SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NY 14853-7501

TECHNICAL REPORT NO. 927

September 1990

AN ALGORITHM TO FIND A 2-ISOMORPHIC
DEPTH-FIRST SEARCH IMAGE
OF A TREE

by

Chun W. Ko

1This work has been supported in part by a Mathematical Sciences Institute Fellowship and NSF grants DDM-8813054 and DMS-8706133, and by AFOSR, NSF, and ONR under NSF grant No. DMS-8920550 to Cornell University.
An Algorithm to Find a 2-Isomorphic Depth-first Search Image of a Tree

Chun W. Ko

Abstract

Two graphs $G_1$ and $G_2$ are 2-isomorphic if there is a bijective mapping of the edges of $G_1$ onto the edges of $G_2$ that also preserves cycles. A graph-tree pair $(G, T)$ is a depth-first search pair if $T$ is a depth-first search tree in the connected graph $G$. If $T_1$ is a spanning tree of $G_1$, and $G_2$ is a 2-isomorphic image of $G_1$ such that the edges of $T_1$ induce a depth-first tree in $G_2$, then the graph-tree pair $(G_1, T_1)$ is 2-isomorphic to a depth-first search pair. This paper gives a linear time sequential algorithm to decide if a given graph-tree pair $(G, T)$ is 2-isomorphic to any depth-first search pair.

---

\[1\] This work has been supported in part by a Mathematical Sciences Institute Fellowship and NSF grants DDM-8813054 and DMS-8706133, and by AFOSR, NSF, and ONR under NSF Grant No. DMS-8929550 to Cornell University.
1 Introduction

Depth-first search is a fundamental and powerful graph search technique. Every depth-first search on a connected graph $G$ finds a spanning tree of $G$, called a depth-first search tree. Conversely, a spanning tree $T$ of $G$ is a depth-first search tree of $G$ if there is some depth-first search on $G$ that finds $T$. Several papers have studied the structural properties of depth-first search trees. Korach and Ostfeld gave both a linear time sequential algorithm and an $O(\log^2 |V|)$ time parallel algorithm to decide if a spanning tree $T$ of a connected graph $G$ is a depth-first search tree ([KO-a],[KO-b]). They also gave several characterizations of those graphs $G$ where every spanning tree of $G$ is a depth-first search tree of $G$ ([KO-b]). Ko and Bland ([KB]) characterized by excluded configurations all graph-tree pairs $(G, T)$ where the spanning tree $T$ is a depth-first search tree of $G$.

Let $G' = (V', E')$ and $G'' = (V'', E'')$ be two undirected graphs. $G'$ and $G''$ are 2-isomorphic, or cyclically isomorphic, if there is a bijection $\Phi : E' \rightarrow E''$ such that $C \subseteq E'$ is a cycle in $G'$ if and only if $\Phi(C)$ is a cycle in $G''$. $G'$ and $G''$ are 2-isomorphic if and only if $\mathcal{M}(G')$ and $\mathcal{M}(G'')$, the cycle matroids of $G'$ and $G''$ respectively, are isomorphic matroids. Shinoda et al. ([SCY+]) showed that every spanning tree of a series-parallel graph $G$ is a depth-first search tree of a 2-isomorphic copy of $G$. Syslo ([Sys]) gave an algorithm that for a given series-parallel graph $G$ and a spanning tree $T$ of $G$, finds a 2-isomorphic depth-first search image $(G', T')$ of $(G, T)$; i.e., there is an edge-map $\Phi$ such that $G' = \Phi(G)$ is 2-isomorphic to $G$ and $T' = \Phi(T)$ is a depth-first search tree of $G'$.

Let the basis $B$ of a graphic matroid $\mathcal{M}$ be called a depth-first search basis of $\mathcal{M}$ if there is a realization $(G, T)$ of the matroid-basis pair $(\mathcal{M}, B)$, where $T$ is a depth-first search tree of $G$. (See [KB].) Let $(G, T)$ be a realization of some given $(\mathcal{M}, B)$. If $G$ is a triconnected graph, then $(G, T)$ is the unique realization of $(\mathcal{M}, B)$, and it is easy to determine if $T$ is a depth-first search tree of $G$, and thus if $B$ is a depth-first search basis of $\mathcal{M}$. However, if $G$ is not triconnected and $T$ is not a depth-first search tree of $G$, there still may exist a 2-isomorphic depth-first search image of $(G, T)$. (Figure 1 contains such an example, where tree edges are represented by thick lines, and non-tree edges by thin lines.) In this paper a linear time sequential algorithm is developed that determines if a given graph-tree pair $(G, T)$ has any 2-isomorphic depth-first search images, and finds such an image if any exists.

