数理解析研究所講究録
1120 巻 1999 年 161-173

指定された全域木を含む最小 2 辺連結部分グラフを求める近似アルゴリズム

Approximating a Smallest 2-Edge-Connected Subgraph Containing a Specified Spanning Tree

永持 仁
Hiroshi NAGAMOCHI

京都大学大学院情報学研究科
〒 606-8501 京都市左京区吉田本町
naga@kuamp.kyoto-u.ac.jp

Abstract: Given a graph $G=(V,E)$ and a tree $T=(V,F)$ with $E\cap F=\emptyset$ such that $G+T=(V,F\cup E)$ is 2-edge-connected, we consider the problem of finding a smallest 2-edge-connected spanning subgraph $(V,F\cup E')$ of $G+T$ containing T. The problem, which is known to be NP-hard, admits a 2-approximation algorithm. However, obtaining a factor better than 2 for this problem has been one of the main open problems in the graph augmentation problem. In this paper, we show that the problem is $(1.92+\epsilon)$-approximable in $O(n^{1/2}m+n^2)$ time for any constant $\epsilon>0$, where $n=|V|$ and $m=|E\cup F|$.

Key words: approximation algorithm, edge-connectivity, spanning tree, spanning subgraph, graph augmentation

1 Introduction

Given a 2-edge-connected undirected multigraph $H=(V,E)$ with n vertices and m edges and a spanning subgraph $H_0=(V,E_0)$, we consider the problem of finding a smallest 2-edge-connected spanning subgraph $H_1=(V,E_1)$ that contains H_0. Note that the problem can be regarded as a graph augmentation problem of finding a smallest subset $E'\subseteq E-E_0$ of edges to augment H_0 to a 2-edge-connected graph $H_1=(V,E_1=E_0\cup E')$. The problem is shown to be NP-hard [3] even if $E_0=\emptyset$. In the case of $E_0=\emptyset$, the problem, which is called *the minimum 2-edge-connected spanning subgraph problem* (2-ECSS), has been extensively studied and several approximation algorithms are known [1, 2, 7]. The currently best approximation ratio for 2-ECSS is $\frac{17}{12}$ due to Cheriyan *et al.* [1]. On the other hand, if H_0 is connected, H_0 can be assumed to be a spanning tree of H without loss of generality (since every 2-edge-connected component in H_0 can be contracted into a single vertex without losing the property of the problem). Let us call the problem with a tree H_0 *the minimum 2-edge-connected subgraph problem containing a spanning tree* (2-ECST), which is shown to be NP-hard by Frederickson and J. JáJá [4] (even if the height of a spanning tree H_0 is 2 and every edge in $E-E_0$ connects two leaf vertices of H_0). The 2-ECST has an application to the problem of realizing rectangular dual graphs in floor-planning [10]. In the special case of H being a complete graph, 2-ECST is the problem of augmenting a tree H_0 to a 2-edge-connected graph by adding a minimum number of new edges, for which Eswaran and Tarjan [3] presented a linear time algorithm (which creates no multiple edges). If H is a general graph, we are permitted to add to H_0 only edges from $E-E_0$. For general 2-ECST, there is a 2-approximation algorithm [4, 6], which relays on the minimum branching algorithm. In this paper, we present a $(1.92+\epsilon)$-approximation algorithm for 2-ECST, where $\epsilon > 0$ is an arbitrary constant. Our algorithm is based on the maximum matching algorithm and a certain decomposition of a tree. Its running time is $O(n^{1/2}m + n^2)$, where $n = |V|$ and $m = |E|$.

2 Definitions

A singleton set $\{x\}$ may be simply written as x, and " $\subset$ " implies proper inclusion while " $\subseteq$ " means " $\subset$ " or " $=$ ". For an undirected graph $H = (V, E)$ and an edge set E', we denote by $H + E'$ (resp., $H - E'$) the graph obtained from H by adding (resp., removing) edges in E'. The vertex set (resp., edge set) of a graph H may be denoted by $V(H)$ (resp., $E(H)$). For a subset $X \subseteq V$, let $\overline{X}$ denote $V - X$, and $H - X$ means the graph obtained from H by removing the vertices in X together with the incident edges. A maximal 2-edge-connected subgraph $H[X]$ of H induced by a subset $X \subseteq V$ is called a *2-edge-connected component.*

Let $G = (V, E)$ be an undirected graph, and $T = (V, F)$ be a tree on the same vertex set V, where $E \cap F = \emptyset$ is assumed, but there possibly exits a pair of edges $e \in E$ and $f \in F$ such that e and f have the same end vertices. For a subset $E' \subseteq E$, $V(E')$ denotes the set of end vertices of edges in E'. For a subset $X \subset V$, $E_G(X)$ denotes the set of edges in E connecting a vertex in X and a vertex in $V - X$. In particular, $E_G(u)$ is the set of edges in E which are incident to a vertex $u \in V$. For two vertices $u, v \in V$, let $P_T(u, v)$ denote the path connecting u and v in T. We say that an edge $e = (u, v) \in E$ *covers* an edge $f \in F$ if $P_T(u, v)$ contains f, and that an edge set $E' \subseteq E$ *covers* an edge set $F' \subseteq F$ if each edge in F' is covered by an edge in E'. Clearly, $T + E'$ is 2-edge-connected for a subset $E' \subseteq E$ if and only if E' covers F.

We choose an arbitrary vertex $r \in V$ as the root of T, which defines a parent-child relation among vertices in V on T. The parent of a non-root vertex u is denoted by $p(u)$. For a vertex $u \in V$, let $Ch(u)$ denote the set of children of u, and $D(u)$ denote the set of all descendents of u (including u). For two vertices $u, v \in V$, we say that u is *lower* than v (or v is *higher* than u) if $u \in D(v) - v$. We write $v \prec u$ (resp., $v \preceq u$) if $u \in D(v) - v$ (resp., $u \in D(v)$). For two vertices u and v with $u \in D(v)$ or $v \in D(u)$, $\min(u, v)$ (resp., $\max(u, v)$) denotes the higher (resp., lower) vertex in $\{u, v\}$ if $u \neq v$ (or any of u and v if $u = v$). For an edge $e = (u, v) \in E$, we denote by $lca(e)$ the least (lowest) common ancestor of end vertices u and v in the rooted tree T. For a vertex set $X \subseteq V$, $High(X)$ is defined to be the subset of $E_G(X)$ such that, for any $e \in E_G(X) - High(X)$, there is an $e' \in High(X)$ with $lca(e') \prec lca(e)$ and for any two $e_1, e_2 \in High(X)$, neither $lca(e_1) \prec lca(e_2)$ nor $lca(e_2) \prec lca(e_1)$ (thus $High(X)$ contains those edges e with the highest $lca(e)$).

The subgraph $T[D(u)]$ of T induced by $D(u)$ is called the subtree at u (which is connected). A vertex u is called a *leaf vertex* if u has no child, and is called a *fringe vertex* if all the children of u are leaf vertices. For a vertex $u \in V$, let $LEAF(u)$ (resp., $FRINGE(u)$) denote the set of all leaf vertices (resp., fringe vertices) in the subtree $T[D(u)]$. An edge $f = (u, v) \in F$ with $u \prec v$ is called a *leaf edge* (resp., *fringe edge*) of v if v is a leaf vertex (resp., a fringe vertex). The subtree $T[D(u)]$ at a vertex u is called a *leaf tree* if u is a fringe vertex.

We call a subtree $T[D(v)]$ *l-closed* in G if G has no edge between $LEAF(v)$ and $\overline{D(v)}$. Clearly, $T = T[D(r)]$ is l-closed.

