ANALYSIS OF A SPANNING TREE ENUMERATION ALGORITHM

R. JAYAKUMAR
K. THULASIRAMAN
Computer Centre
Indian Institute of Technology, Madras 600 036

1. INTRODUCTION

Enumeration of all the spanning trees of a graph is one of the widely studied graph problems. Several methods have been proposed for listing all the spanning trees of a graph. Minty's method [1] and Gabow and Meyer's method [2] are known to be very efficient. In 1968 Char [3] presented a conceptually simple and elegant algorithm. In this paper we present an analysis of Char's algorithm and report some interesting behaviour of this algorithm. For notations, we follow Harary [4].

2. CHAR'S ALGORITHM

Consider an undirected graph G. Let the vertices of G be denoted as 1, 2, ..., n. Consider any sequence \( (i_1, i_2, ..., i_{n-1}) \) with \( 1 \leq i_j \leq n \) whenever \( 1 \leq j \leq n-1 \). Each such sequence may be considered as representing the following (not necessarily distinct) edges of G.

\[
(1, i_1), (2, i_2), ..., (n-1, i_{n-1}).
\]

Char's algorithm is based on the following result.

Theorem 1 (Tree compatibility test). The sequence \( (i_1, i_2, ..., i_{n-1}) \) represents a spanning tree of G if and only if for each \( j \leq n-1 \) there exists a sequence of edges (chosen from among the set \( (1, i_1), (2, i_2), ..., (n-1, i_{n-1}) \)) with \( (j, i_j) \) as the starting edge, which leads to a vertex \( k > j \).

The sequences passing the tree compatibility test are called tree sequences and those failing the test are called non-tree sequences. We represent the sequence \( (i_1, i_2, ..., i_{n-1}) \) by an array \( \text{DIGIT} \), where \( \text{DIGIT}(j) = i_j, 1 \leq j \leq n-1 \). To start with, Char's algorithm selects a spanning tree \( T \), called the initial spanning tree, of G and numbers the vertices of G so that the tree sequence \( (\text{DIGIT}(1), \text{DIGIT}(2), ..., \text{DIGIT}(n-1)) \)
DIGIT(n-1)) corresponding to 1 satisfies the property

$$\text{DIGIT}(i) > i, \ 1 \leq i \leq n-1$$  \hspace{1cm} \ldots \ (1)$$

For a description of Char's algorithm, see [3]. If \( t_0 \) and \( t \) denote respectively the numbers of non-tree sequences and tree sequences, then it can be shown that Char's algorithm is of complexity \( O(n + n(t + t_0)) \).

3. ANALYSIS OF CHAR'S ALGORITHM

We obtain, in this section, a characterisation of the non-tree subgraphs which correspond to the non-tree sequences generated by Char's algorithm and develop a formula for computing \( t_0 \).

Consider a connected undirected graph \( G = (V, E) \), the spanning trees of which have been enumerated by Char's algorithm. Let the initial tree sequence be \((\text{REF}(1), \text{REF}(2), \ldots, \text{REF}(n-1))\). We know that \( \text{REF}(i) > i, \ 1 \leq i \leq n-1 \). Let \( G_1 = (V_1, E_1) \) be the initial spanning tree of \( G \).

For any \( k, \ 2 \leq k \leq n-1, \) consider a non-tree sequence \((\text{DIGIT}(1), \text{DIGIT}(2), \ldots, \text{DIGIT}(k-1), \text{DIGIT}(k), \text{DIGIT}(k+1), \ldots, \text{DIGIT}(n-1))\) which fails the tree compatibility test at position \( k \). In other words, this non-tree sequence is obtained when \( \text{DIGIT}(k) \) of the previous sequence is changed and it is found that there exists, in \( G \), a sequence of edges, with the edge \((k, \text{DIGIT}(k))\) as the first edge, which leads to vertex \( k \). Note that \( \text{DIGIT}(i) = \text{REF}(i), \ k+1 \leq i \leq n-1 \). Thus the non-tree sequence is \((\text{DIGIT}(1), \text{DIGIT}(2), \ldots, \text{DIGIT}(k-1), \text{DIGIT}(k), \text{REF}(k+1), \ldots, \text{REF}(n-1))\).

Let \( \overline{G} = (\overline{V}, \overline{E}) \) be the subgraph of \( G \) corresponding to this non-tree sequence.

Let \( G_A = (V_A, E_A) \) be the subgraph of \( \overline{G} \) such that

\( V_A = \{k+1, k+2, \ldots, n-1, \text{REF}(k+1), \text{REF}(k+2), \ldots, \text{REF}(n-1)\} \)

and

\( E_A = \{(k+1, \text{REF}(k+1)), (k+2, \text{REF}(k+2)), \ldots, (n-1, \text{REF}(n-1))\} \).