Every connected graph can be decomposed into its triconnected components, each of which is...
Figure 1: A non-dfs pair and one of its dfs 2-isomorphic images.
a triconnected graph, a chordless cycle, or a set of parallel edges between two vertices. Hopcroft
and Tarjan gave an efficient procedure for this decomposition ([HT]). If two graphs \( G' \) and \( G'' \)
are 2-isomorphic, then there is a matching of the triconnected components of \( G' \) to those of \( G'' \)
so that matched triconnected components are isomorphic; furthermore, \( G' \) can be derived from \( G'' \)
by appropriately combining the triconnected components of \( G'' \). The crux of our algorithm is the
determination of if and how the triconnected components of the given graph-tree pair \((G, T)\) can be
combined to form a graph-tree pair \((G', T')\) such that \( T' \) is a depth-first search tree of \( G' \).

The remainder of this paper is organized as follows. Section 2 establishes the notation and
terminology used in this paper. Section 3 investigates the properties of the decomposition into
triconnected components. The main algorithm is developed in Section 4. The general graph notation
and terminology used in this paper are consistent with those in [BM] and [KB].

2 Notation and Terminology

Let \( G \) be a connected graph, and let \( T \) be a spanning tree of \( G \). If there is a depth-first search on
\( G \) that gives \( T \) as the resulting spanning tree, then \( T \) is a \textit{dfs tree} of \( G \), and \((G, T)\) is a \textit{dfs pair}. If
the depth-first search that finds \( T \) begins at vertex \( r \), then \( r \) is the \textit{dfs root} of the dfs tree \( T \) of \( G \).
If the edges in \( T \) are directed away from the vertex \( r \), then the vertex \( u \in T \) is an \textit{ancestor} of the
vertex \( v \in T \), and \( v \) is a \textit{descendant} of \( u \), if there is a directed path in \( T \) from \( u \) to \( v \). If \((u, v)\) is a
tree edge and \( u \) is an ancestor of \( v \), then \( u \) is a \textit{parent} of \( v \) and \( v \) is a \textit{child} of \( u \).

A \textit{mixed graph} is a graph where each edge \((u, v)\) is undirected, or is directed from \( u \) to \( v \), from
\( v \) to \( u \), or in both directions. The \textit{underlying graph} \( H \) of a mixed graph \( H \) has the same set of
vertices as \( H \), and \((u, v)\) is an undirected edge in \( H \) if and only if \((u, v)\) is a directed edge in \( H \). The
\textit{in-degree} of a vertex \( u \) in a mixed graph \( H \) is the number of (directed) edges directed into \( u \), and
the \textit{out-degree} of \( u \) is the number of (directed) edges directed out of \( u \). A \textit{root} in a mixed graph is
a vertex with in-degree zero, and a \textit{leaf} in a mixed graph is a vertex with out-degree zero. A leaf
in an undirected graph is a vertex with degree one. A mixed graph \( R \) is a \textit{rooted tree} with a fixed
root \( r \) if every edge is directed only one way, each vertex in \( R - r \) has in-degree one, and \( r \) has
in-degree zero. A mixed graph \( H' \) is embedded in the mixed graph \( H \) if \( H' \) is a subgraph of \( H \), and
\((u, v) \in H' \) is directed from \( u \) to \( v \) only if \((u, v) \) is also directed from \( u \) to \( v \) in \( H \).
Let \((G,T)\) be a dfs pair with dfs root \(r\), and let \(T\) be directed such that \(T\) is a rooted tree with root \(r\). A property of depth-first search is that if \((u,v)\) is an edge in \(G\) but not an edge in \(T\), then \(u\) is either an ancestor or a descendant of \(v\). If the non-tree edges in \(G\) are directed from descendants to ancestors, then all fundamental cycles with respect to \(T\) in \(G\) are directed. The pair \((G,T)\) has a \textit{dfs orientation} if the edges in \(G\) can be directed such that \(T\) is a rooted tree and all fundamental cycles with respect to \(T\) in \(G\) are directed. \((G,T)\) is a dfs pair if and only if it has a dfs orientation ([KB]).

3 Decomposition of a Graph-tree Pair

The following is a recap of the decomposition theory in [HT]. The reader is referred to the original paper for a detailed discussion.

Let \(G\) be a biconnected, but not triconnected, undirected graph, and \(\{u,v\}\) a vertex cut of \(G\). Then the edges of \(G\) can be divided into non-empty equivalence classes \(E_1, \ldots, E_k\) such that

1. two edges which lie on a common path not containing either \(u\) or \(v\) except as an end vertex are in the same class;

2. if \(k = 2\) then both \(E_1\) and \(E_2\) contain more than one edge; and

3. if \(k = 3\) then at least one of \(E_1\), \(E_2\) or \(E_3\) contains more than one edge.

