A Polynomial-time Algorithm for Cluster Deletion on (Diamond, Butterfly)-free Graphs

Sabrine Malek [1] Wady Naanaa [2]
Faculty of Economics and Management of Sfax - Tunis [1]
National Engineering School of Tunis - Tunis [2]
LIMTIC laboratory - ISI Ariana, Tunis [1,2]

sabrine.malek@gmail.com [1] wady.naanaa@fsm.rnu.tn [2]

Abstract

The cluster deletion (CD) problem consists in transforming an input graph into a disjoint union of cliques by removing as few edges as possible. For general graphs, this is a combinatorial optimisation problem that belongs to the NP-hard computational complexity class.

In the present paper, we identify a new polynomially solvable CD subproblem. Specifically, we propose a two-phase polynomial algorithm that solves CD on (butterfly,diamond)-free graphs.

Keywords Cluster Deletion (CD); maximal clique; (butterfly,diamond)-free graphs

1 Introduction

Clustering is a central task that can be helpful for data analysis and graph mining [10]. From a graph-theoretic point of view, a cluster graph is a vertex-disjoint union of cliques [5] or, alternatively, a graph which does not contain any induced P_3 subgraph, where P_3 denotes a path composed of three vertices and two edges. Clustering problems consist in making the fewest changes to the vertex and/or to the edge set of an input graph so as to obtain a cluster graph. There exist several clustering variants, including Cluster Completion (CC), Cluster Deletion (CD) and Cluster Editing (CE) [5, 15]. In the cluster completion variant, edges can only be added

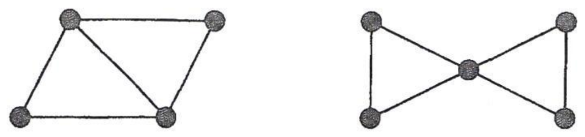
Figure 2: Diamond and Butterfly graphs.

An optimal solution for CD is, therefore, obtained by removing, from the initial graph, as few edges as possible in order to obtain a P_3-free graph (or cluster graph). CD belongs, therefore, to the class of edge modification problems, where one has to minimally change the edge set of a graph so as to satisfy a certain property [7].

3 (butterfly,diamond)-free CD is polynomially solvable

In this section, we identify a new class of graphs on which CD can be solved in polynomial time. This is the class of (butterfly,diamond)-free graphs, which contains all graphs that do not admit the two graphs shown in Figure 2 as induced subgaphs. We propose an algorithm that efficiently solves CD for (butterfly,diamond)-free graphs based on the tractability of the maximal cliques enumeration problem on the super-class of diamond-free graphs. Indeed, a crucial property of the diamond-free class is that the maximal cliques of the graphs in this class can be enumerated is polynomial time [8, 9]. In what follows, we use this result to show that CD is tractable on graphs belonging to the (butterfly,diamond)-free class.

As it has been shown in [12], a graph is diamond-free if and only if it has the *one-edge-overlap* property, which ensures that every edge of the graph is contained in, at most, one maximal clique. Starting from this property, we show the following lemma:

Lemma 1. *Let G be a (butterfly,diamond)-free graph, then any vertex of G appears in, at most, one maximal clique of G whose size is three or more.*

Proof. Let C and C' be two maximal cliques in G having a cardinality of three or more. Suppose that there exists a vertex, or more, which appear in both C and C', and proceed to get a contradiction. We distinguish the following two cases:

- $|C \cap C'| = 1$: this means that the maximal cliques C and C' share exactly one vertex, say v. Since C and C' have a size of three or more, each should contain at least, two other vertices. Assume, therefore, that $\{t, u, v\} \subseteq C$ and $\{v, w, x\} \subseteq C'$. Since C and C' are maximal cliques, there must exist $y \in C$ such that x-$y \notin E$. It follows that $(\{t, y, v, w, x\}, E(\{t, y, v, w, x\}))$ has an induced butterfly or an induced diamond, which contradicts the hypothesis.

- $|C \cap C'| \geq 2$: this means that C and C' share, at least, two common vertices, say v and w. Moreover, we have v-$w \in E$, which means that C and C' share a common edge. This contradicts the *one-edge-overlap* property for diamond-free graphs (see Proposition 1 of [12]).

□

Solving CD on a (butterfly,diamond)-free graph can be split into two phases. The first phase consists in calculating all maximal cliques of G having size three or more. By Lemma 3, the number of such maximal cliques in a (butterfly,diamond)-free graph with vertex set V cannot exceed $|V|$. Furthermore, we show that every CD instance admits a solution that comprises all maximal cliques having size three or more.