Since \( \text{REF}(i) > i, \ k+1 \leq i \leq n-1 \), we get

\( V_A = \{k+1, k+2, \ldots, n\} \).

Note that \( G_A \) is a subgraph of the initial spanning tree \( G_1 \) of \( G \) and there is a path in \( G_A \) from each one of the vertices \( k+1, k+2, \ldots, n-1 \) to the vertex \( n \).
So \( G_A \) is a connected subgraph of \( G \).

Let \( G_B = (V_B, E_B) \) be the subgraph of \( G \) such that

\[ V_B = \{1, 2, \ldots, k, \text{DIGIT}(1), \text{DIGIT}(2), \ldots, \text{DIGIT}(k)\} \]

and

\[ E_B = \{(1, \text{DIGIT}(1)), (2, \text{DIGIT}(2)), \ldots, (k, \text{DIGIT}(k))\}. \]

If, in \( G \), there is a path from vertex \( k \) to a vertex in \( V_A \), then \( \text{DIGIT}(k) \) would have passed the tree compatibility test. Thus, in \( G \), there is no path from vertex \( k \) to any vertex in \( V_A \). Let \( V_C \) be the subset of the vertex set \( V_B - \{k\} \) such that in \( G \) there is a path from each vertex in \( V_C \) to a vertex in \( V_A \), and let \( E_C \subset E \) be the set of all the edges in such paths. Let \( G_C = (V_C, E_C) \). Now define the graphs \( G_1 \) and \( G_2 \) as follows:

\[ G_1 = (V_1, E_1) = G_A \cup G_C, \]

\[ G_2 = (V_2, E_2) = G - G_1. \]

It can now be seen that a non-tree sequence which fails the tree compatibility test at position \( k \) corresponds to a subgraph of \( G \) which is of one of the following two types.

**Type 1.** Spanning subgraph \( G' \) of \( G \) such that

1. \( G' \) has exactly two components \( G_1 \) and \( G_2 \).
2. The edges \((k+1, \text{REF}(k+1)), (k+2, \text{REF}(k+2)), \ldots, (n-1, \text{REF}(n-1))\) are in the component \( G_1 \) of \( G' \) and the vertex \( k \) is in the other component \( G_2 \).
3. There is exactly one circuit in \( G' \) and it passes through the vertex \( k \).

**Type 2.** Spanning 2-tree \( T' \) of \( G \) such that

1. The edges \((k+1, \text{REF}(k+1)), (k+2, \text{REF}(k+2)), \ldots, (n-1, \text{REF}(n-1))\) are in the component \( G_1 \) of \( T' \) and the vertex \( k \) is in the other component \( G_2 \).
2. The component of \( T' \) containing vertex \( k \) has at least two vertices.

Note that for the subgraphs of Type 1, the circuit passing through vertex \( k \) can be traversed in one of two directions and hence each subgraph of Type 1 corresponds to two different non-tree sequences. Similarly each spanning 2-tree \( T' \) of Type 2 corres-
pends to \( d^{(k)}(T') \) distinct non-tree sequences where \( d^{(k)}(T') \) is the degree of vertex \( k \) in \( T' \).

From the above observations we get the following

**Theorem 2.** Let \( G = (V, E) \) be a connected undirected graph. The number of non-tree sequences generated by Char's algorithm which fail the tree compatibility test at position \( k \) is equal to \( 2|S_1| + \sum_{T' \in S_2} d^{(k)}(T') \), where \( S_1 \) denotes the set of all spanning \( G' \subseteq G \) of Type 1, \( S_2 \) denotes the set of all spanning 2-trees \( T' \) of Type 2, and \( d^{(k)}(T') \) is the degree of vertex \( k \) in \( T' \).

For any \( k, 2 \leq k \leq n-1 \), let \( \pi_k = (V_1, V_2) \) be a partition of the vertex set \( V \) of \( G \) where \( V_1 \) and \( V_2 \) satisfy the following:

1. \( \{k+1, k+2, \ldots, n\} \subseteq V_1 \) ... (2)
2. \( k \in V_2 \) ... (3)
3. \( |V_2| \geq 2 \) ... (4)

Note that the vertex sets \( V_1 \) and \( V_2 \) of the subgraphs \( G_1 \) and \( G_2 \) defined in the previous section form a partition of \( V \) which satisfy the above properties.

