véletlen döntéseket is hoz, így az általa kapott célfüggvényérték egy valószínűségi változó, és a versenyképességi hányados definíciójában ezen valószínűségi változó várható értéke szerepel. Mivel a disszertációban csak determinisztikus online algoritmusokkal fogunk foglalkozni, ezért a véletlenített algoritmusokra vonatkozó fogalmakat nem részletezzük.

A disszertációban három részterülettel foglalkozunk. Elsőként a gépköltséges ütemezési problémákra elért eredményeinket mutatjuk be, majd különböző ládapakolási és ládafedési modellekkel foglalkozunk. Végül az online klaszterezés területén elért eredményeket ismertetjük.

2. Online ütemezés gépköltséggel

A legelterjedtebb online ütemezési modell a lista modell, ahol adott m darab azonos gép és amelyben a munkák egy listáról érkeznek. Amikor egy munkát megkapunk a listáról, akkor ismerjük meg az elvégzéséhez szükséges végrehajtási időt, és ezt követően ütemeznünk kell a munkát valamely gépen hozzárendelve a kezdési és befejezési időt, amelyeket később már nem változtathatunk meg, és csak ezt követően kapjuk meg a listáról a következő munkát. A legszélesebb körben vizsgált célfüggvény a maximális befejezési idő minimalizálása. Ebben az esetben (miként a probléma offline változatában is) elegendő olyan algoritmusokkal foglalkoznunk, amelyek nem hagynak üres időintervallumokat a gépeken, azaz amelyekben az egyes gépeken a munkák szünet nélkül követik egymást. Ekkor minden gépre a maximális befejezési idő megegyezik a géphez rendelt munkák végrehajtási idejeinek összegével. A gépen levő munkák végrehajtási idejeinek összegét a gépen levő töltésnek hívjuk. Tehát ebben az esetben elegendő csak azt megmondani, hogy az adott munkát, melyik géphez rendeljük, és a cél a maximális töltés minimalizálása. Ez az egyik első online probléma, ami publikálásra került. A [21] cikkben a Lista algoritmust elemezték, amely az aktuális munkát

mindig ahhoz a géphez rendeli, ahol a töltés minimális. Az algoritmus versenyképességi hányadosa $2 - 1/m$.

Ütemezési feladatok esetén általában a gépek száma adott paramétere a feladatnak. Ugyanakkor számos alkalmazás esetén a gépek száma megváltoztatható. Az ilyen problémák vizsgálhatóak a gépköltséges modellben, amit a [27] cikkünkben vezettünk be. Ebben a modellben a gépek száma nem adott, hanem az algoritmusnak meg kell vásárolnia azokat, és a cél a gépek vásárlására költött költség és a maximális befejezési idő (ami ebben a modellben megegyezik a maximális töltéssel) összegének a minimalizálása. A [27] cikkben a gépköltséges feladatokra a következő algoritmusosztályt vizsgáltuk meg. Egy tetszőleges növekvő $\varrho = (0 = \varrho_1, \varrho_2, \dots, \varrho_i \dots)$ sorozatra definiálhatjuk a következő A_ϱ algoritmust. Amikor a j_ℓ munka megérkezik A_ϱ annyi gépet vásárol (ha szükséges), hogy a gépek i száma teljesítse a $\varrho_i \leq P < \varrho_{i+1}$ egyenlőtlenséget, ahol P az eddig megérkezett munkák végrehajtási idejeinek az összege. Az A_ϱ algoritmus a fentiekben említett LISTA algoritmus alapján ütemezi a munkákat, az esetleges gépvásárlást követően hozzárendeli az aktuális munkát ahhoz a géphez, ahol a töltés minimális.

A [27] cikket követően számos eredményt publikáltak a probléma és annak különböző változatainak megoldására, de a [26] cikket megelőzően ezek mindegyike feltételezte, hogy a gépek vásárlási költsége minden gépre megegyezik. Ebben a cikkben egy általánosabb modellt vizsgáltunk, ahol a gépek költségét egy c monoton nemcsökkenő költségfüggvény írja le. Ekkor $c(i)$ az első i darab gép megvásárlásának költségét adja meg, azaz az i-dik gép ára $c(i)-c(i-i)$, a $c(0) = 0$ értéket használva. Ebben az általános modellben is vizsgáltuk az A_ϱ algoritmusokat és az alábbi eredményeket kaptuk.

1. Tétel. *([26]) Ha* $\varrho = (0, c(2)\varphi, 2c(3)\varphi, \dots, (i-1)c(i)\varphi, \dots)$*, akkor az* A_ϱ *algoritmus* $1 + \varphi \approx 2.618$*-versenyképes, ahol* $\varphi = (1 + \sqrt{5})/2$*.*

2. Tétel. *([26]) Ha* $\varrho = (0, c(2), 2c(3), \dots, (i-1)c(i), \dots)$ *és minden munka mérete legfeljebb* $\max_i\{c(i)-c(i-1)\}$ *akkor az* A_ϱ *algoritmus versenyképességi hányadosa* 2*.*

Az alábbi alsó korlát, amit a lehetséges versenyképességi hányadosokra igazoltunk mutatja, hogy a kis munkák esetén sikerült optimális versenyképességű algoritmust találnunk.

3. Tétel. *([26]) Van olyan c költségfüggvény, amelyre az általános költségű gépköltséges ütemezési feladatra nincs olyan online algoritmus, amelynek kisebb a versenyképességi hányadosa, mint* 2. *Az alsó korlát fennáll olyan speciális esetre is, ahol minden munka mérete legfeljebb* $\max_i\{c(i) - c(i-1)\}$.

Szintén megvizsgáltunk egy olyan változatot is, ahol a gépeknek különböző sebességei lehetnek. Ekkor a munka megadott végrehajtási idejét osztani kell a gép sebességével és így kapjuk meg az adott gépen megjelenő töltést. A vizsgált gépköltséges változatban két géphalmazunk van, S_1 olyan gépeket tartalmaz, amelyek sebessége 1, és S_2 olyan gépeket, amelyek sebessége $s > 1$. Az algoritmus mindkét halmazból vásárolhat gépeket. Az S_1 halmaz gépeinek költségét egy c_1 nemcsökkenő függvény írja le ($c_1(k)$ a halmaz első k gépének megvásárlási költsége) és az S_2 halmaz gépeinek költségét egy c_2 nemcsökkenő függvény írja le ($c_2(k)$ a halmaz első k gépének megvásárlási költsége). A célfüggvény itt is, a gépek vásárlására költött összegnek és a maximális befejezési időnek az összegének a minimalizálása.

A következő GRM (Greedy for Related Machines) algoritmusnak vizsgáltuk a versenyképességét. Az algoritmus minden lépésben használja az OPT_ℓ értéket, ami az első ℓ munkából álló input optimális megoldásának költsége. Amikor egy új j_ℓ munka érkezik a GRM algoritmus annyi gépet vesz (amennyiben szükséges), hogy a gépek i_1, i_2 számára a két osztályban teljesüljön $c_1(i_1) \leq OPT_\ell < c_1(i_1+1)$ és $c_2(i_2) \leq OPT_\ell < c_2(i_2+1)$. Ezt követően az algoritmus a munkát a LISTA algoritmus különböző gépekre vonatkozó kiterjesztése alapján ütemezi, arra a gépre rakja, ahol a munka ütemezését követően a töltés minimális lesz. Ha több ilyen gép is van, akkor a gyorsabb gépek közül választja a legkisebb indexűt. Az algoritmus versenyképességét a következő tétel adja meg.

4. Tétel. *[26] A* GRM *algoritmus versenyképességi hányadosa 6.*

Fontos megemlíteni, hogy OPT_ℓ meghatározása egy NP-nehéz feladat, így az algoritmusunk futási ideje exponenciális. Másrészt az optimális megoldás értéke helyett használhatunk egy approximációs algoritmus vagy approximációs séma által kapott értéket is, csak akkor a versenyképességi hányados is növekszik (c-approximációs algoritmus esetén $4 + 2c$-re).