Let \(E' = \bigcup_{i=1}^{j} E_i\), \(E'' = \bigcup_{i=l+1}^{k} E_i\), such that \(|E'| \geq 2\) and \(|E''| \geq 2\). Then \(G\) can be \textit{split} into two \textit{split graphs} \(G_1\) and \(G_2\) at \(\{u,v\}\), where \(G_1 = (V(E'), E' + (u,v;\alpha))\) and \(G_2 = (V(E''), E'' + (u,v;\alpha))\). The edges \((u,v;\alpha)\) added to \(E'\) and \(E''\) are \textit{split edges}, where \(\alpha\) identifies this particular splitting operation. Hence associated with each splitting of \(G\) are two unique split edges, one in each split graph. The split edges need to be identified with the split operation, for there may be more than one splitting at \(\{u,v\}\).

If two graphs \(G_1 = (V_1, E_1)\) and \(G_2 = (V_2, E_2)\) both contain the split edge \((u,v;\alpha)\), then \(G_1\) and \(G_2\) can be \textit{merged} into a single graph \(G\) at \(\{u,v\}\), where \(G = (V_1 \cup V_2, E_1 - (u,v;\alpha) \cup E_2 - (u,v;\alpha))\). Let \(G\) be split at \(\{u,v\}\) into \(G_1, G_2\). If the labels of \(u\) and \(v\) are interchanged in \(G_2\), and \(G'\) is the graph resulting from merging \(G_1\) and the new \(G_2\), then \(G\) and \(G'\) are 2-isomorphic, and \(G'\) is said
to be derived from $G$ by a \textit{twisting} at $\{u, v\}$. Two graphs are 2-isomorphic if and only if one can be derived from the other by a sequence of twistings [Wel].

By a sequence of splittings, a graph $G$ can be decomposed into three sets of graphs $\mathcal{B}$, $\mathcal{H}$ and $\mathcal{T}$, where each graph in $\mathcal{B}$ has two vertices and at least three parallel edges (a \textit{bond}), each graph in $\mathcal{H}$ is a chordless cycle with at least three vertices (a \textit{hole}), and each graph in $\mathcal{T}$ is triconnected, such that $|\mathcal{B}|, |\mathcal{H}|$ are minimal (i.e. one cannot merge bonds into bigger bonds, and holes into bigger holes). The set $B \cup H \cup T$ is called the set of \textit{triconnected components} of $G$.

\textbf{Lemma 1. [Mac] [HT]} \textit{The triconnected components of a graph $G$ are unique.}

Note that if $\{u, v\}$ is a vertex cut in $G$, then either there is some split edge $(u, v; \alpha)$ in some triconnected components of $G$, or $u$ and $v$ are non-adjacent vertices in some hole in $\mathcal{H}$.

The decomposition of graphs into triconnected components can be extended to a decomposition of graph-tree pairs into smaller graph-tree pairs. Let $(G, T)$ be a graph-tree pair, and $\{u, v\}$ a vertex cut. If $G$ is split at $\{u, v\}$, let $G_1$ and $G_2$ be defined as before, and let $T_1' = E' \cap T$ and $T_2' = E'' \cap T$. If $T_i'$ is not connected, let $T_i = T_i' + (u, v; \alpha)$, and otherwise let $T_i = T_i'$. It follows that the \textit{split pairs} $(G_1, T_1)$ and $(G_2, T_2)$ are also graph-tree pairs. Conversely, merging $(G_1, T_1)$ and $(G_2, T_2)$ results in a graph-tree pair. If $G_i$ is a triconnected component of $G$, then $(G_i, T_i)$ is a \textit{triconnected pair} of the pair $(G, T)$.

Let the underlying graph $G$ of the pair $(G, T)$ be decomposed into its triconnected components in $B' \cup H' \cup T'$. Let $B$, $H$ and $T$ denote the corresponding sets of triconnected pairs of $(G, T)$.

\textbf{Lemma 2.} \textit{If $G$ is not a triconnected graph, a hole or a bond, then there are at least two graphs in $B' \cup H' \cup T'$ with exactly one split edge.}

\textbf{Proof.} Let $G$ be a counter-example with minimum number of triconnected components. Let $(u, v; \alpha)$ be a split edge contained in $X$ and $Y$, two triconnected components of $G$. Split $G$ at $\{u, v\}$ into $G'$, $G''$ such that $X$ (respectively $Y$) is a triconnected component of $G'$ (respectively $G''$). If $G'$ is a triconnected graph, a hole or a bond, then $G'$ is the only triconnected component in the decomposition of itself, and thus $G'$ contains only one split edge, $(u, v; \alpha)$, in the decomposition of $G$. Otherwise, in the decomposition of $G'$ there are at least two triconnected components such that each has exactly one split edge, and at most one of these triconnected components can contain
(u, v; α). Hence at least one triconnected component of G', and thus of G, contains only one split edge in the decomposition of G. Similarly there is such a triconnected component of G''.

