Vol.2018-AL-168 No.8
2018/5/26

COOMA: A Components Overlaid Mining Algorithm for Enumerating Connected Subgraphs with Common Itemsets

Kazuya Haraguchi[1,a)] Yusuke Momoi[2,b)] Aleksandar Shurbevski[3,c)] Hiroshi Nagamochi[3,d)]

Abstract: Consider a data set that is represented by a tuple (G, I, σ) of a graph $G = (V, E)$, an item set I, and a function $\sigma : V \to 2^I$, and denote by $A_\sigma(X)$, $X \subseteq V$, the set of items common to $\sigma(v)$ for all vertices $v \in X$, i.e., $A_\sigma(X) = \bigcap_{v \in X} \sigma(v)$. A vertex subset $X \subseteq V$ is called a *connector* if (i) the subgraph $G[X]$ induced from G by X is connected; and (ii) adding any vertex $v \in V \setminus X$ to X loses the connectivity of the subgraph or decreases the common item set, i.e., $G[X \cup \{v\}]$ is disconnected or $|A_\sigma(X \cup \{v\})| < |A_\sigma(X)|$. In the present paper, we propose a novel algorithm named *COOMA* (*components overlaid mining algorithm*) for enumerating all connectors. For every item, COOMA enumerates connectors by "overlaying" an already discovered component on a subgraph induced by the item. We show that COOMA is more efficient than a depth-first-search based algorithm, COPINE [Sese et al., 2010], especially for instances that have a large number of connectors (e.g., more than 10^6), where we use real genetic networks as well as random instances for the benchmark. We also discuss the enumeration problems under various conditions, e.g., $|A_\sigma(X)|$ should be no less than an input parameter θ, and how to adapt COOMA to the variants.

1. Introduction

Many existing data are stored in the form of a graph [5]. In graph data, a vertex is often associated with items. For example, in a social network, each vertex corresponds to a user and two users are joined by an edge if they are friends. A user may be associated with products that he or she has purchased so far. In a genetic network, each vertex may correspond to an SNP (single nucleotide polymorphism), and two SNPs are joined by an edge if they have significant relationship in some context. An SNP may be associated with individuals that possess the SNP [14].

We consider a data mining problem for such graph. Suppose that we are given a tuple (G, I, σ) of a graph $G = (V, E)$, an item set I, and a function $\sigma : V \to 2^I$. For each vertex $v \in V$, the subset $\sigma(v)$ represents the set of items with which v is associated. For $X \subseteq V$, we denote by $A_\sigma(X)$ the set of items common to $\sigma(v)$ for all vertices $v \in X$, i.e., $A_\sigma(X) = \bigcap_{v \in X} \sigma(v)$. A vertex subset X is called a *connector* if (i) the subgraph $G[X]$ induced from G by X is connected; and (ii) adding any vertex $v \in V \setminus X$ to X loses the connectivity of the subgraph or decreases the common item set, i.e., $G[X \cup \{v\}]$ is disconnected or $|A_\sigma(X \cup \{v\})| < |A_\sigma(X)|$.

In the context of social networks, a connector X may represent a maximal subset of users such that any two of them are connected by a sequence of individuals in the set who are pairwise friends, and that all of them have purchased the products in $A_\sigma(X)$. It should be meaningful to obtain connectors in terms of marketing. For example, we may recommend a product i in $A_\sigma(X)$ to a user u who is not in X but has a friend in X, expecting that u may like i and thus buy it.

We consider the problem of enumerating all connectors for a given instance (G, I, σ). This problem was first introduced for biological networks and an algorithm named COPINE was proposed [9], [10]. Okuno studied parallelization of COPINE [8]. COPINE is a straightforward algorithm in some sense. Based on gSpan [13], the algorithm traverses a search tree in the depth-first manner.

We claim that there should be room for exploring better algorithms. For enumeration problems, several algorithmic frameworks have been invented so far; e.g., reverse search [2], BDD/ZDD [6] and dynamic programming [3]. These frameworks have been applied to various enumeration problems [4], [7], [12]. COPINE is not the only algorithmic solution to our problem. We may develop other enumeration algorithms, aiming at a better graph mining tool for practitioners.

With this in mind, we propose a novel enumeration algorithm named COOMA, which stands for a components overlaid mining algorithm. The interesting feature of COOMA is that it does not utilize the conventional algorithmic frameworks mentioned above. For every item, COOMA enumerates connectors by "overlaying" an already discovered connector on a subgraph induced by the item.

[1] Faculty of Commerce, Otaru University of Commerce
[2] School of Informatics and Mathematical Science, Faculty of Engineering, Kyoto University
[3] Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University
[a)] haraguchi@res.otaru-uc.ac.jp
[b)] y.momoi@amp.i.kyoto-u.ac.jp
[c)] shurbevski@amp.i.kyoto-u.ac.jp
[d)] nag@amp.i.kyoto-u.ac.jp

For the problem of enumerating connectors, various extensions are possible. In fact, COPINE is originally developed for the problem of enumerating all connectors X such that $|A_\sigma(X)| \geq \theta$, where θ is an input parameter. We may also restrict the size of a desired connector to no less than an input parameter, supposing that a connector of too small a size is meaningless. We will discuss how to adapt COOMA to these extensions. For future work, we may deal with vertex-weighted and/or edge-weighted graphs. The edge density may be taken into account in the definition of desired subgraphs, as in [1], [11].

The organization of the paper is as follows. For preliminaries, we introduce notation and terminologies and formalize the enumeration problem in Section 2. We then propose COOMA in Section 3. In Section 4, we discuss three points on this new enumeration algorithm. First, we describe the difference between COOMA and COPINE in how connectors are enumerated. Second, we propose a sophisticated version of the algorithm, R-COOMA, based on reorganization of (I, σ). Third, we show useful techniques that reduce the input size by preprocessing. In Section 5, we compare the enumeration algorithms in terms of computation time that is taken to enumerate all target subgraphs. We show the efficiency of COOMA and R-COOMA, especially for instances that have a large number of connectors (e.g., more than 10^6). We use real genetic networks as well as random instances for the benchmark. Finally we give concluding remarks in Section 6.

2. Preliminaries

A graph stands for a simple undirected graph in this paper. The vertex set (resp., edge set) of a graph H is denoted by $V(H)$ (resp., $E(H)$).

Let $G = (V, E)$ be a graph with a vertex set V and an edge set E. For a vertex $v \in V$, let $N_G(v)$ denote the set $\{u \in V \mid uv \in E\}$ of neighbors of v in G. Let X be a subset of V. Define $N_G(X)$ to be the set $\cup_{v \in X} N_G(v) \setminus X$ of neighbors of X in G. Let F be a subset of E. Define $X[F]$ to be the set of vertices $x \in X$ such that x is an end-vertex of an edge in F, $F[X]$ to be the set of edges $e = uv \in F$ with $u, v \in X$, and $G[X]$ (resp., $G[X, F]$) be the subgraph $(X, E[X])$ (resp., $(X[F], F[X])$). A vertex subset Z of a graph H is called a *component* of H if $H[Z]$ is connected and $H[Z \cup \{v\}]$ is not connected for any vertex $v \in V(H) \setminus Z$. Let $\mathcal{C}(X, F)$ denote the family of all components of the graph $G[X, F]$.

For the example in Figure 1, let us take $X = \{v_1, v_2, v_3, v_6, v_9\}$. Then $E[X]$, the edge set of $G[X]$, is $\{v_1v_2, v_2v_3, v_2v_6, v_2v_9, v_3v_6, v_6v_9\}$. For an edge set $F = \{v_2v_6, v_2v_9, v_5v_9\}$, we have $X[F] = \{v_2, v_6, v_9\}$ and $F[X] = \{v_2v_6, v_2v_9\}$. The subgraph $G[X, F]$ has just one component.

