Improved algorithm for the symmetry number problem on trees [1]

Yijie Han
School of Computing and Engineering
University of Missouri at Kansas City
5100 Rockhill Road
Kansas City, MO 64110, USA.
hanyij@umkc.edu

Abstract

The symmetry number of a tree is defined as the number of nodes of the maximum subtree of the tree that exhibits axial symmetry. The best previous algorithm for computing the symmetry number for an unrooted unordered tree is due to Mitra et al. [8] and runs in $O(n^3)$ time. In this paper we show an improvement on this time complexity by encoding small trees. Our algorithm runs in time $O(n^3(\log\log n/\log n)^2)$.

Keywords: Algorithms, graph drawing, symmetry number, weighted matching.

1 Introduction

Graph drawing deals with generating geometric representations of graphs. Aesthetic drawings of graphs is a main goal of graph drawing techniques. Aesthetic guidelines given in the literature [1, 10] for drawing pretty graphs include minimizing the number of edge crossings, minimizing the variance of edge lengths, minimizing the number of bends, angular resolution, drawing edges orthogonally, etc. Symmetry as another aesthetic criterion has received increasing attention recently in the graph drawing community[2, 3, 4, 6, 7, 8]. However, deciding whether a graph has an axial or rotational symmetry is computationally intractable[6, 7]. The symmetry num-

[1]This research is supported in part by National Science Foundation Grant 0310245.

ber of an unrooted tree T is the number of nodes in the subtree of T that exhibits axial symmetry and has maximum cardinality[2]. The symmetry number problem is to find the symmetry number of a given tree T. Chin and Yen have given an $O(n^4)$ time algorithm[2] for computing the symmetry number of an unordered unrooted tree. Later Mitra et al. improved the time complexity to $O(n^3)$. In this paper we encode the small trees and by using this encoding we show time complexity $O(n^3(\log\log n/\log n)^2)$ for computing the symmetry number of an unordered unrooted tree.

2 Previous algorithms

In this section we outline previous algorithms due to Chin and Yen [2] and the improvements made by Mitra et al. [8]. The presentation of Chin and Yen's algorithm [2] here basically follows that presented in [8].

Let $T = (V, E)$ be a rooted tree. Given a node v, T_v is the rooted subtree of T that has v as root and contains all the vertices that are descendants of v. An r-subtree of T is a rooted tree generated from T by cutting off some of T's subtrees. Tree T and any of its non-empty r-subtrees share the same root.

In [2], Chin and Yen solve the symmetry number problem by solving two subproblems. Two functions $A(v_i)$ and $B(v_i, v_j)$ are used to solve these two subproblems. Function $A(v_i)$ computes and returns the number of nodes in the maximum symmetric r-subtree of T_{v_i} under the constraint that v_i must be on the symmetry axis. Function $B(v_i, v_j)$ returns the number of nodes in the maximum r-subtree of T_{v_i} that is isomorphic to an r-subtree of T_{v_j}.

The procedures for computing $A(v_i)$ and $B(v_i, v_j)$ are given below.

Procedure $A(v_i)$

1. **if** v_i is a leaf **then return** 1;
2. **else**
3. construct a weighted complete graph $G_{v_i} = (V', E')$ with weight function w such that $V' = \{v_i\} \cup C_{v_i}$ $\{C_{v_i}$ is the set of v_i's children$\}$, $w_{il} = A(v_l)$, $\forall v_l \in C_{v_i}$, and $w_{pq} = 2B(v_p, v_q)$, $\forall v_p, v_q \in C_{v_i}$;
4. **return** 1+(weight of maximum matching of G_{v_i});
5. **end if**

Procedure $B(v_i, v_j)$

1. **if** v_i or v_j is a leaf **then return** 1;
2. **else**
3. construct a weighted complete bipartite graph $G_{v_i,v_j} = (V_i \cup V_j, E')$ with weight function w such that $V_i = C_{v_i}$, $V_j = C_{v_j}$, and $w_{pq} = B(v_p, v_q)$, $\forall v_p \in V_i$ and $\forall v_q \in V_j$;
4. **return** 1+(weight of maximum weight matching of G_{v_i,v_j});
5. **end if**

The proof of correctness for these two procedures can be found in [2]. Computing $A(v_i)$ for all nodes v_i in $V(T)$ can be done in $O(n^3)$ time which is dominated by the time for computing maximum matching on weighted graphs[5, 9]. The complexity of a single call of $B(v_i, v_j)$ is $O(n_i n_j \sqrt{n} \log n)$, where $n_i = |C_{v_i}|$ (the number of children of v_i) and $n_j = |C_{v_j}|$ (the number of children of v_j). The complexity of computing $B(v_i, v_j)$ for all possible pairs for a single root is $O(n^{2.5} \log n)$ as shown in [2]. Therefore, for a rooted tree, computing the symmetry number needs $O(n^3 + n^{2.5} \log n) = O(n^3)$ time. Since each node $v \in V(T)$ can be the root, computing

the symmetry number of an unrooted tree T has time $O(n^4)$ as derived by Chin and Yen[2].

