Online algoritmusok

Online problémáról beszélünk azokban az esetekben, ahol nem ismert az egész input, hanem az algoritmus az inputot részenként kapja meg, és a döntéseit a megkapott részletek alapján a további részekre vonatkozó információk nélkül kell meghoznia. Online algoritmusokat sok probléma esetén alkalmaznak az operációkutatás, az elméleti számítástudomány és a közgazdaságtan különböző területein.

Az algoritmusok hatékonyságát általában a versenyképességi analízis alapján mérik (a másik használt elemzési módszer az átlagos eset analízis).

Optimalizálási feladatok esetén minden I inputra $OPT(I)$ jelöli az optimális megoldás célfüggvényértékét és egy A algoritmusra $A(I)$ jelöli az algoritmus által az A inputon felvett célfüggvényértéket.

Minimalizálási feladatok esetén egy online A algoritmusra azt mondjuk C-versenyképes, ha minden I inputra teljesül $A(I) \leq C \cdot OPT(I)$. Gyakran a definícióban megengednek egy inputtól független D additiv konstanst, azaz az $A(I) \leq C \cdot OPT(I) + D$ egyenlőtlenségnek kell teljesülnie minden inputra. Bizonyos esetekben a C függhet az input méretétől. Ekkor az $A(I) \leq C(n) \cdot OPT(I)$ egyenlőtlenségnek kell teljesülnie minden n méretű inputra. Egy algoritmus versenyképességi hányadosának a legkisebb C számot nevezzük, amire C-versenyképes.

Ütemezés

A legegyszerűbb modellekben feladatunk az, hogy a munkákhoz, amelyeknek ismerjük a megmunkálási idejeit hozzárendeljük a gépet, amelyen a munkát végrehajtjuk és a megmunkálás kezdési és befejezési időpontját, amely időpontok különbsége a megmunkálási idő kell legyen.

Amennyiben a munka megmunkálási ideje minden gépen ugyanannyi, akkor azonos párhuzamos gépekről beszélünk.

Amennyiben a gépekhez hozzá van rendelve egy s_i sebesség, a munkáknak van egy p_j megmunkálási súlya és a j-edik munka megmunkálási ideje az i gépen p_j/s_i, akkor hasonló párhuzamos gépekről beszélünk.

Ha a j-edik munka megmunkálási ideje tetszőleges $P_j = (p_j(1), \ldots, p_j(m))$ nemnegatív vektor lehet, ahol a munka megmunkálási ideje az i-edik gépen $p_j(i)$, akkor független párhuzamos gépekről beszélünk.

Ütemezési problémák esetén számos ütemezési célfüggvényt szokás vizsgálni, mi itt csak azt a modellt vizsgáljuk, amelyben a cél a maximális befejezési idő minimalizálása.

Online ütemezés

Lista modell: Amikor egy munkát megkapunk a listáról, akkor ismerjük meg a szükséges megmunkálási időt, ezt követően ütemeznünk kell a munkát, hozzárendelve a kezdési és befejezési időt, amelyeket később már nem változtathatunk meg, és csak ezt követően kapjuk meg a listáról a következő munkát.

Ha a célfüggvény a maximális befejezési idő, akkor (miként az offline esetben is) elegendő olyan algoritmusokkal foglalkoznunk, amelyekben az egyes gépeken a munkák szünet nélkül követik egymást. Ebben az esetben minden gépre a maximális befejezési idő megegyezik a géphez rendelt megmunkálási idők összegével. Minden gépre a gépen levő megmunkálási idők összegét a gépen levő *töltésnek* hívjuk.

Idő modell: Itt a munkáknak egy r_j *érkezési ideje* is van, a munkáról semmit sem tudunk a érkezési ideje előtt (még a létezését sem), de a munka érkezése után bármikor megkezdhetjük a munka végrehajtását, ha van olyan gépünk, amely nem dolgozik. Amennyiben egy munkát elkezdünk végrehajtani, akkor nem szakíthatjuk félbe.

Lista algoritmus

Előkészítő rész. A j_1 munkát rendeljük az M_1 géphez, továbbá legyen $r := 1$.

