PH.D. ÉRTEKEZÉS TÉZISEI

HEURISZTIKUS ALGORITMUSOK LEGROSSZABB-ESET VIZSGÁLATA

Békési József

Szeged
1996

Optimalizáláson általában valamilyen tartományon értelmezett függvény minimumának vagy maximumának meghatározását értjük. Az optimalizálás klasszikus matematikai elmélete azt feltételezi, hogy ez a tartomány végtelen. A gyakorlati problémák esetében azonban általában csak véges számú lehetőség van, így ebben az esetben a fent említett tartomány véges. Az ilyen jellegű optimalizálást *kombinatorikus optimalizálásnak* nevezzük. Klasszikus matematikai szempontból a kombinatorikus optimalizálás nem tűnhet túl érdekesnek, mivel ha felsoroljuk az összes esetet, akkor ezek közül mindig ki tudjuk választani a legjobbat. Azonban nagyméretű problémák esetén olyan sok lehetőség van, hogy még a leggyorsabb számítógépek sem képesek elfogadható időn belül ezeket mind megvizsgálni.

Azokban az esetekben, amikor az optimumot túl nehéz megtalálni, sokszor használnak közelítő algoritmusokat, úgynevezett heurisztikákat. Nagyon fontos, hogy ezek a heurisztikák jók legyenek, azaz az esetek többségében minél jobban megközelítsék az optimumot. Természetes módon vetődik fel a kérdés, hogy miként lehet mérni a közelítő algoritmus jóságát. Erre több lehetőség van. A dolgozatban legrosszabb-eset elemzéssel foglakozunk.

Legrosszabb-eset elemzés esetén olyan viselkedés jellemzést adunk meg, amely bármilyen bemenő adatra teljesül. Ezzel a módszerrel tehát az algoritmus szempontjából kritikus adatokat vizsgáljuk, és megadjuk, hogy a legrosszabb esetben mekkora lehet az eltérés az optimális megoldástól.

A dolgozat 2. fejezetében ismertetjük az adattömörítéssel kapcsolatos legfontosabb alapfogalmakat. Bemutatjuk a statisztikai tömörítő eljárások alapelvét és ehhez kapcsolódóan az információelmélet néhány fontos eredményét. Ez az elmélet az információval kapcsolatos kérdéseket vizsgálja, beleértve az üzenetek tárolását, és a kommunikációt. Egyik legfontosabb alkalmazási területévé a tömörítés vált. A dolgozat áttekinti az adattömörítés során használt legfontosabb információelméleti tételeket, majd bemutatja a legismertebb statisztikai tömörítő

eljárásokat, így a HUFFMAN [14] és az aritmetikai kódolást [20].

A dolgozatunk további részében szöveghelyettesítésen alapuló eljárásokról esik szó. Ennek alapelve, hogy a szövegben egymás után következő karaktercsoportokat egy kóddal, vagy egy szótárra vonatkozó index-szel, esetleg mutatóval helyettesíti. A szótár olyan szavak, vagy szótöredékek listája, amelyek várhatóan gyakran fodulnak elő az adott szövegben. A szótárkódolások három fő csoportba oszthatók:

- statikus,
- szemiadaptív,
- adaptív eljárások.

A statikus kódoló eljárás a kódolandó szövegtől függetlenül mindig ugyanazzal a szótárral dolgozik. E modell hátránya, hogy amennyiben a rendelkezésre álló szótár nem illeszkedik a kódolandó szöveghez, akkor rossz eredmény születhet. Ha a szótárba sok szót, vagy szótöredéket felveszünk, akkor túl nagy lesz a mérete, ezáltal a tárolás illetve a keresés nehézkessé válik.

A szemiadaptív eljárások már jobban alkalmazkodnak a kódolandó szöveghez, ugyanis a szótárat ez alapján állítják össze. Egy adott szöveghez az optimális szótár meghatározása a szöveg hosszát tekintve $\mathcal{NP}$-teljes probléma. Az erre vonatkozó algoritmus megtalálható [2]-ben.

Az adaptív kódolás ötlete egy 1967-es cikkből származik. A gondolat lényege, hogy egy ismétlődő karaktersorozatot egy korábbi, már kódolt előfordulására vonatkozó hivatkozással helyettesítjük. A hivatkozás megvalósítása általában mutatókkal történik. Ezt az eljárást részletesen JACOB ZIV és ABRAHAM LEMPEL dolgozta ki 1977-ben[25].

A 3. fejezet kizárólag *statikus szótár* segítségével történő tömörítéssel foglalkozik. A szótár nem más, mint (*forrásszó,kódszó*) rendezett párok halmaza, ahol a forrásszó és a kódszó egy-egy véges ábécé betűiből képzett karaktersorozat, és a kódszavakat arra használjuk, hogy a tömörítendő karatersorozat megfelelő részeit, ti. a forrásszavakat a hozzájuk tartozó kódszavakkal helyettesítsük. A statikus szótárak kifejezetten hasznosak olyan esetekben, amikor egy adatbázis rekordjait kell külön-külön tömöríteni, és ugyanaz a szó, vagy szótöredék a rekordokban gyakran előfordul. Például, ha egy könyvtári katalógus bejegyzéseit szeretnénk tömöríteni, ahol szinte minden rekordban szerepelnek a szerző, cím, ISBN, könyv, stb. szavak. Feltéve, hogy statikus szótárat alkalmazunk, az adatbázis bármely rekordjával kezdhetjük a tömörítést. A célunk az, hogy az adott szótár segítségével a tömöríteni kívánt karaktersorozatot optimális módon tömörítsük, azaz úgy, hogy a minimális hosszúságú kódot kapjuk.