Lemma 2. *Let G be a (butterfly,diamond)-free graph. Then there exists a CD solution for G that contains all cliques of size three or more.*

Proof. Let $G^* = (V, E^*)$ be a CD solution for G. Suppose that G^* does not contain all maximal cliques of G that have size three or more. To prove the lemma, we show how to derive a CD solution for G that contains one more maximal clique, having size three or more, than G^*.

Let C be a maximal clique of G, with $|C| \geq 3$. Assume that C is not a clique of G^*. This implies that $E(C) \nsubseteq E^*$, where $E(C)$ denotes all the edges whose both endpoints are in C. If $E(C)$ is not entirely included in E^*, then C may be uniquely bi-partitioned into two subsets C_{in} and C_{out} such that C_{in} is the smallest subset of C that verifies the following

$$E(C_{in}) = E(C) \cap E^* \qquad \text{and} \qquad E(C_{out}) \cap E^* = \varnothing \tag{3.1}$$

This choice of C_{in} and C_{out} implies the following two facts: (i) If C_{in} is not empty then it should contain, at least, two vertices. (ii) C_{out} is not empty, because otherwise, $E(C)$ would be entirely included in E^*, which contradicts the assumption that C is not a clique of G^*.

This choice of C_{in} and C_{out} implies that any edge with one endpoint in C_{in} and the other one in C_{out} cannot be in E^*, because otherwise, either C would not be a clique or $E(C_{in}) \neq E(C) \cap E^*$.

The last step consists in proving that G^* is an optimal solution. Suppose there exist a spanning P_3-free subgraph, $G' = (V, E')$, of G such that $|E'| > |\hat{E} \cup M|$. By Lemma 2, G' must contain all cliques of G having size three or more, i.e., the C_i's. It follows that E' contains all the edges involved in these cliques, i.e., the elements of $\hat{E}$. Then, we can write $E' = \hat{E} \cup M'$. Moreover, the elements of M' are edges of the subgraph of G induced by $V \backslash \hat{C}$. Indeed, the endpoints of every edge e' of M' must be vertices of $V \backslash \hat{C}$, because, otherwise, e' will intersect with a C_i, and this contradicts the fact that G' is a P_3-free graph. Moreover, the edges of M' must be pairwise non incident edges, because these edges cannot form any clique of size three or more and G' is a P_3-free graph. It follows that M' is a matching of the subgraph of G induced by $V \backslash \hat{C}$. This implies that $|M'| \leq | M|$, since M is a maximum matching of the same subgraph, that is, the subgraph of G induced by $V \backslash \hat{C}$. This results in a contradiction with the hypothesis.

As a last step of the proof, we show that G^* can be constructed in polynomial time. It is well established that the problem of enumerating all maximal cliques in a simple graph can be achieved in $O(|V|^2\kappa)$, where κ denotes the number of maximal cliques in G. By the *one-edge-overlap* property, κ cannot exceed $|E|$ in (diamond)-free graphs. It follows that the cliques of G can be extracted in $O(|V|^2|E|)$. In addition, there is a $O(\sqrt{|V|}|E|)$ time algorithm that finds a maximum matching in any graph [16]. It follows that CD is polynomial-time solvable for the class of (butterfly,diamond)-free graphs. □

The first phase comprised in the two-phase algorithm requires finding all maximal cliques. To achieve this task, we used the *BronKerbosch algorithm* [8], which is one of the most successful maximal clique enumeration algorithms. It is a simple backtracking procedure that recursively enumerates all cliques whose size is bounded by a parameter of the algorithm [8, 11]. The complexity of the BronKerbosch algorithm is polynomial in the number of maximal cliques which is in turn bounded by the number of edges in (butterfly,diamond)-free graphs.

In turn, the second phase of the proposed algorithm can be achieved in $O(\sqrt{|V|}|E|)$ steps by executing the maximum matching algorithm proposed in [16].

Example 1. *Figure 4 illustrates how the two-phase algorithm operates on the graph of Figure 3. It is easy to see that this graph is (butterfly,diamond)-free. In the first phase, four maximal cliques having size three are detected. In the second phase, starting from the residual graph, a maximum matching is calculated. The CD solution is therefore obtained by putting together the cliques obtained in the first phase with those comprised in the maximum*

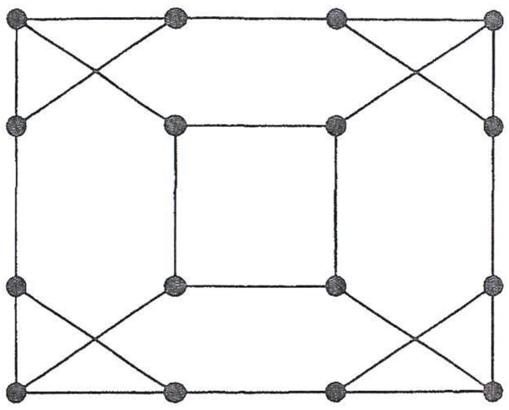

Figure 3: A (diamond,butterfly)-free graph.

matching (see Figure 5).