Külön vizsgáltuk azt a problémát is, amelyben mindkét halmazban adott a gépek száma, és csak ütemezni kell a munkákat. Ez felfogható úgy is, hogy a gépek vásárlási függvénye olyan, hogy az egyes halmazokból k illetve m gépet ingyen megkapunk, a többiekért pedig egy végtelen nagy összeget kell fizetni. Ebben az esetben a GRM algoritmus 4-versenyképes, és megadtunk egy, a [25] cikkünkben korábban ismertetett algoritmus továbbfejlesztésén alapuló 3-versenyképes algoritmust is.

Az [12] cikkben egy más irányú kiterjesztésével foglalkoztunk a gépköltséges ütemezési feladatnak. Ezt az ütemezési problémát geometriai értelemben felfoghatjuk úgy is, hogy egység és végrehajtási idő oldalakkal rendelkező téglalapokat kell elhelyeznünk a gépek által kijelölt sávokban minimalizálva a gépek számának és a felhasznált sávok maximális hosszának az összegét. Ennek a folytonos változata az, hogy a téglalapokat tetszőlegesen helyezhetjük el egy befoglaló téglalapban, a befoglaló téglalap oldalainak összegének minimalizálásával. Pontosabban egy általánosabb $\gamma H+W$ célfüggvényt vizsgáltunk, ahol H a befoglaló téglalap magassága, W a befoglaló téglalap szélessége, $\gamma > 0$ pedig a feladat egy paramétere. A probléma azt az általános erőforrás allokációs modellt írja le, ahol az inputként érkező téglalapok szélessége a feladat végrehajtáshoz szükséges erőforrás mennyiségét, a magassága pedig a végrehajtáshoz szükséges időt adja meg. A befoglaló téglalap oldalai pedig a teljes sorozat végrehajtásához egy időben igénybe vett maximális erőforrás mennyiségét és a végrehajtáshoz szükséges időt adják meg. Ezek súlyozott összege a minimalizálandó célfüggvény. Az online változatban a téglalapok egyenként jönnek, és minden téglalapot a további téglalapokra vonatkozó információk nélkül kell elhelyeznünk az eddigiekkel való átfedés nélkül a síkon. A befoglaló téglalap a legkisebb olyan téglalap lesz, ami minden lerakott kis téglalapot tartalmaz az inputból.

Amennyiben a befoglaló téglalap egyik oldalának a mérete rögzített, akkor a probléma úgy fogalmazható meg, hogy egy adott szélességű sávba kell pakolnunk átfedés és forgatás nélkül téglalapokat, a felhasznált rész magasságát minimalizálva. Ezt a sávpakolási feladatot, amit a ládapakolási probléma kétdimenziós kiterjesztéseként is lehet definiálni több tanulmányban is vizsgálták. A jelenleg ismert legjobb algoritmusok 6.623-versenyképesek, ilyen algoritmusokat publikáltak [23] és [33] cikkekben. Több cikkben is pub-

likáltak egyre nagyobb alsó korlátokat a lehetséges versenyképességi hányadosra, a jelenlegi legnagyobb alsó korlát 2.589, amit a [22] cikkben igazoltak. Aszimptotikus versenyképesség szempontjából az elérhető legkisebb aszimptotikus versenyképességi hányados $h_\infty \approx 1.69103$, amit egy, a [10] cikkben bemutatott polcpakolási algoritmus ér el. A polcpakolási algoritmusok lényege, hogy vízszintes részsávokat (polcokat) hozunk létre a befoglaló sávon belül és az aktuális téglalapot mindig valamelyik polcra helyezzük el.

Az általunk vizsgált általánosabb téglalap pakolási problémára a következő polcpakolási algoritmust fejlesztettük ki. A SHELF(α) algoritmus az érkező téglalapokra elsőként mindig meghatározza azok típusát. Egy $p_i = (w_i, h_i)$ méretű téglalap érkezése esetén ez az a k szám lesz, melyre a $2^{k-1} < h_i \leq 2^k$ egyenlőtlenség teljesül. Ezt követően, ha van nyitott k típusú (2^k magas) polc, akkor rakjuk a téglalapot erre a polcra annyira balra, amennyire lehetséges. Amennyiben, ezt követően a polcon felhasznált szélesség eléri a befoglaló téglalap aktuális magasságának α-szorosát zárjuk le a polcot. Ha pedig nincs nyitott k típusú polc (még egyáltalán nem hoztunk ilyet létre vagy az utolsó k típusú téglalapnál lezártuk a k típusú polcot), akkor hozzunk létre egy ilyen polcot a meglevő polcok tetején, és rakjuk a téglalapot a polc bal sarkába. Amennyiben, ezt követően a polcon felhasznált szélesség eléri a befoglaló téglalap aktuális magasságának α-szorosát zárjuk le a polcot.

Az algoritmus versenyképességére vonatkozik az alábbi tétel.

5. Tétel. *[12] A* SHELF(α) *algoritmus* $\left(4\frac{\alpha}{\gamma} + \sqrt{\frac{\alpha}{\gamma}} + 4 + \sqrt{\frac{\gamma}{\alpha}}\right)$*-versenyképes. Ha úgy választjuk meg az* α *paramétert, hogy* $\sqrt{\alpha/\gamma} = 0.46161$ *teljesüljön, akkor az algoritmus* 7.4803*-versenyképes.*

Szintén megvizsgáltuk azt a félig-online esetet, ahol előre tudjuk, hogy a téglalapok csökkenő magasság szerinti sorrendben érkeznek. Ez a tulajdonság nagymértékben könnyebbé teszi a feladat megoldását, miként azt a következő algoritmus mutatja. A SDH(β) algoritmus az első téglalaphoz definiál egy polcot, amelynek a magassága ezen téglalap magassága, és amely bal sarkában ez a téglalap van. Ez a polc lesz az aktuális polc. A további téglalapokat az alábbi szabály szerint pakoljuk. Ha a szélessége az aktuális polcnak legfeljebb β-szor akkora, mint az aktuális magassága a befoglaló téglalapnak, akkor rakjuk a téglalapot erre a polcra annyira balra,

amennyire lehetséges. Ellenkező esetben zárjuk be az aktuális polcot, amit nem fogunk többet használni. Nyissunk egy új polcot az eddigi polcok tetején, legyen a magassága az aktuálisan elpakolandó téglalap magassága és rakjuk a téglalapot az új polc bal sarkába.

Ezen algoritmus versenyképességére vonatkozik az alábbi állítás.

6. Tétel. *[12] A* SDH(β) *algoritmus* 2.5*-versenyképes, amennyiben* $\beta = 0.6\gamma$.

Számos gyakorlati problémában fordul elő, hogy egy elvégzendő feladat esetén sem az ahhoz rendelt erőforrás mennyisége sem pedig a végrehajtás ideje nem rögzített, hanem megtehetjük azt, hogy növelve a használt erőforrás mennyiségét csökkentjük a végrehajtási időt vagy csökkentve a felhasznált erőforrás mennyiségét növeljük a végrehajtási időt. A sávpakolási problémákban ez úgy írható le, hogy a téglalapok mérete megváltoztatható. Egy ilyen változatát az online sávpakolási problémának, ahol a téglalapok megnyújthatóak a terület fixen hagyása mellett, vizsgáltunk a [24] dolgozatban. Ezeket az eredményeket nem tartalmazza a disszertáció. A disszertációban a változtatható téglalapok esetét az általánosabb modellre vizsgáltuk, ahol a befoglaló téglalap egyik oldala sem rögzített. Ekkor a téglalapnak az inputban csak a területét kapjuk meg, az algoritmusnak kell eldöntenie azt, hogy milyen formájú téglalapot szeretne elpakolni a befoglaló téglalapba.

A módosítható méretű téglalapok esetére, a $\gamma = 1$ esetben a következő EXPAND(1) algoritmust fejlesztettük ki, amely alapötlete, hogy megpróbál négyzethez minél hasonlóbb befoglaló téglalapot létrehozni. Ha a k-adik téglalap területét $T(k)$, a k-adik téglalap érkezését megelőzően a befoglaló téglalap oldalait $A(k) \leq B(k)$ jelölik, akkor a k-adik téglalap $a(k), b(k)$ oldalait és az elhelyezkedését az alábbi szabályokkal adhatjuk meg.