Lemma 3. The triconnected pairs decomposition of a graph-tree pair (G, T) is unique.

Proof. Let P = B U H U T be the set of triconnected pairs of (G, T), where (G, T) is a counterexample to the lemma with minimum |P|. Let C = B' U H' U T' be the set of triconnected components of G. Clearly the underlying graphs of the triconnected pairs in P are exactly the graphs in C. By Lemma 2, there is a triconnected component X in C such that X contains only one split edge (u, v; α). Split (G, T) at {u, v} into (X, TX) and (G', T'). By definition, the choices of TX and T' as spanning trees of X and G' are unique. If P' is the unique set of triconnected pairs of (G', T') then it follows that P = P' U {(X, TX)} is also unique, a contradiction.  

See Figure 2 for the decomposition of the graph-tree pair (G', T') shown in Figure 1. In the figure split edges are represented by dash lines and are indexed by capital letters, and the triconnected pairs are indexed by lower case letters.

Let P be the set of triconnected pairs of (G, T) and let P' be a subset of P.

Definition 4. M(P') is the set of graphs resulting from doing all possible merges of triconnected pairs in P', and replacing all unused split edges (u, v; α) by normal edges (u, v).

The next two lemmas show how splitting and merging preserve dfs-ness.

Lemma 5. Let (G1, T1) and (G2, T2) be two dfs pairs with a common split edge (u, v; α), such that u is a dfs root of (G2, T2) and u is an ancestor of v in some dfs orientation. Then the pair (G, T) resulting from merging (G1, T1) and (G2, T2) is also a dfs pair.

Proof. Give (G1, T1) and (G2, T2) dfs orientations such that u is an ancestor of v in T1, and u is the dfs root in (G2, T2). With the edges of (G, T) inheriting their orientations from (G1, T1) and (G2, T2), the fundamental cycles of (G, T) are also directed. The ancestors of u in (G, T) are exactly those of u in (G1, T1). Hence there can be only one root in (G, T), and therefore (G, T) is a dfs pair.

Lemma 6. Let (G, T) be a dfs pair with dfs root r and let {u, v} be a vertex cut, such that splitting (G, T) at {u, v} results in the split pairs (G1, T1) and (G2, T2). Then both (G1, T1) and (G2, T2) are
Figure 2: Decomposition of a graph-tree pair.
 dfs pairs. In addition, if there is a copy of \( r \) in \((G_1, T_1)\), then either \( u \) or \( v \) is a dfs root in \((G_2, T_2)\), and vice-versa.

**Proof.** Direct the edges in \( G \) with respect to the dfs orientation where \( r \) is the dfs root. Without loss of generality let \( u \) be an ancestor of \( v \) in \( T \) in this orientation. If the split edge \((u, v; \alpha)\) is an edge in \( T_i, i = 1, 2 \), direct it from \( u \) to \( v \) in \( T_i \); otherwise direct it from \( v \) to \( u \). Let the other edges in \( G_i, i = 1, 2 \), inherit their orientations from \( G \). In this orientation, all fundamental cycles in \((G_i, T_i)\), \( i = 1, 2 \), are also directed. If there is a copy of \( r \) in \((G_1, T_1)\), then the directed path in \( T \) from \( r \) to \( u \) is completely contained in \( T_1 \). Thus \( u \) must be a root in \( T_2 \), and \( T_1 \) and \( T_2 \) are rooted trees. Hence \((G_1, T_1)\) and \((G_2, T_2)\) are dfs pairs. \( \square \)

4 The Algorithm

Let \((G, T)\) be a graph-tree pair where \( G \) is a biconnected graph and \( \mathcal{P} \) is the set of triconnected pairs in the decomposition of \((G, T)\). As noted before, twisting \((G, T)\) at a vertex cut \{\( u, v \)\} gives the graph-tree pair in \( M(\mathcal{P}) \) after either twisting a hole containing \( u \) and \( v \) as non-adjacent vertices, or interchanging the labeling of \( u \) and \( v \) in some triconnected pair with the split edge \((u, v; \alpha)\).

