József Attila Tudományegyetem

Heurisztikus algoritmusok legrosszabb-eset vizsgálata

Ph.D. értekezés

Készítette: Békési József
Témavezető: dr. Galambos Gábor
főiskolai tanár

Szeged
1996

Tartalomjegyzék

Köszönetnyilvánítás

Nagyon sok embernek tartozom köszönettel azért, hogy ez a dolgozat elkészülhetett. Elsőként témavezetőmnek, *Galambos Gábornak* szeretném megköszönni, hogy megismertetett a legrosszabb-eset analízis alapelveivel és az ezen a területen használt legfontosabb technikákkal. Lehetőséget teremtett számomra hogy kapcsolatba kerüljek a kombinatorikus optimalizálás több jeles nemzetközi képviselőjével, akikkel közös kutatómunkát végezhettünk. Ez meghatározó volt a dolgozat anyagának elkészítésében. Hálás vagyok témavezetőmnek a dolgozatommal kapcsolatos tanácsaiért, esetenkénti kritikájáért, amelynek nagy szerepe volt a végső forma kialakításában.

Köszönettel tartozom további szerzőtársaimnak, *Hans Kellerernek*, *Ulrich Pferschynek*, *Timo Raitanak* és *Gerhard Woegingernek* is. A velük végzett munka fontos szerepet játszott szakmai fejlődésemben. Nagy segítség volt számomra, hogy néhány napot tölthettem a Graz-i Műszaki Egyetemen. Az egyetem kombinatorikus optimalizálással foglalkozó csoportjában színvonalas és hatékony munkának voltam tanúja.

A fenti látogatáson túl az OTKA, az ÖAD és a FEFA pénzügyi támogatásának köszönhetően részt vehettem nemzetközi konferenciákon is, ahol lehetőségem volt előadásokat tartani. Úgy érzem ez is fontos része volt szakmai fejlődésemnek.

Végül, de nem utolsósorban köszönettel tartozom kollégáimnak, munkatársaimnak, akiknek támogató ötleteiből, tanácsaiból sokat merítettem.

1. fejezet

Bevezetés

A gyakorlati életben gyakran szembe találjuk magunkat azzal a problémával, hogy hogyan oldjunk meg egy feladatot optimális módon. Ilyen lehet például ha egy számítógép merevlemezéről minimális számú hajlékony lemezre szeretnénk lemásolni néhány állományt. Más esetben az előző háttértároló tartalmát szeretnénk úgy tömöríteni, hogy a lehető legkevesebb helyet foglalja el a tömörített változat. Előfordulhat az is, hogy bizonyos nyersanyagból kell levágnunk valamekkora méretű darabokat úgy, hogy a veszteségünk a lehető legkisebb legyen. Általában ezen problémák mindegyike valamilyen optimalizálási feladatra vezethető vissza. Legáltalánosabban úgy fogalmazhatunk, hogy optimalizáláson 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.

Bonyolultságelméleti szempontból azt tekintjük elfogadhatónak, ha az optimum meghatározásához szükséges lépésszám a bemenő adatok számának polinomiális függvénye. Ezen belül is gyakorlati szempontból sok esetben csak akkor megfelelő az eljárás, ha a polinom fokszáma kicsi, például lineáris, vagy másodfokú. Gyakran azonban a célfüggvény túl bonyolult, vagy a probléma mérete túl nagy, ezért nem lehetséges a problémát polinomiális időben megoldani. A matematika, illetve a számítástudomány külön elméletet dolgozott ki az ilyen problémákra. Ez az NP-teljesség elmélete [12].

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. Általában hármat szoktak alkalmazni: a tapasztalati becslést, az átlagos–eset analízist és a legrosszabb–eset elemzést.

Tapasztalati becslés esetén több inputon lefuttatjuk az algoritmust, majd a számítógép segítségével kiszámítjuk, hogy a kapott eredmények mennyire térnek el az optimális megoldástól. Sok esetben azonban az optimumot nehéz megtalálni, ezért ezzel a módszerrel nem mindig lehet meghatározni, milyen távol vagyunk az optimumtól. Így a módszer inkább heurisztikák gyakorlati összehasonlítására alkalmazható.

Az átlagos–eset analízis elegánsabb matematikai módszer. Miután a bemenő adatokat valamilyen valószínűségi eloszlás szerint választjuk, az optimális megoldás és a heurisztika eredménye véletlen változók lesznek, amelyekből átlagos–eset viselkedést lehet számolni. Az eltérést különböző statisztikai mértékekkel jellemezzük (pl. átlag, szórás, stb.).

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. COFFMAN és LUEKER könyve [8] részletes elemzést ad a különböző technikákról.

A dolgozat heurisztikus algoritmusok legrosszabb–eset viselkedését vizsgálja. Szerkezetét tekintve két fő részre osztható. Az első részben az adattömörítés elméletéről szóló általános ismertető után különböző, szótárkódoláson alapuló adattömörítési heurisztikák legrosszabb–eset elemzését tárgyalja. Ezen belül a dolgozat választ ad néhány ismert heurisztikára vonatkozó eddig nyitott problémára, majd egy új heurisztikát mutat be. A dolgozat második részében egy új ládapakolási algoritmus, és annak analízise található. Az értekezés nagy része publikált, illetve közlésre leadott cikkeken alapszik.

2. fejezet

Az adattömörítés

2.1. Bevezetés

Tömörítésen általában olyan eljárást értünk, amelynek segítségével valamilyen információt kevesebb bit, illetve byte segítségével adhatunk meg. Információ többféle formában létezhet, beszélhetünk szöveges, kép, illetve hang jellegű információról. Tömörítési szempontból a szövegeknél figyelembe kell venni azt, hogy az eredeti változatnak mindig pontosan rekonstruálhatónak kell lenni a tömörített formából. Kép, illetve hang rekonstruálásakor keletkező apróbb eltérések még elfogadhatóak lehetnek. A dolgozatban adattömörítési eljáráson olyan módszert értünk, amely biztosítja a pontos visszaállíthatóságot.

Annak ellenére, hogy napjainkban a számítógépes tárkapacitások nagyarányú növekedést mutatnak, a tömörítésnek mégis nagy jelentősége van. A fő ok talán az, hogy az emberek mindig szívesebben veszik, ha viszonylag alacsony költséggel sikerül "megnövelni" a tárolók kapacitását, mintsem hogy újabb beruházásokat kelljen végezniük. Fontos szerepe van a tömörítésnek az információ átvitelénél is. A kommunikációs vonalak sebessége napjainkban még nem olyan jó, hogy nagyobb mennyiségű információt gyorsan lehessen továbbítani. Ezért lényeges időt és költséget lehet megtakarítani, ha egy adott mennyiségű információt rövidebb

formában viszünk át. Mindezek jól indokolják azt a fejlődést, amelyen a tömörítő eljárások az utóbbi néhány évtizedben átmentek.

2.2. Alapvető definíciók, megjegyzések

2.2.1. DEFINÍCIÓ. *Tömörítésen olyan eljárást értünk, amely egy D információ mennyiséget egy kisebb $\Delta(D)$ információ mennyiséggé kódol. Veszteség nélküli tömörítő eljárásnak nevezzük azt az eljárást, amelynél a $\Delta(D)$-ből az eredeti D pontosan visszanyerhető. Veszteséges tömörítő eljárásnak nevezzük az olyan eljárást, amelynél az eredeti információ csak közelítőleg nyerhető vissza a tömörített formából.*

Ebben a dolgozatban csak veszteség nélküli eljárásokkal foglalkozunk, ezért a továbbiakban mindig feltételezzük, hogy célunk a pontos visszaállíthatóság. Általában ezeket az eljárásokat alkalmazzák adatbázisok, szövegfile-ok, egyéb adatállományok tömörítésére. A tömörítés olyan eljárás, amelynek hatékonyságát jelentősen befolyásolja a tömörítendő információ jellege, és természetesen az alkalmazott módszer is. Ezzel kapcsolatban már itt fontosnak tartunk megemlíteni egy közismert, lényeges állítást.

2.2.2. TÉTEL. [2] *Nem létezik olyan eljárás, amely minden adatot képes tömöríteni.*

□

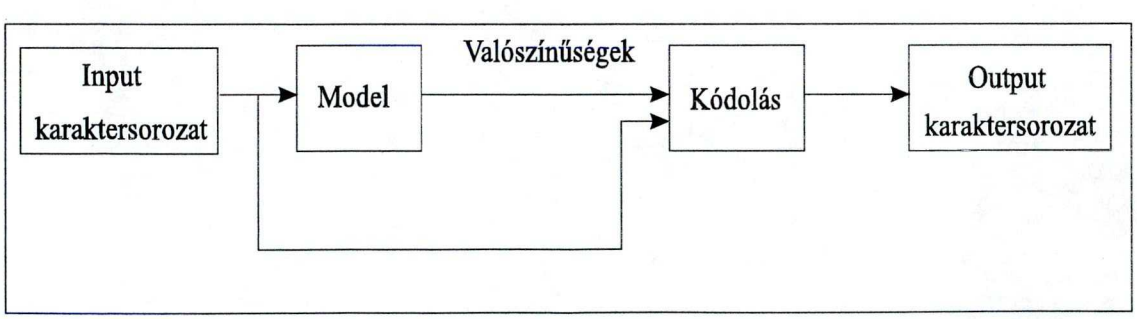
A tömörítéssel kapcsolatban a legalapvetőbb kérdés, hogy egy adatállomány mekkora információtartalommal rendelkezik. Általában ugyanis az általa ténylegesen elfoglalt tárterület nem tükrözi ezt jól. Azt mondhatjuk, hogy az adatok sokszor redundánsak, valódi információtartalmuknál nagyobb tárterületet igényelnek. A tömörítés célja ezen redundanciák megszüntetése. A tömörítő

eljárás a gyakorlatban legtöbbször úgy történik, hogy kapunk egy karaktersorozatot, és a szimbólumokat kódokká alakítjuk. Az, hogy egy szimbólumhoz milyen kódot rendelünk, függ az alkalmazott modelltől. A modell a kódolandó szimbólumsorozatra vonatkozó adatokat, szabályokat tartalmazza. A modellezés célja, hogy felismerjük a karaktersorozat tömörítés szempontjából lényeges tulajdonságait.

Az eddig ismert veszteség nélküli tömörítő eljárásokat két nagy csoportra oszthatjuk. Az egyikbe tartoznak a statisztikai, a másikba pedig a szótárkódoláson alapuló eljárások. Bár a dolgozatban saját eredmények csak a második témából vannak, mégis a teljesség kedvéért mindkét csoporttal részletesebben foglalkozunk.

2.3. Statisztikai tömörítő eljárások

A statisztikai kódoló eljárásoknál a modellezés alapja az egyes szimbólumok előfordulási valószínűségének meghatározása. A következő ábra a tömörítés folyamatát mutatja:



1. ábra

A statisztikai kódolások elméleti alapjait az információelmélet néhány fontos eredménye adja. (Az információelmélet a matematika egyik ága, mely CLAUDE SHANNON munkája nyomán alakult ki az 1940-es években [23]. Később nagyon

sok publikáció, könyv jelent meg ebben a témában [1],[9],[15].) Az elmélet az információval kapcsolatos kérdéseket vizsgálja, beleértve az üzenetek tárolását, és a kommunikációt. Egyik legfontosabb alkalmazási területévé a tömörítés vált. A leglényegesebb fogalom az *entrópia*, amely azt méri, hogy egy karaktersorozat mekkora információtartalommal rendelkezik. A fogalom a termodinamikából ered, ott is hasonló jelentése van. Minél nagyobb egy üzenet entrópiája, annál több információtartalommal bír. A következőkben áttekintjük az adattömörítés során használt legfontosabb fogalmakat és fontosabb tételeket.

2.3.1. DEFINÍCIÓ. *Ábécének nevezünk egy A véges, nem üres halmazt. Az ábécé elemeit karaktereknek, vagy szimbólumoknak nevezzük. Az ábécé elemszámát $|A|$ jelöli. Karaktersorozaton az ábécé karaktereiből alkotott sorozatot értünk. Általában feltételezzük, hogy egy karaktersorozat véges.*

2.3.2. DEFINÍCIÓ. *Legyen $A = \{a_1, ..., a_n\}, n \geq 1$ egy ábécé. Forrásnak nevezünk egy olyan F eljárást, amely A karaktereit bocsátja ki. A forrást elsőrendűnek nevezzük, ha A karaktereihez független $p_1, ..., p_n (p_i > 0, i = 1, ..., n)$ valószínűségek rendelhetők, amelyek az egyes karakterek kibocsátási valószínűségét jelentik.*

2.3.3. DEFINÍCIÓ. *Legyen $A = \{a_1, ..., a_n\}, n \geq 1$ tetszőleges ábécé, továbbá legyen $K = \{\alpha_1, ..., \alpha_n\}$ véges hosszúságú bináris sorozatoknak valamilyen n elemű halmaza. A sorozatok lehetnek különböző hosszúak is. Betű szerinti kódolásnak nevezzük azt az eljárást, amely az ábécé minden betűjének egy $\alpha_i \in K$ bináris sorozatot feleltet meg. K-t kódnak, az $\alpha_1, ..., \alpha_n$ elemeket kódszavaknak nevezzük.*

2.3.4. DEFINÍCIÓ. *Két bináris sorozat, illetve karaktersorozat szorzatán a sorozatok egymásután írásával kapott sorozatot értjük. Ennek megfeleően egy sorozat n. hatványán a hagyományos hatványfogalmat értjük a fenti szorzással.*

2.3.5. DEFINÍCIÓ. *A $K = \{\alpha_1, ..., \alpha_n\}$ kódot felbonthatónak nevezzük, ha tetszőleges bináris sorozat legfeljebb egyféleképpen bontható fel kódszavak szorzatára.*

2.3.6. DEFINÍCIÓ. *A $K = \{\alpha_1, ..., \alpha_n\}$ kódot prefix kódnak nevezzük, ha egyetlen kódszó sem valódi kezdőszelete (prefixuma) egy másik kódszónak.*

A prefix kódoknak azért van nagy jelentősége, mert ha ilyen kódot használunk, akkor a tömörített formából történő dekódolás egyértelműen végrehajtható. Ezt mutatja a következő tétel.

2.3.7. TÉTEL. [9] *Minden prefix kód felbontható.*

□

Tegyük fel, hogy adott egy F elsőrendű jelforrás, ami az $A = \{a_1, ..., a_n\}, n \geq 1$, ábécé betűit bocsátja ki. Jelölje p_i annak a valószínűségét, hogy az F által kibocsátott jel a_i. Ekkor $p_i > 0$, és $\sum_{i=1}^{n} p_i = 1$.

Tegyük fel továbbá, hogy az F jelforrás által kibocsátott jelek A ábécéjét a $K = \{\alpha_1, ..., \alpha_n\}$ kóddal kódoltuk, és jelölje $l_1, ..., l_n$ az $\alpha_1, ..., \alpha_n$ kódszavak hosszát.

2.3.8. DEFINÍCIÓ. *Az $L(K) = \sum_{i=1}^{n} p_i l_i$ számot a K kód F jelforrás melletti költségének nevezzük.*

2.3.9. DEFINÍCIÓ. *A K^0 felbontható kódot az F jelforrásra nézve optimálisnak mondjuk, ha bármely K felbontható kódnak az F mellett számított $L(K)$ költsége nem kisebb $L(K^0)$-nál.*

A célunk, hogy olyan kódrendszert találjunk, amely az adott valószínűségeket figyelembe véve optimális, azaz a legkisebb költséggel rendelkezik. (Ezenkívül persze szeretnénk biztosítani a visszaállíthatóságot is.) A következő tételek azt bizonyítják, hogy ilyen kódrendszer létezik, és az is látható, hogy az egyes kódszavak hosszát hogyan kell megválasztani, hogy optimális kódot kapjunk.

2.3.10. TÉTEL. [9] *Tetszőleges F jelforráshoz létezik optimális prefix kód.*

□

2.3.11. DEFINÍCIÓ. *A* $H(F) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$ *számot az F forrás entrópiájának nevezzük.*

2.3.12. TÉTEL. [9] *Egy F forráshoz tartozó tetszőleges K felbontható kódra* $L(K) \geq H(F)$ *teljesül.*

□

A statisztikai kódoló eljárások az előbbiekben ismertetett elméleti eredményeken alapulnak. Az eljárások olyan felbontható kódot igyekeznek meghatározni, amelynek költsége optimális, vagy azt jól közelíti. A 2.3.10. tétel szerint ilyen létezik. A 2.3.12. tétel alapján az egyes szimbólumokhoz tartozó kódokat úgy kell megválasztani, hogy azok hossza a szimbólum valószínűsége reciprokának logaritmusát minél jobban megközelítse. Erre több eljárást is kifejlesztettek. A legismertebbek a SHANON-FANO [10], a HUFFMAN [14] és az aritmetikai kódolás [20]. Ezek a módszerek mindig feltételezik, hogy adottak a szimbólumok előfordulási valószínűségei. A gyakorlatban a valószínűségek becslése legegyszerűbben az egyes szimbólumok relatív gyakoriságának meghatározásával történhet. Ebben az esetben ez jelenti a modellezést.
A következőkben a két legismertebb és leggyakrabban használt statisztikai tömörítő eljárást ismertetjük.

HUFFMAN kódolás [14]:
A módszer a következő:

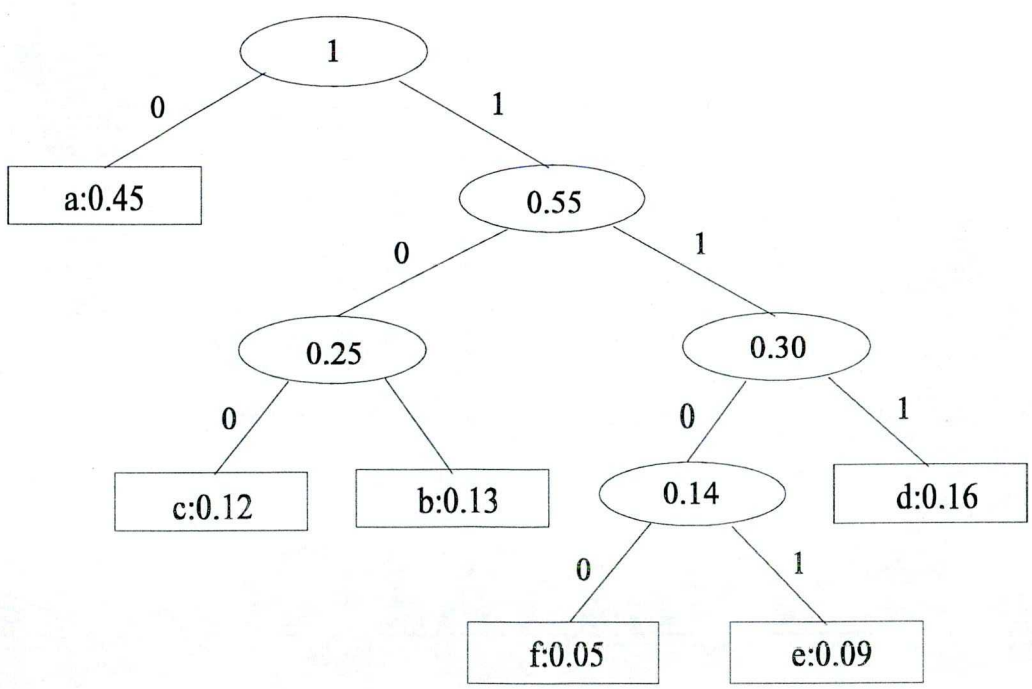
- Határozzuk meg a karakterek előfordulási gyakoriságát.

- Listázzuk az összes szimbólumot valószínűségeik szerint növekvő sorrendben.
- Tekintsük a két legkisebb valószínűségű szimbólumot.
- Helyettesítsük ezeket a szimbólumokat egy őket tartalmazó halmazzal, amelynek valószínűsége a két szimbólum valószínűségének összege. Az egyik szimbólum kódjához egy 0 bitet, a másikéhoz 1-es bitet rendeljünk.
- Ismételjük az előző három lépést, amíg el nem jutunk egy olyan a listához, amely egy elemet tartalmaz. Később a szimbólumok helyett a keletkezett halmazokat kell tekintenünk.

