IEICE TRANS. INF. & SYST., VOL.E105–D, NO.3 MARCH 2022

PAPER Special Section on Foundations of Computer Science - New Trends of Theory of Computation and Algorithm -

A Polynomial Delay Algorithm for Enumerating 2-Edge-Connected Induced Subgraphs*

Taishu ITO[†a)], **Yusuke SANO**[†], ***Nonmembers***, **Katsuhisa YAMANAKA**[†b)], ***and*** **Takashi HIRAYAMA**[†c)], ***Members***

SUMMARY The problem of enumerating connected induced subgraphs of a given graph is classical and studied well. It is known that connected induced subgraphs can be enumerated in constant time for each subgraph. In this paper, we focus on highly connected induced subgraphs. The most major concept of connectivity on graphs is vertex connectivity. For vertex connectivity, some enumeration problem settings and enumeration algorithms have been proposed, such as k-vertex connected spanning subgraphs. In this paper, we focus on another major concept of graph connectivity, edge-connectivity. This is motivated by the problem of finding evacuation routes in road networks. In evacuation routes, edge-connectivity is important, since highly edge-connected subgraphs ensure multiple routes between two vertices. In this paper, we consider the problem of enumerating 2-edge-connected induced subgraphs of a given graph. We present an algorithm that enumerates 2-edge-connected induced subgraphs of an input graph G with n vertices and m edges. Our algorithm enumerates all the 2-edge-connected induced subgraphs in $O(n^3 m|\mathcal{S}_G|)$ time, where $\mathcal{S}_G$ is the set of the 2-edge-connected induced subgraphs of G. Moreover, by slightly modifying the algorithm, we have a $O(n^3 m)$-delay enumeration algorithm for 2-edge-connected induced subgraphs.

key words: *enumeration algorithm, 2-edge-connected induced subgraph, reverse search, polynomial delay*

1. Introduction

Enumerating substructures of enormous data is a fundamental and important problem. An enumeration is one of the strong and appealing strategies to discover some knowledge from enormous data in various research areas such as data mining, bioinformatics and artificial intelligence. From this viewpoint, various enumeration algorithms have been designed.

Graphs are used to represent the relationship of objects. In web-graphs, web pages are represented by vertices of graphs and links between web pages are represented by edges. For social networks, users are represented by vertices of graphs and their friendship relations are represented by edges. In the area of bioinformatics, molecular interactions are represented by graphs. To discover valuable knowledge from practical graphs, enumeration algorithms for subgraphs with some properties are studied, such as simple/induced paths [2]–[5], simple/induced cycles [2]–[5], subtrees [6], spanning trees [4], [7], [8], k-vertex-connected spanning subgraphs [9], [10], k-edge-connected spanning subgraphs [11], maximal k-edge-connected subgraphs [12], cliques [13], [14], pseudo cliques [15], k-degenerate subgraphs [16], matchings [8], induced matchings [17], connected induced subgraphs [8], [18], [19], and so on. Several years ago, a good textbook on enumeration has been published [20]. Recently, Conte and Uno [21] proposed a new framework for enumerating maximal subgraphs with various properties in polynomial delay.

Some of the existing results above focus on closely related subgraphs. This comes from the fact that some applications on knowledge discovery need to find closely related community on graph structures. In this paper, we focus on highly edge-connected induced subgraphs. This is motivated by the problem of finding evacuation routes of road networks in time of disaster. It is easy to imagine that many roads would be broken, submerged, or closed in the disaster-hit areas. Knowledge of a single route from the current position to a shelter is apparently insufficient to secure evacuation routes in case of emergency. From this point of view, the problem of finding subgraphs with high edge-connectivity is important, since high edge-connectivity of graphs ensure multiple routes between two places. Now, we have the following question: Can we efficiently enumerate all k-edge-connected induced subgraphs? Here, an efficient enumeration implies an output polynomial or a polynomial delay enumeration. Haraguchi and Nagamochi [22] answered this question and proposed a polynomial delay enumeration algorithm for k-edge-connected induced subgraphs**. The delay of their algorithm is $O(\min\{k+1, n\}m^6)$, where n is the number of vertices and m is the number of edges in a graph.

In this paper, we focus on the problem of enumerating all 2-edge-connected induced subgraphs of a given graph and propose a more efficient enumeration algorithm. The algorithm is based on reverse search [18]. First, we define a tree structure, called a family tree, on a set of 2-edge-connected induced subgraphs of a given graph. Then, by traversing the tree, we enumerate all the 2-edge-connected induced subgraphs. For an input graph G with n vertices

Manuscript received March 25, 2021.
Manuscript publicized July 2, 2021.

[†]The authors are with Iwate University, Morioka-shi, 020–8551 Japan.

*A preliminary version appeared in the proceedings of the 14th International Frontiers of Algorithmics Workshop (FAW2020), Lecture Notes in Computer Science, vol.12340, pp.13–24, 2020 [1].

a) E-mail: shoes14@kono.cis.iwate-u.ac.jp
b) E-mail: yamanaka@cis.iwate-u.ac.jp
c) E-mail: hirayama@cis.iwate-u.ac.jp

DOI: 10.1587/transinf.2021FCP0005

**They proposed polynomial delay enumeration algorithms for connectors and various types of subgraphs. Their results include an enumeration algorithm of k-edge-connected subgraphs in a graph.