The information of how \((G, T)\) can be twisted is abstracted in the mixed graph constructed as follows. If every triconnected pair in \( \mathcal{P} \), the set of triconnected pairs of \((G, T)\), is a dfs pair, then let \( S(G, T) = (P, D) \) be the graph where

1. there is a vertex \( z \) in \( P \) if and only if there is a triconnected pair \((X, T_X)\) in \( \mathcal{P} \);
2. \((z, y)\) is an edge in \( D \) if and only if there is a split edge \( e = (u, v; \alpha) \) such that \( e \in X \) and \( e \in Y \);
3. the edge \((x, y)\) is directed from \( x \) to \( y \) if
   (a) either \( u \) or \( v \) is a dfs root of \( T_Y \) in \( Y \), or
   (b) \((Y, T_Y)\) is a hole \(((Y, T_Y) \in \mathcal{H})\).

Figure 3 contains the graph \( S(G, T) \) corresponding to the decomposition depicted in Figure 2.

The graph \( S(G, T) \) possesses the following properties.
Lemma 7. \( S(G, T) \), the underlying undirected graph of \( S(G, T) \), is a tree.

Proof. Induct on \(|P|\). The lemma is trivially true if \(|P| = 1\). Suppose the lemma holds for \( S(G', T') \), where \((G', T')\) is any pair with less than \(k\) triconnected pairs in its decomposition. Let \((G, T)\) be a pair with \(|P| = k\). By Lemma 2, there is a triconnected pair \((X, T_X)\) in \(P\) with only one split edge, say \((u, v; \alpha)\). Let \((G_1, T_1)\) be the only graph-tree pair in \(M(P - (X, T_X))\). \(S(G_1, T_1)\) is the subgraph of \(S(G, T)\) with the vertex \(x\) removed, and by the induction hypothesis \(S(G_1, T_1)\) is a tree. It follows that \(S(G, T)\) is also a tree. \(\square\)

The structure of \(S(G, T)\) is preserved under twistings.

Lemma 8. If \((G_1, T_1)\) is a pair all of whose triconnected pairs are dfs pairs, and \((G_2, T_2)\) is 2-isomorphic to \((G_1, T_1)\), then \(S(G_1, T_1)\) is isomorphic to \(S(G_2, T_2)\).

Proof. It is an easy observation that if \((G_1, T_1)\) and \((G_2, T_2)\) are 2-isomorphic, then \(S(G_1, T_1)\) and \(S(G_2, T_2)\) are isomorphic undirected trees. Thus it may be assumed that \(S(G_1, T_1)\) and \(S(G_2, T_2)\) have the same set of vertices. The lemma is proved by showing that every edge \((x, y) \in S(G_1, T_1)\) is directed as it is in \(S(G_2, T_2)\). Let \(P_1\) be the set of triconnected pairs of \((G_1, T_1)\), and \(P_2\) be that of \((G_2, T_2)\). Let \((x, y)\) be an arbitrary edge in \(S(G_1, T_1)\) (and thus in \(S(G_2, T_2)\)). Let \((X_1, T_{X_1})\)
and \((Y_1, T_{Y_1})\) be the pairs in \(P_1\) corresponding to \(x\) and \(y\), respectively, and \((X_2, T_{X_2}), (Y_2, T_{Y_2})\) be their counterparts in \(P_2\). Then there is a unique split edge \((u_1, v_1; \alpha_1)\) in both \(X_1\) and \(Y_1\), and a corresponding split edge \((u_2, v_2; \alpha_2)\) in both \(X_2\) and \(Y_2\). Without loss of generality let \((x, y)\) be directed from \(x\) to \(y\) in \(S(G_1, T_1)\). Then either \(Y_1\) is a hole, or one of \(u_1, v_1\) is a dfs root of \((Y_1, T_{Y_1})\). In twisting \((G_1, T_2)\) to obtain \((G_2, T_2)\), \(Y_1\) maybe twisted to become \(Y_2\), or the labels of \(u_1\) and \(v_1\) in \(Y_1\) are interchanged to become \(u_2\) and \(v_2\) in \(Y_2\), or neither occur. In any case, \((x, y)\) is also directed from \(x\) to \(y\) in \(S(G_2, T_2)\). By symmetry then, the orientation, or the lack of one, of \((x, y)\) is the same in \(S(G_1, T_1)\) and \(S(G_2, T_2)\) for every edge \((x, y)\). Hence \(S(G_1, T_1)\) and \(S(G_2, T_2)\) are isomorphic. □

If \((G, T)\) is a dfs pair, then there is a rooted tree embedded in \(S(G, T)\).

**Lemma 9.** If \((G, T)\) is a dfs pair with dfs root \(r \in G\), then for every triconnected pair \((X, T_X) \in P\) that contains a copy of \(r\), there is a rooted tree \(R\) with root \(x\) embedded in \(S(G, T)\).