Tekintsük a következő példát, a halmazokat fával reprezentrálva.
Az input adatok:

szimbólum	f	e	c	b	d	a
valószínűség	0.05	0.09	0.12	0.13	0.16	0.45

A keletkezett HUFFMAN kód:



2. ábra

Aritmetikai kódolás [20]:

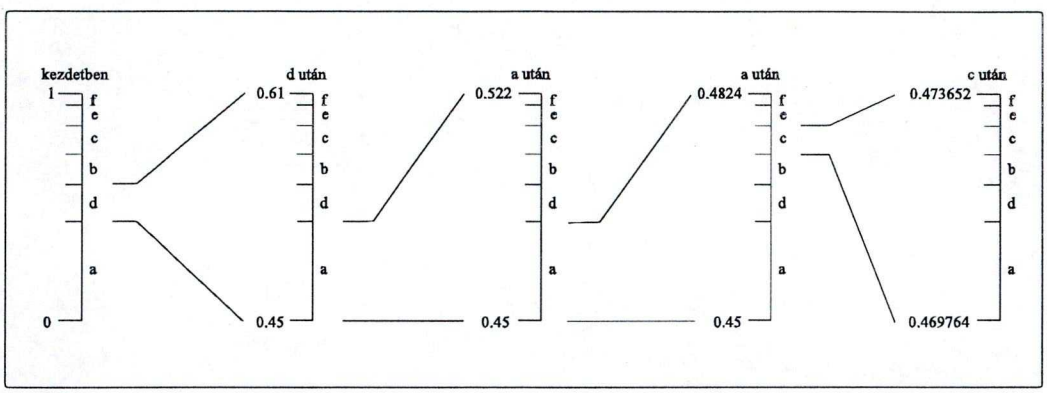
Ennél a kódolásnál az input szöveget egy 0 és 1 közötti számmal reprezentáljuk. A módszer legnagyobb problémája, hogy amint a szöveg hossza növekszik, a hozzá tartozó szám egyre kisebbé válik, ezáltal egyre több bit szükséges az ábrázolásához. Az egymást követő szimbólumok az előző intervallumot a valószínűségeknek megfelelően osztják tovább. A kevésbé valószínű szimbólumok jobban csökkentik az intervallumot, ezáltal több bitet adva az ábrázoláshoz, míg a valószínűbb szimbólumok kevésbé csökkentik az intervallumot.
Az eljárás menete a következő:

- Legyen a kezdő intervallum a $[0, 1)$.
- Osszuk fel a megfelelő intervallumot az egyes szimbólumok valószínűségeinek arányában.
- Vegyük a szöveg következő szimbólumát, és tekintsük új intervallumként a hozzá tartozó részintervallumot.
- Ismételjük az előző két lépést, amíg a szöveg végére nem érünk. A szöveg reprezentálására a végső intervallum egy tetszőleges eleme alkalmas.

Tekintsük ismét az előzőekben említett példát. Tegyük fel, a kódolandó karaktersorozat *daac*. Az intervallumok az alábbiak szerint alakulnak:

kezdetben	$[0, 1)$
d után	$[0.45, 0.61)$
a után	$[0.45, 0.522)$
a után	$[0.45, 0.4824)$
c után	$[0.469764, 0.473652)$

A következő ábra a kódolási eljárás egy másik reprezentációját mutatja:



3. ábra

Az aritmetikai kódolás implementációjánál általában egész aritmetikát szokás használni. A túlcsordulások elkerülése végett a kódoláshoz már nem szükséges biteket egyből továbbítani kell az outputra. Az aritmetikai kódolás egy C nyelvű implementációja megtalálható [2]-ben.

Dolgozatunk további részében a szöveghelyettesítésen alapuló eljárásokat tárgyaljuk.

2.4. Szöveghelyettesítésen alapuló algoritmusok

A szöveghelyettesítésen alapuló kódolás, vagy más néven szótárkódolás 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 dolgozat következő fejezetében ezekkel az eljárásokkal részletesen foglakozunk.

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 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]. Azóta is az egyik legismertebb tömörítő eljárás, amelynek nagyon sok változata keletkezett. Lássuk most az eredeti, LZ77-nek nevezett eljárást részleteiben.

Az LZ77 algoritmus [25]:

Az LZ77 eljárásnál a mutatók a kódolandó szövegrésznek az őt megelőző fix méretű "ablakban" történő előfordulására mutatnak. Egy L_S paraméter jelzi a maximális rész hosszát, ami helyettesíthető mutatóval. Ez lehetővé teszi, hogy az eljárást egy n karakterből álló "ablakban" folytassuk, ami mindig a kódolandó szöveg egy részét tartalmazza. Ebből $n - L_S$ már kódolt, a maradék alkotja a vizsgálandó buffert. A következő lépésnél az "ablakban" megkeressük a leghosszabb szövegrészt, amely megegyezik a már kódolt szöveg egy részével. A két illeszkedő sorozat között lehet átfedés is, de nem egyezhetnek meg teljesen. Ezt azután egy $\langle i, j, a \rangle$ hármassal kódoljuk, ahol i az indexe a megtalált karakter-

sorozatnak a bufferben, j a hossza, míg a az első olyan karakter, ami már nem egyezik meg a lekódolt sorozattal. Ezután az "ablakot" jobbra toljuk a szövegen $j+1$ karakterrel, és az eljárást folytatjuk tovább. Az a karakter hozzáillesztése (i,j)-hez biztosítja, hogy a módszer akkor is működik, ha nincs illeszkedés.

Következzék ezután a módszer formális leírása. Először bevezetünk néhány jelölést.

Legyen n az alkalmazott buffer hossza, A az alap ábécé, és S a kódolandó karaktersorozat. Jelölje L_S a már említett paramétert, és legyen $L_C = 1 + \lceil \log(n - L_S) \rceil + \lceil \log(L_S) \rceil$, ahol a logaritmus alapja $|A|$. Itt L_C a kódok fix hosszát jelenti, amelyek szintén az A ábécéből képződnek. Legyen $S(1,j)$ az S karaktersorozat valódi kezdőszelete, és legyen $i, 1 \leq i \leq j$ adott egész szám. Legyen továbbá $L(i) = \max\{l : S(i, i+l-1) = S(j+1, j+l)\}$, és $L(p) = \max_{1 \leq i \leq j} L(i)$. Az $S(j, j+L(p))$ karaktersorozatot az $S(1,j)$ prefix S-re való helyettesíthető kiterjesztésének nevezzük.
Például $S = 00101011, j = 3$ esetén $L(1) = 1, L(2) = 4, L(3) = 0$. Így $S(3+1, 3+4) = 0101$ az $S(1,3)$ helyettesíthető kiterjesztése S-re $p = 2$-vel.
Ezek után az LZ77 algoritmus a következő:

- Legyen $B_1 = 0^{n-L_s} S(1, L_S)$, ahol $0^{n-L_s} n - L_S$ darab 0-t jelent, legyen továbbá $i = 1$.

- Tekintsük a $B_i, i \geq 1$ buffert, és legyen $S_i = B_i(n - L_S + 1, n - L_S + l_i)$, ahol S_i-nek az $l_i - 1$ hosszú prefixe a $B_i(1, n - L_S)$ prefix $B_i(1, n-1)$-re való helyettesíthető kiterjesztése.

- Ha p_i az S_i meghatározásánál használt mutató, akkor az S_i-hez tartozó C_i kód $C_i = C_{i1} C_{i2} C_{i3}$, ahol C_{i1} a $p_i - 1$, C_{i2} az $l_i - 1$ szám $|A|$ számrendszerbeli előállítása, míg C_{i3} az S_i utolsó szimbóluma.

- Módosítsuk a B_i buffer tartalmát úgy, hogy elhagyjuk az első l_i karaktert, majd a bufferbe töltjük a következő l_i darabot. Növeljük i értékét 1-gyel,

és folytassuk az algoritmust a második lépéstől.

Példa:

Tekintsük az alábbi hármas számrendszerbeli sorozatot. ($|A| = 3$)

$S = 001010210210212021021200.$

$L_S = 9, n = 18 \Rightarrow L_C = 1 + \log_3(18 - 9) + \log_3 9 = 5.$

$$B_1 = 000000000|001010210 \quad p_1 = 9, l_1 = 3 \quad C_1 = 22|02|1$$
$$B_2 = 000000001|010210210 \quad C_2 = 21|10|2$$
$$B_3 = 000010102|102102120 \quad C_3 = 20|21|2$$

Ziv és Lempel bebizonyította [25], hogy az LZ77 eljárással legalább olyan jó kódolási eredményt kapunk, mint bármely más, speciálisan a szöveghez illesztett szótár segítségével, ha a buffer hossza elég nagy. A legnagyobb problémát az jelenti, hogy ha a buffer hosszát megnöveljük, akkor a keresés lassúvá válik, így a tömörítési folyamat nem lesz elég hatékony. Ezt a problémát megfelelő adatstruktúrák használatával küszöbölhetjük ki, ekkor azonban megnövekedhet a tárigény. Erről részletesebb információk találhatók [2]-ben.

3. fejezet

Statikus szótárkódoló algoritmusok elemzése

3.1. Bevezetés és definíciók

Ahogy az előzőekben láttuk, egy adott karaktersorozat tömörítésének egy lehetséges módja a *szótár* segítségével történő tömörítés. 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 továbbiakban csak olyan eljárásokat fogunk tekinteni, amelyek *statikus szótárakat* használnak.

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 (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 olyan karakterből áll, amelyek megegyeznek az eredeti karaktersorozatban az $i+1, \dots, i+d$ pozíciókon álló karakterekkel. 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.

Tekintsünk most egy példát a fentiek illusztrálására. Vegyük az

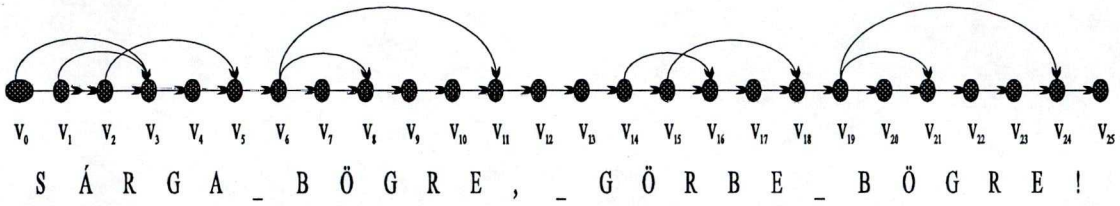
$$S = \text{SÁRGA\_BÖGRE,\_GÖRBE\_BÖGRE!}$$

karaktersorozatot és a következő szótárat:

forrásszó	A	Á	B	E	G	Ö	R	S	_
kódszó	a	b	c	d	e	f	g	h	i
súly	x	x	x	x	x	x	x	x	x

forrásszó	,	!	ÁR	BÖ	ÖR	RBE	RGA	SÁR	BÖGRE
kódszó	j	k	l	m	n	o	p	q	r
súly	x	x	x	x	x	x	x	x	x

Ebben az esetben a megfelelő gráf



4. ábra

Látható, hogy a legrövidebb út v_0-ból v_{25}-be például a

$$v_0v_3v_4v_5v_6v_{11}v_{12}v_{13}v_{14}v_{15}v_{18}v_{19}v_{24}v_{25}$$

út, feltéve, hogy az egyes élek súlya – ahogy a szótár definíciójánál is látható – azonos.

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 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 az algoritmus 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 optimális megoldást. Ha szeretnénk meghatározni, mennyire lehet rossz egy heurisztika, a legkézenfekvőbb módszer, hogy összehasonlítjuk az optimum által kapott eredménnyel. Az összehasonlítás történhet átlagos–eset analízissel, vagy pedig *legrosszabb–eset vizsgálattal.* Dolgozatunkban ez utóbbival foglalkozunk részletesebben.

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:

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

$$cmin(D) = \min\{\|c_i\| \; i = 1, \ldots, k\}$$
$$cmax(D) = \max\{\|c_i\| \; i = 1, \ldots, k\},$$

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 $lmax \geq 2$.

Mint látható, a fenti definícióknál feltételeztünk néhány, a gyakorlati életben megszokott dolgot. Ilyen például az, hogy a kód ábécé bináris, vagy hogy a forrás ábécé karaktereit binárisan kódoljuk valahány biten. A tételek bizonyítása során ezeket a feltevéseket nem használjuk ki, így azok általánosabb definíciók esetén is igazak. (Például $\|c_i\|$ tekinthető általánosan a c_i kódszó súlyának.) Viszont ezzel is szerettünk volna utalni arra, hogy a tételek gyakorlati problémákból erednek.

Nem meglepő, hogy egy heurisztika legrosszabb-eset viselkedése erősen függ az adott szótár tulajdonságaitól. Dolgozatunkban a következő típusú szótárakat fogjuk vizsgálni:

Egy szótárt *általánosnak* nevezünk, ha az input ábécé minden szimbólumát tartalmazza mint forrásszót (ez biztosítja, hogy a heurisztika minden forrásszóvegre el fogja érni az adott gráf nyelőjét, és ezzel 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 következő részben rátérünk a dolgozat lényegi részére és bemutatjuk a legismertebb heurisztikus algoritmusokat, illetve elemezzük ezek legrosszabb–eset viselkedését különböző típusú szótárakra.

3.2. A leghosszabb illesztés módszere (Longest Matching)

A *leghosszabb illesztés(Longest Matching*, továbbiakban *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 és éles korlátokat bizonyított ezen tulajdonságok minden lehetséges kombinációjára. A legérdekesebb tételek a következők.

3.2.1. TÉTEL. (KATAJAINEN-RAITA [17]) *Tetszőleges D általános szótárra*

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

□

Ebből a tételből is látható, hogy az általános esetben az LM heurisztika meglehetősen rossz eredményt is szolgáltathat. Közvetlen következményként adódik az alábbi tétel.

3.2.2. TÉTEL. (KATAJAINEN-RAITA [17]) *Legyen D egy* egyenlő kódhosszúságú *szótár. Ekkor*

$$R_{LM}(D) = lmax - 1.$$

□

Felmerül a kérdés, vajon létezik–e olyan szótártípus, amelyre jobb eredményt kapunk. Erről szólnak a következő tételek.

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

$$R_{LM}(D) = \frac{cmax}{cmin}.$$

□

3.2.4. TÉTEL. (KATAJAINEN-RAITA [17]) Egyenlő kódhosszúságú suffix *szótárra az LM algoritmus által kapott kód optimális bármilyen karaktersorozatra.*

□

3.2.5. TÉTEL. (KATAJAINEN-RAITA [17]) *Legyen D egy* nemhosszító *szótár és S egy karaktersorozat. Ekkor az aszimptotikus legrosszabb–eset hányadosra igaz*

a következő:

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

□

KATAJAINEN és RAITA [17] sejtése az volt, hogy a prefix szótárakra vonatkozó tömörítési eredmények gyengébbek mint a suffix szótárakra bizonyítottak, ezt azonban nem sikerült igazolniuk. Mint azt látni fogjuk, sejtésük igaznak bizonyult.

Dolgozatunkban éles legrosszabb eset korlátokat fogunk bizonyítani *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.

BIZONYÍTÁS. A felső korlát közvetlenül adódik a 3.2.1. tételből. Az alsó korlát helyességét a következő, az $\{u, v, w\}$ 3-elemű ábécé betűiből képzett prefix szótár segítségével igazolhatjuk:

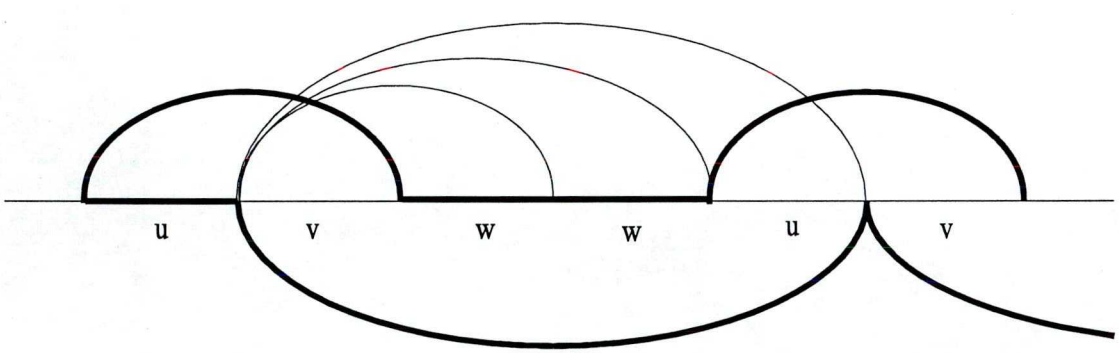
forrásszó	u	v	w	uv	vw^j $j=1,\ldots,lmax-2$	$vw^{lmax-2}u$
kódszó	a	b	c	d	e_j	f
súly	$cmax$	$cmax$	$cmax$	$cmax$	$cmax$	$cmin$

Legyen $i > 0$, és tekintsük az $S_i = u(vw^{lmax-2}u)^i$, $i(lmax+1)+1$ hosszú karaktersorozatokat. A hozzátartozó gráfot megvizsgálva ellenőrizhető, hogy $OPT(D, S_i) = af^i$ és $LM(D, S_i) = (dc^{lmax-2})^i a$. Ekkor

$$R_{LM}(D) \geq \lim_{n\to\infty} \frac{\|LM(D,S_i)\|}{\|OPT(D,S_i)\|} = \lim_{i\to\infty} \frac{i(lmax-1)cmax + cmax}{i \cdot cmin + cmax} = \frac{(lmax-1)cmax}{cmin},$$

teljesül LM-re.

□



5. ábra

A 3.2.6. tételben definiált S_i illusztrációja $lmax = 4$ esetén. Az optimális út a vizszintes vonal alatt, míg az LM-út a vonal felett halad.

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.

BIZONYÍTÁS. A felső korlát következik a 3.2.5. tételből.

Az alsó korlátot igazoló példához vegyük a következő szótárat, az egyes esetekre különböző súlyokat alkalmazva:

forrásszó	u	v	w	uv	vw^j $j=1,\ldots,lmax-2$	$vw^{lmax-2}u$
kódszó	a	b	c	d	e_j	f
súly($cmax \leq Bt$)	$cmin$	$cmax$	$cmax$	$cmax$	$cmax$	$cmin$
súly($Bt < cmax < 2Bt$)	$cmin$	Bt	Bt	$cmax$	$cmax$	$cmin$
súly($2Bt \leq cmax$)	$cmin$	Bt	Bt	$2Bt$	q	$cmin$

ahol $q = \min(cmax, (j+1)\,Bt)$.

Ebből a kívánt eredmény egyszerű számolással adódik. Például a $2Bt \leq cmax$ esetben

$$R_{LM}(D) \geq \lim_{n\to\infty} \frac{\|LM(D,S_i)\|}{\|OPT(D,S_i)\|} = \lim_{i\to\infty} \frac{i \cdot lmax \cdot Bt + cmin}{(i+1)cmin} = \frac{lmax \cdot Bt}{cmin}.$$

□

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.

BIZONYÍTÁS. A felső korlát közvetlenül adódik a 3.2.1. tételből. Legyen $cmin = cmax$ a 3.2.6. tétel bizonyításában, és ekkor az alsó korlát is adódik.

□

Ellenőrizhető, hogy a fenti korlát igaz *prefix, nemhosszító* és *egyenlő kódhosszúságú* szótárakra is.

A következő táblázat összefoglalja a leghosszabb illesztés módszerére vonatkozó eredményeket:

D szótár				
Prefix	Suffix	Egyenlő kódhosszúságú	Nemhosszító	$R_{LM}(D)$
x	–	–	–	$\frac{(lmax-1)cmax}{cmin}$
x	–	x	–	$lmax-1$
x	–	–	x	$\frac{lmax \cdot Bt}{cmin}$
x	–	x	x	$lmax-1$
–	x	–	–	$\frac{cmax}{cmin}$
–	x	x	–	1
–	x	–	x	$\frac{cmax}{cmin}$
–	x	x	x	1
–	–	–	–	$\frac{(lmax-1)cmax}{cmin}$
–	–	x	–	$lmax-1$
–	–	–	x	$\frac{lmax \cdot Bt}{cmin}$ $(cmax \geq 2Bt)$
–	–	x	x	$lmax-1$

3.3. A különbségen alapuló greedy algoritmus

A GONZALEZ–SMITH és STORER [13] által definiált *greedy* heurisztika, amelyet *különbségen alapuló* (*differential greedy*, 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.

Nyilvánvaló, hogy bár a heurisztika lokálisan optimális, globálisan igen gyenge eredményt is adhat. Másrészt elképzelhető, hogy a greedy heurisztika optimális eredményt ad olyan inputra, amelyre a leghosszabb illesztés módszere a lehető legrosszabb kódolást hozza létre. A dolog fordítva is igaz lehet. Intuitív módon nyilvánvalónak tűnik, hogy a DG heurisztika az LM heurisztika javított változata. Azokban az esetekben, amikor a két algoritmus megegyezik, alkalmazhatjuk az LM heurisztikára az előzőekben igazolt korlátokat. A DG heurisztikát KATAJAINEN és RAITA vizsgálta néhány szótártípusra, és a következő eredményeket kapta:

3.3.1. TÉTEL. (KATAJAINEN-RAITA [17]) *A DG és az LM heurisztikák megegyező kódolási eredményt adnak, azaz, $DG(D, S) = LM(D, S)$, tetszőleges* egyenlő kódhosszúságú *D szótárra és S karaktersorozatra.*

□

3.3.2. TÉTEL. (KATAJAINEN-RAITA [17]) *Legyen D egy* általános *szótár. Ekkor*

$$R_{DG}(D) \leq \begin{cases} \dfrac{cmin + (lmax-1)cmax}{cmin + (lmax-1)Bt} & ha\ (lmax-1)^2 cmax \cdot Bt \leq cmin^2 \\ & és\ \left\lfloor \frac{cmax-cmin}{Bt} \right\rfloor \geq lmax - 1 \\ \dfrac{(lmax-1)cmax}{cmin} & különben \end{cases} \tag{3.1}$$

és a fenti korlátok élesek.

□

3.3.3. TÉTEL. (KATAJAINEN-RAITA [17]) *Legyen D egy* 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.

□

Prefix szótárt a fenti szerzők nem vizsgáltak.

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.