A fenti probléma ekvivalens egy megfelelően megválasztott, irányított, súlyozott gráfban történő legrövidebb út keresési feladattal (ld. SCHUEGRAF és HEAPS [22]). Egy adott $S = s_1 s_2 \dots s_n$ karaktersorozatra definiáljuk az $N = (V, A)$ irányított gráfot a $V = \{v_0, v_1, \dots, v_n\}$ csúcshalmazon. A gráfban definíció szerint pontosan akkor van egy $(v_i, v_{i+d}) \in A$ él, ha létezik olyan (forrásszó, kódszó) pár, ahol a forrásszó d darab olyan karaterből áll, amelyek megegyeznek az eredeti karaktersorozatban az $i+1, \dots, i+d$ pozíciókon álló karaterekkel. Egy ilyen él súlya a megfelelő kódszó bitjeinek számával egyezik meg. Látható, hogy az N gráfban a legrövidebb út v_0-tól v_n-ig éppen az S karaktersorozat optimális tömörítését adja.

A fenti modellt alkalmazva a probléma megoldása egyszerűvé válik, mivel alkalmazhatjuk az irányított, súlyozott gráfokra vonatkozó, minimális hosszúságú utat kereső, polinomiális időbonyolultságú algoritmusokat. Amennyiben a gráfnak sok *vágóéle* van (azaz olyan él, amely az eredeti problémát független részproblémákra osztja) és a keletkezett részfeladatok kicsik, a problémát az *optimális*

algoritmus használatával megoldhatjuk. Sajnos a gyakorlatban sokszor nem ez az eset áll fenn, és előfordulhat, hogy az optimális algoritmus nem megfelelő sebességű nagyon hosszú karaktersorozatok esetén. Ezért több olyan heurisztikus algoritmust fejlesztettek ki, amelyek az optimumhoz közeli megoldást adnak.

Már az 1970-es években voltak hatékony heurisztikus algoritmusok (például a *longest fragment first heurisztika* (LFF), ld. SCHUEGRAF és HEAPS[22]), azonban ezeket nem vizsgálták elméleti szempontból, csupán tapasztalati eredmények léteztek.

Sokszor előfordul, hogy olyan nagy adatállományt kell tömöríteni, amelyet egyben nem, vagy csak nehezen lehet vizsgálni, ugyanakkor szeretnénk az ilyen adatokat is viszonylag gyorsan kódolni. Az ilyen esetekben kifejezetten hasznos lehet az úgynevezett *on-line* technika. Ennek segítségével nagyon gyors heurisztikákat lehet kifejleszteni. Egy *on-line adattömörítő algoritmus* mindig a v_0 csúcsból indul, megvizsgálja az ebből kiinduló összes élt, és egy bizonyos szabály alapján választ közülük egyet. Ezután a kiválasztott él másik végénél található csúcstól folytatja tovább a kódolást. Nincs lehetőség azonban arra, hogy egy döntést a későbbiek ismeretében az algoritmus megváltoztasson.

Természetesen az on-line heurisztikák általában nem szolgáltatnak mindig optimális megoldást. Ha szeretnénk meghatározni, mennyire lehet rossz egy heurisztika, a legkézenfekvőbb módszer, hogy összehasonlítjuk az optimum által kapott eredménnyel.

Egy heurisztika *legrosszabb-eset viselkedését* általában az úgynevezett *aszimptotikus legrosszabb-eset hányadossal* szokás mérni, amelyet a következőképpen definiálnak: Legyen $D = \{(w_i, c_i) : i = 1, \ldots, k\}$ egy statikus szótár, ahol w_i a megfelelő forrásszót, c_i pedig a hozzátartozó kódszót jelenti. Tekintsünk továbbá egy tetszőleges A adattömörítő algoritmust. Legyen $A(D,S)$ illetve $OPT(D,S)$ S-nek az A illetve az optimális algoritmus által kapott tömörített kódja. Ezen kódok hosszát jelölje $\|A(D,S)\|$ illetve $\|OPT(D,S)\|$. Ekkor az A algoritmus

aszimptotikus legrosszabb-eset hányadosa

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

ahol $S(n)$ az összes n karakterből álló, a megfelelő ábécéből képzett karaktersorozatot jelenti.

Az irodalomban négy paramétert használnak az aszimptotikus legrosszabb–eset vizsgálatok során:

$$\begin{aligned} Bt(S) &= S \text{ egyes szimbólumainak hossza bitekben} \\ lmax(D) &= \max\{|w_i| \; i = 1, \ldots, k\} \\ cmin(D) &= \min\{\|c_i\| \; i = 1, \ldots, k\} \\ cmax(D) &= \max\{\|c_i\| \; i = 1, \ldots, k\}, \end{aligned}$$

ahol $|w_i|$ a w_i karaktersorozat hosszát jelenti karakterekben, $\|c_i\|$ pedig a c_i kódszó hossza bitekben. A következőkben az egyes input karakterek hosszát egyszerűen Bt-vel jelöljük, és elhagyjuk a szótárra történő hivatkozást, azaz például $lmax$-ot használunk $lmax(D)$ helyett. Az $lmax = 1$ eset itt nem érdekes, mert ekkor a már korábban vizsgált betűkódolásról van szó, ezért mindig feltételezhetjük, hogy $lmax \geq 2$.

Nem meglepő, hogy egy heurisztika legrosszabb–eset viselkedése erősen függ az adott szótár tulajdonságaitól. A dolgozat a következő típusú szótárakat vizsgálja:

Egy szótárt *általánosnak* nevezünk, ha az input ábécé minden szimbólumát tartalmazza mint forrásszót (ez biztosítja, hogy a heurisztika minden forrásszövegre el fogja érni az adott gráf nyelőjét, és ezért minden esetben befejeződik az eljárás). Ebben a dolgozatban csak általános szótárakkal foglalkozunk.

Egy általános szótár

1. *egyenlő kódhosszúságú*, ha minden kódszó hossza azonos, ($\|c_i\| = \|c_j\|$, $1 \leq i, j \leq k$),

2. *nemhosszító*, ha egy kódszó hossza sohasem haladja meg a megfelelő forrásszó hosszát ($\|c_i\| \leq |w_i|Bt$, $1 \leq i \leq k$),

3. *suffix*, ha minden w forrásszó mellett tartalmazza annak minden hátsó szeletét (azaz ha $w = \omega_1\omega_2\cdots\omega_q$ forrásszó $\Rightarrow \omega_h\omega_{h+1}\cdots\omega_q$ szintén forrásszó minden $2 \leq h \leq q$-ra),

4. *prefix*, ha minden w forrásszó mellett tartalmazza annak minden első szeletét (azaz ha $w = \omega_1\omega_2\cdots\omega_q$ forrásszó $\Rightarrow \omega_1\omega_2\cdots\omega_h$ szintén forrásszó minden $1 \leq h \leq q-1$-ra). Megjegyezzük, hogy ez a prefix tulajdonság nem egyezik meg a kódokra már korábban definiált prefix tulajdonsággal. Szótárak forrásszavaira éppen ellenkező értelemben használjuk a prefix jelzőt.