- Ha a következő téglalap kicsi, azaz $T(k) \leq B(k)^2$, akkor legyen $b(k) = B(k)$ és $a(k) = \frac{T(k)}{b(k)} \leq B(k)$. Majd ragasszuk az így kapott téglalapot a nagyobbik oldalával a befoglaló téglalaphoz, azaz legyen $B(k+1) = \max\{B(k), A(k) + a(k)\}$ és $A(k+1) = \min\{B(k), A(k) + a(k)\}$.

- Ha a következő téglalap nagy, azaz $T(k) > B(k)^2$, akkor legyen $a(k) = b(k) = \sqrt{T(k)}$. Majd ragasszuk az így kapott négyzetet

a befoglaló téglalap nagyobbik oldalához, azaz legyen $A(k+1) = a(k)$ and $B(k+1) = A(k) + a(k)$.

Az általános γ esetét visszavezettük a $\gamma = 1$ speciális esetre, az igazi téglalapok és igazi befoglaló téglalap helyett módosított virtuális téglalapokat használva. Az így kapott EXPAND(γ) algoritmus versenyképességére vonatkozik az alábbi állítás.

7. Tétel. *[12] Az* EXPAND(γ) *algoritmus* $\frac{\sqrt{21}}{4} \approx 1.1456$*-versenyképes.*

3. Ládapakolási és fedési problémák

A ládapakolási problémában inputként tárgyak egy sorozatát kapjuk meg, ahol az i-edik tárgyat a mérete határozza meg, ami egy $s_i \in (0, 1]$ érték. Célunk a tárgyak elhelyezése a lehető legkevesebb egység méretű ládába. Formálisabban megfogalmazva a tárgyakat minimális számú olyan csoportba akarjuk szétosztani, hogy minden csoportra a benne levő tárgyakra a $\sum s_i \leq 1$ feltétel teljesüljön. A feladat online változatában a tárgyak egyenként érkeznek és az aktuális tárgyat a további tárgyakra vonatkozó információk nélkül kell elhelyeznünk valamely ládába. Egy ládára a benne levő tárgyak méreteinek összegét a láda töltésének hívjuk.

Az online ládapakolásra és változataira számos algoritmust fejlesztettek ki. Ezen algoritmusok elemzésére többnyire az aszimptotikus versenyképességi hányadost használják. Az algoritmusok egy nagy osztálya a fit típusú algoritmusokat tartalmazza, amelyek csak akkor nyitnak új ládát, ha az aktuális tárgy egyetlen nyitott ládába sem fér el. Több algoritmus került kifejlesztésre attól függően, hogy a használható ládák közül melyiket választjuk. A két legismertebb ilyen algoritmus a következő. A FF algoritmus soha nem zár be ládákat és az aktuális tárgyhoz az első olyan ládát választja, amelybe a tárgy belefér. A NF algoritmus pedig mindig csak egy ládát tart nyitva, és ha abban az aktuális tárgy nem fér el, akkor a ládát bezárja és egy új ládát nyit a tárgynak. Egy másik nagy osztálya az algoritmusoknak a harmonic típusú algoritmusok osztálya, ahol a tárgyakat méret szerint online osztályozzuk és a különböző osztályokat külön ládahalmazokba pakoljuk. A legkisebb aszimptotikus versenyképességgel a [30] cikkben publikált algoritmus rendelkezik, amely 1.58889-versenyképes. A probléma de-

finiálása óta többször javították a versenyképességre vonatkozó alsó korlátokat, a jelenlegi legjobb korlát $248/161 \approx 1.54037$, amit a [5] cikkben publikáltak.

A ládafedési feladat a ládapakolási feladat duálisa. Az inputot ugyanúgy a tárgyak méretei adják, de a célunk most a maximális számú egységláda lefedése. Azaz a tárgyakat a maximális számú olyan csoportba szeretnénk csoportosítani, ahol minden csoportban legalább 1 a tárgyak összege. A feladatot a [2] cikkben kezdték el vizsgálni, ahol igazolták, hogy az online NF algoritmus, ami a tárgyakat addig rakja az aktuális ládába, amíg le nem fedi és csak utána nyit új ládát, 2-versenyképes. Ez a korlát egyből adódik mivel minden ládában a töltés legfeljebb 2. A cikkben offline approximációs algoritmusokat is vizsgáltak egy $\frac{3}{2}$ és egy $\frac{4}{3}$ approximációs algoritmust adtak meg. Később a [9] cikkben igazolást nyert, hogy nem adható meg olyan online algoritmus a ládafedési problémára, amelynek az aszimptotikus versenyképessége kisebb, mint 2.

A klasszikus ládapakolási feladatban nagyon kicsi a rés az alsó és felső korlátok között, a ládafedés esetén pedig ismert a legkisebb versenyképességű algoritmus. A kutatások főleg különböző változatok illetve kiterjesztések területén folynak. Mi a disszertációban korlátozott ládapakolási és ládafedési feladatokkal foglalkoztunk, ahol a láda tartalmára a kapacitásán kívül további korlátok is vannak. Ezeket az eredményeket foglaljuk össze a következőkben. Majd egy olyan ládafedési változattal foglalkozunk, ahol a ládák száma adott és a cél a használt tárgyak méreteinek összegének minimalizálása.

3.1. Színkorlátos ládapakolás

A színkorlátos ládapakolási feladatban, a tárgyaknak nem csak mérete van, hanem minden tárgynak adott egy c_i színe is és adott egy k szám. A ládák tartalmára a méretek összegére vonatkozó korláton kívül még teljesülnie kell annak is, hogy a tárgyak legfeljebb k különböző színosztályba tartoznak. Az online problémát elsőként a [31, 32] cikkekben vizsgálták, ahol két FIRST FIT (FF) típusú algoritmus elemeztek. Az első FF-nek nevezett algoritmus egy tárgy érkezésekor az első olyan ládába pakolja azt, ahol nem sérti sem a méretekre sem a színek számára vonatkozó korlátot. Ha nincs ilyen láda, akkor az algoritmus új ládát nyit a tárgynak. A másik vizsgált eljárás a (CSFF) (COLOR SETS FIRST FIT) algoritmus. Ebben az al-

goritmusban a színeket online csoportosítjuk k elemű halmazokba, az elsőként megjelent k szín tartozik az első csoportba, utána mindig a még nem látott következő k szín alkotja a következő csoportot. Minden színosztályra külön ládahalmazokon futtatjuk a FF algoritmust. A [31] cikkben azt a speciális esetet vizsgálták, ahol minden tárgy mérete ugyanakkora. Igazolták, hogy mind a CSFF mind pedig a FF algoritmus aszimptotikus versenyképessége 2. Az általános esetet, ahol különböző méretűek lehetnek a tárgyak a [32] cikkben vizsgálták. Itt egy, a tárgyak méret szerinti osztályozásán alapuló algoritmust elemeztek, amely aszimptotikusan 2.75-versenyképes. Ha minden tárgy színe különböző, akkor a színekre vonatkozó korlát arra redukálódik, hogy minden ládába legfeljebb k darab tárgyat lehet rakni. Ezt a modellt elemszámkorlátos ládapakolási feladatnak nevezik, és több tanulmány is foglalkozott az online és offline problémával, részletek találhatóak a [4, 14] cikkekben és az ott szereplő hivatkozásokban.

A problémát a [16] cikkben tovább vizsgáltuk, ezeket az eredményeket foglaljuk össze az alábbiakban. Tovább elemeztük a CSFF algoritmust az általános méretű tárgyak esetére és az alábbi eredményt igazoltuk.

8. Tétel. *([16]) A* CSFF *algoritmus aszimptotikusan* $(2 + \frac{k-1}{k})$*-versenyképes a színkorlátos ládapakolási feladatra.*

Továbbá kifejlesztettünk egy módszert, amely segítségével a ládapakolási algoritmusok egy széles osztálya transzformálható az általánosabb, színosztályokkal rendelkező feladatra a versenyképességi hányados növelése mellett.