BIZONYÍTÁS. A 3.3.2. tétel miatt csupán azt kell igazolnunk, hogy a fenti korlátok elérhetők prefix szótárakkal. Két esetet különböztetünk meg:
A eset: Tegyük fel hogy $(lmax-1)^2 cmax \cdot Bt \leq cmin^2$ és $\left\lfloor \frac{cmax-cmin}{Bt} \right\rfloor \geq lmax-1$. A következő D szótárat alkalmazhatjuk:

forrásszó	u	v	uv^j $j=1,\ldots,lmax-1$
kódszó	a	b	c_j
súly	$cmin$	$cmax$	$cmin + jBt$

Legyen $i > 0$, és tekintsük az $S_i = (uv^{lmax-1})^i$, $i \cdot lmax$ hosszú karaktersorozatokat. Látható, hogy $OPT(D, S_i) = c_{lmax-1}^i$ és $LM(D, S_i) = (ab^{lmax-1})^i$. Ekkor

$$R_{DG}(D) \geq \lim_{n\to\infty} \frac{\|DG(D,S_i)\|}{\|OPT(D,S_i)\|} = \lim_{i\to\infty} \frac{i(cmin + (lmax-1)cmax)}{i(cmin + (lmax-1)Bt)} = \frac{cmin + (lmax-1)cmax}{cmin + (lmax-1)Bt},$$

teljesül DG-re. Megjegyezzük, hogy a DG algoritmus a fenti szótárra a feltételtől függetlenül mindig ezt az eredményt adja, azonban ez a korlát csak a feltétel teljesülése esetén lesz nagyobb a B esetben vizsgált korlátnál.
B eset: Tegyük fel, hogy $(lmax-1)^2 cmax \cdot Bt > cmin^2$ vagy $\left\lfloor \frac{cmax-cmin}{Bt} \right\rfloor < lmax-1$. Ebben az esetben a 3.2.6. tétel bizonyításában használt szótár adja a kívánt eredményt.

□

3.3.5. TÉTEL. [3] *Legyen D egy* prefix *és* nemhosszító *szótár. Ekkor*

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

és a fenti korlátok élesek.

BIZONYÍTÁS. A felső korlát a 3.3.3. tételből következik. Ahhoz, hogy belássuk a korlát élességét, ugyanazon szótárt és súlyokat használhatjuk, mint a 3.2.7. tétel esetén.

□

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. [3] *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*

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

BIZONYÍTÁS. Két esetet különböztetünk meg:

A eset: Ha $lmax \cdot Bt \leq 2cmax - Bt$, akkor tekintsük a következő szótárat:

forrásszó	u	w^j $_{j=1,\ldots,lmax-1}$	uw	$w^j u$ $_{j=1,\ldots,lmax-2}$	$w^{lmax-1}u$
kódszó	a	b_j	c	d_j	e
súly	Bt	jBt	$2Bt$	$(j+1)Bt$	$cmin$

Legyen $i \geq 1$, definiáljuk az $S_i = u(w^{lmax-1}u)^i$, $i \cdot lmax + 1$ hosszú karaktersorozatokat. Ekkor kapjuk, hogy $DG(D, S_i) = (cb_{lmax-2})^i a$ és $OPT(D, S_i) = ae^i$. A legrosszabb–eset hányadost kiszámítva,

$$R_{DG}(D) \geq \lim_{n\to\infty} \frac{\|DG(D, S_i)\|}{\|OPT(D, S_i)\|} = \lim_{i\to\infty} \frac{i \cdot lmax \cdot Bt + Bt}{Bt + i \cdot cmin} = \frac{lmax \cdot Bt}{cmin}.$$

B eset: Ha $lmax \cdot Bt \geq 2cmax - Bt + 1$, tegyük fel, hogy $cmax = \alpha Bt$, ahol $\alpha = 1, ..., lmax$. Ebből következik, hogy $2\alpha - 1 < lmax$. Vegyük a következő szótárat:

forrásszó	u^j $j=1,..,\alpha-1$	u^α	uw^j $j=1,..,\alpha-1$	w^j $j=1,..,\alpha-2$
kódszó	a_j	b	c_j	d_j
súly	Bt	$2Bt$	$(j+1)Bt$	jBt

forrásszó	$w^{\alpha-1}$	$w^{2(\alpha-1)}u$	$w^j u$ $j=1,..,2\alpha-3$ $j\neq\alpha-1$	$w^{\alpha-1}u$
kódszó	e	f	g_j	h
súly	$cmax - Bt$	$cmin$	$cmin$	$cmax$

$S_i = u^\alpha(w^{2(\alpha-1)}u)^i$ esetén kapjuk, hogy $OPT(D, S_i) = bf^i$ és $DG(D, S_i) = a_{\alpha-1}(c_{\alpha-1}e)^i a_1$.

Vannak olyan pontok, ahol a DG által választott él nem egyértelmű. Amikor DG $u^{\alpha-1}$-t választja, egy másik lehetséges jelölt is van, mégpedig u^α. A két különbség, amelyet a DG kiszámít $(\alpha-1)Bt - Bt$ illetve $\alpha Bt - 2Bt$. Feltételezzük azonban, hogy a DG $u^{\alpha-1}$-t választja. Hasonlóan az algoritmus $uw^{\alpha-1}$-t választja u és $w^{\alpha-1}$-t $w^{\alpha-1}u$ helyett. Ebben az esetben a legrosszabb–eset hányadosra kapjuk,

hogy

$$\begin{aligned} R_{DG}(D) &\geq \lim_{n\to\infty} \frac{\|DG(D,S_i)\|}{\|OPT(D,S_i)\|} \\ &= \lim_{i\to\infty} \frac{i(2cmax - Bt) + 2Bt}{i \cdot cmin + 2Bt} \\ &= \frac{2cmax - Bt}{cmin}. \end{aligned}$$

□

A *DG* heurisztika *suffix* szótárakra történő elemzését illetően, a szerzők megemlítik [17], hogy a heurisztika viselkedését ebben az esetben meglehetősen nehéz analizálni. Az egyetlen ismert felső korlát *azonos volt az általános szótárakra* vonatkozóval:

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

$$R_{DG}(D) \leq \begin{cases} \dfrac{cmin + (lmax-1)cmax}{cmin + (lmax-1)Bt} & \text{ha } (lmax-1)^2\ cmax\ Bt < cmin^2 \text{ és} \\ & \lfloor (cmax - cmin)/Bt \rfloor \geq lmax - 1 \\ \dfrac{(lmax-1)cmax}{cmin} & \text{különben.} \end{cases}$$

□

Ebben az esetben éles korlát nem volt ismert. GALAMBOS GÁBORRAL, ULRICH PFERSCHYVEL és GERHARD WOEGINGERREL közösen sikerült éles korlátokat bizonyítanunk a *DG* heurisztika legrosszabb–eset viselkedésére suffix szótárak esetén. (Vegyük észre, hogy ha $lmax = 2$, akkor minden szótár prefix tulajdonságú.)

3.3.9. TÉTEL. [3] *Legyen D egy* suffix *szótár és tegyük fel hogy $lmax \geq 3$. Ekkor*

$$R_{DG}(D) \leq \begin{cases} \dfrac{2cmax - Bt}{cmin} & ha\ cmax \leq 3/2\ Bt \\ \dfrac{(2cmax + Bt)^2}{8Bt \cdot cmin} & ha\ 3/2\ Bt < cmax \leq (lmax - 3/2)Bt \\ L & ha\ (lmax - 3/2)Bt < cmax, \end{cases} \tag{3.2}$$

ahol

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

A fenti korlátok élesek.

BIZONYÍTÁS. Legyen $N = (V, A)$ az S karaktersorozatból képzett irányított gráf, és legyen D a bevezetésben megadott módon definiálva. Legyen v_i és v_j két egymásutáni *vágóéle* N-nek. Ez maga után vonja, hogy mindketten rajta vannak mind az optimum, mind a DG által megadott úton is.

Először a felső korlátot fogjuk bebizonyítani $R_{DG}(D)$-re vonatkozóan. Feltételezhetjük, hogy az optimumnak csupán egyetlen maximális, azaz $lmax$ hosszú éle van v_i-ből v_j-be mimimális, azaz $cmin$ súllyal. Bevezetjük a következő jelöléseket. Feltételezzük, hogy a DG–út a $(v_i, v_{i_1}, v_{i_2}, \ldots, v_{i_{k+1}}, v_j)$ sorozatból áll. A $(v_{i_p}, v_{i_{p+1}})$ él hosszát, illetve súlyát t_p illetve c_p jelöli, $1 \leq p \leq k$.

Mivel a szótár prefix tulajdonságú, a v_{i_p} csúcsban a DG algoritmus a $(v_{i_p}, v_{i_{p+1}})$ és a (v_{i_p}, v_j) élek közül fog választani. Az utóbbi súlya legfeljebb $cmax$. Mivel a DG heurisztika a v_{i_p} csúcson átmenő utat választ, ezért

$$t_p Bt - c_p \geq \left(\sum_{l=p}^{k} t_l + 1 \right) Bt - cmax$$

teljesül. Ezt minden p-re összegezve kapjuk, hogy

$$\sum_{p=1}^{k} c_p \leq k\ cmax + Bt \sum_{p=1}^{k} t_p - Bt \sum_{p=1}^{k} \sum_{l=p}^{k} t_l - k\ Bt. \tag{3.3}$$

Először a jobb oldal harmadik tagját fogjuk becsülni.

$$\sum_{p=1}^{k}\sum_{l=p}^{k} t_l = \sum_{p=1}^{k} pt_p. \tag{3.4}$$

Vezessük be a $T = \sum_{l=1}^{k} t_l + 1$ jelölést. Nyilvánvaló, hogy

$$T \leq lmax - 1.$$

Ellenőrithető, hogy (3.4) a minimumát pontosan akkor éri el, ha $t_1 = T - k$ és $t_p = 1,\ p = 2, \ldots, k$. Ebből kapjuk, hogy

$$\min_{t_l} \sum_{p=1}^{k}\sum_{l=p}^{k} t_l = (T-k) + \sum_{i=2}^{k} i = T + \frac{(k+1)(k-2)}{2}$$

Ezt behelyettesítve (3.3)-ba adódik, hogy

$$\begin{aligned} \sum_{p=1}^{k} c_p &\leq \max_{1\leq k\leq lmax-2} \left\{ k\, cmax - Bt(k+1) - Bt\frac{(k+1)(k-2)}{2} \right\} \\ &= \frac{1}{2} \max_{1\leq k\leq lmax-2} \{2k\, cmax - (k^2+k)Bt\}. \end{aligned} \tag{3.5}$$

A jobboldali kifejezés konkáv függvénye k-nak és maximumát $k = \frac{2cmax-Bt}{2Bt}$-nél veszi fel. Ezután három esetet különböztetünk meg:

1. Ha $cmax \leq \frac{3}{2}Bt$, a maximum $k = 1$-re adódik.
2. Ha $\frac{3}{2}Bt < cmax \leq (lmax - \frac{3}{2})Bt$, a maximum $k = \frac{2cmax-Bt}{2Bt}$-re adódik.
3. Ha $(lmax - \frac{3}{2})Bt < cmax$, a maximum $k = lmax - 2$-re adódik.

A megfelelő k értékeket (3.5) jobb oldalába helyettesítve, és a végső v_j-n áthaladó csúcshoz $cmax$ nagyságú súlyt rendelve kapjuk a kívánt felső korlátot.

A bizonyítás második részében azt fogjuk belátni, hogy a fenti korlátok valóban élesek.

A eset: $cmax \leq \frac{3}{2}Bt$ esetén tekintsük a következő szótárat.

forrásszó	u	w	uw	$w^j u$ $j=1,\ldots,lmax-2$	$w^{lmax-1}u$	w^j $j=1,\ldots,lmax-1$
kód	a	b	c	d_j	d_{lmax-1}	e_j
súly	$cmax - Bt$	$cmax$	$cmax$	$cmax$	$cmin$	$cmax - Bt$

Az $S_i = u(w^{lmax-1}u)^i$, i $lmax+1$ hosszú karaktersorozatokat tömörítjük. Mivel $OPT(D,S_i) = ad^i_{lmax-1}$ és $DG(D,S_i) = (ce_{lmax-2})^i a$,

$$\begin{aligned} R_{DG}(D) &\geq \lim_{n\to\infty} \frac{\|DG(D,S_i)\|}{\|OPT(D,S_i)\|} \\ &= \lim_{i\to\infty} \frac{i(2cmax - Bt) + cmax - Bt}{(cmax - Bt) + i \cdot cmin} = \frac{2\,cmax - Bt}{cmin}. \end{aligned}$$

B eset: Ha $\frac{3}{2}Bt < cmax \leq (lmax - \frac{3}{2})Bt$, feltételezzük, hogy Bt páratlan, és $cmin \leq \frac{Bt}{2}$ továbbá $cmax = (2\alpha+1)\frac{Bt}{2}$ valamely α, $1 \leq \alpha \leq lmax - 4$ egész számra. Egy $lmax$ betűből álló ábécéből képzett szótárat tekintünk. Az ábécé betűit $u, v, w_1, \ldots, w_{lmax-2}$-vel jelöljük. Legyen $k = \frac{2cmax-Bt}{2Bt}$

forrásszó	u	uv	v^j $j=1,\ldots,lmax-2-k$	$v^{lmax-1-k}$
kód	a	b	c	d_0
súly	$cmax$	$cmax$	$cmax$	$\frac{1}{2}Bt$

forrásszó	w_j $j=1,\ldots,k-1$	$v^j w_1 \ldots w_{k-1}u$ $j=1,\ldots,lmax-1-k$	$v^{lmax-k}w_1 \ldots w_{k-1}u$	$w_j \ldots w_{k-1}u$ $j=1,\ldots,lmax-1$
kód	d_j	e_j	e_{lmax-k}	f_j
súly	$(2j+1)\frac{Bt}{2}$	$cmax$	$cmin$	$cmax$

és a kódolandó karaktersorozat $S_i = u(v^{lmax-k}w_1 \ldots w_{k-1}u)^i$. Ellenőrizhető, hogy $OPT(D,S_i) = ae^i_{lmax-k}$. A DG–utat tekintve látható, hogy először az uv élt választja az algoritmus. Ezen él végénél több él is található. Ezek közül egy átlépi

a v karaktert, a többi pedig a $v^j, : j = 1, \ldots, lmax - 1 - k$ karaktersorozatokat. Ezért a DG a $v^{lmax-1-k}$ élt választja. A karaktersorozat maradék részén (amíg ismét az u karakterher ér) a DG algoritmusnak a $w_j \ldots w_{k-1}u, : j = 1, \ldots, k-1$ élek, és a w_j karakter között kell dönteni. Az egyenlőséget az algoritmus úgy oldja fel, hogy minden egyes esetben a w_j-t választja. Ily módon kapjuk, hogy $DG(D, S_i) = (bd_0d_1 \ldots d_{k-1})^i a$, amiből

$$\begin{aligned} R_{DG}(D) &\geq \lim_{n\to\infty} \frac{\|DG(D,S_i)\|}{\|OPT(D,S_i)\|} \\ &= \lim_{i\to\infty} \frac{i\left(cmax + \frac{Bt}{2}\sum_{j=0}^{k-1}(2j+1)\right) + cmax}{cmax + i \cdot cmin} \\ &= \lim_{i\to\infty} \frac{i\left(cmax + \frac{Bt}{2}\frac{(2cmax-Bt)^2}{4Bt^2}\right) + cmax}{cmax + i \cdot cmin} \\ &= \frac{(2cmax + Bt)^2}{8Bt \cdot cmin} \end{aligned}$$

C eset: Végül $(lmax - \frac{3}{2})Bt < cmax$ esetén ismét egy $lmax$ betűből képzett szótárak tekintünk, a betűk jele ezúttal is $u, w_1, \ldots, w_{lmax-1}$.

forrásszó	u	w_j $j=1,\ldots,lmax-1$	uw_{lmax-1}	$w_j \ldots w_1 u$ $j=lmax-1,\ldots,1$	$w_{lmax-1} \ldots w_1 u$
kódsszó	a	b_j	c	d_j	e
súly	$cmax$	$cmax - j\dot{B}t$	$cmax$	$cmax$	$cmin$

Az $S_i = u(w_{lmax-1} \ldots w_1 u)^i$,$i \cdot lmax + 1$ hosszú karaktersorozatokra kapjuk, hogy $OPT(D, S_i) = ae^i$ és $DG(D, S_i) = (cw_{lmax-2}w_{lmax-3} \ldots w_1)^i a$. Így

$$\begin{aligned} R_{DG}(D) &\geq \lim_{n\to\infty} \frac{\|DG(D,S_i)\|}{\|OPT(D,S_i)\|} \\ &= \lim_{i\to\infty} \frac{i\left(cmax + \sum_{j=1}^{lmax-2}(cmax - j\dot{B}t)\right) + cmax}{cmax + i \cdot cmin} \\ &= \lim_{i\to\infty} \frac{i(lmax-1)(2cmax - (lmax-2)Bt) + cmax}{2cmax + 2i \cdot cmin} \\ &= \frac{(lmax-1)(2cmax - (lmax-2)Bt)}{2cmin}, \end{aligned}$$

amiből a bizonyítandó állítás következik.

□

Végül foglaljuk össze ismét egy táblázatban a különbségen alapuló greedy algoritmusra vonatkozó eredményeket:

D szótár				
Prefix	Suffix	Egyenlő kódhosszúságú	Nemhosszitó	$R_{DG}(D)$
x	–	–	–	lásd 3.1 formula
x	–	x	–	$lmax - 1$
x	–	–	x	$\frac{lmax \cdot Bt}{cmin}$
x	–	x	x	$lmax - 1$
–	x	–	–	lásd 3.2 formula
–	x	x	–	1
–	x	–	x	$\frac{\min\{lmax \cdot Bt, 2cmax - Bt\}}{cmin}$
–	x	x	x	1
–	–	–	–	lásd 3.1 formula
–	–	x	–	$lmax - 1$
–	–	–	x	$\frac{lmax \cdot Bt}{cmin}$ $(cmax \geq 2Bt)$
–	–	x	x	$lmax - 1$

3.4. A hányadoson alapuló greedy algoritmus

Az előző fejezetekben láthattuk, hogy az LM–heurisztika egyáltalán nem foglalkozik a kódszavak hosszával, míg a DG–algoritmus az abszolút különbséget próbálja maximalizálni, és ezáltal figyelmen kívül hagyja a rövid, de relative jó tömörítési arányt biztosító éleket. Az alábbi eljárás más szempont alapján dönti el, melyik élet választja.

A *hányadoson alapuló greedy algoritmus (fractional greedy*, 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

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

Nyilvánvaló, hogy sok esetben ez a heurisztika jobb eredményt fog adni, mint az LM vagy a DG heurisztika.
Ezt illusztrálja a következő példa (legyen $a^1 = a, a^{i+1} = aa^i, i \in \mathbb{N}$, tetszőleges a karakterre):

Példa:

> Tekintsük a következő nemhosszító szótárat a $cmax = 4$, $cmin = 1$ feltételekkel, és ahogy az ASCII kódolásnál megszokott, legyen $Bt = 8$.

forrásszó	u	v	uv	$v^{lmax-1}u$
kódszó	1∅	11∅1	11∅∅	∅
súly	2	4	4	1

Az $S_i = u(v^{lmax-1}u)^i$ $8(lmax \cdot i + 1)$ bitből álló karaktersorozatokat tömörítve az LM– vagy a DG–algoritmusokkal, mindkét esetben a $(11\emptyset\emptyset(11\emptyset 1)^{lmax-2})^i 1\emptyset$ kódsorozatot kapjuk, amelynek hossza $4(lmax-1)i + 2$ bit. Az FG–heuriszikát alkalmazva ugyanerre az esetre az $1\emptyset(\emptyset)^i$ kódsorozatot kapjuk, ami csupán $i + 2$ bit hosszú.

Bár a példa nagyon speciális szótárral készült, ez is mutatja a hányadoson alapuló greedy algoritmus esetleges előnyeit.

Ebben a fejezetben felső korlátokat fogunk bizonyítani a hányadoson alapuló greedy heurisztika legrosszabb–eset viselkedésére, különböző típusú szótárakat tekintve, ahogy azt a bevezetőben definiáltuk. Belátjuk továbbá, hogy ezek a korlátok élesek abban az értelemben, hogy vannak olyan szótárak illetve karaktersorozatok, amelyek a adott korlátokat elérik. (Suffix szótárakra egy kis eltérés adódik az aszimptotikus technikák miatt.)

3.4.1. TÉTEL. [6] *Legyen D egy* általános *szótár. Ekkor*

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

és a fenti korlát éles.

BIZONYÍTÁS. Legyent S egy karaktersorozat. A felső korlát bizonyításához elegendő S-nek egy olyan $s_{i+1}, ..., s_j$ részsorozatát tömörítenünk, amelynél az FG és az OPT-út közös pontjai a v_i és a v_j csúcsok. Legyen az FG-út $v_i v_p ... v_j$, az OPT-út pedig $v_i v_t ... v_j$. A megfelelő kódsorozatokat $c_0 c_1 ... c_q$-val, illetve $c_0' c_1' ... c_q'$-vel jelöljük. Az v_p és v_t legelső csúcsok relatív helyzetétől függően két esetet különböztetünk meg.

A eset: Tegyük fel, hogy $i < p < t < \min\{i + lmax, j\}$. Mivel az FG algoritmus a legkisebb hányadossal bíró élet választja v_i-ből indulva, teljesül a

$$\frac{(p-i)\,Bt}{\|c_0\|} \geq \frac{(t-i)\,Bt}{\|c_0'\|}$$

egyenlőtlenség.