A következőkben a dolgozat bemutatja a legismertebb heurisztikus algoritmusokat, illetve elemzi ezek legrosszabb-eset viselkedését különböző típusú szótárakra.

A *leghosszabb illesztés(Longest Matching*, röviden LM) módszere az egyik legismertebb és legegyszerűbb on-line heurisztika, amely az adott gráfban az éppen aktuális csúcsból kiinduló élek közül mindig a leghosszabbat választja ki, és ezzel folytatja a kódolást. Egyenlő hosszúságú élek esetén bármelyiket választhatja az algoritmus.

KATAJAINEN ÉS RAITA [17] elemezte az LM algoritmus legrosszabb-eset viselkedését különböző típusú szótárakra a prefix kivételével és éles korlátokat bizonyított ezen tulajdonságok minden lehetséges kombinációjára.

A dolgozatban éles legrosszabb eset korlátokat bizonyítunk *prefix* szótárak minden lehetséges kombinációjára más típusú szótárakkal. Belátjuk, hogy a leghosszabb illesztés módszere a lehető legrosszabb módon is viselkedhet prefix típusú szótárak esetén. Minden *prefix* és valamilyen további $\mathcal{P}$ tulajdonsággal bíró szótárra a megfelelő korlátok megegyeznek a $\mathcal{P}$ tulajdonsággal rendelkező *általános* szótárra vontakozó korlátokkal; más szóval a prefix tulajdonság semmit sem javít az algoritmus legrosszabb-eset viselkedésén.

3.2.6. TÉTEL. [4] *Legyen D egy* prefix *szótár. Ekkor*

$$R_{LM}(D) \leq \frac{(lmax-1)cmax}{cmin}$$

és a fenti korlát éles.

3.2.7. TÉTEL. [4] *Legyen D egy* prefix *és* nemhosszító *szótár. Ekkor*

$$R_{LM}(D) \leq \begin{cases} \dfrac{(lmax-1)cmax}{cmin} & ha\ cmax \leq Bt \\ \dfrac{(lmax-2)Bt+cmax}{cmin} & ha\ Bt < cmax < 2Bt \\ \dfrac{lmax \cdot Bt}{cmin} & ha\ 2Bt \leq cmax \end{cases}$$

és a fenti korlát éles.

3.2.8. TÉTEL. [4] *Legyen D egy* prefix *és* egyenlő kódhosszúságú *szótár. Ekkor*

$$R_{LM}(D) \leq lmax - 1$$

és a fenti korlát éles.

A GONZALEZ–SMITH és STORER [13] által definiált *greedy* heurisztika, amelyet *különbségen alapuló (differential greedy*, a továbbiakban DG) algoritmusnak fogunk nevezni, minden egyes pozícióban a lehetséges (w_i, c_i) szótárelemek közül azzal fog kódolni, amelyre a legnagyobb "helyi tömörítést" éri el, azaz amelynél a $|w_i|Bt - \|c_i\|$ különbség maximális. Egyenlőség esetén bármelyiket választhatja az algoritmus. A DG-re vonatkozó eredmények a következők:

3.3.4. TÉTEL. [3] *Legyen D egy* prefix *szótár. Ekkor*

$$R_{DG}(D) \leq \begin{cases} \dfrac{cmin+(lmax-1)cmax}{cmin+(lmax-1)Bt} & ha\ (lmax-1)^2 cmax \cdot Bt \leq cmin^2 \\ & és\ \left\lfloor \frac{cmax-cmin}{Bt} \right\rfloor \geq lmax-1 \\ \dfrac{(lmax-1)cmax}{cmin} & különben \end{cases}$$

és a fenti korlátok élesek.

3.3.5. TÉTEL. [4] *Legyen D egy* prefix *és* nemhosszító *szótár. Ekkor*

$$R_{DG}(D) \leq \begin{cases} \dfrac{(lmax-1)cmax}{cmin} & ha\ cmax \leq Bt \\ \dfrac{(lmax-2)Bt+cmax}{cmin} & ha\ Bt < cmax < 2Bt \\ \dfrac{lmax \cdot Bt}{cmin} & ha\ 2Bt \leq cmax \end{cases}$$

és a fenti korlátok élesek.

Nemhosszító, suffix szótárakra Katajainen és Raita a következő tételben megfogalmazott eredményt látta be.

3.3.6. TÉTEL. (KATAJAINEN-RAITA [17]) *Legyen D egy* nemhosszító, suffix *szótár. Ekkor*

$$R_{DG}(D) \leq \frac{\min\{lmax \cdot Bt, 2cmax - Bt\}}{cmin}.$$

A legrosszabb-eset hányados pontos értéke azonban nyitott probléma volt. A következő tétel erre a kérdésre ad választ. Lényegében azt mondja ki, hogy a fenti tételben megadott korlát éles.

3.3.7. TÉTEL. (KÖZÖS GALAMBOS GÁBORRAL, ULRICH PFERSCHYVEL ÉS GERHARD WOEGINGERREL) *Pozitív egész számok végtelen sok Bt, $lmax$, $cmin$ és $cmax$ négyesére a $cmin \leq Bt$, $cmin \leq cmax$ és $cmax \leq lmax \cdot Bt$, feltételek teljesülése esetén létezik olyan* nemhosszító, suffix *D szótár, amelyre*

$$R_{DG}(D) \geq \frac{\min\{lmax \cdot Bt, 2cmax - Bt\}}{cmin}.$$

3.3.9. Tétel. (Közös Galambos Gáborral, Ulrich Pferschyvel és Gerhard Woegingerrel) *Legyen D egy* suffix *szótár és tegyük fel hogy $lmax \geq 3$. Ekkor*

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

ahol

$$L = \frac{(lmax - 1)(2cmax - (lmax - 2)Bt)}{2cmin}.$$

A fenti korlátok élesek.

