

# Approximate Association via Dissociation

Jie You<sup>1,2</sup>, Jianxin Wang<sup>1</sup>, and Yixin Cao<sup>2</sup>(✉)

<sup>1</sup> School of Information Science and Engineering,  
Central South University, Changsha, China  
jxwang@mail.csu.edu.cn

<sup>2</sup> Department of Computing, Hong Kong Polytechnic University,  
Hong Kong, China  
yixin.cao@polyu.edu.hk

**Abstract.** A vertex set  $X$  of a graph  $G$  is an association set if each component of  $G - X$  is a clique, or a dissociation set if each component of  $G - X$  is a single vertex or a single edge. Interestingly,  $G - X$  is then precisely a graph containing no induced  $P_3$ 's or containing no  $P_3$ 's, respectively. We observe some special structures and show that if none of them exists, then the minimum association set problem can be reduced to the minimum (weighted) dissociation set problem. This yields the first nontrivial approximation algorithm for the association set problem, with approximation ratio is 2.5. The reduction is based on a combinatorial study of modular decomposition of graphs free of these special structures. Further, a novel algorithmic use of modular decomposition enables us to implement this approach in  $O(mn + n^2)$  time.

## 1 Introduction

A *cluster graph* comprises a family of disjoint cliques, each an *association*. Cluster graphs have been an important model in the study of clustering objects based on their pairwise similarities, particularly in computational biology and machine learning [3]. If we represent each object with a vertex, and add an edge between two objects that are similar, we would expect a cluster graph. If this fails, a natural problem is then to find and exclude a minimum number of vertices such that the rest forms a cluster graph; this is the *association set* problem. This problem has recently received significant interest from the community of parameterized computation, where it is more commonly called *cluster vertex deletion* [4, 15]. The cardinality of a minimum association set of a graph is also known as its *distance to clusters*. It is one of the few structural parameters for dense graphs [9, 10], in contrast with a multitude of structural parameters for sparse graphs, thereby providing another motivation for this line of research. For example, Bruhn et al. [5] recently showed that the boxicity problem (of deciding the minimum  $d$  such that a graph  $G$  can be represented as an intersection graph

---

Supported in part by NSFC under grants 61572414 and 61420106009, and RGC under grant 252026/15E.

of axis-aligned boxes in the  $d$ -dimension Euclidean space) is fixed-parameter tractable parameterized by the distance to clusters.

The association set problem belongs to the family of vertex deletion problems studied by Yannakakis et al. [16, 18]. The task in these problems is to delete the minimum number of vertices from a graph so that the remaining subgraph satisfies a hereditary property; recall that a graph property is *hereditary* if it is closed under taking induced subgraphs [16]. It is known that a hereditary property can be characterized by a (possibly infinite) set of forbidden induced subgraphs. In our case, the property is “being a cluster graph,” and the forbidden induced subgraphs are  $P_3$ ’s (i.e., paths on three vertices). A trivial approximation algorithm of ratio 3 can be derived as follows. We search for induced  $P_3$ ’s, and we delete all its three vertices if one is found. This trivial upper bound is hitherto the best known. Indeed, this is a simple application of Lund and Yannakakis’s observation [18], which applies to all graph classes with finite forbidden induced subgraphs.

Closely related is the cluster editing problem, which allows us to use, instead of vertex deletions, both edge additions and deletions [3]. Approximation algorithms of the cluster editing problem have been intensively studied, and the current best approximation ratio is 2.5 [1, 2, 8]. Our main result is the first non-trivial approximation algorithm for the association set problem, with a ratio matching the best ratio of the closely related cluster editing problem. As usual,  $n$  and  $m$  denote the numbers of vertices and edges respectively in the input graph. Without loss of generality, we assume throughout the paper that the input graph contains no isolated vertices (vertices of degree 0), hence  $n = O(m)$ .

**Theorem 1.** *There is an  $O(mn)$ -time approximation algorithm of ratio 2.5 for the association set problem.*

Our approach is to reduce the association set problem to the weighted dissociation set problem. Given a vertex-weighted graph, the *weighted dissociation set* problem asks for a set of vertices with the minimum weight such that its deletion breaks all  $P_3$ ’s, thereby leaving a graph of maximum degree 1 or 0. This problem was first studied by Yannakakis [25], who proved that its unweighted version is already NP-hard on bipartite graphs. Note that a  $P_3$  that is not induced must be in a triangle. Thus, in triangle-free graphs, the weighted version of the association set problem is equivalent to the weighted dissociation set problem. It is easy to observe that for the association set problem, vertices in a twin class (i.e., whose vertices have the same closed neighborhood) are either fully contained in or disjoint from a minimum solution. This observation inspires us to transform the input graph  $G$  into a vertex-weighted graph  $Q$  by identifying each twin class of  $G$  with a vertex of  $Q$  whose weight is the size of the corresponding twin class. We further observe that there are five small graphs such that if  $G$  has none of them as an induced subgraph, then  $Q$  either has a simple structure, hence trivially solvable, or is triangle-free, and can be solved using the ratio-2 approximation algorithm for the weighted dissociation set problem [21, 22]. From the obtained solution for  $Q$  we can easily retrieve a solution for the original graph

$G$ . Since each of these five graphs has at most five vertices and at least two of them need to be deleted to make it free of induced  $P_3$ 's, the approximation ratio 2.5 follows.

