Encoding Spanning Trees of Complete Multipartite Graphs Quickly

Glenn H. Hurlbert*
Department of Mathematics and Statistics
Arizona State University
Tempe, AZ 85287-1804
phone: 480-965-5008
fax: 480-965-0461
email: hurlbert@asu.edu

September 29, 2004

*Partially supported by National Security Agency grant #MDA9040210095.

Abstract

Deo and Micikevicius recently gave a new bijection for spanning trees of complete bipartite graphs. Their method had advantages of linear running time for encoding and decoding, as well as for computing the diameter, center and radius directly from the encoding without having to construct the tree. In this paper we devise a generalization of Deo and Micikevicius's method, which also a modification of Olah's method for encoding the spanning trees of any complete multipartite graph $K(n_1, \ldots, n_r)$. The advantages are the same: we can encode and decode such trees in $O(n)$ time, as well as compute the diameter, center and radius in $O(n)$ time directly from the encoding without having to construct the tree itself.

2000 AMS Subject Classification: 05C05, 05C85, 68R10

Key words: spanning tree, complete multipartite graph, bijection

1 Introduction

Given a graph $G = (V, E)$ with vertices $V = V(G)$ and edges $E = E(G)$, a *spanning tree* $T = (V, E')$ of G is a connected subgraph of G having no cycles. That is, T is a connected graph with $V(T) = V(G)$, $E'(T) \subseteq E(G)$, and $|E'| = |V| - 1$. One very natural and old problem is to determine the number $\tau(G)$ of labelled spanning trees for a fixed graph G, or better yet, a formula for each in a family $\mathcal{G} = \{G_n\}_{n=0}^{\infty}$.

1.1 Applications

There are more reasons for being interested in such formula than can be listed here, although a number of important applications stand out.

The first and most famous result of this kind is due to Cayley [3], who proved that

$$\tau(K_n) \;=\; n^{n-2} \;,$$

where K_n is the complete graph on n vertices. The result was originally stated in terms of counting the number of labelled trees on n vertices, his motivation coming from the enumeration of certain chemical isomers. A wonderful collection of many different proofs of this result is found in [14]. Another notable application of spanning tree enumeration is in the computation of the total resistence along an edge in an electrical network (see [2]).

As is typical in enumeration, one is most pleased with bijective proofs, or encodings, of formulas. In the modern age of computers, moreover, the effi-

cient use of storage space and quick retrieval of data require such encodings. Examples of using encodings of trees in genetic algorithms appear in [7] and [11]. In addition a random spanning tree of a graph can easily be generated if an encoding is known. In particular, in the case of Cayley's formula, one can generate easily a random n-ary string of length $n-2$, and then use one of many known encodings to create a random tree (see [17]).

Lastly, we mention the use of tree enumeration in the enumeration and generation of DeBruijn cycles. A k-ary DeBruijn cycle of order n is an infinite, periodic k-ary string of period k^n having the property that every k-ary n-tuple appears as a contiguous block exactly once in each period. Such cycles were thought to have been discovered independently by DeBruijn [5] and Good [12], but actually were discovered earlier by Flye-Saint Marie [10]. They can be used in cryptographic applications as approximations of random sequences, for example (see [20]). In order to use such an application safely, it is imperative to know how many DeBruijn cycles exist for each choice of k and n. We denote this number $d(n,k)$.

We digress momentarily to present explicitly what is implicitly found in the work of several authors. For a given digraph G we say that a spanning tree is *rooted at vertex* v if, from each vertex $w \neq v$, the unique path from w to v has every edge oriented toward v.

Fact 1 *Let G be an eulerian digraph with outdegree sequence $d_1, \ldots, d_n$. Let $\tau(G)$ be the number of spanning trees of G rooted at a fixed vertex v, and let*