Ebből következik, hogy

$$\left\|c_0'\right\| \geq \frac{t-i}{p-i} \left\|c_0\right\|.$$

Az élekre vonatkozóan igazak a következők:

$$q + r \leq j - p \text{ and } r \geq \tfrac{j-t}{lmax},$$

amiből adódik, hogy

$$q \leq j - p - r \leq j - p - \frac{j-t}{lmax} = t - p + \frac{(lmax-1)(j-t)}{lmax}.$$

Mivel

$$\frac{x}{y} \geq \frac{x+a}{y+ca}$$

minden $a, c > 0$-ra, ha $xc \geq y > 0$, és

$$\frac{t-p}{p-i} \geq \frac{1}{lmax-1}$$

kapjuk, hogy

$$\begin{aligned} R_{FG}(D) &\leq \frac{\left\|c_0\right\| + q\ cmax}{\left\|c_0'\right\| + r\ cmin} \\ &\leq \frac{\left\|c_0\right\| + (t-p)\ cmax + \frac{(lmax-1)(j-t)cmax}{lmax}}{\left(\frac{t-p}{p-i}+1\right)\left\|c_0\right\| + \frac{(j-t)cmin}{lmax}} \\ &\leq \frac{cmin + (t-p)\ cmax + \frac{(lmax-1)(j-t)cmax}{lmax}}{\frac{lmax}{lmax-1}cmin + \frac{(j-t)cmin}{lmax}}. \end{aligned}$$

Legyen $x = t - p, y = j - t$, a fenti hányadost pedig jelölje $f(x, y)$. Tudjuk, hogy teljesülnek az $1 \leq x \leq lmax - 1$ és az $y \geq 0$ feltételek. Láthatjuk, hogy a hányados maximumát az $(lmax - 1, \infty)$ pontban veszi fel. Ebből adódik a kívánt felső korlát.

B eset: Tegyük fel, hogy $i < t < p < \min\{i + lmax, j\}$. Az FG-út éleire vonatkozóan ismét adódik, hogy

$$q \leq j - t - r \leq \frac{(lmax - 1)(j - t)}{lmax}.$$

Ekkor

$$\begin{aligned} R_{FG}(D) &\leq \frac{\|c_0\| + q\ cmax}{\left\|c_0'\right\| + r\ cmin} \\ &\leq \frac{\|c_0\| + \frac{(lmax-1)(j-t)cmax}{lmax}}{\left\|c_0'\right\| + \frac{(j-t)cmin}{lmax}} \\ &\leq \frac{cmax + \frac{(lmax-1)(j-t)cmax}{lmax}}{cmin + \frac{(j-t)cmin}{lmax}}, \end{aligned}$$

amiből ismét megkapjuk a megfelelő korlátot.

Ezután már csak azt kell bizonyítanunk, hogy az felső korlátot az FG–heurisztika eléri. Tekintsük a következő szótárat:

forrásszó	u	w	u^2	$uw^{lmax-2}u$
kódszó	a	b	c	d
súly	$cmax$	$cmax$	$cmax$	$cmin$

Az $S_i = (u^2 w^{lmax-2})^i$, $i \cdot lmax$ hosszú karaktersorozatokat tömörítjük. Az optimális algoritmus az $OPT(D, S_i) = ad^{i-1}ab^{lmax-2}$ kódszavakat generálja, míg az FG–algoritmus eredménye $FG(D, S_i) = (cb^{lmax-2})^i a$, amiből

$$\begin{aligned} R_{FG}(D) &\geq \limsup_{n\to\infty} \frac{\|FG(D, S_i)\|}{\|OPT(D, S_i)\|} = \lim_{i\to\infty} \frac{i(lmax - 1)cmax + cmax}{lmax \cdot cmax + (i - 1)cmin} \\ &= \frac{(lmax - 1)cmax}{cmin}. \end{aligned}$$

□

3.4.2. TÉTEL. [6] *Legyen D* egyenlő kódhosszúságú *szótár. Ekkor*

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

és a fenti korlát éles.

BIZONYÍTÁS. A $cmax = cmin$ feltételt használva a fenti alsó korlát példában kapjuk a kívánt eredményt.

□

Az előzőekben bizonyított korlátok azt sejtetik, hogy az FG–algoritmus legrosszabb–eset viselkedése azonos az LM– és a DG–heurisztikákra vonatkozóan akkor is, ha más típusú szótárakat alkalmazunk. Ez valóban igaz *nemhosszító* szótárakra.

3.4.3. TÉTEL. [6] *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.

BIZONYÍTÁS. A $cmax$ és a Bt közötti kapcsolat alapján három esetet különböztetünk meg:

A eset: $cmax \leq Bt$:

Ekkor minden szótár nemhosszító, és a 3.4.1. tétel változtatás nélkül alkalmazható erre az esetre (beleértve a legrosszabb–eset példát is).

B eset: $Bt < cmax < 2\,Bt$:

Először vegyük észre, hogy $lmax = 2$ esetén a korlát megegyezik a 3.4.1. tételben bizonyított, általános esetre vonatkozó korláttal. Így a bizonyítás során feltételezzük, hogy $lmax \geq 3$.

A 3.4.1. tételben alkalmazott modellt részletesebben ki kell dolgoznunk. Ismét abból indulunk ki, hogy a hányadoson alapuló greedy algoritmus és az optimális algoritmus összehasonlításához csupán azokat a forrásszavakat kell tekintenünk, amelyeken az FG–út és az OPT–út csúcsfüggetlen és közös végpontjaik vannak.

Ezeket az útvonalakat $P_{FG} = v, v_1, v_2, \ldots, v_q, \bar{v}$ illetve $P_{OPT} = v, v'_1, v'_2, \ldots, v'_r, \bar{v}$-sal jelöljük. Legyen p illetve p' karakter a v és v_1 illetve a v'_1 között és j karakter a v és a $\bar{v}$ között. A (v, v_1) és a (v, v'_1) élekhez tartozó kódszavak legyenek c_0 illetve c'_0.

Amikor az FG algoritmus eléri a v csúcsot a kódolási eljárás során, a következő lépésnél a "legjobb hányadossal bíró" kimenő élet fogja választani, amiből következik, hogy

$$\frac{\|c_0\|}{p \cdot Bt} \leq \frac{\|c'_0\|}{p' \cdot Bt},$$

és így

$$\|c'_0\| \geq \frac{p'}{p}\|c_0\|.$$

Másrészt mivel P_{FG}-nek és P_{OPT}-nak nincsen közös csúcsa, és így minden egyes él "elfogyaszt" legalább egy karaktert, a v és $\bar{v}$ között lévő élekre igaz, hogy

$$q + r + 2 \leq j + 1 - (\min\{p, p'\} - 1)$$

és így

$$q \leq j - r - \min\{p, p'\}.$$

Tudjuk, hogy

$$r \geq \left\lceil \frac{j - p'}{lmax} \right\rceil.$$

Az FG–úton lévő éleket két részre bontjuk. Tartalmazza Q_1 az összes 1 hosszú élet és Q_2 a többit. Ezen halmazok számosságát jelölje $q_1 = |Q_1|$ és $q_2 = q - q_1 = |Q_2|$. Így kapjuk, hogy

$$R_{FG}(D) \leq \frac{\|c_0\| + q_1 Bt + q_2\, cmax}{\|c'_0\| + r \cdot cmin}. \tag{3.6}$$

Két esetet különböztetünk meg:

B.1. eset: $0 < p < p' \leq \min\{j, lmax\}$:

Egy felső korlátot fogunk megadni az FG–úton lévő élek súlyaira vonatkozóan. Nemhosszító szótár esetén a legrosszabb–eset akkor áll fenn, ha minden karaktert Bt bittel kódol az algoritmus. Mivel r csúcs van az optimális algoritmus által

definiált úton, amelyeket az FG–heurisztikának el kell kerülni, ezért r csúcsot mindenképpen át kell lépni Q_2-beli élekkel. Ha $cmax < 2\,Bt$ a lehető legrosszabb eset az, ha $q_2 = r$ és Q_2 minden éle 2 hosszú és $cmax$ súlyú. A fenti korlátokat alkalmazva r-re és q-ra kapjuk, hogy

$$\begin{aligned} q_1 Bt + q_2\, cmax &\leq (j - p - 2r)Bt + r\, cmax \\ &= (j-p)Bt + r(cmax - 2\,Bt) \\ &\leq (j-p)Bt + \frac{j-p'}{lmax}(cmax - 2\,Bt) \\ &= (p'-p)Bt + \frac{j-p'}{lmax}(cmax + (lmax-2)Bt). \end{aligned}$$

Ezt (3.6)-ba helyettesítve, és a $\|c_0'\|$-re és r-re vonatkozó korlátokat használva adódik, hogy

$$R_{FG}(D) \leq \frac{\|c_0\| + (p'-p)Bt + ((lmax-2)Bt + cmax)\frac{j-p'}{lmax}}{\frac{p'}{p}\|c_0\| + (j-p')\frac{cmin}{lmax}}.$$

Ellenőrizhető, hogy az egyenlőtlenség jobb oldala j-nek növekvő függvénye, kivéve a $p' = lmax$ és $p = 1$ esetet. A $j \to \infty$ határértéket véve kapjuk az $((lmax - 2)Bt + cmax)/cmin$ korlátot.

A $p' = lmax$ és $p = 1$ esetet külön kell kezelnünk. Egyszerű számítással adódik, hogy a fenti kifejezés szintén növekvő j-ben $lmax \geq 3$ esetén, amiből kapjuk ugyanezt a felső korlátot.

B.2. eset: $0 < p' < p \leq \min\{j, lmax\}$:

Ebben az esetben igaz, hogy

$$\begin{aligned} q_1 Bt + q_2\, cmax &\leq (j - p' - 2r)Bt + r\, cmax \\ &\leq (j-p')Bt + \frac{j-p'}{lmax}(cmax - 2\,Bt). \end{aligned}$$

A B.1. esethez hasonlóan (3.6)-ból kapjuk, hogy

$$R_{FG}(D) \leq \frac{\|c_0\| + (j-p')Bt + (cmax - 2Bt)\frac{j-p'}{lmax}}{\frac{p'}{p}\|c_0\| + (j-p')\frac{cmin}{lmax}}.$$

Legyen $x = j - p'$, ekkor a fenti egyenlőtlenség jobb oldalát tekinthetjük úgy, mint egy háromváltozós függvényt, ahol a változókat x, p és p' jelöli:

$$f(x,p,p') := \frac{\|c_0\| + ((lmax-2)Bt + cmax)\frac{x}{lmax}}{\frac{p'}{p}\|c_0\| + \frac{cmin}{lmax}\,x}$$

Nyilvánvaló, hogy f monoton függvény x-ben. Az x szerinti parciális deriváltja

$$\frac{\partial f}{\partial x} = \frac{\|c_0\|}{lmax} \cdot \frac{((lmax-2)Bt + cmax)\frac{p'}{p} - cmin}{\left(\frac{p'}{p}\|c_0\| + \frac{cmin}{lmax}\,x\right)^2}.$$

A derivált pozitív minden lehetséges x, p és p' értékre, kivéve a $p' = 1$, $p = lmax$ és $2\,cmin > cmax$ esetet. Ha a derivált pozitív, és így a függvény növekvő x-ben, a kívánt korlát adódik, ha vesszük a $j \to \infty$ határértéket.

Különben (ha $p' = 1$, $p = lmax$ és $2\,cmin > cmax$) egy másik becslést használunk $\|c_0'\|$-ra, nevezetesen

$$\|c_0'\| \geq cmin > \frac{cmax}{2} \geq \frac{\|c_0\|}{2}.$$

Ebben az esetben (3.6)-ből adódik, hogy

$$R_{FG}(D) \leq \frac{\|c_0\| + (j-1)Bt + (cmax - 2Bt)\frac{j-1}{lmax}}{\frac{\|c_0\|}{2} + (j-1)\frac{cmin}{lmax}}$$

ami ismét növekvő függvény j-ben. A $j \to \infty$ határérték adja a kívánt korlátot. Annak a bizonyítása, hogy a B.2. esetben igazolt korlát a lehető legjobb, ugyanazon szótár segítségével történik, mint amit az általános esetben is használtunk, csupán a súlyokat módosítjuk a következőképpen:

forrásszó	u	w	u^2	$uw^{lmax-2}u$
kódszó	a	b	c	d
súly	Bt	Bt	$cmax$	$cmin$

C eset: $2Bt \leq cmax$:

Mivel az adott felső korlát triviális felső korlát minden nemhosszító szótárra (lásd [17]) elegendő megmutatni, hogy elérhető ugyanazzal a szótárral, más súlyokat alkalmazva.

forrásszó	u	w	u^2	$uw^{lmax-2}u$
kódszó	a	b	c	d
súly	Bt	Bt	$2Bt$	$cmin$

□

A következő tétel azt állítja, hogy a *prefix* tulajdonság nem javít a felső és az alsó korlátokon.

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.

BIZONYÍTÁS. Annak bizonyításához, hogy az előző tételekben már igazolt felső korlátok prefix szótárakra is élesek, a következő, három szimbólumos $\{u, v, w\}$ ábécéből képzett szótárat alkalmazhatjuk:

forrásszó	u	v	w	uv	vw^j $j=1,\ldots,lmax-2$	$vw^{lmax-2}u$
kódszó	a	b	c	d	e_j	f
súly	$cmax$	$cmax$	$cmax$	$cmax$	$cmax$	$cmin$

$i > 0$-ra vegyük az $S_i = u(vw^{lmax-2}u)^i$,$i(lmax+1)+1$ hosszú karaktersorozatokat. Nyilvánvaló, hogy $OPT(D, S_i) = af^i$ és $FG(D, S_i) = (dc^{lmax-2})^i a$. Így

$$R_{FG}(D) \geq \lim_{n\to\infty} \frac{\|FG(D,S_i)\|}{\|OPT(D,S_i)\|} = \lim_{i\to\infty} \frac{i(lmax-1)cmax + cmax}{i \cdot cmin + cmax} = \frac{(lmax-1)cmax}{cmin}.$$

Egyenlő kódhosszúságú és nemhosszító szótárakra vonatkozó példák hasonló módon konstruálhatók.

□

Összefoglalva, az általános, egyenlő kódhosszúságú, nemhosszító és prefix illetve ezen tulajdonságok összes értelmes kombinációjának megfelelő jellemzőjű szótárak esetén az LM– és az FG–heurisztikák legrosszabb–eset viselkedése azonos. A DG–heurisztika egy kicsit különbözik ezektől az általános esetben, egyébként azonos korlátokkal rendelkezik.

A következőkben suffix szótárakra fogunk tételeket bizonyítani. Mindvégig feltételezni fogjuk, hogy $lmax \geq 3$, mivel $lmax = 2$ esetén minden szótár egyben prefix tulajdonságú is, így az előző tételek alkalmazhatók. A következő tétel azt mutatja, hogy *suffix szótárakra*a hányadoson alapuló greedy algoritmus egészen másképpen viselkedik, mint más heurisztikák.

3.4.5. TÉTEL. [6] *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

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

BIZONYÍTÁS. Ismét olyan karaktersorozatokat tekintünk, amelyre az FG–út és az OPT–út diszjunkt, és közös kezdő, illetve végpontjaik vannak. Jelölje az FG–utat tetszőleges két v_a, v_{a+1} csúcsa között az OPT–útnak $z_1, z_2, \ldots, z_{k+1}$. Feltételezzük, hogy létezik olyan él az FG–úton, amely z_1-et a v_a-t megelőző

csúccsal köti össze és egy olyan él, amely z_{k+1}-ből egy v_{a+1}utáni csúcsba vezet (úgynevezett "átlépő élek"), kivéve természetesen a karaktersorozat elejét és végét.

Az FG–úton található (z_i, z_{i+1}) él hosszát illetve súlyát jelölje t_i illetve c_i. Továbbá, a z_{k+1} és a v_{a+1} közötti karakterek száma legyen β.

A következőkben két egyszerű megfigyelést használunk:

$$\sum_{i=1}^{k} t_i \leq lmax - \beta - 1 \tag{3.7}$$

$$\frac{c_i}{t_i} \leq \frac{cmax}{\sum_{s=i}^{k} t_s + \beta} \quad \forall i \tag{3.8}$$

A jelölés egyszerűsítésére legyen $T := \sum_{i=1}^{k} t_i + \beta$. Ekkor

$$T \leq lmax - 1.$$

Tekintsük a (3.8) összefüggést, és adjuk össze minden i-re, $1 \leq i \leq k$, így egy felső korlátot kapunk a súlyok összegére:

$$\sum_{i=1}^{k} c_i \leq cmax \sum_{i=1}^{k} \frac{t_i}{T - \sum_{s=1}^{i-1} t_s} \leq cmax \ln T$$

A második egyenlőtlenség a 3.4.6. lemmából következik.

A kettőt összerakva, és figyelembe véve hogy a z_{k+1}-ből induló, és v_{a+1}-t átlépő él hossza legfeljebb $cmax$, azt kapjuk, hogy

$$R_{FG}(D) \leq \frac{\sum_{i=1}^{k} c_i + cmax}{cmin} \leq \frac{\ln(lmax - 1)cmax + cmax}{cmin}.$$

A következő konstrukció megadja a D_0 szótárat és a kívánt alsó korlátot. Tekintsünk egy $lmax$ betűből álló ábécét, legyenek a betűk $u, v, w_1, \ldots, w_{lmax-2}$ és vegyük a következő D_0 szótárat:

forrás–szó	u	v	w_j $j=1,\ldots,lmax-2$	uv	$w_j \ldots w_{lmax-2}u$ $j=1,\ldots,lmax-2$	$vw_1 \ldots w_{lmax-2}u$
kódszó	a	b	c_j	d	e_j	f
súly	$cmax$	$cmax$	$\frac{cmax}{lmax-j}$	$cmax$	$cmax$	$cmin$

Legyen $cmax = (lmax-1)!\,cmin$ és $S_i = u(vw_1 \dots w_{lmax-2}u)^i$ a tömöríteni kívánt karaktersorozatok, amelyek hossza $n = i \cdot lmax + 1$. Látható, hogy ha az FG algoritmus egyenlőség esetén a c_j élt választja, akkor $OPT(D_0, S_i) = af^i$ és $FG(D_0, S_i) = (dc_1 \dots c_{lmax-2})^i a$. Ebből adódik, hogy

$$\begin{aligned} R_{FG}(D_0) &\geq \limsup_{n\to\infty} \frac{\|FG(D_0,S_i)\|}{\|OPT(D_0,S_i)\|} = \lim_{i\to\infty} \frac{i\left(cmax + \sum_{j=1}^{lmax-2} \frac{cmax}{lmax-j}\right) + cmax}{cmax + i\cdot cmin} \\ &= \lim_{i\to\infty} \frac{i\cdot cmax\left(1 + \sum_{j=2}^{lmax-1} \frac{1}{j}\right) + cmax}{cmax + i\cdot cmin} \\ &\geq \frac{cmax\left(1 + \frac{1}{lmax-1} + \ln(lmax-1) - \ln 2\right)}{cmin}., \end{aligned}$$

Az utolsó egyenlőtlenségnél felhasználtuk, hogy

$$\sum_{k=2}^{n} \frac{1}{k} \geq \ln(n+1) - \ln 2\,. \tag{3.9}$$

□

Be kell még látnunk a fenti bizonyításban említett lemmát.

3.4.6. LEMMA. [6] *Tetszőleges $s_1, \dots, s_k, b$ $s_i \geq 1$, $b \geq 1$ egész számokra és $S = \sum_{i=1}^{k} s_i + b$-re*

$$\sum_{i=1}^{k} \frac{s_i}{S - \sum_{j=1}^{i-1} s_j} \leq \sum_{i=1}^{S-b} \frac{1}{S-i+1} \leq \ln S$$

teljesül.

BIZONYÍTÁS. Az első egyenlőtlenség bizonyításához megmutatjuk, hogy a baloldal $s_i = 1$-re, $i = 1, \dots, k$, veszi fel a maximumát.

Ha valamelyik s_i, például s_m nagyobb vagy egyenlő mint 2, akkor helyettesíthetjük két értékkel, mégpedig az $s_{m_1} = s_m - 1$ és az $s_{m_2} = 1$ értékekkel.

Ekkor a baloldal változása

$$\frac{s_{m_1}}{S - \sum_{j=1}^{m-1} s_j} + \frac{s_{m_2}}{S - \sum_{j=1}^{m-1} s_j - s_{m_1}} - \frac{s_m}{S - \sum_{j=1}^{m-1} s_j}.$$

Legyen $\bar{S} := S - \sum_{j=1}^{m-1} s_j$, és így a változás

$$\frac{1}{\bar{S} - s_m + 1} - \frac{1}{\bar{S}} > 0$$

ami ellentmond a feltevésnek, hogy a baloldal növelhető azzal, ha valamely s_i-t 1-nél nagyobbra választunk.

A lemma második egyenlőtlensége a harmonikus sorra vonatkozó korlátból adódik:

$$\sum_{i=1}^{S-b} \frac{1}{S-i+1} = \sum_{i=b+1}^{S} \frac{1}{i} \leq \sum_{i=2}^{S} \frac{1}{i} \leq \ln S$$

□

Ha szeretnénk összehasonlítani az FG heurisztika legrosszabb–eset viselkedését az LM és a DG algoritmusokéval suffix szótárakra, a 3.2.3., 3.3.9. és 3.4.5. tételeket kell megvizsgálnunk.

