VARN CODES AND GENERALIZED FIBONACCI TREES

Julia Abrahams
Mathematical Sciences Division, Office of Naval Research, Arlington, VA 22217-5660
(*Submitted June 1993*)

INTRODUCTION AND BACKGROUND

Varn's [6] algorithm solves the problem of finding an optimal code tree, optimal in the sense of minimum average cost, when the code symbols are of unequal cost and the source symbols are equiprobable. He addresses both exhaustive codes, for which the code tree is a full tree, as well as nonexhaustive codes, but only the exhaustive case will be of concern here. In particular, for code symbol costs $c(1) \le c(2) \le \cdots \le c(r)$ and a uniform source of size N, where $(N-1)/(r-1)$ is an integer, the Varn code tree is generated as follows. Start with an r-ary tree consisting of a root node from which descend r leaf nodes labeled from left to right by $c(1), c(2), \ldots, c(r)$, the costs associated with the corresponding code symbols. Select the lowest cost node, let c be its cost, and let descend from it r leaf nodes labeled from left to right by $c+c(1), c+c(2), \ldots, c+c(r)$. Continue, by selecting the lowest cost node from the new tree, until N leaf nodes have been created.

Horibe [3] studied a sequence of binary trees and showed that each tree in the sequence is a Varn code tree for $c(1)=1, c(2)=2$. In particular, the k^{th} tree has the $k-1^{\text{st}}$ tree as its left subtree and the $k-2^{\text{nd}}$ tree as its right subtree; for $k=1$ and $k=2$, the tree is only the root; $c(1)$ is associated with the left descendant of a node and $c(2)$ with the right descendant. These trees are called Fibonacci trees, and the number of leaves in the k^{th} tree is the k^{th} Fibonacci number. Note that some integers N are not equal to the k^{th} Fibonacci number for any k so that not every Varn code tree for $c(1)=1, c(2)=2$ is a Fibonacci tree.

Chang [1] studied a sequence of r-ary trees that reduces to Horibe's sequence of Fibonacci trees for $r=2$. In particular, the k^{th} tree has the $k-i^{\text{th}}$ tree as its i^{th} leftmost subtree, $i=1,\ldots,r$; for $k=1,\ldots,r$, the tree is only the root; and $c(i)=i, i=1,\ldots,r$ is associated with the descendants of a node in left to right order. For these particular costs, $c(i)=i, i=1,\ldots,r$, Chang's trees are Varn code trees, and the number of leaves in the k^{th} tree is determined according to an integer sequence that general-izes the Fibonacci sequence.

It is the purpose of this note to examine sequences of trees that are recursively constructed and are Varn code trees for integer costs $c(1),\ldots,c(r)$ whose greatest common divisor is 1. Since common factors shared by all costs do not affect Varn's algorithm, the costs considered here are essentially all rational costs or all sets of rational costs with a common irrational multiplier. Thus, previous work on recursive characterizations of Varn code trees for particular integer code symbol costs is extended to the case of arbitrary integer code symbol costs.

RECURSIVE CONSTRUCTION OF TREES

For fixed integer costs $c(1) \le c(2) \le \cdots \le c(r)$ with greatest common divisor 1, we will have $c(r)$ "types" of leaf nodes denoted by $a_1, a_2, \ldots, a_{c(r)}$. The $k+1^{\text{st}}$ tree $T(k+1)$ is constructed from the previous tree $T(k)$ according to the following set of rules. A leaf node of type a_1 in $T(k)$ will be replaced by r descendant nodes of types $a_{c(1)}, a_{c(2)}, \ldots, a_{c(r)}$ in left to right order in $T(k+1)$. A node of type a_j in $T(k)$ will be replaced by a node of type a_{j-1} in $T(k+1)$, $j=2$, ...,

$c(r)$. The sequence of trees begins with $T(1)$, which consists of a single root node of type $a_{c(r)}$. This construction generalizes Horibe [3] and Chang [1].

An example of trees constructed in this fashion is given in Table 1 for the costs $c(1) = 2$, $c(2) = 3$, $c(3) = 3$, $c(4) = 5$. The trees are described using the following compact notation. Sibling nodes in left to right order are separated by + signs, and parentheses are used to indicate depth in tree from the root so that, for example, $((a_2 + a_3 + a_3 + a_5) + a_1 + a_1 + a_3)$ denotes a 4-ary tree with 4 depth 2 leaves descending from the root through a common intermediate node and 3 depth 1 leaves descending from the root in left to right order and labeled according to type in left to right order as $a_2, a_3, a_3, a_5, a_1, a_1, a_3$, respectively.