Megjegyezzük, hogy az $lmax = 2$ esetben minden szótár prefix tulajdonságú, így ezt az esetet nem kell külön vizsgálni.

A *hányadoson alapuló greedy algoritmus (fractional greedy*, a továbbiakban *FG*) [6] a minden aktuális pozícióban a legnagyobb hányadossal rendelkező élet választja, azaz, ha I az adott csúcsból kiinduló élek indexhalmaza, akkor azt az i_0 indexű élet fogja választani az algoritmus, amelyre

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

3.4.1. Tétel. (Közös Galambos Gáborral, Ulrich Pferschyvel és Gerhard Woegingerrel) *Legyen D egy* általános *szótár. Ekkor*

$$R_{FG}(D) \leq \frac{(lmax - 1)cmax}{cmin}$$

és a fenti korlát éles.

3.4.2. TÉTEL. (KÖZÖS GALAMBOS GÁBORRAL, ULRICH PFERSCHYVEL ÉS GERHARD WOEGINGERREL) *Legyen D* egyenlő kódhosszúságú *szótár. Ekkor*

$$R_{FG}(D) \leq lmax - 1$$

és a fenti korlát éles.

3.4.3. TÉTEL. (KÖZÖS GALAMBOS GÁBORRAL, ULRICH PFERSCHYVEL ÉS GERHARD WOEGINGERREL) *Legyen D* nemhosszító *szótár. Ekkor*

$$R_{DG}(D) \leq \begin{cases} \dfrac{(lmax-1)cmax}{cmin} & ha\ cmax \leq Bt \\ \dfrac{(lmax-2)Bt+cmax}{cmin} & ha\ Bt < cmax < 2Bt \\ \dfrac{lmax \cdot Bt}{cmin} & ha\ 2Bt \leq cmax \end{cases}$$

és a fenti korlát éles.

3.4.4. TÉTEL. [6] *Legyen D_1 egy* általános, prefix *szótár, D_2 egy* prefix *és* egyenlő kódhosszúságú *szótár és legyen D_3 egy* prefix *és* nemhosszító *szótár. Ekkor*

$$R_{FG}(D_1) \leq \frac{(lmax-1)cmax}{cmin}$$

$$R_{FG}(D_2) \leq lmax - 1$$

$$R_{FG}(D_3) \leq \begin{cases} \dfrac{(lmax-1)cmax}{cmin} & ha\ cmax \leq Bt \\ \dfrac{(lmax-2)Bt+cmax}{cmin} & ha\ Bt < cmax < 2Bt \\ \dfrac{lmax \cdot Bt}{cmin} & ha\ 2Bt \leq cmax \end{cases}$$

és a fenti korlátok elérhetőek.

3.4.5. TÉTEL. (KÖZÖS GALAMBOS GÁBORRAL, ULRICH PFERSCHYVEL ÉS GERHARD WOEGINGERREL) *Legyen D egy* suffix *szótár. Ekkor*

$$R_{FG}(D) \leq \frac{cmax(\ln(lmax-1)+1)}{cmin}$$

és létezik olyan D_0 suffix szótár, amelyre

$$\frac{cmax(\ln(lmax-1)+1-\ln 2)}{cmin} < R_{FG}(D_0).$$

3.4.7. TÉTEL. (KÖZÖS GALAMBOS GÁBORRAL, ULRICH PFERSCHYVEL ÉS GERHARD WOEGINGERREL) *Legyen D egy* nemhosszító, suffix *szótár. Ekkor*

$$R_{FG}(D) \leq \frac{\min\{lmax\ Bt, cmax\left(\ln\left(\frac{lmax\cdot Bt}{cmax}\right)+3\right)-Bt\}}{cmin}.$$

3.4.8. TÉTEL. [6] *Létezik olyan* nemhosszító, suffix *D_1 szótár, amelyre*

$$R_{FG}(D_1) \geq \frac{cmax\left(\ln\left(\frac{lmax\cdot Bt}{cmax}\right)+1\right)}{cmin}.$$

Több mint 12 évvel ezelőtt SHUEGRAF és HEAPS [22] vezette be a *leghosszabb szelet (longest fragment first,* a továbbiakban *LFF)* heurisztikát. Azt feltételezték, hogy a tömöríteni kívánt adatok azonos hosszúságú rekordokból épülnek fel. Az ötlet a következő: az aktuális rekordon belül az algoritmus kiválasztja a leghosszabb olyan karaktersorozatot, amely pontosan megegyezik valamely szótárbeli szóval, majd ezt kódolja. Egyenlőség esetén bármelyiket választhatja az algoritmus. Ezután a maradék részt valamilyen on-line algoritmussal tömöríti az eljárás (például használható az LM heurisztika). Ha a file nem rekord struktúrájú, tekinthetünk helyette egy buffert, ami mindig a file aktuális részét tartalmazza (ezért a továbbiakban rekord helyett bufferről fogunk beszélni). Mielőtt az algoritmus a következő részt a bufferbe olvassa, annak teljes tartalmát kódolnia kell. A teljes buffer kódolási eljárást *lépésnek* fogjuk nevezni. Dolgozatunkban csak az LM heurisztikával kombinált LFF algoritmussal foglalkozunk. Ezt az eljárást LFF_{LM}-mel jelöljük. A kétszeres indexelés elkerülése érdekében $R_{LFF_{LM}}(D)$ helyett $R_{LFF}(LM, D)$-t fogunk írni.

Mostanáig nem léteztek egzakt bizonyítások az LFF_{LM} algoritmus legrosszabb-esetére vonatkozóan. Galambos Gáborral és Timo Raitával közösen éles korlátokat bizonyítottunk különböző típusú szótárakra. A következőkben ezen eredményeket mutatjuk be részletesen.

