Check for updates

A Linear Delay Algorithm for Enumeration of 2-Edge/Vertex-Connected Induced Subgraphs

Takumi Tada(✉) and Kazuya Haraguchi

Graduate School of Informatics, Kyoto University, Kyoto, Japan
{tada,haraguchi}@amp.i.kyoto-u.ac.jp

Abstract. In this paper, we present the first linear delay algorithms to enumerate all 2-edge-connected induced subgraphs and to enumerate all 2-vertex-connected induced subgraphs for a given simple undirected graph. We treat these subgraph enumeration problems in a more general framework based on set systems. For an element set V, $(V, \mathcal{C} \subseteq 2^V)$ is called a *set system*, where we call $C \in \mathcal{C}$ a *component*. A nonempty subset $Y \subseteq C$ is a *removable set of* C if $C \setminus Y$ is a component and Y is a *minimal removable set* (*MRS*) *of* C if it is a removable set and no proper nonempty subset $Z \subsetneq Y$ is a removable set of C. We say that a set system has *subset-disjoint* (*SD*) property if, for every two components $C, C' \in \mathcal{C}$ with $C' \subsetneq C$, every MRS Y of C satisfies either $Y \subseteq C'$ or $Y \cap C' = \emptyset$. We assume that a set system with SD property is implicitly given by an oracle that returns an MRS of a component which is given as a query. We provide an algorithm that, given a component C, enumerates all components that are subsets of C in linear time/space with respect to $|V|$ and oracle running time/space. We then show that, given a simple undirected graph G, the pair of the vertex set $V = V(G)$ and the family of vertex subsets that induce 2-edge-connected (or 2-vertex-connected) subgraphs of G has SD property, where an MRS in a 2-edge-connected (or 2-vertex-connected) induced subgraph corresponds to either an ear or a single vertex with degree greater than two.

Keywords: Enumeration of subgraphs · 2-edge-connectivity · 2-vertex-connectivity · Binary partition · Linear delay

1 Introduction

Given a graph, subgraph enumeration asks to list all subgraphs that satisfy required conditions. It could find interesting substructures in network analysis. Enumeration of cliques is among such problems [2,3], where a clique is a subgraph such that every two vertices are adjacent to each other and thus may represent a group of so-called SNS users that are pairwise friends. Pursuing further applications, there have been studied enumeration of subgraphs that satisfy weaker connectivity conditions, e.g., pseudo-cliques [12].

The work is partially supported by JSPS KAKENHI Grant Number 20K04978 and 22H00532. A preprint of this paper appeared as [8].

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
S.-Y. Hsieh et al. (Eds.): IWOCA 2023, LNCS 13889, pp. 368–379, 2023.
https://doi.org/10.1007/978-3-031-34347-6_31

In this paper, we consider enumeration of subgraphs that satisfy fundamental connectivity conditions; 2-edge-connectivity and 2-vertex-connectivity. For a graph G, let $V(G)$ and $E(G)$ denote the set of vertices of G and the set of edges of G, respectively. Let $n := |V(G)|$ and $m := |E(G)|$. An enumeration algorithm in general outputs many solutions, and its *delay* refers to computation time between the start of the algorithm and the first output; between any consecutive two outputs; and between the last output and the halt of the algorithm. The algorithm attains *polynomial delay* (resp., *linear delay*) if the delay is bounded by a polynomial (resp., a linear function) with respect to the input size.

The main results of the paper are summarized in the following two theorems.

Theorem 1. *For a simple undirected graph G, all 2-edge-connected induced subgraphs of G can be enumerated in $O(n+m)$ delay and space.*

Theorem 2. *For a simple undirected graph G, all 2-vertex-connected induced subgraphs of G can be enumerated in $O(n+m)$ delay and space.*

We achieve the first linear delay algorithms for enumerating 2-edge/vertex connected induced subgraphs. Ito et al. [7] made the first study on enumeration of 2-edge-connected induced subgraphs, presenting a polynomial delay algorithm based on reverse search [1] such that the delay is $O(n^3m)$. For an element set V, $(V, \mathcal{C} \subseteq 2^V)$ is called a *confluent set system* if, for every three components $X, Y, Z \in \mathcal{C}$, $Z \subseteq X \cap Y$ implies $X \cup Y \in \mathcal{C}$. Haraguchi and Nagamochi [5] studied an enumeration problem in a confluent set system that includes enumeration of k-edge-connected (resp., k-vertex-connected) induced subgraphs as special cases, which yields $O(\min\{k+1, n\}n^5m)$ (resp., $O(\min\{k+1, n^{1/2}\}n^{k+4}m)$) delay algorithms. Wen et al. [13] proposed an algorithm for enumerating maximal vertex subsets that induce k-vertex-connected subgraphs such that the total time complexity is $O(\min\{n^{1/2}, k\}m(n+\delta(G)^2)n)$, where $\delta(G)$ denotes the minimum degree over the graph G.

We deal with the two subgraph enumeration problems in a more general framework. For a set V of elements, let $\mathcal{C} \subseteq 2^V$ be a family of subsets of V. A pair $(V, \mathcal{C})$ is called a *set system* and a subset $C \subseteq V$ is called a *component* if $C \in \mathcal{C}$. A nonempty subset $Y \subseteq C$ of a component C is a *removable set of* C if $C \setminus Y \in \mathcal{C}$. Further, a removable set Y of C is minimal, or a *minimal removable set* (*MRS*), if there is no $Z \subsetneq Y$ that is a removable set of C. We denote by $\textsc{Mrs}_{\mathcal{C}}(C)$ the family of all MRSs of C. Let us introduce the notion of SD property of set system as follows.