4 Recognizing (butterfly,diamond)-free graphs

The goal of this section is to recognize (butterfly,diamond)-free graphs. To this end, we propose an efficient algorithm (see Algorithm 1) that identifies the graphs belonging to this class. Recall that, whenever an input graph is recognized as (diamond,butterfly)-free, a CD solution for this graph can be computed using the two-phase algorithm.

We proceed by checking the absence of the forbidden graphs, that is the butterfly and the diamond, simultaneously. Recognizing (butterfly,diamond)-free graphs can be done by focusing, in turn, on the neighbourhood of every vertex of the input graph. We start from the observation that a graph is (butterfly,diamond)-free if and only if the neighbourhood of every one of its vertices induces a $(2K_2, P_3)$-free graph (see Figure 1). In turn, the recognition of $(2K_2, P_3)$-free graphs relies on the observation that a graph is $(P_3, 2K_2)$-free if and only if it has at most one clique component of size two or more, and the other connected components are just isolated vertices. Checking this latter property can be done by computing, first, the connected components of the reduced graph $(N(v), E(N(v)))$, for every vertex v of the input graph. This can be done in $O(|V| + |E|)$ steps. Next, the algorithm checks the number of connected components containing more than one vertex. If there is a single connected component with more than one vertex then it must be a clique, because otherwise, $(N(v), E(N(v)))$ will contain an induced P_3. This latter step can be executed in $O(|V| + |E|)$, which results in an overall recognition algorithm that runs in $O(|V|^2 + |V||E|)$.

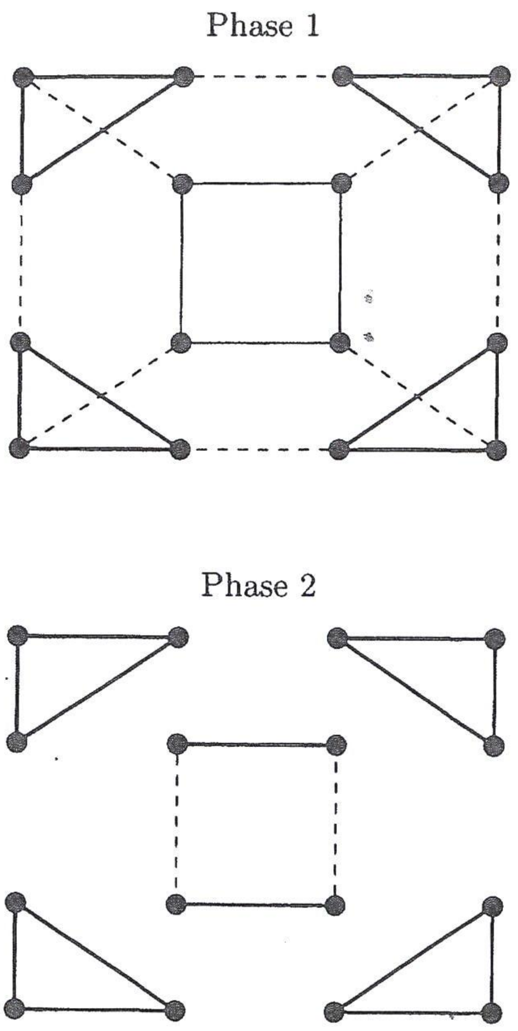

Figure 4: Application of the two-phase algorithm on the graph of Figure 3. The deleted edges are represented by dashed segments.

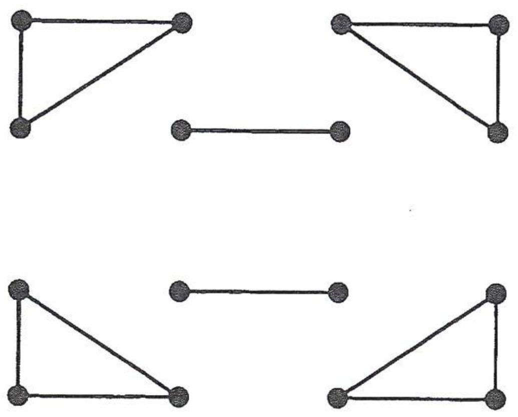

Figure 5: An optimal CD solution for the (butterfly,diamond)-free graph of Figure 3 obtained by means of the two-phase algorithm.