The main idea of this paper appears in the argument justifying the reduction from the (unweighted) association set problem to the weighted dissociation set problem. Indeed, we are able to provide a stronger algorithmic result that implies the aforementioned combinatorial result. We develop an efficient algorithm that detects one of the five graphs in  $G$ , solves the problem completely, or determines that  $Q$  is already triangle-free. Our principal tool is modular decomposition. A similar use of modular decomposition was recently invented by the authors [17] in parameterized algorithms. It is worth noting that the basic observation on vertex deletion problems to graph properties with finite forbidden induced subgraphs has been used on both approximation and parameterized algorithms, by Lund and Yannakakis [18] and by Cai [6] respectively.

After a preliminary version of this work appeared in arxiv, Fiorini et al. [11] managed to further improve the ratio to  $7/3$ . The first part of their algorithm is similar as ours, with more small induced subgraphs taken into consideration, while their analysis, using the “local ratio” technique, is quite different from ours.

As a final remark, cluster editing has a  $2k$ -vertex kernel [7], while it remains an open problem to find a linear-vertex kernel for the association set (cluster vertex deletion) problem.

## 2 Preliminaries

This paper will be only concerned with undirected and simple graphs. The vertex set and edge set of a graph  $G$  are denoted by  $V(G)$  and  $E(G)$  respectively. For  $\ell \geq 3$ , let  $P_\ell$  and  $C_\ell$  denote respectively an induced path and an induced cycle on  $\ell$  vertices. A  $C_3$  is also called a *triangle*. For a given set  $\mathcal{F}$  of graphs, a graph  $G$  is  $\mathcal{F}$ -free if it contains no graph in  $\mathcal{F}$  as an induced subgraph. When  $\mathcal{F}$  consists of a single graph  $F$ , we use also  $F$ -free for short. For each vertex  $v$  in  $V(G)$ , its *neighborhood* and *closed neighborhood* are denoted by  $N_G(v)$  and  $N_G[v]$  respectively.

A subset  $M$  of vertices forms a *module* of  $G$  if all vertices in  $M$  have the same neighborhood outside  $M$ . In other words, for every pair of vertices  $u, v \in M$ , a vertex  $x \notin M$  is adjacent to  $u$  if and only if it is adjacent to  $v$  as well. The set  $V(G)$  and all singleton vertex sets are modules, called *trivial*. A graph on at least four vertices is *prime* if it contains only trivial modules, e.g., a  $P_4$  and a  $C_5$ . Given any partition  $\{M_1, \dots, M_p\}$  of  $V(G)$  such that  $M_i$  for every  $1 \leq i \leq p$  is a module of  $G$ , we can derive a  $p$ -vertex *quotient graph*  $Q$  such that for any pair of distinct  $i, j$  with  $1 \leq i, j \leq p$ , the  $i$ th and  $j$ th vertices of  $Q$  are adjacent if and only if  $M_i$  and  $M_j$  are adjacent in  $G$  (every vertex in  $M_i$  is adjacent to every vertex in  $M_j$ ). It should be noted that a single-vertex graph and  $G$  itself are both trivial quotient graphs of  $G$ , defined by the trivial module partitions  $\{V(G)\}$  and  $\{\{v_1\}, \dots, \{v_n\}\}$  respectively.

A module  $M$  is *strong* if for every other module  $M'$  that intersects  $M$ , one of  $M$  and  $M'$  is a proper subset of the other. All trivial modules are clearly strong. We say that a strong module  $M$  different from  $V(G)$  is *maximal* if the only strong module properly containing  $M$  is  $V(G)$ . (It can be contained by non-strong modules, e.g., in a graph that is a clique, the maximal strong modules are simply the singletons, while every subset of vertices is a module.) The set of maximal strong modules of  $G$  partitions  $V(G)$ , and defines a special quotient graph of  $G$ , denoted by  $\tilde{Q}(G)$ .<sup>1</sup> The reader who is unfamiliar with modular decomposition is referred to the survey of Habib and Paul [13] for more information. The following proposition will be crucial for our algorithm.

**Proposition 1.** [12, 20] *If a graph  $G$  is connected, then  $\tilde{Q}(G)$  is either a clique or prime. Any prime graph contains an induced  $P_4$ .*

Let  $Q$  be a quotient graph of  $G$ , and let  $M$  be a module of  $G$  in the module partition defining  $Q$ . By abuse of notation, we will also use  $M$  to denote the corresponding node of  $Q$ ; hence  $M \in V(Q)$  and  $M \subseteq V(G)$ , and its meaning will be clear from context. Accordingly, by  $N_G(M)$  we mean those vertices of  $G$  adjacent to  $M$  in  $G$ , and by  $N_Q(M)$  we mean those nodes of  $Q$  adjacent to  $M$  in  $Q$ —note that the union of those vertices of  $G$  represented by  $N_Q(M)$  is exactly  $N_G(M)$ . Sets  $N_G[M]$  and  $N_Q[M]$  are understood analogously.

The weighted versions of the associated set problem and the dissociation set problem are formally defined as follows.

Associated set

*Input:* A vertex-weighted graph  $G$ .

*Task:* find a subset  $X \subset V(G)$  of the minimum weight such that every component of  $G - X$  is a clique.

Dissociation set

*Input:* A vertex-weighted graph  $G$ .

*Task:* find a subset  $X \subset V(G)$  of the minimum weight such that every component of  $G - X$  is a single vertex or a single edge.