3 Decomposing the problem

In this section, we describe how a given instance $(T = (V, F), G = (V, E))$ of the 2-ECST problem can be decomposed into smaller problem instances. For a subset $F' \subseteq F$, we define

- $\beta(F')$ as the size of the smallest set $E' \subseteq E$ that covers F' (where E' does not necessarily cover edges in $F - F'$),

- $E\langle F'\rangle$ as the set of all edges in E that cover at least one edge in F',
- $\widetilde{F'}$ as the set of all edges in F covered by $E\langle F'\rangle$ (where trivially $F' \subseteq \widetilde{F'}$).

(For example, if we consider the set F_{leaf} of all leaf edges in an l-closed subtree $T[D(v)]$, then any edge $e = (u, u') \in E\langle F_{leaf}\rangle$ satisfies $\{u, u'\} \subseteq D(v)$, and hence $\widetilde{F_{leaf}}$ is contained in $T[D(v)]$.)

Assume that there are subsets $F_1, F_2, \ldots, F_k \subseteq F$ such that

$$E\langle F_i\rangle \cap E\langle F_j\rangle = \emptyset,\ 1 \le i < j \le k$$

(hence $F_i \cap F_j = \emptyset$). Since there is no edge $e \in E$ that can cover two edges from distinct F_i and F_j, it holds

$$\beta(F) \ge \beta(F_1) + \beta(F_2) + \cdots + \beta(F_k).$$

Suppose that we are able to compute an edge set $E_i^{apx} \subseteq E$ that covers F_i and satisfies $|E_i^{apx}| \le c\beta(F_i)$ for some constant c. Then $E^{apx} = E_1^{apx} \cup \cdots \cup E_k^{apx}$ becomes a c-approximation solution to the original problem (T, G), provided that E^{apx} covers the entire F.

Let us consider a procedure for finding such F_i and E_i^{apx}. With initial setting $F' := F$, $E' := E$ and $i := 1$, we repeat the following procedure until all edges in F are covered.

> Choose a subset $F_i \subset F'$, and compute a subset $E_i^{apx} \subseteq E'$ that covers $\widetilde{F_i}$ and satisfies $|E_i^{apx}| \le c\beta(F_i)$. Let F_i'' ($\supseteq \widetilde{F_i}$) denote the set of all edges covered by E_i^{apx}.
>
> Let $F' := F' - F_i''$; $E' := E' - E_i^{apx}$; $i := i+1$. (To remove F_i'' from F' effectively, we contract all vertices in $V(F_i'')$ into a single vertex if the graph $(V(F_i''), F_i'')$ is connected.) □

Importantly, $\widetilde{F_i} \subseteq F_i''$ implies $E\langle F_i\rangle \cap E\langle F_{i+1}\rangle = \emptyset$ for any choice of F_{i+1} in the $(i+1)$-th iteration. If F' becomes empty after the i^*-th iteration, $E_1^{apx} \cup \cdots \cup E_{i^*}^{apx}$ covers F and is a c-approximation solution.

4 Lower bounds

Let F_{leaf} and F_{fringe} be respectively the sets of leaf edges and fringe edges in $T[D(v)]$. In this section, we introduce some lower bounds on $\beta(F_{leaf})$ and $\beta(F_{leaf} \cup F_{fringe})$.

LEMMA **4.1 (lower bound)** *Let $G = (V, E)$ be a graph and $T = (V, F)$ be a tree rooted at r with $E \cap F = \emptyset$. For a non-leaf vertex v in T, let F_{leaf} be the set of all leaf edges in the subtree $T[D(v)]$, and let E_{leaf} be the set of all edges $e = (u, u') \in E$ with $u, u' \in LEAF(v)$. Then*

$$\beta(F_{leaf}) \ge |LEAF(v)| - |M^*|,$$

where $M^* \subseteq E$ is a maximum matching in the graph $(LEAF(v), E_{leaf})$.

Proof: Omitted. □

Let us derive a stronger lower bound on $\beta(F_{leaf} \cup F_{fringe})$. For this, we introduce prime edges of type-1 and type-2. For a leaf tree $T[D(u)]$ with exactly two leaf vertices $\{w, w'\} = Ch(u)$, we call an edge $g = (w, w') \in E$ a *prime edge of type-1*. Let $f = (v'', v') \in F$ ($v'' \prec v'$) be an edge in T such that $FRINGE(v') - v'$ contains exactly one fringe vertex v, and $LEAF(v')$ contains exactly three leaf vertices u_1, u_2 and u_3 (where $\{u_1, u_2\} = Ch(v)$ and $u_3 \in Ch(v')$ are assumed without loss of generality). We call edges (u_3, u_1) and (u_3, u_2) *prime edges of type-2* if

$$\text{for } i = 1, 2,\ \{(u_1, u_2), (u_i, u_3)\} \subseteq E_G(u_i) \text{ and } w \in D(v') - u_3 \text{ for all } (u_i, w) \in E_G(u_i). \quad (1)$$

See Fig. 1 (where (u_1, u_2) is a prime edge of type-1 by definition). In this case, the edge $f = (v'', v') \in F$ is called a *pseudo-fringe edge*, and the vertices in $D(v') - u_3 - D(v)$ are called *pseudo-fringe vertices*. We denote by $PFRINGE(u)$ the set of fringe and pseudo-fringe vertices in $T[D(u)]$.

LEMMA **4.2 (lower bound)** *Let $G = (V, E)$ be a graph and $T = (V, F)$ be a tree rooted at r with*

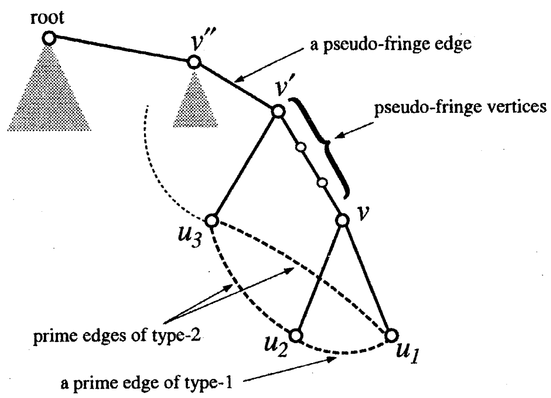

図 1: Definition of prime edges of type-2, and pseudo-fringe edges and vertices.

$E \cap F = \emptyset$. *For a vertex* $v \in V - LEAF(r) - FRINGE(r)$, *let* E_{leaf} *be the set of all edges* $e = (u, u') \in E$ *with* $u, u' \in LEAF(v)$, E_{prime} *be the set of prime edges of type-1 and type-2 in* E_{leaf}, F_{leaf} *be the set of leaf edges in* $T[D(v)]$, *and* F_{fringe} *be the set of fringe or pseudo-fringe edges in* $T[D(v)]$. *Then for* $F_v = F_{leaf} \cup F_{fringe}$,

$$\beta(F_v) \geq \frac{2}{3}|LEAF(v)| - \frac{1}{3}|M^*|,$$

where $M^* \subseteq E$ *is a maximum matching in the graph* $(LEAF(v), E_{leaf} - E_{prime})$.

Proof: Omitted. □

We call a subtree $T[D(v)]$ *lf-closed* if G has no edge between $LEAF(u) \cup PFRINGE(u)$ and $\overline{D(u)}$. Clearly, $T = T[D(r)]$ is lf-closed. A subtree $T[D(v)]$ is called *minimally lf-closed* if $T[D(v)]$ is lf-closed and there is no proper subtree $T[D(u)]$ of $T[D(v)]$ which is lf-closed.

5 Some reducible cases

In this section, we show four cases where we can reduce the size of a given instance (T, G) without loss of generality.

