PTAS for Huffman coding with unequal letter costs

Mordecai Golin (HKUST), Claire Mathieu (Brown)
and Neal E. Young (University of California, Riverside)

February 12, 2009

introduction

Huffman coding

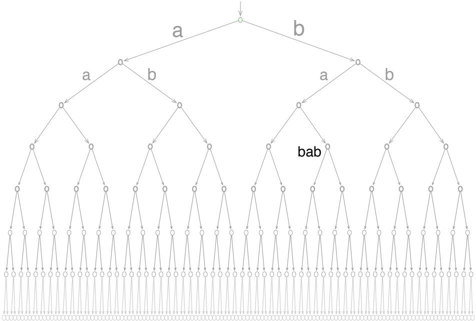
Huffman coding with unequal letter costs

A polynomial-time approximation scheme

Open questions.

Huffman coding

n frequencies
$p_1 = 4$
$p_2 = 4$
$p_3 = 2$
$p_4 = 1$
$p_5 = 1$



given: frequencies $p_1 \geq p_2 \geq \cdots \geq p_n$

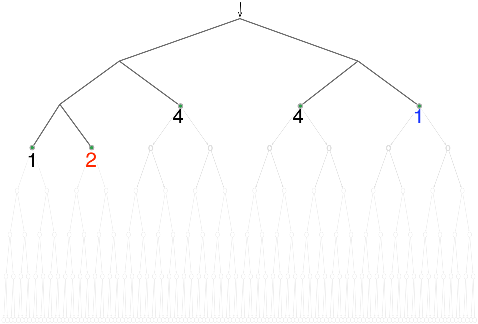
find: binary codewords $w_1, w_2, \ldots, w_n$

objective: minimize wtd average codeword length $\sum_i p_i \, |w_i|$

prefix-free: no codeword is a prefix of any other codeword

A **prefix-free** code of cost 27

frequency $\rightarrow$ "word"	
4 $\rightarrow$ "ab",	cost 8
4 $\rightarrow$ "ba",	cost 8
2 $\rightarrow$ "aab",	cost 6
1 $\rightarrow$ "aaa",	cost 3
1 $\rightarrow$ "bb",	cost 2
	27



given: frequencies $p_1 \geq p_2 \geq \cdots \geq p_n$

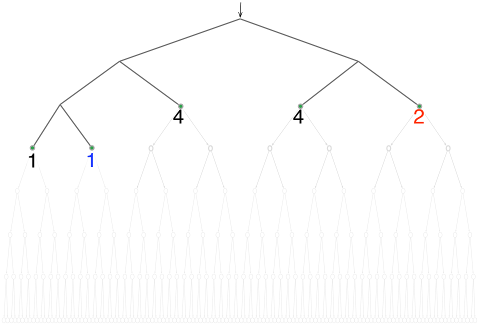
find: binary codewords $w_1, w_2, \ldots, w_n$

objective: minimize wtd average codeword length $\sum_i p_i |w_i|$

prefix-free: no codeword is a prefix of any other codeword

A **monotone** prefix-free code (lower cost)

4 $\longrightarrow$ "ab"
4 $\longrightarrow$ "ba"
2 $\longrightarrow$ "bb"
1 $\longrightarrow$ "aaa"
1 $\longrightarrow$ "aab"

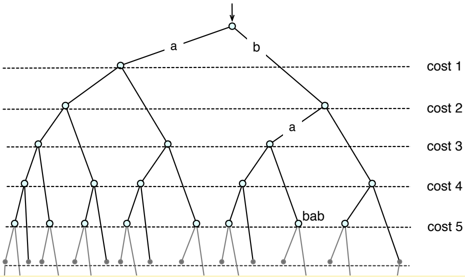


Highest frequencies are assigned to shortest codewords.

Huffman coding with **unequal letter costs**

$p_1 = 4$
$p_2 = 4$
$p_3 = 2$
$p_4 = 1$
$p_5 = 1$

each "a" costs 1
each "b" costs 2



given: letter costs $\ell_0 \le \ell_1$ ... *in general case can have more than two letters*
frequencies $p_1 \ge p_2 \ge \cdots \ge p_n$

find: binary codewords $w_1, w_2, \ldots, w_n$

objective: minimize wtd average codeword cost, $\sum_i p_i \mathit{cost}(w_i)$

prefix-free: no codeword is a prefix of any other codeword

Doris Altenkamp and Kurt Melhorn.
Codes: Unequal probabilies, unequal letter costs.
JACM, 27(3):412–427, July 1980.

N. M. Blachman.
Minimum cost coding of information.
IRE Transactions on Information Theory, PGIT-3:139–149, 1954.

N. Cot.
Complexity of the variable-length encoding problem.
6th Southeast Conference on Combinatorics, Graph Theory and Computing, pages 211–244, 1975.

Norbert Cott.
Characterization and Design of Optimal Prefix Codes.
PhD Thesis, Stanford University, June 1957.