Let  $\text{asso}(G)$  and  $\text{diss}(G)$  denote respectively the weights of minimum association sets and minimum dissociation sets of a weighted graph  $G$ . It is routine to verify that  $\text{asso}(G) \leq \text{diss}(G)$ . Their gap can be arbitrarily large, e.g., if  $G$  is a clique on  $n$  vertices, then  $\text{asso}(G) = 0$  and  $\text{diss}(G) = n - 2$ . A vertex set  $X$  is an association set or a dissociation set of a graph  $G$  if and only if  $G - X$  contains no  $P_3$  as an induced subgraph or as a subgraph, respectively. The following proposition follows from the fact that every  $P_3$  in a  $C_3$ -free graph is induced.

<sup>1</sup> If  $G$  is a clique or an independent set, then  $\tilde{Q}(G)$  is isomorphic to  $G$  and is the largest quotient graph of  $G$ ; if  $\tilde{Q}(G)$  is prime, then it is the smallest *nontrivial* quotient graph of  $G$ , both cardinality-wise and inclusion-wise (see Lemma 5). Otherwise, there can be other quotient graphs larger or smaller than  $\tilde{Q}(G)$ .

**Proposition 2.** *If a graph  $G$  is  $C_3$ -free, then  $\text{asso}(G) = \text{diss}(G)$ .*

**Theorem 2** ([21,22]). *There is an  $O(mn)$ -time approximation algorithm of ratio 2 for the weighted dissociation set problem.*

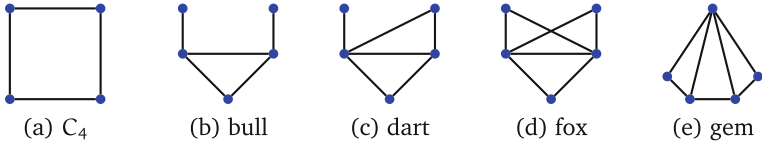
Note that an unweighted graph can be treated as a special weighted graph where every vertex receives a unit weight. In this case,  $\text{asso}(G)$  is the same as the cardinality of the minimum association set of  $G$ .

A  $\{C_4, P_4\}$ -free graph is called a *trivially perfect graph*. A vertex is *universal* if it is adjacent to all other vertices in this graph, i.e., has degree  $n - 1$ . It is easy to verify that each universal vertex is a maximal strong module of the graph.

**Proposition 3** ([14,23,24]). *Every connected trivially perfect graph has a universal vertex. One can in  $O(m)$ -time either decide that a graph is a trivially perfect graph, or detect an induced  $P_4$  or  $C_4$ .*

### 3 The Approximation Algorithm

The association set problem admits a naive 3-approximation algorithm [18]. It finds an induced  $P_3$  and deletes from  $G$  all the three vertices in this  $P_3$ , and repeats. Since any minimum association set has to contain some of the three vertices, the approximation ratio is at most 3. A  $P_3$  can be found in linear time, while the process can be repeated at most  $n/3$  times, and thus the algorithm can be implemented in time  $O(mn)$ . We present here a very simple 2.5-approximation algorithm, which runs in a high-order polynomial time, and we will show in the next section how to implement it in an efficient way to achieve the running time claimed in Theorem 1.



**Fig. 1.** Small subgraphs on 4 or 5 vertices.

Let  $\mathcal{F}$  denote the set of five small graphs depicted in Fig. 1, i.e.,  $\{C_4, \text{bull}, \text{dart}, \text{fox}, \text{gem}\}$ . A quick glance of Fig. 1 convinces us that from each induced subgraph in  $\mathcal{F}$ , at least two vertices need to be deleted to make it  $P_3$ -free.

**Proposition 4.** *Let  $X \subseteq V(G)$ . If  $G[X] \in \mathcal{F}$ , then  $\text{asso}(G - X) \leq \text{asso}(G) - 2$ .*

In polynomial time we can decide whether  $G$  contains an induced subgraph in  $\mathcal{F}$ , and find one if it exists. We delete all its vertices if it is found. If  $G$  is not connected, then we work on its components one by one. In the rest of this section we may focus on connected  $\mathcal{F}$ -free graphs. In such a graph, every

nontrivial module  $M$  induces a  $\{C_4, P_4\}$ -free subgraph: A  $P_4$  in  $G[M]$ , together with any  $v \in N_G(M)$  (it exists because  $G$  is connected and  $M$  is nontrivial), makes a gem.

One may use the definition of modular decomposition to derive the following combinatorial properties of  $\mathcal{F}$ -free graphs. Since we will present a stronger result in the next section that implies this lemma, its proof is omitted here.

