JOURNAL OF ALGORITHMS 8, 106–112 (1987)

$O(n^{2.5})$ Time Algorithms for the Subgraph Homeomorphism Problem on Trees

MOON JUNG CHUNG

Department of Computer Science, Rensselaer Polytechnic Institute
Troy, New York 12181

Received August 15, 1985

The complexity of the subgraph homeomorphism problems have been open. We show $O(n^{2.5})$ time algorithms when the problems are restricted to trees, directed or undirected. The algorithm can be applied to the subtree isomorphism problem for unrooted trees with the same complexity, and improves over Reyner's $O(n^{3.5})$ algorithm for the subtree isomorphism problem.

1. INTRODUCTION

Let G and H be graphs. G is homeomorphic to H if H can be obtained from G by repeatedly removing any node of degree 2 and adding the edge joining its two neighbors. The subgraph homeomorphism problem (SHP) is defined as follows.

Subgraph Homeomorphism Problem (SHP)

Instance: Undirected graphs G and P.
Question: Does G have a subgraph G' which is homeomorphic to P?

In this paper G is called the "test graph" and P is called the "pattern graph." The problem has wide applications. For example, a graph G is planar if it does not have a subgraph which is homeomorphic to $K_{3,3}$ or K_5. The SHP is NP-complete by reduction from the Hamiltonian path problem [2]. If the pattern graph P is fixed, several complicated polynomial algorithms have been found for a particular graph P, such as a triangle [4] and two disjoint edges [6]. But the problem is open whether it is NP-complete or not for all fixed P. For the case when graphs are directed, the problems have been considered by Fortune, Hopcraft and Willie [1].

In this paper we will restrict graphs to be trees. An undirected tree (or free tree) is called just a tree, and a directed tree is called a rooted tree. Recently Vlades [7] showed that the SHP on trees can be solved in time $O(n^{4.5})$. Reyner [5] considered subgraph isomorphism problem for trees. His algorithm has complexity $O(n^{2.5})$ for rooted trees, and $O(n^{3.5})$ for

0196-6774/87 $3.00

undirected trees, where n is the number of nodes in the trees. Our algorithm presented here can also be used for subgraph isomorphism problem for trees. The complexity $O(n^{2.5})$ given here is an improvement for undirected trees. In Section 2, we first describe $O(n^{2.5})$ algorithm for the subgraph homeomorphism problems for rooted trees. In Section 3, it is shown that SHP for trees can be solved in $O(n^{2.5})$ time. If the pattern graph is a fixed tree, it is shown that we can solve the problem in linear time.

2. Polynomial Time Algorithms on Rooted Trees

We first describe an $O(n^{2.5})$ algorithm for the subgraph homeomorphism problem for rooted trees, where n is the number of nodes in T. A tree T with a set of nodes V_T and a set of edges E_T is represented by $T = (V_T, E_T)$. A tree T with a root node r is called a rooted tree and is represented by $T_r = (V_T, E_T, r)$. The root node of the tree implies the direction for each edge which points away from the root. For a rooted tree $T_r = (V_T, E_T, r)$, $T_r(v)$ denotes the subtree of T_r generated by node v (see Fig. 1). A node v is a descendant of u if there is a path (of length ≥ 0) from u to v. Let $T_r = (V_T, E_T, r)$, $P_{r'} = (V_P, E_P, r')$ be the test and the pattern tree, respectively. For each node of v of T, define $S^r(v)$ as follows:

$S^r(v) = \{x \in V_P |$ there is a subgraph of $T_r(v)$ which is homeomorphic to $P_{r'}(x)\}$. Thus, if $r' \in S^r(v)$, then the rooted tree T_r has a subgraph homeomorphic to P. The computation of $S^r(v)$, can be done as follows. First, compute $S^r(v)$, for all leaf nodes of T_r. Next, compute $S^r(v)$ for node v of T_r whenever we have computed the set $S^r(w)$ for all the children w of v.

Let v be a node of T_r with children $x_1, x_2, \ldots, x_s$, and u be a node of P with children $y_1, \ldots, y_t$. The following lemma is trivial.