Ehhez definiáltuk az uniform pakolási algoritmusok fogalmát. Legyen $\mathcal{A}$ egy olyan, egy t egész paramétertől függő ládapakolási algoritmus, amely szétosztja a tárgyakat kicsi (a $(0, \frac{1}{t+1}]$ intervallumba eső méretű) és nagy (a többiek) tárgyakra, ahol a nagy tárgyak esetleg további részhalmazokra oszthatók. Az algoritmust, akkor nevezzük uniformnak, ha a nagy tárgyakat a kis tárgyaktól függetlenül pakolja egy külön ládahalmazba, a kis tárgyakat pedig a NF algoritmus szerint.

Egy ilyen algoritmust a következőképpen terjeszthetünk ki a színkorlátos pakolási probléma megoldására. A nagy tárgyakat az $\mathcal{A}$ algoritmusnak megfelelően pakoljuk. Mivel minden tárgy nagyobb, mint $\frac{1}{t+1}$ ezért legfeljebb t ilyen tárgyat tartalmaznak a ládák. Ha

$t \leq k$, akkor ezek a ládák biztosan kielégítik a színek számára vonatkozó korlátokat. A kis tárgyakat pedig a NF algoritmus helyett annak a CSNF kiterjesztésével pakoljuk, ami a CSFF algoritmusnál használt módon szín szerinti csoportokat pakol diszjunkt ládahalmazokba, csak nem a FF hanem a NF algoritmus szerint. Ezt az algoritmust CS(A)-val jelöljük. A kiterjesztés csak kismértékben növeli a versenyképességi hányadost, amint ezt az alábbi állítás mutatja.

9. Tétel. *([16]) Legyen $\mathcal{R}$ egy felső korlát egy $\mathcal{A}$ uniform algoritmus aszimptotikus versenyképességi hányadosára a klasszikus ládapakolás esetén. Ekkor a* CS(A) *algoritmus aszimptotikusan $\mathcal{R} + 1$ versenyképes a színkorlátos ládapakolási feladatra.*

Nagy k esetén a fenti tételt használva ismert uniform algoritmusokra, kisebb k-ra pedig egy új, a tárgyak partícióján alapuló algoritmust fejlesztve igazoltuk az alábbi állítást.

10. Tétel. *([16]) Minden k érték esetén létezik egy aszimptotikusan legfeljebb* 2.63492*-versenyképes algoritmus a színkorlátos ládapakolási feladatra.*

Az alábbi alsó korlátot is igazoltuk, a lehetséges versenyképességi hányadosra.

11. Tétel. *([16]) Tetszőleges online algoritmusnak a színkorlátos ládapakolási feladatra az aszimptotikus versenyképességi hányadosa legalább 1.5652. A korlát már a $k = 2$ esetben teljesül.*

3.2. Színkorlátos ládalefedés

A színkorlátos ládalefedés a színkorlátos ládapakolás duálisa. Itt is minden tárgyhoz egy méret és egy szín van rendelve és adott egy k paraméter a feladatban. Egy színkorlátos ládafedésen a tárgyak egy olyan csoportosítását (ládákhoz rendelését) értjük, ahol minden csoportra teljesül, hogy a benne levő tárgyak méreteinek összege legalább 1 és az is, hogy a láda legalább k különböző színosztályból tartalmaz tárgyakat. A célunk egy olyan csoportosítás megtalálása, ahol a csoportok száma maximális. Erre a problémára mi értük el az első eredményeket a [17] dolgozatban.

A probléma egy speciális esetét vizsgáltuk, ahol minden tárgynak ugyanakkora a mérete. Ebben az esetben a probléma leírható két

pozitív egész számmal B-vel és k-val. Egy láda akkor lesz lefedve, ha legalább B tárgyat rakunk a ládába és ezek a tárgyak legalább k különböző színosztályhoz tartoznak. Ez azokat az eseteket írja le, ahol a tárgyak egységes mérete az $[1/B, 1/(B-1))$ intervallumba esik. Feltehetjük, hogy $B \geq k$, hisz $B < k$ esetén nem változtat a problémán, ha B értékét k-ra növeljük.

A probléma megoldására elsőként FF típusú algoritmusokat vizsgáltunk. Az FF(1) algoritmus az alábbi módon pakolja a tárgyakat. Mikor egy új tárgy érkezik a c színosztályból, akkor azt az első olyan ládához rendeljük, amelynek a lefedéséhez hozzá tud járulni. Ez azt jelenti, hogy az első olyan ládát választjuk ami kevesebb, mint B tárgyat tartalmaz, vagy legalább B tárgyat tartalmaz, de az tárgyak kevesebb, mint k színosztályból kerülnek ki és nincs c-beli tárgy a ládában. Ha nincs ilyen láda, akkor egy új ládát nyitunk az tárgynak. Az algoritmus hatékonyságát az alábbi tétel határozza meg.

12. Tétel. *([17]) Az* FF*(1) algoritmus aszimptotikus versenyképességi hányadosa* $B + k - 1$ *minden* $k \geq 2$ *és* B *értékre.*

Az FF(2) algoritmus az FF(1) egy olyan továbbfejlesztése, ami azt is figyelembe veszi, hogy ha egy ládában már vannak tárgyak de hiányzik $t \leq k - 1$ szín a lefedéshez, akkor ez a t extra tárgy hozzájárul a méret szerinti lefedéshez is. Tehát, ha egy ládában $k-t$ színosztályból $B-t$ tárgy van, akkor az olyan további tárgyak, amelyek már a ládában szereplő színosztályokhoz tartoznak valójában nem nyújtanak segítséget a láda lefedéséhez. Mindezek alapján azt mondjuk, hogy egy tárgynak egy még nem lefedett ládához való hozzárendelése akkor hasznos, ha a ládában még nincs ilyen színű tárgy, vagy ha van ilyen színű tárgy, és a ládából még hiányzó színosztályok száma kisebb, mint a ládából hiányzó tárgyak száma. Használva ezt a fogalmat az FF(2) algoritmus úgy definiálható, hogy az aktuális tárgyat az első olyan ládába teszi, amelyhez hasznos a hozzárendelése, ha nincs ilyen akkor pedig új ládát nyit. Az algoritmus versenyképességét adja meg az alábbi tétel.

13. Tétel. *([17]) Az* FF*(2) algoritmus aszimptotikus versenyképességi hányadosa* B *minden* $k \geq 2$ *esetén.*

Kifejlesztettünk egy további algoritmust is. A COLOR&SIZE algoritmus alapötlete az, hogy online módon csoportosítja a tárgyakat

C típusú (színező) és S típusú (töltésnövelő) tárgyakra. Az egyes csoportokat egy FF algoritmus szerint pakoljuk, de egymástól függetlenül csak a színeket illetve csak a ládák töltését figyelembe véve. Tehát egy S típusú tárgyat az első olyan ládába rakunk, amelyben kevesebb, mint B darab S típusú tárgy van, ha nincs ilyen akkor új ládát nyitunk. Egy C típusú tárgyat pedig az első olyan ládába rakunk, ahol még nincs az adott színből C típusú tárgy és ahol legfeljebb $k-1$ különböző színből választottunk még C típusú tárgyat. Ha nincs ilyen láda, akkor új ládát nyitunk.

Az algoritmus pontos specifikációjához meg kell még adnunk, hogy milyen szabály szerint csoportosítjuk a tárgyakat. Az algoritmus egy p egész paramétert használ a csoportosítás során. Tegyük fel, hogy a j-edik tárgy érkezik, és ennek a tárgynak a színe az i-edik fajta szín, ami megjelent a sorozatban. Ekkor ha $i \mod p \neq 0$ és $j \mod p \neq 0$, akkor ez a tárgy C típusú lesz, továbbbá ha $i \mod p = 0$ és $j \mod p \neq 1$ akkor is C típusú lesz a tárgy. A további esetekben pedig S típusú.

Az algoritmus versenyképességét adja meg az alábbi állítás.

14. Tétel. *([17]) A* Color&Size *algoritmus aszimptotikusan $O(k)$-versenyképes megfelelő p paraméter választása esetén (az additív konstans is $O(k)$).*

Szintén igazoltunk egy alsó korlátot a lehetséges versenyképességi hányadosokra. Ezt adja meg az alábbi tétel.