**Lemma 1.** *Let  $G$  be an  $\mathcal{F}$ -free graph that is not a clique, and let  $Q = \tilde{Q}(G)$ . Either  $G$  consists of a set of universal vertices and two disjoint cliques, or  $Q$  is  $C_3$ -free and the following hold for every maximal strong module  $M$  of  $G$ :*

- (1) *The subgraph  $G[M]$  is a cluster graph. If it is not a clique, then  $|N_G(M)| = 1$ .*
- (2) *If  $|N_G(M)| > 2$ , then the module  $M$  is trivial (consisting of a single vertex of  $G$ ).*

In the first case,  $G$  has simply two intersecting cliques  $C_1$  and  $C_2$ , and the problem is trivial: We delete either  $C_1 \cap C_2$  (i.e., all universal vertices), or one of  $C_1 \setminus C_2$  and  $C_2 \setminus C_1$ , whichever is smaller. Therefore, we focus on the other case where  $\tilde{Q}(G)$  is  $C_3$ -free. If some maximal strong module  $M$  does not induce a clique in a connected  $\mathcal{F}$ -free graph  $G$ , then we can delete the unique neighbor of  $M$  and consider the smaller graph  $G - N_G[M]$ . Now that  $G$  is not a clique but every maximal strong module  $M$  of  $G$  is, we can define a vertex-weighted graph  $Q$  isomorphic to the quotient graph  $\tilde{Q}(G)$ , where the weight of each vertex in  $Q$  is the number of vertices in the corresponding module, i.e.,  $|M|$ . We apply the algorithm of Tu and Zhou [21] to find a dissociation set of this weighted graph  $Q$ . Since  $Q$  is  $C_3$ -free, by Proposition 2 and Theorem 2, the total weight of the obtained dissociation set is at most  $2\text{diss}(Q) = 2\text{asso}(Q) = 2\text{asso}(G)$ . Putting together these steps, an approximation algorithm with ratio 2.5 follows (see Fig. 2).

**Theorem 3.** *The output of algorithm APPROX-ASSO( $G$ ) is an association set of the input graph  $G$  and its size is at most  $2.5\text{asso}(G)$ .*

## 4 An Efficient Implementation

We now discuss the implementation issues that lead to the claimed running time. A simpleminded implementation of the algorithm given in Fig. 2 takes  $O(n^6)$  time, which is decided by the disposal of induced subgraphs in  $\mathcal{F}$  (step 3). It is unclear to us how to detect them in a more efficient way than the  $O(n^5)$ -time enumeration. But we observe that what we need are no more than the conditions stipulated in Lemma 1, for which being  $\mathcal{F}$ -free is sufficient but not necessary. The following relaxation is sufficient for our algorithmic purpose: We either detect an induced subgraph in  $\mathcal{F}$  or determine that  $G$  has already satisfied these conditions. Once a subgraph is found, we can delete all its vertices and repeat the process. In summary, we are after an  $O(mn)$ -time procedure that finds a set of subgraphs in  $\mathcal{F}$  such that its deletion leaves a graph satisfying the conditions of Lemma 1.

**Algorithm** APPROX-ASSO( $G$ )INPUT: a graph  $G$ .OUTPUT: a set  $X$  of vertices such that  $G - X$  is a cluster.

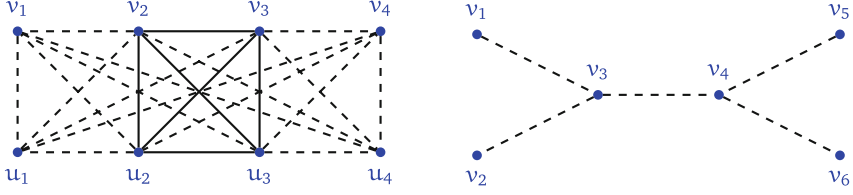
1. if  $G$  is a cluster graph **return**  $\emptyset$ ;
2. if  $G$  is not connected **then**  
     **return**  $\bigcup \{\text{APPROX-ASSO}(C) : C \text{ is a component of } G\}$ ;
3. if there exists  $X$  such that  $G[X] \in \mathcal{F}$  **then**  
     **return**  $\text{APPROX-ASSO}(G - X) \cup X$ ;
4. if  $G$  consists of two intersecting maximal cliques  $C_1$  and  $C_2$  **then**  
     **return** the smaller of  $C_1 \cap C_2$ ,  $C_1 \setminus C_2$ , and  $C_2 \setminus C_1$ ;
5. find all maximal strong modules of  $G$  and build  $\tilde{Q}(G)$ ;
6. if a maximal strong module  $M$  is not a clique **then**  
     **return**  $\text{APPROX-ASSO}(G - N_G[M]) \cup N_G(M)$ ;
7. define a weighted graph  $Q$ , and call the algorithm of [21] with it;  
     **return** vertices of  $G$  corresponding to its output.

**Fig. 2.** Outline of the approximation algorithm for association set.

Toward this end a particular obstacle is the  $C_3$ -free condition in the second case of Lemma 1. Indeed, the detection of triangles in linear time is a notorious open problem that we are not able to solve. Therefore, we may have to abandon the simple “search and remove” approach. The first idea here is that we may dispose of *all* triangles of  $\tilde{Q}(G)$  in  $O(mn)$  time. This is, however, still not sufficient, because after deleting a set  $X$  of some vertices, its maximal strong modules change, and more importantly,  $\tilde{Q}(G - X)$  may *not* be an (induced) subgraph of  $\tilde{Q}(G)$ ; see, e.g., Fig. 3. Our observation is that  $\tilde{Q}(G - X)$  is either a clique, an independent set, or an induced subgraph of  $\tilde{Q}(G[M])$  for some (not necessarily maximal) strong module  $M$  of  $G$ .

