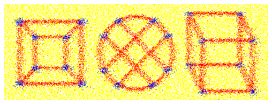
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 18, no. 4, pp. 557–576 (2014)
DOI: 10.7155/jgaa.00337

Editing Simple Graphs

Peter Damaschke Olof Mogren

Department of Computer Science and Engineering,
Chalmers University, 41296 Göteborg, Sweden

Abstract

We study the complexity of turning a given graph, by edge editing, into a target graph whose critical-clique graph is any fixed graph. The problem came up in practice, in an effort of mining huge word similarity graphs for well structured word clusters. It also adds to the rich field of graph modification problems. We show in a generic way that several variants of this problem are in SUBEPT. As a special case, we give a tight time bound for edge deletion to obtain a single clique and isolated vertices, and we round up this study with NP-completeness results for a number of target graphs.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
March 2014	July 2014	July 2014	July 2014	December 2014

Published:
December 2014

Article type: Regular paper
Communicated by: S. Pal and K. Sadakane

E-mail addresses: ptr@chalmers.se (Peter Damaschke) mogren@chalmers.se (Olof Mogren)

1 Introduction

Graphs in this paper are undirected and have no loops or multiple edges. In an edge modification problem, an input graph must be modified by edge insertions or deletions or both, to get a target graph with some prescribed property. Edge editing means both insertions and deletions. Edge insertion is also known as fill-in. The computational problem, for a given target graph property and a given type of editsm is to use a minimum number k of edits. There is a rich literature on the complexity of such problems for a number of target graph properties, and also on their various applications. Here we cannot possibly survey them all, we only refer to a few representative papers on hardness results [1, 13]. Other edit operations related to graph minors are studied in [9], and the target graph is a single fixed graph. Ironically, results are missing on edge modification problems for some structurally very simple target graphs. Informally, "simple" here means that the graph becomes small after the identification of its twin vertices (see Section 2 for technical definitions). For any fixed graph H, our target graphs will be the graphs obtained from H by replacing vertices with bags of true twins.

Our motivation of this type of problem is the concise description of graphs with very few cliques (that may overlap) and some extra or missing edges. They appear, e.g., as subgraphs in co-occurence graphs of words, and they constitute meaningful word clusters there. Within a data mining project we examined a similarity matrix of some 26,000 words, where similarity is defined by co-occurence in English Wikipedia. By thresholding we obtain similarity graphs (Figure 1 shows a part of such a graph), and we consider subgraphs that have small diameter and only few cut edges to the rest of the graph. Words occurring in the same contexts form nearly cliques. These are often not disjoint, as words appear in several contexts. Furthermore, synonyms may not always co-occur (as different authors prefer different expressions), but they co-occur with other words. Relations like this give rise to various cluster structures. As opposed to partitioning entire graphs into overlapping clusters (as in [7]), we want to single out simple subgraphs of the aforementioned type. Experience in our project shows that some existing standard clustering methods generate poor word clusters which are either too small or dragged out and not internally dense. This suggested the idea of defining the clusters directly by the desired properties, and then to determine them by edge editing of candidate subgraphs. Next, instead of describing the clusters naively as edge lists we can list their vertices along with the few edited edges (to achieve cliques). Altogether this yields very natural word clusters, and by varying the threshold we also obtain different granularities. Applications of word clusters include sentence similarity measures for text summarization, search query result diversification, and word sense disambiguation. Thus, we believe that the problems are of importance, but they are also interesting as pure graph-algorithmic problems.