$\rho(G)$ be the number of eulerian circuits in G. Then

$$\rho(G) \;=\; \tau(G)\prod_{i=1}^{n}(d_i-1)!\ .$$

Proof. Fix an eulerian circuit starting with the edge vu; without loss of generality we assume $v = v_1$ and $u = v_2$. After traversing the entire circuit label the edges as follows. At vertex v_i label the d_i outedges in the reverse of the order they were traversed, except that at vertex v_1 do this for all edges other than v_1v_2, choosing labels from $\{2, \ldots, d_i\}$. Notice that (1) the arcs labelled '1' form a spanning tree rooted at v_1 and (2) this labelling induces a permutation of the $d_i - 1$ outedges of v_i not in the tree. The proof is completed by recognizing that the choice of any spanning tree rooted at v, together with a "starter" edge vu (fixed throughout), and a permutation of the non-tree outedges at each vertex, determines an eulerian circuit simply by traversing arcs according to the maximum unused label available at each vertex. □

In the case of DeBruijn cycles we let G be the graph $B(n, k)$ whose vertices are all the k-ary $(n-1)$-tuples and whose oriented edges go from each vertex $(a_1, a_2, \ldots, a_{n-1})$ to each vertex $(a_2, \ldots, a_{n-1}, a_n)$. The standard proof of the existence of DeBruijn cycles comes from their one-to-one correspondence with the set of all eulerian circuits of $B(n, k)$.

1.2 Methods

The focus of this paper is on the methods of calculating τ. We do not attempt to give a history of such results but instead note some of the highlights along

the way (see [15] for a comprehensive treatment).

Kirchoff [13] is responsible for developing the Matrix Tree Theorem, which states that $\tau(G)$ equals the value of any cofactor of the matrix $D - A$, where $D = D(G)$ is the diagonal matrix of degrees of G and $A = A(G)$ is the adjacency matrix of G ($a_{i,j} = 1$ when $v_i v_j$ is an edge and 0 otherwise). One advantage of this method is that one can easily compute τ for reasonably sized graphs.

Similarly, Temperly [21] showed that $\tau(G) = \det(J + D - A)/n^2$, where J is the $n \times n$ matrix of all ones. This formulation has the added advantage of a relation to the eigenvalues of G (see [1]). If G is r-regular and A has eigenvalues $\lambda_1 \leq \cdots \leq \lambda_n$ then

$$\tau(G) \;=\; \frac{1}{n} \prod_{i=1}^{n-1} (r - \lambda_i) \; .$$

(It is known that $\lambda_{n-1} < \lambda_n = r$.)

An analogue of the Matrix Tree Theorem is known as the BEST Theorem [4, 19]. Interestingly, the BEST Theorem yields the enumeration

$$\tau(B(n,k)) \;=\; k^{k^n - 1 - n}$$

which, combined with Fact 1, implies that there are $(k!)^{k^{n-1}}/k^n$ DeBruijn cycles, a fact that is found in [10] without proof.

The first bijective method was given by Prufer [18]. It is possibly the simplest proof of Cayley's Theorem. A generalization of Prufer's encoding was given by Olah [16] for complete multipartite graphs $K(n_1, \ldots, n_r)$. This

graph has $n = \sum_{i=1}^{r} n_i$ vertices, partitioned into sets of sizes $n_1, \ldots, n_r$, and edges between every pair of vertices from diferent parts. The encoding proves that

$$\tau(K(n_1, \ldots, n_r)) \;=\; n^{r-2} \prod_{i=1}^{r} (n - n_i)^{n_i - 1} \;. \tag{1}$$

Another wonderful bijection for these graphs is found in [8, 9], where the authors give a full description of the q-analog properties of their bijections.

We make one last digression regarding spanning trees of planar graphs and those of their planar duals. Let G be a planar graph and G_e one of its embeddings in the plane. Then define H_e to be its planar dual, having a vertex for each face of G and an edge between every pair of vertices coresponding to incident faces of G_e.

Fact 2 *With G_e and H_e defined as above, every graph G satisfies*

$$\tau(G_e) \;=\; \tau(H_e) \;.$$

The proof of this fact given in [1] invokes the Matrix Tree Theorem, but a more appealing bijective method follows.

Proof. Let G_e be drawn on the surface of a sphere instead of the plane, and consider one of its spanning trees T. Imagine using scissors to cut the surface of the sphere along the edges of T, and then lie the cut surface stretched flat on the plane. Define the subgraph T' of H_e, by removing those edges of H_e which cross the edges of T. As is well known, minimal cutsets of G_e correspond to cycles in H_e, and vice-versa. Hence T' is a spanning tree of H_e. The construction is clearly one-to-one, completing the proof. □

Recently, a new method has been found which offers benefits over Prufer's encoding on two counts. The method of Deo and Micikevicius [6] allows one both to encode and decode a spanning tree of K_n in $O(n)$ time. In addition, they present an $O(n)$ algorithm to compute the diameter, center, and radius of the tree without having to construct the tree itself.