I. Csisz'ar.
Simple proofs of some theorems on noiseless channels.
Inform. Contr., 514:285–298, 1969.

E. N. Gilbert.
How good is morse code.
Inform. Control, 14:585–565, 1969.

E. N. Gilbert.
Coding with digits of unequal costs.
IEEE Trans. Inform. Theory, 41:596–600, 1995.

Richard Karp.
Minimum-redundancy coding for the discrete noiseless channel.
IRE Trans. on Information Theory, IT-7:27–39, January 1961.

R. M. Krause.
Channels which transmit letters of unequal duration.
Inform. Contr., 5:13–24, 1962.

Abraham Lempel, Shimon Even, and Martin Cohen.
An algorithm for optimal prefix parsing of a noiseless and memoryless channel.
IEEE Trans. on Information Theory, 19(2):208–214, March 1973.

R.S. Marcus.
Discrete Noiseless Coding.
M.S. Thesis, MIT, 1957.

K. Mehlhorn.
An efficient algorithm for constructing nearly optimal prefix codes.
IEEE Trans. Inform. Theory, 26:513–517, September 1980.

L. E. Stanfel.
Tree structures for optimal searching.
JACM, 17(3):508–517, July 1970.

NP-hard? c-approx?

PTAS (main result)

Theorem (GMY - STOC 2002)

For Huffman coding with unequal letter costs, for any fixed $\varepsilon > 0$, a $(1+\varepsilon)$-approximate solution can be computed in time poly(n).

algorithm

1. Scale and round the letter costs.
2. Find a minimum-cost t-relaxed code c.
3. "Round" c to make it prefix free.

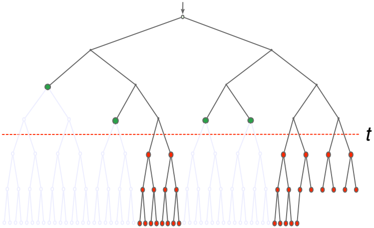
algorithm

1. Scale and round the letter costs.
2. Find a minimum-cost *t-relaxed* code c.
3. "Round" c to make it prefix free.

t-relaxed: words of cost $\geq t$ *can* be prefixes of other words

4 codewords cost $< t$:

31 codewords cost $\geq t$:



Lemma (lower bound on opt)

cost(optimal t-relaxed code) $\leq$ *cost(optimal prefix-free code)*

will take $t = O_\varepsilon(1)$ — a constant (dependent on ε)

algorithm

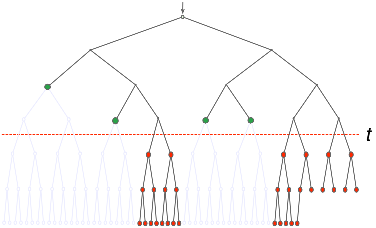
1. Scale and round the letter costs.
2. Find a minimum-cost *t-relaxed* code c.
3. "Round" c to make it prefix free.

finding a minimum-cost t-relaxed code

choose words of cost $< t$
by exhaustive search

$t \approx \log(1/\varepsilon)/\varepsilon \longrightarrow$

choose words of cost $\geq t$
greedily



exhaustive search: *...for dealing with bigger-than binary alphabets*

In each level $1, 2, .., t$, only *number* of codewords matters.
$\Rightarrow$ at most n^t equivalence classes of codes.
$\Rightarrow$ $n^{O(t)}$ time to search them all.

algorithm
1. Scale and round the letter costs.

2. Find a minimum-cost t-relaxed code c.

3. "Round" c to make it prefix free.

Making a t-relaxed code prefix free:

for each codeword w of cost $\geq t$:

Split w as $w = x\,y$ where $\text{cost}(x) \approx t$.

Replace w with $w' = x\,|y|\,y$, where $|y|$ is encoded in binary.

example: $w =$ aabaaababaaabbaaabbaaab
→ aabaaaba1100baaabbaaabbaaab
→ aabaaababbbbaaaaabbaaabbaaabbaaab

Lemma: Cost of code increases by $1 + O(\varepsilon)$ factor.

Cost of w increases by $2\log_2 \text{cost}(w)$.
Increase is at most $\varepsilon\,\text{cost}(w)$ since $\text{cost}(w) \geq t \approx \log(1/\varepsilon)/\varepsilon$.

algorithm

1. Scale and round the letter costs.
2. Find a minimum-cost t-relaxed code c.
3. "Round" c to make it prefix free.

Theorem

The cost of the code produced by the algorithm is at most $(1 + O(\varepsilon))$ *times the minimum cost of any prefix-free code.*

Proof.

$\text{cost}(c)$ is at most the minimum cost of any prefix-free code.
Making c prefix-free increases its cost by a $1 + O(\varepsilon)$ factor. □

Run time: $O(n \log n) + O(f(\varepsilon) \log^2 n)$ [GMY - 2009]

Still open...

NP-hard? In P?