Case-1. There is an l-closed leaf tree $T[D(v)]$: Now $\widetilde{F_{leaf}} = F_{leaf}$. In this case, a smallest set $E_v^{opt} \subseteq E$ that covers the set F_{leaf} of all leaf edges in $T[D(v)]$ can be found by the next procedure (P1).

(P1) Compute a maximum matching M^* in the graph $(Ch(v), E_{leaf})$, and choose an arbitrary edge $e_w \in E_G(w)$ for each unmatched vertex $w \in Ch(v) - V(M^*)$ (where $E_G(w) \neq \emptyset$ by the 2-edge-connectivity of $T + E$). Retain $E_v^{opt} = M^* \cup \{e_w \mid w \in Ch(v) - V(M^*)\}$ as part of the solution to cover the current T. Contract all vertices in $Ch(v) \cup \{v\}$ into a single vertex v' both in T and G, and delete any resulting self-loops (where the vertex v' becomes a new leaf vertex in the resulting tree).

Obviously, E_v^{opt} covers F_{leaf}, and satisfies $|E_v^{opt}| = |M^*| + |Ch(v)| - 2|M^*| = |LEAF(v)| - |M^*|$. By Lemma 4.1, $|E_v^{opt}| = \beta(F_{leaf})$ is the minimum among all subsets of E that cover F_{leaf}. □

For a fringe vertex v, let $u \in Ch(v)$. Vertex u is called *isolated* if u is not adjacent via edges in $E_G(u)$ to any sibling (i.e., other child) of v. Note that u is isolated if $|Ch(v)| = 1$. Vertex u is called *trivial* if $|E_G(u)| = 1$; we must use the unique edge in $E_G(u)$ to cover the leaf edge $f = (v, u)$. For a nontrivial u, let $E_G(u) = \{e_1 = (u, v_1), e_2 = (u, v_2), \ldots, e_p = (u, v_p)\}$, where $p = |E_G(u)| \geq 2$. An edge $e_i = (u, v_i)$ with $v_i = v$ is called *redundant* if $E_G(u)$ contains some $e_j = (u, v_j)$ with $v_j \neq v$. If all edges in $E_G(u)$ are multiple edges of (v, u), then we choose an arbitrary edge (say e_1) in $E_G(u)$ and call the other edges e_i, $i = 2, \ldots, p$ *redundant*. (Even if G is originally simple, our algorithm will repeat contracting some vertices and may produce multiple edges in the resulting G.) It is not difficult to see that there is an optimal subset $E^{opt} \subseteq E$ that covers F without using any redundant edge.

Case-2. There is a leaf tree $T[D(v)]$ such that $T[D(v)]$ is not l-closed and there is an isolated leaf vertex $u \in Ch(v)$ (this includes the case of $|Ch(v)| = 1$): There is the parent $v' = p(v)$ of v (since v is not the root by the non-l-closedness of $T[D(v)]$). Let I_v denote the set of all isolated vertices in $Ch(v)$.

For each non-trivial leaf vertex $u \in I_v$ (if any), we first remove all redundant edges in $E_G(u)$ from G. For each trivial leaf vertex $u' \in I_v$ such that $E_G(u') = \{(u', v)\}$ (if any), we retain the edge (u', v) as part of the solution to cover the original T and contract u' and v into a vertex both in T and G. Now if there remains an isolated vertex $u'' \in I_v$, then any edge in E covering the leaf edge $f = (v, u'')$ also covers the fringe edge $f' = (v', v)$ of v, because $E_G(u'')$ contains no redundant edge. Thus $\beta(F) = \beta(F - f')$. For this reason, we contract the end vertices of the fringe edge $f' = (v', v)$ into a single vertex both in T and G, and delete any resulting self-loops. The procedure in Case-2 is described as follows.

(P2) For each non-trivial leaf vertex $u \in I_v$, remove all redundant edges in $E_G(u)$ from G. For each trivial leaf vertex $u' \in I_v$ such that $E_G(u') = \{(u', v)\}$, retain the edge $(u', v) \in E_G(u')$ and contract u' and v into v. If there remains an isolated vertex in I_v, then contract $v' = p(v)$ and v into a vertex.

□

Case-3. There is a leaf tree $T[D(v)]$ such that $T[D(v)]$ is not l-closed, $|Ch(v)| = 3$ holds, and $Ch(v)$ contains no isolated vertex: We first remove all redundant edges incident to $u \in Ch(v)$. If there is a trivial vertex $u \in Ch(v)$ (i.e., $|E_G(u)| = 1$), then choose such a vertex u. Now the edge $e \in E_G(u)$ connects u and a sibling $u' \in Ch(v)$ of u (since u is not isolated). To cover the leaf edge $f = (v, u)$, the edge $e = (u, u')$ must be used. Therefore, we retain the edge (u, u') as part of the solution, and contract $\{u, u', v\}$ into a single vertex v both in T and G, deleting any resulting self-loops.

On the other hand, if $|E_G(u)| \geq 2$ holds for all $u \in Ch(v)$, then we claim that the fringe edge $f' = (v', v) \in F$, where $v' = p(v)$, can be contracted without loss of generality. Let $Ch(v) = \{u_1, u_2, u_3\}$. Consider an arbitrary subset $E' \subseteq E$ that covers all edges in T. Suppose that E' contains no edge between $Ch(v)$ and $\overline{D(v)}$. That is, all leaf edges in $T[D(v)]$ are covered by (at least) two edges $e_1 = (u_i, u_j), e_2 = (u_j, u_h) \in E'$. Since $T[D(v)]$ is not l-closed, E contains an edge e_0 between a vertex $u \in Ch(v)$ and $w \in \overline{D(v)}$. If there is such an edge $e_0 = (w, u_i)$ (resp., $e_0 = (w, u_h)$), then we easily see that $\tilde{E} = (E' - e_1) \cup \{e_0\}$ (resp., $\tilde{E} = (E' - e_2) \cup \{e_0\}$) covers all edges in T. If all such edges e_0 are incident to u_j, then by $|E_G(u_i)| \geq 2$, E contains an edge $e_3 = (u_i, u_h)$. In this case, $\tilde{E} = (E' - \{e_1, e_2\}) \cup \{e_0, e_3\}$ covers all edges in T. In any case, we can assume that at least one edge between $Ch(v)$ and $\overline{D(v)}$ is used in E'. For this reason, we contract the end vertices of the fringe edge $f' = (v', v)$ into a single vertex both in T and G, and delete any resulting self-loops. The procedure in Case-3 is summarized as follows.

(P3) Remove all redundant edges incident to $u \in Ch(v)$. If there is a trivial vertex $u \in Ch(v)$, retain the edge $(u, u') \in E_G(u)$ and contract $\{u, u', v\}$ into a single vertex v. Otherwise, contract the fringe edge $f' = (v', v)$.

Given a solution E' to the instance (T', G') resulting from contracting f', we can modify E' (if necessary) so that f' is also covered in the original instance (T, G) without increasing the size of E'.

□

Case-4. There is an edge $f' = (v'', v')$ in T ($v'' \prec v'$) such that $FRINGE(v') - v'$ contains exactly one fringe vertex v (where its leaf tree $T[D(v)]$ is not l-closed and no child in $Ch(v)$ is isolated), $LEAF(v')$ contains exactly three leaf vertices u_1, u_2 and u_3 (where $\{u_1, u_2\} = Ch(v)$ and $u_3 \in Ch(v')$ are assumed without loss of generality), and there is an edge $(u_3, u_2) \in E$, but f' is not a pseudo-fringe edge. See Fig. 2.