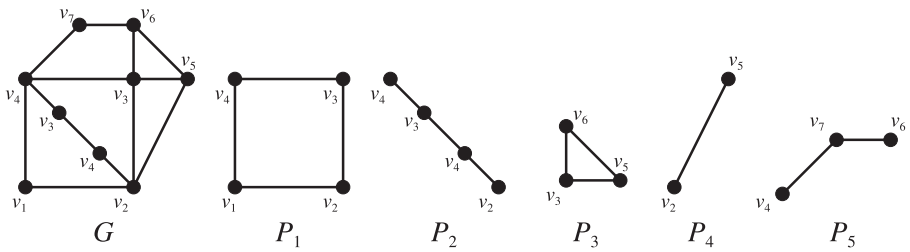

Fig. 1 An example of a closed-ear decomposition. The graph G has a closed-ear decomposition P_1, P_2, P_3, P_4, P_5.

and m edges, our algorithm runs in $O(n^3 m|\mathcal{S}_G|)$, where $\mathcal{S}_G$ is the set of the 2-edge-connected induced subgraphs of G. By applying the alternative output technique by Nakano and Uno [23], we have an enumeration algorithm that runs in $O(n^3 m)$ delay.

2. Preliminary

2.1 Graphs and Notations

In this paper, we assume that all graphs are simple, undirected, and unweighted. Let $G = (V(G), E(G))$ be a graph with vertex set $V(G)$ and edge set $E(G)$. We define $n = |V(G)|$ and $m = |E(G)|$. The *neighbor set* of a vertex v, denoted by $N(v)$, is the set of vertices adjacent to v. The *degree* of v, denoted by $d(v)$, is the number of vertices in $N(v)$. A *subgraph* of a graph G is a graph $H = (V(H), E(H))$ such that $V(H) \subseteq V(G)$ and $E(H) \subseteq \{\{u, v\} \mid u, v \in V(H) \text{ and } \{u, v\} \in E(G)\}$.

A *path* of G is an alternating sequence $\langle v_1, e_1, v_2, e_2, \ldots, e_{k-1}, v_k\rangle$ of vertices and edges, where $e_i = \{v_i, v_{i+1}\}$ for $1 \leq i \leq k-1$, such that $e_i \in E(G)$ holds. The *length*, denoted by $|P|$, of a path P is the number of the edges in the path. The path is *simple* if the path contains distinct vertices and distinct edges. Let $P = \langle v_1, e_1, v_2, e_2, \ldots, e_{k-1}, v_k\rangle$ be a simple path. We write $P = \langle v_1, v_2, \ldots, v_k\rangle$ by omitting the internal edges of P. A simple path P is an *open ear* of G if $d(v) = 2$ for each internal vertex v and $d(u) > 2$ for each endpoint u holds. A path $P = \langle v_1, v_2, \ldots, v_k\rangle$ is a *cycle* if $v_1 = v_k$ holds. A cycle is *simple* if a cycle has distinct internal vertices and distinct edges. A simple cycle P is a *closed ear* of G if $d(v_i) = 2$ for $i = 2, 3, \ldots, k-1$ and $d(v_1) > 2$.

Let $G_1 = (V(G_1), E(G_1))$ and $G_2 = (V(G_2), E(G_2))$ be two graphs. The union of G_1 and G_2 is the graph $G_1 \cup G_2 = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$. A *decomposition* of a graph G is a list $H_1, H_2, \ldots, H_k$ of subgraphs such that each edge of G appears in exactly one subgraph in the list and $G = H_1 \cup H_2 \cup \cdots \cup H_k$ holds. A *closed-ear decomposition* of G is a decomposition $P_1, P_2, \ldots, P_k$ such that P_1 is a cycle and P_i for $i \geq 2$ is either an open ear or a closed ear in $P_1 \cup P_2 \cup \cdots \cup P_i$[†]. See Fig. 1 for an example.

An *edge-cut* of G is a set $F \subseteq E(G)$ if the removal of edges in F makes G unconnected. A graph is *k-edge-connected* if every edge-cut has at least k edges. A *bridge* is an edge-cut consisting of one edge. For bridges we have the following characterization:

Theorem 1 ([24],p.23): An edge is a bridge if and only if it belongs to no cycle.

From the above theorem, if there is a cycle including an edge e, e is not a bridge. From the definition, we have the following observation.

Observation 1: A graph is 2-edge-connected if and only if the graph has no bridge.

A 2-edge-connected graph has another characterization:

Theorem 2 ([24],p.164): A graph is 2-edge-connected if and only if it has a closed-ear decomposition and every cycle in a 2-edge-connected graph is the initial cycle in some such decomposition.

An *induced subgraph* of G is a subgraph $H = (V(H), E(H))$ such that $V(H) \subseteq V(G)$ and $E(H) = \{\{u, v\} \mid u, v \in V(H) \text{ and } \{u, v\} \in E(G)\}$. We say that H is a subgraph of G induced by $V(H)$ and denoted by $G[V(H)]$. Let S be a subset of $V(G)$. We define $H + S$ as the subgraph induced by $V(H) \cup S$. Similarly, we define $H - S$ as the subgraph induced by $V(H) \setminus S$. Let $H_1 = (V(H_1), E(H_1))$ and $H_2 = (V(H_2), E(H_2))$ be two induced subgraphs of G. We define $H_1 + H_2$ as the subgraph induced by $V(H_1) \cup V(H_2)$. We define $H_1 - H_2$ as the subgraph induced by $V(H_1) \setminus V(H_2)$. An induced subgraph is an *induced path* and *induced cycle* if it forms a simple path and simple cycle, respectively.

Observation 2: Let $G = (V(G), E(G))$ be a 2-edge-connected graph. Then, G has a closed-ear decomposition that ends up with an induced cycle of G.

Proof. Let C be a induced cycle of G. Since G is 2-edge-connected, from Theorem 2, there exists a closed-ear decomposition that ends up with C. Therefore, the claim is proved. □

Now, let $H = (V(H), E(H))$ be a 2-edge-connected induced subgraph of G, and let $S \subseteq V(H)$ be a subset of $V(H)$. A vertex v in S is a *boundary* of S if v is adjacent to a vertex in $V(H) \setminus S$. A vertex subset S is *removable* if $H - S$ is 2-edge-connected.

Lemma 1: Let G be a graph, and let H be a 2-edge-connected induced subgraph of G. Suppose that H is not

[†]A closed-ear decomposition is an *ear decomposition* if every ear in the decomposition is an open ear.

an induced cycle. Then, H has a removable set.