15. Tétel. *([17]) Minden online algoritmusra teljesül, hogy az aszimptotikus versenyképességi hányadosa legalább $1 + H_{k-1} = \Omega(\log k)$, ahol $H_k = \sum_{i=1}^{k} 1/i$.*

3.3. Elemszámkorlátos ládafedés

A [18] cikkben az elemszámkorlátos ládafedési problémát vizsgáltuk. Ebben a modellben a tárgyak mérete mellett adott egy k szám is, amit elemszámkorlátnak nevezünk. Egy ládát akkor tekintünk lefedettnek, ha a benne levő tárgyak méreteinek összege legalább 1, és legalább k tárgyat helyeztünk el a ládába. A probléma speciális esete a fentiekben tárgyalt színkorlátos ládafedési modellnek, ha minden tárgy színe különböző, akkor a színkorlátos modell az elemszámkorlátos modellre redukálódik. A probléma szintén speciális esete a

vektorfedési feladatnak ([1]), ahol d-dimenziós vektorokat kell csoportosítanunk maximális számú csoportba úgy, hogy minden csoportban az oda rendelt vektorokra minden koordinátában legalább 1 legyen az összeg. Ha az egyik koordinátában a tárgy méretét adjuk meg, a másikban pedig $1/k$-t, akkor ez a két-dimenziós vektorfedési feladat az elemszámkorlátos ládafedésre redukálódik. Így a kétdimenziós vektorfedésből automatikusan adódik egy 4-versenyképes algoritmus, ezért csak annál kisebb versenyképességű algoritmusok az érdekesek.

A következő CLASSIFY algoritmust fejlesztettük ki a probléma megoldására. Ez az algoritmus egy α paramétert használ, melyre $\frac{1}{2} \leq \alpha < 1$. A $(0, 1]$ intervallumot méret szerinti intervallumokra osztjuk, ahol az intervallumok $I_i = (\alpha^{i+1}, \alpha^i]$ alakúak. Azt mondjuk egy j tárgy a C_i osztályba tartozik, ha a mérete az I_i intervallumba esik. Az algoritmus használ két, k-tól függő $0 < q_1 < q_2$ pozitív egész paramétert.

Minden tárgy vagy A vagy B típusú. Rendre jelölje n_A^i és n_B^i az eddig megérkezett A és B típusú tárgyak számát a C_i osztályból. Ha az aktuális tárgy esetén $(n_A^i + n_B^i \bmod q_2) < q_2 - q_1$, akkor a tárgy típusa B lesz, egyébként (amennyiben $(n_A^i + n_B^i \bmod q_2) \geq q_2 - q_1$) a tárgy típusa A. Az algoritmus úgy pakolja a tárgyakat, hogy minden lefedett láda pontosan $(k-2)$ darab B típusú tárgyat tartalmaz és legalább 2 darab A típusú tárgyat, amelyek méreteinek összege nagyobb, mint 1.

Egy olyan ládát, ami már tartalmaz $(k-2)$ darab B típusú tárgyat, B szerint telinek nevezünk. Egy olyan ládát, amelyben az A típusú tárgyak mérete szigorúan nagyobb, mint 1 pedig A szerint telinek nevezünk. Azért használunk szigorúan nagyobb feltételt, mert ez garantálja, hogy legalább két darab A típusú tárgyat fog tartalmazni a láda. Nyilván, ha egy láda A és B szerint is teli, akkor az már fedett. Az algoritmus a FF eljárást használja az egyes típusokra egymástól függetlenül. Tehát egy A típusú tárgy a legelső olyan ládába kerül, ami A szerint nincsen teli, egy B típusú tárgy a legelső olyan ládába, ami B szerint nincsen teli.

Az algoritmus versenyképességére az alábbi állítást igazoltuk

16. Tétel. *([18]) Minden $\varepsilon > 0$ esetén létezik olyan α érték, hogy a $q_1 = 2k$ és $q_2 = 3k-2$ értékválasztás mellett a* CLASSIFY *algoritmus aszimptotikus versenyképessége legfeljebb $\frac{3k-2+\varepsilon}{k}$.*

A dolgozatban szintén igazoltuk az alábbi alsó korlátot a lehetséges versenyképességi hányadosra.

17. Tétel. *([18]) Minden $k \geq 4$ esetén teljesül, hogy nincs olyan online algoritmus az elemszámkorlátos ládapakolási feladatra, amelynek az aszimptotikus versenyképességi hányadosa kisebb, mint $\frac{5k-4}{2k}$.*

Megjegyezzük, hogy a disszertációban (és a [18] cikkben is) vizsgáltuk a probléma félig-online esetét is, ahol feltételeztük, hogy a tárgyak méret szerint monoton nemnövekvő sorrendben érkeznek. Ebben az esetben sikerült egy a CLASSIFY algoritmushoz hasonló elven működő aszimptotikusan 2-versenyképes algoritmust kifejleszteni és igazolni azt is, hogy ennél kisebb versenyképességi hányadossal nem rendelkezik algoritmus.

3.4. Online ládafedés a használt tárgyak összsúlyát minimalizálva

A [7] cikkben a ládafedés alábbi változatát vizsgáltuk. Adott tárgyak egy online listája, és adott m darab egység méretű láda. Feladatunk ezen ládák lefedése a tárgyak listájának minimális összsúlyú kezdőszeletével. A feladat online, azaz a tárgyak egyenként érkeznek és az adott tárgyat a további tárgyakra vonatkozó ismeretek nélkül kell egy ládába helyeznünk. Az eljárás akkor ér véget ha mind az m ládát lefedtük. Mivel a tárgyak mérete legfeljebb 1, ezért ha egy algoritmus egy lefedett ládába nem rak már további tárgyakat, akkor az az eljárás végén minden ládához legfeljebb 2 mennyiségű tárgyat rendel. Ebből adódik, hogy minden ilyen algoritmus 2-versenyképes, így ennél kisebb versenyképességgel rendelkező eljárást kerestünk. Fontos kiemelnünk, hogy az offline algoritmusnak is a tárgyak listájának egy kezdőszeletét kell használnia, az offline algoritmus sem rendezheti át a tárgyak sorrendjét.

Az általunk felvetett probléma duálisának tekinthető ládapakolási problémát vizsgálták a [3] dolgozatban. Ott a cél a ládákba elhelyezhető tárgyak számának maximalizálása volt. Vizsgálták azt az általunk tekintett modellt, ahol a tárgyak listájának maximális kezdőszeletét kellett elpakolni, és azt is, ahol a tárgyak visszautasíthatóak voltak. A [3] cikkben elért eredmények általános méretű ládákra vonatkozó kiterjesztését a [15] dolgozatban mutatták be.

Külön vizsgáltuk az $m = 2$ esetet. Ebben az esetben az aszimptotikus versenyképességnek nincs értelme, hiszen a két láda lefedéséhez mindig elegendő konstans, legfeljebb 4 összméretű tárgy. Ebben az $m = 2$ esetben a parametrikus változatot is vizsgáltuk, azaz azt az esetet, ha ki van kötve, hogy a tárgyak mérete legfeljebb $\frac{1}{p}$ lehet valamely p egészre. A $p = 1$ eset felel meg az általános problémának. Az általános gépszám esetén már az aszimptotikus hányadost használtuk, hiszen ekkor egy m-től független additív konstans nem oldja meg a feladatot, de segít az algoritmus kezdeti lépéseiben.

3.4.1. Az $m = 2$ eset

Két láda esetére a következő TwoBins algoritmust dolgoztuk ki. Az algoritmus leírásához jelölje $A_1 \geq A_2$ a ládákban levő töltést (feltesszük, hogy $A_2 < 1$, mivel ellenkező esetben már le lennének feddve a ládák). Az új j tárgyat a következő szabályok szerint helyezzük el. Ha a tárgy s_j méretére $1 \leq A_2 + s_j \leq \frac{2p+2}{2p+1}$ teljesül, akkor j-t az A_2 töltésű ládába helyezzük el. Egyébként ha $A_1 < 1$ és $A_1 + s_j \leq \frac{2p+2}{2p+1}$, akkor j-t az A_1 töltésű ládába helyezzük el. Végül, ha egyik feltétel sem teljesül, akkor j-t az A_2 töltésű ládába helyezzük el.