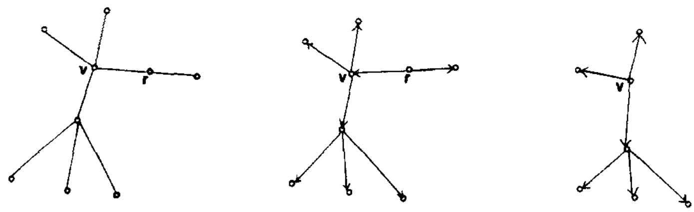
Lemma 1. *$S^r(v)$ contains u iff either*

(i) *there is a child x_i of v such that $u \in S^r(x_i)$, or*

(ii) *there are $i_1, \ldots, i_t$ such that $i_j \neq i_k$, for all $j \neq k$, and $y_k \in S^r(x_{i_k})$, for all k $(1 \leq k \leq t)$.*

Thus, after computing $S^r(x_i)$, for $1 \leq i \leq s$, we can decide if $u \in S^r(v)$ by solving a bipartite matching problem as follows. Construct a bipartite graph G with bipartition X and Y, where X is the set of children of v, Y is the set of children of u, and (x_i, y_j) is an edge of G iff $y_j \in S^r(x_i)$. If G has a matching of size $t = |Y|$, u is in $S^r(v)$.

Theorem 1. *The* SHP *for rooted trees can be solved in time $O(n^{2.5})$, where n is the number of nodes in T_r.*



a b c

FIG. 1. (a) Tree T, (b) tree T with root r, and (c) $T_r(v)$.

Proof. The following algorithm decides whether T_r has a subtree homeomorphic to $P_{r'}$ or not. Suppose $V_T = \{v_1, \ldots, v_n\}$ and $V_P = \{u_1, \ldots, u_m\}$. Now consider the time complexity of computing $S^r(v_i)$, for each v_i of T_r. Let t_i and s_i be the number of children of node u_i of $P_{r'}$ and node v_i of T_r, respectively. Then we have,

$$\sum_{i=1}^{i=n} s_i = n - 1 \qquad \text{and} \qquad \sum_{i=1}^{i=m} t_i = m - 1.$$

Since the matching problem on a bipartite graph with node bipartition of size t and size s nodes can be solved in time $ct^{3/2}s$[3], for some constant c, the time complexity of computing $S^r(v_i)$ is bounded by

$$\sum_{j=1}^{j=m} cs_i t_j^{1.5} = cs_i \sum_{j=1}^{j=m} t_j^{1.5} \le cs_i m^{1.5}.$$

Since we must compute $S^r(v_i)$, for $1 \le i \le n$, the time complexity of the Algorithm A is

$$\sum_{i=1}^{n} cs_i m^{1.5} \le cm^{1.5} \sum_{i=1}^{n} s_i \le cm^{1.5} n.$$

Algorithm A

Input: Rooted trees $T = (V_T, E_T, r)$ and $P = (V_P, E_P, r')$.
Output: Yes, if T_r has a subtree which is homeomorphic to P.

0. (*Initially all nodes are not marked*)
1. for each leaf v of T_r do $S^r(v) = \{x \mid x \text{ is a leaf node of } P\}$
2. mark all leaf nodes of T
3. for each node v of T such that all of the children of v are marked do begin
4. compute $S^r(v)$

5. mark v
   end
6. if $r' \in S^r(r)$ then return(true) else return(false)

The step 4, $S^r(v)$ can be computed as follows.

Algorithm Compute $S^r(v)$

1. Set $S^r(v) = \cup_{x \in X} S^r(x)$, where X is the set of children of v.
2. For each node u of P do step 3.
3. 3.1 Construct a bipartite graph G with node bipartition X and Y such that
   X is the set of children of v, Y is the set of children of u, and connect between $x \in X$ and $y \in Y$ iff $y \in S^r(x)$.
   3.2 If G has a matching of size $|Y|$, insert u into $S^r(v)$.
4. return

3. $O(n^{2.5})$ Algorithm on Trees

Let T and P be unrooted trees. First, we can pick an arbitrary node r' of P to get the rooted tree $P_{r'}$. Next, check whether there is a node r such that the tree T_r has a subtree which is homeomorphic to $P_{r'}$. This method will take time $O(n^{3.5})$. But we can improve the time complexity to $O(n^{2.5})$. The basic idea is that after choosing r as the root of T and computing $S^r(v)$ for all v in T, we can compute $S^r(v)$ much faster for different choice of r'' as the root of T. The following lemma is used in the analysis of our algorithm.