TABLE 1. $\boldsymbol{T(k)}$ **for** $\boldsymbol{c(1) = 2, c(2) = 3, c(3) = 3, c(4) = 5}$

k	$T(k)$
1	a_5
2	a_4
3	a_3
4	a_2
5	a_1
6	$(a_2 + a_3 + a_3 + a_5)$
7	$(a_1 + a_2 + a_2 + a_4)$
8	$((a_2 + a_3 + a_3 + a_5) + a_1 + a_1 + a_3)$
9	$((a_1 + a_2 + a_2 + a_4) + (a_2 + a_3 + a_3 + a_5) + (a_2 + a_3 + a_3 + a_5) + a_2)$
10	$(((a_2 + a_3 + a_3 + a_5) + a_1 + a_1 + a_3) + (a_1 + a_2 + a_2 + a_4) + (a_1 + a_2 + a_2 + a_4) + a_1)$
11	$(((a_1 + a_2 + a_2 + a_4) + (a_2 + a_3 + a_3 + a_5) + (a_2 + a_3 + a_3 + a_5) + a_2)$ $+ ((a_2 + a_3 + a_3 + a_5) + a_1 + a_1 + a_3) + ((a_2 + a_3 + a_3 + a_5)$ $+ a_1 + a_1 + a_3) + (a_2 + a_3 + a_3 + a_5))$
...	...

By induction, $T(k), k > c(r)$, has $T(k - c(i))$ as its i^{th} leftmost subtree, $i = 1, \ldots, r$. Because of the recursive tree construction, it is easy to give recurrence relations for the number of leaf nodes of each type in $T(k)$. Use $f^j(k)$ to denote the number of leaves of type a_j, $j = 1, \ldots, c(r)$, in $T(k)$. Then

$$f^j(k) = \sum_{1 \le i \le r} f^j(k - c(i)), \tag{1}$$

where our initialization is $f^j(k) = 1$ for $k + j = c(r) + 1$, $1 \le k \le c(r)$, and $f^j(k) = 0$ for $k + j \ne c(r) + 1$, $1 \le k \le c(r)$. Clearly, the number of leaves in $T(k)$, $f(k)$, is given by

$$f(k) = \sum_{1 \le j \le c(r)} f^j(k) = \sum_{1 \le i \le r} f(k - c(i)). \tag{2}$$

VARN CODES FOR $N = f(k)$

The reason the recursive tree construction of the previous section is interesting is because the trees constructed are the minimum average codeword cost code trees for equiprobable sources of size $f(k)$, the Varn code trees for these source sizes. This is apparent because the construction rule splits the lowest cost leaf node at each stage, the node of type a_1, and the evolution of the node types from $T(k)$ to $T(k+1)$ keeps track of the relative node costs, that is, how many trees until that node type becomes the least cost node. Thus, the analysis of the average cost of $T(k)$, $C(T(k))$, is of interest.

To find $C(T(k))$, the assumption is that the tree is being used to encode an equiprobable source of $f(k)$ source symbols, and the costs of the codewords are the costs of the leaves of the tree. In $T(k)$, a leaf node of type a_j costs $k-(c(r)+1-j)$ by induction on k. Thus, $C(T(k))$ is given by

$$C(T(k)) = \sum_{1\le j\le c(r)} (k-(c(r)+1-j))f^j(k)/f(k). \tag{3}$$

We now need to analyze these recurrence relations. By the method of generating functions (see, e.g., [2]), we have from (1) and its initialization that $f^j(k)$ satisfies

$$\sum_{1\le k\le\infty} f^j(k)x^k = x^{c(r)+1-j}\left(1-\sum_{1\le i\le r} I(c(i)<j)x^{c(i)}\right)\Big/\left(1-\sum_{1\le i\le r} x^{c(i)}\right),$$

where $I(c(i)<j)=1$ if $c(i)<j$ and 0 otherwise. The coefficients of x^k obtained from the right-hand side of this expression give $f^j(k)$.

For the example of Table 1 with $F^j(x)=\sum_{1\le k\le\infty} f^j(k)x^k$, we have

$$\begin{aligned}
F^1(x) &= x^5/(1-x^2-2x^3-x^5) = x^5+x^7+2x^8+x^9+5x^{10}+5x^{11}+\cdots,\\
F^2(x) &= x^4/(1-x^2-2x^3-x^5) = x^4+x^6+2x^7+x^8+5x^9+5x^{10}+8x^{11}+\cdots,\\
F^3(x) &= x^3(1-x^2)/(1-x^2-2x^3-x^5) = x^3+2x^6+3x^8+4x^9+3x^{10}+12x^{11}+\cdots,\\
F^4(x) &= x^2(1-x^2-2x^3)/(1-x^2-2x^3-x^5) = x^2+x^7+x^9+2x^{10}+x^{11}+\cdots,\\
F^5(x) &= x(1-x^2-2x^3)/(1-x^2-2x^3-x^5) = x^1+x^6+x^8+2x^9+x^{10}+5x^{11}+\cdots,
\end{aligned}$$

from which it can be observed that

$$\begin{aligned}
f^2(k) &= f^1(k+1),\\
f^3(k) &= f^1(k+2)-f^1(k),\\
f^4(k) &= f^1(k+3)-f^1(k+1)-2f^1(k),\\
f^5(k) &= f^1(k+4)-f^1(k+2)-2f^1(k+1).
\end{aligned}$$