Definition 1. A set system $(V, \mathcal{C})$ has *subset-disjoint (SD) property* if, for any two components $C, C' \in \mathcal{C}$ such that $C \supsetneq C'$, either $Y \subseteq C'$ or $Y \cap C' = \emptyset$ holds for every MRS Y of C.

We consider the problem of enumerating all components that are subsets of a given component in a set system with SD property. We assume that the set system is implicitly given by an oracle such that, for a component $C \in \mathcal{C}$ and a subset $X \subseteq C$, the oracle returns an MRS Y of C that is disjoint with X

if exists; and NIL otherwise. We denote the time and space complexity of the oracle by θ_t and θ_s, respectively. We show the following theorem that is a key for proving Theorems 1 and 2.

Theorem 3. *Let $(V, \mathcal{C})$ be a set system with SD property, $C \in \mathcal{C}$ be a component and $n := |C|$. All components that are subsets of C can be enumerated in $O(n+\theta_t)$ delay and $O(n + \theta_s)$ space.*

The paper is organized as follows. After making preparations in Sect. 2, we present an algorithm that enumerates all components that are subsets of a given component in a set system with SD property, along with complexity analyses in Sect. 3, as a proof for Theorem 3. Then in Sect. 4, we provide proofs for Theorems 1 and 2. There are two core parts in the proofs. In the first part, given a 2-edge-connected (resp., 2-vertex-connected) graph G, we show that a set system $(V, \mathcal{C})$ has SD property if $V = V(G)$ and $\mathcal{C}$ is the family of all vertex subsets that induce 2-edge-connected (resp., 2-vertex-connected) subgraphs. This means that 2-edge/vertex-connected induced subgraphs can be enumerated by using the algorithm developed for Theorem 3. Then in the second part, we explain how we design the oracle to achieve linear delay and space.

For some lemmas, we omit the proofs due to space limitation. The omitted proofs are found in the preprint of this paper [8].

2 Preliminaries

Let $\mathbb{Z}$ and $\mathbb{Z}_+$ denote the set of integers and the set of nonnegative integers, respectively. For two integers $i, j \in \mathbb{Z}$ $(i \leq j)$, let us denote $[i, j] := \{i, i + 1, \ldots, j\}$.

For any sets P, Q of elements, when $P \cap Q = \emptyset$, we may denote by $P \sqcup Q$ the disjoint union of P and Q in order to emphasize that they are disjoint.

Set Systems. Let $(V, \mathcal{C})$ be a set system which does not necessarily have SD property. For any two subsets $U, L \subseteq V$, we denote $\mathcal{C}(U, L) := \{C \in \mathcal{C} \mid L \subseteq C \subseteq U\}$. Recall that, for a component $C \in \mathcal{C}$, we denote by $\mathrm{MRS}_{\mathcal{C}}(C)$ the family of all MRSs of C. Further, for $X \subseteq C$, we denote $\mathrm{MRS}_{\mathcal{C}}(C, X) := \{Y \in \mathrm{MRS}_{\mathcal{C}}(C) \mid Y \cap X = \emptyset\}$. For any two components $C, C' \in \mathcal{C}$ with $C \supsetneq C'$, let $Y_1, Y_2, \ldots, Y_\ell \subseteq C \setminus C'$ be subsets such that $Y_i \cap Y_j = \emptyset$, $1 \leq i < j \leq \ell$; and $Y_1 \sqcup Y_2 \sqcup \cdots \sqcup Y_\ell = C \setminus C'$ (i.e., $\{Y_1, Y_2, \ldots, Y_\ell\}$ is a partition of $C \setminus C'$). Then $(Y_1, Y_2, \ldots, Y_\ell)$ is an *MRS-sequence* (*between C and C'*) if

- $C' \sqcup Y_1 \sqcup \cdots \sqcup Y_i \in \mathcal{C}$, $i \in [1, \ell]$; and
- $Y_i \in \mathrm{MRS}_{\mathcal{C}}(C' \sqcup Y_1 \sqcup \cdots \sqcup Y_i)$, $i \in [1, \ell]$.

One easily sees that there exists an MRS-sequence for every $C, C' \in \mathcal{C}$ such that $C \supsetneq C'$. The following lemma holds regardless of SD property.

Lemma 1. *For any set system $(V, \mathcal{C})$, let $C \in \mathcal{C}$ and $X \subseteq C$. It holds that $\mathrm{MRS}_{\mathcal{C}}(C, X) = \emptyset \iff \mathcal{C}(C, X) = \{C\}$.*

As we described in Sect. 1, we assume that a set system $(V, \mathcal{C})$ with SD property is given implicitly by an oracle. We denote by $\textsc{ComputeMrs}_{\mathcal{C}}$ the oracle. Given a component $C \in \mathcal{C}$ and a subset $X \subseteq C$ as a query to the oracle, $\textsc{ComputeMrs}_{\mathcal{C}}(C, X)$ returns one MRS in $\textsc{Mrs}_{\mathcal{C}}(C, X)$ if $\textsc{Mrs}_{\mathcal{C}}(C, X) \neq \emptyset$, and Nil otherwise, where we denote by θ_t and θ_s the time and space complexity, respectively.