LEMMA 2. *Let $G = (X \cup Y, E)$ be a bipartite graph with node bipartition $X \cup Y$, and $X_i = X - \{x_i\}$, where $X = \{x_1, x_2, \ldots, x_s\}$, and $t = |Y| \le s = |X|$. Let M_i be a maximal cardinality matching between X_i and Y, for $i = 1, \ldots, s$. Computing all $|M_i|$, $(i = 1, \ldots, s)$ can be solved in time $O(t^{1.5}s)$, where $t = |Y|$.*

Proof. Find a maximum cardinality matching M between X and Y. Let X' be the set of unmatched nodes of X. Let u be a node not in $(X \cup Y)$. Construct a directed graph

$$G' = (V, A), \text{ where}$$
$$V = \{u\} \cup X \cup Y$$

and

$$A = \{(x \to y) | (x, y) \in (E - M), x \in X, y \in Y)\}$$
$$\cup \{(x \leftarrow y) | (x, y) \in M, x \in X, y \in Y)\} \cup \{(u \to x) | x \in X'\}.$$

$(x \to y)$ denotes a directed edge from x to y.

Do depth first search of G' starting from u. Let X'' be the set of nodes of X which can be reached from u. Compute $|M_i|$ as follows.

Case 1. $x_i \in X''$.

$$\text{Set } |M_i| = |M|.$$

Case 2. $x_i \notin X''$.

$$\text{Set } |M_i| = |M| - 1.$$

Finding a maximum cardinality matching takes $O(t^{1.5}s)$, and depth first search takes $O(s \cdot t)$ time. Thus the entire computation can be done in $O(t^{1.5}s)$.

THEOREM 2. *The* SHP *for trees can be solved in time* $O(nm^{1.5})$, *where* n *is the number of nodes of a test tree and* m *is the number of nodes of a pattern tree. If the pattern tree is fixed, we can solve the problem in linear time.*

Proof. Let $T = (V_T, E_T)$ be a test tree and $P = (V_P, E_P)$ a pattern tree. Choose an arbitrary node r' of P as a root node of P. Suppose that r is chosen as the root of T. Recall that $T_r(v)$ is a subtree of T_r generated by v, and $S^r(v)$ is the subset of V_P such that w is in $S^r(v)$ iff $T_r(v)$ has a subtree which is homeomorphic to $P_{r'}(w)$. The algorithm has two stages, the forward stage and the backward stage. During the forward stage, we compute $S^r(v)$ for all v of T and check if $r' \in S^r(r)$ using Algorithm A. During the backward stage, we change the root of T, and for all r'' of T, we check if $T_{r''}$ has a subtree homeomorphic to $P_{r'}$. Choosing r'' as the new root of T means changing of arc direction from r to r''.

Forward Stage. Compute $S^r(v)$ for all v of T_r. For a node v ($v \neq r$) in T, let $X = \{x_1, x_2, \dots, x_s\}$ be the set of nodes adjacent to v. Suppose that x_1 is the father of v in T_r. Note that $S^r(v)$ is equal to $S^{x_1(v)}$. Thus, during the forward stage, we have computed $S^{x_1(v)}$ and $S^v(x_j)$, for $j = 2, \dots, s$.

Backward Stage. First, compute $S^w(r)$, for all the children w of root r (i.e., the root of T is changed from r to w). Then, compute $S^w(w)$ and apply the backward stage recursively. Let us explain this step in detail. Let v be a node in T, $X = \{x_1, \dots, x_s\}$ be the set of nodes adjacent to v, and x_1 be the father of v by choosing r as the root of T. During the backward stage, at node x_1, we compute $S^v(x_1)$. Since $S^v(x_j)$, $(2 \leq j \leq s)$, has been computed during the forward stage, we can now compute $S^v(v)$. Next, compute $S^{x_j}(v)$, for $j = 2, \dots, s$, by deciding, for each $u_k \in V_P$, if $u_k \in S^{x_j}(v)$ as follows.

(i) Construct a bipartite graph $G = (X \cup Y, E)$, where Y is the set of the children of u_k, and $x_j \in X$ is adjacent to $y \in Y$ iff $y \in S^v(x_j)$.

(ii) Compute the size $|M_i|$ of maximal cardinality matching between X_i and Y, for $i = 1, \ldots, s$, where $X_i = X - \{x_i\}$.