Iterációs rész (r-edik iteráció). Ha $r = n$, akkor vége az eljárásnak. Ellenkező esetben a j_{r+1} munkát rendeljük ahhoz a géphez, amelyre a gépen levő töltés minimális. Ha több ilyen gép is van, akkor válasszuk ezek közül a legkisebb indexűt. Növeljük r értékét 1-gyel, és folytassuk az eljárást a következő iterációval.

Tétel Egyforma párhuzamos gépek esetén a LISTA algoritmus versenyképességi hányadosa $2-1/m$, ahol m a gépek száma.

Élesség: Vegyük a következő sorozatot: m(m-1) munka 1 mérettel és 1 munka m mérettel.
Ekkor $LISTA(\sigma) = 2m-1$ és $OPT(\sigma) = m$, így a hányadosuk $2-1/m$.

Bizonyítás

Legyen j_l az a munka, amely a legkésőbb fejeződik be. Vizsgáljuk ezen munka S_l kezdési idejét. Mivel egyetlen gép sem kezdte el ezt a munkát ütemezni S_l előtt, ezért minden gép szünet nélkül dolgozott az S_l időpontig. Ebből azt kapjuk, hogy

$$S_l \leq \frac{1}{m}\sum_{\substack{j=1\\ j\neq l}}^{n} p_j = \frac{1}{m}\Big(\sum_{j=1}^{n} p_j - p_l\Big) = \frac{1}{m}\Big(\sum_{j=1}^{n} p_j\Big) - \frac{1}{m}p_l.$$

Következésképp

$$LISTA(\sigma) = S_l + p_l \leq \frac{1}{m}\Big(\sum_{j=1}^{n} p_j\Big) + \frac{m-1}{m}p_l.$$

Másrészt az optimális ütemezésben is végre kell hajtani az összes munkát, így $OPT(\sigma) \geq \frac{1}{m}(\sum_{j=1}^{n} p_j)$. Továbbá a p_l munkát is végre kell hajtani valamely gépen, így $OPT(\sigma) \geq p_l$. Ezen becslések alapján egyből adódik, hogy

$$LISTA(\sigma) \leq \Big(1+\frac{m-1}{m}\Big)OPT(\sigma),$$

amivel bizonyítottuk, hogy LISTA $2-1/m$ versenyképes algoritmus.

Mohó algoritmus független gépekre

Bonyolultabb gépmodellek esetén, nem elég a töltést figyelembe venni, ekkor az alábbi algoritmust használhatjuk.

Mohó algoritmus: ütemezzük a munkát azon a gépen, ahol a töltés a munka ütemezése után minimális lesz. Ha több ilyen gép is van, akkor ezek közül azt választjuk, ahol a munka megmunkálási ideje a legkisebb és ha ilyen gépből is több van, akkor ezek közül a legkisebb indexű gépet választjuk.

Tétel: A mohó algoritmus versenyképességi hányadosa m, független gépek esetén.

Biz: Az első munkára a megmunkálási idő az első gépen 1, az m-edik gépen $1+\varepsilon$, a többi gépen ∞, (vagyis $p_1(1) = 1, p_1(i) = \infty, i = 2,\ldots,m-1, p_1(m) = 1+\varepsilon$), ezt követően a j-edik munkára a megmunkálási idő j a j-edik gépen, $1+\varepsilon$ a $j-1$-edik gépen, és ∞ a többi gépen (vagyis $p_j(j-1) = 1+\varepsilon$, $p_j(j) = j$, $p_j(i) = \infty$, ha $i \neq j-1$ és $i \neq j$).

Ezen munkasorozatra, a MOHÓ algoritmus a j-edik munkát a j-edik gépen ütemezi, és a maximális befejezési idő m. Másrészt az optimális offline algoritmus az első munkát az m-edik gépen, utána a j-edik munkát a $(j-1)$-edik gépen ütemezi, így az optimális maximális befejezési idő $1+\varepsilon$. A hányados $m/(1+\varepsilon)$. Ez az érték m-hez tart, ha az ε érték 0-hoz tart, amivel az állítást igazoltuk.

Most megmutatjuk, hogy az algoritmus m-versenyképes. Legyen az optimális maximális befejezési idő L^*, továbbá legyen $L(k)$ az első k munka MOHÓ algoritmus általi ütemezésében a maximális befejezési idő!