Proof. From Observation 2, H has a closed-ear decomposition that ends up with an induced cycle. Let $C = (V(C), E(C))$ be such an induced cycle of G. Then, we can observe that $V(H) \setminus V(C)$ is a removable set. □

A removable set S of H is *minimal* if any $S' \subset S$ is not a removable set of H. We have the following properties of minimal removable sets.

Lemma 2: Let G be a graph, and let H be a 2-edge-connected induced subgraph of G. Let S be a minimal removable set of H. Then, $G[S]$ is connected.

Proof. Suppose for a contradiction that $G[S]$ is unconnected. Let S' be a subset of S such that $G[S']$ is a connected component in $G[S]$. Then, S' is a removable set, which contradicts to the minimality of S. □

Lemma 3: Let G be a graph, and let H be a 2-edge-connected induced subgraph of G. Any minimal removable set S, $|S| \geq 2$, of H has exactly two boundaries.

Proof. Proof by contradiction. We first assume that the number of boundaries of S is 1. Let v be the boundary in S. From 2-edge-connectivity of H, v is adjacent to two or more vertices in $V(H) \setminus S$ (otherwise, v is incident to a bridge, which is contradiction). Hence, $H - (S \setminus \{v\})$ is 2-edge-connected. This means that $S \setminus \{v\}$ is a removable set, which contradicts to the minimality of S. Next, we assume that the number of boundaries of S is 3 or more. From Lemma 2, $G[S]$ is connected. Hence, for any two boundaries in S, there is a path between them in $G[S]$. Let x, y be two boundaries of S such that the shortest path $P_{x,y}$ between them has no other boundary as its internal vertex. Note that $P_{x,y}$ is an induced path of H. Then, $S \setminus V(P_{x,y})$ is a removable set of H. This contradicts to the minimality of S. □

Now, we have the following key lemma.

Lemma 4: Let G be a graph, and let H be a 2-edge-connected induced subgraph of G. Let S, $|S| \geq 2$, be a minimal removable set of H. Then, $G[S]$ is a path of length $|S| - 1$.

Proof. We assume for a contradiction that $G[S]$ is not a path. From Lemma 3, S has two boundaries. Let u, v be the two boundaries in S. From Lemma 2, there is a path between u and v. Let $P_{u,v}$ be a shortest path between u and v. Note that $P_{u,v}$ has no other boundary as its internal vertex. Then, $S \setminus V(P_{u,v})$ is a removable set of H which contradicts to minimality of S. □

From Lemma 3 and Lemma 4, we can write a minimal removable set as a sequence $S = \langle u_1, u_2, \ldots, u_k \rangle$. Moreover, any internal vertex is not boundary of S. That is, the endpoints of S are boundaries. From now on, we assume that the two endpoints u_1 and u_k are boundaries of S.

A path $P = \langle w_1, w_2, \ldots, w_\ell \rangle$ with $|V(P)| \geq 2$ of H is an *internal ear* if (1) w_1 and w_k are the two boundaries of $V(P)$ and (2) $d(w_i) = 2$ in H holds for $i = 1, 2, \ldots, \ell$. An internal ear P is *maximal* if there is no internal ear P' such that P' includes P as its subpath. We have the following observation on forms of minimal removable sets.

Observation 3: Let G be a graph, and let H be a 2-edge-connected induced subgraph of G. Let $S = \langle u_1, u_2, \ldots, u_k \rangle$ be a minimal removable set of H with two boundaries u_1, u_k. Then,

1. if $|S| = 1$, S forms a path in H with length 0 (in this case, $u_1 = u_k$) and
2. if $|S| \geq 2$, S forms a maximal internal ear of H.

Proof. The case of $|S| = 1$ is trivial. Therefore, in this proof, we consider the case of $|S| \geq 2$, below.

From Lemma 4, S forms a path with two boundaries u_1 and u_k. We first show that $d(u_1) = 2$ in H. From the 2-edge-connectivity of H, it can be observed that $d(u_1) \geq 2$ holds. Now, suppose that $d(u_1) > 2$ holds. Then, $S \setminus \{v_1\} \subset S$ is removable, which is a contradiction. Hence, $d(u_1) = 2$ holds. The same discussion can be applied to the case of u_k.

Next, we show the maximality of S. Assume for a contradiction that S is not maixmal. Let S' be an internal ear such that it includes S as a subpath. From the definition of internal ears, it can be observed that $H - S$ includes degree-1 vertex, and hence $H - S$ is not 2-edge-connected. This contradicts that S is removable. □

From Observation 3, a minimal removable set of a 2-edge-connected induced subgraph H forms a maximal internal ear of H. However, note that the reverse direction is not always true.

2.2 Enumeration Algorithms

For algorithms of normal decision problems or optimization problems, we estimate the running time of the whole algorithm as a function of input size. On the other hands, in enumeration problems, we sometimes have exponential outputs for input size. For enumeration algorithms, we use particular running-time analysis. In this subsection, we introduce some analysis ways for enumeration algorithms.

Let $\mathcal{A}$ be an enumeration algorithm for an enumeration problem Π with input size n and output size α. The algorithm $\mathcal{A}$ is *output polynomial* if $\mathcal{A}$ solves Π in $O(n^c \alpha^d)$ time, where c, d are some constants. The algorithm $\mathcal{A}$ *P-enumerates* if $\mathcal{A}$ solves Π in $O(n^c \alpha)$ [25]. Then, we say that $\mathcal{A}$ enumerates every solution of Π in $O(n^c)$ time for each. A *delay* of $\mathcal{A}$ is a computation time between two consecutive outputs. The algorithm $\mathcal{A}$ is *polynomial delay* if (1) the first solution is output in $O(n^c)$ time for some constant c and (2) the delay of $\mathcal{A}$ is bounded above by $O(n^c)$ [26]. Then, we say that, $\mathcal{A}$ enumerates every solution in $O(n^c)$ delay.

