Simpler Computation of Minimal Removable Sets in 2-Edge-Connected Systems for Undirected Graphs

SHOTA, Kan HARAGUCHI, Kazuya

November 27, 2025

1 Introduction

2 Preliminaries

Throughout the paper, we assume that a graph is simple and undirected.

For a graph $G = (V, E)$ with a vertex set V and an edge set E, we may abbreviate an edge $\{u, v\} \in E$ into uv (or equivalently vu) for simplicity. For a vertex $v \in V$, we denote by $N_G(v)$ the set of neighbors of v. For $S \subseteq V$, we denote by $G[S]$ the subgraph of G induced by S. We denote by $G - S$ the subgraph obtained by removing S and all edges to incident to S, that is, $G - S = G[V - S]$. If $S = \{v\}$, then we simply write $G - v$ instead of $G - \{v\}$.

A *DFS-tree* of a connected undirected graph $G = (V, E)$ is a spanning tree T obtained by performing a depth-first search (DFS) on G . An edge in E is called a *tree edge* if it belongs to T, and a *back edge* otherwise. Let T be rooted at a vertex $r \in V$. For $u, v \in V$, we say that u is a *ancestor* of v if u lies on the unique r–v-path in T. If u is an ancestor of v and $u \neq v$, then we say that u is a *proper ancestor* of v, and that v is a *descendant* (resp., a *proper descendant*) of u. If uv is a tree edge and u is an ancestor of v, then u is the *parent* of v; we denote it by $\pi(v) = u$.

During the DFS, each vertex is assigned a unique integer called *DFS-index*, which indicates the order of visiting time in the DFS. For $v \in V$, we denote by $\mathit{dfsId}(v)$ the DFS-index of v. An ancestor has a smaller DFS-index than any of its descendants.

We may orient edges based on the DFS-tree T: for a tree edge $\pi(v)v$, we orient it from $\pi(v)$ to v; and for a back edge uv, we orient it from the descendant to the ancestor. We may write (u, v) to indicate the orientation of edge uv and refer to it as the *outgoing edge* from u and the *incoming edge* to v. For $v \in V$, we denote by T_v the subtree of T rooted at v.

We say that a tree edge $\pi(v)v$ is *covered* by a back edge $e = (x, y)$ if x is a proper descendant of v and y is a proper ancestor of v. In this case, e together with $\pi(v)v$ forms part of a cycle in G. Note that every tree edge in a 2-edge-connected graph is covered by at least one back edge. For a tree edge uv, we denote by $\mathit{Cov}(uv) =$

Algorithm 1 Computation of Cover Multiplicities by DFS

Input: A 2-edge-connected undirected graph $G = (V, E)$, its DFS-tree T rooted at $r \in V$, cover-delta values δ for all vertices in V, and a tree edge (u, v) of G

Output: The cover multiplicities of all tree edges in T_v

```
procedure CoverMultiplicity(G, T, δ, (u, v))
    Initinialize CovMult(uv) := δ(v);
    for each child w of v in T do
        Execute CoverMultiplicity(G, T, δ, (v, w));
        CovMult(uv) := CovMult(uv) + CovMult(vw)
    end for;
    return CovMult(uv);
end procedure
```

$\{(x, y) \in E \mid (x, y)$ is a back edge covering $uv\}$ the set of back edges that cover uv. We refer to $\mathit{CovMult}(uv) := |\mathit{Cov}(uv)|$ as the *cover multiplicity* of the tree edge uv. Furthermore, we define the *cover-source multiplicity* of uv to be $\mathit{CovSrcMult}(uv) = \big|\{\, x \mid$ There is y s.t. $(x, y) \in \mathit{Cov}(uv)\,\}\big|$, that is, the number of distinct descendants that serve as outgoing endpoints of back edges covering uv.

For a vertex $v \in V$, every vertex u with a back edge vu lies on the unique $\pi(v)$–r path in T. Among such vertices u, let u^* be the one with the smallest DFS-index, that is, the one closest to $\pi(v)$ in T. We call the back edge vu^* the *longest back edge outgoing from* v and denote it by $\mathit{lbe}(v)$ and u^* by $\mathit{lbeTarget}(v)$.