(iii) If $|M_i| = |Y|$, then $u_k \in S^{x_i}(v)$, otherwise $u_k \notin S^{x_i}(v)$.

If u_k has degree t_k, it takes $O(t_k^{3/2}s)$ time to do step (i)–(iii) by lemma 2. After computing $S^{x_j}(v)$, we do the same step recursively on node x_j, for all children x_j of v.

Now let us consider the time complexity. Let $V_T = \{v_1, \ldots, v_n\}$, $V_P = \{u_1, \ldots, u_m\}$, u_k has degree t_k, and v_i has degree s_i. For each v_i, computing $S^{x_j}(v_i)$, for all x_j adjacent to v_i, has time complexity

$$\sum_{u_k \in E_p} c \cdot s_i t_k^{1.5} \leq c \cdot s_i m^{1.5}$$

Thus, total time complexity is $\sum_{i=1}^{n} c s_i m^{1.5} \leq cnm^{1.5}$. In the following algorithm, steps 1–3 of Algorithm B correspond to the forward stage, and steps 4–6 correspond to the backward stage.

Algorithm B

Input: Trees $T = (V_T, E_T)$, $P = (V_P, E_P)$.
Output: Yes, if T has a subtree which is a homeomorphic to P.

1. Pick an arbitrary node r' of P as the root of P.
2. Pick an arbitrary node r of T as the root of T.
3. Apply algorithm A with graphs $T = (V_T, E_T, r)$, $P = (V_P, E_P, r')$. At this moment, for every $v \neq r$, we have computed $S^{v'}(v)$, where v' is the father of v.
4. Call SCOMP(r) (*SCOMP(v) compute $S^x(v)$, for every node x adjacent to v*).
5. For every child x of r do
   TRAVERSE(x, $S^x(r)$).
6. if $r' \in S^x(v)$, for some v and x then return(yes) else return(no).

In step 4, TRAVERSE treverses the tree in preorder and computes $S^x(v)$ for all v and children of v in T_r.

Procedure TRAVERSE(v, S)

```
0.  begin
1.      Compute S^v(v).
2.      Call SCOMP(v).
3.      For every x which is a child of v do
4.              TRAVERSE(x, S^x(v)).
    end
```

The SCOMP(v) compute $S^x(v)$, for every node x adjacent to v.
Procedure SCOMP(v)

0. (*Let $X = \{x_1, \ldots, x_s\}$ be the set of nodes adjacent to v in T^*)
   begin
1. Set $S^{x_i}(v) = \phi$, for all x_i, $1 \leq i \leq s$.
2. For every $u_k \in V_P$ do
3. begin
3.1 Construct a bipartite graph $G = (X \cup Y, E)$, Y is the set of children of u_k, and $(x, y) \in E$ iff $y \in S^v(x)$
3.2 Compute $|M_i|$, for all i, $1 \leq i \leq s$.
3.3 If $|M_i| = |Y|$ then insert $u_k \in S^{x_i}(v)$
   end
   end

The following corollary can be proved by modifying the computation step of $S^r(v)$. It is an improvement over $O(n^{3.5})$ time complexity in [5] for unrooted trees.

COROLLARY. *The subtree isomorphism problem, for unrooted or rooted trees, can be solved in $O(n^{2.5})$ time.*

REFERENCES

1. S. Fortune, J. Hopcroft, and J. Willie, The directed subgraph homeomorphism problem, *Theoret. Comput. Sci.* **9** (1980), 111–121.
2. M. R. Garey and D. S. Johnson, "Computers and Intractability," Freeman, San Francisco, 1979.
3. J. E. Hopcroft and R. M. Karp, A $n^{5/2}$ algorithm for maximum matching in bipartite graphs, *J. SIAM Comput.* **2** (1973), 225–231.
4. A. S. LaPaugh and R. L. Rivest, The subgraph homeomorphism problems, *in* Proc. 10th STOC, pp. 40–50, 1978.
5. S. W. Reyner, An analysis of a good algorithm for the subtree problems, *SIAM J. Comput.* **6** (1977), 730–732.
6. Y. Shiloah, "The Two Path Problem Is Polynomial," Technical Rep., No. STANCS-78-654, Computer Science Dept. Stanford Univ.
7. J. Valdes, Subtree homeomorphism and tree contractability, unpublished manuscript.