Since u_1 is assumed to be a non-isolated vertex, it has edge $(u_1, u_2) \in E_G(u_1)$. We show that if no edge in $E_G(u_1)$ is incident to any vertex $\overline{D(v')} \cup \{u_3\}$, then we can retain (u_1, u_2) as

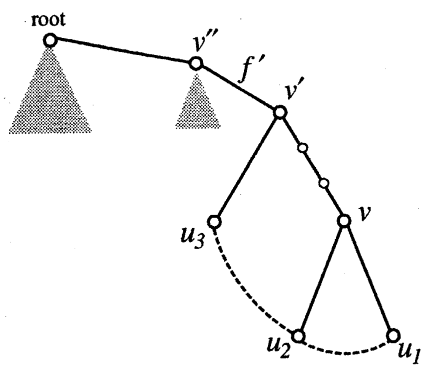

図 2: Illustration for a subtree $T[D(v')]$ in Case-4.

part of the solution to cover T. Let E^* be a smallest edge set $E^* \subseteq E$ covering F, and assume that E^* contains an edge $(u_1, w) \in E$ with $w \in D(v') - u_3$, but do not contain (u_1, u_2). To cover the leaf edge $(v, u_2) \in F$, E^* has some edge $e' = (u_2, w') \in E_G(u_2) - (u_1, u_2)$. It is clear that $(E^* - (u_1, w)) \cup \{(u_1, u_2)\}$ (resp., $(E^* - \{(u_1, w), e'\}) \cup \{(u_1, u_2), (u_2, u_3)\}$) still covers F if $w' \notin D(v') - v'$ (resp., if $w' \in D(v') - v'$). Thus, removal edges in $E_G(u_1) - (u_1, u_2)$ from E never increases $\beta(F)$, and we can contract $D(v)$ into a single vertex after retaining (u_1, u_2) as part of the solution to cover T.

Assume that $E_G(u_1)$ contains an edge (u_1, w) such that $w = u_3$ or $w \in \overline{D(v')}$. For the edge $f' = (v'', v')$, we next claim that $\beta(F) = \beta(F - f')$ holds if there is an edge $(u_1, w) \in E_G(u_1)$ with $w \in \overline{D(v')}$. To see this, consider the instance (T', G') obtained from the current (T, G) by contracting v'' and v' into a single vertex, and let $E^{**} \subseteq E$ be a smallest edge set covering the edges in T' (i.e., $F - f'$). Assume that E^{**} does not cover f' in T (otherwise we are done). Thus, the edges in $T[D(v')]$ are covered by two edges (say e_1, e_2) in E^{**} by the minimality of $|E^{**}|$. For the edge $e_3 = (u_3, u_2)$ and an edge $e_4 = (u_1, w) \in E_G(u_1)$ with $w \in \overline{D(v')}$, we see that $(E^{**} - \{e_1, e_2\}) \cup \{e_3, e_4\}$ covers all edges in T. Therefore, $\beta(F) = \beta(F - f')$ and we contract the end vertices of edge $f' = (v'', v')$ into a single vertex both in T and G, deleting any resulting self-loops.

The remaining case is that $(u_1, u_3) \in E_G(u_1)$. Since f' is not a pseudo-fringe edge (i.e., (1) does not hold), there is an edge $(u_2, w') \in E_G(u_2)$ with $w' \in \overline{D(v')}$ and in this case we can also contract f' by applying the above argument exchanging the roles of u_1 and u_2. The procedure in Case-4 is summarized as follows.

(P4) If no edge in $E_G(u_1)$ is incident to any vertex $\overline{D(v')} \cup \{u_3\}$, then retain (u_1, u_2) as part of the solution to cover T and contract $D(v)$ into a single vertex. Otherwise contract edge $f' = (v'', v')$.

Given a solution E^{**} to the instance (T', G') resulting from contracting f', we can modify E^{**} (if necessary) so that f' is also covered in the original instance (T, G) without increasing the size of E^{**}.

□

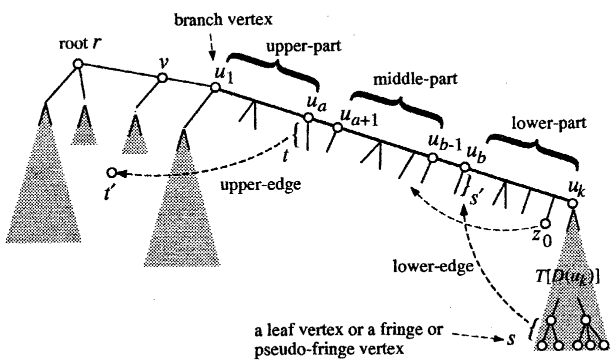

図 3: Definition of lower-, middle- and upper-parts of a chain $P_T(u_1, u_k)$.

6 Structure of $T + E$

A leaf vertex is called a *thorn vertex* if its parent is not a fringe vertex, and a vertex u is called a *branch vertex* if $u = r$ or $Ch(u)$ contains at least two non-leaf vertices. Let $THORN(v)$ denote the set of all thorn vertices in $T[D(v)]$. Note that the number of branch vertices is at most $|FRINGE(v)|$. For each branch vertex u, a path $P_T(u, u')$ with $u' \in D(u)$ is called a *chain* of u if u' is a fringe or branch vertex and $P_T(u, u') - \{u, u'\}$ contains no fringe or branch vertex. (Thus any internal vertex u'' in a chain has exactly one non-leaf vertex in $Ch(u'')$.) The number of chains in a tree $T[D(v)]$ is at most $2|FRINGE(v)| - 1$.

In what follows, we assume that $T+E$ is 2-edge-connected and $T[D(v)]$ is a minimally lf-closed subtree of T. In this case, v is the root of $T[D(v)]$ and is treated as a branch vertex. Consider a chain $P_T(u_1, u_k)$ of $T[D(v)]$, where $u_1 \prec \cdots \prec u_k$ for $V(P_T(u_1, u_k)) = \{u_1, \ldots, u_k\}$ (see Fig. 3). Let u_a be the lowest vertex in $\{u_1, \ldots, u_k\}$ such that all the edges in $P_T(u_1, u_a)$ are covered by a single edge $(t, t') \in E$ (where $a \geq 2$ since such (t, t') exists by the 2-edge-connectivity of $T+E$), and call the subpath $P_T(u_1, u_a)$ the *upper-part* of chain $P_T(u_1, u_k)$. The edge $(t, t') \in E$ that defines u_a is called the *upper-edge* of the chain, where $lca((t, t')) \preceq u_1 \preceq t$ holds and t may belong to $D(u_k)$. Similarly the highest vertex $u_b \in \{u_1, \ldots, u_k\}$ such that the edges in $P_T(u_b, u_k)$ are covered by a single edge $(s, s') \in E$ with $s \in LEAF(u_k) \cup PFRINGE(u_k) - u_k$ (where $u_b = u_k$ if no such (s, s') exists), and call the subpath $P_T(u_b, u_k)$ the *lower-part* of chain $P_T(u_1, u_k)$. If $u_b \neq u_k$, the edge $(s, s') \in E$ that covers the lower-part is called the *lower-edge* of chain $P_T(u_1, u_k)$, where s' possibly belongs to $\overline{D(u_1)}$. If $u_1 \prec u_b$, then there must be a thorn vertex $z_0 \in D(u_b) - (D(u_k) \cup \{u_b\})$ such that an edge $e \in E$ connects z_0 and a vertex in $\overline{D(u_b)}$ (otherwise $T[D(u_b)]$ would be lf-closed). We say that a subpath $P_T(u_i, u_j)$ *has* a thorn vertex w if the parent $p(w)$ is contained in $P_T(u_i, u_j)$.