Látható, hogy az LM–algoritmus jobb legrosszabb–eset hányadossal rendelkezik mint az FG–heurisztika, amely a fontosabb esetekben jobb mint a DG–módszer. Hogy pontosak legyünk, ha $cmax \leq 3/2Bt$, akkor R_{DG} kisebb mint R_{FG}. Ha $cmax$ $3/2Bt$ és $(lmax - 3/2)Bt$ között van, akkor nincs jelentős különbség. (vagyis ha $cmax = (lmax - 2)Bt$, akkor FG jobb $lmax \geq 7$-re, ha $cmax = 2\,Bt$, akkor DG jobb minden $lmax \geq 3$-ra.) Az utolsó esetben, vagyis ha $(cmax > (lmax - 3/2)Bt)$, akkor elemi számításokkal ellenőrizhető, hogy az FG mindig jobb, mint a DG.

A kombinált, suffix és nemhosszító szótárakra vonatkozóan bonyolultabb analízis szükséges.

3.4.7. TÉTEL. [6] *Legyen D egy* nemhosszító, suffix *szótár. Ekkor*

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

BIZONYÍTÁS. A minimum zárójelben található kifejezés triviális felső korlát bármely nemhosszító szótárra, amely jobb mint a második korlát, ha $cmax$ közel van $lmax \cdot Bt$-hez.

A második kifejezés bizonyításához ugyanazt a jelölést használjuk, mint a 3.4.5. tételben. Továbbá feltételezzük, hogy $cmax = \alpha\,Bt$ valamilyen $\alpha > 0$-ra. Ha az FG algoritmus a z_i csúcsnál egy t_i hosszú és c_i súlyú élt választ, akkor ennek az

élnek "jobbnak" kell lenni, mint a közvetlenül v_{a+1}-be futó suffix él, amelynek súlya $cmax$ is lehet. Ebből következik, hogy

$$\frac{c_i}{t_i} \leq \frac{cmax}{\sum_{s=i}^{k} t_s + \beta},$$

továbbá a nemhosszító tulajdonság miatt

$$\frac{c_i}{t_i} \leq Bt.$$

Jelölje z_j a legkisebb indexű olyan csúcsot a $z_1, ..., z_k$ közül, amelyre $Bt \leq cmax/(\sum_{s=i}^{k} t_s + \beta)$ és legyen $T_i = \sum_{s=i}^{k} t_s + \beta$. Ekkor $Bt \leq \alpha\, Bt/T_j$ és így $T_j \leq \alpha$.

Összeadva az összes súlyra kapjuk, hogy

$$\begin{aligned}
\sum_{i=1}^{k} c_i &\leq \left(\sum_{i=1}^{j-1} \frac{t_i}{T_i} cmax + \sum_{i=j}^{k} t_i Bt \right) \\
&= cmax \left(\sum_{i=1}^{j-1} \frac{T_i - T_{i+1}}{T_i} + \frac{1}{\alpha}(T_j - \beta) \right) \\
&\leq cmax \left((j-1) - \sum_{i=1}^{j-2} \frac{T_{i+1}}{T_i} - \frac{T_j}{T_{j-1}} + 1 - \frac{1}{\alpha} \right).
\end{aligned}$$

(Az utolsó egyenlőtlenségnél a $\beta \geq 1$ és a $T_j \leq \alpha$ egyenlőtlenségeket alkalmaztuk.)

A számtani és a mértani középre vonatkozó összefüggésekből adódik, hogy

$$\sum_{i=1}^{j-2} \frac{T_{i+1}}{T_i} \geq (j-2) \sqrt[j-2]{\frac{T_{j-1}}{T_1}} \geq (j-2) \sqrt[j-2]{\frac{\alpha}{T_1}}. \tag{3.10}$$

Az $x - x\sqrt[x]{y} \leq \ln(1/y)$ egyenlőtlenségből $x, y > 0$-ra kapjuk, hogy

$$\begin{aligned}
\sum_{i=1}^{k} c_i &\leq cmax \left((j-2) - (j-2) \sqrt[j-2]{\frac{\alpha}{T_1}} + 2 - \frac{1}{\alpha} \right) \\
&\leq cmax \left(\ln\left(\frac{T_1}{\alpha}\right) + 2 - \frac{1}{\alpha} \right) \\
&\leq cmax\, \ln\left(\frac{lmax\, Bt}{cmax}\right) + 2\, cmax - Bt\,.
\end{aligned} \tag{3.11}$$

Ha hozzáadjuk ehhez az átlépő él súlyát, és elosztjuk *cmin*-nel, megkapjuk a kívánt korlátot.

□

Annak igazolására, hogy a fenti korlát csaknem a legjobb, megadunk egy általános alsó korlátot suffix, nemhosszító szótárakra. Pontos korlátot nem sikerült konstruálni a fenti bizonyításban alkalmazott becslések miatt. Természetesen a triviális, $lmax \cdot Bt/cmin$ érték elérhető (lásd [17]).

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

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

BIZONYÍTÁS. Ugyanazt a D_1 szótárat tekintjük, amit a 3.4.5. tétel bizonyításában alkalmaztunk. A szótár $lmax$ betűs ábécéből épül fel, ezek az $u, v, w_1, \ldots, w_{lmax-2}$ karakterek, de ezúttal más súlyokat használunk. Legyen $cmax = (lmax-1)!\, cmin$ és $cmax = \alpha\, Bt$, $\alpha \in \mathbb{N}$ esetén.

forrás-szó	u	v	w_j $j=1,\ldots,lmax-2$	uv	$w_j \ldots w_{lmax-2}u$ $j=1,\ldots,lmax-2$	$vw_1 \ldots w_{lmax-2}u$
kódszó	a	b	c_j	d	e_j	f
súly	Bt	Bt	γ_j	$2Bt$	$\gamma_j(lmax-j)$	$cmin$

ahol

$$\gamma_j = \begin{cases} \frac{cmax}{lmax-j} & j = 1, \ldots, lmax-\alpha \\ Bt & j = lmax-\alpha+1, \ldots, lmax-2 \end{cases}$$

Legyenek $S_i = u(vw_1 \ldots w_{lmax-2}u)^i$ az $n = i \cdot lmax + 1$ hosszú, tömörítendő karaktersorozatok. Ahogy a 3.4.5. tételnél, ha az FG algoritmus egyenlőség esetén a c_j élet választja, akkor $OPT(D_1, S_i) = af^i$ és $FG(D_1, S_i) = (dc_1 \ldots c_{lmax-2})^i a$.

A példából következik, hogy

$$\begin{aligned} R_{FG}(D_1) &\geq \limsup_{n\to\infty} \frac{\|FG(D_1,S_i)\|}{\|OPT(D_1,S_i)\|} \\ &= \lim_{i\to\infty} \frac{i\left(2\,Bt + \sum_{j=1}^{lmax-\alpha} \frac{cmax}{lmax-j} + \sum_{j=lmax-\alpha+1}^{lmax-2} Bt\right) + Bt}{Bt + i\cdot cmin} \\ &= \frac{2Bt + cmax\sum_{j=\alpha}^{lmax-1}\frac{1}{j} + (\alpha-2)Bt}{cmin} \\ &\geq \frac{cmax\ln\left(\frac{lmax}{\alpha}\right) + \alpha\,Bt}{cmin} \\ &= \frac{cmax\left(\ln\left(\frac{lmax\,Bt}{cmax}\right)+1\right)}{cmin} \end{aligned}$$

felhasználva a (3.9) egyenlőtlenséget.

□

Az FG más heurisztikákkal való összehasonlításához suffix és nemhosszító szótárak esetén a 3.2.3., 3.3.6. és a 3.4.8. tételeket használhatjuk. Meglepő módon ilyen típusú szótárakra mind az LM, mind a DG algoritmusok egy kicsit jobb eredményt adnak mint az FG.

A hányadoson alapuló greedy algoritmusra vonatkozó összes eredményt a következő táblázat foglalja össze:

D szótár				
Prefix	Suffix	Egyenlő kódhosszúságú	Nemhosszító	$R_{FG}(D)$
x	–	–	–	$\frac{(lmax-1)cmax}{cmin}$
x	–	x	–	$lmax-1$
x	–	–	x	$\frac{lmax\ Bt}{cmin}$
x	–	x	x	$lmax-1$
–	x	–	–	$\frac{cmax(\ln(lmax-1)+1)}{cmin}$
–	x	x	–	1
–	x	–	x	$\frac{cmax\left(\ln\left(\frac{lmax\cdot Bt}{cmax}\right)+1\right)}{cmin}$
–	x	x	x	1
–	–	–	–	$\frac{(lmax-1)cmax}{cmin}$
–	–	x	–	$lmax-1$
–	–	–	x	$\frac{lmax\ Bt}{cmin}\ (cmax \geq 2Bt)$
–	–	x	x	$lmax-1$

3.5. A leghosszabb szelet algoritmus

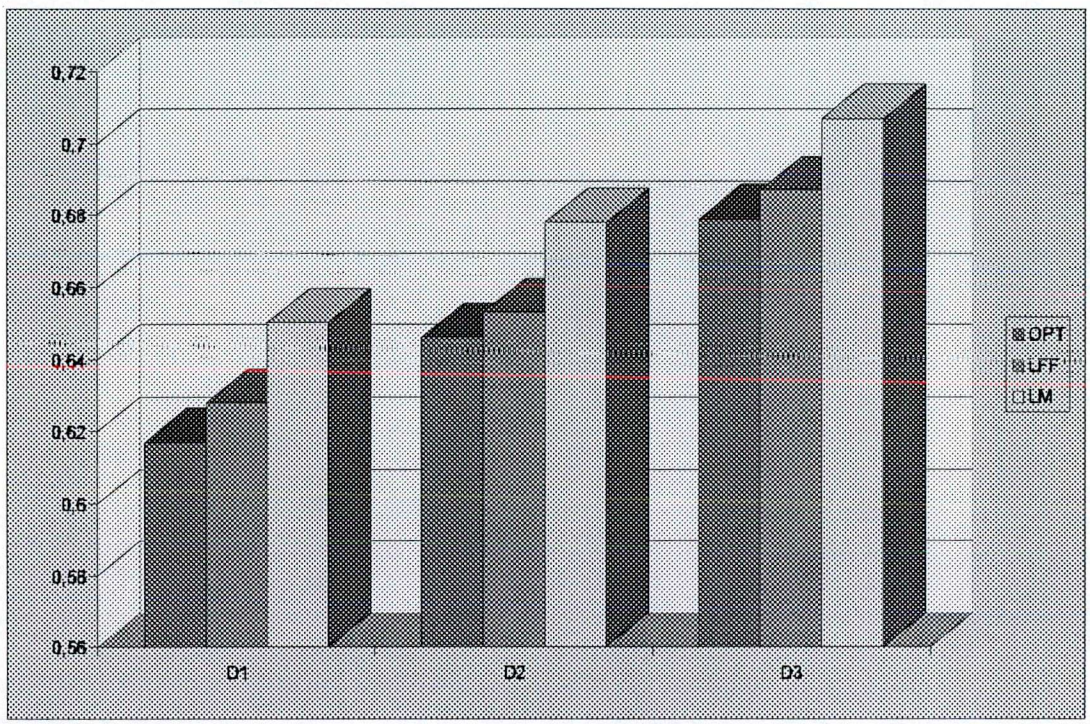
Több mint 12 évvel ezelőtt SHUEGRAF és HEAPS [22] vezette be a *leghosszabb szelet (longest fragment first,* továbbiakban *LFF)* heurisztikát. Azt feltételezték, hogy a tömöríteni kívánt adatok azonos hosszúságú rekordokból épülnek fel. Az ötlet a következő: az aktuális rekordon belül az algoritmus kiválasztja 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.

Látható, hogy az LFF_{LM} algoritmus feladja a szigorú értelemben vett on-line tulajdonságot, mivel előre néz a bufferben, és ezzel több információt szerez annak tartalmáról, mint amivel az on–line algoritmusok rendelkeznek. Az az érzésünk támadhat: minél több információnk van a tömöríteni kívánt szövegről, annál jobb legrosszabb eset hányadost kaphatunk. Ez azt sejteti számunkra, hogy az LFF_{LM} algoritmus jobban viselkedik, mint az on–line algoritmusok. A tapasztalati eredmények azt mutatták [22], hogy az LFF típusú eljárások jobb tömörítési eredményeket adnak, mint a leghosszabb illesztés módszere, vagy a különbségen alapuló greedy algoritmus. Ez látható az alábbi táblázatban, amely három különböző szótárra (D1, D2, D3) mutatja be a kapott tömörítési arányokat az optimális algoritmus (OPT), a leghosszabb illesztés módszere (LM) és a leghosszabb szelet algoritmus(LFF) esetén. Az adatok a SHUEGRAF és HEAPS által elvégzett elemzésekből származnak [21], [22].

	OPT	LFF	LM
D1	0,617	0,628	0,65
D2	0,646	0,653	0,678
D3	0,679	0,687	0,707

A táblázat adatai grafikus formában megjelenítve a következőképpen néznek ki:



6. ábra

Mostanáig azonban nem léteztek egzakt bizonyítások az LFF_{LM} algoritmus legrosszabb–esetére vonatkozóan. GALAMBOS GÁBORRAL és TIMO RAITÁVAL közösen éles korlátokat bizonyítottunk különböző típusú szótárakra. A kö vet ke zők ben 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.

BIZONYÍTÁS. Legyen S egy tetszőleges karaktersorozat a D által megadott ábécéből. Legyen a buffer hossza $t \cdot lmax$ ahol $t \geq 2$ egy paraméter. Mivel az aszimptotikus legrosszabb eset hányados érdekel bennünket, elegendő olyan szövegekkel foglalkozni, amelyre $|S| = s \cdot t \cdot lmax$, ahol $s \geq 1$. Ekkor $s = \frac{|S|}{t \cdot lmax}$ az LFF_{LM} algoritmus által végrehajtott lépések száma. Mivel $t \geq 2$, a buffer minden lépésben tartalmaz legalább egy élt az S optimális kódolásából. Jelölje az optimális kódolás leghosszabb éleinek hosszát az egyes bufferekben, illetve az LFF_{LM} algoritmus által kiválasztott leghosszabb éleket a lépések során $l_1, l_2, ..., l_s$ illetve $t_1, t_2, ..., t_s$. Nyilvánvaló, hogy $t_i \geq l_i$, $i = 1, ..., s$, így kapjuk, hogy

$$OPT(D,S) \geq \left(s + \frac{|S| - \sum_{i=1}^{s} l_i}{lmax} \right) cmin = \left(\frac{|S|}{t \cdot lmax} + \frac{|S| - \sum_{i=1}^{s} l_i}{lmax} \right) cmin.$$

A $t \cdot lmax$ hosszú buffer tartalma a legrosszabb esetben az i-edik lépésben legfeljebb

$$(t \cdot lmax - t_i - (t-2) + 1)\, cmax \leq (t \cdot lmax - l_i - (t-2) + 1)\, cmax \quad (3.12)$$

bittel kódolható, így

$$\begin{aligned} LFF_{LM}(D,S) &\leq \left(s + s \cdot t \cdot lmax - \sum_{i=1}^{s} l_i - s\,(t-2) \right) cmax \\ &= \left(\frac{|S|}{t \cdot lmax} + |S| - \sum_{i=1}^{s} l_i - \frac{|S|\,(t-2)}{t \cdot lmax} \right) cmax. \end{aligned}$$

Vezessük be az $L(s,t) = (\sum_{i=1}^{s} l_i)/\,|S|$ jelölést. Ekkor

$$\begin{aligned} R_{LFF}(LM,D) &\leq \lim_{|S| \to \infty} \frac{\frac{|S|}{t \cdot lmax} + |S| - |S|\,L(s,t) - \frac{|S|(t-2)}{t \cdot lmax}}{\frac{|S|}{t \cdot lmax} + \frac{|S| - |S| L(s,t)}{lmax}} \frac{cmax}{cmin} \\ &= \frac{\frac{1}{t \cdot lmax} + 1 - L(s,t) - \frac{(t-2)}{t \cdot lmax}}{\frac{1}{t \cdot lmax} + \frac{1 - L(s,t)}{lmax}} \frac{cmax}{cmin}. \end{aligned}$$

Nyilvánvaló, hogy $\frac{1}{t \cdot lmax} \leq L(s,t) \leq \frac{1}{t}$, és az előző kifejezés a maximumát $L(s,t) = \frac{1}{t}$-nél veszi fel. Ebből adódik, hogy

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

□

3.5.2. TÉTEL. [5] *A 3.5.1. tételben megadott korlát éles LFF_{LM}-re vonatkozóan.*

BIZONYÍTÁS. A korlátot a következő konstrukcióval tudjuk elérni. Legyen D az alábbi szótár:

forrásszó	u	v	w	uw	wv	u^{lmax}	v^{lmax}	w^{lmax}	$u^{lmax-1}v$
kódszó	a	b	c	d	e	f	g	h	j
súly	$cmax$	$cmax$	$cmax$	$cmax$	$cmax$	$cmax$	$cmax$	$cmax$	$cmin$

Vegyük a következő karaktersorozatokat:

$$S_i = \begin{cases} \left(u^{lmax}\left(v^{lmax}w^{lmax}\right)^{\frac{t-2}{2}}v^{lmax}\right)^i & \text{ha } t \text{ páros,} \\ \left(u^{lmax}\left(v^{lmax}w^{lmax}\right)^{\frac{t-1}{2}}\right)^i & \text{ha } t \text{ páratlan.} \end{cases}$$

S_i optimális kódolása a következő:

$$OPT(D, S_i) = \begin{cases} \left(a\,(gh)^{\frac{t-2}{2}}\,g\right)^i & \text{ha } t \text{ páros,} \\ \left(a\,(gh)^{\frac{t-1}{2}}\right)^i & \text{ha } t \text{ páratlan.} \end{cases}$$

Az LFF_{LM} algoritmus S_i-t a következőképpen kódolja:

$$LFF_{LM}(D, S_i) = \begin{cases} \left(ajb^{lmax-2}dc^{lmax-2}x^{\frac{t-4}{2}}db^{lmax-1}\right)^i & \text{ha } t \text{ páros,} \\ \left(ajb^{lmax-2}dc^{lmax-2}x^{\frac{t-3}{2}}c\right)^i & \text{ha } t \text{ páratlan,} \end{cases}$$

ahol $x = db^{lmax-2}dc^{lmax-2}$.

A fentiekben azt feltételeztük, hogy egyenlőség esetén az algoritmus bármely lehetséges élet választhatja. Ekkor

$$\begin{aligned} R_{LFF}(LM, D) &\geq \lim_{i\to\infty} \frac{\|LFF_{LM}(D, S_i)\|}{\|OPT(D, S_i)\|} \\ &= \lim_{i\to\infty} \frac{i\left((t-1)\,lmax - (t-3)\right) cmax}{i \cdot t \cdot cmin} \\ &= \frac{(t-1)lmax - (t-3)}{t} \frac{cmax}{cmin}. \end{aligned}$$

□

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.

BIZONYÍTÁS. Feltéve, hogy $cmax = cmin$ a 3.5.1. és a 3.5.2. tételekből adódik a kívánt eredmény.

□

3.5.4. TÉTEL. [5] *Legyen D egy* nemhosszító *szótár. Ekkor*

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

ahol

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

és a fenti korlát éles.

BIZONYÍTÁS. Három esetet különböztetünk meg a $cmax$ és a Bt közötti kapcsolat alapján.

A eset: Tegyük fel, hogy $cmax \leq Bt$. Ezen feltételek mellett minden szótár nemhosszító, és a 3.5.1. illetve 3.5.2. tételek vátoztatás nélkül alkalmazhatók.

B eset: Tegyük fel, hogy $Bt < cmax \leq 2\ Bt$. Ugyanazt a technikát használjuk, mint a 3.5.1. tétel bizonyításánál, de 3.12 helyett ezúttal a következőt írhatjuk: Az LFF úton található élek teljes súlya az i. lépésben legfeljebb

$$(t \cdot lmax - t_i - 2\,(t-2))\,Bt + (t-2)\,cmax + cmax.$$

Hasonlóan mint a 3.5.1. tételnél, kapjuk, hogy

$$R_{LFF}(LM, D) \leq \frac{(t-1)lmax\ Bt - t(2Bt - cmax) + 4Bt - cmax}{t \cdot cmin}.$$

C eset: Legyen $2Bt < cmax \leq lmax \cdot Bt$. Ebben az esetben az élek teljes súlya az LFF-úton az i. lépésben legfeljebb $(t \cdot lmax - t_i)\,Bt + cmax$. Ebből adódik, hogy

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

Ahhoz, hogy belássuk a fenti korlátok élességét, módosítanunk kell a 3.5.2. tételben használt szótár súlyait úgy, hogy ha $cmax > l \cdot Bt$, akkor $cmax$ helyett mindig $l \cdot Bt$-t írunk, ahol l a megfelelő forrásszó hossza.

□

3.5.5. TÉTEL. [5] *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.

BIZONYÍTÁS. Azt használjuk ki, hogy az egyenlő kódhosszúságú és nemhosszító szótárak esetén minden kódszó hossza azonos.