What Mitra et al. [8] observed is that no matter which node is considered to be the root, $B(v_i, v_j)$ needs to be computed just once. Therefore computing $B(v_i, v_j)$ for all possible roots takes $O(n^{2.5} \log n)$ time. They also observed that if v_i has n_i neighbors, then $A(v_i)$ needs to be computed only $n_i + 1$ times, once for each neighbor to be the root and once for v_i to be the root. Mitra et al. [8] also found out that only the first time for computing $A(v_i)$ the data structure needs to be set up and maximum weight matching algorithm needs to be called which involves $O(n_i^3)$ time. Each subsequent calls to $A(v_i)$ can be done by updating the maximum weight matching and therefore needs only $O(n_i^2)$ time. Thus each $A(v_i)$ can be computed in $O(n_i^3)$ time. Summing for all $v_i \in V(T)$ they got $O(n^3)$ for computing the symmetry number for an unrooted trees.

3 Improving the Time Complexity by Encoding Small Trees

Each small tree having no more than $\log n/(2 \log \log n)$ nodes can be encoded into an integer with $\log n/2$ bits. We do this by using adjacency list to represent the tree. Each edge is represented by $\log \log n$ bits. Since we have $\log n/(2 \log \log n)$ nodes in the tree, the whole tree can be encoded with $\log n/2$ bits. As our computation has n input nodes we can assume that each word used in our computation has $\log n$ bits and is sufficient to encode such a small tree.

Since we are only to improve the time complexity to $O(n^3(\log \log n/ \log n)^2)$ we do not need to improve the complexity for computing $B(v_i, v_j)$'s which requires only $O(n^{2.5} \log n)$ time.

Now consider computing $A(v_i)$. Let n_i be the number of neighbors of v_i. If $n_i < 2n \log\log n/\log n$ then the computation of $A(v_i)$ needs only $O(n_i^3) = O((n \log\log n/\log n)^3)$ time. If $n_i \geq 2n \log\log n/\log n$ then there are only $2n \log\log n/\log n$ neighbors has subtree rooted at the neighbor with size larger than $\log n/(2 \log\log n)$. For those subtrees with size no more than $\log n/(2 \log\log n)$ only $\sqrt{n}$ of them can be distinct because only $\log n/2$ bits are used to represent them. The rest of the small trees are isomorphic to one of these $\sqrt{n}$ trees. We can pair-off these small trees until there no more than $\sqrt{n}$ of them left. The small trees being paired-off need not participate in the computation of maximum weight matching. Therefore we have no more than $2n \log\log n/\log n + \sqrt{n} = O(n \log\log n/\log n)$ neighbors need participate in the maximum weight computation. Thus the complexity for computing $A(v_i)$'s for all $v_i \in V(T)$ is $\sum_{v_i \in V(T)} O(n_i^3)$ subject to $n_i \leq O(n \log\log n/\log n)$. This complexity is maximized when all $n_i = n \log\log n/\log n$ and therefore the time complexity is $O(n^3(\log\log n/\log n)^2)$.

Therefore we obtain the following theorem.

Theorem: The symmetry number of a unordered unrooted tree can be computed in $O(n^3(\log\log n/\log n)^2)$ time.

References

[1] G. Battista, P. Eades, R. Tamassia. Graph drawing: Algorithms for visualization of graphs. Prentice Hall, New York, 1999.

[2] K. Chin, H. Yen. The symmetry number problem for trees. Inform. Process. Lett. 79(2001) 73-79.

[3] P. Eades, X. Lin. Spring algorithms and symmetry. Computing and Combinatorics (COCOON'97), Lecture Notes in Comput. Sci., Vol 1276, Springer, Berlin, 1997, pp. 202-211.

[4] S. Hong, O. Eades, A. Quigley, S. Lee. Drawing algorithms for serial-parallel digraphs in two and three dimensions. Graph Drawing'98,, Lecture Notes in Comput. Sci., Vol. 1547, Springer, Berlin, 1998, pp. 198-209.

[5] E. Lawler. Combinatorial Optimization: Networks and Matroids. Holt, Rinehart and Winston, New York, 1976.

[6] J. Manning. Geometric symmetry in graphs. Ph.D. Dissertation. Dept. of Computer Science, Purdue University, West Lafayette, IN. 1990.

[7] J. Manning, M. Attallah. Fast detection and display of symmetry in trees. Congr. Numer. 64 (1988) 159-169.

[8] P. P. Mitra, M. A. U. Abedin, M. A. Kashem. Algorithms for solving the symmetry number problem on trees. Inform. Process. Lett. 91 (2004) 163-169.

[9] C.H. Papadimitrou, K. Steiglitz. Combinatorial Optimization: Algorithms and Complexity. Prentice Hall, New York, 1982.

[10] R. Tamassia. Graph drawing. In J.E. Goodman, J. O'Rourke (Eds.) CRC Handbook of Discrete and Computational Geometry. CRC Press, Rockville, MD, 1997.