We start from recalling some simple facts about modular decomposition. For each maximal strong module  $M$  of  $G$ , we can further take the maximal strong modules and the quotient graph  $\tilde{Q}(G[M])$ . This process can be recursively applied until every module consists of a single vertex. If we represent each module used in this process as a node, and add edges connecting every  $M$  with all maximal strong modules of  $G[M]$ , we obtain a tree rooted at  $V(G)$ , called the *modular decomposition tree* of  $G$ . The nodes of the modular decomposition tree are precisely all strong modules of  $G$ , where the leaves are all singleton vertex sets, and for every non-leaf node  $M$ , its children are the maximal strong modules of  $G[M]$  [12]. It is known that the modular decomposition tree can be constructed in linear time [19].

**Proposition 5.** *If  $\tilde{Q}(G)$  is prime, then every nontrivial quotient graph of  $G$  contains  $\tilde{Q}(G)$  as an induced subgraph.*



(a) For  $U = \{v_2, v_3, u_2, u_3\}$ ,  $\tilde{Q}(G[U])$  is a clique, while  $\tilde{Q}(G)$  is an edge.

(b) For  $U = \{v_1, v_2, v_5, v_6\}$ ,  $\tilde{Q}(G[U])$  is an independent set, while  $\tilde{Q}(G)$  is a  $P_4$ .

**Fig. 3.**  $\tilde{Q}(G[U])$  may not be an induced subgraph of  $\tilde{Q}(G)$ .

On the one hand, since  $V(G)$  itself is a strong module of  $G$ , every vertex set  $U \subseteq V(G)$  is contained in some strong module. On the other hand, since two strong modules are either disjoint or one containing the other, there is a unique one that is inclusion-wise minimal of all strong modules containing  $U$ .

**Theorem 4.** *Let  $U \subseteq V(G)$  be a subset of vertices of  $G$ , and let  $M$  be the inclusion-wise minimal strong module of  $G$  that contains  $U$ . If  $\tilde{Q}(G[U])$  is prime, then it is a subgraph of  $\tilde{Q}(G[M])$  induced by those maximal strong modules of  $G[M]$  that intersect  $U$ .*

We remark that if  $\tilde{Q}(G[U])$  is a clique or independent set, then it is not necessarily an induced subgraph of  $\tilde{Q}(G[M])$ ; see, e.g., Fig. 3.

We are now ready to present the efficient implementation for the first phase, which would replace the first three steps of algorithm APPROX-ASSO (Fig. 2).

**Lemma 2.** *In  $O(mn)$  time we can find a set  $\mathcal{H}$  of disjoint induced subgraphs of  $G$  such that each  $H \in \mathcal{H}$  is in  $\mathcal{F}$  and  $G - \bigcup_{H \in \mathcal{H}} V(H)$  satisfies the conditions of Lemma 1.*

*Proof.* We use the procedure described in Fig. 4. Step 0 is trivial. Step 1 uses the algorithm of McConnell and Spinrad [19], and step 2 uses simple enumeration, i.e., for each edge  $uv$ , we find all the common neighbors of  $u$  and  $v$ , which can be done in time  $O(nm)$ . This leads the disposal of triangles in step 3. During its progress, a maximal strong module  $M$  of the input graph  $G$  may not remain a maximal strong module of the current graph (i.e.,  $G - X$ ). But if  $M$  is not completely deleted (i.e.,  $M \not\subseteq X$ ), then its remnant (i.e.,  $M \setminus X$ ) is always a module of  $G - X$ .

Note that the three modules in each triangle must have the same parent in the modular decomposition tree. For each triangle  $\{M_1, M_2, M_3\}$ , we focus on their parent  $M$  (in the modular decomposition tree) and the subgraph  $G[M]$  (step 3.1). All the modules mentioned in steps 3.2–3.7 are maximal strong modules of subgraph  $G[M]$ ; they correspond to  $V(Q)$ . If either of the conditions of steps 3.2 and 3.3 is true, then the triangle has been disposed of and we can continue to the next one. If the deletion of vertices in previous iterations has