Although we do not have a convenient closed form expression for $f^1(k)$ in terms of k, it is interesting to note that

$$f(k) = -2f^1(k)-2f^1(k+1)+f^1(k+3)+f^1(k+4).$$

From (2), we have $F(x)=\sum_{1\le j\le c(r)} F^j(x)$ which, for the example, becomes

$$\begin{aligned}
F(x) &= (x+x^2-2x^4-2x^5)/(1-x^2-2x^3-x^5)\\
&= x+x^2+x^3+x^4+x^5+4x^6+4x^7+7x^8+13x^9+16x^{10}+31x^{11}+\ldots.
\end{aligned}$$

From (3), we have that the generating function for the unnormalized cost,

$$\sum_{1\le k\le\infty} f(k)C(T(k))x^k,$$

becomes

$$xdF(x)/dx - \sum_{1\le j\le c(r)} (c(r)+1-j)F^j(x).$$

For the example, this generating function becomes

$$\begin{aligned}&(13x^6+13x^7+6x^8-2x^9-2x^{10})/(1-x^2-2x^3-x^5)^2\\ &=13x^6+13x^7+32x^8+76x^9+101x^{10}+241x^{11}+\cdots,\end{aligned}$$

so that the normalized costs $C(T(k))$ are as given in Table 2.

TABLE 2. $C(T(k))$ for $c(1)=2, c(2)=3, c(3)=3, c(4)=5$ and Its Entropy Lower Bound

k	$C(T(k))$	$-\log_t f(k)$
6	13/4 = 3.25	3.00
7	13/4 = 3.25	3.00
8	32/7 = 4.57	4.21
9	76/13 = 5.85	5.55
10	101/16 = 6.31	6.00
11	241/31 = 7.77	7.43
...	...	...

Performance bounds on the minimum expected cost of code trees for unequal costs are given in Krause [4] in terms of the source entropy base t, where t is the unique positive root of $1-\sum_{1\le i\le r} x^{c(i)}=0$. For $f(k)$ equiprobable source symbols, this entropy is $-\log_t f(k)$, and the bounds are

$$-\log_t f(k) \le C(T(k)) \le -\log_t f(k)+c(r).$$

However, the code whose cost satisfies the upper bound is not necessarily exhaustive; thus, only the lower bound is relevant here. For the example used here, with $c(1)=2, c(2)=3, c(3)=3, c(4)=5, t\approx 0.63$, and the source entropy base t is also provided in Table 2 for comparison with $C(T(k))$. Also of interest in this connection are the new performance bounds due to Savari [5].

A few comments should be made about this approach to Varn codes. First, the indexing of trees in the order generated by the construction procedure is key; that is, the recurrence relations are elegant stated with this indexing but, possibly, disconcerting aspects of the indexing arise, such as $T(7)$ and $T(6)$ in the example being identical trees with respect to node costs (although different with respect to node types). Also, for some choices of costs, $c(1), c(2), \dots, c(r)$, it is convenient to solve the recurrences explicitly, particularly when the roots of $1-\sum_{1\le i\le r} x^{c(i)}$ are easy to find, as in the case in which $r=2$.

REFERENCES

1. D. K. Chang. "On Fibonacci *k*-ary Trees." *The Fibonacci Quarterly* **24.3** (1986):258-62.
2. R. P. Grimaldi. *Discrete and Combinatorial Mathematics.* 2nd ed. Reading, Mass.: Addison-Wesley, 1989.
3. Y. Horibe. "Notes on Fibonacci Trees and Their Optimality." *The Fibonacci Quarterly* **21.2** (1983):118-28.
4. R. M. Krause. "Channels Which Transmit Letters of Unequal Duration." *Inform. Contr.* **5** (1962):13-24.
5. S. A. Savari. "Some Notes on Varn Coding." *IEEE Trans. Info. Th.* **40** (1994):181-86.
6. B. F. Varn. "Optimal Variable Length Codes (Arbitrary Symbol Cost and Equal Code Word Probabilities)." *Inform. Contr.* **19** (1971):289-301.

AMS Classification Numbers: 11B39, 94A45

❖❖❖