2 New Encoding

In this paper we devise a generalization of Deo and Micikevicius's method which also a modification of Olah's method for encoding the spanning trees of any complete multipartite graph $K(n_1, \ldots, n_r)$.[1] The advantages are the same: we can encode and decode such trees in $O(n)$ time, as well as compute the diameter, center and radius in $O(n)$ time without having to construct the tree itself.

2.1 Tree to Code

Let T be a given spanning tree of $K = K(n_1, \ldots, n_r)$, and let $n = \sum_{j=1}^{r} n_j$ be the number of its vertices. Let the partition of the vertices V given by K be $V_1, \ldots V_r$, where each $|V_j| = n_j$. Let $s_j = \sum_{i=1}^{j} n_i$. Then we may assume that $V_j = \{s_{j-1} + 1, \ldots, s_j\}$. Define $V(k) = V_j$, where $k \in V_j$. The code for T will be a set of sequences $C = \{C_0, C_1, \ldots, C_r\}$, where the length of C_0 is $r - 2$ and the length of each C_j $(1 \leq j \leq r)$ is $n_j - 1$. Thus the total length

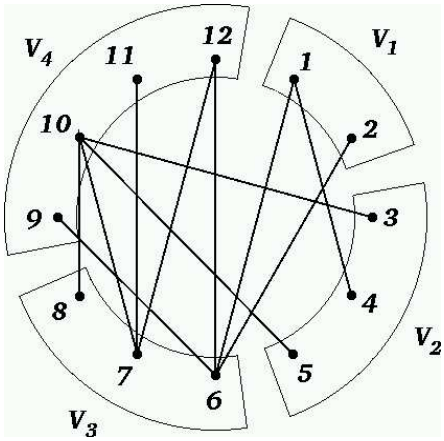
[1] It is worth correcting a small typographical error that appears in Deo and Micikevicius's paper: In each of lines 4 and 5 of Figure 3 in [6] the variable *used*[i] should be replaced by *used*[$C(i)$].

1. **for** i from 1 to n **do**
   (a) **if** $\deg(i) = 1$
      i. **then** add i to the tail of Q_0
2. **set** $i = 1, j = 0$
3. **while** $i \leq n - 2$ **do**
   (a) **while** $Q_j \neq \emptyset$ **do**
      i. remove k from the head of Q_j
      ii. **if** $|C_{V(k)}| < n_i - 1$
         A. **then** add k^+ to the tail of $C_{V(k)}$
         B. **else** add k^+ to the tail of C_0
      iii. remove k from T
      iv. **if** $\deg(k^+) = 1$
         A. **then** add k^+ to the tail of Q_{j+1}
      v. increment i
   (b) increment j

Figure 1: Algorithm **E** for encoding a tree T by a code C

of the code is $(r-2)+\sum_{j=1}^{r}(n_j-1) = n-2$, as expected. Figure 1 shows the algorithm **E** used to encode T by C. The sequence C_0 can contain any of the labels, while every other sequence C_j will contain the labels of the neighbors, at appropriate stages of the algorithm, of the vertices in V_j (in some specified order). The set of all possible such codes has cardinality matching the right hand side of (1). When k is a leaf of T denote its unique neighbor in T by k^+.

It is clear that algorithm **E** creates a code C as described above. Assuming that T is stored in adjacency lists, the degrees of all vertices can be computed in $O(n)$ time, and the neighbor of a leaf can be found in constant time. Thus,



C_0	10	6	
C_1	6		
C_2	10	1	
C_3	10	12	
C_4	6	7	7

Figure 2: The encoding C of a particular tree T

following the same argument found in [6] we find that algorithm **E** runs in $O(n)$ time. Figure 2 shows an example of such an encoding.