Consider an edge $g = (x_1, x_2) \in E$ such that both parents $p(x_1)$ and $p(x_2)$ belong to the same chain $P_T(u_1, u_k)$; $p(x_1) \preceq p(x_2)$ is assumed without loss of generality. In this case, we denote the parents $p(x_1)$ and $p(x_2)$ by $up(g)$ and $dwn(g)$, respectively. Such edge g is called a *swing edge* if path $P_T(p(x_1), p(x_2))$ has no thorn vertex other than x_1 and x_2 (some other edge $e \in E$ may be incident to x_1, x_2 or $P_T(p(x_1), p(x_2))$). See Fig. 4, where g_1, g_2, g_3 are not swing edges.

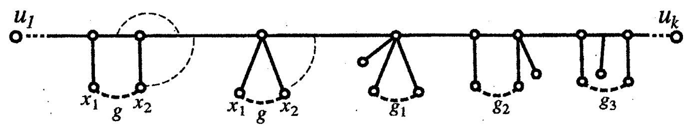

図 4: Definition of swing edges g.

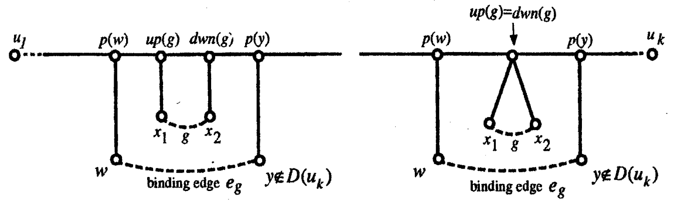

図 5: Definition of binding edges e_g of a swing edge g in the case of (B1).

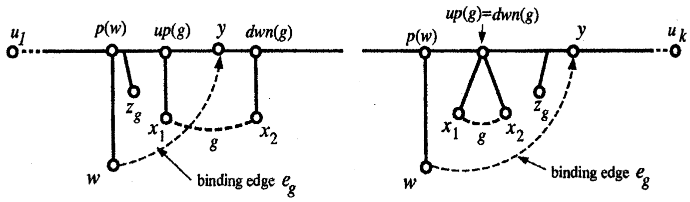

図 6: Definition of binding edges e_g of a swing edge g in the case of (B2).

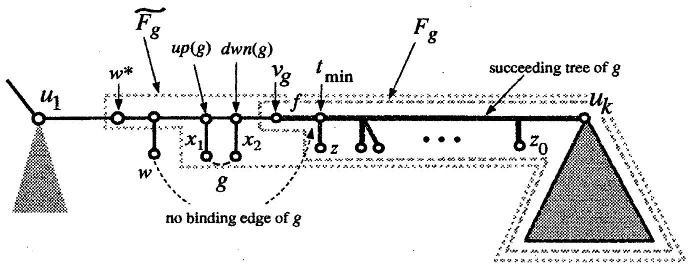

図 7: Definition of a succeeding tree of a solo edge g.

If $u_{a+1} \preceq u_{b-1}$, then the subpath $P_T(u_{a+1}, u_{b-1})$ is called the *middle-part* of chain $P_T(u_1, u_k)$. In this case, for a swing edge $g = (x_1, x_2) \in E$ with $u_{a+1} \preceq up(g) \preceq dwn(g) \preceq u_{b-1}$, we call an edge $e_g = (w, y) \in E$ a *binding edge* of g if e_g satisfies one of the following (B1) and (B2).

(B1) $w, y \in THORN(v)$ and $up(e_g) \prec up(g) \preceq dwn(g) \prec dwn(e_g)$ (see Fig. 5).

(B2) $\{w\} = \{w, y\} \cap THORN(v)$, $p(w) \prec up(g) \preceq y$ and path $P_T(p(w), \max(dwn(g), y))$ has a thorn vertex $z_g \in THORN(v) - \{x_1, x_2, w\} - D(u_k)$, where possibly $y \in D(u_k)$ (see Fig. 6).

Notice that for any binding edge $e_g = (w, y)$, it must hold $p(w) \in D(u_1) - u_1$ in cases (B1) and (B2) and $y \notin D(u_k)$ in case (B1) by the choice of u_a and u_b.

A swing edge $g \in E$ is called a *solo edge* if

(B3) $u_{a+1} \preceq up(g) \preceq dwn(g) \preceq u_{b-1}$, and g has no binding edge in (B1) or (B2).

For a solo edge $g = (x_1, x_2) \in E$ defined on the chain $P_T(u_1, u_k)$, we define the succeeding tree of g as follows. Let t_{min} be the highest vertex in $P_T(dwn(g), u_k) - \{dwn(g), u_k\}$ such that there is a thorn vertex $z \in THORN(v) - \{x_1, x_2\}$ incident to t_{min}. (By definition of u_b and the minimal lf-closedness of $T[D(v)]$, there exists such thorn vertex z_0.) Let $f = (v_g, t_{min}) \in F$ be the edge with $v_g \prec t_{min}$ (possibly $v_g = dwn(g)$). We call this vertex v_g the *succeeding vertex* of g, and call the subtree $T[D(v_g)]$ the *succeeding tree* of g. See Fig. 7.

7 Covering minimally lf-closed subtrees

Let $T[D(v)]$ be a minimally lf-closed subtree in T (where we can assume that v is not a fringe vertex in T by Case-1). Such a $T[D(u)]$ always exits, since $T = T[D(r)]$ is lf-closed. In this section, we consider how to choose edges from E to cover *all* edges in the $T[D(v)]$.

7.1 Outline

Assume that none of Cases-1,2,3 and 4 holds in $T[D(v)]$. Thus $T[D(v)]$ satisfies that

(A1) Every fringe vertex u satisfies $|Ch(u)| \neq 1, 3$, and each non-root and non-fringe vertex $v' \in D(v)$ with $|LEAF(v')| = 3$ satisfies (1),

(A2) Every fringe vertex u has an edge $e = (w, w') \in E$ such that $\{w, w'\} \subseteq Ch(u)$.

In this section, we assume that the following condition holds in a given minimally lf-closed tree $T[D(v)]$.

(A3) For any solo edges $g \in E$ on the middle-part of a chain in $T[D(v)]$, its succeeding tree $T[D(v_g)]$ has at most five leaf vertices (i.e., $|LEAF(v_g)| \leq 5$).

(We discuss in section 8 the case in which condition (A3) does not hold.) If there are three disjoint solo edges g, g', g'' on a path from v and to a leaf vertex w in $T[D(v)]$. For the highest edge g among these three edges, it is easy to see that $|LEAV(v_g)| \geq 6$ for its succeeding vertex v_g. Thus, if (A3) holds, then there are at most two disjoint solo edges in the path from v to any fringe vertex in $T[D(v)]$.

We sketch a procedure COVER for computing a subset $E^{apx} \subseteq E$ that covers all edges in a minimally lf-closed subtree $T[D(v)]$. Let F_{leaf} be the set of leaf edges in $T[D(v)]$, and E_{leaf} denote the set of all edges $e = (u, u') \in E$ with $u, u' \in LEAF(v)$, and $E_{prime} \subseteq E_{leaf}$ be the set of prime edges. The procedure consists of the following three phases, where the details of Phases-2 and 3 are described in the next subsections.

Procedure COVER

If $|LEAF(v)| \leq 3$, then it is easy to find a subset $E^{apx} \subseteq E$ that covers $T[D(v)]$ and satisfies $\frac{3}{2}\beta(F_{leaf} \cup F_{fringe})$.In what follows, $|LEAF(v)| \geq 4$ is assumed.