Az algoritmus versenyképességére vonatkozik az alábbi állítás.

18. Tétel. *[7] A* TwoBins *algoritmus a $p \geq 1$ parametrikus probléma esetén $\frac{(4p+1)(p+1)}{2p(2p+1)}$ versenyképes. Ez a legjobb elérhető versenyképesség, nincs olyan online algoritmus, amelynek a versenyképessége kisebb, mint ez az érték.*

Megvizsgáltuk az ütemezés területéről átvett Lista algoritmus viselkedését is, amely a tárgyat azon ládába rakja, ahol a legkisebb a töltés. Ha több ilyen láda is van, akkor a legelső ilyen ládát használja. Az algoritmus azon speciális félig-online esetekben igazán hatékony, amikor a tárgyakat méret szerint monoton sorrendben kapjuk. Ezt mutatják az alábbi eredmények.

19. Tétel. *[7] Ha a parametrikus esetben a tárgyak méret szerint monoton nemcsökkenő sorrendben érkeznek, akkor a* Lista *algoritmus versenyképessége $\frac{2p+1}{2p}$. Ebben a speciális esetben ez a legjobb*

elérhető versenyképesség, nincs olyan félig-online algoritmus, amelynek a versenyképessége kisebb, mint ez az érték.

20. Tétel. *[7] Ha $p = 1$ és a tárgyak méret szerint monoton nemnövekvő sorrendben érkeznek, akkor a* Lista *algoritmus versenyképessége $\frac{6}{5}$. Ebben a speciális esetben is ez a legjobb elérhető versenyképesség.*

A $p > 1$ esetben nemnövekvő sorozatokra a Lista algoritmus nem éri el a lehető legkisebb versenyképességet. Erre az esetre a TwoBinsDecreasing (TBD) algoritmust fejlesztettük ki. Az első p darab tárgyat az első ládához, a második p darab tárgyat a második ládához rendeljük. A további tárgyakat mindig ahhoz a ládához rendeljük, ahol kisebb a töltés. Az algoritmus versenyképességét határozza meg az alábbi állítás.

21. Tétel. *[7] A* TBD *algoritmus $\frac{2p+3}{2p+2}$-versenyképes monoton nemnövekvő tárgysorozatok esetén minden $p \geq 2$-re. Minden $p \geq 2$ esetén ez a legkisebb versenyképesség, ami monoton nemnövekvő sorozatokra elérhető.*

3.4.2. Általános ládaszám

Az általános esetben csak a félig-online monoton sorozatok esetét vizsgáljuk. Az, hogy van -e 2-nél kisebb aszimptotikus versenyképességgel rendelkező algoritmus az általános esetben továbbra is nyílt kérdés.

Arra az esetre, ahol a tárgyak monoton nemcsökkenő sorrendben érkeznek a PackIncreasing (PI) algoritmust definiáltuk. Az algoritmus két ($\frac{3}{4} \leq \alpha \leq \frac{5}{6}$ és $\frac{1}{8} \leq \beta \leq \frac{1}{4}$) paramétert használ, amelyeket a versenyképességi elemzés során optimalizáltunk. A tárgyakat méret szerint osztályokra bontjuk. A legfeljebb $\frac{1}{2}$ méretű tárgyakat kicsinek, azon tárgyakat, amelyek mérete a $(\frac{1}{2}, \alpha]$ intervallumba esik közepesnek, az α-nál nagyobb méretű tárgyakat pedig nagynak hívjuk. Mivel monoton nemcsökkenő sorrendben érkeznek a tárgyak, ezért tudjuk, hogy elsőként a kicsi, majd a közepes, végül a nagy tárgyak fognak megérkezni. A másik β paraméter szerepe az, hogy az első $\lfloor \beta m \rfloor$ ládát foglaltnak nevezzük (tulajdonképpen ezek a ládák arra várnak, hogy a később esetlegesen megjelenő nagy tárgyakat jól

tudjuk pakolni). A többi ládát szabadnak hívjuk. A PI algoritmus a következőképpen működik.

Amíg kis tárgyak érkeznek és van legalább egy foglalt láda, amiben a töltés kisebb, mint $1-\alpha$, addig pakoljuk a tárgyakat a LISTA algoritmus alapján a foglalt ládákba. Ha ezt azért hagyjuk abba, mert elfogynak a kicsi tárgyak és az első közepes tárgy megérkezik, akkor a NF algoritmust használjuk a szabad ládákra, ami mindig az aktuális ládába rakja a tárgyakat, amíg azt le nem fedte és ezt követően tér át a következő ládára. A szabad ládák lefedése után az LISTA algoritmust használjuk a foglalt ládákra, amíg azokat is lefedjük. Ha az algoritmus első részét nem egy közepes tárgy miatt, hanem azért hagyjuk abba, mert minden foglalt láda töltése elérte az $1-\alpha$ korlátot, akkor a pakolást az NF eljárással folytatjuk elsőként a szabad ládákat, majd a foglalt ládákat pakolva.

Az eljárás versenyképességére vonatkozik az alábbi állítás.

22. Tétel. *[7] Az α és β paraméterek megfelelő választása mellett a* PI *algoritmus aszimptotikus versenyképessége legfeljebb* 1.931215 *monoton nemcsökkenő méretű sorozatok esetén.*

Abban az esetben, ahol a tárgyak sorozata méret szerint monoton nemnövekvő sorrendben érkezik hatékonyabb algoritmust tudtunk fejleszteni. Ennek oka az, hogy a fő nehézségeket valójában a nagyobb méretű tárgyak okozhatják, így előnyös ha azokat az algoritmus már akkor látja, amikor sok döntési lehetősége van. A kifejlesztett eljárás ismertetéséhez jelölje h azon tárgyak számát, amelyek mérete a $\left(\frac{2}{3},1\right]$ intervallumban, ℓ azon tárgyak számát, amelyek mérete a $\left[\frac{1}{2},\frac{2}{3}\right]$ intervallumban van. Fontos megjegyezni, hogy ezeket a számokat nem ismerjük az eljárás kezdetekor, ezeket online tudjuk meg, ahogy a tárgyak érkeznek.

A PD algoritmus ezen értékektől függően pakolja a beérkező tárgyakat. Feltesszük, hogy nincs olyan tárgy, amelynek a mérete 1. Ez nem jelent megszorítást, hiszen az ilyen tárgyak önmagukban lefednek egy ládát és mivel elsőként érkeznének, ezért ezt az online algoritmus is meg tudja tenni velük. Ebből adódóan ilyen tárgyak jelenléte csak javítaná a versenyképességet. Az első $\min\{h,m\}$ tárgyak mindegyikét különböző ládákba (az első $\min\{h,m\}$ ládába) pakoljuk. Ezt követően az alábbi 3 esetet különböztetjük meg:

1. eset $h \geq m$. Ekkor a következő m tárgyat is egymástól különböző ládákba helyezzük, úgy hogy az első $2m$ tárgy után az

i-edik láda az i-edik és a $2m+1-i$-edik tárgyakat tartalmazza. Ha maradt még olyan láda, amit nem fedtünk le, használjuk a NF algoritmust ezek lefedéséhez.
2. eset: $h < m$, $2(m-h) \leq \ell$. Ekkor pakoljuk a következő $2(m-h)$ tárgyat kettesével rendre a $h+1, \ldots, m$ indexű ládákba, majd a következő h tárgyat rakjuk az első h ládába, az ott található 2/3-nál nagyobb tárgyak mellé. Végül, ha maradt még olyan láda, amit nem fedtünk le, használjuk a NF algoritmust ezek lefedéséhez.
3. eset: $h < m$, $2(m-h) > \ell$. Ebben az esetben legyen $h' = \lfloor \frac{\ell}{2} \rfloor$. A következő $2h'$ tárgyat pakoljuk páronként a $h+1, \ldots, h+h'$ indexű ládákba, azaz a következő még üres ládákba, amik nem tartalmaznak 2/3-nál nagyobb tárgyat. Megjegyezzük, hogy ezeket a ládákat az algoritmus a párokkal le is fedi. Ha ℓ páratlan, akkor az utolsó ilyen tárgyat a $h+h'+1 \leq m$ indexű ládába rakjuk. Ezt követően a további $2(m-h)-\ell$ tárgyat úgy pakoljuk el, hogy az első nagy tárgyakat tartalmazó h darab ládába ne kerüljön belőlük és a többi láda mindegyikében pontosan két tárgy legyen. Majd a következő $m-h'$ tárgyból rakjunk egyet-egyet a még nem lefedett ládák mindegyikébe az első h ládával kezdve. Végül, ha maradt még lefedetlen láda, használjuk a NF algoritmust ezek lefedésére.