Mivel az i-edik munka ütemezéséhez valamely gépen legalább $\min_j p_i(j)$ idő szükséges, és egyik gépen sem használhatunk L^*-nál több időt, ezért $mL^* \geq \sum_{i=1}^{n} \min_j p_i(j)$.

Most igazoljuk teljes indukcióval, hogy $L(k) \leq \sum_{i=1}^{k} \min_j p_i(j)$. Mivel az első munkát ahhoz a géphez rendeljük, ahol a leghamarabb végrehajtható, ezért az állítás $k = 1$-re igaz. Most legyen $1 \leq k < n$ és tegyük fel, hogy az állítás k-ra igaz.

Tekintsük a $k+1$-edik munkát. Legyen az l-edik gép, amelyre a munka megmunkálási ideje minimális. Ezen a gépen végrehajtva a munkát a megmunkálási idő legfeljebb $L(k) + p_{k+1} \leq \sum_{i=1}^{k+1} \min_j p_i(j)$ (az indukciós feltevés alapján).

Mivel a MOHÓ algoritmus által kapott maximális befejezési idő legfeljebb annyi, mint abban az esetben, ha a $k+1$-edik munkát az l-edik géphez rendeljük, ezért $L(k+1) \leq \sum_{i=1}^{k+1} \min_j p_i(j)$, azaz az állítást igazoltuk $k+1$-re, így tetszőleges n-nél nem nagyobb egészre.

Következésképpen azt kapjuk, hogy $mL^* \geq \sum_{i=1}^{n} \min_j p_i(j) \geq L(n)$.

Tétel A MOHÓ algoritmus versenyképességi hányadosa $\Theta(\log m)$ hasonló párhuzamos gépek esetén.

Intervallum algoritmus az Idő modellben

INTV Algoritmus

amíg a munkasorozat nem ér véget

1. legyen H a t időpontig megérkezett és ütemezetlen munkák halmaza
2. legyen OFF ezen munkákra egy optimális offline ütemezés
3. rendeljük hozzá a munkákat a gépekhez a kapott ütemezésnek megfelelően
4. legyen q a kapott maximális befejezési idő
5. ha érkezett a $(t,q]$ intervallumban új munka vagy a munkasorozatnak vége van, legyen $t := q$
6. egyébként legyen t a következő munka érkezési ideje

Tétel: Az Idő modellben az INTV algoritmus 2-versenyképes.

Bizonyítás: Vegyük a munkáknak az algoritmus által kapott ütemezését. Az algoritmus tulajdonképpen fázisokban dolgozik, minden offline ütemezési megoldás végrehajtása egy fázisnak felel meg. Jelölje i ezen fázisok számát, vagyis azt ahányszor az optimális ütemezést megkerestük a már megérkezett de még nem ütemezett munkákra. Továbbá jelölje t_j a j-edik fázis kezdetét minden j-re és T az ütemezés befejezési idejét.

Legyen $T_3 = T - t_i$, $T_2 = t_i - t_{i-1}$, $T_1 = t_{i-1}$ és T_{OPT} az optimális offline költség. Ekkor $T_2 \leq T_{OPT}$. Másrészt $T_1 + T_3 \leq T_{OPT}$, mert az i-edik lépésben ütemezett munkák mindegyikének legalább $T_1 = t_{i-1}$ az érkezési ideje. Mivel az algoritmus által kapott ütemezés költsége $T_1 + T_2 + T_3$, ezért a tétel állítása következik.

Visszautasításos ütemezési modell

Ebben a modellben is m darab gép van, minden munkának van egy p_j megmunkálási ideje de a munkáknak van egy b_j büntetés értéke, amely a visszautasítás költségét adja meg. A munkákat vissza lehet utasítani, az elfogadott munkákat ütemezni kell, a minimalizálandó teljes költség a kapott ütemezés maximális befejezési idejének és a visszautasított munkák összbüntetésének az összege. P_0 azon munkák halmaza, amelyekre $b_j \leq p_j/m$.

RP_α Algoritmus:

- Ha a munkára $b_j \leq \alpha p_j$ teljesül, akkor utasítsuk vissza a munkát, egyébként ütemezzük a LISTA algoritmus szerint. Az algoritmusban α egy pozitív paraméter.

Tétel: Két gép esetén az $RP_{\varphi-1}$ algoritmus φ versenyképes.