The following lemma states a necessary condition of SD property which is not sufficient.

Lemma 2. *Suppose that a set system $(V, \mathcal{C})$ with SD property is given. For every component $C \in \mathcal{C}$, the minimal removable sets in $\textsc{Mrs}_{\mathcal{C}}(C)$ are pairwise disjoint.*

Graphs. Let G be a simple undirected graph. For a vertex $v \in V(G)$, we denote by $\deg_G(v)$ the degree of v in the graph G. We let $\delta(G) := \min_{v \in V(G)} \deg_G(v)$. Let $S \subseteq V(G)$ be a subset of vertices. A *subgraph induced by* S is a subgraph G' of G such that $V(G') = S$ and $E(G') = \{uv \in E(G) \mid u, v \in S\}$ and denoted by $G[S]$. For simplicity, we write the induced subgraph $G[V(G) \setminus S]$ as $G - S$. Similarly, for $F \subseteq E(G)$, we write as $G - F$ the subgraph whose vertex set is $V(G)$ and edge set is $E(G) \setminus F$.

A *cut-set of* G is a subset $F \subseteq E(G)$ such that $G - F$ is disconnected. In particular, we call an edge that constitutes a cut-set of size 1 a *bridge*. We define the *edge-connectivity* $\lambda(G)$ *of* G to be the cardinality of the minimum cut-set of G unless $|V(G)| = 1$. If $|V(G)| = 1$, then $\lambda(G)$ is defined to be ∞. G is called *k-edge-connected* if $\lambda(G) \geq k$. A *vertex cut* of G is a subset $S \subseteq V(G)$ such that $G - S$ is disconnected. In particular, we call a vertex cut whose size is two a *cut point pair* and a vertex that constitutes a singleton vertex cut an *articulation point*. For a subset $S \subseteq V(G)$, let $\textsc{Art}(S)$ denote the set of all articulation points in $G[S]$. We define the *vertex-connectivity* $\kappa(G)$ *of* G to be the cardinality of the minimum vertex cut of G unless G is a complete graph. If G is complete, then $\kappa(G)$ is defined to be $|V(G)| - 1$. G is called *k-vertex-connected* if $|V(G)| > k$ and $\kappa(G) \geq k$. Obviously, G is 2-edge-connected (resp., 2-vertex-connected) if and only if there is no bridge (resp., no articulation point) in G.

Proposition 1 ([15]). *Suppose that we are given a simple undirected graph G. If $|V(G)| \geq 2$, then it holds that $\kappa(G) \leq \lambda(G) \leq \delta(G)$.*

3 Enumerating Components in Set System with SD Property

In this section, we propose an algorithm that enumerates all components that are subsets of a given component in a set system $(V, \mathcal{C})$ with SD property and conduct complexity analyses, as a proof for Theorem 3.

Let us introduce mathematical foundations for a set system with SD property that are necessary for designing our enumeration algorithm.

Algorithm 1. An algorithm to enumerate all components in $\mathcal{C}(C, I)$, where $C \in \mathcal{C}$ is a component in a set system $(V, \mathcal{C})$ with SD property and I is a subset of C

Input: A component $C \in \mathcal{C}$ and a subset $I \subseteq C$
Output: All components in $\mathcal{C}(C, I)$
1: **procedure** List(C, I)
2: Output C;
3: $X \leftarrow I$;
4: **while** ComputeMrs$_{\mathcal{C}}(C, X) \neq$ Nil **do**
5: $Y \leftarrow$ ComputeMrs$_{\mathcal{C}}(C, X)$;
6: List$(C \setminus Y, X)$;
7: $X \leftarrow X \cup Y$
8: **end while**
9: **end procedure**

Lemma 3. *For a set system $(V, \mathcal{C})$ with SD property, let $C \in \mathcal{C}$ be a component and $I \subseteq C$ be a subset of C. For any $Y \in \mathrm{Mrs}_{\mathcal{C}}(C, I)$, it holds that $\mathcal{C}(C, I) = \mathcal{C}(C \setminus Y, I) \sqcup \mathcal{C}(C, I \sqcup Y)$.*

Lemma 4. *For a set system $(V, \mathcal{C})$ with SD property, let $C \in \mathcal{C}$ be a component, $I \subseteq C$ be a subset of C, and $\mathrm{Mrs}_{\mathcal{C}}(C, I) := \{Y_1, Y_2, \ldots, Y_k\}$. It holds that*

$$\mathcal{C}(C, I) = \{C\} \sqcup \Big(\bigsqcup_{i=1}^{k} \mathcal{C}(C \setminus Y_i, I \sqcup Y_1 \sqcup \cdots \sqcup Y_{i-1}) \Big).$$

Algorithm. Let $C \in \mathcal{C}$, $I \subseteq C$ and $\mathrm{Mrs}_{\mathcal{C}}(C, I) := \{Y_1, Y_2, \ldots, Y_k\}$. Lemma 4 describes a partition of $\mathcal{C}(C, I)$ such that there exists a similar partition for $\mathcal{C}(C_i, I_i)$, where $C_i = C \setminus Y_i$ and $I_i = I \sqcup Y_1 \sqcup \cdots \sqcup Y_{i-1}$, $i \in [1, k]$. Then we have an algorithm that enumerates all components in $\mathcal{C}(C, I)$ by outputting C and then outputting all components in $\mathcal{C}(C_i, I_i)$ for each $i \in [1, k]$ recursively.