3. Family Tree of 2-Edge-Connected Induced Subgraphs

In this section, we define a tree structure among the set of

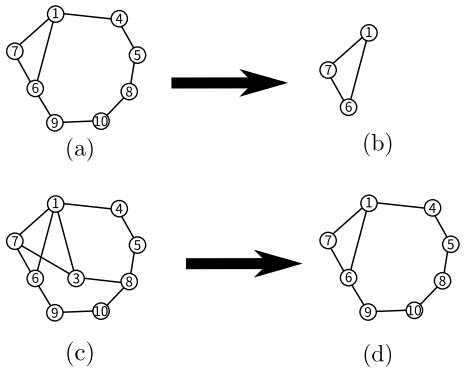

Fig. 2 (a) A 2-edge-connected induced subgraph H_1. (b) The parent $\mathcal{P}(H_1)$ of H_1. $\mathcal{P}(H_1)$ is obtained from H_1 by removing $\{4, 5, 8, 10, 9\}$, which is the smallest minimal removable set of H_1. (c) A 2-edge-connected induced subgraph H_2. (d) The parent $\mathcal{P}(H_2)$ of H_2. $\mathcal{P}(H_2)$ is obtained from H_2 by removing $\{3\}$, which is the smallest minimal removable set of H_2.

2-edge-connected induced subgraphs of an input graph. The vertices of the tree structure corresponds to the set of 2-edge-connected induced subgraphs, each edge corresponds to a parent-child relation between two 2-edge-connected induced subgraphs, and the root is the empty graph.

We define some notations. Let $G = (V(G), E(G))$ be a graph with a labeled-vertex set $V(G) = \{v_1, v_2, \ldots, v_n\}$ and an edge set $E(G)$. Let $\mathcal{S}_G$ be the set of the 2-edge-connected induced subgraphs of G, and let $C_G \subseteq \mathcal{S}_G$ be the set of the induced cycles of G. We say that v_i is *smaller* than v_j, denoted by $v_i \prec v_j$, if $i < j$ holds. Let $H = (V(H), E(H))$ be a 2-edge-connected induced subgraph of G. Let $S_1 = \langle u_1, u_2, \ldots, u_k \rangle$, $(u_1 \prec u_k)$, and $S_2 = \langle w_1, w_2, \ldots, w_k \rangle$, $(w_1 \prec w_k)$, be two minimal removable sets of H. S_1 is smaller than S_2 if $u_1 \prec w_1$ holds. Note that, for two minimal removable sets S_1 and S_2, $S_1 \cap S_2 = \emptyset$ holds.

Let S be the smallest minimal removable set of H. Then, we define the *parent* of H, as follows.

$$\mathcal{P}(H) := \begin{cases} \emptyset & (H \in C_G) \\ H - S & (H \in \mathcal{S}_G \setminus C_G), \end{cases}$$

where $\emptyset$ reperesents the empty graph, which is the graph with 0 vertex and 0 edge. We say that H is a *child* of the parent of $\mathcal{P}(H)$. Examples of parents are shown in Fig. 2. If $H \in \mathcal{S}_G \setminus C_G$ holds, H has a removable set from Lemma 1. Hence, H always has its parent. Moreover, the parent is defined uniquely, since the smallest minimal removable set is unique in H. Note that $\mathcal{P}(H)$ is also a 2-edge-connected induced subgraph of G. For a 2-edge-connected induced subgraph in C_G, we define its parent as the empty graph $\emptyset$. By repeatedly finding parents from H, we obtain a sequence of 2-edge-connected induced subgraphs of G or the empty graph. We define the sequence $\mathcal{PS}(H) = \langle H_1, H_2, \ldots, H_\ell \rangle$, where $H_1 = H$ and $H_i = \mathcal{P}(H_{i-1})$ for $i = 2, 3, \ldots, \ell$, the *parent sequence* of H. An example of a parent sequence is shown in Fig. 3. This sequence ends up with the empty graph, as shown in the following lemma.

Lemma 5: Let H be a 2-edge-connected induced subgraph of a graph G, and let $\mathcal{PS}(H) = \langle H_1, H_2, \ldots, H_\ell \rangle$ be the parent sequence of H. Then, H_ℓ is the empty graph $\emptyset$.

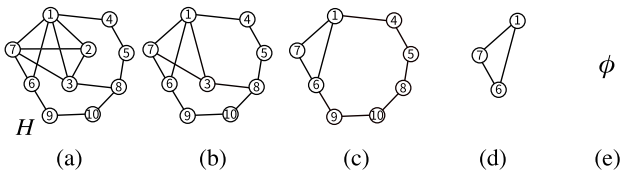

Fig. 3 An example of the parent sequence of a 2-edge-connected induced subgraph. (a) A 2-edge-connected induced subgraph H. (b) The parent $\mathcal{P}(H)$ of H. $\mathcal{P}(H)$ is obtained from H by removing $\{2\}$. (c) The parent $\mathcal{P}(\mathcal{P}(H))$ of $\mathcal{P}(H)$. $\mathcal{P}(\mathcal{P}(H))$ is obtained by removing $\{3\}$. (d) The parent $\mathcal{P}(\mathcal{P}(\mathcal{P}(H)))$ of $\mathcal{P}(\mathcal{P}(H))$. $\mathcal{P}(\mathcal{P}(\mathcal{P}(H)))$ is obtained by removing $\{4, 5, 8, 10, 9\}$. (e) Finally, the root, the empty graph, is obtained by removing $\{1, 6, 7\}$.