Let

1. \( G_a(\pi_k) \) be the subgraph of \( G \) induced by the vertex set \( V_1 \),
2. \( G_a^{(s)}(\pi_k) \) be the graph obtained from \( G_a(\pi_k) \) by coalescing the vertices \( k+1, k+2, \ldots, n \) and
3. \( G_b(\pi_k) \) be the subgraph of \( G \) induced by the vertex set \( V_2 \).

Further let

1. \( t_a^{(s)}(\pi_k) \) and \( t_b(\pi_k) \) be the number of spanning trees of \( G_a^{(s)}(\pi_k) \) and \( G_b(\pi_k) \) respectively, and
2. \( d_b^{(k)}(\pi_k) \) be the degree of vertex \( k \) in \( G_b(\pi_k) \).

Using Theorem 2, the number \( t_o \) of non-tree sequences generated by Char's algorithm can be obtained as

\[
t_o = \sum_{k=2}^{n-1} \sum_{\pi_k} d_b^{(k)}(\pi_k) t_a^{(s)}(\pi_k) t_b(\pi_k) \quad \ldots \quad (5)
\]
Note that the vertices of \( G \) are numbered with respect to the initial spanning tree. Hence the subgraphs \( G_{a}(n_{k}) \), \( G_{a}(n_{k}) \), and \( G_{b}(n_{k}) \) depend on the initial spanning tree. Hence the value of \( t_{o} \) depends on the choice of the initial spanning tree.

Using (5) we can see that \( t_{o} \leq n^{2}t \), so that the complexity of Char's algorithm is \( O(m+n+n^{3}t) \). However the inequality \( t_{o} \leq n^{2}t \) is not tight as will be seen from the results of the next section.

4. ANALYSIS OF CHAR'S ALGORITHM FOR SPECIAL CLASSES OF GRAPHS

In this section, we report the behaviour of Char's algorithm in the case of certain special classes of graphs.

Let \( G(n-1) \) denote the class of all \( n \)-vertex connected graphs in which there exists a vertex with degree \( n-1 \). Let \( P_{n-1} + K_{1} \) be called an \( n \)-vertex ladder \( L_{n} \), \( C_{n-1} + K_{1} \) be called an \( n \)-vertex wheel \( W_{n} \). Let \( t_{0}(G) \) and \( t(G) \) be the number of non-tree sequences and tree sequences generated by Char's algorithm when applied on a graph. In the following we assume that \( K_{1,n-1} \) has been chosen as the initial spanning tree. The following results are based on [5] and the fact that the spanning trees of ladders are alternate numbers in the Fibonacci sequence.

**Theorem 3.**

(i) For \( G \in G(n-1) \), \( t_{0}(G) \leq t(G) \).

(ii) Complexity of Char's algorithm for all graphs in \( G(n-1) \) is \( O(m+n+n^{3}t) \).

**Theorem 4.**

(i) \( t_{0}(L_{n}) = t(L_{n-1}) \).

(ii) \[ \lim_{n \to \infty} \frac{t_{0}(L_{n})}{t(L_{n})} = 0.382. \]

**Theorem 5.**

(i) \( t_{0}(W_{n}) = t(L_{n}) + 1 \).

(ii) \[ \lim_{n \to \infty} \frac{t_{0}(W_{n})}{t(W_{n})} = 0.4472. \]

**Theorem 6.**

\[ t_{0}(K_{n}) = \sum_{k=2}^{n} \left( \frac{k-1}{\rho} \right) (n-k)(n-k+1)(k-1) t(K_{n-k}). \]

We conjecture that

\[ \lim_{n \to \infty} \frac{t_{0}(K_{n})}{t(K_{n})} = 1. \]
Yet another interesting result with respect to $K_n$ is that $t_o(K_n)$ is independent of the choice of the spanning tree. This can be proved using equation (5).

Proofs of all the above results are given in [5].

5. COMPUTATIONAL EXPERIENCE

All the three algorithms due to Char, Minty, and Gabow and Myer have been programmed in PL/I and tested on a number of randomly generated graphs. In all the cases tested it has been found that Char's algorithm is superior to the other two algorithms, Char's algorithm becomes more and more efficient as the number of trees generated increases.

We have proved in Theorem 3 that $t_o(G) \leq t(G)$, if $G \in O^{(n-1)}$. Computational results suggest that $t_o \leq t$ even in the case of graphs which do not belong to $O^{(n-1)}$.

There is strong computational evidence to believe that Char's algorithm is a very efficient one. This would be so if we could prove that $t_o \leq t$ in the case of all graphs. This is an interesting open problem.

REFERENCES