3 Computation of Cover multiplicities

In this section, we present an $O(n + m)$-time algorithm to compute the cover multiplicity and the cover-source multiplicity for all tree edges in a 2-edge-connected undirected graph $G = (V, E)$ with $n = |V|$ and $m = |E|$.

The algorithm consists of two phases. In the first phase, we process each back edge to update an auxiliary value named *cover-delta value* associated with its endpoints (see Algorithm 2). In the second phase, we perform a DFS on T to aggregate these values along the tree and compute the multiplicity for each tree edge (see Algorithm 1). The details are described below.

If we want to compute the cover-source multiplicities, we can modify Algorithm 2 so that, in line 2 of Algorithm 2, we only consider the longest back edges outgoing from each vertex.

Lemma 1 *Given a 2-edge-connected undirected graph $G = (V, E)$ with $n = |V|$ and $m = |E|$, its DFS-tree T rooted at $r \in V$, and the cover-delta values δ for all vertices in V, Algorithm 1 computes the cover multiplicities for all tree edges in T_v in $O(|V(T_v)|)$ time for a tree edge (u, v) of G.*

Algorithm 2 Preprocessing Back Edges

Input: A 2-edge-connected undirected graph $G = (V, E)$ and its DFS-tree T rooted at $r \in V$

Output: The cover-delta values δ for all vertices in V

1: Initialize $\delta(v) := 0$ for all $v \in V$;
2: **for each** back edge (x, y) of G **do**
3: $\quad\delta(x) := \delta(x) + 1$;
4: $\quad\delta(y) := \delta(y) - 1$
5: **end for**;
6: **return** δ

PROOF: We first show the case with v being a leaf. In this case, since there is no child of v in T, the algorithm returns $\mathit{CovMult}(uv) = \delta(v)$ that is equal to the number of back edges outgoing from v that cover uv. This is correct by the definition of δ. Next, we consider the case with v being an internal vertex or the root. By the definition of δ, the sum of δ values for all vertices in T_v is equal to the number of back edges outgoing from vertices in T_v that cover uv, which is equal to $|\mathit{Cov}(uv)|$. The algorithm aggregates the δ values for all vertices in T_v by performing a DFS on T_v, and hence, it correctly computes $|\mathit{Cov}(uv)|$. The time complexity is $O(|V(T_v)|)$, since the algorithm visits each vertex in T_v exactly once. □

Theorem 1 *Given a 2-edge-connected undirected graph $G = (V, E)$ with $n = |V|$ and $m = |E|$, we can compute the cover multiplicity and the cover-source multiplicity for all tree edges in G in $O(n + m)$ time.*

PROOF: We first perform a DFS on G to obtain its DFS-tree T rooted at an arbitrary vertex $r \in V$ in $O(n + m)$ time. Next, we execute Algorithm 2 to compute the cover-delta values for all vertices in V in $O(n + m)$ time. Finally, we execute Algorithm 1 with each tree edge (r, v) incident to the root r to compute the cover multiplicities for all tree edges in T. By Lemma 1, this step takes $O(n)$ time. Thus, the total time complexity is $O(n+m)$. □

4 Mathematical Properties of Minimal Removable Sets

4.1 DFS-Tree Leaf MinRSs

In this section, we consider MinRSs that consist of only leaves of a DFS-tree.

We first show a basic property of leaves in a DFS-tree of a 2-edge-connected undirected graph.

Lemma 2 *Let $G = (V, E)$ be a 2-edge-connected undirected graph and T be a DFS-tree of G rooted at $r \in V$. Any leaf is not an articulation point of G.*

PROOF: Since T is a spanning tree of G, every path in T also exists in G. For any leaf $v \in V$ of T, $T - v$ is connected. Thus, $G - v$ is also connected, which means that v is not an articulation point of G. $\square$

Lemma 3 *Let $G = (V, E)$ be a 2-edge-connected undirected graph with $|V| \geq 3$ and, T be a DFS-tree of G rooted at $r \in V$. For each leaf v, let $lbe(v) = (v, u^*)$. Then, $\{v\}$ is a MinRS of G if and only if, for any tree edge (x, y) on the $u^*, \pi(v)$-path in T, there exists a back edge outgoing from a vertex in $T_y - v$ that covers (x, y).*

PROOF: ($\Longrightarrow$) We show by contraposition. Suppose that there exists a tree edge (x, y) on the $u^*, \pi(v)$-path in T that is not covered by any back edge outgoing from a vertex other than v. Since any vertex in $T_y - v$ cannot have a back edge to a proper ancestor of x in $G - v$, which means that xy is a bridge in $G - v$. Thus, $G - v$ is not 2-edge-connected.