For $C \in \mathcal{C}$ and $I \subseteq C$, Algorithm 1 summarizes a procedure to enumerate all components in $\mathcal{C}(C, I)$. Procedure List in Algorithm 1 outputs C in line 2 and computes $\mathcal{C}(C_i, I_i)$, $i \in [1, k]$ recursively in line 6. For our purpose, it suffices to invoke List$(C, \emptyset)$ to enumerate all components in $\mathcal{C}(C, \emptyset)$.

To analyze the complexities and prove Theorem 3, we introduce a detailed version of the algorithm in Algorithm 2. We mainly use a stack to store data, where we can add a given element (push), peek the element that is most recently added (last), remove the last element (pop), and shrink to a given size by removing elements that are most recently added (shrinkTo) in constant time, respectively, by using an array and managing an index to the last element.

In Algorithm 2, the set X of Algorithm 1 is realized by a stack. Note that X in Algorithm 2 however contains a subset of elements in each container. In addition, a stack *Seq* stores MRSs, an array *Siz* stores the number of the MRSs of a given component, and an integer *depth* represents the depth of recursion in Algorithm 1. We can see that the algorithm enumerates all components that

are subsets of C in $\mathcal{C}$ by Lemma 4. To bound the time complexity by that for processing a node in the search tree, we apply the *alternative method* to our algorithm in line 4–6 and 12–14 and reduce the delay [11]. We next discuss the space complexity. We see that *Seq* stores MRS-sequence between the current component C' and C since *Seq* pops the MRS after traversing all children of C', and so it consumes $O(n)$ space. The stack X uses $O(n)$ space by Lemma 2 and the definition of ComputeMrs$_{\mathcal{C}}$, and moreover *Siz* uses only $O(n)$ space since the maximum depth is n.

Proof for Theorem 3. We can see that Algorithm 2 surely enumerates all components in $\mathcal{C}(C, \emptyset)$ by Lemma 4. We first prove that Algorithm 2 works in $O(n + \theta_t)$ delay. If depth is odd, then the current component C' is outputted. If depth is even and $\textsc{Mrs}_{\mathcal{C}}(C', X) = \emptyset$, then C' is outputted. If depth is even and $\textsc{Mrs}_{\mathcal{C}}(C', X) \neq \emptyset$, then $C' \setminus Y$ will be outputted for Y in line 8 in the next iteration. Then it suffices to show that operations from line 4 to 19 can be done in $O(n + \theta_t)$ time. A component can be outputted in $O(n)$ time. The difference can be traced by subtracting and adding an MRS before and after the depth changes, thus it takes $O(n)$ time. In addition, ComputeMrs$_{\mathcal{C}}$ works in θ_t time by definition and another operations are adding and subtracting an MRS, where computation time is $O(n)$.

We next discuss the space complexity of Algorithm 2. The maximum size of the depth is n since the size of the component C' is monotonically decreasing while the depth increases and the termination condition that $\textsc{Mrs}_{\mathcal{C}}(C', X)$ is initially empty is satisfied at most n depth. The rest to show is that the space for *Seq*, X, and *Siz* are $O(n)$. For a component $C' \in \mathcal{C}(C, \emptyset)$, we obtain that the *Seq* is equivalent to the MRS-sequence between C' and C since *Seq* store a new MRS before the depth increases and discards before the depth decreases. X can be hold in $O(n)$ space since for any subset $I \subseteq C'$, I and $\textsc{Mrs}_{\mathcal{C}}(C', I)$ are pairwise disjoint. It is obvious that *Siz* uses $O(n)$ space since the maximum depth is n. We then obtain that whole space is $O(n + \theta_s)$ space. □

4 Enumerating 2-Edge/Vertex-Connected Induced Subgraphs

In this section, we provide proofs for Theorems 1 and 2. Suppose that we are given a simple undirected graph G that is 2-edge-connected (resp., 2-vertex-connected). In Sect. 4.1, we show that a set system $(V, \mathcal{C})$ has SD property if $V = V(G)$ and $\mathcal{C}$ is the family of all vertex subsets that induce 2-edge-connected subgraphs (resp., 2-vertex-connected subgraphs). This indicates that all 2-edge/vertex-connected induced subgraphs can be enumerated by the algorithm in the last section. Then in Sect. 4.2, we show how to design the oracle for generating an MRS so that the required computational complexity is achieved.

Algorithm 2. An algorithm to enumerate all components that are subsets of $C \in \mathcal{C}$ in $(V, \mathcal{C})$ with SD property

Input: A set system $(V, \mathcal{C})$ with SD property and a component $C \in \mathcal{C}$
Output: All components that are subsets of C in $\mathcal{C}$

```
 1: Seq, X ← empty stack; Siz ← an array of length n, filled by 0;
 2: C' ← C; depth ← 1;
 3: while depth ≠ 0 do
 4:     if depth is odd then
 5:         Output C'
 6:     end if;
 7:     if ComputeMrs_C(C', X) ≠ Nil then                ▷ emulate recursive call
 8:         Y ← ComputeMrs_C(C', X);
 9:         Seq.push(Y); C' ← C' \ Y; Siz[depth]← Siz[depth]+1;
10:         depth ← depth + 1
11:     else                                             ▷ trace back
12:         if depth is even then
13:             Output C'
14:         end if;
15:         C' ← C' ∪ Seq.last();
16:         X.shrinkTo(X.length() −Siz[depth]); Siz[depth]← 0;
17:         X.push(Seq.last());
18:         Seq.pop(); depth ← depth − 1
19:     end if
20: end while
```