Másrészt az algoritmus egyetlen α-ra sem konstans versenyképes ha a gépek száma tart a végtelenbe.

RTP_α Algoritmus

- Amennyiben a munka a P_0 halmaz eleme, akkor utasítsuk el.
- Egyébként legyen B a P_0 halmazon kívüli, visszautasított munkák összbüntetése. Ha $B + b_j \leq \alpha p_j$, akkor utasítsuk vissza a munkát, ellenkező esetben ütemezzük a LISTA algoritmus szerint.

Tétel: Az $\mathrm{RTP}_{\varphi-1}$ algoritmus $(1+\varphi)$-versenyképes.

Ládapakolási probléma

A ládapakolási problémában bemenetként tárgyak egy L sorozatát kapjuk meg, ahol az i-edik tárgyat a mérete határozza meg, ami egy $a_i \in (0,1]$ érték. Célunk a tárgyak elhelyezése a lehető legkevesebb, egység méretű ládába. Formálisabban megfogalmazva, a tárgyakat olyan csoportokba akarjuk osztani, hogy minden csoportra a benne levő tárgyakra a $\sum a_i \leq 1$ feltétel teljesüljön, és a csoportok száma legyen minimális.

A ládapakolási problémák elemzésére a versenyképességi hányados helyett az aszimptotikus versenyképességi hányadost vizsgálják. (A versenyképességi hányadost a legtöbb ládapakolási modellben nem vizsgálják, ha igen akkor abszolút versenyképességi hányadosnak hívják.) Egy A algoritmus aszimptotikus versenyképességi hányadosa (R_A^∞) a következő formulákkal definiálható:

$$R_A^n = \max\{A(L)/OPT(L) \mid OPT(L) = n\}$$

$$R_A^\infty = \limsup_{n\to\infty} R_A^n.$$

Az aszimptotikus hányados fő tulajdonsága az, hogy azt vizsgálja, miként viselkedik az algoritmus akkor, ha a bemenet mérete nő, pontosabban ha az optimális költség végtelenhez tart. Ez azt jelenti, hogy az algoritmus szabadon helyezheti el a kezdeti elemeket a listáról.

Az NF algoritmus

Amennyiben a tárgy elfér a nyitott ládában tegyük oda! Ellenkező esetben zárjuk be a nyitott ládát, nyissunk egy új ládát és tegyük abba a tárgyat!

Tétel Az NF algoritmus aszimptotikus versenyképességi hányadosa 2.

Biz: Jelölje n az NF algoritmus által használt ládák számát, továbbá legyen S_i, $i = 1,\ldots,n$ az i-edik ládában levő tárgyak méreteinek összege.

Ekkor $S_i + S_{i+1} \geq 1$, hiszen ellenkező esetben az $(i+1)$-edik láda első eleme elfért volna az i-edik ládában.

Másrészt az optimális offline algoritmus sem rakhat 1-nél több összméretű tárgyakat egy ládába, így $OPT(\sigma) \geq \lfloor n/2 \rfloor$.

$$\frac{NF(\sigma)}{OPT(\sigma)} \leq \frac{n}{\lfloor n/2 \rfloor} \leq 2 + 1/\lfloor n/2 \rfloor.$$

Másrészt ha n tart végtelenbe, akkor $1/\lfloor n/2 \rfloor$ tart a 0-hoz, amivel igazoltuk, hogy az algoritmus aszimptotikusan 2-versenyképes.

Vegyük a $4n$ tárgyból álló sorozatot: a $(2i-1)$-edik tárgy mérete $1/2$, a $2i$-edik tárgy mérete $1/2n$, ahol $i = 1,\ldots,2n$. Ekkor $NF(\sigma_n) = 2n$ és $OPT(\sigma_n) = n+1$.

Az FF algoritmus

FF algoritmus: A tárgyat a legkorábban kinyitott ládába tesszük ahol elfér. Ha nem fér el egyik ládában sem, kinyitunk egy új ládát és abba rakjuk.

BF algoritmus: A tárgyat a legnagyobb össztöltéssel rendelkező ládába tesszük ahol elfér. Ha nem fér el egyik ládában sem, kinyitunk egy új ládát és abba rakjuk.