□

A következő tételek azt bizonyítják, hogy prefix szótárak esetén a legrosszabb–eset viselkedés ennél az algoritmusnál is azonos az általános szótárakra vonatkozóval.

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.

BIZONYÍTÁS. Mivel a 3.5.1. tételben megadott korlát érvényes bármilyen szótárra, csupán egy olyan prefix szótárat kell konstruálnunk, ami a megadott korlátot eléri.

forrásszó	u	v	w	uw	wv	u^j	v^j	w^j	$u^{lmax-1}v$
kódszó	a	b	c	d	e	f_j	g_j	h_j	l
súly	$cmax$	$cmax$	$cmax$	$cmax$	$cmax$	$cmax$	$cmax$	$cmax$	$cmin$

ahol $j = 1, .., lmax$.

Mivel ez a szótár a fentitől csupán néhány "extra" élben különbözik, és ellenőrizhető, hogy az LFF_{LM} algoritmus ugyanazokat az éleket választja, ezért a 3.5.2. tétel bizonyítását szóról szóra megismételhetjük.

□

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.

BIZONYÍTÁS. Ha kombináljuk a 3.5.6. és a 3.5.4. tételeket, kapjuk a kívánt eredményt.

□

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.

BIZONYÍTÁS. A 3.5.3., 3.5.4. és 3.5.6. tételekből, valamint abból a tulajdonságból, hogy ha a szótár egyenlő kódhosszúságú és nemhosszító akkor minden kódszó hossza azonos, következik a tétel állítása.

□

A következőkben suffix szótárakra vonatkozó tételeket bizonyítunk.

3.5.9. TÉTEL. [5] *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.

BIZONYÍTÁS. Tudjuk, hogy suffix szótárak esetén az LM algoritmusnál az optimális úton található élek száma nem lehet kevesebb mint az LM-úton.(Lásd Katajainen and Raita [17]). Sajnos elképzelhető, hogy a buffer végén az LFF_{LM} algoritmus nem "lát" egy élt, és annak hátsó szeleteit. Ez azt jelenti, hogy a legrosszabb esetben kapunk $lmax - 1$ plusz élt, amelyek súlya $cmax$ is lehet. Az is elképzelhető, hogy az LFF_{LM} által az adott bufferből először kiválasztott él "levág" egy élt, a hátsó szeleteivel. Ez a legrosszabb esetben további $lmax - 1$

$cmax$ hosszú élt jelenthet. Mindkét eset előfordulhat, ha $t \geq 3$, de csak egyik, ha $t = 2$. Tegyük fel, hogy az optimális algoritmus n élt használ az S karaktersorozat kódolásához. Ez azt jelenti, hogy az LFF_{LM} legfeljebb $n + 2\,(lmax - 1)$-ot, ha $t \geq 3$, és $n + (lmax - 1)$-ot, ha $t = 2$. Definiáljuk p-t a következőképpen:

$$p = \begin{cases} 1 & \text{ha } t = 2 \\ 2 & \text{ha } t \geq 3. \end{cases}$$

Ekkor

$$\begin{aligned} R_{LFF}(LM, D) &\leq \lim_{|S|\to\infty} \tfrac{n+ps(lmax-1)}{n} \tfrac{cmax}{cmin} \\ &\leq \lim_{|S|\to\infty} \left(1 + \tfrac{p\frac{|S|}{t\cdot lmax}(lmax-1)}{\frac{|S|}{lmax}}\right) \tfrac{cmax}{cmin} \\ &= \left(1 + \tfrac{p(lmax-1)}{t}\right) \tfrac{cmax}{cmin}. \end{aligned} \tag{3.13}$$

Most olyan példákat adunk, amelyek elérik a fenti korlátokat. Két esetet különböztetünk meg.

A eset: Tegyük fel, hogy $t = 2$. Legyen D a következő suffix szótár:

forrásszó	u	v	$u^k v$	$w^k v$	w^j	vw^{lmax-1}	$u^{lmax-1}v$	$w^{lmax-1}v$
kódszó	a	b	c_k	d_k	e_j	f	g	h
súly	$cmax$	$cmax$	$cmax$	$cmax$	$cmax$	$cmax$	$cmin$	$cmin$

ahol $k = 1, ..., lmax - 2$ és $j = 1, ..., lmax - 1$.

Tekintsük a következő karaktersorozatokat:

$$S_i = \left(u^{lmax-1}vw^{lmax-1}v\right)^i, i \geq 1.$$

Ekkor $LFF_{LM}\,(D, S_i) = \left(a^{lmax-1}fb\right)^i$ és $OPT\,(D, S_i) = (gh)^i$. Ebből kapjuk, hogy

$$\begin{aligned} R_{LFF}(LM, D) &\geq \lim_{i\to\infty} \frac{\|LFF_{LM}(D, S_i)\|}{\|OPT(D, S_i)\|} \\ &= \lim_{i\to\infty} \frac{i\,(lmax + 1)\,cmax}{2\;i \cdot cmin} \\ &= \frac{(lmax + 1)}{2} \frac{cmax}{cmin}. \end{aligned}$$

B eset: Legyen most $t \geq 3$. Legyen D a következő suffix szótár:

forrásszó	u	v	$v^k u$	$w^k u$	w^j	uw^{lmax-1}	$v^{lmax-1}u$	$w^{lmax-1}u$
kódszó	a	b	c_k	d_k	e_j	f	g	h
súly	$cmax$	$cmax$	$cmax$	$cmax$	$cmax$	$cmax$	$cmin$	$cmin$

ahol $k = 1, ..., lmax - 2$ és $j = 1, ..., lmax - 1$.

Tekinstük a következő karaktersorozatokat:

$$S_i = u\left(v^{lmax-1}u\left(w^{lmax-1}u\right)^{t-2}v^{lmax-1}u\right)^i, \text{ ha } i \geq 1.$$

Ekkor $LFF_{LM}(D, S_i) = \left(ab^{lmax-1}f^{t-2}ab^{lmax-1}\right)^i a$ és $OPT(D, S_i) = a\left(hl^{t-2}l\right)^i$.

Ebből kapjuk, hogy

$$\begin{aligned} R_{LFF}(LM, D) &\geq \lim_{i\to\infty} \frac{\|LFF_{LM}(D, S_i)\|}{\|OPT(D, S_i)\|} \\ &= \lim_{i\to\infty} \frac{i\,(2lmax + t - 2)\,cmax + cmax}{i \cdot t \cdot cmin + cmin} \\ &= \left(1 + \frac{2\,(lmax - 1)}{t}\right)\frac{cmax}{cmin}. \end{aligned}$$

□

3.5.10. TÉTEL. [5] *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.

BIZONYÍTÁS. Ha $cmax \leq Bt$ akkor a 3.5.9. tétel bizonyítása változtatás nélkül alkalmazható. Ha $cmax > Bt$, akkor a legrosszabb esetben az extra élek súlya az LFF-úton legfeljebb $2\,(lmax - 1)\,Bt$ ha $t \geq 3$, és $(lmax - 1)\,Bt$ ha $t = 2$. Ebből adódik a tétel állítása.

□

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

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

BIZONYÍTÁS. A $cmax = cmin$ feltételt kihasználva a 3.5.9. és a 3.5.10. tételekből kapjuk az állítást.

□

A következő táblázat összefoglalja a leghosszabb szelet módszerére vonatkozó eredményeket:

D szótár				
Prefix	Suffix	Egy.k.	Nemh.	$R_{LFF}(LM, D)$
x	–	–	–	$\frac{(t-1)lmax-(t-3)}{t}\frac{cmax}{cmin}$
x	–	x	–	$\frac{(t-1)lmax-(t-3)}{t}$
x	–	–	x	$\frac{(t-1)lmax\ Bt+cmax}{t \cdot cmin}$ $(cmax \geq 2Bt)$
x	–	x	x	$\frac{(t-1)lmax-(t-3)}{t}$
–	x	–	–	$\left(1+\frac{2(lmax-1)}{t}\right)\frac{cmax}{cmin}$ $(t \geq 3)$
–	x	x	–	$1+\frac{2(lmax-1)}{t}$ $(t \geq 3)$
–	x	–	x	$\frac{cmax}{cmin}+\frac{2(lmax-1)}{t}\frac{Bt}{cmin}$ $(t \geq 3$ és $cmax \geq 2Bt)$
–	x	x	x	$1+\frac{2(lmax-1)}{t}$ $(t \geq 3)$
–	–	–	–	$\frac{(t-1)lmax-(t-3)}{t}\frac{cmax}{cmin}$
–	–	x	–	$\frac{(t-1)lmax-(t-3)}{t}$
–	–	–	x	$\frac{(t-1)lmax\ Bt+cmax}{t \cdot cmin}$ $(cmax \geq 2Bt)$
–	–	x	x	$\frac{(t-1)lmax-(t-3)}{t}$

Végezetül mindenképpen érdemes megemlíteni, hogy amint a fenti táblázatból jól látható, amennyiben a buffer hossza a végtelenbe tart, eredményeink konvergálnak a megfelelő, LM algoritmusra vonatkozó korlátokhoz. Pontosabban a következőt kaptuk a legrosszabb–eset hányadosra:

$$\lim_{t\to\infty} R_{LFF}(LM, D) = R_{LM}(D).$$

Ez az eredmény nyilvánvaló, mivel minél hosszabb a buffer, annál kevesebb hatása van az első él kiválasztásának. Másrészt az is nyilvánvaló, hogy az LM kiválasztása teljesen véletlenszerű volt. Ugyanígy használhattuk volna a DG vagy más on–line algoritmust. Többé kevésbé az is nyilvánvalónak látszik, hogy ha A

egy tetszőleges on–line algoritmus és D egy adott szótár, akkor

$$\lim_{t\to\infty} R_{LFF}(A, D) = R_A(D),$$

de hogyan kellene ezt bizonyítani?

Másodsorban megfigyelhetjük, hogy a határértéket alulról közelítjük. Ez azt jelenti, hogy a legjobb eredményt a legkisebb bufferhosszra kapjuk, azaz ha $t = 2$. Ebben az esetben az LM heurisztikára vonatkozó korlátnál egy 2-es faktorral kapunk jobbat, és minden hosszabb buffer esetén az eredmény romlik a hossz függvényében. A kérdés nyilvánvaló: tudunk-e mondani az LFF algoritmusnak egy olyan változatát, amely a bufferhossztól független konstans faktorral javítja meg az LM-re vonatkozó eredményt? Javaslatunk a következő:
Használjuk először az LFF algoritmust a buffer kódolásához. Az első választás után - az LM helyett - kódoljuk a megmaradó részét a buffernek iteratív módon magával az LFF-fel. Ezt az algoritmust iterált leghosszabb szelet ($ILFF$) eljárásnak nevezzük. Sejtésünk az, hogy ha D egy a korábbi tulajdonságoknak eleget tevő szótár, akkor

$$R_{ILFF}(LM, D) = \frac{1}{2} R_{LM}(D).$$

További kérdések merülhetnek fel. Egy közülük: Ha azokat az algoritmusokat tekintjük, amelyek korlátozott bufferméretet használnak- ezeket *tárkorlátos eljárásoknak* nevezhetjük - akkor kérdés, létezik-e olyan tárkorlátos algoritmus, amely jobb eredményt ad, mind az $ILFF$-re vonatkozó sejtés?

4. fejezet

Ládapakolási algoritmusok elemzése

4.1. Bevezetés

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 előzőekben 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 szakaszt). 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 dolgozat következő részében 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. Bebizonyítjuk, hogy az algoritmusra (amelyet H_7-tel fogunk jelölni) teljesül a $H_7(L) \leq \frac{5}{4}L^* + 5$ egyenlőtlenség, bármely L listára.

A dolgozat következő része MARTEL eredményeit tárgyalja. Az utána következő szakaszban kerül sor a H_7 heurisztika ismertetésére és analizálására.

4.2. A Martel eredmény

Mint az előzőekben már említettük, erősen kihasználjuk a [18]-ban található eredményeket. Ezért most röviden ismertetjük MARTEL algoritmusát. Az eljárás az adott lista elemeinek egy egyszerű osztályozásán alapszik, amely az alábbi módon definált:

$$C_0 = \left\{x_i | 1 \geq x_i > \tfrac{2}{3}\right\},$$
$$C_1 = \left\{x_i | \tfrac{2}{3} \geq x_i > \tfrac{1}{2}\right\},$$
$$C_2 = \left\{x_i | \tfrac{1}{2} \geq x_i > \tfrac{1}{3}\right\},$$

$$C_3 = \left\{x_i | \tfrac{1}{3} \geq x_i > \tfrac{1}{4}\right\},$$
$$C_4 = \left\{x_i | \tfrac{1}{4} \geq x_i > 0\right\}.$$

Legyen $c_i = |C_i|, i = 0, \ldots, 4$. Egy $x \in C_i$ elemet C_i-elemnek fogunk nevezni. Amint MARTEL említette, ezen osztályozás motivációja, hogy lehetővé tegyük az elemeknek azon halmaz alapján történő elhelyezését, amelyhez tartoznak. Az algoritmust H_4-gyel jelöljük. Az eljárás a következő:

1. Alakítsuk ki a C_i, $i = 0, \ldots, 4$ halmazokat.
2. Legyen $k = \left\lceil \frac{\min(c_1, c_2)}{2} \right\rceil$. Vágjuk szét C_1-et két részhalmazra: C_1^s tartalmazza a legkisebb k elemét C_1-nek, és C_1^b a maradék elemeket. Hasonló módon bontsuk fel a C_2 halmazt a C_2^s és C_2^b részhalmazokra. Vegyünk ki tetszőlegesen egy-egy elemet a C_1^s és a C_2^s halmazokból. Ha belefférnek egy ládába, akkor nyissunk egy új ládát, és helyezzük el őket benne. Ha nem férnek el egy ládába, tegyük a C_1^s-elemet egy üres ládába.
3. Tegyünk minden egyes C_0-elemet és C_1^b-elemet külön ládába.
4. Tegyük a maradék C_2-elemeket (C_2^s és C_2^b-elemek) párosával külön ládákba.
5. Amíg el nem fogynak a C_3-elemek, tegyük a C_3-elemeket olyan ládákba, amelyek egy egyedülálló C_1-elemet tartalmaznak.
6. A maradék C_3-elemeket tegyük hármasával külön ládákba.
7. Pakoljuk el a C_4-elemeket a Next-Fit szabály segítségével (a Next-Fit szabály leírását többek között [11] tartalmazza).

A fő tétel bizonyításához MARTEL egy fontos lemmát alkalmazott, amelyet most általánosabb formában ismertetünk.

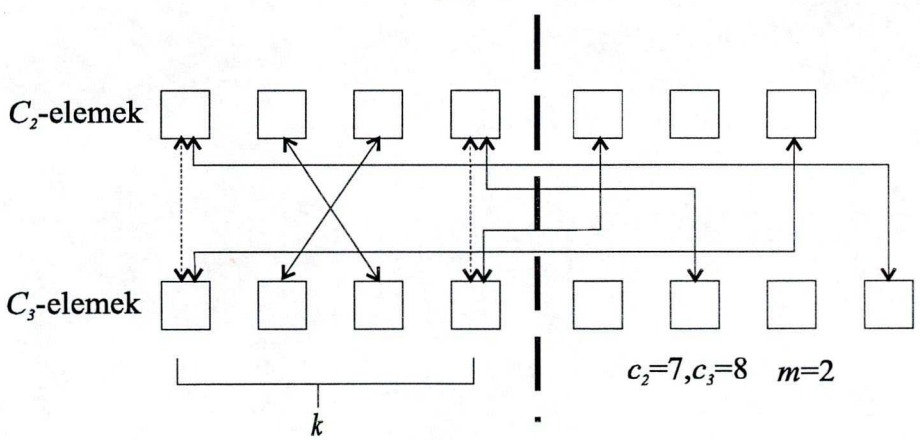
4.2.1. LEMMA. (MARTEL, [18]): *Két tetszőleges C_i, C_j diszjunkt elem halmazra legyen $k = \left\lceil \frac{\min(c_i, c_j)}{2} \right\rceil$. Legyen H tetszőleges heurisztika, amely kettévágja C_i-t és*

C_j-t két részhalmazra, úgy hogy C_i^s tartalmazza a k legkisebb elemét C_i-nek, és C_i^b a maradék elemeket. Hasonló művelet történik C_j-re. Ekkor H véletlenszerűen kivesz két elemet a C_i^s és a C_j^s halmazokból. Ha beleférnek egy ládába, akkor elhelyezi őket abban. Tegyük fel, hogy H $m\,(\leq k)$ elemet párosít össze ilyen módon. Ekkor a C_i és C_j elemekből egy optimális pakolásban legfeljebb $m+k$ olyan ládát tudunk képezni, amelyek egy-egy elemet tartalmaznak a két halmazból.

BIZONYÍTÁS. Ha $m = k$, akkor az állítás nyilvánvaló. Tegyük fel, hogy $m < k$. Ekkor vannak olyan $x_i \in C_i^s$ és $x_j \in C_j^s$ elemek, hogy $x_i + x_j > 1$. A legjobb lehetséges párosítási technika, hogy összerakunk legfeljebb m elemet C_i^b-ből C_j^s-beli elemekkel, legfeljebb m elemet C_i^s-ből C_j^b-beli elemekkel, és a C_i^s maradék $k-m$ elemét C_j^s-beli elemekkel. Ilyen módon legfeljebb $2m + (k-m) = m+k$ párt képezhetünk.

□

Az alábbi ábra a 4.2.1. lemma bizonyításának lényegét mutatja be egy példa segítségével.



7. ábra

A következőkben ismertetjük MARTEL tételét, amelyre GALAMBOS GÁBOR egy egyszerűbb bizonyítást talált. Most ezt mutatjuk be az eredeti helyett.

4.2.2. TÉTEL. (MARTEL, [18]): *Tetszőleges L listára* $H_4(L) \leq \frac{4}{3}L^* + 2$.

BIZONYÍTÁS. [11] Tegyük fel, hogy állításunk nem igaz. Ekkor létezik egy *minimális ellenpélda*, azaz egy olyan L lista, amelyre $H_4(L) > \frac{4}{3}L + 2$ és L számossága minimális. Nyilvánvalóan ez a lista nem tartalmazhat C_4-elemet. Két esetet különböztetünk meg:

A eset: Tegyük fel, hogy a 6. lépés létrehoz legalább egy ládát. Ebben az esetben

$$L^* \geq c_0 + c_1 + \left\lceil \frac{(c_2 + c_3) - (c_0 + c_1)}{3} \right\rceil,$$

és így

$$\begin{aligned} H_4(L) &= c_0 + c_1 + \left\lceil \tfrac{c_2 - m}{2} \right\rceil + \left\lceil \tfrac{c_3 - (c_1 - m)}{3} \right\rceil \\ &\leq c_0 + \tfrac{2}{3}c_1 + \tfrac{1}{2}c_2 + \tfrac{1}{3}c_3 - \tfrac{m}{6} + 2 \\ &\leq L^* + \tfrac{1}{3}c_0 + \tfrac{1}{6}c_2 - \tfrac{m}{6} + 2 \\ &\leq L^* + \tfrac{1}{3}\left(c_0 + \tfrac{1}{2}c_2\right) + 2 \\ &\leq \tfrac{4}{3}L^* + 2 \end{aligned}$$

ami ellentmondás.

B eset: Tegyük fel, hogy a 6. lépés nem hoz létre új ládákat. Alkalmazva a 4.2.1. lemmát a 2. lépésre kapjuk, hogy

$$L^* \geq c_0 + c_1 + \left\lceil \frac{c_2 - (k + m)}{2} \right\rceil \geq c_0 + c_1 + \frac{1}{2}c_2 - \frac{k}{2} - \frac{m}{2}.$$

Mivel $m \leq k = \min\left\{\frac{c_2}{2}, \frac{c_1}{2}\right\}$, azt kapjuk, hogy

$$\begin{aligned} H_4(L) &= c_0 + c_1 + \left\lceil \tfrac{c_2 - m}{2} \right\rceil \\ &\leq c_0 + c_1 + \tfrac{1}{2}c_2 - \tfrac{m}{2} + 1 \\ &\leq L^* + \tfrac{k}{2} + 1 \\ &\leq L^* + \tfrac{1}{4}c_1 + 1 \\ &\leq \tfrac{5}{4}L^* + 1. \end{aligned}$$

ami ismét ellentmondás.

□

Sok olyan lista van, amely bizonyítja, hogy a fenti korlát éles. Például vegyük a következő L_n listát $n = 2m$-re és $m \in \mathbb{N}$-re. L_n tartalmazzon m darab C_0-elemet és m darab C_3-elemet, amelyek nagysága legyen $3/4 - \varepsilon$ illetve $1/4 + \varepsilon$. Ekkor $H_4(L_n) = m + \frac{m}{3} = \frac{4}{3}m$, és $L_n^* = m$. Ebből következik, hogy $\frac{H_4(L_n)}{L_n^*} = 4/3$ minden n-re.

4.3. Az 5/4-es algoritmus

Megfigyelhetjük, hogy a H_4 algoritmus sokszor jobb eredményt ad mint a legrosszabb esetben (lásd a B eset bizonyítását). MARTEL megemlítette, hogy ha a C_3 és a C_4 elemeket jobban tudnánk kezelni (esetleg a C_4 halmaz két részre bontásával, amelyek a $\left(0, \frac{1}{5}\right]$, és az $\left(\frac{1}{5}, \frac{1}{4}\right]$ intervallumba eső elemeket tartalmaznák), akkor megjavíthatnánk a legrosszabb–eset hányadost.