Tegyük fel, hogy az eljárás során $t \cdot lmax$ hosszú buffert használunk, ahol $t \geq 2$ egy paraméter. Először általános szótárakra vonatkozó tételeket bizonyítunk.

3.5.1. Tétel. [5] *Legyen D egy* általános *szótár. Ekkor az LFF_{LM} algoritmusra*

$$R_{LFF}(LM, D) \leq \frac{(t-1)lmax - (t-3)}{t} \cdot \frac{cmax}{cmin}.$$

teljesül.

3.5.2. Tétel. [5] *A fenti tételben megadott korlát éles LFF_{LM}-re vonatkozóan.*

3.5.3. Tétel. [5] *Legyen D egy* egyenlő kódhosszúságú *szótár. Ekkor*

$$R_{LFF}(LM, D) \leq \frac{(t-1)lmax - (t-3)}{t}$$

és a fenti korlát éles.

3.5.4. Tétel. (közös Galambos Gáborral és Timo Raitával) *Legyen D egy* nemhosszító *szótár. Ekkor*

$$R_{LFF}(LM, D) \leq \begin{cases} \dfrac{(t-1)lmax - (t-3)}{t} \dfrac{cmax}{cmin} & ha\ cmax \leq Bt \\ T & ha\ Bt < cmax < 2\,Bt \\ \dfrac{(t-1)lmax\ Bt + cmax}{t \cdot cmin} & ha\ 2\,Bt \leq cmax \end{cases}$$

ahol

$$T = \frac{(t-1)lmax\ Bt - t(2Bt - cmax) + 4Bt - cmax}{t \cdot cmin},$$

és a fenti korlát éles.

3.5.5. TÉTEL. (KÖZÖS GALAMBOS GÁBORRAL ÉS TIMO RAITÁVAL) *Legyen* D *egy* egyenlő kódhosszúságú *és* nemhosszító *szótár. Ekkor*

$$R_{LFF}(LM, D) \leq \frac{(t-1)lmax - (t-3)}{t}$$

és a fenti korlát éles.

3.5.6. TÉTEL. [5] *Legyen* D *egy* prefix *szótár. Ekkor*

$$R_{LFF}(LM, D) \leq \frac{(t-1)lmax - (t-3)}{t} \frac{cmax}{cmin}$$

és a fenti korlát éles.

3.5.7. TÉTEL. [5] *Legyen* D *egy* prefix, nemhosszító *szótár. Ekkor*

$$R_{LFF}(LM, D) \leq \begin{cases} \dfrac{(t-1)lmax - (t-3)}{t} \dfrac{cmax}{cmin} & ha\ cmax \leq Bt \\ T & ha\ Bt < cmax \leq 2Bt \\ \dfrac{(t-1)lmax\ Bt + cmax}{t \cdot cmin} & ha\ 2Bt \leq cmax, \end{cases}$$

ahol

$$T = \frac{(t-1)lmax \cdot Bt - t(2Bt - cmax) + 4Bt - cmax}{t \cdot cmin},$$

és a fenti korlátok élesek.

3.5.8. TÉTEL. [5] *Legyen* D_1 *egy* prefix,egyenlő kódhosszúságú *és* D_2 *egy* prefix, egyenlő kódhosszúságú *és* nemhosszító *szótár. Ekkor*

$$R_{LFF}(LM, D_i) \leq \frac{(t-1)lmax - (t-3)}{t} \quad i = 1, 2\text{-re.}$$

és a fenti korlát éles.

3.5.9. TÉTEL. (KÖZÖS GALAMBOS GÁBORRAL ÉS TIMO RAITÁVAL) *Legyen* D *egy* suffix *szótár. Ekkor*

$$R_{LFF}(LM, D) \leq \begin{cases} \left(1 + \dfrac{2(lmax-1)}{t}\right)\dfrac{cmax}{cmin} & ha\ t \geq 3 \\ \left(\dfrac{lmax+1}{2}\right)\dfrac{cmax}{cmin} & ha\ t = 2 \end{cases}$$

és a fenti korlátok élesek.

3.5.10. TÉTEL. (KÖZÖS GALAMBOS GÁBORRAL ÉS TIMO RAITÁVAL) *Legyen* D *egy* suffix, nemhosszító *szótár. Ekkor*

$$R_{LFF}(LM, D) \leq \begin{cases} \left(1 + \dfrac{2(lmax-1)}{t}\right)\dfrac{cmax}{cmin} & ha\ t \geq 3\ és\ cmax \leq Bt \\ \left(\dfrac{lmax+1}{2}\right)\dfrac{cmax}{cmin} & ha\ t = 2\ és\ cmax \leq Bt \\ \dfrac{cmax}{cmin} + \dfrac{2(lmax-1)}{t}\dfrac{Bt}{cmin} & ha\ t \geq 3\ és\ cmax > Bt \\ \dfrac{cmax}{cmin} + \dfrac{lmax-1}{2}\dfrac{Bt}{cmin} & ha\ t = 2\ és\ cmax > Bt \end{cases}$$

és a fenti korlátok élesek.

3.5.11. TÉTEL. (KÖZÖS GALAMBOS GÁBORRAL ÉS TIMO RAITÁVAL) *Legyen* D_1 *egy* egyenlő kódhosszúságú,suffix *és* D_2 *egy* egyenlő kódhosszúságú, suffix *és* nemhosszító *szótár. Ekkor*

$$R_{LFF}(LM, D_i) \leq \left\{ \begin{array}{ll} 1 + \frac{2(lmax-1)}{t} & ha\ t \geq 3 \\ \frac{lmax+1}{2} & ha\ t = 2. \end{array} \right\}\ i = 1, 2\text{-re.}$$