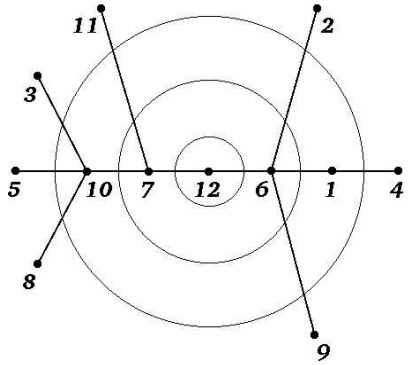
2.2 Code to Tree

Suppose C is a code satisfying the conditions described in Section 2.1. We describe Algorithm **D** (see Figure 3), which constructs from C the tree T that Algorithm **E** encodes as C. label terms of the n-tuple U are initialized in the loop at line 2 to one less than the degrees of the vertices so that the leaves become evident, forming the queue Q_0 (we set $n_0 = r - 1$) in the loop at line 3. The process of constructing T is facilitated by the iterative construction in the loop at line 5 of each queue $Q_1, Q_2, \ldots$, essentially reversing the process in **E** . The variables k and k' are redefined each time they are removed; that is, in line 5(a)i for example, k is defined to be the head of Q_j and then it is removed.

One can check Algorithm **D** against Figure 2. It should be fairly clear that Algorithm **D** is the inverse of Algorithm **E**, and that Algorithm **D** also

1. **for** i from 1 to n **do**
   (a) **set** $U(i) = 0$
2. **for** j from 0 to r **do**
   (a) **for** i from 1 to $n_j - 1$ **do**
      i. increment $U(C_j(i))$
3. **for** i from 1 to n **do**
   (a) **if** $U(i) = 0$
      i. **then** add i to the tail of Q_0
4. **set** $i = 1, j = 0$
5. **while** $i \leq n - 2$ **do**
   (a) **while** $Q_j \neq \emptyset$ **do**
      i. remove k from the head of Q_j
      ii. **if** $C_{V(k)} \neq \emptyset$
         A. **then** remove k' from the head of $C_{V(k)}$
         B. **else** remove k' from the head of C_0
      iii. add $\{k, k'\}$ to $E(T)$
      iv. decrement $U(k')$
      v. **if** $U(k') = 0$
         A. **then** add k^+ to the tail of Q_{j+1}
      vi. increment i
      vii. **if** $i = n - 1$
         A. **then if** $Q_j \neq \emptyset$
            (I) **then** remove k from the head of Q_j
            (II) **else** remove k from the head of Q_{j+1}
   (b) increment j
6. remove k' from the head of Q_j
7. add $\{k, k'\}$ to $E(T)$

Figure 3: Algorithm **D** for decoding a tree T from a code C



Q_0	2	3	4	5	8	9	11
Q_1	1	10					
Q_2	6	7					
Q_3	12						

Figure 4: The layers $Q_1, \ldots, Q_3$ of a particular tree T

runs in $O(n)$ time.

Evidently, when **D** halts, the diameter of T can be computed as $2j + |Q_j| - 1$, the radius of T is $j + |Q_j| - 1$, and the center of T equals Q_j (this takes into account the cases from line 5(a)viiA of Algorithm **D** that distinguish whether the final queue has size 1 or 2, respectively). This can be seen most easily by recognizing that, for every $j > 0$, every label $k \in Q_j$ is adjacent to some label in Q_{j-1} (see Figure 4). Thus in some sense the set of Q_js "peel off" the layers of T from the outside in. Thus, an alorithm that calculates these values directly from C, without constructing T, is given by removing lines 5(a)iii and 7 from Algorithm **D** .

In Deo and Micikevicious's algorithm (Figure 4 of [6]) that computes the diameter of T, the coefficient of n for its running time is an improvement of that for their tree construction algorithm (Figure 3 of [6]) in the case that the diameter of T is not of order n. The improvement comes from being able to isolate the last occurrences of the labels in C. In our case, because of how C is partitioned, it is difficult to determine that information without

constructing the Q_js, which results in the aforementioned modification of Algorithm **D** as a necessary method.

3 Open Problems

We close by offering a few challenges. Notice that for each of the following graph families the familiar pattern of

$$\tau(G) \;=\; \frac{1}{|V|}\prod_{i=1}^{|V|-1} f_i$$

emerges for its number of spanning trees, where $V = V(G)$ and f_i is some set of factors. Such a pattern naturally opens itself to possible encodings.

Let $B(n,k)$ be the DeBruijn digraph defined above.