For a set I of items, let σ be a function $\sigma : V \to 2^I$. Let X be a non-empty subset of V. The *item set* $A_\sigma(X)$ of X is defined to be $A_\sigma(X) \triangleq \bigcap_{v \in X} \sigma(v)$.

In our problem, an instance is given by a triple (G, I, σ). We call $X \subseteq V$ a *connector* of (G, I, σ) if;

- $G[X]$ is connected; and
- for each vertex $v \in V \setminus X$, $G[X \cup \{v\}]$ is not connected or $A_\sigma(X) \setminus \sigma(v) \neq \emptyset$ (i.e., $A_\sigma(X) \supsetneq A_\sigma(X \cup \{v\})$).

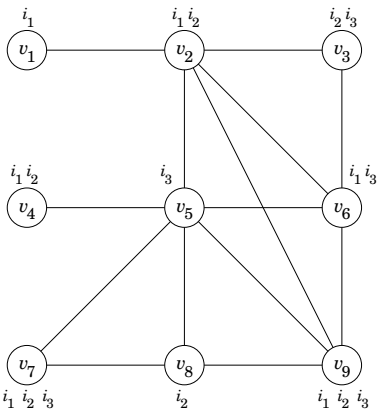

Fig. 1 An example of (G, I, σ)

Table 1 All connectors X and their item sets $A_\sigma(X)$ of the instance given by Figure 1

X	$A_\sigma(X)$
$\{v_1, v_2, v_6, v_9\}$	$\{i_1\}$
$\{v_2, v_3, v_7, v_8, v_9\}$	$\{i_2\}$
$\{v_3, v_5, v_6, v_7, v_9\}$	$\{i_3\}$
$\{v_4\}$	$\{i_1, i_2\}$
$\{v_2, v_9\}$	$\{i_1, i_2\}$
$\{v_6, v_9\}$	$\{i_1, i_3\}$
$\{v_3\}$	$\{i_2, i_3\}$
$\{v_7\}$	$\{i_1, i_2, i_3\}$
$\{v_9\}$	$\{i_1, i_2, i_3\}$

We call a connector X a *θ-connector* for an integer $\theta \geq 0$ if $|A_\sigma(X)| \geq \theta$. Let $\mathcal{M}(G, I, \sigma)$ denote the family of all connectors of (G, I, σ). Let $\mathcal{M}_\theta(G, I, \sigma)$ denote the family of all θ-connectors of (G, σ).

Table 1 shows all connectors and their item sets of the instance (G, I, σ) given by Figure 1.

Now we summarize the problem of enumerating all θ-connectors as follows.

Connector Enumeration Problem
Input: An instance (G, I, σ) that consists of a graph $G = (V, E)$, an item set I and a function $\sigma : V \to 2^I$, and an integer $\theta \geq 0$.
Output: $\mathcal{M}_\theta(G, I, \sigma)$.

For a non-empty subset $J \subseteq I$, define $V_{\langle J \rangle} \triangleq \{v \in V \mid J \subseteq \sigma(v)\}$, and $E_{\langle J \rangle} \triangleq \{uv \in E \mid J \subseteq \sigma(u) \cap \sigma(v)\}$. For $J = \{j\}$, we may denote $V_{\langle J \rangle}$ and $E_{\langle J \rangle}$ by $V_{\langle j \rangle}$ and $E_{\langle j \rangle}$, respectively. Let $\mathcal{C}_{G,I,\sigma}$ denote the union of all components in $G[V_{\langle i \rangle}]$, $i \in I$, i.e., $\mathcal{C}_{G,I,\sigma} = \bigcup_{i \in I} \mathcal{C}(V_{\langle i \rangle}, E)$. In Figure 1,

$$\mathcal{C}(V_{\langle i_1 \rangle}, E) = \{\{v_1, v_2, v_6, v_9\}, \{v_4\}, \{v_7\}\},$$
$$\mathcal{C}(V_{\langle i_2 \rangle}, E) = \{\{v_2, v_3, v_7, v_8, v_9\}, \{v_4\}\},$$
$$\mathcal{C}(V_{\langle i_3 \rangle}, E) = \{\{v_3, v_5, v_6, v_7, v_9\}\},$$

and $\mathcal{C}_{G,I,\sigma}$ is the union of these three collections.

We call a subfamily $\mathcal{M}' \subseteq \mathcal{M}(G, I, \sigma)$ *self-contained* if for every two sets $X \in \mathcal{M}'$ and $C \in \mathcal{M}' \cap \mathcal{C}_{G,I,\sigma}$, it holds that $\mathcal{C}(X \cap C, E) \subseteq \mathcal{M}'$. This means that, if $\mathcal{M}'$ is self-contained, then the intersection $\mathcal{M}' \cap \mathcal{C}_{G,I,\sigma}$ is empty, or for any minimal connector $X \in \mathcal{M}' \setminus \mathcal{C}_{G,I,\sigma}$, there is a connector $C \in \mathcal{M}' \cap \mathcal{C}_{G,I,\sigma}$ such that $X \subseteq C$. For the instance in Figure 1,

$\mathcal{M}' = \{\{v_1, v_2, v_6, v_9\}, \{v_3, v_5, v_6, v_7, v_9\}, \{v_6, v_9\}\}$ is self-contained. On the other hand, $\mathcal{M}' = \{\{v_2, v_3, v_7, v_8, v_9\}, \{v_6, v_9\}\}$ is not self-contained since the intersection $\{v_9\}$ is not in $\mathcal{M}'$.

3. Algorithm COOMA

In this section, we present the algorithm COOMA for the connector enumeration problem. We first derive significant properties of connectors that are needed to design the algorithm. We then show the algorithm, along with an example of how it enumerates all connectors, followed by the correctness proof.

Lemma 1. *Let $(G = (V, E), I, \sigma : V \to 2^I)$ be an instance.*

(i) $C_{G,I,\sigma} \subseteq \mathcal{M}(G, I, \sigma)$.

(ii) *For any two connectors $X_1, X_2 \in \mathcal{M}(G, I, \sigma)$, it holds $C(X_1 \cap X_2, E) \subseteq \mathcal{M}(G, I, \sigma)$.*

(iii) *For each connector $Y \in \mathcal{M}(G, I, \sigma) \setminus C_{G,I,\sigma}$ and each item $i \in A_\sigma(Y)$, there is a connector $X \in \mathcal{M}(G, I, \sigma)$ with $X \supsetneq Y$ such that $Y \in C(X \cap C, E)$ for the component $C \in C(V_{\langle i \rangle}, E)$ that contains Y.*

(iv) *If a subfamily $\mathcal{M}' \subseteq \mathcal{M}(G, I, \sigma)$ is self-contained and contains $C_{G,I,\sigma}$, then $\mathcal{M}' = \mathcal{M}(G, I, \sigma)$.*

Proof. **(i)** Let X be a set in $C_{G,I,\sigma}$, where $G[X]$ is a connected subgraph of the graph $G[V_{\langle i \rangle}]$ and $i \in A_\sigma(X)$ for some $i \in I$. For each vertex $v \in V \setminus X$, if $i \in \sigma(v)$ (resp., if $i \notin \sigma(v)$) then $G[X \cup \{v\}]$ is not connected (resp., $i \in A_\sigma(X) \setminus \sigma(v) \neq \emptyset$). Hence X is a connector of (G, I, σ).