Tétel Az FF algoritmus aszimptotikus versenyképességi hányadosa 1.7.

$$w(x) = \begin{cases} 6x/5 & 0 \le x \le 1/6 \\ 9x/5 - 1/10 & 1/6 \le x \le 1/3 \\ 6x/5 + 1/10 & 1/3 \le x \le 1/2 \\ 6x/5 + 2/5 & 1/2 < x. \end{cases}$$

A tárgyak egy tetszőleges H halmazára legyen $w(H) = \sum_{i \in H} w(a_i)$. Ekkor a súlyfüggvényre teljesülnek az alábbi állítások.

3.1. lemma. *Amennyiben tárgyak egy H halmazára teljesül, hogy $\sum_{i \in H} a_i \le 1$, akkor ezen tárgyakra $w(H) \le 17/10$.*

3.2. lemma. *Tárgyak tetszőleges L listájára $w(L) > FF(L) - 2$.*

Mivel az optimális offline algoritmus el tudja pakolni a lista elemeit $OPT(L)$ ládába úgy, hogy minden ládába a tárgyak méreteinek összege legfeljebb 1, ezért az első lemma alapján $w(L) \le \frac{17}{10} OPT(L)$. Másrészt a második lemma alapján $FF(L) - 2 \le w(L)$, így azt kapjuk, hogy $FF(L) \le \frac{17}{10} OPT(L) + 2$, amiből következik, hogy az algoritmus 1.7-versenyképes.

Megjegyzés További, kissé jobb versenyképességgel rendelkező nem Fit jellegű algoritmusok a Harmonikus algoritmusok. Ezek a tárgyakat méret szerint online csoportosítják és minden csoportit külön a hozzá rendelt ládákba pakolnak.

Egy egyszerű alsó korlát

Tétel: Nincs olyan online algoritmus a ládapakolási problémára, amelynek az aszimptotikus versenyképességi hányadosa kisebb, mint $4/3$.

Biz: Legyen A egy tetszőleges online algoritmus. Tekintsük tárgyaknak a következő sorozatát. Legyen $\varepsilon < 1/12$ és L_1 egy n darab $1/3+\varepsilon$ méretű tárgyból álló sorozat, L_2 pedig n darab $1/2+\varepsilon$ méretű tárgyból álló sorozat. Elsőként az algoritmus megkapja az L_1 listát.

Ekkor az algoritmus bizonyos ládákba két tárgyat tesz, bizonyos ládákba egyet. Jelölje k azon ládák számát, amelyek két tárgyat tartalmaznak.

Ekkor az algoritmus költsége $A(L_1) = k + (n - 2k) = n - k$. Másrészt az optimális offline algoritmus minden ládába két tárgyat tesz, így a költség $OPT(L_1) = n/2$.

Amennyiben ugyanez az algoritmus az L_1L_2 összetett listát kapja, akkor az első részben szintén k ládát használ két tárgynak. Következésképp $A(L_1L_2) \ge n - k + (n - (n - 2k)) = n + k$. Mászrészt az optimális offline algoritmus minden ládába egy kisebb, $1/3+\varepsilon$ méretű és egy nagyobb, $1/2+\varepsilon$ méretű tárgyat tesz, így $OPT(L_1L_2) = n$.

Tehát az A online algoritmusra van olyan L lista, amelyre

$$A(L)/OPT(L) \ge \max\left\{\frac{n-k}{n/2}, \frac{n+k}{n}\right\} \ge 4/3.$$

Másrészt a fenti hányadosokban az $OPT(L)$ érték legalább $n/2$, ami tetszőlegesen nagynak választható. Így a fenti egyenlőtlenségből adódik, hogy az A algoritmus aszimptotikus versenyképességi hányadosa legalább $4/3$, amivel a tétel állítását igazoltuk.

Pakolási minták módszere

Tekintsük a következő tárgysorozatot. Legyen $L = L_1L_2 \ldots L_k$, ahol L_i $n_i = \alpha_i n$ egyforma méretű tárgyat tartalmaz, amelyek mérete a_i. Amennyiben egy A algoritmus C-versenyképességű, akkor minden j-re teljesülnie kell a

$$C \geq \limsup_{n\to\infty} \frac{A(L_1 \dots L_j)}{OPT(L_1 \dots L_j)}$$

feltételnek. A fentiekben tekinthetjük azt az algoritmust, amelyre az általunk adható alsó korlát minimális, így célunk az

$$R = \min_A \max_{j=1,\dots,k} \limsup_{n\to\infty} \frac{A(L_1 \dots L_j)}{OPT(L_1 \dots L_j)}$$

érték meghatározása, amely érték egy alsó korlát lesz a versenyképességi hányadosra.