**Procedure** REDUCE( $G$ )  
 INPUT: a graph  $G$ .  
 OUTPUT: a set of vertices specified in Lemma 2.

0.  $X \leftarrow \emptyset$ ; if  $G$  is a clique **then return**  $\emptyset$ ;
1. build the modular decomposition tree of  $G$ ;
2. find all triangles  $\mathcal{C}$  of  $\tilde{Q}(G[M])$  for all strong modules  $M$  of  $G$ ;
3. **while**  $\mathcal{C}$  is nonempty **do**
- 3.0. take a triangle  $\{M_1, M_2, M_3\}$  from  $\mathcal{C}$ ;
- 3.1. let  $M$  be the parent of  $\{M_1, M_2, M_3\}$  and let  $Q = \tilde{Q}(G[M])$ ;
- 3.2. **if** there is  $1 \leq i \leq 3$  such that  $M_i \subseteq X$  **then**  
     delete  $\{M_1, M_2, M_3\}$  from  $\mathcal{C}$ ; **continue**;
- 3.3. **if** there are  $1 \leq i < j \leq 3$  such that  $M_i, M_j$  were merged **then**  
     delete  $\{M_1, M_2, M_3\}$  from  $\mathcal{C}$ ; **continue**;
- 3.4. **if** there are  $1 \leq i < j \leq 3$  such that  $N_Q[M_i] = N_Q[M_j]$  **then**  
     merge  $M_i$  and  $M_j$  into a single module;  
     delete  $\{M_1, M_2, M_3\}$  from  $\mathcal{C}$ ; **continue**;
- 3.5. find  $M' \in V(Q)$  that is adjacent to only one of  $M_1$  and  $M_2$ ;  
      $\parallel$  Assume that  $M'$  is adjacent to  $M_2$  but not  $M_1$ .
- 3.6. **if**  $M'$  is adjacent to  $M_3$  **then**  
     find  $M'' \in V(Q)$  adjacent to only one of  $M_2$  and  $M_3$ ;  
     find gems or darts or  $C_4$  and move their vertices to  $X$ ;
- 3.7. **else**  $\parallel M'$  is nonadjacent to  $M_3$ .  
     find  $M'' \in V(Q)$  adjacent to only one of  $M_1$  and  $M_3$ ;  
     find darts or bulls or  $C_4$  or gem and move their vertices to  $X$ ;
4. **repeat** until  $X$  is unchanged **do**
- 4.0.  $G \leftarrow G - X$ ;  $\parallel$  All modules below are maximal strong modules of  $G$ .
- 4.1. **if**  $G$  is not connected **then**  
     **return**  $\bigcup \{\text{REDUCE}(C) : C \text{ is a component of } G\} \cup X$ ;
- 4.2. **if**  $G$  is a clique or consists of two intersecting cliques **then return**  $X$ ;
- 4.3. let  $Q$  denote  $\tilde{Q}(G)$ ;  $\parallel Q$  is a clique or  $C_3$ -free.
- 4.4. **if** two non-clique modules are adjacent in  $Q$  **then**  
     find a  $C_4$  of  $G$  and move its vertices to  $X$ ;
- 4.5. **else if**  $Q$  is a clique but not an edge **then**
- 4.5.0. let  $M$  be the only nontrivial module;  $\parallel$  It exists because  $G$  is not a clique.
- 4.5.1. **if**  $G[M]$  has a  $P_4$  or  $C_4$  **then**  
     find a gem or  $C_4$  of  $G$  and move its vertices to  $X$ ;
- 4.5.2. **else if**  $G[M]$  has two components **then**  
     find a dart of  $G$  and move its vertices to  $X$ ;
- 4.5.3. **else** find a fox of  $G$  and move its vertices to  $X$ ;
- 4.6. **else if** a module  $M$  is not a cluster **then**
- 4.6.1. **if**  $G[M]$  has a  $P_4$  or  $C_4$  **then**  
     find a gem or  $C_4$  of  $G$  and move its vertices to  $X$ ;
- 4.6.2. **else if**  $G[M]$  is not connected **then**  
     find a dart of  $G$  and move its vertices to  $X$ ;
- 4.6.3. **else** find a dart of  $G$  and add its vertices to  $X$ ;  $\parallel Q$  is not a clique.
- 4.7. **else if** a non-clique module  $M$  has two neighbors  $M_1, M_2$  **then**  
     find a  $C_4$  of  $G$  and move its vertices to  $X$ ;
- 4.8. **else if** a nontrivial module  $M$  has neighbors  $M_1, M_2, M_3$  **then**  
     find a fox of  $G$  and move its vertices to  $X$ ;
5. **return**  $X$ .

**Fig. 4.** Procedure for the first phase.

made  $N_Q[M_i] = N_Q[M_j]$  for some  $1 \leq i < j \leq 3$ , then  $M_i \cup M_j$  is a module of  $G[M \setminus X]$ . Note that after they are merged, both  $M_i$  and  $M_j$  refer to the new module. Now that the procedure has passed steps 3.2–3.4, for each  $1 \leq i < j \leq 3$ , we can find a module adjacent to only one of  $M_i$  and  $M_j$ . This justifies step 3.5, and we may assume without loss of generality that the module  $M'$  is adjacent to  $M_2$  but not  $M_1$ ; the other case can be dealt with a symmetric way, which is omitted. In step 3.6, depending on the adjacency between  $M''$  and  $M_1, M'$ , we are in one of the following three cases: – if  $M''$  is adjacent to neither of  $M_1, M'$ , then there is a dart; – if  $M''$  is adjacent to precisely one of  $M_1, M'$ , then there is a gem; or – otherwise ( $M''$  is adjacent to both of  $M_1, M'$ ), there is a  $C_4$ ;

This forbidden subgraph can be constructed by taking one vertex from each of  $M_1, M_2, M_3, M'$ , and  $M''$ . We can actually find  $\min\{|M_1|, |M_2|, |M_3|, |M'|, |M''|\}$  number of gems or darts, or  $\min\{|M_1|, |M_3|, |M'|, |M''|\}$  number of  $C_4$ 's, which we all move into  $X$ . It is similar for step 3.7.

After step 3,  $G$  might become disconnected. Then we work on its components one by one. Steps 4.1 and 4.2 are simple. The fact that the quotient graph  $Q$  built in step 4.3 is either a clique or is  $C_3$ -free can be argued using Theorem 4. Suppose for contradiction that  $Q$  is not a clique but contains a  $C_3$ ; by Proposition 1,  $Q$  is prime. Then by Theorem 4,  $Q$  is a subgraph of  $\tilde{Q}(G[M])$  for some strong module  $M$  of  $G$ . Let  $\{M_1, M_2, M_3\}$  be the triangle of  $\tilde{Q}(G[M])$  corresponding to a triangle in  $Q$ . But in step 3, either one of  $\{M_1, M_2, M_3\}$  has been completely put into  $X$ , or two of them have been merged (then unless  $Q$  is a clique or an independent set, they will always be in the same maximal strong module). Therefore,  $Q$  must be  $C_3$ -free if it is not a clique.