Fontos megjegyezni, hogy a 2. és 3. eset ugyanúgy pakolja a tárgyakat, amíg ℓ értékét meg nem tudjuk, így valóban online végrehajtható az eljárás. Az algoritmus versenyképességére vonatkozik az alábbi állítás.

23. Tétel. *[7] A* PD *algoritmus aszimptotikusan* $\frac{4}{3}$*-versenyképes, ha a tárgyak sorozata méret szerint monoton nemnövekvő.*

Az $m = 2$ esettel ellentétben nem sikerült a lehető legkisebb aszimptotikus versenyképességgel rendelkező algoritmusokat megtalálni. Az alábbi alsó korlátot igazoltuk a lehetséges versenyképességi hányadosra.

24. Tétel. *[7] Nincs olyan félig-online algoritmus, amely aszimptotikus versenyképessége kisebb, mint* 1.302017 *monoton nemcsökkenő sorozatok esetén. Nincs olyan félig-online algoritmus, amely aszimptotikus versenyképessége kisebb, mint* $\frac{10}{9} \approx 1.111$ *monoton nemnövekvő sorozatok esetén.*

Megemlítjük, hogy a disszertációban a félig-online esetekben az

alsó korlátokat a parametrikus esetekre igazoltuk, de itt csak a $p = 1$ esetre vonatkozó eredményt emeltük ki.

4. Online klaszterezés

Az online klaszterezési problémákban egy metrikus tér pontjait szeretnénk online csoportosítani. Ez azt jelenti, hogy egyenként érkeznek kérések a tér pontjaiba és minden kérést egy klaszterhez kell rendelnünk az érkezésekor a további kérésekre vonatkozó információk nélkül. Egy pontot vagy egy meglevő klaszterhez rendelhetünk vagy új klasztert definiálunk hozzá. A célunk a klaszterezés teljes költségének minimalizálása, amely költség az egyes klaszterek költségeinek összege. A klaszterek költsége függhet a klaszterhez rendelt pontoktól illetve a klaszter középpontjától.

Az egyik széles körben vizsgált online klaszterezési probléma az, amelyben egységnyi méretű klasztereket kell létrehozni, azaz a célunk az, hogy minimális számú egységgömbbel fedjük le a pontokat. Ezt a problémát két különböző online modellben vizsgálták attól függően, hogy milyen mértékben kell rögzítenie egy létrehozott gömbnek a pozícióját az algoritmusnak. A szigorú modellben a gömb létrehozásakor rögzíteni kell annak a helyét is és a gömb nem mozgatható később. Ezt modellt a [6] cikkben definiálták, ahol egy $O(2^d d \log d)$-versenyképes algoritmust adtak meg d-dimenziós térre és egy $\Omega(\log d / \log\log\log d)$ alsó korlátot. Egy és két dimenzióban éles korlátokat bizonyítottak, az optimális online algoritmusok 2 illetve 4-versenyképesek. A laza modellben a körök mozgathatóak azon feltétel mellett, hogy az eddig lefedett kéréseket továbbra is lefedik. Ezzel a kérdéssel már egy dimenzióban is több cikk foglalkozott, egyre jobb algoritmusokat definiáltak és az alsó korlát is növekedett az első publikációt követően. A jelenlegi legjobb algoritmus, amit a [13] cikkben publikáltak, 5/3-versenyképes. A legjobb alsó korlátot, ami azt mondja ki, hogy nincs jobb algoritmus, mint 1.625-versenyképes a [28] cikkben publikálták.

Egy másik vizsgált online klaszterezési feladat az online kiszolgáló elhelyezési probléma. A kiszolgáló elhelyezési problémában adott egy metrikus tér kérések egy multihalmazával (minden kérés a tér egy pontja). A cél kiszolgálóknak olyan elhelyezése a térben, amely elhelyezésre minimális a kiszolgálók létrehozási költségének és a ki-

szolgálás költségének az összege. Minden kérésre a kiszolgálás költsége megegyezik a legközelebbi kiszolgálótól vett távolsággal. Az online változatban a kérések egyenként jelennek meg és minden kérés érkezése esetén van lehetősége az algoritmusnak új kiszolgálót, vagy kiszolgálókat nyitnia. A feladatot a [29] cikkben definiálták, ahol igazolták, hogy nincs konstans versenyképes algoritmus a feladat megoldására és megadtak egy véletlenített $O(\log n)$-versenyképes algoritmust. Később a [19] cikkben oldották meg teljesen a problémát, ahol egy $O(\log n/\log\log n)$-versenyképes algoritmust adtak meg, és igazolták, hogy nincs olyan algoritmus, amely versenyképessége kisebb, mint $\Omega(\log n/\log\log n)$.

Mi az egységgömbökkel való klaszterezés egy olyan kiterjesztését vizsgáltuk, ahol a lefedő klaszterek mérete nem feltétlenül egyenlő, hanem a klaszter mérete is része a célfüggvénynek. A problémát egy dimenzióban vizsgáltuk az eredményeink a [8] cikkben kerültek publikálásra. A modellben keletkezett klaszterezés költsége a létrehozott klaszterek költségeinek összege, és egy klaszter költsége egy egységnyi setup költség plusz a klaszter átmérőjének a hossza. Attól függően, hogy milyen mértékben kell specifikálnia az algoritmusnak egy klasztert annak létrehozásakor két különböző változatot vizsgáltunk. A szigorú modellben a létrehozott klaszternek meg kell adni a méretét és rögzíteni az elhelyezkedését. A laza modellben a klaszter mérete növelhető, és a klaszter is mozgatható azon feltétel mellett, hogy az addig lefedett pontokat továbbra is lefedi. Megjegyezzük, hogy a [8] cikkben egy átmeneti modellt is elemeztünk, ahol a klaszter létrehozásakor rögzíteni kell annak méretét, de a klaszter mozgatható azon feltétel mellett, hogy az addig lefedett pontokat továbbra is lefedi, de ezeket az eredményeket nem mutattuk be a disszertációban. Érdemes megjegyezni, hogy míg a szigorú modellben egy klaszter költsége eldől annak létrehozásakor, addig a laza modellben ez a költség az algoritmus futása közben növekedhet.

A laza modellben az ECC Extend closed clusters algoritmust vizsgáltuk. Az algoritmus ismertetéséhez jelölje p az érkező pont helyét. Ha az algoritmusnak van olyan intervalluma (klasztere), ami tartalmazza p-t, akkor rendeljük p-t ehhez az intervallumhoz. Egyébként pedig legyen q a p-hez legközelebbi az eddig megérkezett pontok közül. Ha q és p távolsága legfeljebb $\phi = (1+\sqrt{5})/2$, akkor rendeljük p-t a q-t tartalmazó klaszterhez, és nyújtsuk meg az klasztert leíró intervallumot p-ig. Egyébként pedig nyissunk egy új

klasztert, egy intervallumot, ami csak a p pontot tartalmazza.

Sikerült meghatároznunk az algoritmus versenyképességi hányadosát, és azt is igazoltuk, hogy az algoritmus a lehető legjobb a versenyképesség szempontjából.

25. Tétel. *[8] Az ECC algoritmus* $(1+\sqrt{5})/2$*-versenyképes. Nincs olyan algoritmus a laza modellben, amely versenyképessége kisebb, mint* $(1+\sqrt{5})/2$.

Vizsgáltuk a probléma félig-online változatát is, ahol feltételezzük, hogy a pontok monoton növekvő sorrendben érkeznek. Ebben a speciális esetben megadható az optimális offline megoldás online módon is, akkor kell új klaszter nyitni, ha az új pontnak és az utoljára nyitott klaszter szélének a távolsága nagyobb, mint 1.