4.1 Constructing Set Systems with SD Property

We define $\mathcal{C}_e \triangleq \{C \subseteq V(G) \mid G[C]$ is 2-edge-connected and $|C| > 1\}$ and $\mathcal{C}_v \triangleq \{C \subseteq V(G) \mid G[C]$ is 2-vertex-connected$\}$. For $S \subseteq V(G)$, we call S an *e-component* (resp., a *v-component*) if $S \in \mathcal{C}_e$ (resp., $S \in \mathcal{C}_v$). We deal with a set system $(V, \mathcal{C})$ such that $V = V(G)$ and $\mathcal{C} \in \{\mathcal{C}_e, \mathcal{C}_v\}$. We use the notations and terminologies for set systems that were introduced in Sect. 2 to discuss the problem of enumerating all e-components or v-components. Although a singleton is a 2-edge-connected component by definition, we do not regard it as an e-component since otherwise the system $(V, \mathcal{C}_e)$ would not have SD property.

The following lemma is immediate by Proposition 1.

Lemma 5. *For a simple undirected graph G, it holds that $\mathcal{C}_v \subseteq \mathcal{C}_e$.*

By Lemma 5, every v-component is an e-component. Let $\textsc{Max}_e(S)$ (resp., $\textsc{Max}_v(S)$) denote the family of all maximal e-components (resp., v-components) among the subsets of S. For an e-component $C \in \mathcal{C}_e$ (resp., a v-component $C \in \mathcal{C}_v$), we write the family $\textsc{Mrs}_{\mathcal{C}_e}(C)$ (resp., $\textsc{Mrs}_{\mathcal{C}_v}(C)$) of all minimal removable sets of C as $\textsc{Mrs}_e(C)$ (resp., $\textsc{Mrs}_v(C)$) for simplicity. We call an MRS in $\textsc{Mrs}_e(C)$ (resp., $\textsc{Mrs}_v(C)$) an *e-MRS of C* (resp., a *v-MRS of C*). A minimal e-component $C \in \mathcal{C}_e$ induces a cycle (i.e., $G[C]$ is a cycle) since no singleton is contained in $\mathcal{C}_e$, and in this case, it holds that $\textsc{Mrs}_e(C) = \emptyset$.

A *block* of a simple undirected graph G is a maximal connected subgraph that has no articulation point. Every block of G is an isolated vertex; a cut-edge (i.e., an edge whose removal increases the number of connected components); or a maximal v-component; e.g., see Remark 4.1.18 in [14]. The following lemma is immediate.

Lemma 6. *For a given simple undirected graph G and an e-component $C \in \mathcal{C}_e$, it holds that $C = \bigcup_{H \in \mathrm{MAX}_v(C)} H$.*

For $P \subseteq S \subseteq V(G)$, P is called a *two-deg path in $G[S]$* (or *in S* for short) if $\deg_{G[S]}(u) = 2$ holds for every $u \in P$. In particular, P is a *maximal two-deg path in S* if there is no two-deg path P' in S such that $P \subsetneq P'$. It is possible that a maximal two-deg path consists of just one vertex. For an e-component $C \in \mathcal{C}_e$, we denote by $\mathrm{CAN}_{=2}(C)$ the family of all maximal two-deg paths in C. We also denote $\mathrm{CAN}_{>2}(C) := \{\{v\} \mid v \in C, \deg_{G[C]}(v) > 2\}$ and define $\mathrm{CAN}(C) \triangleq \mathrm{CAN}_{=2}(C) \sqcup \mathrm{CAN}_{>2}(C)$. It is clear that every vertex in C belongs to either a maximal two-deg path in $\mathrm{CAN}_{=2}(C)$ or a singleton in $\mathrm{CAN}_{>2}(C)$, where there is no vertex $v \in C$ such that $\deg_{G[C]}(v) \leq 1$ since $G[C]$ is 2-edge-connected.

The following Lemma 7 states that an e-MRS of an e-component C is either a maximal two-deg path in C or a single vertex whose degree in $G[C]$ is more than two.

Lemma 7 (Observation 3 in [7]). *For a simple undirected graph G, let $C \in \mathcal{C}_e$. It holds that $\mathrm{MRS}_e(C) = \{Y \in \mathrm{CAN}(C) \mid C \setminus Y \in \mathcal{C}_e\}$.*

If G is 2-edge-connected, then the set system $(V(G), \mathcal{C}_e)$ has SD property, as shown in the following Lemma 8.