Note that the algorithm enters at most one of steps 4.4–4.8. The correctness of step 4.4 is clear. If  $Q$  is a clique and it passes step 4.4, then all but one maximal strong module are trivial: Recall that each universal vertex is a maximal strong module. A  $P_4$  of  $M$  together with a vertex in  $N_G(M)$  makes a gem (4.5.1). Now that  $G[M]$  is  $\{P_4, C_4\}$ -free, and has no universal vertex (a universal vertex of  $G[M]$  is a universal vertex of  $G$  as well), according to Proposition 3,  $G[M]$  is disconnected. Since step 4.2 does not apply, in step 4.5.2, at least one component is not a clique, and has a  $P_3$ , which, together with a vertex  $u \in N_G(M)$  and any vertex from another component of  $G[M]$ , makes a dart. Otherwise (step 4.5.3),  $G[M]$  has at least three components, and we can find a fox by taking three vertices from different components of  $G[M]$  and two vertices from  $N_G(M)$ : Recall that when it enters step 4.5,  $G$  must have at least two universal vertices. In step 4.6,  $Q$  is not a clique, and assume that there is a module  $M$  that is not a cluster, then it must be  $\{P_4, C_4\}$ -free since the same reason as step 4.5.1. Now that if  $M$  is not connected, then a  $P_3$  can be found in some component, which, together with some vertex in some other component of  $M$  and a vertex in  $N_G(M)$ , forms a dart. Therefore,  $M$  is connected and contains a  $P_3$ . It is sure that  $N_G(N[M])$  is not empty since  $Q$  is not a clique, thus a dart can be found: Recall that if there are only two modules then  $G[M]$  cannot have universal vertices. After step 4.6,  $Q$  is always  $C_3$ -free. Therefore, modules  $M_1, M_2$  in step 4.7 and modules  $M_1, M_2, M_3$  in step 4.8 are (pairwise) nonadjacent.

If  $G$  consists of two intersecting cliques, the procedure returns at step 4.2. Hence we may assume that it is not the case. The quotient graph  $\bar{Q}(G)$  is  $C_3$ -free because Theorem 4 and the algorithm has passed step 4.5. Conditions (1) and (2) of Lemma 1 follow from the correctness argument for steps 4.6–4.8. We now calculate the running time of the procedure. Note that the total number of edges of the subgraphs induced the strong modules of  $G$  is upper bounded by  $m$ . Thus, all the triangles can be listed in  $O(mn)$  time in step 2. Each iteration of step 3 takes  $O(m)$  time, and it decreases the order of  $Q$  by at least one, and thus step 3 takes  $O(mn)$  time in total. Each iteration of step 4 takes  $O(m)$  time, and it decreases the order of  $G$  by at least one, and hence step 4 takes  $O(mn)$  time in total. This concludes the proof of this lemma.  $\square$

## References

1. Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: ranking and clustering. *J. ACM* **55**(5), (Article 23) 1–27 (2008). doi:[10.1145/1411509.1411513](https://doi.org/10.1145/1411509.1411513). A preliminary version appeared in STOC 2005
2. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. *Mach. Learn.* **56**(1), 89–113 (2004). doi:[10.1023/B:MACH.0000033116.57574.95](https://doi.org/10.1023/B:MACH.0000033116.57574.95). A preliminary version appeared in FOCS 2002
3. Ben-Dor, A., Shamir, R., Yakhini, Z.: Clustering gene expression patterns. *J. Comput. Biol.* **6**(3/4), 281–297 (1999). doi:[10.1089/106652799318274](https://doi.org/10.1089/106652799318274)
4. Boral, A., Cygan, M., Kociumaka, T., Pilipczuk, M.: A fast branching algorithm for cluster vertex deletion. *Theory Comput. Syst.* **58**(2), 357–376 (2016). doi:[10.1007/s00224-015-9631-7](https://doi.org/10.1007/s00224-015-9631-7)
5. Bruhn, H., Chopin, M., Joos, F., Schaudt, O.: Structural parameterizations for boxicity. *Algorithmica* **74**(4), 1453–1472 (2016). doi:[10.1007/s00453-015-0011-0](https://doi.org/10.1007/s00453-015-0011-0). A preliminary version appeared in WG 2014
6. Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.* **58**(4), 171–176 (1996). doi:[10.1016/0020-0190\(96\)00050-6](https://doi.org/10.1016/0020-0190(96)00050-6)
7. Cao, Y., Chen, J.: Cluster editing: kernelization based on edge cuts. *Algorithmica* **64**(1), 152–169 (2012). doi:[10.1007/s00453-011-9595-1](https://doi.org/10.1007/s00453-011-9595-1)
8. Charikar, M., Guruswami, V., Wirth, A.: Clustering with qualitative information. *J. Comput. Syst. Sci.* **71**(3), 360–383 (2005). doi:[10.1016/j.jcss.2004.10.012](https://doi.org/10.1016/j.jcss.2004.10.012). A preliminary version appeared in FOCS 2003
9. Chopin, M., Nichterlein, A., Niedermeier, R., Weller, M.: Constant thresholds can make target set selection tractable. *Theory Comput. Syst.* **55**(1), 61–83 (2014). doi:[10.1007/s00224-013-9499-3](https://doi.org/10.1007/s00224-013-9499-3)
10. Doucha, M., Kratochvíl, J.: Cluster vertex deletion: a parameterization between vertex cover and clique-width. In: Rován, B., Sassone, V., Widmayer, P. (eds.) MFCS 2012. LNCS, vol. 7464, pp. 348–359. Springer, Heidelberg (2012)
11. Fiorini, S., Joret, G., Schaudt, O.: Improved approximation algorithms for hitting 3-vertex paths. In: Louveaux, Q., Skutella, M. (eds.) IPCO 2016. LNCS, vol. 9682, pp. 238–249. Springer, Heidelberg (2016). doi:[10.1007/978-3-319-33461-5\\_20](https://doi.org/10.1007/978-3-319-33461-5_20)
12. Gallai, T.: Transitiv orientierbare graphen. *Acta Mathematica Academiae Scientiarum Hungaricae* **18**, 25–66 (1967). (Trans: Maffray, F., Preissmann, M.: Perfect Graphs. In: Ramírez-Alfonsín, J.L., Reed, B.A. (eds.), pp. 25–66. Wiley (2001). doi:[10.1007/BF02020961](https://doi.org/10.1007/BF02020961)