Phase-1 (Covering all leaf edges in $T[D(v)]$): Compute a maximum matching $M^* \subseteq E$ in the graph $(LEAF(v), E_{leaf} - E_{prime})$, and denote by W the set of unmatched vertices in $LEAF(v)$. A prime edge $g \in E_{prime}$ is called an *unmatched prime edge* if both end vertices of g are unmatched, and denote by M_1' (resp., M_2') the set of all unmatched prime edges of type-1 (resp., of type-2). For each unmatched prime edge (w, w')

of type-2, where $w \prec w'$, we see by (1) that there is an unmatched prime edge (w, w'') of type-2 such that w'' is the sibling of w' (also $(w', w'') \in M_1'$). For each such pair of unmatched prime edges (w, w') and (w, w''), we choose arbitrarily one of them, and denote by M_2'' the resulting set of unmatched prime edges of type-2 (hence $|M_2''| = |M_2'|/2$).

For each vertex $w \in W - V(M_1' \cup M_2')$ (where $E_G(w) \neq \emptyset$ by the 2-edge-connectivity of $T + E$), choose an edge $e_w \in E_G(w)$ as follows. If w is incident to a binding edge e_g in (B1) or (B2) for a swing edge g with $w \prec up(g)$, then let $e_w = e_g$ (by choosing one arbitrarily if there is more than such binding edge). Otherwise, let $e_w \in High(w)$.

For each $g = (u, u') \in M_1'$, we choose an edge $e^{(g)}$ as follows. If no unmatched prime edge of type-2 is adjacent to g, then let $e^{(g)} \in High(\{u, u', p(u)\})$. Otherwise, if an unmatched prime edge (w, u) of type-2 is adjacent to g, then let $e^{(g)} \in High(D(p(w)))$. Denote $E_1 = M^* \cup M_1' \cup M_2'' \cup \{e^{(g)} \mid g \in M_1'\} \cup \{e_w \mid w \in W - V(M_1' \cup M_2')\}$.

Phase-2 (Merging 2-edge-connected components in $T + E_1$): Consider all nontrivial 2-edge-connected components in $T + E_1$. To reduce the number of those components, we choose an appropriate set $E_2 \subseteq E - E_1$ of edges which combine different components in $T + E_1$.

Phase-3 (Making $T[D(v)]$ 2-edge-connected): For each 2-edge-connected component B in $T + (E_1 \cup E_2)$ containing an edge in M^*, we choose an edge $e^{(B)} \in High(X)$ for $X = V(B) \cap (LEAF(v) \cup FRINGE(v))$. Let E_3 be the set of the edges $e^{(B)}$ chosen for all those components B. (To be precise, Phase-3 of our algorithm may divide some 2-edge-connected component into several components (without separating two end vertices of any edge in $M_1' \cup M_2''$) or may treat some 2-edge-connected components as a single component before computing $e^{(B)}$ for each component B.) Output $E^{apx} = E_1 \cup E_2 \cup E_3$. □

The solution $E^{apx} = E_1 \cup E_2 \cup E_3$ covers all the edges in $T[D(v)]$, as will be shown in the next subsection. We first note that $|E_1| = |LEAF(v)| - |M^*|$ holds, because by $|W| = |LEAF(v)| - 2|M^*|$, we have $|E_1| = |M^*| + |M_1'| + |M_2''| + |\{e^{(g)} \mid g \in M_1'\}| + |\{e_w \mid w \in W - V(M_1' \cup M_2')\}| = |LEAF(v)| - |M^*|$. Hence we have

$$|E^{apx}| \leq |LEAF(v)| - |M^*| + |E_2| + |E_3|.$$

Let us assume that E_2 and E_3 are chosen so that the next two properties hold.

PROPERTY **7.1** $|E_2| + |E_3| \leq |M^*|$. □

PROPERTY **7.2** *For some constant* $\theta \geq 0$, $(2 + \theta)(|E_2| + |E_3|) \leq |LEAF(v)|$. □

For the set F_v of all leaf and fringe edges in $T[D(v)]$, we see by Lemma 4.2 that $\beta(F_v) \geq \frac{1}{3}(2|LEAF(v)| - |M^*|)$. Therefore,

$$\begin{aligned}
\frac{|E^{apx}|}{\beta(F_v)} &\leq \frac{3(|LEAF(v)| - |M^*| + |E_2| + |E_3|)}{2|LEAF(v)| - |M^*|} \\
&\leq \frac{3|LEAF(v)|}{2|LEAF(v)| - (|E_2| + |E_3|)} \\
&\quad \text{(by Property 7.1 and by } |LEAF(v)| \\
&\quad + |E_2| + |E_3| \leq 2|LEAF(v)| \\
&\quad \text{which follows from Property 7.2)} \\
&\leq \frac{3|LEAF(v)|}{2|LEAF(v)| - \frac{1}{2+\theta}|LEAF(v)|} \\
&\quad \text{(by Property 7.2)} \\
&= \frac{6 + 3\theta}{3 + 2\theta} = 2 - \frac{\theta}{3 + 2\theta},
\end{aligned}$$

which is strictly smaller than 2 unless $\theta = 0$. We design Phases-2 and 3 such that Property 7.1 and Property 7.2 with some $\theta > 0$ hold.

7.2 Phases-2 and 3

In this subsection, we describe the details of Phases-2 and 3, and then prove some properties of the obtained sets E_2 and E_3.

Phase-2 (Merging 2-edge-connected components in $T + E_1$):

Step 1. A matching edge $g = (z, z') \in M^*$ with $z, z' \in THORN(v)$ is called *upward* in a chain $P_T(u_1, u_k)$ in $T[D(v)]$ if

(B4) both z and z' are incident to $P_T(u_1, u_k)$, and one of z and z' is incident to $P_T(u_1, u_a) - u_1$, where $P_T(u_1, u_a)$ is the upper-part of $P_T(u_1, u_k)$

(note that g is not upward if $p(z) = u_k$ or $p(z') = u_k$). A chain $P_T(u_1, u_k)$ is called *active* if it has at least one upward matching edge and $P_T(u_1, u_a)$ does not belong to a single 2-edge-connected component in $T + E_1$. A branch vertex u_1 is also called *active* if it has an active chain $P_T(u_1, u_k)$.

For each active chain $P = P_T(u_1, u_k)$ in $T[D(v)]$, we choose its upper-edge $e^{(P)}$. For each active branch vertex v', let $E_{upper}(v')$ be the set of the upper-edges $e^{(P)}$ chosen for all active chains $P = P_T(v' = u_1, u_k)$ of v'. Let E_{upper} denote the union of $E_{upper}(v')$ for all active branch vertices v'.

Step 2. Consider the graph $T + (E_1 \cup E_{upper})$. A 2-edge-connected component A in this graph is called *small* if it contains a matching edge $g = (x_1, x_2) \in M^*$, but has no leaf vertex other than x_1 and x_2.

By (A1) and (A2) and $M^* \cap E_{prime} = \emptyset$, the two leaf vertices x_1 and x_2 in a small component A are both thorn vertices. From definitions (B1)-(B3), the matching edge $g = (x_1, x_2)$ in a small component A satisfies one of the following cases.

(a) g is not a swing edge, i.e., the path $P_T(p(x_1), p(x_2))$ between the parents $p(x_1)$ and $p(x_2)$ contains a branch vertex. (In the following (b) and (c), g is assumed to be a swing edge.)
(b) One of the parents $p(x_1)$ and $p(x_2)$ belongs to the lower-part of a chains.
(c) Both $p(x_1)$ and $p(x_2)$ belong to the middle-part of the same chain, where

(1) g has a binding edge $e_g = (w, y) \in E$ satisfying one of (B1) and (B2), or

(2) g is a solo edge.