Lemma 8. *For a simple undirected 2-edge-connected graph G, the set system $(V(G), \mathcal{C}_e)$ has SD property.*

Proof. We see that $V(G) \in \mathcal{C}_e$ since G is 2-edge-connected. Let $C, C' \in \mathcal{C}_e$ be e-components such that $C \supsetneq C'$ and $Y \in \mathrm{MRS}_e(C)$ be an e-MRS of C. We show that either $Y \subseteq C'$ or $Y \cap C' = \emptyset$ holds. The case of $|Y| = 1$ is obvious. Suppose $|Y| > 1$. By Lemma 7, Y induces a maximal two-deg path in $G[C]$ such that for any $u \in Y$ it holds $\deg_{G[C]}(u) = 2$. If $Y \not\subseteq C'$ and $Y \cap C' \neq \emptyset$, then there would be two adjacent vertices $v, v' \in Y$ such that $v \in C \setminus C'$ and $v' \in C'$, where we see that $\deg_{G[C']}(v') \leq 1$ holds. The C' is an e-component and thus $|C'| \geq 2$. By Proposition 1, we obtain $1 \geq \delta(G[C']) \geq \lambda(G[C'])$, which contradicts that $C' \in \mathcal{C}_e$. □

We can derive analogous results for $(V(G), \mathcal{C}_v)$. In Lemma 9, we show that a v-MRS of a v-component C is either a maximal two-deg path in C or a single vertex whose degree in $G[C]$ is more than two. Then in Lemma 10, we show that $(V(G), \mathcal{C}_c)$ has SD property when G is 2-vertex-connected.

Lemma 9. *For a simple undirected graph G, let $C \in \mathcal{C}_v$. It holds that $\mathrm{MRS}_v(C) = \{Y \in \mathrm{CAN}(C) \mid C \setminus Y \in \mathcal{C}_v\}$.*

Lemma 10. *For a simple undirected 2-vertex-connected graph G, the set system $(V(G), \mathcal{C}_v)$ has SD property.*

4.2 Computing MRSs in Linear Time and Space

Let G be a simple undirected graph. We describe how we compute an e-MRS of an e-component in linear time and space. Specifically, for a given e-component $C \in \mathcal{C}_e$ and subset $X \subseteq C$, we design the oracle $\textsc{ComputeMrs}_{\mathcal{C}_e}(C, X)$ so that it outputs *one* e-MRS $Y \in \textsc{Mrs}_e(C, X)$ if $\textsc{Mrs}_e(C, X) \neq \emptyset$, and $\textsc{Nil}$ otherwise, in linear time and space. In what follows, we derive a stronger result that *all* e-MRSs in $\textsc{Mrs}_e(C, X)$ can be enumerated in linear delay and space.

The scenario of the proof is as follows.

(1) We show that, to enumerate e-MRSs in $\textsc{Mrs}_e(C, X)$, it suffices to examine $\textsc{Mrs}_e(S, X)$ for each $S \in \textsc{Max}_v(C)$ respectively. This indicates that we may assume C to be a v-component. It is summarized as Corollary 1, followed by Lemma 11.

(2) Using a certain auxiliary graph, we show that it is possible to output in linear time and space all candidates in $\textsc{Can}(C)$ that are e-MRSs of C (Lemma 13); recall that all candidates of e-MRSs are contained in $\textsc{Can}(\mathrm{C})$ by Lemma 7.

The case of computing a v-MRS of a v-component can be done almost analogously.

Lemma 11. *Given a simple undirected 2-edge-connected graph G that is neither a cycle nor a single vertex, let $V := V(G)$ and $Y \subsetneq V$ be any nonempty proper subset of V. Then Y is an e-MRS of V if and only if there is $S \in \textsc{Max}_v(V)$ such that*

(i) *$Y \cap S' = \emptyset$ holds for every $S' \in \textsc{Max}_v(V)$ such that $S' \neq S$; and*
(ii) *Y is either a path that consists of all vertices in S except one or an e-MRS of S.*

Proof. For the necessity, every v-component $S \in \textsc{Max}_v(V)$ is an e-component. By the definition of SD property, either $Y \subsetneq S$ or $Y \cap S = \emptyset$ should hold. Suppose that there are two distinct v-components $S, S' \in \textsc{Max}_v(V)$ such that $Y \subsetneq S$ and $Y \subsetneq S'$. This leads to $|Y| = 1$ since $1 \geq |S \cap S'| \geq |Y| \geq 1$, where the first inequality holds by the fact that two blocks share at most one vertex (e.g., Proposition 4.1.19 in [14]). Then Y is a singleton that consists of an articulation point of G, contradicting that $V \setminus Y$ is connected. There is at most one $S \in \textsc{Max}_v(V)$ that contains Y as a proper subset, and such S surely exists since there is at least one v-component in $\textsc{Max}_v(V)$ that intersects Y by Lemma 6, which shows (i).

To show (ii), suppose that $G[S]$ is a cycle. Then it holds that $|S \cap \textsc{Art}(V)| = 1$; if $|S \cap \textsc{Art}(V)| \geq 2$, then no singleton or path in S is an e-MRS of V, contradicting that $Y \subsetneq S$; if $|S \cap \textsc{Art}(V)| = 0$, then S is a connected component in G. This contradicts that G is connected since G is not a cycle and hence $S \subsetneq V$. Then Y should be the path in S that consists of all vertices except the only articulation point. Suppose that $G[S]$ is not a cycle. Let $u, v \in S$ be two distinct vertices. We claim that every path between u and v should not visit a vertex out of S; if there is such a path, then the union of v-components visited

by the path would be a v-component containing S, contradicting the maximality of S. In the graph $G-Y$, there are at least two edge-disjoint paths between any two vertices $u, v \in S-Y$. These paths do not visit any vertex out of S, and thus $S-Y$ is an e-component. It is easy to see that $Y \in \textsc{Can}(S)$.