Proof. We define a potential function $\phi(I) = |V(I)|$ for an induced subgraph I. It is observed that $\phi(I) = 0$ if and only if I is the empty graph. Let H_i, $1 \leq i \leq \ell - 1$, be a 2-edge-connected induced subgraph on $\mathcal{PS}(H)$. If $H_i \in \mathcal{S}_G \setminus C_G$, then there always exists its parent $\mathcal{P}(H_i)$ from Lemma 1. Otherwise, $H_i \in C_G$, its parent is the empty graph. Therefore, $\phi(\mathcal{P}(H_i)) \leq \phi(H_i)$ holds. Hence, $H_i \neq H_j$ holds for any i, j with $i \neq j$. Thus, the claim is proved. □

From Lemma 5, by merging the parent sequences of all 2-edge-connected induced subgraphs of G, we have the tree structure, called *family tree*, in which (1) the root is the empty graph, (2) the vertices except the root are 2-edge-connected induced subgraphs of G, and (3) each edge corresponds to a parent-child relation of two induced subgraphs of G. An example of the family tree is shown in Fig. 4.

4. Enumeration Algorithm

In this section, we present an enumeration algorithm for the 2-edge-connected induced subgraphs of an input graph. In the previous section, we defined the family tree rooted at the empty graph. Our algorithm enumerates 2-edge-connected induced subgraphs by traversing the tree. The children of the root in the tree are the induced cycles of an input graph. Our algorithm first enumerates the induced cycles of an input graph. Then, for each induced cycle, we traverse the subtree rooted at the cycle. To traverse the family tree, we have to design an enumeration algorithm for induced cycles of a graph and a child-enumeration algorithm for any 2-edge-connected induced subgraph. Fortunately, an efficient induced-cycle-enumeration algorithm is already known [5]. In our algorithm, we use the existing algorithm for enumerating the induced cycles of a graph. Now, in this section, we present a child-enumeration algorithm for 2-edge-connected induced subgraphs of a graph, below.

Let $G = (V(G), E(G))$ be a graph with a labeled-vertex set $V(G) = \langle v_1, v_2, \ldots, v_n \rangle$ and an edge set $E(G)$. Let $\mathcal{S}_G$ be the set of 2-edge-connected induced subgraphs of G, and let $C_G \subseteq \mathcal{S}_G$ be the set of the induced cycles of G. To generate a child, we do the reverse operation for finding parents which is to attach a maximal internal ear to H. If the vertex

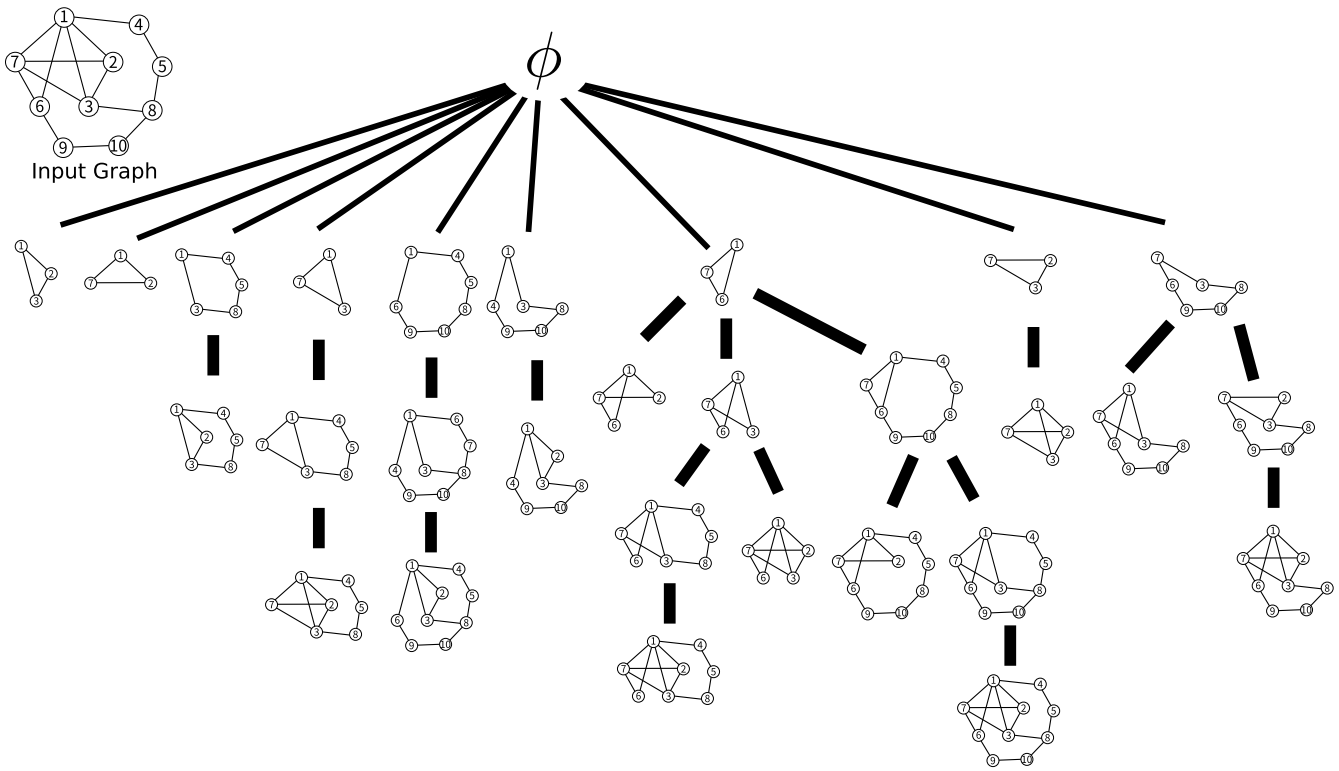

Fig. 4 An example of family tree of the input graph, drawn in the upper-left of the figure.