(ii) Let Y be a set in $C(X_1 \cap X_2, E)$, where $G[Y]$ is connected but $G[Y \cup \{v\}]$ is not connected for any vertex $v \in X_1 \cap X_2 \setminus Y$. Let $v \in V \setminus X_1 \cap X_2$ be a vertex. It suffices to show that $G[Y \cup \{v\}]$ is not connected or $A_\sigma(Y) \setminus \sigma(v) \neq \emptyset$. Since $X_i \in \mathcal{M}(G, I, \sigma)$, $i = 1, 2$, $G[X_i \cup \{v\}]$ is not connected or $A_\sigma(X_i) \setminus \sigma(v) \neq \emptyset$. Hence we see that $G[Y \cup \{v\}]$ is also not connected or $A_\sigma(Y) \setminus \sigma(v) \supseteq A_\sigma(X_i) \setminus \sigma(v) \neq \emptyset$ for $i = 1$ or 2, as required.

(iii) If $A_\sigma(Y) = A_\sigma(C)$ then $Y = C \in C_{G,I,\sigma} \subseteq \mathcal{M}(G, I, \sigma)$ would hold by (i). Since $Y \notin C_{G,I,\sigma}$, there is an item $j \in A_\sigma(Y) \setminus A_\sigma(C)$. Since $Y \notin C_{G,I,\sigma}$, $Y \subsetneq C'$ for the component $C' \in C(V_{\langle j \rangle}, E)$, where $C' \in C_{G,I,\sigma} \subseteq \mathcal{M}(G, I, \sigma)$ holds by (i). Moreover, Y is contained in a component of the graph $G[C' \cap C]$. This means that $\mathcal{M}(G, I, \sigma)$ contains a connector X with $X \supsetneq Y$ such that Y is contained in a component of the graph $G[X \cap C]$. We choose X as a minimal subset among all such connectors. Let Z denote the component of the graph $G[X \cap C]$ that contains Y, where $Z \in \mathcal{M}(G, I, \sigma)$ by (ii). If $Z \neq Y$, then $Y \in G[Z \cap C]$, contradicting the choice of X. Hence $Z = Y$ and the connector X satisfies the lemma.

(iv) Obviously $\mathcal{M}' \subseteq \mathcal{M}(G, I, \sigma)$. We show the converse. To derive a contradiction, assume that there is a set $Y \in \mathcal{M}(G, I, \sigma) \setminus \mathcal{M}'$, where we choose Y as a maximal subset among all such connectors. Let $i \in A_\sigma(Y)$ be an item and denote by C the component in $C(V_{\langle i \rangle}, E)$ that contains Y, where $C \neq Y$, since $Y \notin \mathcal{M}' \supseteq C_{G,I,\sigma} \supseteq C(V_{\langle i \rangle}, E)$. By (iii), there is a connector $X \in \mathcal{M}(G, I, \sigma)$ with $X \supsetneq Y$ such that $Y \in C(X \cap C, E)$. By the choice of Y, $X \in \mathcal{M}'$. This, however, means that $Y \in C(X \cap C, E) \setminus \mathcal{M}'$ for the connector $X \in \mathcal{M}'$, contradicting that $\mathcal{M}'$ is self-contained. □

From Lemma 1(ii), $\mathcal{M}(G, I, \sigma)$ is closed under intersection. From Lemma 1(iii), any inclusion-wise maximal connector is a member of $C_{G,I,\sigma}$ since for any connector Y not in $C_{G,I,\sigma}$, there is a connector $X \supsetneq Y$. The converse does not necessarily hold true; in Figure 1, $\{v_7\} \in C_{G,I,\sigma}$ is not an inclusion-wise maximal connector.

The algorithm COOMA works as follows. COOMA starts with an initial collection $\mathcal{M}_{\text{current}} \subseteq C_{G,I,\sigma}$ that consists of all components of $G[V_{\langle i_1 \rangle}, E_{\langle i_1 \rangle}]$ and all singletons in $C_{G,I,\sigma}$, where the item i_1 can be determined arbitrarily. The algorithm repeatedly enlarges $\mathcal{M}_{\text{current}}$ by "overlaying" every X in the current $\mathcal{M}_{\text{current}}$ on the subgraph $G[V_{\langle i \rangle}, E_{\langle i \rangle}]$, over all items $i \in I \setminus \{i_1\}$. By overlaying, we mean to generate connectors by taking the intersection between X and each component in $G[V_{\langle i \rangle}, E_{\langle i \rangle}]$. Upon the completion of the iteration on each item i, the components of $G[V_{\langle i \rangle}, E_{\langle i \rangle}]$ are added to $\mathcal{M}_{\text{current}}$ so that $\mathcal{M}_{\text{current}}$ is self-contained.

Now we summarize COOMA in Algorithm 1. We illustrate how it works by taking the instance of Figure 1. In line 3, the initial $\mathcal{M}_{\text{current}}$ is set to;

$$\mathcal{M}_{\text{current}} = \{\{v_1, v_2, v_6, v_9\}, \{v_4\}, \{v_7\}\}.$$

We observe the remaining part of the algorithm, where in the for-loop from lines 4 to 15, we set i to i_2 and i_3 in this order.

($i = i_2$) In line 5, the subgraph G_{i_2} consists of only one component, that is $\{v_2, v_3, v_7, v_8, v_9\}$. By overlaying $X \in \mathcal{M}_{\text{current}}$ on G_{i_2} in the while-loop (lines 7 to 13), we obtain a new connector $\{v_2, v_9\} = \{v_1, v_2, v_6, v_9\} \cap \{v_2, v_3, v_7, v_8, v_9\}$. Note that the collection $\mathcal{M}_{\text{temp}}$ is used to store $\mathcal{M}_{\text{current}}$ as of the beginning of the iteration on the item i. After the while-loop, the component $\{v_2, v_3, v_7, v_8, v_9\}$ in G_{i_2} is added to $\mathcal{M}_{\text{current}}$ in line 14. Then $\mathcal{M}_{\text{current}}$ becomes;

$$\mathcal{M}_{\text{current}} = \{\{v_1, v_2, v_6, v_9\}, \{v_2, v_3, v_7, v_8, v_9\}, \{v_2, v_9\}, \{v_4\}, \{v_7\}\}.$$

($i = i_3$) In line 5, the subgraph G_{i_3} consists of only one component, that is $\{v_3, v_5, v_6, v_7, v_9\}$. By overlaying $X \in \mathcal{M}_{\text{current}}$ on G_{i_3} in the while-loop, we obtain three new connectors as follows;

$$\{v_6, v_9\} = \{v_1, v_2, v_6, v_9\} \cap \{v_3, v_5, v_6, v_7, v_9\},$$
$$\{v_3\}, \{v_9\} \in \{v_2, v_3, v_7, v_8, v_9\} \cap \{v_3, v_5, v_6, v_7, v_9\}.$$

Note that $\{v_3, v_9\} = \{v_2, v_3, v_7, v_8, v_9\} \cap \{v_3, v_5, v_6, v_7, v_9\}$ is not a connector since $G[\{v_3, v_9\}]$ is disconnected. After the while-loop, the component $\{v_3, v_5, v_6, v_7, v_9\}$ in G_{i_3} is added to $\mathcal{M}_{\text{current}}$ in line 14. Finally we have the following output;

$$\mathcal{M}_{\text{current}} = \{\{v_1, v_2, v_6, v_9\}, \{v_2, v_3, v_7, v_8, v_9\}, \{v_3, v_5, v_6, v_7, v_9\}, \{v_2, v_9\}, \{v_3\}, \{v_4\}, \{v_6, v_9\}, \{v_7\}, \{v_9\}\}.$$

The following theorem shows the correctness of COOMA.