A szigorú modellben az alábbi CENTER algoritmust vizsgáltuk. Az algoritmus megadásához jelölje az új pont helyét p. Ha az algoritmusnak van olyan intervalluma (klasztere), ami tartalmazza p-t, akkor rendeljük p-t ehhez az intervallumhoz. Ellenkező esetben legyen ℓ a legközelebbi balra eső fedett pont (ha nincs ilyen $\ell = -\infty$), és legyen r a legközelebbi jobbra eső fedett pont (ha nincs ilyen $r = \infty$) és legyen $\alpha = \frac{\sqrt{2}}{2}$. Nyissuk meg p-hez a $[\max\{\ell, p-\alpha\}, \min\{p+\alpha, r\}]$ klasztert. Tehát az alapötlet az, hogy egy p közepű $\sqrt{2}$ méretű klasztert nyitunk. Másrészt, ha ez a klaszter belemetszene a szomszédos, már meglevő klaszterekbe akkor a méretét csökkentjük. Sikerült meghatároznunk az algoritmus versenyképességi hányadosát, és azt is igazoltuk, hogy az algoritmus a lehető legjobb algoritmus versenyképesség szempontjából.

26. Tétel. *[8] Az* CENTER *algoritmus* $(1+\sqrt{2})$*-versenyképes. Nincs olyan algoritmus a szigorú modellben, amely versenyképessége kisebb, mint* $1+\sqrt{2}$.

Ebben a modellben is vizsgáltuk azt a félig online esetet, ha a pontok monoton növekvő sorrendben érkeznek. Itt egy 2-versenyképes algoritmust adtunk meg és igazoltuk, hogy nincs olyan algoritmus, amelynek kisebb a versenyképessége.

Később az általunk bevezetett modell többdimenziós kiterjesztéseivel kapcsolatos eredmények is publikálásra kerültek. Azt a modellt, ahol a klaszter költsége egy konstans setup költségnek és a klaszter átmérőjének az összege, a [20] cikk vizsgálta, ahol igazolást

nyert, hogy magasabb dimenziókban nincs konstans versenyképes algoritmus. A két-dimenziós modellt, ahol a költség az átmérő négyzetétől függ a [11] cikkben vizsgáltuk. Igazoltuk, hogy ezen függvény mellett van konstans versenyképes eljárás.

Hivatkozások

[1] N. Alon, Y. Azar, J. Csirik, L. Epstein, S.V. Sevastianov, A.P.A. Vestjens, G.J. Woeginger, On-line and off-line approximation algorithms for vector covering problems, *Algorithmica*, **21**, 104–118, 1998.

[2] S.F. Assmann, D.S. Johnson, D.J. Kleitman, J.Y.T. Leung, On a dual version of the one-dimensional bin packing problem, *Journal of Algorithms*, **5(4)**, 502–525, 1984.

[3] Y. Azar, J. Boyar, L. Epstein, L.M. Favrholdt, K.S. Larsen, M.N. Nielsen, Fair versus unrestricted bin packing, *Algorithmica*, **34**, 181–196, 2002.

[4] L. Babel, B. Chen, H. Kellerer, V. Kotov, Algorithms for on-line bin-packing problems with cardinality constraints, *Discrete Applied Mathematics*, **143(1-3)** 238–251, 2004.

[5] J. Balogh, J. Békési, G. Galambos, New lower bounds for certain classes of bin packing algorithms, *Theoretical Computer Science*, **440–441**, 1–13, 2012

[6] M. Charikar, C. Chekuri, T. Feder, R. Motwani, Incremental clustering and dynamic information retrieval, *SIAM Journal on Computing*, **33(6)**, 1417–1440, 2004.

[7] J. Csirik, L. Epstein, Cs. Imreh, A. Levin, On the sum minimization version of the online bin covering problem, *Discrete Applied Mathematics*, **158(13)** 1381–1393, 2010.

[8] J. Csirik, L. Epstein, Cs. Imreh, A. Levin, Online Clustering with Variable Sized Clusters, *Algorithmica*, **65(2)**, 251–274, 2013.

[9] J. Csirik, V. Totik. On-line algorithms for a dual version of bin packing. *Discrete Applied Mathematics*, **21**, 163–167, 1988.

[10] J. Csirik, G. Woeginger, Shelf algorithms for on-line strip packing, *Information Processing Letters*, **63**, 171–175, 1997.

[11] G. Divéki, Cs. Imreh, An online 2-dimensional clustering problem with variable sized clusters, *Optimization and Engineering*, **14(4)**, 575–593, 2013.

[12] Gy. Dósa, Cs. Imreh, The generalization of scheduling with machine cost, *Theoretical Computer Science*, **510**, 102–110, 2013.

[13] M.R. Ehmsen, K.S. Larsen, Better bounds on online unit clustering, In *Proc. of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT2010)*, 371–382, 2010.

[14] L. Epstein, Online bin packing with cardinality constraints, *SIAM Journal on Discrete Mathematics*, **20(4)**, 1015–1030, 2006.

[15] L. Epstein, L.M. Favrholdt, On-Line maximizing the number of items packed in variable-sized bins, *Acta Cybernetica*, **16**, 57–66, 2003.

[16] L. Epstein, Cs. Imreh, A. Levin, Class constrained bin packing revisited, *Theoretical Computer Science*, **411(34–36)**, 3073–3089, 2010

[17] L. Epstein, Cs. Imreh, A. Levin, Class Constrained Bin Covering, *Theory of Computing Systems*, **46(2)**, 246–260, 2010.

[18] L. Epstein, Cs. Imreh, A. Levin, Bin covering with cardinality constraints, *Discrete Applied Mathematics*, **161(13–14)**, 1975–1987, 2013.

[19] D. Fotakis, On the competitive ratio for online facility location, *Algorithmica*, **50(1)**, 1–57, 2008.

[20] D. Fotakis, P. Koutris, Online Sum-Radii Clustering, *Theoretical Computer Science*, **540** 27–39, 2014

[21] R.L. Graham, Bounds for certain multiprocessor anomalies, *Bell System Technical J.*, **45**, 1563–1581, 1966.

[22] R. Harren, W. Kern, Improved lower bound for online strip packing, in *Proceedings of the 9th International Workshop on Algorithms and Online Algorithms, LNCS 7164*, 211–218, 2011.

[23] J.L. Hurink, J.J. Paulus, Improved online algorithms for parallel job scheduling and strip packing, *Theoretical Computer Science*, **412(7)**, 583–593, 2011.

[24] Cs. Imreh, Online strip packing with modifiable boxes, *Operations Research Letters*, **29**, 79–86, 2001.

[25] Cs. Imreh, Scheduling problems on two sets of identical machines, *Computing*, **70**, 277–294, 2003.

[26] Cs. Imreh, On-line scheduling with general machine cost functions, *Discrete Applied Mathematics*, **157**, 2070–2077, 2009.

[27] Cs. Imreh, J. Noga, Scheduling with Machine Cost, In *Randomization Approximation and Combinatorial Optimization Algorithms and Techniques ed. D. Hochbaum and K. Jansen*, 1999, 168–176.

[28] J. Kawahara, K.M. Kobayashi, An improved lower bound for one-dimensional online unit clustering, *Theoretical Computer Science*, **600**, 171–173, 2015

[29] A. Meyerson, Online facility location, In *Proc. of the 42nd Annual Symposium on Foundations of Computer Science (FOCS2001)*, 426–431, 2001.

[30] S.S. Seiden, On the online bin packing problem, *Journal of the ACM*, **49(5)** 640–671, 2002.

[31] H. Shachnai, T. Tamir, Tight bounds for online class-constrained packing, *Theoretical Computer Science*, **321(1)**, 103–123, 2004.

[32] E.C. Xavier, F.K. Miyazawa. The class constrained bin packing problem with applications to video-on-demand, *In Proc. of the 12th Annual International Conference on Computing and Combinatorics (COCOON 2006)*, 439–448, 2006.

[33] D. Ye, X. Han, G. Zhang, A note on online strip packing, *Journal of Combinatorial Optimization*, **17(4)**, 417–423, 2009.