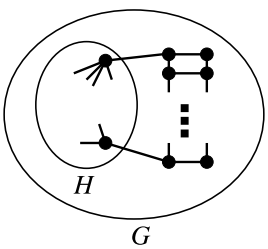

Fig. 5 A case that 2-edge-connected induced subgraph H of an input graph G has exponential child-candidates.

set S of the attached path is the smallest minimal removable set in $H + S$, then $H + S$ is a child of H. Otherwise, $H + S$ is not a child. Let $\mathcal{I}(H, s, t)$ be the set of paths P such that P is a maximal internal ear in $H + P$ from s to t for $s, t \in N(H)$, where $N(H) := \bigcup_{u \in V(H)} N(u) \cap (V(G) \setminus V(H))$. For any different two $s, t \in N(H)$ and for any $P \in \mathcal{I}(H, s, t)$, $H + P$ is a candidate of a child, that is, $H + P$ may be a child. Therefore, if we generate all the paths in $\mathcal{I}(H, s, t)$ for every $s, t \in N(H)$, then all the children of H are enumerated by checking whether or not $H + P$ for each $P \in \mathcal{I}(H, s, t)$ is a child. However, this method may take exponential time. It can be observed that $|\mathcal{I}(H, s, t)|$ can be an exponential of the number of vertices in $V(G) \setminus V(H)$ when $G - H$ has a "ladder" subgraph, as shown in Fig. 5 (one can choose to pass or not each rung in a ladder subgraph). Hence, there may be exponential child-candidates. If all the exponential candidates are non-children, the above child-enumeration method takes exponential time for checking whether or not H has at least one child. However, fortunately, we can check whether or not H has at least one child in polynomial time, as follows.

$\mathcal{M}(H)$ denotes the set of minimal removable sets of H. We can observe that, for any $P, P' \in \mathcal{I}(H, s, t)$, $\mathcal{M}(H + P) \setminus \{P\} = \mathcal{M}(H + P') \setminus \{P'\}$ holds. Therefore, if, for any $P \in \mathcal{I}(H, s, t)$, P is the smallest minimal removable set among $\mathcal{M}(H + P)$, then every $P' \in \mathcal{I}(H, s, t) \setminus \{P\}$ is also the smallest one among $\mathcal{M}(H + P')$. This implies that, if $H + P$ for $P \in \mathcal{I}(H, s, t)$ is a child of H, then $H + P'$ for every $P' \in \mathcal{I}(H, s, t) \setminus \{P\}$ is also a child of H. Hence, we focus on determining whether or not $H + P$ for a path $P \in \mathcal{I}(H, s, t)$, where $s, t \in N(H)$ and $s \prec t$, is a child of H. In the following case-analysis, we consider whether or not attaching $P \in \mathcal{I}(H, s, t)$ to H produces a child.

Case 1: $s \prec v$ for every $v \in V(H)$.

Obviously, for any path $P \in \mathcal{I}(H, s, t)$, $V(P)$ is the smallest minimal removable set in $H + P$. Therefore, $H + P$ is a child of H.

Case 2: Otherwise.

Let P be any path in $\mathcal{I}(H, s, t)$. If $V(P)$ is the smallest one among $\mathcal{M}(H + P)$, then $H + P$ is a child, and hence every $P' \in \mathcal{I}(H, s, t) \setminus \{P\}$ is also a child. Otherwise, $H + P$ is a non-child, and hence every $P' \in \mathcal{I}(H, s, t) \setminus \{P\}$ is also a non-child.

Examples of children are shown in Fig. 6. The figure includes three child-candidates of a 2-edge-connected induced subgraph.

Now, from the above case-analysis, we have the algorithm described in **Algorithm 1** and **Algorithm 2**. In **Algo-**

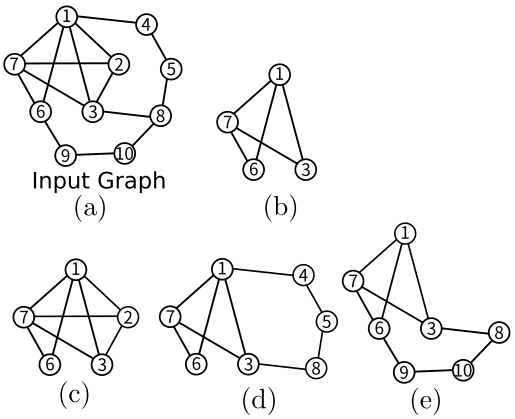

Fig. 6 Examples of children. (a) An input graph G. (b) A 2-edge-connected induced subgraph H of G. The smallest minimal removable set of H is {3}. (c) The induced subgraph obtained from H by inserting {2}. This is a child, since the smallest minimal removable set is {2}. (d) The induced subgraph obtained from H by inserting $\{4, 5, 8\}$. This subgraph is a child, since $\{4, 5, 6\}$ is the smallest minimal removable set. Note that, in this case, although $3 \prec 4$ holds, {3} turns non-removable. (e) The induced subgraph obtained from H by inserting $\{8, 10, 9\}$. Here, the smallest minimal removable set is {1}. Hence, this is not a child. Note that, in this case, the set {1} turns removable.

Algorithm 1: ENUM-2-EDGE-CONN-IND-SUBGRAPHS($G = (V(G), E(G))$)

```
1  begin
2      /* An input is a simple, unweighted, and
          undirected graph G = (V(G), E(G)).  Outputs
          are all the 2-edge-connected induced
          subgraphs of G.                          */
3      Let C(G) be the set of the induced cycles of G.
4      foreach C ∈ C(G) do
5          Call FIND-CHILDREN(G, C)
```

Algorithm 2: FIND-CHILDREN($G = (V(G), E(G))$, $H = (V(H), E(H))$)

```
1  begin
2      /* Find all the children of a given
          2-edge-connected induced subgraph H.     */
3      Output H.
4      foreach s, t ∈ N(H), s ≺ t do
5          if s ≺ v for every v ∈ V(H) then
6              foreach P ∈ I(H, s, t) do
7                  FIND-CHILDREN(G, H + P)
8          else
9              Let P be any path in I(H, s, t).
10             Construct M(H + P).
11             if V(P) is the smallest one among M(H + P) then
12                 foreach P ∈ I(H, s, t) do
13                     FIND-CHILDREN(G, H + P)
```

rithm 1, we are required to enumerate the induced cycles in an input graph G. This enumeration can be done using an existing algorithm [3], which enumerates all the induced cycles in C_G in $\tilde{O}(m + n|C_G|)$ time, where $\tilde{O}(f)$ is a shorthand for $O(f \cdot \text{polylog } n)$.