Theorem 1. *Algorithm 1 outputs $\mathcal{M}(G, I, \sigma)$ correctly.*

Proof. Let i_k, $k = 2, \ldots, p$ denote the items chosen from I in line 4 in this order, and let $\mathcal{M}^k_{\text{current}}$ denote the set $\mathcal{M}_{\text{current}}$ after the iteration on item i_k of the for-loop. By induction on $k = 2, \ldots, p$, we prove that $\mathcal{M}^k_{\text{current}}$ after the iteration on item i_k of the for-loop satisfies:

(a) $\mathcal{M}^k_{\text{current}} \subseteq \mathcal{M}(G, I, \sigma)$;

(b) $\mathcal{M}^k_{\text{current}} \supseteq C(V_{\langle i_j \rangle}, E)$ for all items $i_j \in I$ with $j < k$; and

Algorithm 1 COOMA

Input: a graph $G = (V, E)$, an item set I and a function $\sigma : V \to 2^I$
Output: $\mathcal{M}(G, I, \sigma)$

1: Compute $C(V_{\langle i \rangle}, E)$ and $\mathcal{F}_i := C(V_{\langle i \rangle}, E_{\langle i \rangle})$ for each item $i \in I$, and $\mathcal{S} := \bigcup_{i \in I}(C(V_{\langle i \rangle}, E) \setminus C(V_{\langle i \rangle}, E_{\langle i \rangle}))$;
▷ $C(V_{\langle i \rangle}, E) \setminus C(V_{\langle i \rangle}, E_{\langle i \rangle})$ consists of singletons
2: Choose an item $i_1 \in I$;
3: $\mathcal{M}_{\text{current}} := \mathcal{F}_{i_1} \cup \mathcal{S}$;
4: **for** each item $i \in I - \{i_1\}$ **do**
5: Construct the graph $G_i := G[V_{\langle i \rangle}, E_{\langle i \rangle}]$;
6: $\mathcal{M}_{\text{temp}} := \mathcal{M}_{\text{current}}$;
7: **while** $\exists X \in \mathcal{M}_{\text{temp}}$ with $|X| \geq 2$ **do**
8: Choose a set $X \in \mathcal{M}_{\text{temp}}$ with maximum $|X| \geq 2$;
9: $\mathcal{M}_{\text{temp}} := \mathcal{M}_{\text{temp}} \setminus \{X\}$;
10: Compute $C(X, E_{\langle i \rangle})$ in the graph G_i;
11: $\mathcal{M}_{\text{current}} := \mathcal{M}_{\text{current}} \cup C(X, E_{\langle i \rangle})$ without creating duplications;
12: $\mathcal{M}_{\text{temp}} := \mathcal{M}_{\text{temp}} \setminus C(X, E_{\langle i \rangle})$;
13: **end while**
14: $\mathcal{M}_{\text{current}} := \mathcal{M}_{\text{current}} \cup \mathcal{F}_i$ without creating duplications;
15: **end for**
16: Output $\mathcal{M}_{\text{current}}$ as $\mathcal{M}(G, I, \sigma)$.

(c) $\mathcal{M}^k_{\text{current}}$ is self-contained.
Let $\mathcal{M}^1_{\text{current}}$ denote the set $\mathcal{M}_{\text{current}}$ in line 3. Then $\mathcal{M}^1_{\text{current}}$ is a subset of $\mathcal{M}(G, I, \sigma)$, since $C(V_{\langle i_1 \rangle}, E) \subseteq \mathcal{F}_{i_1} \cup \mathcal{S} \subseteq \mathcal{C}_{G,I,\sigma} \subseteq \mathcal{M}(G, I, \sigma)$ by Lemma 1(i). Obviously $\mathcal{M}^1_{\text{current}}$ is self-contained since any pair of sets are either disjoint, or one is a singleton and a subset of the other. Hence $\mathcal{M}^1_{\text{current}}$ satisfies conditions (a)-(c) for $k = 1$.

For $k \geq 2$, assume that $\mathcal{M}^{k-1}_{\text{current}}$ before the iteration on item i_k of the for-loop satisfies conditions (a)-(c). We prove that $\mathcal{M}^k_{\text{current}}$ satisfies conditions (a)-(c). The set $\mathcal{M}^{k-1}_{\text{current}}$ will be augmented during the while-loop for item i_k and in line 14. Note that each set Y in $C(X, E_{\langle i \rangle})$ in line 10 is a component in $C(X \cap C, E)$ for a component $C \in C(V_{\langle i_k \rangle}, E_{\langle i_k \rangle}) \subseteq C(V_{\langle i_k \rangle}, E)$ with $|C| \geq 2$. The while-loop adds to $\mathcal{M}^{k-1}_{\text{current}}$ the sets Y in $C(X \cap C, E)$ for each pair (X, C) of a set $X \in \mathcal{M}^{k-1}_{\text{current}}$ ($\subseteq \mathcal{M}(G, I, \sigma)$) with $|X| \geq 2$ and a set $C \in C(V_{\langle i_k \rangle}, E)$ with $|C| \geq 2$. All sets Y in $C(X \cap C, E)$ are connectors in $\mathcal{M}(G, I, \sigma)$ by Lemma 1(ii). Line 14 adds to $\mathcal{M}^{k-1}_{\text{current}}$ the family $C(V_{\langle i_k \rangle}, E_{\langle i_k \rangle})$, which is a subset of $\mathcal{M}(G, I, \sigma)$ by Lemma 1(i). Hence after the iteration, conditions (a) and (b) hold for $\mathcal{M}^k_{\text{current}}$. To prove that (c) holds after the iteration, it suffices to show that for each set $X \in \mathcal{M}^{k-1}_{\text{current}}$, the family $C(X \cap C, E)$ with any set $C \in C(V_{\langle i_k \rangle}, E)$ with $|C| \geq 2$ is included in $\mathcal{M}^k_{\text{current}}$, since (c) holds before the iteration. If $X \in \mathcal{M}^{k-1}_{\text{current}}$, then $C(X \cap C, E)$ for any set $C \in C(V_{\langle i_k \rangle}, E)$ with $|C| \geq 2$ is added to $\mathcal{M}^{k-1}_{\text{current}}$ during the while-loop. If X is a newly added set to $\mathcal{M}^k_{\text{current}}$, then there is a set $C_X \in C(V_{\langle i_k \rangle}, E)$ with $C_X \supseteq X$, where the family $C(X \cap C, E)$ for each set $C \in C(V_{\langle i_k \rangle}, E)$ is empty or contains C_X. Therefore (c) holds after the iteration.

After the last iteration of the for-loop, $\mathcal{M}^p_{\text{current}}$ is a self-contained subfamily of $\mathcal{M}(G, I, \sigma)$ with $\mathcal{M}^p_{\text{current}} \supseteq \mathcal{C}_{G,I,\sigma}$, which means that $\mathcal{M}^p_{\text{current}} = \mathcal{M}(G, I, \sigma)$ by Lemma 1(iv). □

We analyze the time complexity of COOMA. For a family $\mathcal{M}$ of connectors, let us represent by $\|\mathcal{M}\|$ the total of the graph size (i.e., $|V[X]| + |E[X]|$) over all connectors X in $\mathcal{M}$. In the for-loop (from lines 4 to 15) with the item i, the subgraph $G[X]$ is searched for each $X \in \mathcal{M}^i_{\text{current}}$. Then the running time is bounded by;

$$O(\sum_{i \in I} \|\mathcal{M}^i_{\text{current}}\|) = O(|I| \cdot \|\mathcal{M}(G, I, \sigma)\|).$$

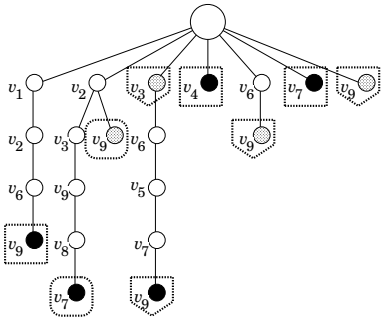

Fig. 2 The search tree of COPINE on the example of Figure 1

4. Discussion

4.1 Difference between COOMA and COPINE

An existing algorithm COPINE [9], [10] traverses a search tree in the depth-first manner. In Figure 2, we show the search tree for the instance of Figure 1. In the search tree, each node except the root is associated with a vertex in G, and accordingly, it is also associated with a subset of vertices that are on the path from the root to the node. The black nodes represent connectors in $\mathcal{C}_{G,I,\sigma}$, whereas the gray nodes represent connectors in $\mathcal{M}(G, I, \sigma) \setminus \mathcal{C}_{G,I,\sigma}$. COPINE identifies whether the vertex subset X of the visited node is a connector or not, and outputs X if it is so. It has a mechanism for pruning the tree, by which redundant search is avoided. For example, if $G[X]$ is disconnected or $|A_\sigma(X)| < \theta$, then we can skip the search of the descendants of the current node.