For the sufficiency, suppose that $G[S]$ is a cycle. There is no e-MRS of S by definition. The set Y should be a path that consists of all vertices in S except one, and by (i), the vertex that is not contained in Y should be an articulation point (which implies $|S \cap \textsc{Art}(V)| = 1$). Suppose that $G[S]$ is not a cycle. An e-MRS of S exists since S is a non-minimal e-component. Let Y be an e-MRS of S that satisfies (i), that is, Y contains no articulation points in $\textsc{Art}(V)$. In either case, it is easy to see that $V \setminus Y$ is an e-component and that $Y \in \textsc{Can}(V)$ holds, showing that Y is an e-MRS of V. □

Corollary 1. *For a given simple undirected graph G, let $C \in \mathcal{C}_e$ be an e-component. Then it holds that*

$$\textsc{Mrs}_e(C) = \Big(\bigsqcup_{S \in \textsc{Max}_v(C):\ G[S] \text{ is not a cycle}} \{Y \in \textsc{Mrs}_e(S) \mid Y \cap \textsc{Art}(C) = \emptyset\} \Big) \sqcup \Big(\bigsqcup_{S \in \textsc{Max}_v(C):\ G[S] \text{ is a cycle and } |S \cap \textsc{Art}(C)|=1} (S \setminus \textsc{Art}(C)) \Big).$$

By the corollary, to obtain e-MRSs of an e-component C, it suffices to examine all maximal v-components in $\textsc{Max}_v(C)$ respectively.

We observe the first family in the right hand in Corollary 1. Let C be a v-component such that $G[C]$ is not a cycle. For each path $P \in \textsc{Can}_{=2}(C)$, there are exactly two vertices $u, v \in \textsc{Can}_{>2}(C)$ such that u is adjacent to one endpoint of P and v is adjacent to the other endpoint of P. We call such u, v *boundaries of* P. We denote the pair of boundaries of P by $B(P)$, that is, $B(P) := \{u, v\}$. We define $\Lambda_{>2}(C) \triangleq \{uv \in E(G) \mid u, v \in \textsc{Can}_{>2}(C)\}$. Let $\Lambda(C) := \textsc{Can}_{=2}(C) \sqcup \Lambda_{>2}(C)$. We then define an auxiliary graph H_C so that

$$\begin{aligned} V(H_C) &:= \textsc{Can}_{>2}(C) \sqcup \textsc{Can}_{=2}(C) \sqcup \Lambda_{>2}(C) \\ &= \textsc{Can}_{>2}(C) \sqcup \Lambda(C) = \textsc{Can}(C) \sqcup \Lambda_{>2}(C), \\ E(H_C) &:= \{uP \subseteq V(H_C) \mid u \in \textsc{Can}_{>2}(C),\ P \in \textsc{Can}_{=2}(C),\ u \in B(P)\} \\ &\quad \sqcup \{ue \subseteq V(H_C) \mid u \in \textsc{Can}_{>2}(C),\ e \in \Lambda_{>2}(C),\ u \in e\}. \end{aligned}$$

We call a vertex in $\textsc{Can}_{>2}(C)$ an *ordinary* vertex, whereas we call a vertex in $\Lambda(C) = \textsc{Can}_{=2}(C) \sqcup \Lambda_{>2}(C)$ an *auxiliary* vertex.

For $P \in \textsc{Can}_{=2}(C)$, we denote by $E(P)$ the set of all edges in the path $P \cup B(P)$. For $e \in \Lambda_{>2}(C)$, we denote $E(e) := \{e\}$. We see that $E(G[C]) = \bigsqcup_{h \in \Lambda(C)} E(h)$ holds.

Lemma 12. *Given a simple undirected graph G, let $C \in \mathcal{C}_v$ be a v-component such that $G[C]$ is not a cycle and $Y \in \textsc{Can}(C)$. Then $Y \in \textsc{Mrs}_e(C)$ holds if and only if there is no auxiliary vertex $h \in \Lambda(C)$ such that $\{Y, h\}$ is a cut point pair of H_C.*

Proof. For the necessity, suppose that there is $h \in \Lambda(C)$ such that $\{Y, h\}$ is a cut point pair of H_C. Then h is an articulation point of $H_C - Y$. Every edge $e \in E(h)$ is a bridge in $G[C] - Y$, indicating that $G[C] - Y$ is not 2-edge-connected, and hence $Y \notin \textsc{Mrs}_e(C)$.

For the sufficiency, suppose that $Y \notin \textsc{Mrs}_e(C)$. Then $G[C] - Y$ is not 2-edge-connected but should be connected. There exists a bridge, say e, in $G[C] - Y$. Let $h \in \Lambda(C)$ be the auxiliary vertex such that $e \in E(h)$. We see that h is a cut point of $H_C - Y$, indicating that $\{Y, h\}$ is a cut point pair of H_C. □

Lemma 13. *Suppose that a simple undirected 2-edge-connected graph G is given. Let $V := V(G)$. For any subset $X \subseteq V$, all e-MRSs in $\textsc{Mrs}_e(V, X)$ can be enumerated in $O(n + m)$ time and space.*

Proof. We can complete the required task as follows. (1) We obtain $\textsc{Art}(V)$ and decompose V into maximal v-components. For each maximal v-component C, (2) if $G[C]$ is a cycle and $|C \cap \textsc{Art}(V)| = 1$, then output $C \setminus \textsc{Art}(V)$ if it is disjoint with X; and (3) if $G[C]$ is not a cycle, then we construct an auxiliary graph H_C, compute all cut point pairs of H_C, and output all $Y \in \textsc{Can}(C)$ that are disjoint with $X \cup \textsc{Art}(V)$ and that are not contained in any cut point pair together with an auxiliary vertex. The correctness of the algorithm follows by Corollary 1 and Lemma 12.