Annak ellenére, hogy ez az ötlet meglehetősen egyszerűnek tűnt, csaknem 10 év telt el, és a probléma megoldatlan maradt. A következőkben megadunk egy új lineáris futási idejű algoritmust, amely $\frac{5}{4}$-es legrosszabb–eset hányadossal rendelkezik. A heurisztikát H_7-tel jelöljük. Mielőtt ismertetnénk az eljárást, osztályozzuk 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:

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^s 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 $C_{5,6}^s$ és $C_{5,6}^b$ részhalmazokra. Vegyünk ki tetszőlegesen egy-egy elemet a C_1^s és a $C_{5,6}^s$ halmazokból. Ha belefférnek egy ládába, akkor nyissunk egy új ládát, és helyezzük el őket benne. Ha nem férnek el egy ládába, tegyük a C_1^s-elemet egy üres ládába. Továbbá rakjuk a C_1^b-elemeket is üres ládákba.

4. Legyen $k_2 := \lceil \min\left\{\frac{c_2}{2}, \frac{c_3+c_4}{2}\right\}\rceil$. Vágjuk szét C_2-t két részhalmazra: C_2^s 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 $C_{3,4}^s$ és $C_{3,4}^b$ részhalmazokra. Vegyünk ki tetszőlegesen egy-egy elemet a C_2^s és a $C_{3,4}^s$ halmazokból. Ha belefférnek 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^s-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 $k_3 := \lceil \min\left\{\frac{c_5}{2}, \frac{c_4}{4}\right\} \rceil$. Vágjuk szét C_5-öt két részhalmazra: C_5^s 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^s tartalmazza a legkisebb $2k_3$ elemét C_4-nek, és C_4^b a maradék elemeket. Vegyünk ki tetszőlegesen egy elemet a C_1^s és két elemet a C_4^s halmazokból. Ha belefférnek egy ládába, akkor nyissunk egy új ládát, és helyezzük el őket benne. Ha nem férnek el egy ládába, tegyük a két C_4^s-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.

Az algoritmus működést egy példán keresztül is illusztráljuk. Legyen adott az alábbi L lista.

$$L = \left\{ \begin{array}{c} 0.25, 0.33, 0.34, 0.36, 0.68, 0.5, 0.53, 0.9, 0.21, 0.65, 0.65, 0.29, 0.75, 0.82, \\ 0.45, 0.36, 0.3, 0.22, 0.4, 0.72, 0.7, 0.36, 0.3, 0.24, 0.3, 0.25, 0.23, 0.36 \end{array} \right\}$$

Az egyes lépések a L esetén a következők:

1. lépés:

$C_0 = \{0.9, 0.82\}$

$C_1 = \{0.68, 0.75, 0.72, 0.7\}$

$C_2 = \{0.53, 0.65, 0.65\}$

$C_3 = \{0.5, 0.45, 0.4\}$

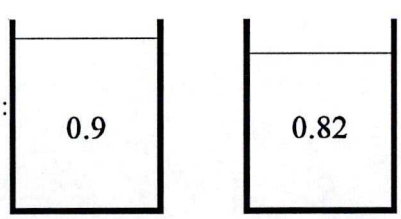
$C_4 = \{0.34, 0.36, 0.36, 0.36, 0.36\}$

$C_5 = \{0.33, 0.29, 0.3, 0.3, 0.3\}$

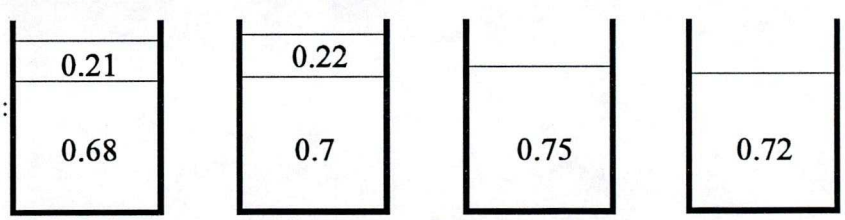
$C_6 = \{0.25, 0.21, 0.22, 0.24, 0.25, 0.23\}$

$C_7 = \emptyset$

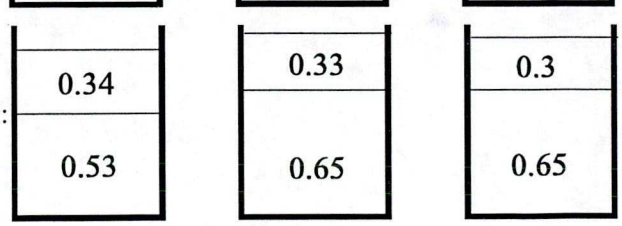
2. lépés:



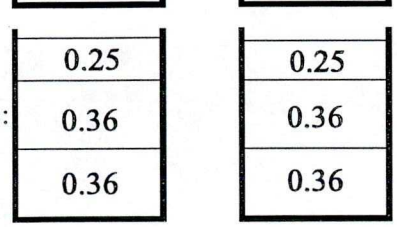
3. lépés:



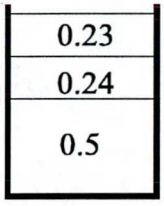
4. lépés:



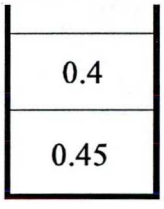
5. lépés:



6. lépés:

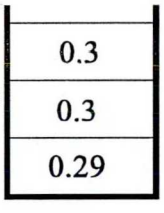


7. lépés:



8. lépés: $\emptyset$

9. lépés:



10., 11., 12., lépés: $\emptyset$.

A következőkben belátjuk, hogy $H_7(L) \leq \frac{5}{4}L^* + 3$ tetszőleges L listára. Ehhez az alábbi 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.

- Azokat az elemeket, amelyeket a 11. lépésben pakol el az algoritmus, nevezzük *megmaradt elemeknek*. Vegyük észre, hogy legfeljebb egy C_4-elem, két C_5-elem és a 6. lépés miatt legfeljebb egy C_3-elem és egy C_6-elem vagy nulla C_3-elem és legfeljebb három C_6-elem maradhat. Következésképpen legfeljebb két láda mindig elegendő a maradék elemek elpakolásához.

Az összes lehetséges kombinációt figyelembe véve, egy optimális pakolásban a ládák száma a következőképpen írható fel:

$$\begin{aligned} L^* = {} & y_1 + y_{15} + y_{16} + y_2 + y_{23} + y_{24} + y_{25} + y_{26} + y_{256} + y_{266} + \\ & + y_3 + y_{33} + y_{34} + y_{35} + y_{36} + y_{336} + y_{345} + y_{346} + y_{355} + \\ & + y_{356} + y_{366} + y_{3666} + y_4 + y_{44} + y_{45} + y_{46} + y_{445} + y_{446} + \\ & + y_{455} + y_{456} + y_{466} + y_{4566} + y_{4666} + y_5 + y_{55} + y_{56} + y_{555} + \\ & + y_{556} + y_{566} + y_{5556} + y_{5566} + y_{5666} + y_6 + y_{66} + y_{666} + y_{6666}. \end{aligned} \tag{4.1}$$

Az egyes osztályokban lévő elemeket a következő módon adhatjuk meg:

$$c_1 = y_1 + y_{15} + y_{16} \tag{4.2}$$

$$c_2 = y_2 + y_{23} + y_{24} + y_{25} + y_{26} + y_{256} + y_{266} \tag{4.3}$$

$$\begin{aligned} c_3 = {} & y_{23} + y_3 + 2y_{33} + y_{34} + y_{35} + y_{36} + 2y_{336} + y_{345} + y_{346} + y_{355} + y_{356} + \\ & + y_{366} + y_{3666} \end{aligned} \tag{4.4}$$

$$\begin{aligned} c_4 = {} & y_{24} + y_{34} + y_{345} + y_{346} + y_4 + 2y_{44} + y_{45} + y_{46} + 2y_{445} + 2y_{446} + y_{455} + \\ & + y_{456} + y_{466} + y_{4566} + y_{4666} \end{aligned} \tag{4.5}$$

$$\begin{aligned} c_5 = {} & y_{15} + y_{25} + y_{35} + y_{45} + y_{256} + y_{345} + 2y_{355} + y_{356} + y_{445} + 2y_{455} + y_{456} + \\ & + y_{4566} + y_5 + 2y_{55} + y_{56} + 3y_{555} + 2y_{556} + y_{566} + 3y_{5556} + 2y_{5566} + y_{5666} \end{aligned} \tag{4.6}$$

$$\begin{aligned} c_6 = {} & y_{16} + y_{26} + y_{36} + y_{46} + y_{56} + y_{256} + 2y_{466} + 2y_{266} + y_{336} + y_{346} + y_{356} + \\ & + 2y_{366} + y_{446} + y_{456} + y_{556} + 2y_{566} + 3y_{3666} + 3y_{4666} + 2y_{4566} + y_{5556} + \\ & + 2y_{5566} + 3y_{5666} + y_6 + 2y_{66} + 3y_{666} + 4y_{6666} \end{aligned} \tag{4.7}$$

Mielőtt hozzákezdenénk a legrosszabb–eset viselkedésre vonatkozó tétel bizonyításhoz, két lemmát kell igazolnunk, amelyeket a későbbiekben alkalmazni fogunk.

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.*

BIZONYÍTÁS. Két esetet különböztetünk meg. Ha feltesszük hogy $2c'_3 \leq c'_6$, akkor a 7. lépés üres, és így

$$H_7(L) \leq c'_3 + \left\lfloor \frac{c'_6 - 2c'_3}{4} \right\rfloor \leq c'_3 + \frac{c'_6 - 2c'_3}{4} = \frac{c'_3}{2} + \frac{c'_6}{4}.$$

Ha $c'_6 < 2c'_3$, akkor a 10. lépés üres, és

$$H_7(L) \leq \left\lfloor \frac{c'_6}{2} \right\rfloor + \left\lfloor \frac{c'_3 - \left\lfloor \frac{c'_6}{2} \right\rfloor}{2} \right\rfloor \leq \left\lfloor \frac{c'_6}{2} \right\rfloor + \frac{c'_3}{2} - \frac{\left\lfloor \frac{c'_6}{2} \right\rfloor}{2} \leq \frac{c'_3}{2} + \frac{c'_6}{4}.$$

amiből adódik az állítás.

□

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,*

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

BIZONYÍTÁS. A feltétel szerint $m_{445} < k_3$. Ezért vannak olyan $x_i, x_j \in C_4^s$ és $x_k \in C_5^s$ elemek, hogy $x_i + x_j + x_k > 1$. Következésképpen, ha $x_i, x_j \in C_4^b$ és $x_k \in C_5^b$, akkor $x_i + x_j + x_k > 1$. Tekintsük az összes olyan C_4^s és C_5^s-elemet, amelyet nem pakolt el a H_7 algoritmus hármasával. Jelöljük ezeket a halmazokat C_4^{sb}-vel illetve C_5^{sb}-vel. Jelöljük azon C_5-elemek halmazát amelyek C_2-elemekkel kerültek párba a 4. lépésben C_5^{b2}-vel, illetve azon C_4-elemek halmazát amelyek C_6-elemekkel

kerültek párba az 5. lépésben C_4^{b6}-tal. Az algoritmus definíciója szerint a C_5^{b2}-elemek mérete nem lehet kisebb mint a C_5^s és C_5^{sb} halmazokban található elemeké. Hasonló állítás igaz a C_4-elemekre. Vegyük észre, hogy ha $x_i, x_j \in C_4^b \cup C_4^{b6} \cup C_3$ és $x_k \in C_5^b \cup C_5^{b2}$ akkor $x_i + x_j + x_k > 1$. Most belátjuk, hogy az olyan $\langle 445 \rangle$ vagy $\langle 345 \rangle$ típusú ládák száma, amelyeket a $C_4^{sb} \cup C_5^{sb} \cup C_4^b \cup C_5^b \cup C_4^{b6} \cup C_5^{b2} \cup C_3$ halmaz elemeiből képezünk legfeljebb $k_3 - m_{445} - 1$ lehet. Tegyük fel az ellenkezőjét, vagyis hogy $T \geq k_3 - m_{445}$ ilyen típusú ládát tudunk képezni. Feltehetjük, hogy az összes $C_4^{sb} \cup C_5^{sb}$-beli elem szerepel a pakolásban, mert különben egy nagy elem helyettesíthető lenne egy hiányzó elemmel a $C_4^{sb} \cup C_5^{sb}$ halmazból, anélkül hogy bármi változna. Jelöljük az ezen ládákban levő elemek méretének összegét S-sel. Mivel a pakolások jók, ezért $T \geq S$. Másrészt az elemeket csoportosítani tudjuk úgy hármasával, hogy minden egyes hármasban az elemek összege nagyobb mint 1. Így kapunk T hármast. Ebből $S > T$ következik, ami ellentmondás.
Eddig nem vettük figyelembe azokat a C_5 és C_4-elemeket, amelyek a 8. lépésben hármasával kerültek elpakolásra, illetve C_1 vagy C_2-elemekkel rakta őket össze az algoritmus a 3. vagy a 4. lépésben. Az ilyen elemek száma $3m_{445} + m_{15} + m_{24}$. A lehető legjobb pakolás az, ha minden ilyen elemet két $C_4^b \cup C_5^b \cup C_4^{b6} \cup C_5^{b2} \cup C_3$-beli elemmel pakol össze az eljárás, feltéve hogy van elegendő számú ilyen elem. Ily módon legfeljebb $3m_{445} + m_{15} + m_{24}$ további ládát kaphatunk. Mivel definíció alapján $k_3 \leq \frac{c_4 - m_{24} - 2(c_6 - m_{16})}{4}$, az állítás adódik.

□

Ezek után következhet a fő tétel.

4.3.3. TÉTEL. [7] $H_7(L) \leq \frac{5}{4}L^* + 3$ *tetszőleges* L *listára. Továbbá végtelen sok olyan* L_n *lista létezik, amelyre* $\frac{H_7(L_n)}{L^*} = 5/4$.

BIZONYÍTÁS. Először csupán olyan olyan listákat fogunk tekinteni, amelyek nem tartalmaznak sem C_7 sem C_0-elemeket. Legyen L egy ilyen lista. Az egyenlőtlenség bizonyításához különböző eseteket fogunk tekinteni. Aszerint fogunk különbséget tenni, hogy a 4. és az 5. lépésben elfogynak-e vagy sem a C_5 és a C_6-elemek.

Jelölje $\tilde{c}_5$ azon C_5-elemek számát, amelyek nem fogynak el az 6. lépés kezdetéig (eltekintve a megmaradt elemektől), és $\tilde{c}_6$ az ugyanilyen C_6-elemek számát.

A eset: $\tilde{c}_5 = 0$, $\tilde{c}_6 = 0$

Könnyen kiszámítható, hogy ebben az esetben

$$H_7(L) \leq c_1 + c_2 + \left\lfloor \frac{c_3 - m_{23}}{2} \right\rfloor + \left\lfloor \frac{c_4 - m_{24}}{2} \right\rfloor + 2 \leq c_1 + c_2 + \frac{c_3 - m_{23}}{2} + \frac{c_4 - m_{24}}{2} + 2.$$

A 4.2.1. lemmát alkalmazva a C_2 és a $C_{3,4}$ halmazokra adódik, hogy

$$L^* \geq c_1 + c_2 + \frac{c_3 + c_4 - k_2 - m_{23} - m_{24}}{2},$$

és így

$$H_7(L) \leq L^* + \frac{k_2}{2} + 2 \leq L^* + \frac{c_2}{4} + 2 \leq \frac{5}{4}L^* + 2.$$

B eset: $\tilde{c}_5 = 0$, $\tilde{c}_6 > 0$

A 4.3.1. lemma segítségével kiszámíthatjuk a H_7 által felhasznált ládák számát

$$\begin{aligned} H_7(L) &\leq c_1 + c_2 + \frac{c_3 - m_{23}}{2} + \left\lfloor \frac{c_4 - m_{24}}{2} \right\rfloor + \frac{c_6 - m_{16} - m_{26} - \left\lfloor \frac{c_4 - m_{24}}{2} \right\rfloor}{4} + 2 \\ &\leq c_1 + c_2 + \frac{1}{2}c_3 + \frac{3}{8}c_4 + \frac{1}{4}c_6 - \frac{1}{4}m_{16} - \frac{1}{2}m_{23} - \frac{3}{8}m_{24} - \frac{1}{4}m_{26} + 3. \end{aligned}$$

A (4.2) – (4.7) egyenletek segítségével kapjuk, hogy

$$\begin{aligned} H_7(L) \leq\ & y_1 + y_{15} + \frac{5}{4}y_{16} + y_2 + \frac{3}{2}y_{23} + \frac{11}{8}y_{24} + y_{25} + \frac{5}{4}y_{26} + \\ & + \frac{5}{4}y_{256} + \frac{3}{2}y_{266} + \frac{1}{2}y_3 + y_{33} + \frac{7}{8}y_{34} + \frac{1}{2}y_{35} + \frac{3}{4}y_{36} + \frac{5}{4}y_{336} + \\ & + \frac{7}{8}y_{345} + \frac{9}{8}y_{346} + \frac{1}{2}y_{355} + \frac{3}{4}y_{356} + y_{366} + \frac{5}{4}y_{3666} + \frac{3}{8}y_4 + \frac{3}{4}y_{44} + \\ & + \frac{3}{8}y_{45} + \frac{5}{8}y_{46} + \frac{3}{4}y_{445} + y_{446} + \frac{3}{8}y_{455} + \frac{5}{8}y_{456} + \frac{7}{8}y_{466} + \frac{7}{8}y_{4566} + \\ & + \frac{9}{8}y_{4666} + \frac{1}{4}y_{56} + \frac{1}{4}y_{556} + \frac{1}{2}y_{566} + \frac{1}{4}y_{5556} + \frac{1}{2}y_{5566} + \frac{3}{4}y_{5666} + \frac{1}{4}y_6 + \\ & + \frac{1}{2}y_{66} + \frac{3}{4}y_{666} + y_{6666} - \frac{1}{4}m_{16} - \frac{1}{2}m_{23} - \frac{3}{8}m_{24} - \frac{1}{4}m_{26} + 3. \end{aligned}$$

Ha elhagyunk illetve növelünk néhány negatív együtthatót, és hozzáadunk néhány nemnegatív értéket a jobb oldalhoz, a (4.1) egyenlet segítségével kapjuk, hogy

$$\begin{aligned} H_7(L) \leq\ & L^* + \frac{1}{4}\Big[2(y_{23}+y_{266}) + \frac{3}{2}y_{24} + y_{16} + y_2 + y_{26} + y_{256} + y_{336} + y_{3666} + \\ & + m_{15} + \frac{1}{2}(y_{346}+y_{4666}) - \frac{1}{2}(y_{345}+y_{4566}) - (m_{23}+m_{24}+m_{26}) - \\ & - (y_{35}+y_{356}+y_{45}+y_{445}+y_{456}+y_5+y_{56}+y_{566}+y_{5666} - \\ & - 2(y_{355}+y_{455}+y_{556}+y_{55}+y_{5566}) - 3(y_{555}+y_{5556})\Big] + 3. \end{aligned}$$

Mivel $\tilde{c}_5 = 0$ és $\tilde{c}_6 > 0$, $m_{26} = c_2 - m_{23} - m_{24} - (c_5 - m_{15})$ teljesül, és így

$$\begin{aligned} H_7(L) \leq\ & L^* + \frac{1}{4}\Big[y_{15} + y_{16} + y_{23} + y_{256} + y_{266} + y_{336} + y_{3666} + \\ & + \frac{1}{2}(y_{24} + y_{345} + y_{346} + y_{4566} + y_{4666})\Big] + 3 \leq \frac{5}{4}L^* + 3. \end{aligned}$$

C eset: $\tilde{c}_5 > 0$, $\tilde{c}_6 > 0$

A 4.3.1. lemma alapján kapjuk $H_7(L)$-re hogy

$$\begin{aligned} H_7(L) \leq\ & c_1 + c_2 + \frac{c_3 - m_{23}}{2} + \left\lfloor \frac{c_4 - m_{24}}{2} \right\rfloor + \frac{c_6 - m_{16} - \left\lfloor \frac{c_4 - m_{24}}{2} \right\rfloor}{4} + \\ & + \left\lfloor \frac{c_5 - m_{15} - m_{25}}{3} \right\rfloor + 2 \\ \leq\ & c_1 + c_2 + \frac{1}{2}c_3 + \frac{3}{8}c_4 + \frac{1}{3}c_5 + \frac{1}{4}c_6 - \frac{1}{3}m_{15} - \frac{1}{4}m_{16} - \\ & - \frac{1}{2}m_{23} - \frac{3}{8}m_{24} - \frac{1}{3}m_{25} + 3. \end{aligned}$$