COOMA enumerates connectors in a completely different way. The nodes indicated by a rectangle represent the connectors in $\mathcal{M}_{\text{current}}$ as of line 3. In the first (resp., second) for-loop from lines 4 to 15 where i is set to i_2 (resp., i_3) the connectors indicated by a rounded rectangle (resp., a pentagon) are found.

When we enumerate θ-connectors for a large θ, we expect COPINE to be more efficient than COOMA since COPINE has a mechanism for pruning the search tree based on the item set size, while COOMA does not. To enumerate θ-connectors by COOMA, we have to run the algorithm completely and to collect X such that $|A_\sigma(X)| \geq \theta$. In this strategy, the computation time of COOMA hardly changes with respect to θ.

An extension of the connector enumeration problem may ask for enumeration of connectors X such that $|X| \geq s$ for a parameter s, supposing that a connector of too small a size is meaningless. Then, we expect COOMA to be superior for this extension; we can implement COOMA so that $\mathcal{M}_{\text{current}}$ is maintained separately by the set size. To be more precise, preparing an appropriate data structure to store the found connectors (e.g., trie, binary tree and hash table) for each set size $|X| \in \{1, \ldots, |V|\}$, we store a discovered connector X by inserting X to the data structure for the size $|X|$. We can ignore a connector X and thus discard it if $|X| < s$. On the other hand, it may be hard to prune the search tree of COPINE based on $|X|$.

Algorithm 2 R-COOMA

Input: a graph $G = (V, E)$, an item set I and a function $\sigma : V \to 2^I$
Output: $\mathcal{M}(G, I, \sigma)$

1: Compute $C(V_{\langle i \rangle}, E)$ and $C(V_{\langle i \rangle}, E_{\langle i \rangle})$ for each item $i \in I$, and $\mathcal{S} := \bigcup_{i \in I}(C(V_{\langle i \rangle}, E) \setminus C(V_{\langle i \rangle}, E_{\langle i \rangle}))$ without creating duplications;
▷ $C(V_{\langle i \rangle}, E) \setminus C(V_{\langle i \rangle}, E_{\langle i \rangle})$ consists of singletons
2: $\mathcal{F} := \bigcup_{i \in I} C(V_{\langle i \rangle}, E_{\langle i \rangle})$ without creating duplications;
3: $j := 0$;
4: **while** $\mathcal{F} \neq \emptyset$ **do**
5: $j := j + 1$;
6: Find a maximal family $\mathcal{F}_j$ of disjoint sets in $\mathcal{F}$ by the minimum-cardinality greedy method (i.e., after initializing $\mathcal{F}_j := \emptyset$, add a set $X \in \mathcal{F} \setminus \mathcal{F}_j$ with minimum $|X|$ to $\mathcal{F}_j$);
7: $\mathcal{F} := \mathcal{F} \setminus \mathcal{F}_j$;
8: **end while**
9: $p := j$;
10: $\mathcal{M}_{\text{current}} := \mathcal{F}_1 \cup \mathcal{S}$;
11: **for** each index $i = 2, 3, \ldots, p$ **do**
12: Construct the graph $G_i := G[V, \cup_{S \in \mathcal{F}_i} E[S]]$;
13: $\mathcal{M}_{\text{temp}} := \mathcal{M}_{\text{current}}$;
14: **while** $\exists X \in \mathcal{M}_{\text{temp}}$ with $|X| \geq 2$ **do**
15: Choose a set $X \in \mathcal{M}_{\text{temp}}$ with maximum $|X| \geq 2$;
16: $\mathcal{M}_{\text{temp}} := \mathcal{M}_{\text{temp}} \setminus \{X\}$;
17: Compute $C(X, E(G_i))$ in the graph G_i;
18: $\mathcal{M}_{\text{current}} := \mathcal{M}_{\text{current}} \cup C(X, E(G_i))$ without creating duplications;
19: $\mathcal{M}_{\text{temp}} := \mathcal{M}_{\text{temp}} \setminus C(X, E(G_i))$;
20: **end while**
21: $\mathcal{M}_{\text{current}} := \mathcal{M}_{\text{current}} \cup \mathcal{F}_i$ without creating duplications;
22: **end for**
23: Output $\mathcal{M}_{\text{current}}$ as $\mathcal{M}(G, I, \sigma)$.

4.2 Reorganization of (I, σ)

The implementation of COOMA as in Algorithm 1 repeats the for-loop from lines 4 to 15 over the items in $I \setminus \{i_1\}$. If we could reorganize (I, σ) to (I', σ') so that $\mathcal{M}(G, I, \sigma) = \mathcal{M}(G, I', \sigma')$ and $|I| > |I'|$, then COOMA would enumerate all connectors in $\mathcal{M}(G, I, \sigma)$ more quickly, from the instance (G, I', σ').

In Algorithm 2, we show such a variant of COOMA, named R-COOMA (reorganized COOMA). The part that reorganizes (I, σ) is from lines 2 to 9. In line 2, all connectors $X \in C_{G,I,\sigma}$ such that $|X| \geq 2$ are collected into $\mathcal{F}$. Then from lines 4 to 8, the "new item set" is constructed by partitioning $\mathcal{F}$ into $\mathcal{F}_1, \ldots, \mathcal{F}_p$. Each $\mathcal{F}_j$ $(j = 1, \ldots, p)$ is the union of disjoint subsets in $\mathcal{F}$, and the subsets are chosen from $\mathcal{F}$ by the minimum-cardinality greedy method.

The for-loop from lines 11 to 22 is almost the same as that of COOMA in Algorithm 1. The only exception is line 12. We take the subgraph $G_i := G[V, \cup_{S \in \mathcal{F}_i} E[S]]$ instead of the subgraph $G_i := G[V_{\langle i \rangle}, E_{\langle i \rangle}]$ (see line 5 in Algorithm 2).

Let us describe why we employ the minimum-cardinality greedy method in constructing $\mathcal{F}_j$. If a connector is found during the algorithm, then it is overlaid to the subsequent subgraphs G_i. Connectors that are found earlier in the algorithm are overlaid more times than ones that are found later. Hence, it must be better to use the minimum-cardinality greedy method to construct $\mathcal{F}_j$.

Also we expect that $p < |I|$ holds and that R-COOMA runs faster than COOMA.

The next theorem states the correctness of R-COOMA.

Theorem 2. *Algorithm 2 outputs $\mathcal{M}(G, I, \sigma)$ correctly.*

Proof. For $k = 2, 3, \ldots, p$ let $\mathcal{M}^k_{\text{current}}$ denote the set $\mathcal{M}_{\text{current}}$ at the end of the k-th iteration of the for-loop in lines 11 to 22, and let $\mathcal{M}^1_{\text{current}}$ be the set $\mathcal{M}_{\text{current}}$ obtained in line 10. We will first show that each $\mathcal{M}^k_{\text{current}}$, $k = 1, 2, \ldots, p$, satisfies:

(a) $\mathcal{M}^k_{\text{current}} \subseteq \mathcal{M}(G, \sigma)$;
(b) $\mathcal{M}^k_{\text{current}}$ is self-contained;
and finally,
(c) $\mathcal{M}^p_{\text{current}} \supseteq C(V_{\langle i \rangle}, E)$, for all $i \in I$.