A *pakolási minta* egy k-dimenziós vektor $(p_1, \dots, p_k)$, amelynek a p_j koordinátája azt adja meg, hány elemet tartalmaz a láda az L_j részlistából. Pakolási minta olyan nemnegatív egész koordinátájú vektor lehet, amelyre a $\sum_{j=1}^{k} a_j p_j \leq 1$ feltétel teljesül. A minták halmaza T és jelölje $n(p)$ minden $p \in T$ esetén azon ládák számát, amely ládákat a p mintának megfelelően pakolt az algoritmus.

Minden j-re legyen T_j azon minták halmaza, amelyeknek az első pozitív együtthatója a j-edik. (Egy p minta a T_j halmazba kerül, ha $p_i = 0$ minden $i < j$ esetén, és $p_j > 0$.)

az algoritmus által az $L_1 \dots L_j$ részlista pakolása során kinyitott ládák száma a következőképpen adható meg az $n(p)$ értékekkel:

$$A(L_1 \dots L_j) = \sum_{i=1}^{j} \sum_{p\in T_i} n(p).$$

Tehát egy adott n-re a keresett A értéket a következő vegyes egészértékű programozási feladat megoldásával számíthatjuk ki.

Min R

$\sum_{p\in T} p_j n(p) = n_j, \qquad 1 \leq j \leq k$

$\sum_{i=1}^{j} \sum_{p\in T_i} n_p \leq R \cdot OPT(L_1 \dots L_j), \qquad 1 \leq j \leq k$

$n(p) \in \{0, 1, \dots\}, \qquad p \in T$

Az első k feltétel azt írja le, hogy az összes tárgyat el kell helyeznünk a ládákban. A második k feltétel az írja le, hogy az R érték valóban nem kisebb, mint az algoritmus költségének és az optimális költségnek a hányadosa a vizsgált részlistákra. Az $L_1 L_2 \dots L_k$ lista alapján a pakolási minták T halmaza és az optimális $OPT(L_1 \dots L_j)$ értékek meghatározhatók.

Többdimenziós általánosítások

A fentiekben definiált ládapakolási problémának számos változata, általánosítása van.

- Az első általánosítás a vektorpakolás, amelyben a tárgyak d-dimenziós vektorok és úgy kell elpakolni őket, hogy minden ládában minden koordinátára az ott szereplő értékek összege legfeljebb 1 legyen. A FF algoritmus aszimptotikusan $d + 7/10$ versenyképes.

- A második általánosítás a dobozpakolás, ahol többdimenziós dobozokat kell pakolni többdimenziós egységládákba. Az algoritmusok többsége a dimenziócsökkentésen alapul.

- Végül a harmadik általánosítás a sávpakolás, ahol egy egységnyi széles sávba pakolunk tárgyakat és célunk a szükséges magasság minimalizálása.

- Fontos még megemlítenünk a ládafedési problémát, amelyben célunk, hogy a tárgyakkal a lehető legtöbb ládát töltsük legalább egységnyi méretűre.

Polc algoritmusok

Egy POLC algoritmus minden téglalapot egy polcra helyez. Miután az algoritmus kiválasztotta azt a polcot, amely a téglalapot tartalmazni fogja, az algoritmus a téglalapot elhelyezi a polcon annyira balra, amennyire lehetséges a már a polcon levő egyéb téglalapok átfedése nélkül.

Tehát a téglalap érkezése után az eljárásnak két döntést kell hoznia. Az első döntés az, hogy az eljárás kialakít-e egy új polcot vagy sem. Ha új polcot alakítunk ki, meg kell határoznunk a polc magasságát is. Az újonnan kialakított polcokat mindig az előző polc tetejére helyezzük, az első polc a sáv legalján van.

A második döntés, hogy az algoritmusnak ki kell választani azt a polcot, amelyre a téglalapot helyezi.

Az NFS_r algoritmusnak egy $r < 1$ paramétertől függ. Az algoritmus minden j-re legfeljebb egy r^j magasságú aktív polcot tart fent és egy tárgy érkezése után a következő szabállyal definiálhatjuk.