13. Habib, M., Paul, C.: A survey of the algorithmic aspects of modular decomposition. *Comput. Sci. Rev.* **4**(1), 41–59 (2010). doi:[10.1016/j.cosrev.2010.01.001](https://doi.org/10.1016/j.cosrev.2010.01.001)
14. Heggernes, P., Kratsch, D.: Linear-time certifying recognition algorithms and forbidden induced subgraphs. *Nord. J. Comput.* **14**(1–2), 87–108 (2007)
15. Hüffner, F., Komusiewicz, C., Moser, H., Niedermeier, R.: Fixed-parameter algorithms for cluster vertex deletion. *Theory Comput. Syst.* **47**(1), 196–217 (2010). doi:[10.1007/s00224-008-9150-x](https://doi.org/10.1007/s00224-008-9150-x)
16. Lewis, J.M., Yannakakis, M.: The node-deletion problem for hereditary properties is NP-complete. *J. Comput. Syst. Sci.* **20**(2), 219–230 (1980). doi:[10.1016/0022-0000\(80\)90060-4](https://doi.org/10.1016/0022-0000(80)90060-4). Preliminary versions independently presented in STOC 1978
17. Liu, Y., Wang, J., You, J., Chen, J., Cao, Y.: Edge deletion problems: branching facilitated by modular decomposition. *Theoret. Comput. Sci.* **573**, 63–70 (2015). doi:[10.1016/j.tcs.2015.01.049](https://doi.org/10.1016/j.tcs.2015.01.049)
18. Lund, C., Yannakakis, M.: The approximation of maximum subgraph problems. In: Lingas, A., Karlsson, R.G., Carlsson, S. (eds.) *Automata, Languages, Programming (ICALP)*. LNCS, vol. 700, pp. 40–51. Springer, Heidelberg (1993). doi:[10.1007/3-540-56939-1\\_60](https://doi.org/10.1007/3-540-56939-1_60)
19. McConnell, R.M., Spinrad, J.P.: Modular decomposition, transitive orientation. *Discrete Math.* **201**(1–3), 189–241 (1999). doi:[10.1016/S0012-365X\(98\)00319-7](https://doi.org/10.1016/S0012-365X(98)00319-7). Preliminary versions appeared in SODA 1994 and SODA 1997
20. Sumner, D.P.: Graphs indecomposable with respect to the X-join. *Discrete Math.* **6**(3), 281–298 (1973). doi:[10.1016/0012-365X\(73\)90100-3](https://doi.org/10.1016/0012-365X(73)90100-3)
21. Tu, J., Zhou, W.: A factor 2 approximation algorithm for the vertex cover  $P_3$  problem. *Inf. Process. Lett.* **111**(14), 683–686 (2011). doi:[10.1016/j.ipl.2011.04.009](https://doi.org/10.1016/j.ipl.2011.04.009)
22. Tu, J., Zhou, W.: A primal-dual approximation algorithm for the vertex cover  $P_3$  problem. *Theoret. Comput. Sci.* **412**(50), 7044–7048 (2011). doi:[10.1016/j.tcs.2011.09.013](https://doi.org/10.1016/j.tcs.2011.09.013)
23. Wolk, E.S.: The comparability graph of a tree. *Proc. Am. Math. Soc.* **13**, 789–795 (1962). doi:[10.1090/S0002-9939-1962-0172273-0](https://doi.org/10.1090/S0002-9939-1962-0172273-0)
24. Yan, J.-H., Chen, J.-J., Chang, G.J.: Quasi-threshold graphs. *Discrete Appl. Math.* **69**(3), 247–255 (1996). doi:[10.1016/0166-218X\(96\)00094-7](https://doi.org/10.1016/0166-218X(96)00094-7)
25. Yannakakis, M.: Node-deletion problems on bipartite graphs. *SIAM J. Comput.* **10**(2), 310–327 (1981). doi:[10.1137/0210022](https://doi.org/10.1137/0210022)

Graph-Theoretic Concepts in Computer Science  
42nd International Workshop, WG 2016, Istanbul,  
Turkey, June 22-24, 2016, Revised Selected Papers  
Heggernes, P. (Ed.)  
2016, X, 307 p. 51 illus., Softcover  
ISBN: 978-3-662-53535-6