For each generated induced cycle, we traverse the subtree rooted at the cycle using **Algorithm 2**. In the line 10 in **Algorithm 2**, the algorithm constructs $\mathcal{M}(H + P)$. This can be done, as follows. For each vertex u with degree 3 or more in $H + P$, we check whether or not $H + P - \{u\}$ is 2-edge-connected. If the answer is yes, $\{u\}$ is a member of $\mathcal{M}(H + P)$. This check can be done in $O(m)$ time using a depth-first search. For each vertex u with degree 2 in $H + P$, we first find the maximal internal ear including u by traversing from u (this can be found in $O(n)$ time). Let P' be the found path. Then, we check whether or not $H + P - P'$ is 2-edge-connected (this takes $O(m)$ time using a depth-first search). If the answer is yes, $V(P')$ is a member of $\mathcal{M}(H + P)$. The above process can list all the members of $\mathcal{M}(H + P)$ up and done in $O(nm) + O(nm)$. Note that the number of the internal ears in $H + P$ is bounded by $O(n)$. All the paths in $\mathcal{I}(H, s, t)$ can be enumerated using an enumeration algorithm for induced paths [3], which enumerates all the paths in $\mathcal{I}(H, s, t)$ in $\tilde{O}(m + n|\mathcal{I}(H, s, t)|)$ time. Now, we estimate the running time of one recursive call of **Algorithm 2**. Suppose that H has k children. For each $O(n^2)$ pairs of s and t, we construct $\mathcal{M}(H + P)$. This takes $O(n^3m)$ time. Since H has k children, the enumeration algorithm for induced paths runs at most k times in lines 6 or 12. Let $\{s_1, t_1\}, \{s_2, t_2\}, \ldots, \{s_x, t_x\}$ be pairs of vertices in $N(H)$ such that, for $P \in \mathcal{I}(H, s_i, t_i)$, $H + P$ is a child of H. Let $k_i = |\mathcal{I}(H, s_i, t_i)|$. Note that there exists k_i children generated by attaching $P \in \mathcal{I}(H, s_i, t_i)$. Then, $k = \sum_{1 \le i \le x} k_i$ holds. Now, the total running time for enumerating paths in $\mathcal{I}(H, s_i, t_i)$ is bounded by $\sum_{1 \le i \le x} \tilde{O}(m + nk_i)$, which is bounded by $k \cdot \tilde{O}(m + n)$. Hence, each recursive call of **Algorithm 2** takes $O(n^3m) + k \cdot \tilde{O}(m + n)$ time. Therefore, we have the following theorem.

Theorem 3: Let $G = (V(G), E(G))$ be a graph with n vertices and m edges. Let $\mathcal{S}_G$ be the set of the 2-edge-connected induced subgraphs of G. One can enumerate all the 2-edge-connected induced subgraphs of G in $O(n^3m|\mathcal{S}_G|)$ time.

Proof. All the children of the root of the family tree, namely the induced cycles of G, can be enumerated in $\tilde{O}(m + n|C_G|)$ time [3]. The recursive call for any 2-edge-connected induced subgraph H takes $O(n^3m) + k \cdot \tilde{O}(m + n)$ if H has k children. Hence, the total running time of our algorithm is bounded by $O(n^3m|\mathcal{S}_G|)$. □

Now, we show that one can enumerate 2-edge-connected induced subgraphs in $O(n^3m)$ delay. Suppose that there exist $O(T_C)$-delay and $O(T_P)$-delay enumeration algorithms for induced cycles and s-t paths, respectively. As shown in **Algorithm 1** and **Algorithm 2**, first, we enumerate induced cycles of an input graph G by the $O(T_C)$-delay enumeration algorithm. Then, for each induced cycle, we traverse the subtree rooted at the cycle. In that traversal, we apply the alternative output technique [23]. In the technique, 2-edge-connected induced subgraphs with even-depth in the

tree are output before their children and 2-edge-connected induced subgraphs with odd-depth in the tree are output after their children. In the above traversal, we have an output per at most 3 edge traversals in each subtree. Each edge traversal takes $O(\max\{n^3m, T_P\})$ time. Hence, we have the following corollary.

Corollary 1: Let $G = (V(G), E(G))$ be a graph with n vertices and m edges. One can enumerate every 2-edge-connected induced subgraph of G in $O(\max\{n^3m, T_P, T_C\})$ delay.

Uno and Satoh [5] proposed efficient enumeration algorithms for induced cycles and s-t paths of a given graph. From [5], one can see that $T_C, T_P \in O(n^3m)$ holds. Hence, we have the following corollary.

Corollary 2: Let $G = (V(G), E(G))$ be a graph with n vertices and m edges. One can enumerate every 2-edge-connected induced subgraph in $O(n^3m)$ delay.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Numbers JP18H04091 and JP19K11812.

References

[1] Y. Sano, K. Yamanaka, and T. Hirayama, "A polynomial delay algorithm for enumerating 2-edge-connected induced subgraphs," Frontiers in Algorithmics - 14th International Workshop, FAW 2020, Haikou, China, Oct. 19-21, 2020, Proceedings, ed. M. Li, Lecture Notes in Computer Science, vol.12340, pp.13–24, Springer, 2020.

[2] E. Birmelé, R. Ferreira, R. Grossi, A. Marino, N. Pisanti, R. Rizzi, and G. Sacomoto, "Optimal Listing of Cycles and st-Paths in Undirected Graphs," Proc. 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012), pp.1884–1896, Jan. 2012.

[3] R. Ferreira, R. Grossi, R. Rizzi, G. Sacomoto, and M.F. Sagot, "Amortized $\tilde{O}(|V|)$-delay algorithm for listing chordless cycles in undirected graphs," Proc. 22nd European Symposium on Algorithms (ESA 2014), Lecture Notes in Computer Science, vol.8737, pp.418–429, 2014.