Algorithm 1: Recognizing (butterfly,diamond)-free graphs.

```
Input: a graph $G=(V,E)$
Output: $G$ is (butterfly,diamond)-free or not
forall the $v \in V$ do
    $\mathcal{U}$ := ConnectedComponents($N(v), E(N(v))$)
    bigClq := $\varnothing$
    forall the $U \in \mathcal{U}$ do
        if $|U| \geq 2$ then
            if bigClq $= \varnothing$ then
                bigClq := $U$
            else
                return false
            end if
        end if
    end forall
    // Checking whether the unique connected component
       with more than one vertex is a clique
    if $|U|(|U|-1)/2 \neq |E(N(v))|$ then return false
end forall
return true
```

5 Conclusion

In this paper, we identified a polynomial-time solvable cluster deletion (CD) subproblem. We presented a two-phase algorithm, which benefits from the limited overlap between adjacent maximal cliques in (butterfly,diamond)-free graphs. The first phase consists in calculating all maximal cliques having size three or more. This results in a partial solution, which is subsequently extended by the edges of a maximum matching of a sparse spanning subgraph. The proposed two-phase algorithm and its correctness proof show that CD is polynomial-time solvable in (butterfly,diamond)-free graphs. We also presented an efficient algorithm that recognizes (butterfly,diamond)-free graphs. This proves that CD is tractable on (butterfly,diamond)-free graphs.

Work is in progress in order to derive wider classes of cluster deletion problem by targeting graphs emerging in specific applications and forbidding induced subgraphs that are rarely induced in the targeted graphs.

References

[1] Mamalis, B. Gavalas, D. Konstantopoulos, C. Pantziou, G.: Chapter 12: Clustering in Wireless Sensor Networks, RFID and Sensor Networks: Architectures, Protocols, Security, and Integrations, (2009).

[2] Zahn, C. T.: Approximating symmetric relations by equivalence relations. J. Soc. Ind. Appl. Math.,(Vol. 12, pp. 840–847), (1964).

[3] Pipenbacher, P. Schliep, A. Schneckener, S. Schnhuth, A. Schomburg, D. Schrader, R.: ProClust: improved clustering of protein sequences with an extended graph-based approach. Bioinformatics, pp. 182–191, (1964).

[4] Guo, J.: A more effective linear kernelization for cluster editing. Theoretical Computer Science, Vol. 410, pp. 718–726, (2009).

[5] Shamir, R. Sharan, R. Tsur, D.: Cluster graph modification problems, Discrete Applied Mathematics 144 (12) 173–182, (2002).

[6] Bonomo, F. Durn, G. Napoli, A. Valencia-Pabon, M.: A one-to-one correspondence between potential solutions of the cluster deletion problem and the minimum sum coloring problem, and its application to P_4-sparse graphs, Inform. Process. Lett. 115 (2015) 600–603.

[7] Natanzon, A. Shamir, R. Sharan, R.: Complexity classification of some edge modification problems, Discrete Appl. Math. 113, 109–128, (2001).

[8] Bron, C. Kerbosch, J.: Finding all cliques of an undirected graph (algorithm 457). Commun. ACM, 575–576, (1973).

[9] Tomita, E. Kameda, T.: An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. J. Global Optimization, (2007).

[10] Ur Rehman, S. Ullah, K. A. Fong, S.: Graph mining: A survey of graph mining techniques. International Conference on Digital Information Management, ICDIM, 88–92, (2012).

[11] Cazals, F. Karande, C.: A note on the problem of reporting maximal cliques. Theor. Comput. Sci. 407(1-3):564 – 568, (2008).

[12] Fellows, M. R., Guo, J., Komusiewicz, C., Niedermeier, R., Uhlmann,J.: Graph-based data clustering with overlaps, Discrete Optimization, 8, 2–17, (2011).

13] M. C. Golumbic, U. Rotics. On the Clique-Width of Some Perfect Graph Classes. Int. J. Found. Comput. Sci. 11(3): 423–443 (2000)

14] V. V. Lozin. Minimal classes of graphs of unbounded clique-width. Ann. Comb. 15(4) 707-722 (2011).

15] Gao, Y., Donovan R. Hare, James N.: The cluster deletion problem for cographs, Discrete Mathematics, 313(23), 2763–2771 (2013).

[16] Micali, S., Vazirani, V, V.: An O($\sqrt{|V|} \cdot |E|$) Algorithm for Finding Maximum Matching in General Graphs, Proceedings of the 21st Annual Symposium on Foundations of Computer Science, 17–27 (1980).