First, observe that (a) and (b) are satisfied by $\mathcal{M}^1_{\text{current}} = \mathcal{F}_1 \cup \mathcal{S}$ in line 10, since by Lemma 1(i) it holds that $\mathcal{F}_1 \cup \mathcal{S} \subseteq C_{G,I,\sigma} \subseteq \mathcal{M}(G, I, \sigma)$, and any two sets in $\mathcal{M}^1_{\text{current}}$ are either disjoint or one is a singleton and a subset of the other. Next, for $k \geq 2$, assume that $\mathcal{M}^{k-1}_{\text{current}}$ satisfies (a) and (b).

The set $\mathcal{M}^{k-1}_{\text{current}}$ will be updated in the while-loop of lines 14 to 20, and in line 21. Now, each set $Y \in C(X, E(G_i))$ in line 17 is a component in $C(X \cap C, E)$ for a connector $C \in C(V, \cup_{S \in \mathcal{F}_i} E[S]) \subseteq C_{G,I,\sigma}$. The while-loop of lines 14 to 20 adds to $\mathcal{M}^{k-1}_{\text{current}}$ the sets Y in $C(X \cap C, E)$ for each pair of a set $X \in \mathcal{M}^{k-1}_{\text{current}}$ with $|X| \geq 2$ and a set $C \in C(V, \cup_{S \in \mathcal{F}_i} E[S])$ with $|C| \geq 2$. By Lemma 1(ii), all such sets Y are connectors in $\mathcal{M}(G, I, \sigma)$. In line 21 all sets of the family $\mathcal{F}_i \subseteq C_{G,I,\sigma} \subseteq \mathcal{M}(G, I, \sigma)$ are added to $\mathcal{M}^k_{\text{current}}$, and therefore $\mathcal{M}^k_{\text{current}}$ satisfies (a).

To see that $\mathcal{M}^k_{\text{current}}$ also satisfies (b), it suffices to show that, for each $X \in \mathcal{M}^{k-1}_{\text{current}}$ and all $C \in C(V, \cup_{S \in \mathcal{F}_i} E[S])$, the family $C(X \cap C, E)$ is included in $\mathcal{M}^k_{\text{current}}$, since (b) is satisfied for $\mathcal{M}^{k-1}_{\text{current}}$ by the inductive assumption. If $X \in \mathcal{M}^{k-1}_{\text{current}}$, then this is achieved by the while-loop of lines 14 to 20. Otherwise, i.e., if X is newly added to $\mathcal{M}^{k-1}_{\text{current}}$, then there must be a set $C_X \in C(V, \cup_{S \in \mathcal{F}_i} E[S])$ with $C_X \supseteq X$. For each set $C \in C(V_{\langle i_k \rangle}, E)$, the family $C(X \cap C, E)$ is empty or contains C_X, and therefore (b) also holds after the iteration.

Finally, at the end of the for-loop, $\mathcal{M}^p_{\text{current}}$ contains $C_{G,I,\sigma} = \mathcal{S} \cup \bigcup_{i=1}^p \mathcal{F}_i$, and thus (c) is satisfied. By Lemma 1(iv), it holds that $\mathcal{M}^p_{\text{current}} = \mathcal{M}(G, I, \sigma)$. □

4.3 Reduction

We may reduce the input size by preprocessing. Here we introduce some such techniques.

Reduction 1. *Any vertex $v \in V$ with $|\sigma(v)| < \theta$ can be removed from G.*

This is possible because $\mathcal{M}_\theta(G, I, \sigma)$ remains unchanged after v is removed from G. Analogously, we can remove an edge uv with $|A_\sigma(\{u, v\})| < \theta$.

Reduction 2. *Any edge $uv \in E$ with $|A_\sigma(\{u, v\})| < \theta$ can be removed from G.*

For any edge uv with $\sigma(u) = \sigma(v)$, it holds that $|X \cap \{u, v\}| = 0$ or 2 for each set $X \in \mathcal{M}(M, I, \sigma)$. This leads to the following reduction.

Reduction 3. *We can contract any edge $uv \in E$ with $\sigma(u) = \sigma(v)$ to obtain a smaller graph.*

Note that Reduction 3 can be applied to a leaf edge $uv \in E$ with $\sigma(u) = \sigma(v)$.

5. Computational Experiments

In this section, we compare the three algorithms, COOMA,

R-COOMA and COPINE, in terms of computation time. The benchmark instances include real genetic networks as well as random instances. We show the efficiency of COOMA and R-COOMA, especially for instances that have a large number of connectors.

Experimental Settings

We implemented COOMA and R-COOMA in C++. For COPINE, we employ the source code (written in C) that is used in [8]. We apply Reductions 1 and 2 to reduce the input size.

All the experiments are conducted on a workstation that carries an Intel Core i7-4770 Processor (up to 3.90GHz by means of Turbo Boost Technology) and 8GB main memory. The installed OS is Ubuntu 16.04. Under this environment, it takes 0.25 s, 1.54 s and 5.90 s approximately to execute the `dmclique` benchmark (`http://dimacs.rutgers.edu/pub/dsj/clique/`) for instances `r300.5.b`, `r400.5.b` and `r500.5.b`, respectively. We compile the source codes of COOMA and R-COOMA by the g++ compiler (ver. 5.4.0) with `-O2` option, and the source code of COPINE by the gcc compiler (ver. 5.4.0) with `-O2` option.

Random Instances

We generate a random instance (G, I, σ) under four parameters, that is the number n of vertices, the number q of items, the edge density $d_E \in [0, 1]$, and the item density $d_I \in [0, 1]$. Specifically, first we take a random graph $G = (V, E)$ of the Erdös-Rényi model such that $|V| = n$ and an edge is drawn between any two vertices with the probability d_E. Let $I = \{i_1, \ldots, i_q\}$. For each vertex $v \in V$, we decide the set $\sigma(v)$ of items so that, for every item $i \in I$, $i \in \sigma(v)$ holds with the probability d_I. For (n, q, d_E, d_I), we use all combinations of the following;

- $n \in \{100, 150, 200, 250, 300\}$;
- $q \in \{20, 30, 40\}$;
- $d_E \in \{0.25, 0.50, 0.75\}$;
- $d_I \in \{0.30, 0.40, 0.50\}$.

We generate five random instances for each (n, q, d_E, d_I). The values that are shown below are averages over the five instances.

First, let us observe how the number $|\mathcal{M}(G, I, \sigma)|$ of connectors changes with respect to the parameters. In Figure 3, we show how the number changes when we increase one parameter value, fixing the other parameters. In general, $|\mathcal{M}(G, I, \sigma)|$ gets larger as a parameter value increases. When the parameters take the largest values (i.e., $(n, q, d_E, d_I) = (300, 40, 0.75, 0.50)$), $|\mathcal{M}(G, I, \sigma)|$ is up to 3.8×10^6.

It is interesting to see that the number $|\mathcal{M}(G, I, \sigma)|$ is the most sensitive to d_I, rather than $q = |I|$; the number becomes more than eight times ($5673.2/677.8 = 8.37$) when we increase d_I from 0.30 to 0.50, while the number becomes just $3081.2/677.8 = 4.55$ times when we double q from 20 to 40. The reason is described as follows; when d_I is larger, the intersection $|\sigma(u) \cap \sigma(v)|$ tends to be larger for any u and v. Then each vertex must be included in more connectors.