A 4. fejezetben egy ládapakolási feladatra vonatkozó algoritmus kerül bemutatásra. Az egyik leggyakrabban vizsgált kombinatorikai probléma az egydimenziós ládapakolási feladat: Adott valós számoknak egy $L = \{x_1, x_2, \ldots, x_n\}$ listája a $[0, 1)$ intevallumból, és végtelen sok egységnyi kapacitású láda. Minden egyes x_i számot egyértelműen hozzá kell rendelnünk egy ládához, úgy hogy a ládához rendelt elemek összege nem haladhatja meg az 1-et. Célunk a felhasznált ládák számának minimalizálása. Jól ismert, hogy egy optimális megoldás megkeresése $\mathcal{NP}$-teljes probléma. Következésképpen nagyon sok olyan publikáció jelent meg, amelyek hatékony, polinomiális futási idejű közelítő algoritmusokat kerestek. Az algoritmusok egy része on-line tulajdonságú. Ezen eljárások úgy helyezik el az éppen soron következő elemet a megfelelő ládába, hogy a később jövő elemekről semmilyen információval nem rendelkeznek (nem ismerik sem a méretüket, sem a számukat). Az úgyenevezett off-line algoritmusoknak több információra van szükségük: Legtöbbjük a teljes listát ismeri, mielőtt "pakolni" kezd.

Az algoritmusok hatékonyságának mérésére itt is az előző részben már bevezetett legrosszabb-eset hányadost fogjuk alkalmazni. Ládapakolási algoritmusok esetén a hányados a következőképpen definálható: Jelöljük a H heurisztika által az L lista elpakolása során elhasznált ládák számát $H(L)$-lel, illetve egy megfelelő optimális pakolásnál szükséges ládák számát L^*-gal. Ha

$$R_H(k) := \max\left\{\frac{H(L)}{k} | L^* = k\right\}$$

jelöli a maximumát a $H(L)/L^*$ hányadosnak tetszőleges olyan listára, amelyre $L^* = k$, akkor a H heurisztika R_H *aszimptotikus legrosszabb eset hányadosa:* $R_H = \limsup_{k\to\infty} R_H(k)$. Egy másik, ezzel ekvivalens definíció adható meg R_H-ra, ha észrevesszük, hogy $R_H \leq K_1$, ha létezik két olyan K_1 és K_2 konstans, hogy

$$H(L) \leq K_1 \cdot L^* + K_2$$

minden L listára. Nyilvánvalóan a legkisebb ilyen K_1 megegyezik R_H-val.

Időrendi sorrendben az első off-line algoritmust D. S. JOHNSON definiálta [16]. A legtöbb publikált algoritmus legalább $O(n \log n)$-es időbonyolultsággal rendelkezik (lásd például a First Fit Decreasing, Best Fit Decreasing heurisztikákat). A lineáris időbonyolultságú algoritmusok mindig "üdítő kivételt" képeztek. Az első ilyen algoritmus a Group Fit Group, amelyet D. S. JOHNSON definiált [16], és 1.5-ös aszimptotikus legrosszabb-eset hányadossal rendelkezett. RAMANAN és társai [19] egy 1.612-es aszimptotikus legrosszabb-eset hányadossal bíró heurisztikát adtak meg. Hosszú ideig a JOHNSON algoritmust nem sikerült felülmúlni, de F. DE LA VEGA és G. S. LUEKER bebizonyította híres cikkében [24], hogy minden $\varepsilon > 0$-ra létezik olyan A algoritmus, hogy $A(L) \leq (1+\varepsilon)L^* + C_\varepsilon$, és A futási ideje $O(n) + D_\varepsilon$. Fontos, hogy a D_ε és C_ε konstansok csak ε-től függenek, n-től nem. LUEKER és DE LA VEGA nem számította ki pontosan ezeket a konstansokat, de azt sejtették, "meglehetősen nagyok" lehetnek, egész pontosan azt is tudták, hogy $1/\varepsilon$-tól exponenciálisan függenek. Néhány évvel később C. U. MARTEL [18]-ban észrevette: $\varepsilon = 1/3$ esetén a cikkben szereplő D_ε nagyobb mint $\binom{49}{7}$. Ennek van egy fontos következménye: Az elméleti szempontból kiváló algoritmus nem használható a gyakorlatban. A fent idézett cikkben MARTEL egy rendkívül szellemes lineáris idejű algoritmust publikált. A lista elemeit "kupacokba" gyűjtötte, és az egyes osztályokban lévő elemek számától függően intelligens módon kombinálta őket (lásd a következő fejezetet). MARTEL algoritmusa 4/3-os aszimptotikus legroszszabb-eset hányadossal rendelkezik. A cikk záradékában MARTEL megemlítette, hogy lehetséges, hogy a technika segítségével az algoritmus megjavítható $O(n)$ időbonyolultságú, 5/4-es aszimptotikus legrosszabb-eset hányadosúvá. Azt javasolta, hogy a kis elemeket ügyesebben kellene kezelni.

Bár a gondolat könnyen megvalósíthatónak tűnt, eddig nem született eredmény. A dolgozatban egy lineáris idejű heurisztika kerül bemutatásra, amelyet GALAMBOS GÁBORRAL és HANS KELLERERREL közösen találtunk. Az eljárás

MARTEL ötletén alapszik és 5/4-es aszimptotikus legrosszabb-eset hányadossal rendelkezik. Bebizonyítottuk, hogy az algoritmusra (amelyet H_7-tel fogunk jelölni) teljesül a $H_7(L) \leq \frac{5}{4}L^* + 5$ egyenlőtlenség, bármely L listára.

Az eljárás az első lépésben osztályozza az elemeket az alábbiak szerint:

$$C_0 = \{x_i | 1 \geq x_i > \tfrac{4}{5}\},$$
$$C_1 = \{x_i | \tfrac{4}{5} \geq x_i > \tfrac{2}{3}\},$$
$$C_2 = \{x_i | \tfrac{2}{3} \geq x_i > \tfrac{1}{2}\},$$
$$C_3 = \{x_i | \tfrac{1}{2} \geq x_i > \tfrac{3}{8}\},$$
$$C_4 = \{x_i | \tfrac{3}{8} \geq x_i > \tfrac{1}{3}\},$$
$$C_5 = \{x_i | \tfrac{1}{3} \geq x_i > \tfrac{1}{4}\},$$
$$C_6 = \{x_i | \tfrac{1}{4} \geq x_i > \tfrac{1}{5}\},$$
$$C_7 = \{x_i | \tfrac{1}{5} \geq x_i > 0\}.$$

Legyen $c_i = |C_i|$, $i = 0, \ldots, 7$. Az algoritmus ismertetése során azonban c_i mindig azon C_i-elemek számát fogja jelölni, amelyeket még *nem* rendelt hozzá az algoritmus valamely ládához.