Problem 3 *Find a bijective proof for the equation*

$$\tau(B(n,k)) \;=\; k^{k^n-1-n} \;=\; \left(\frac{1}{k^n}\right) k^{k^n-1}\ .$$

Let Q^n be the n-dimensional cube, whose vertices are the binary n-tuples and whose edges join n-tuples whose coordinates differ in one position. Let $w(v)$ be the number of 1s in the n-tuple v.

Problem 4 *Find a bijective proof for the equation*

$$\tau(Q^n) \;=\; 2^{-n}\prod_{j=1}^{n}(2j)^{\binom{n}{j}} \;=\; \left(\frac{1}{2^n}\right)\prod_{v\neq 0^n} 2w(v)\ .$$

Let P be the petersen graph, whose vertices are the subsets of $\{1,2,3,4,5\}$ of size two and whose edges join disjoint sets.

Problem 5 *Find a bijective proof for the equation*

$$\tau(P) \;=\; 2000 \;=\; \left(\frac{1}{10}\right) 5^4 2^5 \;.$$

References

[1] N. Biggs, Algebraic Graph Theory, Cambridge, London, 1974.

[2] B. Bollobás, Graph Theory, An Introductory Course, Springer-Verlag, New York, 1979.

[3] A. Cayley, A theorem on trees, Quart. J. Math. 23 (1889) 376–378. Collected papers, Cambridge 13 (1897) 26–28.

[4] T. van Aardenne-Ehrenfest, N.G. de Bruijn, Circuits and trees in ordered linear graphs, Simon Stevin 28 (1951) 203–217.

[5] N.G. de Bruijn, A combinatorial problem, Proc. Nederl. Akad. Wetensch 49 (1946) 758–764.

[6] N. Deo, P. Micikevicius, A new encoding for labeled trees employing a stack and a queue, Bull. Inst. Combin. Appl. 34 (2002), 77–85.

[7] W. Edelson, M.L. Gargano, Constrained minimal spanning trees solved by a GA with feasible encodings, Congr. Numer. 143 (2000), 5–21.

[8] O. Eğecioğlu, J. Remmel, Bijections for Cayley trees, spanning trees, and their q-analogs, J. Combin. Theory Ser. A 42 (1986) 15–30.

[9] O. Eğecioğlu, J. Remmel, A Bijection for Spanning Trees of Complete Multipartite Graphs, Congr. Numer. 100 (1994), 225–243.

[10] C. Flye-Sainte Marie, Solution to problem number 58, l'Inter. Math. 1 (1894) 107–110.

[11] M. Gen, G. Zhou, A note on genetic algorithms for degree-constrained spanning tree problems, Networks 30 (1997), 91–95.

[12] I. J. Good, Normally recurring decimals, J. London Math. Soc. 21 (1946), 167–169.

[13] G. Kirchhoff, Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung galvanischer Ströme gefuhrt wird, Ann. Phys. Chem. 72 (1847) 497–508. Gesammelte Abhandlungen, Leipzig (1882) 22–33.

[14] J.W. Moon, Various proofs of Cayley's formula for counting trees, in: F. Harary, L. Beineke (eds.), A Seminar on Graph Theory, Holt, Rinehart and Winston, New York, 1967, pp. 70–78.

[15] J.W. Moon, Counting labelled trees, Canad. Math. Mono. 1, Canad. Math. Congress, Montreal, 1970.

[16] G. Oláh, A problem on the enumeration of certain trees (Russian), Stud. Sci. Math. Hungar. 3 (1968) 71–80.

[17] E. Palmer, Graphical Evolution, Wiley, New York, 1985.

[18] H. Prüfer, Neuer Beweis eines Satzes über Permutationen, Arch. Math. Phys. 27 (1918) 742–744.

[19] C.A.B. Smith, W.T. Tutte, On universal paths in a network of degree 4, Amer. Math. Monthly 48 (1941) 233–237.

[20] D.R. Stinson, Cryptography. Theory and Practice (2nd ed.), CRC Press Ser. on Disc. Math. and its Appl., Chapman and Hall, Boca Raton, 2002.

[21] H.N.V. Temperly, On the mutual cancellation of cluster integrals in Mayer's fugacity series, Proc. Phys. Soc. 83 (1964) 3–16.