In the case (c)-(1), we choose a binding edge e_g for each g (even if there is more than one binding edge). Initially set $E_{merge} := \emptyset$, and let $\mathcal{A} = \{A_1, \ldots, A_h\}$ be the set of all small components satisfying (c)-(1). The binding edge e_g of the swing edge g in an $A_i \in \mathcal{A}$ is called *merging* if

(B5) adding e_g to the current graph $T + (E_1 \cup E_{upper} \cup E_{merge})$ merges at least three 2-edge-connected components, each of which contains at least one matching edge, into a single 2-edge-connected component.

We repeatedly apply the following procedure until no new merging edge is found.

> **MERGE** Find an $A_i \in \mathcal{A}$ such that the binding edge e_{g_i} of the matching edge g_i in A_i is merging in the current graph $T + (E_1 \cup E_{upper} \cup E_{merge})$. Add e_{g_i} to E_{merge}, letting $\mathcal{A} := \mathcal{A} - A_i$. □

Let $E'_2 := E_{upper} \cup E_{merge}$.

Phase-3 (Making $T[D(v)]$ 2-edge-connected): Consider all 2-edge-connected components C in $T + (E_1 \cup E'_2)$ containing an edge in M^*, and apply the following steps after letting $E_3 := \emptyset$.

Step 3. Consider the 2-edge-connected components C in $T + (E_1 \cup E'_2)$ such that

$$E(C) \cap M^* \neq \emptyset \text{ and } |E(C) \cap M^*| = |E(C) \cap E'_2|.$$

By procedure MERGE, this can occur only when $E(C) \cap E'_2 \subseteq E_{upper}$ holds and the edges in $E(C) \cap M^*$ are all upward, where each $e \in E(C) \cap M^*$ corresponds to an upper-edge $e' \in E(C) \cap E_{upper}$ by Step 1.

For each of such components C, we partition $E(C) \cap M^*$ into subsets $M^*_{up}(v_1), \ldots, M^*_{up}(v_q)$ such that the upward matching edges in each $M^*_{up}(v_i)$ are defined on some chains $P_T(v_i, u)$ of the same branch vertex v_i. For each v_i, let C_{v_i} denote the graph which consists of upward edges in

$M^*_{up}(v_i) \cup E_{upper}(v_i)$ (for notational convenience), and choose an edge $e^{(B)} \in High(V(M^*_{up}(v_i)))$ for component $B = C_{v_i}$. In this case, $e^{(B)}$ is adjacent to an upward matching edge $g' \in M^*_{up}(v_i)$. Let $e^{(P)} \in E_{upper}(v_i)$ be the upper-edge of $P = P_T(v_i, u')$ that has the matching edge g'. See Fig. 8. If $lca(e^{(B)}) \prec v_i$, then we replace $e^{(P)}$ by $e^{(B)}$:

$$E_3 := E_3 \cup \{e^{(B)}\}, \quad E_{upper}(v_i) := E_{upper}(v_i) - \{e^{(P)}\},$$

updating $C_{v_i} := C_{v_i} + e^{(B)} - e^{(P)}$. (Otherwise (i.e., if $v_i \preceq lca(e^{(B)})$) we do nothing by setting $e^{(B)}$ to be empty.) Let $\mathcal{B}^{S3}$ denote the set of all these components $B = C_{v_i}$ computed in this step.

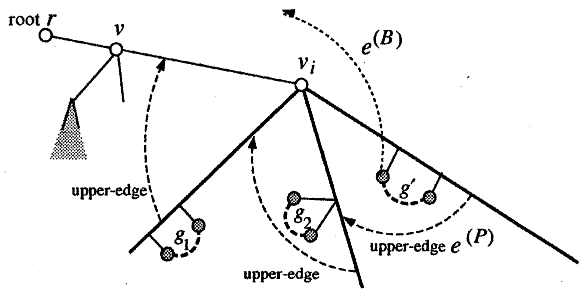

図 8: Illustration for Step 3.

Step 4. Consider the remaining 2-edge-connected components in $T + (E_1 \cup E_2')$ not considered in Step 3. For each component B that contains an edge in M^*, but is not a small component satisfying (b), let $E_3 := E_3 \cup \{e^{(B)}\}$ by choosing an edge $e^{(B)} \in High(X)$ for $X = V(B) \cap (LEAF(v) \cup FRINGE(v))$. Let $\mathcal{B}^{S4}$ be the set of all these components B computed in this step.

Step 5. Finally, we partition the set of all small components satisfying (b) into subsets $\mathcal{A}(P_1), \ldots, \mathcal{A}(P_q)$ such that all components in each $\mathcal{A}(P_i)$ are defined on the same chain P_i. We treat each $\mathcal{A}(P_i)$ as a single component B. For each component $B = \mathcal{A}(P_i)$, let $E_3 := E_3 \cup \{e^{(B)}\}$ by choosing an edge $e^{(B)} \in High(V(\mathcal{A}(P_i)) \cap (LEAF(v) \cup FRINGE(v)))$, where $V(\mathcal{A}(P_i))$ denote the set of all vertices in the small components in $\mathcal{A}(P_i)$. Let $\mathcal{B}^{S5}$ be the set of all these components $B = \mathcal{A}(P_i)$ computed in this step.

For the final E_{upper} and E_3 computed in the above procedure, we denote $E_2 = E_{upper} \cup E_{merge}$ and $E^{apx} = E_1 \cup E_2 \cup E_3$. □

We now show that the obtained E^{apx} covers all edges in $T[D(v)]$ (for this we do not need condition (A3)).

LEMMA **7.1** *Let $G = (V, E)$ be a graph and $T = (V, F)$ be a tree rooted at r with $E \cap F = \emptyset$ such that $T + E = (V, F \cup E)$ is 2-edge-connected. Let $T[D(v)]$ be a minimally lf-closed subtree satisfying conditions (A1) and (A2). Then the subset $E^{apx} = E_1 \cup E_2 \cup E_3 \subseteq E$ obtained by the procedure COVER covers all edges in $T[D(v)]$.*

Proof: Omitted. □

Now we estimate the size of E^{apx}. For each $B \in \mathcal{B}^{S3} \cup \mathcal{B}^{S4} \cup \mathcal{B}^{S5}$, we first show

$$|E(B) \cap M^*| \geq |E(B) \cap E_2| + |E(B) \cap E_3|. \quad (2)$$

By construction of Phase-2, any edges in E_2' are chosen so that if the resulting 2-edge-connected component C has p edges from E_2', then C has at least $p+1$ matching edges, except for the case in Step 3. In Step 3, each component C with $|E(B) \cap M^*| = |E(B) \cap E_2'|$ is divided into several components $B = C_{v_i}$, for which an upper-edge $E(B) \cap E_2'$ is discarded or no edge $e^{(B)}$ is add to E_3. Each $e^{(B)} \in E_3$ is chosen for a component B which contains a matching edge in M^*. Therefore, (2) holds, and we have Property 7.1

$$|M^*| \geq |E_2| + |E_3|.$$

If the minimally lf-closed subtree $T[D(v)]$ satisfies condition (A3), then we can show the next property.

CLAIM **7.1** $(2 + \theta)(|E_2| + |E_3|) \preceq |LEAF(v)|$ *holds for $\theta = \frac{2}{7}$.*

Proof: Omitted. □

Therefore, from the argument at the end of section 7.1, we have the following result.