**Proof.** The proof is by induction on \(|P|\), where \(P\) is the set of triconnected pairs in the decomposition of \((G, T)\). If \(|P| = 1\), then there is nothing to prove. Suppose the lemma is true for any graph-tree pair satisfying the lemma's hypothesis and has less than \(k\) triconnected pairs in its decomposition. Let \((G, T)\) be a dfs pair with \(|P| = k\), and let \((X, T_X) \in P\) be a triconnected pair where \(X\) contains only one split edge \((u, v; \alpha)\). Let \((Y, T_Y)\) be the other triconnected pair containing the split edge \((u, v; \alpha)\). Then there is only one edge \(e = (x, y)\) incident with the vertex \(x\) in \(S(G, T)\). Let \((G', T')\) be the only pair in \(M(P')\), where \(P' = P - (X, T_X)\).

Suppose there is a copy of \(r\) in \(X\). Then by Lemma 6 and the definition of \(S(G, T)\), the edge \(e\) is directed from \(x\) to \(y\). (It may also be directed from \(y\) to \(x\)). In addition, either \(u\) or \(v\) is a dfs root of \((G', T')\), and the pair \((Y, T_Y) \in P'\) contains a copy of both \(u\) and \(v\). By the induction hypothesis, there is a rooted tree \(R'\), with root \(y\), embedded in \(S(G', T')\). Hence, \(R' + e\) is a rooted tree, with root \(x\), embedded in \(S(G, T)\).

If there is a copy of \(r\) in some triconnected pair \((Z, T_Z) \in P'\), then the edge \(e\) is directed from \(y\) to \(x\). Since \(r\) is also a dfs root of \((G', T')\), by the induction hypothesis there is a rooted tree \(R'\), with root \(x\), embedded in \(S(G', T')\). Hence \(R' + e\) is a rooted tree, with root \(x\), embedded in \(S(G, T)\). □

The converse of Lemma 9 does not hold in general. However, if a rooted tree \(R\) is embedded in \(S(G, T)\), then a 2-isomorphic dfs image \((G_d, T_d)\) of \((G, T)\) exists. Indeed, the rooted tree \(R\) shows how \((G_d, T_d)\) can be constructed.
Lemma 10. \((G,T)\) is 2-isomorphic to a dfs pair if and only if

1. every triconnected pair of \((G,T)\) is a dfs pair, and

2. a spanning rooted tree is embedded in the graph \(S(G,T)\).

**Proof.** Let \(\mathcal{P}\) be the set of triconnected pairs of \((G,T)\). The lemma is proved by induction on \(|\mathcal{P}|\). If \(|\mathcal{P}| = 1\) then the lemma follows trivially. Suppose the lemma holds for any graph-tree pair with less than \(k\) triconnected pairs in its decomposition, and let \((G,T)\) be a pair such that \(|\mathcal{P}| = k\).

Suppose all the triconnected pairs of \((G,T)\) are dfs pairs and there is a rooted tree \(R\) with root vertex \(r\) embedded in \(S(G,T)\). Let \(x\) be a leaf of \(R\). Then there is a triconnected pair \((X,T_X)\) in \(\mathcal{P}\) corresponding to \(x\) with just one split edge \(e = (u,v;\alpha)\). Let \((G',T')\) be the only pair in \(M(\mathcal{P} \setminus (X,T_X))\). Then \(S(G',T')\) has an embedded rooted tree \(R - x\). By the induction hypothesis, \((G',T')\) is 2-isomorphic to some dfs pair \((G'',T'')\). By the definition of the edges of \(S(G,T)\), either \(u\) or \(v\) is a dfs root of \((X,T_X)\), or \((X,T_X)\) is a hole, in which case the hole can be twisted such that \(u\) is a dfs root. By Lemma 5 merging \((G'',T'')\) and \((X,T_X)\) results in a dfs pair, which is 2-isomorphic to \((G,T)\).

For the converse, suppose \((G,T)\) is 2-isomorphic to some dfs pair \((G',T')\). By Lemma 6, the triconnected pairs of \((G',T')\) are all dfs pairs, and so those of \((G,T)\) are also dfs pairs. By Lemmas 8 and 9, there must be a rooted tree embedded in \(S(G,T)\). \(\square\)

Lemma 10 is the basis of the following algorithm to determine if a pair \((G,T)\) is 2-isomorphic to a dfs pair.