A $p_i = (w_i, h_i)$ téglalap érkezése után válasszunk egy olyan k értéket, amelyre teljesül, hogy $r^{k+1} < h_i \leq r^k$. Amennyiben van r^k magasságú aktív polc, és a téglalap elhelyezhető ezen a polcon, akkor helyezzük el rajta. Ellenkező esetben alakítsunk ki egy új r^k magasságú polcot, helyezzük el a téglalapot rajta, és a továbbiakban legyen ez a polc az r^k magasságú aktív polc (ha volt korábbi aktív polc, azt lezárjuk).

NFS_r elemzése

Tétel: Az NFS_r algoritmus $\left(\frac{2}{r} + \frac{1}{r(1-r)}\right)$-versenyképes. Az NFS_r algoritmus aszimptotikusan $(2/r)$-versenyképes.

Biz: Tekintsük téglalapok tetszőleges L listáját, és jelölje H a legmagasabb téglalap magasságát. Mivel ekkor $OPT(L) \geq H$.

Jelölje H_A az L lista végén az aktív polcok összmagasságát, H_Z pedig a többi, lezárt polcok összmagasságát.

Jelölje h a legmagasabb aktív polc magasságát. Ekkor az aktív polcok magasságai a hr^i értékeket vehetik fel és minden i-re legfeljebb egy aktív polc van hr^i magassággal. Tehát az aktív polcok összmagasságára $H_A \leq h\sum_{i=0}^{\infty} r^i = \frac{h}{1-r}$.

Másrészt a $H > rh$, így $H_A \leq H/r(r-1)$.

Vegyük egy tetszőleges i-re a hr^i magasságú polcokat, jelölje ezeknek a számát n_i. Minden ilyen lezárt S polcra a következő S' polc elsőként egy olyan elemet tartalmaz, amely már nem volt elhelyezhető, így a két egymást követő polcra az elhelyezett téglalapok teljes szélessége legalább 1. Másrészt a hr^i magasságú polcokon minden tárgy magassága legalább hr^{i+1}, hiszen egyébként a tárgyat egy kisebb magasságú polcra helyeznénk.

Tehát az r^i magas polcokon elhelyezett tárgyak összterülete legalább $n_i hr^{i+1}/2$. Következésképpen az összes téglalapnak az összterülete legalább $\sum_{i=0}^{\infty} n_i hr^{i+1}/2$, így $OPT(L) \geq \sum_{i=0}^{\infty} n_i hr^{i+1}/2$. Másrészt a lezárt polcok összmagassága $H_Z = \sum_{i=0}^{\infty} n_i hr^i$, így azt kaptuk, hogy $H_Z \leq 2OPT(L)/r$. Mivel $NFS_r(L) = H_A + H_Z$.

Nyújtható téglalapok pakolása

Amennyiben a sávpakolási feladattal azt az erőforrás allokációs problémát modellezzük, amelyben a tárgyak szélessége az erőforrásigény a tárgyak magassága pedig az idő, akkor használhatjuk azt a változatot, amelyben a tárgyak mérete nem rögzített, hanem az egyes tárgyakat megnyújthatjuk a területüket változatlanul hagyva.

Az MNFS_r algoritmus annyiban különbözik az NFS-től, hogy mindig az adott polc magasságára nyújtja a tárgyat. A fenti bizonyításból adódik, hogy az algoritmus $\left(2 + \frac{1}{r(1-r)}\right)$-versenyképes a nyújtható modellben.

DS ALGORITMUS

Az első téglalap érkezése után vegyünk egy h_1 magas polcot a sáv legalján és legyen ez az aktív polc. A további elemeket pakoljuk az alábbi szabály szerint:

1. lépés Ha lehetséges a téglalapot berakni az aktív polcra, azt követően, hogy megnyújtjuk az aktív polc magasságára, akkor nyújtsuk meg és rakjuk az aktív polcra annyira balra, amennyire lehetséges. Egyébként menjünk a 2. lépésre.

2. lépés Vegyünk egy új aktív polcot, amely kétszer olyan magas, mint az előző, és tegyük az eddigi polcok tetejére. Lépjünk az 1. lépésre.

Tétel: A DS algoritmus versenyképességi hányadosa 4.