LEMMA **7.2** *Let $G = (V, E)$ be a graph and $T = (V, F)$ be a tree rooted at r with $E \cap F = \emptyset$ such that $T + E = (V, F \cup E)$ is 2-edge-connected. Let $T[D(v)]$ be a minimally lf-closed subtree satisfying conditions $(A1) - (A3)$. Then the subset $E^{apx} = E_1 \cup E_2 \cup E_3 \subseteq E$ obtained by the procedure COVER satisfies $|E^{apx}| \leq 1.92\beta(F_v)$ for the set F_v of leaf and (pseudo-)fringe edges in $T[D(v)]$.* □

8 Reduction by COVER

We consider the remaining case in which a given minimally lf-closed tree $T[D(v)]$ does not satisfy condition (A3). That is, there is a solo edges $g \in E$ on the middle-part of a chain in $T[D(v)]$ such that its succeeding tree $T[D(v_g)]$ has at least six leaf vertices (i.e., $|LEAF(v_g)| \geq 6$). We apply procedure COVER to find an approximate solution to cover the edges in the tree $T[D(v_g)]$.

LEMMA **8.1** *For a solo edge $g = (x_1, x_2) \in E$ defined on a chain $P_T(u_1, u_k)$ $(u_1 \prec u_k)$ in a minimally lf-closed tree $T[D(v)]$, let v_g be the succeeding vertex of g, and let w^* be the highest vertex among all vertices in $P_T(u_1, u_k)$ that are incident to a vertex in $D(v_g) - v_g$ via an edge in E (see Fig. 7). Then for $F_g = E(T[D(v_g)])$ and $x = \min(w^*, up(g))$, it holds*

$$\widetilde{F_g} - F_g \subseteq \{f_1, f_2\} \cup E(P_T(x, v_g)),$$

where f_1 and f_2 are the two leaf edges adjacent to g.

Proof: Omitted. □

Given an edge set $E' \subseteq E$ that covers $F_g = E(T[D(v_g)])$, we note here that an edge set E'' can be constructed to cover $F_g \cup \{f_1, f_2\} \cup E(P_T(x, v_g)) (\supseteq \widetilde{F_g})$ by adding to E' at most three edges (two edges to cover $\{f_1, f_2\}$ and one to cover $E(P_T(x, v_g))$).

LEMMA **8.2** *For a solo edge $g \in E$ defined on a chain $P_T(u_1, u_k)$ $(u_1 \prec u_k)$, let v_g be the succeeding vertex of g. Assume that $T[D(v_g)]$ satisfies condition $(A1) - (A3)$. For $|LEAF(v_g)| \geq 6$ and any fixed $\epsilon > 0$, an edge set $E^+ \subseteq E$ that covers $\widetilde{F_g}$ and has size $|E^+| \leq (1.92 + \epsilon)\beta(F_g)$ can be found in the same time complexity of COVER applied to $T[D(v_g)]$.*

Proof: Omitted. □

If there is a solo edges g such that $|LEAV(v_g)| \geq 6$ for its succeeding vertex v_g, then we can apply Lemma 8.2 to find a $(1.92 + \epsilon)$-approximation solution to cover the edges in the tree $T[D(v_g)]$.

9 Entire description

We are now ready to describe the entire algorithm. Given a graph $H = (V, E')$ and a subset $X \subseteq V$, we denote by H/X the graph obtained from H by contracting X into a single vertex and deleting all the resulting self-loops.

<u>APPROX</u>
Input: A graph $G = (V, E)$ and a tree $T = (V, F)$ rooted at r with $E \cap F = \emptyset$ such that $T + F = (V, F \cup E)$ is 2-edge-connected, and a constant $\epsilon > 0$.
Output: A subset $E' \subseteq E$ that covers F and has size $|E'| \leq (1.92 + \epsilon)\beta(F)$.

```
E' := ∅;
while T contains more than one vertex do
  while one of Cases-1,2,3 and 4 holds do
    Execute procedures (P1),(P2),(P3), (P4)
    in Cases-1,2,3,4, respectively, and add
    to E' the edges retained by the procedure
  end; /* while */
  /* Conditions (A1) and (A2) hold. */
  Choose a minimally lf-closed subtree T[D(v)];
  if condition (A3) holds in T[D(v)] then
    Compute an edge set E^apx ⊆ E which
    covers edges in T[D(v)]
    by procedure COVER;
```

$E' := E' \cup E^{apx}$;
For $X = \{$the end vertices of edges $f \in F$ covered by $E^{apx}\}$, $T := T/X$ and $G := G/X$;
else /* $T[D(v)]$ has a solo edge g such that its succeeding tree $T[D(v_g)]$ contains at least six leaf vertices. */
Choose such succeeding tree $T[D(v_g)]$;
$F_{v_g} := \{$all edges in $T[D(v_g)]\}$;
Compute an edge set $E^+ \subseteq E$ which covers $\widetilde{F_{v_g}}$ by Lemma 8.2 with constant $\epsilon > 0$;
$E' := E' \cup E^+$;
For $X = \{$the end vertices of edges $f \in F$ covered by $E^+\}$, $T := T/X$ and $G := G/X$
end; /* while */
Output E' (after modifying E', if necessary, so that the edges f' contracted in Cases-3 and 4 are also covered in T without increasing the size of E'). □

By using the least common ancestor algorithm [5, 9] and the maximum matching algorithm [8], the above algorithm can be implemented to run in $O(n^{1/2}m + n^2)$ time.

THEOREM **9.1** *Given a graph* $G = (V, E)$ *and a tree* $T = (V, F)$ *with* $E \cap F = \emptyset$ *such that* $T + E = (V, F \cup E)$ *is 2-edge-connected, the problem of finding a smallest 2-edge-connected spanning subgraph* $H = (V, F \cup E')$ *containing* T *is* $(1.92 + \epsilon)$*-approximable in* $O(n^{1/2}m + n^2)$ *time for any fixed constant* $\epsilon > 0$*, where* $n = |V|$ *and* $m = |E \cup F|$. □

参考文献

[1] J. Cheriyan, A. Sebö and Z. Szigeti: "An improved approximation algorithm for minimum size 2-edge connected spanning subgraphs," *Lecture Notes in Computer Science*, 1412, Springer-Verlag, IPCO'98 (1998) 126–136.

[2] J. Cheriyan and R. Thurimella: "Approximating minimum-size k-connected spanning subgraphs via matching," *Proc. 37th IEEE Symp. on Found. Comp. Sci.* (1996) 292–301.

[3] K. P. Eswaran and R. E. Tarjan: "Augmentation problems," *SIAM J. Computing*, 5 (1976) 653–665.

[4] G. N. Frederickson and J. JáJá: "Approximation algorithms for several graph augmentation problems," *SIAM J. Computing*, 10 (1981) 270–283.

[5] D. Harel and R. E. Tarjan: "Fast algorithms for finding nearest common ancestors," *SIAM J. Computing*, 13 (1984) 338–355.

[6] S. Khuller and R. Thurimella: "Approximation algorithms for graph augmentation," *Proc. 19th International Colloquium on Automata, Languages and Programming Conference* (1992) 330–341.

[7] S. Khuller and U. Vishkin: "Biconnectivity approximations and graph carvings," *J. ACM*, 41 (1994) 214–235.

[8] S. Micali and V. V. Vazirani: "An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graph," *Proc. 21st IEEE Symp. on Found. Comp. Sci.* (1980) 17–27.

[9] B. Schieber and U. Vishkin: "On finding lowest common ancestors: simplification and parallelization," *SIAM J. Computing*, 17 (1988) 1253–1262.

[10] S. Tsukiyama, K. Koike and I. Shirakawa: "An algorithm to eliminate all complex triangles in a maximal planar graph for use in VLSI floor-plan," *Proc. ISCAS'86* (1986) 321–324.