Next, we show the distribution of connectors with respect to the cardinality. We show the distribution for a random instance under $(n, q, d_E, d_I) = (300, 40, 0.75, 0.50)$ in Figure 4. Note that the vertical axis is in logarithmic scale. The cardinality of almost all connectors is no more than 20. We see some peaks. For example, the rightmost peak is interpreted as follows; For an item

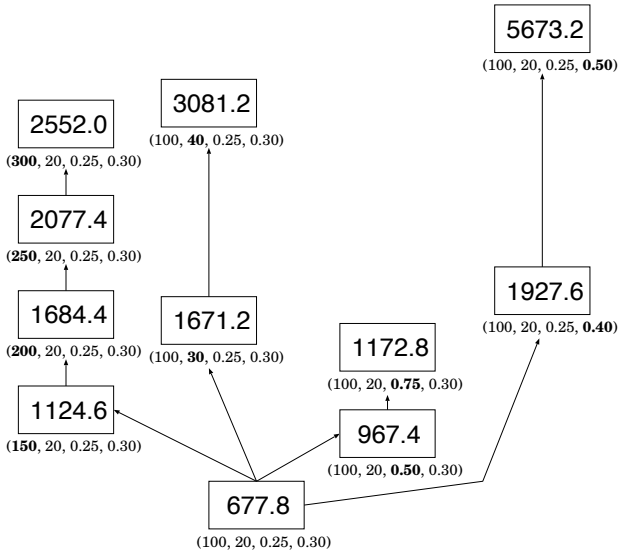

Fig. 3 The numbers $|\mathcal{M}(G, I, \sigma)|$ of connectors in random instances under parameters (n, q, d_E, d_I)

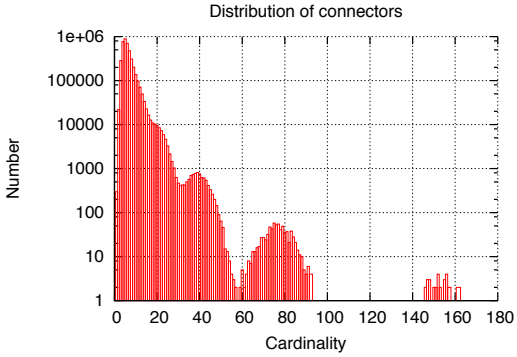

Fig. 4 Distribution of connectors X with respect to the cardinality $|X|$; $(n, q, d_E, d_I) = (300, 40, 0.75, 0.50)$

i, the expected number of vertices that have i is $300 \times 0.5 = 150$. These vertices must belong to the same component because the edge density $d_E = 0.75$ is relatively high. They form a connector, and then we see a peak around $|X| = 150$.

Finally, we compare the three algorithms, COOMA, R-COOMA and COPINE, in terms of computation time. We show their computation time in Figure 5. The vertical axis indicates the computation time, and the horizontal axis indicates $|I| \cdot |\mathcal{M}(G, I, \sigma)|$ of a random instance. We take this value as a rough approximate of $|I| \cdot ||\mathcal{M}(G, I, \sigma)||$, expecting the computation time of COOMA to be proportional to it. The three symbols $\triangle$ (COOMA), $\square$ (R-COOMA) and + (COPINE) on the same vertical line show the computation time for the same instance. As shown, as $|I| \cdot |\mathcal{M}(G, I, \sigma)|$ gets larger, COOMA and R-COOMA are more likely to be faster than COPINE, and their computation time is proportional to $|I| \cdot |\mathcal{M}(G, I, \sigma)|$ in general.

The difference of computation time between COOMA and R-COOMA is not clear in Figure 5. The figure focuses on small instances such that $|I| \cdot |\mathcal{M}(G, I, \sigma)| \leq 10^6$ to observe the difference between the two and COPINE.

We show the computation time for larger instances in Figure 6. In this figure, we plot the computation time of only COOMA and R-COOMA. We see that R-COOMA is more efficient than

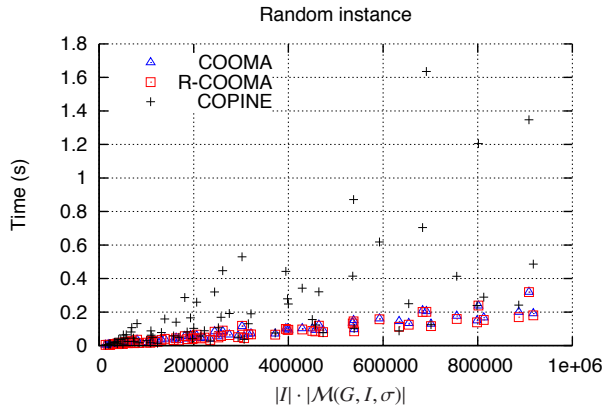

Fig. 5 Computation time of the three algorithms for random instances such that $|I| \cdot |\mathcal{M}(G, I, \sigma)| \leq 10^6$

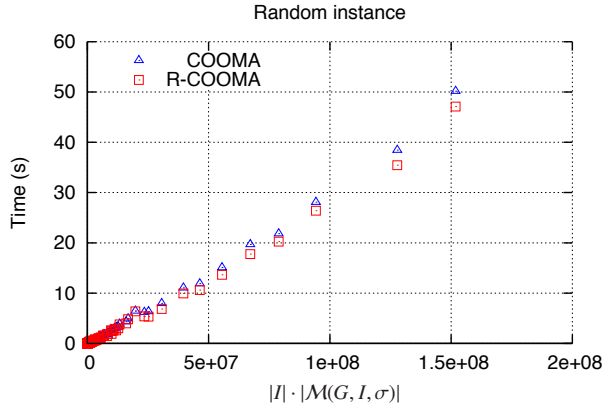

Fig. 6 Computation time of COOMA and R-COOMA for random instances that have larger $|I| \cdot |\mathcal{M}(G, I, \sigma)|$

COOMA, as $|I| \cdot |\mathcal{M}(G, I, \sigma)|$ gets larger.

R-COOMA is an algorithm that reorganizes (I, σ) by preprocessing before running COOMA. The size of the new item set is substituted for p in line 9 of Algorithm 2. In our experiment, $q = p$ holds for all instances. Nevertheless, R-COOMA is faster than COOMA. This indicates that an item i having small connectors in $C(V_{\langle i \rangle}, E)$ should be searched prior to items i' which have larger connectors in $C(V_{\langle i' \rangle}, E)$.

Genetic Networks

We apply the enumeration algorithms to instances generated from real genetic networks. The genetic data are provided by Dr. Jiexun Wang, a biostatistician from Khoo Teck Puat Hospital in Singapore.

We describe an overview of the genetic data. It consists of 22 data sets, one of which corresponds to a pair of autosome chromosomes of a human cell. In each data set, there are several SNPs (single nucleotide polymorphism) that are associated with multiple diseases. Roughly, an SNP is regarded as a proxy to a gene. The association between any two SNPs is measured by an LD (linkage disequilibrium) value, which is a numerical value in $[-1, 1]$.

For each data set, we consider the problem of enumerating all maximal subsets of SNPs such that:

- the SNPs share a common set of diseases; and
- any two SNPs within them are "strongly" associated (i.e., the absolute value of the LD value between the SNPs is high).

We transform a data set into our instance as follows. We regard an SNP as a vertex, a disease as an item, and that the SNPs form a complete graph with weighted edges, where the weight of an edge that connects two SNPs is given by the LD value between them. To focus on the strong edges, we reserve $\gamma\binom{n}{2}$ edges that have the largest absolute LD values, and remove the remaining edges, where n is the number of vertices and $\gamma \in [0, 1]$ is a parameter. What we would like to enumerate is connectors in the resulting graph.