---

**Algorithm 2ISODFS**

**input:** A pair \((G,T)\), where \(G\) is a biconnected graph and \(T\) is a spanning tree of \(G\).

**output:** A dfs pair \((G_d,T_d)\) that is 2-isomorphic to \((G,T)\) if any exists; otherwise \(\emptyset\).

**comment:** Decompose \((G,T)\) into its triconnected pairs.

\((\mathcal{P},E_S) = \text{TRICON\_PAIRS}((G,T));\)

\((E_S\) is the list of split edges.)
comment: Find the possible dfs roots of each triconnected pair.
for each pair \((G', T') \in \mathcal{P}\) do
    if \(\text{DFS.ROOTS}((G', T')) = \emptyset\) then RETURN(\emptyset);

\(S(G, T) = \text{BUILD}\_S(\mathcal{P}, E_S)\);

comment: Find a rooted tree \(R\), if any exists, embedded in \(S(G, T)\).
\((R, r) = \text{FIND.ROOT}(S(G, T))\);
if \((R = \emptyset)\) then RETURN(\emptyset);

\((G_d, T_d) = \text{GROW}(R, r, \mathcal{P})\);
RETURN((G_d, T_d));

END.

---

**procedure** TRICON_PAIRS

**input:** A pair \((G, T)\), where \(G\) is a biconnected graph and \(T\) is a spanning tree of \(G\).

**output:** A set of triconnected pairs \(\mathcal{P} = \mathcal{B} \cup \mathcal{H} \cup \mathcal{T}\) from

the decomposition of \((G, T)\), and the list of all split edges, \(E_S\).

comment: Use the procedure of Hopcroft and Tarjan to decompose \(G\).
\((\mathcal{P}, E_S) = \text{TRICONNECTIVITY}(G)\);

Construct the graph \(S' = S(G, T)\);

while \((S'\) has a leaf \(x))\) do
    comment: \(X \in \mathcal{P}\) has only one split edge \(e\).
    \(y = \text{neighbour of } x \in S'\);
    if \((T(X)\) is connected)
        then mark \(e\) as a non-tree edge in \(X\):
            mark \(e\) as a tree edge in \(Y\);
        else mark \(e\) as a tree edge in \(X\);
mark $e$ as a non-tree edge in $Y$;

$S' = S' - x$;

RETURN($P$, $E_S$);

END.

---

**procedure DFS_ROOT**

**input:** A graph-tree pair $(G, T)$, where $G$ is biconnected.

**output:** All possible dfs roots if $(G, T)$ is a dfs pair; otherwise $\emptyset$.

Use the algorithm from [KO-a].

END.

---

**procedure BUILD_S**

**input:** $E_S$, the list of split edges; $P$, the set of triconnected pairs of $(G, T)$.

**output:** The graph $S(G, T)$.

$P = \emptyset$

for each pair $(X, T_X) \in P$ do

$P = P + x$

for each $(u, v; \alpha) \in E_S$ do

$(X, Y) =$ the two pairs in $P$ containing $(u, v; \alpha)$;

$S(G, T) = S(G, T) + (x, y)$;

if (($u$ or $v$ is possible dfs root in $(X, T_X)$) or

$((X, T_X)$ is a hole))

direct $(x, y)$ from $y$ to $x$;

if (($u$ or $v$ is possible dfs root in $(Y, T_Y)$) or

$((Y, T_Y)$ is a hole))

direct $(x, y)$ from $x$ to $y$;

RETURN($S(G, T)$);

END.
procedure FIND_ROOT
input: The graph $S(G, T)$.
output: A rooted tree $R$, with root $r$, embedded in $S(G, T)$.
while ($S(G, T)$ has more than one vertex) do
    if ($S(G, T)$ has no leaf) RETURN($\emptyset$);
    $y =$ leaf of $S(G, T)$;
    $S(G, T) = S(G, T) - y$;
    $r =$ remaining vertex in $S(G, T)$;
    $R =$ subtree tree of $S(G, T)$ with edges directed away from $r$;
RETURN($R, r$);
END.

procedure GROW
input: A rooted tree $R$ with root $r$, a set of triconnected pairs $\mathcal{P}$.
output: A dfs pair $(G_d, T_d)$ 2-isomorphic to the pair $(G, T)$.
$(G_r, T_r) =$ pair in $\mathcal{P}$ corresponding to the vertex $r \in S(G, T)$;
$u =$ a possible dfs root of $(G_r, T_r)$;
$(G_d, T_d) =$ RECUR_GROW($R, r, u, \mathcal{P}$);
RETURN($(G_d, T_d)$);
END.

procedure RECUR_GROW
input: A rooted tree $R$ with root $r$, a vertex $u \in G_r \in \mathcal{P}$.
output: A dfs pair $(G', T')$, with dfs root $u$, resulting from merging
all pairs corresponding to descendants of $r$ in $R$.
direct edges in $T_r$ away from $u$;
\[(G', T') = (G_r, T_r);\]

for each child \( r \) of \( r \) in \( R \) do

\[(x, y, \alpha) = \text{split edge in both } G_r \text{ and } G_t;\]

if (neither \( x \) nor \( y \) is a dfs root in \( (G_t, T_t) \)) do

comment: \( (G_t, T_t) \) must be a hole.

\[\text{twist } (G_t, T_t) \text{ such that } x \text{ is an end of } T_t;\]

\[x' = \text{a possible dfs root of } (G_t, T_t);\]

\[(G'', T'') = \text{RECUR.GROW}(R, t, x', \mathcal{P});\]

\[(G', T') = \text{pair resulting from the merging of } (G', T') \text{ and } (G'', T''),\]

identifying \( x \in G' \) with \( x' \in G'';\]

RETURN\(((G', T')));\]

END.