A H_7 algoritmus: (KÖZÖS GALAMBOS GÁBORRAL ÉS HANS KELLERERREL)

1. Alakítsuk ki a C_i, $i = 0, \ldots, 7$ halmazokat.

2. Tegyük a C_0-elemeket külön ládába.

3. Legyen $k_1 := \lceil \min\left\{\frac{c_1}{2}, \frac{c_5+c_6}{2}\right\} \rceil$. Vágjuk szét C_1-et két részhalmazra: C_1^a tartalmazza a legkisebb k_1 elemét C_1-nek, és C_1^b a maradék elemeket. Hasonló módon bontsuk fel a $C_{5,6} := C_5 \cup C_6$ halmazt a $C_{5,6}^a$ és $C_{5,6}^b$ részhalmazokra. Vegyünk ki tetszőlegesen egy-egy elemet a C_1^a és a $C_{5,6}^a$ halmazokból. Ha beleférnek egy ládába, akkor nyissunk egy új ládát, és helyezzük el őket benne. Ha nem férnek el egy ládába, tegyük a C_1^a-elemet egy üres ládába. Továbbá rakjuk a C_1^b-elemeket is üres ládákba.

4. Legyen $k_2 := \lceil \min\left\{\frac{c_2}{2}, \frac{c_2+c_4}{2}\right\}\rceil$. Vágjuk szét C_2-t két részhalmazra: C_2^a tartalmazza a legkisebb k_2 elemét C_2-nek, és C_2^b a maradék elemeket. Hasonló módon bontsuk fel a $C_{3,4} := C_3 \cup C_4$ halmazt a $C_{3,4}^a$ és $C_{3,4}^b$ részhalmazokra. Vegyünk ki tetszőlegesen egy-egy elemet a C_2^a és a $C_{3,4}^a$ halmazokból. Ha beleférnek egy ládába, akkor nyissunk egy új ládát, és helyezzük el őket benne. Jelöljük $\tilde{C}_2$-mal a C_2^b-elemeket és a maradék C_2^a-elemeket. Legyen $\hat{c}_5 := \min\{|\tilde{C}_2|, c_5\}$ és $\hat{C}_5$ a legnagyobb $\hat{c}_5$ eleme C_5-nek. Helyezzük $\tilde{C}_2$ elemeit $\hat{C}_5$ -beli elemekkel párosítva üres ládákba. Ha ($C_5 = \emptyset$) akkor párosítsuk $\tilde{C}_2$ elemeit C_6-beli elemekkel, illetve ha ($C_{5,6} = \emptyset$) tegyük őket egyedül üres ládákba.

5. Legyen $q := \min\{c_4, 2c_6\}$. Legyen C_4^q a C_4 halmaz q legnagyobb eleme, illetve legyen C_6^q a C_6 halmaz $\lfloor \frac{q}{2} \rfloor$ legnagyobb eleme. Tegyünk két C_4^q-elemet és egy C_6^q-elemet egy üres ládába, amíg $c_4 \leq 1$ vagy $c_6 = 0$ nem teljesül.

6. Tegyünk egy C_3-elemet és két C_6-elemet egy üres ládába, amíg $c_3 = 0$ vagy $c_6 \leq 1$ nem teljesül.

7. Párosítsunk össze két C_3-elemet, amíg $c_3 \leq 1$ nem teljesül.

8. Legyen $k_3 := \lceil \min\left\{\frac{c_3}{2}, \frac{c_4}{4}\right\}\rceil$. Vágjuk szét C_5-öt két részhalmazra: C_5^a tartalmazza a legkisebb k_3 elemét C_5-nek, és C_5^b a maradék elemeket. Vágjuk szét C_4-et is két részhalmazra: C_4^a tartalmazza a legkisebb $2k_3$ elemét C_4-nek, és C_4^b a maradék elemeket. Vegyünk ki tetszőlegesen egy elemet a C_1^a és két elemet a C_4^a halmazokból. Ha beleférnek egy ládába, akkor nyissunk egy új ládát, és helyezzük el őket benne. Ha nem férnek el egy ládába, tegyük a két C_4^a-elemet egy üres ládába. Továbbá rakjuk a C_4^b-elemeket is párosával üres ládákba, amíg $c_4 \leq 1$ nem teljesül.

9. Helyezzük a megmaradó C_5-elemeket hármasával üres ládákba, amíg $c_5 \leq 2$ nem teljesül.

10. Helyezzük a megmaradó C_6-elemeket négyesével üres ládákba, amíg $c_6 \leq 3$ nem teljesül.

11. A megmaradó C_3, C_4, C_5, C_6 elemeket rakjuk optimális módon üres ládákba.

12. A C_7-elemeket rakjuk el a Next-Fit szabály szerint.

A dolgozatban belátjuk, hogy $H_7(L) \leq \frac{5}{4}L^* + 3$ tetszőleges L listára. A következő jelöléseket fogjuk használni:

- Azt mondjuk, hogy a B láda $\langle i_1 i_2 \dots i_k \rangle$ *típusú*, ha pontosan k elemet tartalmaz, méghozzá úgy, hogy az első elem a C_{i_1} halmazból, a második a C_{i_2} halmazból van, stb. ($i_m = i_n, 1 \leq m, n \leq k$ lehetséges).

- Jelölje $y_{i_1,i_2,\dots,i_k}$ L egy optimális pakolásában az $\langle i_1, i_2, \dots, i_k \rangle$ típusú ládák számát.

- Jelölje m_{15} azon C_1-elemek számát, amelyeket a 3. lépésben C_5-elemmel pakolt össze a H_7 algoritmus. Hasonlóan, m_{16}, m_{23}, m_{24}, m_{25}, m_{26} és m_{445} jelölje a 3., 4., és 8. lépésben képzett különböző párok, illetve hármasok számát.