For the time complexity, (1) can be done in $O(n + m)$ time [9]. For each $C \in \textsc{Max}_v(V)$, let $n_C := |C|$ and $m_C := |E(G[C])|$. We can decide in $O(n_C + m_C)$ time whether C is in (2), (3) or neither of them. If we are in (2), then the task can be done in $O(n_C)$ time. If we are in (3), then the task can be done in $O(n_C + m_C)$ time since H_C can be constructed in linear time and all cut point pairs of a 2-vertex-connected graph H_C can be enumerated in linear time [4,6]. An articulation point v appears in at most $\deg_G(v)$ maximal v-components, and hence $\sum_{C \in \textsc{Max}_v(V)} O(n_C) = O(n+m)$. The number of maximal v-components is $O(n)$, and the overall time complexity over $C \in \textsc{Max}_v(V)$ is $O(n) + \sum_{C \in \textsc{Max}_v(V)} O(n_C + m_C) = O(n + m)$. The space complexity analysis is analogous. □

Proofs for Theorems 1 and 2. For Theorem 1, we see that $\mathcal{C}_e = \bigsqcup_{S \in \textsc{Max}_e(V)} \mathcal{C}_e(S, \emptyset)$. We can enumerate all maximal e-components in $\textsc{Max}_e(V)$ in $O(n+m)$ time and space, by removing all bridges in G [10]. All e-components in $\mathcal{C}_e(S, \emptyset)$ for each $S \in \textsc{Max}_e(V)$ can be enumerated in $O(n + \theta_t)$ delay and in $O(n + \theta_s)$ space by Theorem 3. We can implement $\textsc{ComputeMrs}_{\mathcal{C}_e}$ so that $\theta_t = O(n + m)$ and $\theta_s = O(n + m)$ by Lemma 13. Theorem 2 is analogous. □

Concluding Remarks. The future work includes extension of our framework to k-edge/vertex-connectivity for $k > 2$; and studying relationship between SD property and set systems known in the literature (e.g., independent system, accessible system, strongly accessible system, confluent system).

References

1. Avis, D., Fukuda, K.: Reverse search for enumeration. Discret. Appl. Math. **63**(1–3), 21–46 (1996). https://doi.org/10.1016/0166-218X(95)00026-N
2. Chang, L., Yu, J.X., Qin, L.: Fast maximal cliques enumeration in sparse graphs. Algorithmica **66**(1), 173–186 (2013). https://doi.org/10.1007/s00453-012-9632-8
3. Conte, A., Grossi, R., Marino, A., Versari, L.: Sublinear-space and bounded-delay algorithms for maximal clique enumeration in graphs. Algorithmica **82**(6), 1547–1573 (2019). https://doi.org/10.1007/s00453-019-00656-8
4. Gutwenger, C., Mutzel, P.: A linear time implementation of SPQR-trees. In: Marks, J. (ed.) GD 2000. LNCS, vol. 1984, pp. 77–90. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44541-2_8
5. Haraguchi, K., Nagamochi, H.: Enumeration of support-closed subsets in confluent systems. Algorithmica **84**, 1279–1315 (2022). https://doi.org/10.1007/s00453-022-00927-x
6. Hopcroft, J.E., Tarjan, R.E.: Dividing a graph into triconnected components. SIAM J. Comput. **2**(3), 135–158 (1973). https://doi.org/10.1137/0202012
7. Ito, Y., Sano, Y., Yamanaka, K., Hirayama, T.: A polynomial delay algorithm for enumerating 2-edge-connected induced subgraphs. IEICE Trans. Inf. Syst. **E105.D**(3), 466–473 (2022). https://doi.org/10.1587/transinf.2021FCP0005
8. Tada, T., Haraguchi, K.: A linear delay algorithm for enumeration of 2-edge/vertex-connected induced subgraphs. arXiv cs.DS, 2302.05526 (2023). https://doi.org/10.48550/arXiv.2302.05526
9. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972). https://doi.org/10.1137/0201010
10. Tarjan, R.: A note on finding the bridges of a graph. Inf. Process. Lett. **2**(6), 160–161 (1974)
11. Uno, T.: Two general methods to reduce delay and change of enumeration algorithms. NII Technical Reports (2003)
12. Uno, T.: An efficient algorithm for solving pseudo clique enumeration problem. Algorithmica **56**, 3–16 (2010). https://doi.org/10.1007/s00453-008-9238-3
13. Wen, D., Qin, D., Zhang, Y., Chang, L., Chen, L.: Enumerating k-vertex connected components in large graphs. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), Macao, China, pp. 52–63 (2019). https://doi.org/10.1109/ICDE.2019.00014
14. West, D.B.: Introduction to Graph Theory, 2nd edn. Pearson Modern Classic (2018)
15. Whitney, H.: Congruent graphs and the connectivity of graphs. Amer. J. Math. **54**, 150–168 (1932)