Egyszerűsítve

$$\begin{aligned} H_7(L) \leq\ & y_1 + \frac{4}{3}y_{15} + \frac{5}{4}y_{16} + y_2 + \frac{3}{2}y_{23} + \frac{11}{8}y_{24} + \frac{4}{3}y_{25} + \frac{5}{4}y_{26} + \frac{19}{12}y_{256} + \frac{3}{2}y_{266} + \\ & + \frac{1}{2}y_3 + y_{33} + \frac{7}{8}y_{34} + \frac{5}{6}y_{35} + \frac{3}{4}y_{36} + \frac{5}{4}y_{336} + \frac{29}{24}y_{345} + \frac{9}{8}y_{346} + \frac{7}{6}y_{355} + \\ & + \frac{13}{12}y_{356} + y_{366} + \frac{5}{4}y_{3666} + \frac{3}{8}y_4 + \frac{3}{4}y_{44} + \frac{17}{24}y_{45} + \frac{5}{8}y_{46} + \frac{13}{12}y_{445} + y_{446} + \\ & + \frac{25}{24}y_{455} + \frac{23}{24}y_{456} + \frac{7}{8}y_{466} + \frac{29}{24}y_{4566} + \frac{9}{8}y_{4666} + \frac{1}{3}y_5 + \frac{2}{3}y_{55} + y_{555} + \\ & + \frac{7}{12}y_{56} + \frac{11}{12}y_{556} + \frac{5}{6}y_{566} + \frac{5}{4}y_{5556} + \frac{7}{6}y_{5566} + \frac{13}{12}y_{5666} + \frac{1}{4}y_6 + \frac{1}{2}y_{66} + \\ & + \frac{3}{4}y_{666} + y_{6666} - \frac{1}{3}m_{15} - \frac{1}{4}m_{16} - \frac{1}{2}m_{23} - \frac{3}{8}m_{24} - \frac{1}{3}m_{25} + 3, \end{aligned}$$

és így

$$\begin{aligned} H_7(L) \leq\ & L^* + \frac{1}{4}\Big[\frac{7}{3}y_{256} + 2(y_{23} + y_{266}) + \frac{3}{2}y_{24} + \frac{4}{3}(y_{15} + y_2 + y_{25} + y_{26}) + \\ & + y_{16} + y_{336} + y_{3666} + y_{5556} + \frac{5}{6}(y_{345} + y_{4566}) + \frac{2}{3}(y_{355} + y_{5566}) + \\ & + \frac{1}{2}(y_{346} + y_{4666}) + \frac{1}{3}(y_{445} + y_{356} + y_{5666}) + \frac{1}{6}y_{455} - \\ & - (m_{15} + m_{16}) - \frac{4}{3}(m_{23} + m_{24} + m_{25})\Big] + 3. \end{aligned}$$

Mivel $m_{23} + m_{24} + m_{25} = c_2$, (4.3)-ból adódik, hogy

$$\begin{aligned} H_7(L) \leq\ & L^* + \frac{1}{4}\Big[\frac{4}{3}y_{15} + y_{16} + y_{256} + y_{336} + y_{3666} + y_{5556} + \\ & + \frac{5}{6}(y_{345} + y_{4566}) + \frac{2}{3}(y_{23} + y_{266} + y_{355} + y_{5566}) + \\ & + \frac{1}{2}(y_{346} + y_{4666}) + \frac{1}{3}(y_{356} + y_{445} + y_{5666}) + \\ & + \frac{1}{6}(y_{24} + y_{455}) - (m_{15} + m_{16})\Big] + 3. \end{aligned}$$

A 4.2.1. lemmát alkalmazva C_1-re és $C_{5,6}$-ra adódik, hogy $y_{15} + y_{16} \leq \frac{1}{2}c_1 + m_{15} + m_{16}$. Ekkor

$$\begin{aligned} H_7(L) \leq\ & L^* + \frac{1}{4}\Big[y_{256} + y_{336} + y_{3666} + y_{5556} + \\ & + \frac{5}{6}(y_{15} + y_{345} + y_{4566}) + \frac{2}{3}(y_{23} + y_{266} + y_{355} + y_{5566}) + \\ & + \frac{1}{2}(y_1 + y_{16} + y_{346} + y_{4666}) + \\ & + \frac{1}{3}(y_{356} + y_{445} + y_{5666}) + \frac{1}{6}(y_{24} + y_{455})\Big] + 3 \\ \leq\ & \frac{5}{4}L^* + 3. \end{aligned}$$

Ebből adódik a tétel állítása erre az esetre is.

D eset: $\tilde{c}_5 > 0$, $\tilde{c}_6 = 0$

A H_7 számára szükséges ládák száma ebben az esetben

$$H_7(L) \leq c_1 + c_2 + \left\lfloor \frac{c_3 - m_{23}}{2} \right\rfloor + \left\lfloor \frac{c_4 - m_{24}}{2} \right\rfloor + \left\lfloor \frac{c_5 - m_{15} - m_{25} - m_{445}}{3} \right\rfloor + 2$$
$$\leq c_1 + c_2 + \frac{1}{2}(c_3 + c_4 - m_{23} - m_{24}) + \frac{1}{3}(c_5 - m_{15} - m_{25} - m_{445}) + 3.$$

(4.2) - (4.6)-ból könnyen igazolható, hogy

$$\begin{aligned} H_7(L) \leq\ & y_1 + \frac{4}{3}y_{15} + y_{16} + y_2 + \frac{3}{2}y_{23} + \frac{3}{2}y_{24} + \frac{4}{3}y_{25} + y_{26} + \\ & + \frac{4}{3}y_{256} + y_{266} + \frac{1}{2}y_3 + y_{33} + y_{34} + \frac{5}{6}y_{35} + \frac{1}{2}y_{36} + y_{336} + \\ & + \frac{4}{3}y_{345} + y_{346} + \frac{7}{6}y_{355} + \frac{5}{6}y_{356} + \frac{1}{2}y_{366} + \frac{1}{2}y_{3666} + \frac{1}{2}y_4 + y_{44} + \\ & + \frac{5}{6}y_{45} + \frac{1}{2}y_{46} + \frac{4}{3}y_{445} + y_{446} + \frac{7}{6}y_{455} + \frac{5}{6}y_{456} + \frac{1}{2}y_{466} + \frac{5}{6}y_{4566} + \\ & + \frac{1}{2}y_{4666} + \frac{1}{3}y_5 + \frac{2}{3}y_{55} + \frac{1}{3}y_{56} + y_{555} + \frac{2}{3}y_{556} + \frac{1}{3}y_{566} + y_{5556} + \\ & + \frac{2}{3}y_{5566} + \frac{1}{3}y_{5666} - \frac{1}{3}m_{15} - \frac{1}{2}m_{23} - \frac{1}{2}m_{24} - \frac{1}{3}m_{25} - \frac{1}{3}m_{445} + 3. \end{aligned}$$

A (4.1) egyenlet segítségével adódik H_7-re, hogy

$$\begin{aligned} H_7(L) \leq\ & L^* + \frac{1}{4}\Big[2\,(y_{23} + y_{24}) + \frac{4}{3}\,(y_{15} + y_{25} + y_{256} + y_{345} + y_{445}) + \\ & + \frac{2}{3}\,(y_{355} + y_{455}) - \frac{2}{3}\,(y_{35} + y_{356} + y_{45} + y_{456} + y_{4566}) - \\ & - \frac{4}{3}\,(y_{55} + y_{556} + y_{5566}) - 2\,(y_3 + y_{36} + y_{366} + y_{3666} + y_4 + y_{46} + \\ & + y_{466} + y_{4666}) - \frac{8}{3}\,(y_5 + y_{56} + y_{566} + y_{5666}) - 4\,(y_6 + y_{66} + y_{666} + \\ & + y_{6666}) - 2\,(m_{23} + m_{24}) - \frac{4}{3}\,(m_{15} + m_{25} + m_{445})\Big] + 3. \end{aligned} \tag{4.8}$$

Ha a 4.2.1. lemmát alkalmazzuk C_1-re és $C_{5,6}$-ra kapjuk, hogy $y_{15} + y_{16} \leq \frac{c_1}{2} + m_{15} + m_{16}$ ami ekvivalens az $\frac{1}{3}y_1 - \frac{1}{3}y_{15} - \frac{1}{3}y_{16} + \frac{2}{3}m_{15} + \frac{2}{3}m_{16} \geq 0$ egyenlőtlenséggel. A (4.8) egyenlőtlenség a következőképpen írható fel:

$$\begin{aligned} H_7(L) \leq\ & L^* + \frac{1}{4}\Big[2\,(y_{23} + y_{24}) + \frac{4}{3}\,(y_{25} + y_{256} + y_{345} + y_{445}) + y_{15} + \\ & + \frac{2}{3}\,(y_{355} + y_{455}) + \frac{1}{3}y_1 - \frac{1}{3}y_{16} - \frac{2}{3}\,(y_{35} + y_{356} + y_{45} + y_{456} + y_{4566}) - \end{aligned}$$

$$-\frac{4}{3}(y_{55}+y_{556}+y_{5566})-2(y_3+y_{36}+y_{366}+y_{3666}+y_4+y_{46}+ \quad (4.9)$$
$$+y_{466}+y_{4666})-\frac{8}{3}(y_5+y_{56}+y_{566}+y_{5666})-4(y_6+y_{66}+y_{666}+$$
$$+y_{6666})+\frac{2}{3}m_{16}-\frac{2}{3}m_{15}-\frac{4}{3}(m_{25}+m_{445})-2(m_{23}+m_{24})\Big]+3.$$

A D esetben nem fogynak el a C_5-elemek a 4. lépésben, és így

$$m_{23}+m_{24}+m_{25}-c_2=0. \quad (4.10)$$

Nyilvánvaló, hogy

$$m_{16}\leq c_6 \quad (4.11)$$

Ha van legalább egy olyan hármas a 8. lépésben amely elhelyezhető egy ládában, akkor használhatjuk a 4.3.2. lemmát. A megfelelő egyenlőtlenséget $\frac{2}{3}$-dal megszorozva, (4.11)-gyel együtt adja, hogy

$$\frac{1}{6}c_4+\frac{4}{3}m_{445}+\frac{2}{3}m_{15}+\frac{1}{2}m_{24}-\frac{2}{3}y_{345}-\frac{2}{3}y_{445}\geq 0 \quad (4.12)$$

Ezután megszorozzuk (4.10)-et $\frac{2}{3}$-dal, és (4.12)-vel együtt beszúrjuk (4.9)-be. Felhasználva (4.11)-et, a kívánt egyenlőtlenség adódik. Vegyük észre, hogy az összes negatív együtthatójú tagot elhagytuk.

$$\begin{aligned} H_7(L) \leq\ & L^*+\frac{1}{4}\Big[y_{15}+y_{445}+\frac{5}{6}(y_{24}+y_{345}+y_{346}+y_{446}+ \\ & +y_{455}+y_{4566})+\frac{2}{3}(y_{23}+y_{256}+y_{336}+y_{355}+y_{5556})+ \\ & +\frac{1}{3}(y_1+y_{16}+y_{44})+\frac{1}{6}(y_{34}+y_{456}+y_{4666})\Big]+3. \end{aligned}$$

Ezután feltehetjük, hogy az összes a 8. lépésben összeillesztett hármas belefér a hozzátartozó ládába. Ez azt jelenti, hogy $m_{445}=k_3$. Két esetet különböztetünk meg. Ha $m_{445}=\left\lceil\frac{c_4}{4}-\frac{m_{24}}{4}-\frac{c_6}{2}+\frac{m_{16}}{2}\right\rceil$, akkor ezt átírva

$$\frac{4}{3}m_{445}-\frac{1}{3}c_4+\frac{1}{3}m_{24}+\frac{2}{3}c_6-\frac{2}{3}m_{16}\geq 0. \quad (4.13)$$

Ezúttal (4.10)-et $\frac{4}{3}$-dal szorozzuk és (4.13)-mal együtt hozzáadjuk (4.9)-hez.
A másik eset a következőképpen formalizálható:

$$\frac{2}{3}m_{445} - \frac{1}{3}c_5 + \frac{1}{3}m_{15} + \frac{1}{3}m_{25} \geq 0. \tag{4.14}$$

Ekkor (4.10)-et és (4.14)-et adjuk hozzá (4.9)-hez. m_{16}-ot szintén helyettesítjük, (4.11)-nek megfelelően. Felhasználva (4.5)-öt, (4.6)-ot és (4.7)-et ellenőrizhető, hogy $H_7(L) \leq \frac{5}{4}L^* + 3$ teljesül mindkét esetben. Ezzel az utolsó eset bizonyítását is befejeztük.

Tételünk tehát igaz minden olyan listára, amely nem tartalmaz C_0 és C_7-elemeket. Tegyük fel most, hogy L egy olyan lista, amely tartalmaz C_0-elemeket, de nem tartalmaz C_7-elemeket. Ekkor L bármely pakolásában minden egyes C_0-elem egyedül van egy ládában. Legyen $L_0 = L \setminus C_0$. Kapjuk, hogy $H_7(L_0) = H_7(L) - c_0$, és $L_0^* = L^* - c_0$. Az előzőek alapján, $H_7(L_0) \leq \frac{5}{4}L^* + 3$. Ezért, $H_7(L) = H_7(L_0) + c_0 \leq \frac{5}{4}L_0^* + 3 + c_0 \leq \frac{5}{4}L^* + 3$.

Így a tétel igaz az olyan listákra is, amelyek C_0-elemeket is tartalmaznak. Tekintsünk most egy olyan L listát, amely C_7-elemeket is tartalmaz. Ekkor két lehetőség van:

Ha a H_7 nyit új ládákat a 12. lépésben, akkor minden egyes láda - kivéve esetleg az utolsót - legalább $\frac{4}{5}$ részig tele kell hogy legyen. Így, $L^* \leq \frac{4}{5}(H_7(L) - 1)$, amiből adódik a kívánt eredmény. Következésképpen feltehetjük, hogy a H_7 algoritmus nem nyit új ládákat a 12. lépésben. Legyen $L_7 = L \setminus C_7$. Nyilvánvaló, hogy $H_7(L_7) = H_7(L)$, és $L_7^* \leq L^*$. Mivel az L_7 lista nem tartalmaz C_7-elemet, ezért az egyenlőtlenség teljesül tetszőleges listára is.

Most már csak azt kell belátnunk, hogy van végtelen sok olyan lista, amelyre egyenlőség teljesül. Ezt az alábbi egyszerű konstrukcióval tehetjük meg:

Legyen $n = 3m$ és $m \in \mathbb{N}$. Tekintsük azon L_n listákat, amelyek $2m$ darab C_3-elemet és m darab C_6-elemet tartalmaznak, melyek mérete $3/8 + \varepsilon$ illetve

$1/4 - 2\varepsilon$. Ekkor, $H_7(L_n) = \frac{m}{2} + \frac{3m}{4} = \frac{5}{4}m$, és $L_n^* = m$. Ebből következik, hogy $\frac{H_7(L_n)}{L_n^*} = 5/4$ minden L_n listára.

□

LUEKER és DE LA VEGA eredményei miatt nyilvánvaló, hogy az eredmény megjavítható, és megadható olyan lineáris időbonyolultságú algoritmus, ami $\frac{6}{5}$, vagy annál is jobb aszimptotikus legrosszabb-eset hányadossal rendelkezik. Úgy véljük, hogy bár az algoritmus lineáris marad (mint a DE LA VEGA, LUEKER algoritmus), azonban nagyon bonyolulttá válhat és lehet, hogy az esetek nagy száma miatt a bizonyítás hosszúsága "exponenciálisan" fog növekedni.

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).

Summary

This dissertation deals with two different topics of combinatorial optimization. The first one is data compression, the second one is bin packing. In this thesis we are concerned with a worst-case analysis of some on–line or off–line heuristics for the above mentioned problems.

Usually *optimizing* means finding the maximum or minimum of a certain function, defined on some domain. Classical theories of optimization deal with the case when this domain is infinite. In case of *combinatorial optimization* typical problem instances consist of the maximization or minimization of some objective function over a *finite* feasible set.

Often, when the objective function is too "wild", constraints are too complicated, or the problem size is too large it is impossible to find an optimal solution. Mathematicians and computer scientists have developed theories to make intuitive assertions about the difficulty of certain problems. This is the theory of *NP-completeness.*

In cases when the optimal solutions are too hard to find, algorithms (so called *heuristics*) can often be designed that produce approximately optimal solutions. It is important that these *suboptimal solutions* have a guaranted quality; e.g., for a given maximization problem, the value of the heuristic solution is at least 90% of the optimum for every input.

To measure the efficiency of a heuristic, we can use *worst-case analysis.* Using this technique we can derive results, which hold for every individual problem

instance. To give the exact value of the behaviour of a heuristic, we can use the *asymptotic worst-case ratio*, i.e. the limes superior of the ratios of the solution values given by the heuristic and the optimal algorithm, while the size of the problems goes to infinity.

Another important aspect is the *on-line* solution of combinatorial optimization problems. In many practical situations data come in one by one and decision must be made before the next piece of data arrives. This kind of heuristics are called on–line algorithms. When all data are known before the optimization algorithm is called we talk about *off–line algorithms.*

The second chapter of the dissertation gives a detailed introduction to data compression. It presents the basic definitions, the most important theorems and compression methods.

Compression means making things smaller by applying pressure. Data compression is not about physically squashing data, but about finding ways to represent it in fewer bits or bytes. In this thesis we deal only with those case, where the original data can be exactly reconstitute from the compressed form. There are many other kinds of data reduction, such as voice and picture coding, where some degradation of quality may be tolerable if a more compact representation is thereby achieved.

In the second chapter we introduce fundamental concepts from information theory. We peresent the probabilistic models that are used for data compression. The well–known Huffman's algorithm and the arithmetic coding are among the topics of this of this chapter.

A different approach from statistical methods of modeling and coding is *dictionary encoding.* This kind of coding uses a dictionary to translate the original string to a shorter one. The dictionary consists of ordered pairs *(source–word, code–word)*, which are used to replace a substring in the original string. In the second chapter we describe the very popular Ziv-Lempel algorithm, which uses dictionary encoding.

In the third chapter we are concerned with a worst–case analysis of some on–line heuristics for data compression. These algorithms are based on dictionary encoding.

In this dissertation we consider only methods which use a *static dictionary*, i.e. a fixed dictionary, that cannot be changed or extended during the encoding–decoding process. Our aim is to translate (encode) the source string with the help of dictionary strings into a code–text with minimal length. The problem defined by the above setup is equivalent to the problem of finding a *shortest–path* in a related directed, edge–weighted graph. It is straight forward to see that the problem of finding an optimal compression is equivalent to the computation of a shortest–path in a suitable graph.

If the graph has many *cut vertices* (i.e. vertices which divide the original problem into independent subproblems) and in case that these subproblems are reasonably small, we can solve the problem efficiently and compute the optimal encoding. Unfortunately, in practice this will not be the case and an optimal algorithm cannot be applied as dealing with very long strings would take too much time and storage capacity. Therefore, many *on–line heuristics* have been developed to derive near optimal solutions.

The third chapter of the thesis deals with the worst–case analysis of four different on–line heuristics. It solves some open problems for well–known heuristics (*Longest Matching* and *Differential Greedy*) using different type of dictionaries. Based on experimental reasults a new heuristic (*Fractional Greedy*) is defined and analysed for those types of dictionaries.

The results show that these algorithms have similar behaviour except some special cases. We can conclude that the type of a dictionary plays an important role in the behaviour of a heuristic. An algorithm can be optimal for some type of dictionaries and very bad for other ones.

Because on–line algorithms have only very restricted information about the string to be compressed, we analysed an algorithm (*Longest Fragment First*),

which has not been strictly on–line. This algorithm were analysed experimentally before, but theorectical results have not been known since now. In the last section of this chapter we prove some results for this heuristic. In this case we use more information to compress a string, so the results are obviously better.

The fourth chapter of the dissertation deals with *1–dimensional bin packing problem.* In this problem we are given a list $L = \{x_1, x_2, ..., x_n\}$ of real numbers from $[0, 1)$ and an infinite list of unit capacity bins. Each number x_i has to be assigned to a unique bin such that the sum of the elements in each bin does not exceed 1. Our aim is to minimize the number of used bins. It is well–known that finding an optimal packing is $\mathcal{NP}$-hard. Consequently, a lot of papers have been published which look for polynomial time algorithms with an acceptable behaviour. As we mentioned before algorithms can be on–line and off–line. In this dissertation we deal with off–line algorithms. The off–line algorithms know the whole list before they apply their strategy to pack the items. For measuring the efficiency we use the asymptotic worst–case ratio.

Most of the published algorithms have at least $O(n \log n)$ time–complexity (see for example the *First Fit Decreasing, Best Fit Decreasing* heuristics). Linear time complexity algorithms were always "refreshing exceptions" among the other experiments. It is well–known that for every $\varepsilon > 0$ there is an algorithm A such that $A(L) \leq (1 + \varepsilon) L^* + C_\varepsilon$, and A runs in time $O(n) + D_\varepsilon$. Here $A(L)$ means the number of bins used by algorithm A, and L^* means the number of bins in an optimal packing. It is remarkable that the constants D_ε and C_ε depend on ε only but not on n. Unfortunately these constants may be very large, namely they grow exponentially in $1/\varepsilon$. This has a consequence: The theoretically excellent algorithm is not usable in practice. The best linear time algorithm, which could be used in practice, have had 4/3 asymptotic worst–case ratio until now.

In the last chapter we present a linear time off–line bin packing algorithm, which has a 5/4 *asymptotic worst–case ratio.* The algorithm is based on a *classification of the elements* and a *pairing technique* originally due to MARTEL.

Large parts of the dissertation draw upon publications that appeared earlier or are accepted for publication. These papers are joint works with GÁBOR GALAMBOS, HANS KELLERER, ULRICH PFERSCHY, TIMO RAITA and GERHARD WOEGINGER.