We show the number of connectors and computation time of the algorithms in Table 2. In this experiment, γ is set to 0.50. In these instances, the number n of vertices is from 48 (ID 21) to 335 (ID 6), and the number q of items is from 1234 (ID 21) to 1358 (ID 6). They contain much more items than the random instances in the last experiment, while the item density d_I is lower; it is as small as about 0.05. We plot the computation time of the three algorithms in Figure 7, in the similar manner as Figures 5 and 6.

We see that R-COOMA is clearly faster than COOMA. The computation time of R-COOMA is less than 50% of the compu-

Table 2 Number of connectors (indicated by $|\mathcal{M}|$) and computation time (s) of the algorithms for 22 instances generated from the genetic data: ϵ means that the computation time is less than 0.01 seconds.

ID	n	q	d_I	$\lvert\mathcal{M}\rvert$	COP-INE	COO-MA	R-COOMA	(p)
1	267	1358	0.051	30422	1.13	2.84	1.44	(228)
2	266	1358	0.050	28141	1.05	2.51	1.32	(225)
3	194	1358	0.050	14407	0.40	1.31	0.67	(183)
4	150	1353	0.048	8305	0.16	0.78	0.34	(145)
5	157	1356	0.048	8347	0.18	0.77	0.35	(149)
6	335	1358	0.054	92360	4.63	10.78	4.37	(305)
7	137	1355	0.048	6189	0.11	0.56	0.26	(134)
8	143	1357	0.049	7453	0.14	0.70	0.33	(150)
9	144	1355	0.053	9599	0.20	0.98	0.46	(190)
10	156	1356	0.051	9820	0.21	0.94	0.45	(164)
11	170	1357	0.050	10668	0.25	1.00	0.48	(166)
12	168	1357	0.050	9707	0.22	0.92	0.44	(166)
13	86	1332	0.048	2710	0.03	0.24	0.10	(102)
14	94	1337	0.051	3569	0.04	0.33	0.14	(118)
15	112	1347	0.051	5550	0.08	0.54	0.23	(140)
16	100	1342	0.050	3892	0.05	0.36	0.15	(120)
17	93	1336	0.053	4439	0.05	0.46	0.18	(161)
18	63	1293	0.052	1627	0.01	0.14	0.05	(88)
19	91	1342	0.052	3578	0.04	0.34	0.14	(123)
20	82	1326	0.052	2854	0.03	0.26	0.11	(109)
21	48	1234	0.056	1001	ϵ	0.08	0.03	(84)
22	55	1238	0.054	1231	ϵ	0.10	0.04	(81)

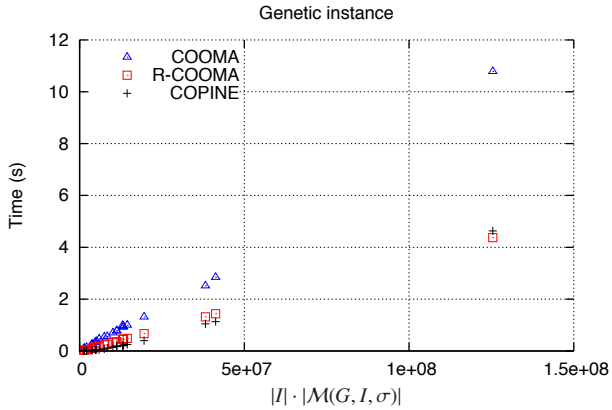

Fig. 7 Computation time of the three algorithms for the genetic instances

tation time of COOMA for all instances. In Table 2, we show p, the size of the "new" item set that R-COOMA obtains as a result of the preprocessing. As shown, the new item set contains fewer than 1/4 items than the original item set for all instances. This must be one of the significant reasons that R-COOMA is faster than COOMA.

COPINE is the fastest for all instances except ID 6, while the difference between COPINE and R-COOMA is quite small. Since R-COOMA is faster than COPINE for the instance ID 6, which has the largest number of connectors, we expect that, for instances that have more connectors, R-COOMA is more efficient than COPINE.

6. Concluding Remarks

In this paper, we propose a novel algorithm COOMA for the problem of enumerating all connectors of a given graph with an item set. We experimentally show the efficiency of COOMA and its sophisticated version, R-COOMA, in comparison with COPINE, an existing algorithm, especially for instances that have a large number of connectors.

Among the future work is improvement of our implementation. COOMA is a graph-search based enumeration algorithm, and it would be interesting to design another algorithm based on other algorithmic frameworks, e.g., family trees. Possible extensions include the formulation of other graph models (e.g., hypergraphs, digraphs and vertex- and/or edge-weighted cases) under various requirements (e.g., k-edge- and/or k-vertex-connectivity, min/max degree and flow values or distance in weighted versions).

Acknowledgments We gratefully acknowledge Dr. Shingo Okuno for sharing his COPINE source code with us. We also thank Dr. Jiexun Wang for providing the genetic data to us.

References

[1] Atzmueller, M., Doerfel, S. and Mitzlaff, F.: Description-oriented community detection using exhaustive subgroup discovery, *Information Sciences*, Vol. 329, pp. 965–984 (2016).
[2] Avis, D. and Fukuda, K.: Reverse search for enumeration, *Discrete Applied Mathematics*, Vol. 65, No. 1, pp. 21–46 (1996).
[3] Bellman, R. E.: *Dynamic Programming*, Princeton University Press (1957).
[4] Imada, T., Ota, S., Nagamochi, H. and Akutsu, T.: Efficient enumeration of stereoisomers of tree structured molecules using dynamic programming, *Journal of Mathematical Chemistry*, Vol. 49, No. 4 (online), available from ⟨http://hdl.handle.net/2433/139571⟩ (2011).
[5] Inokuchi, A., Washio, T. and Motoda, H.: Complete mining of frequent patterns from graphs: Mining graph data, *Machine Learning*, Vol. 50, No. 3, pp. 321–354 (online), DOI: https://doi.org/10.1023/A:102172622 (2003).
[6] Minato, S.: Zero-suppressed BDDs for Set Manipulation in Combinatorial Problems, *Proceedings of the 30th International Design Automation Conference (DAC '93)*, pp. 272–277 (1993).
[7] Minato, S.: Power of Enumeration - Recent Topics on BDD/ZDD-Based Techniques for Discrete Structure Manipulation, *IEICE Transactions on Information and Systems*, Vol. E100.D, No. 8, pp. 1556–1562 (2017).
[8] Okuno, S.: Parallelization of Graph Mining using Backtrack Search Algorithm, PhD Thesis, Kyoto University (2017).
[9] Seki, M. and Sese, J.: Identification of active biological networks and common expression conditions, *2008 8th IEEE International Conference on BioInformatics and BioEngineering*, pp. 1–6 (2008).
[10] Sese, J., Seki, M. and Fukuzaki, M.: Mining Networks with Shared Items, *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10)*, pp. 1681–1684 (2010).
[11] Silva, A., Meira, Jr., W. and Zaki, M. J.: Mining Attribute-structure Correlated Patterns in Large Attributed Graphs, *Proc. VLDB Endow.*, Vol. 5, No. 5, pp. 466–477 (2012).
[12] Wasa, K.: Enumeration of Enumeration Algorithms, *CoRR*, Vol. abs/1605.05102 (online), available from ⟨http://arxiv.org/abs/1605.05102⟩ (2016).
[13] Yan, X. and Han, J.: gSpan: Graph-Based Substructure Pattern Mining, *Proceedings of 2002 IEEE International Conference on Data Mining (ICDM '02)*, pp. 721–724 (2002).
[14] 齊藤有紀, 寺田愛花 and 瀬々潤: 無限次数多重検定法へのグラフ制約の導入とゲノムワイド関連解析への応用, 研究報告数理モデル化と問題解決（*MPS*）, Vol. 2014-MPS-98, No. 4, pp. 1–6 (2014).