[4] R.C. Read and R.E. Tarjan, "Bounds on backtrack algorithms for listing cycles, paths, and spanning trees," Networks, vol.5, no.3, pp.237–252, 1975.

[5] T. Uno and H. Satoh, "An efficient algorithm for enumerating chordless cycles and chordless paths," Proc. 17th International Conference on Discovery Science (DS 2014), vol.8777, pp.313–324, 2014.

[6] K. Wasa, Y. Kaneta, T. Uno, and H. Arimura, "Constant time enumeration of subtrees with exactly k nodes in a tree," IEICE Trans. Inf. & Syst., vol.97-D, no.3, pp.421–430, 2014.

[7] A. Shioura, A. Tamura, and T. Uno, "An optimal algorithm for scanning all spanning trees of undirected graphs," SIAM Journal of Computing, vol.26, no.3, pp.678–692, 1997.

[8] T. Uno, "Constant time enumeration by amortization," Proc. 14th International Symposium on Algorithms and Data Structures (WADS 2015), vol.9214, pp.593–605, 2015.

[9] E. Boros, K. Borys, K.M. Elbassioni, V. Gurvich, K. Makino, and G. Rudolf, "Generating minimal k-vertex connected spanning subgraphs," Proc. 13th Annual International Computing and Combinatorics Conference (COCOON 2007), pp.222–231, 2007.

[10] L. Khachiyan, E. Boros, K. Borys, K.M. Elbassioni, V. Gurvich, and K. Makino, "Enumerating spanning and connected subsets in graphs and matroids," Proc. 14th Annual European Symposium on Algorithms (ESA 2006), vol.4168, pp.444–455, 2006.

[11] K. Yamanaka, Y. Matsui, and S. Nakano, "Enumerating highly-edge-connected spanning subgraphs," IEICE Trans. Fundamentals, vol.102-A, no.9, pp.1002–1006, 2019.

[12] T. Akiba, Y. Iwata, and Y. Yoshida, "Linear-time enumeration of maximal k-edge-connected subgraphs in large networks by random contraction," Proc. 22nd ACM International Conference on Information and Knowledge Management (CIKM 2013), pp.909–918, 2013.

[13] A. Conte, R.D. Virgilio, A. Maccioni, M. Patrignani, and R. Torlone, "Finding all maximal cliques in very large social networks," Proc. 19th International Conference on Extending Database Technology, pp.173–184, 2016.

[14] K. Makino and T. Uno, "New algorithms for enumerating all maximal cliques," Proc. 9th Scandinavian Workshop on Algorithm Theory (SWAT 2004), vol.3111, pp.260–272, 2004.

[15] T. Uno, "An efficient algorithm for solving pseudo clique enumeration problem," Algorithmica, vol.56, no.1, pp.3–16, 2010.

[16] A. Conte, M.M. Kanté, Y. Otachi, T. Uno, and K. Wasa, "Efficient enumeration of maximal k-degenerate subgraphs in a chordal graph," Proc. 23rd Annual International Computing and Combinatorics Conference (COCOON 2017), vol.10392, pp.150–161, 2017.

[17] K. Kurita, K. Wasa, T. Uno, and H. Arimura, "Efficient enumeration of induced matchings in a graph without cycles with length four," IEICE Trans. Fundamentals, vol.101-A, no.9, pp.1383–1391, 2018.

[18] D. Avis and K. Fukuda, "Reverse search for enumeration," Discrete Applied Mathematics, vol.65, no.1-3, pp.21–46, 1996.

[19] S. Maxwell, M.R. Chance, and M. Koyutürk, "Efficiently enumerating all connected induced subgraphs of a large molecular network," Prceedings of The First International Conference on Algorithms for Computational Biology, vol.8542, pp.171–182, 2014.

[20] A. Marino, Analysis and Enumeration, Atlantis press, 2015.

[21] A. Conte and T. Uno, "New polynomial delay bounds for maximal subgraph enumeration by proximity search," Proc. 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC 2019), pp.1179–1190, 2019.

[22] K. Haraguchi and H. Nagamochi, "Design of polynomial-delay enumeration algorithms in transitive systems," CoRR, vol.abs/2004.01904, 2020.

[23] S. Nakano and T. Uno, "Generating colored trees," Proc. 31th Workshop on Graph-Theoretic Concepts in Computer Science, (WG 2005), vol.LNCS 3787, pp.249–260, 2005.

[24] D.B. West, Introduction to Graph Theory, 2 ed., Prentice Hall, Sept. 2000.

[25] L.G. Valiant, "The complexity of computing the permanent," Theoretical Computer Science, vol.8, no.2, pp.189–201, 1979.

[26] D.S. Johnson, C.H. Papadimitriou, and M. Yannakakis, "On generating all maximal independent sets," Information Processing Letters, vol.27, no.3, pp.119–123, 1988.

Taishu Ito received his B.E. degree from Iwate University in 2021. His research interests include combinatorial algorithms and graph algorithms.

Yusuke Sano received his B.E. and M.E. degrees from Iwate University in 2018 and 2020, respectively. His research interests include combinatorial algorithms and graph algorithms.

Katsuhisa Yamanaka is a professor of Faculty of Science and Engineering, Iwate University. He received B.E., M.E. and Dr. Eng. degrees from Gunma University in 2003, 2005 and 2007, respectively. His research interests include combinatorial algorithms and graph algorithms.

Takashi Hirayama received his B.E., M.E., and Ph.D. degrees in computer science from Gunma University in 1994, 1996, and 1999, respectively. From 1999 to 2001 he was a research assistant in the Department of Electrical and Electronics Engineering, Ashikaga Institute of Technology. He is currently a lecturer of Faculty of Science and Engineering, Iwate University. His research interests include high level and logic synthesis and design for testability of VLSIs.