A a legrosszabb-eset viselkedésre vonatkozó tétel bizonyítása előtt két lemmát kellett igazolnunk.

4.3.1. LEMMA. [7] *Jelölje c'_3 azon C_3-elemek számát, amelyeket a 6., 7., illetve 10. lépésben pakol el a H_7 algoritmus, illetve c'_6 az ugyanilyen C_6-elemek számát. Ekkor a heurisztikának legfeljebb $c'_3/2 + c'_6/4$ ládára van szüksége ezen elemek elpakolásához.*

4.3.2. LEMMA. [7] *Tegyük fel, hogy a H_7 heurisztika 8. lépése legalább egy olyan hármast képez, amelyik nem fér el egy ládában. Ekkor,*

$$y_{345} + y_{445} \leq \frac{c_4 - m_{24} - 2(c_6 - m_{16})}{4} + 2m_{445} + m_{15} + m_{24} - 1.$$

4.3.3. TÉTEL. (A BIZONYÍTÁS A, B ÉS C ESETE KÖZÖS GALAMBOS GÁBORRAL ÉS HANS KELLERERREL) $H_7(L) \leq \frac{5}{4}L^* + 3$ *tetszőleges* L *listára. Továbbá végtelen sok olyan* L_n *lista létezik, amelyre* $\frac{H_7(L_n)}{L_n^*} = 5/4$.

Irodalomjegyzék

[1] N. ABRAMSON, Information theory and coding, *McGraw-Hill. New York*, (1963).

[2] T. C. BELL, J. G. CLEARY, I. H. WITTEN, Text compression, *Prentice Hall, Englewood Cliffs NJ*, (1990).

[3] BÉKÉSI J., GALAMBOS G., U. PFERSCHY, G. J. WOEGINGER, Greedy algorithms for on-line data compression, *Operations Research Proceedings 1994, Selected Papers of the International Conference on Operations Research,Berlin*, Eds.: Ulrich Derigs, Achim Bachem, Andreas Drexl, (1994).

[4] BÉKÉSI J., GALAMBOS G., U. PFERSCHY, G. J. WOEGINGER. Worst-case analysis for on-line data compression, *Lecture Notes in Computer Science, Combinatorics and Computer Science, 8th Franco-Japanese - 4th Franco-Chinese Conference on Combinatorics and Computer Science, Brest, France, Selected Papers*, Eds.: Michel Deza, Reinhardt Euler, Ioannis Manoussakis (1995).

[5] BÉKÉSI J., GALAMBOS G., T. RAITA, The Longest Fragment First algorithm for data compression, *Proceedings of the XIII. International Conference on Mathematical Programming, Mátraháza*, közlésre elfogadva, (1996).

[6] BÉKÉSI J., GALAMBOS G., U. PFERSCHY, G. J. WOEGINGER, The fractional greedy algorithm for data compression, *Computing* 56:29-46, (1996).

[7] BÉKÉSI J., GALAMBOS G., H. KELLERER, A 5/4 Linear Time Bin-Packing Algorithm, *SIAM Journal on Computing*, közlésre benyújtva, (1996).

[8] E. G. COFFMAN JR., G. S. LUEKER, Probabilistic Analysis of Packing and Partitioning Algorithms, *John Wiley & Sons, New York*, (1991).

[9] DEMETROVICS J., J. DENEV, R. PAVLOV, A számítástudomány matematikai alapjai, *Tankönyvkiadó, Budapest*, (1984).

[10] R. M. Fano, The transmission of information, *Techninal report* 65, *Research Laboratory of Electronics, MIT, Cambridge,MA*, (1949).

[11] GALAMBOS G., Ládapakolási feladatok közelítő algoritmusainak legrosszabb-eset vizsgálata, *Kandidátusi értekezés*, (1993).

[12] M. R. GAREY, D. S. JOHNSON, Computers and Intractability: A Guide to the Theory of NP-completeness, *W. H. Freeman & Co., San Francisco*, (1979).

[13] M. E. GONZALEZ-SMITH, J. A. STORER, Parallel algorithms for data compression, *Journal of the ACM*, 32:344-373, (1985).

[14] D. A. HUFFMAN, A method for the construction of minimum-redundancy codes, *Proc. Institute of Electrical and Radio Engineers*, 40(9):1098-1101, (1952).

[15] F. JELINEK, Probabilistic information theory, *McGraw-Hill, New York*, (1968).

[16] D. S. JOHNSON, Fast algorithms for bin packing, *J.Comput. Syst. Sci*, 8:272-314 (1974).

[17] J. KATAJAINEN, T. RAITA, An analysis of the longest matching and the greedy heuristic in text encoding, *Journal of the ACM* 39:281-294, (1992).

[18] C. U. MARTEL, A linear time bin-packing algorithm, *Op. Res. Lett*, 4:189-192 (1985).

[19] P. RAMANAN, D. BROWN, C. C. LEE AND D. T. LEE, On-line bin packing in linear time, *J. of Algorithms*, 10:305–326 (1989).

[20] J. J. RISSASSEN, G. G. LANGDON, Arithmetic coding, *IBM J. Research and Development*, 23(2):149-162, (1979).

[21] E. J. SCHUEGRAF, H. S. HEAPS, Selection of equifrequent word fragments for information retrieval, *Inf. Stor. Ret.* 9:697–711, (1973).

[22] E. J. SCHUEGRAF, H. S. HEAPS, A comparison of algorithms for data base compression by use of fragments as language elements, *Inf. Stor. Ret.* 10:309–319, (1974).

[23] C. E. SHANNON, A mathematical theory of communication, *Bell System Technical J.*, 27:398-403, (1948).

[24] W. FERNANDEZ DE LA VEGA, G. S. LUEKER, Bin packing can be solved within $1+\varepsilon$ in linear time, *Combinatorica*, 1:349-355 (1981).

[25] J. ZIV, A. LEMPEL, A universal algorithm for sequential data compression, *IEEE Trans. on Information Theory*, IT-23(3):337-343, (1977).