($\Longleftarrow$) We suffice to show that $G - v$ is 2-edge-connected. By Lemma 2, $G - v$ is connected. We show that there is no bridge in $G - v$. Every back edge (x, y) with $x, y \neq v$ is not a bridge in $G - v$, since xy and the y, x-path in T form a cycle in $G - v$. Next, we consider tree edges in $G - v$. For any tree edge not on yhe $u^*, \pi(v)$-path in T, the covering back edge still exists in $G - v$, which means that such a tree edge is not a bridge in $G - v$. For any tree edge (x, y) on the $u^*, \pi(v)$-path in T, by the assumption, there exists a back edge outgoing from a vertex in $T_y - v$ that covers (x, y), which means that such a tree edge is not a bridge in $G - v$. Thus, there is no bridge in $G - v$, and hence, $G - v$ is 2-edge-connected. $\square$

As in Algorithm 3, for any leaf v of T, we can store the set of tree edges e in T_v such that $\mathit{CovSrcMult}(e) = 1$ and e is covered by back edges outgoing from v.

Lemma 4 *Given a 2-edge-connected undirected graph $G = (V, E)$ with $n = |V|$ and $m = |E|$, its DFS-tree T rooted at $r \in V$, and the cover-source multiplicities CovSrcMult for all tree edges in T, Algorithm 3 computes, for each leaf v of T, the set BE_v of tree edges e in T_v such that $\mathit{CovSrcMult}(e) = 1$ and e is covered by back edges outgoing from v in $O(n)$ time.*

PROOF: TODO $\square$

Theorem 2 *Given a 2-edge-connected undirected graph $G = (V, E)$ with $n = |V|$ and $m = |E|$, we can compute all MinRSs consisting of only leaves of a DFS-tree in $O(n + m)$ time.*

PROOF: As the given graph is 2-edge-connected, each tree edge is covered by at least one back edge. By Lemma 3, $\{v\}$ is a MinRS of G if and only if for any tree edge is covered by back edges outgoing from vertices other than v, which means that $\mathit{CovSrcMult}(e) \geq 2$.

Algorithm 3 Computation of the Set of Tree Edges with Cover-Source Multiplicity 1 for Each Leaf

Input: A 2-edge-connected undirected graph $G = (V, E)$, its DFS-tree T rooted at $r \in V$, and the cover-source multiplicities *CovSrcMult* for all tree edges in T, and a tree edge (u, v) of G

Output: For each leaf v of T, the set BE_v of tree edges with cover-source multiplicity 1 that are covered by back edges outgoing from v; and leaf l to be considered (possibly NIL)

procedure TREEEDGETOBACKEDGE($G, T, \mathit{CovSrcMult}, (u, v)$)
  $l :=$ NIL;
  if v is a leaf **then**
    $l := v$
  end if;
  for each child w of v in T **do**
    Execute TREEEDGETOBACKEDGE($G, T, \mathit{CovSrcMult}, (v, w)$);
    Let l' be the returned leaf;
    if $l' \neq$ NIL and [$l =$ NIL or $\mathit{dfsId}(\mathit{lbeTarget}(l')) < \mathit{dfsId}(\mathit{lbeTarget}(l))$] **then**
      $l := l'$;
    end if
  end for
  $BE_l := BE_l \cup \{uv\}$ **if** $\mathit{CovSrcMult}(uv) = 1$; ▷ l is not NIL here if this line is executed
end procedure

Thus, we can check whether $\{v\}$ is a MinRS of G by checking whether BE_v is empty or not. By Theorem 1, we can compute the cover-source multiplicities for all tree edges in $O(n + m)$ time. By Lemma 4, we can compute the sets BE_v for all leaves v of T in $O(n)$ time. Thus, we can compute all MinRSs consisting of only leaves of T in $O(n + m)$ time. □

4.2 Maximal 2-deg Path MiRSs

TODO

4.3 Single Vertex Degree at Least 3 MinRSs

TODO

4.4 Reuse of DFS-tree and Covering Information

TODO