It is straight-forward to verify that the procedures are correct.

5 Complexity of Algorithm 2ISODFS

Clearly, the algorithm can be made efficient only if the decomposition of \( G \) does not introduce too many new edges and vertices. Let \( E \) be the edge set of \( G \), and let \( E(\mathcal{P}) \) be set of all edges in all graphs in \( \mathcal{P} \), the set of triconnected pairs in the decomposition of \( (G, T) \).

Lemma 11. \( ([HT]) \ |E(\mathcal{P})| \leq 3|E| - 6.\)

Now we analyze the complexity of each major step in Algorithm 2ISODFS.

In procedure TRICON PAIRS, the call to TRICONNECTIVITY finds the triconnected components of \( G \) in \( O(|V| + |E|) \) time ([HT]). With a slight modification of the procedure of Hopcroft and Tarjan, the list of all split edges can also be constructed, such that given a split edge \( e \), the two split components containing \( e \) can be found in \( O(1) \) time. Then the graph \( S' \) can be constructed, like the graph \( S(G, T) \) in procedure BUILD S, in \( O(|E|) \) time. In the while-loop in procedure TRICON PAIRS, each triconnected pair is searched only once; and the set of leafs of \( S' \), by updating
the degree of vertex $y$ in $S'$ at the end of each iteration of the while-loop, can be maintained in $O(|\mathcal{P}|)$ time. Thus the while-loop itself has complexity $O(|E|)$. Hence procedure TRICON_PAIRS takes only $O(|E|)$ time.

Using the procedure of Korach and Ostfeld ([KO-a]), the possible roots of each pair $(G', T') \in \mathcal{P}$ can be found in $(|E(G')|)$ time. Hence procedure DFS_ROOTS takes at most $O(|E|)$ time.

In procedure BUILD_S, checking if a vertex $u$ is a dfs root in a triconnected pair takes only $O(1)$ time after procedure DFS_ROOTS has been called, since each triconnected pair can have at most two possible dfs roots [KO-a]. Thus the graph $S(G, T)$ is constructed in $O(|\mathcal{P}| + |E_S|)$ time.

In procedure FIND_ROOT, each vertex in $S(G, T)$ can be a leaf, and thus can be removed from $S(G, T)$, at most once. Clearly, directing the edges in $S(G, T)$ away from the vertex $r$ to construct the rooted tree $R$ requires only $O(|\mathcal{P}| + |E_S|)$ time. Hence procedure FIND_ROOT has complexity $O(|E|)$.

In procedure RECUR_GROW, directing the edges in $T_r$ takes $O(|E(G_r)|)$ time. Merging two graph-tree pairs $(G', T')$ and $(G'', T'')$, where $u' \in G'$ and $u'' \in G''$ are identified, as are $v' \in G'$ and $v'' \in G''$, involves only appending the adjacency structures of $u''$ and $v''$ to those of $u'$ and $v'$, respectively, which can be done in $O(1)$ time. The $O(1)$ work in the for-loop for each child $t$ of $r$ can be charged to the recursive call RECUR_GROW($R, t, x', \mathcal{P}$). Hence procedure RECUR_GROW, and thus procedure GROW, only requires $O(|E|)$ time.

**Proposition 12.** Given a graph-tree pair $(G, T)$, where $G$ is a biconnected graph, Algorithm 2ISODFS determines if $(G, T)$ is 2-isomorphic to a dfs pair, and if so finds such a pair $(G_d, T_d)$, in $O(|E(G)|)$ time.

If the pair $(G, T)$ is given and $G$ is not biconnected, the biconnected components $G_i$ of $G$ may be found in linear time, and Algorithm 2ISODFS can then be applied to each $G_i$. If any $(G_i, T_i)$ is not 2-isomorphic to a dfs pair, then $(G, T)$ itself cannot be 2-isomorphic to a dfs pair. If $(G_i, T_i)$ is the dfs pair 2-isomorphic to $(G_i, T_i)$, with dfs root $r_i$, and $(G', T')$ is the pair obtained by identifying the roots $r_i$ in all $(G_i, T_i)$, then $(G', T')$ is a dfs pair 2-isomorphic to $(G, T)$.

**References**


