CS 170 DIS 06

Released on 2019-2-25

1 Horn Formula Practice

Find the variable assignment that solves the following horn formulas:

1. $(w \wedge y \wedge z) \Rightarrow x, (x \wedge z) \Rightarrow w, x \Rightarrow y, \Rightarrow x, (x \wedge y) \Rightarrow w, (\bar{w} \vee \bar{x} \vee \bar{y}), (\bar{z})$
2. $(x \wedge z) \Rightarrow y, z \Rightarrow w, (y \wedge z) \Rightarrow x, \Rightarrow z, (\bar{z} \vee \bar{x}), (\bar{w} \vee \bar{y} \vee \bar{z})$

Solution:

1.
- Set everything to false initially
- x must be true since we have the statement $\Rightarrow x$
- y must be true since $x \Rightarrow y$
- w must be true since $(x \wedge y) \Rightarrow w$)
- Not all negative clauses are satisfied at this point, so there is no satisfying assignment.

2.
- z must be true since we have the statement $\Rightarrow z$.
- w must be true since $z \Rightarrow w$
- x and y need not be changed, as all our implications are satisfied.
- All negative clauses are now satisfied, so we've found our satisfying assignment.

2 Longest Huffman Tree

Under a Huffman encoding of n symbols with frequencies $f_1, f_2, \ldots, f_n$, what is the longest a codeword could possibly be? Give an example set of frequencies that would produce this case, and argue that it is the longest possible.

Solution: The longest codeword can be of length $n-1$. An encoding of n symbols with $n-2$ of them having probabilities $1/2, 1/4, \ldots, 1/2^{n-2}$ and two of them having probability $1/2^{n-1}$ achieves this value. No codeword can ever by longer than length $n-1$. To see why, we consider a prefix tree of the code. If a codeword has length n or greater, then the prefix tree would have height n or greater, so it would have at least $n+1$ leaves. Our alphabet is of size n, so the prefix tree has exactly n leaves.

3 Proof of Huffman Coding

In this question, we will prove that Huffman coding indeed produces the best prefix-free code for a given set of characters and associated frequencies. Recall that we are given as input a set of characters $c_1, \ldots, c_n$ and frequencies $f_1, \ldots, f_n$ and the goal is produce a binary tree T where the leaves of the tree correspond to the characters c_i which is as efficient as possible. That is, the tree produced should minimize $\sum_{i=1}^{n} f_i d_T(c_i)$ where $d_T(c_i)$ denotes the depth of c_i in the tree, T. For this question, we will view Huffman coding as a recursive algorithm which proceeds along the following lines:

1. Merge the two characters with the lowest frequencies, say c_1 and c_2, to produce a "meta-character", (c_1, c_2).
2. Run the Huffman tree procedure on the set of characters $(c_1, c_2), c_3, \ldots, c_n$ with frequencies $(f_1 + f_2), f_3, \ldots, f_n$.
3. Let the tree obtained in the previous step be $T^\dagger$. Replace the node corresponding to (c_1, c_2) with an internal node with two children c_1 and c_2 to produce the final tree T.

(a) For the first part of the question, we will prove that every internal node of the optimal tree, T^*, has two children. (*Hint: Does a violation of this property create a contradiction?*)

(b) Now, let c_1 and c_2 be the two characters with the lowest frequencies. Prove that the cost of the optimal tree, T^*, can only reduce if c_1 and c_2 are made siblings in the lowest leaves of the tree.

(c) Conclude via induction that Huffman coding indeed produces the optimal tree. (*Hint: Can you relate the cost of the tree, T, produced by Huffman coding to the cost of $T^\dagger$?*)

Solution:

(a) Suppose for the sake of contradiction that the internal node v does not have two children. Let c and p be the child and parent of v respectively. We can now improve the cost of T^* by simply deleting the internal node v and connecting c directly to p as all the descendants of v decrease their depths by 1.

(b) Let c_i, c_j be siblings in the two lowest leaves of T^* satisfying without loss of generality $f_i \leq f_j$. Therefore, we have $f_1 \leq f_i$ and $f_2 \leq f_j$. Let T' be formed by exchanging the positions of c_i with c_1 and c_j with c_2. Let $C_T = \sum_{i=1}^{n} f_i d_T(c_i)$ denote the cost of tree T. We now have $C_{T'} - C_{T^*} = (f_1 - f_i)(d_{T^*}(c_i) - d_{T^*}(c_1)) + (f_2 - f_j)(d_{T^*}(c_j) - d_{T^*}(c_2))$. Since $f_1 \leq f_i$ and $f_2 \leq f_j$ and the fact that c_i and c_j occupy the two lowest leaves in T^*, $C_{T'} - C_{T^*} \leq 0$. Therefore, T' is also an optimal tree as its cost is at most that of T^*.

(c) To conclude the proof of correctness of Huffman coding, let $P(k)$ be the statement that Huffman coding produces an optimal tree given any set of k characters and frequencies.

Base Case: The base case when $k = 1$ is trivially true.

Inductive Step: We will prove $P(n)$ holds true assuming $P(n-1)$ is true. First, let $T^\dagger$ be the tree produced by the Huffman coding algorithm with input $(c_1, c_2), \ldots, c_n$ with frequencies $(f_1 + f_2), \ldots, f_n$ and let $C_{T^\dagger} = (f_1 + f_2) d_{T^\dagger}((c_1, c_2)) + \sum_{i=3}^{n} d_{T^\dagger}(c_i) f_i$ (That is $C_{T^\dagger}$ is the cost of $T^\dagger$ with the $n-1$ character set given as input.). Therefore, the cost, C_T of the tree returned by Huffman coding is $C_T = C_{T^\dagger} + (f_1 + f_2)$.

Now, consider the optimal tree, T^* for the n character input. From the previous part, we can assume that c_1 and c_2 are siblings occupying the two lowest leaves, l_1 and l_2 in T^* with parent p. We can obtain from T^* a tree $T^\ddagger$ over the $n-1$ characters $(c_1 + c_2), \ldots, c_n$ with frequencies $(f_1 + f_2), \ldots, f_n$ by deleting v and its children and replacing it with a single node for (c_1, c_2). We have by the same argument as before, $C_{T^*} = C_{T^\ddagger} + (f_1 + f_2)$.

Putting the results together, we get:

$$C_T - C_{T^*} = C_{T^\dagger} - C_{T^\ddagger} \leq 0$$

where we know $C_{T^\dagger} \leq C_{T^\ddagger}$ as $T^\dagger$ is the optimal tree for the $n-1$ character set from the inductive hypothesis.