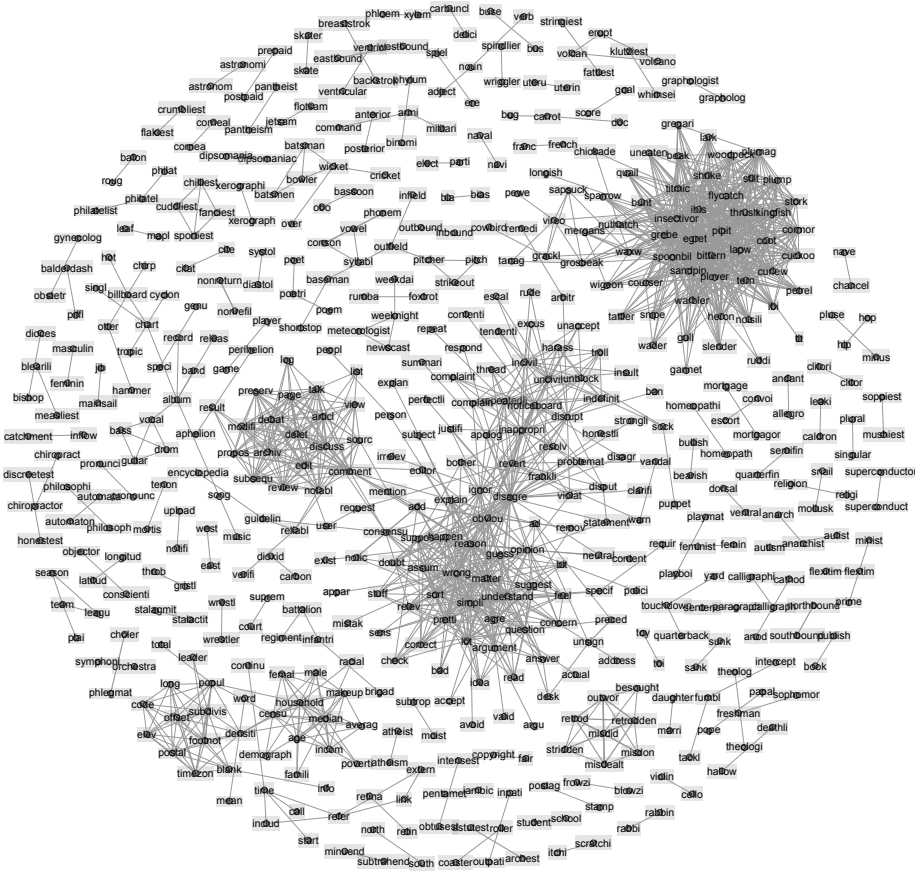

Figure 1: This Gephi visualization shows a small part of our word similarity graph, for some similarity threshold. Words have been stemmed prior to the calculations. One can clearly recognize the "almost cliques" structure, and in the middle we see an example of two overlapping cliques (the $H = P_3$ case). Also, the clusters make sense, in that they comprise related words. The data support our approach to define word clusters by edge-editing towards unions of very few cliques.

Overview of contributions:

For any fixed H, our edge modification problems are easily seen to be fixed-parameter tractable (FPT) with k as the parameter. As our main result we get in Section 3 that they even belong to the smaller class SUBEPT of problems solvable in subexponential time in the parameter. (Not very many natural SUBEPT problems are known so far, as discussed in [8].) Each of our edge modification problems has a $2^{\sqrt{k}\log k}$ time bound. The special case known as p-Cluster Editing, where H is the graph with p vertices and no edges, was recently treated in [8], using techniques like enumeration of small cuts. Our result is more general, and the quite different algorithm looks conceptually simpler, but at the price of a somewhat worse time for the special case. Therefore it remains interesting to tighten the time bounds for other specific graphs H as well.

Consequently, we then turn to the absolutely simplest graphs H: In Section 4 we study the (NP-complete) edge deletion problem towards a single clique plus isolated vertices. We give a refined FPT time bound where the target clique size c appears explicitly. Intuitively, $2k/c^2$ is an "edit density". Using an evident relationship to vertex covers we achieve, for small edit densities, essentially $O^*(1.2738^{k/c})$ time. For large enough k/c we invoke a naive algorithm instead, and the time can be bounded by $O(1.6355^{\sqrt{k \ln k}})$. The base 1.2738 is due to the best known Vertex Cover algorithm from [4]. Moreover, the bound is tight: We show that the base of k/c cannot beat the base in the best FPT algorithm for Vertex Cover.

Section 5 gives a similar FPT time bound for edge editing towards a single clique plus isolated vertices, a problem that has recently been proved to be NP-complete. In Section 6 we make some progress in proving NP-completeness results systematically, for many graphs H. The results indicate that almost all our modification problems, with rather few exceptions, might be NP-complete. But recall that, on the positive side, they are in SUBEPT.

2 Preliminaries

The number of vertices and edges of a graph $G = (V, E)$ is denoted n and m, respectively. The complement graph $\bar{G}$ of G is obtained by replacing all edges with non-edges and vice versa. We also use standard notation for some specific graphs: K_n, C_n, P_n is the complete graph (clique), the chordless cycle, the chordless path, respectively, on n vertices, and $K_{n_1,n_2,\ldots,n_p}$ is the complete multipartite graph with p partite sets of n_i vertices ($i = 1, 2, \ldots, p$). The disjoint union $G + H$ of graphs G and H consists of a copy of G and a copy of H on disjoint vertex sets. $G - v$ denotes the graph G after removal of vertex v and all incident edges. Similarly, $G - X$ denotes the graph G after removal of a vertex set $X \subseteq V$ and all incident edges. The subgraph of G induced by $X \subseteq V$ is denoted $G[X]$.

A vertex cover of $G = (V, E)$ is a subset of V being incident to all edges. The Vertex Cover problem asks for a minimum vertex cover. The vertex

covers in a graph are exactly the complements of independent sets, hence the complements of cliques in $\bar{G}$.

A graph class $\mathcal{G}$ is called hereditary if, for every graph $G \in \mathcal{G}$, all induced subgraphs of G are also members of $\mathcal{G}$. Any hereditary graph class $\mathcal{G}$ can be characterized by its forbidden induced subgraphs: F is a forbidden induced subgraph if $F \notin \mathcal{G}$, but $F - v \in \mathcal{G}$ for every vertex v.

The open neighborhood of a vertex v is the set $N(v)$ of all vertices adjacent to v, and the closed neighborhood is $N[v] := N(v) \cup \{v\}$. For a subset X of vertices, $N[X]$ is the union of all $N[v]$, $v \in X$. Vertices u and v are called true twins if uv is an edge and $N[u] = N[v]$. Vertices u and v are called false twins if uv is a non-edge and $N(u) = N(v)$. The true twin relation is an equivalence relation whose equivalence classes are known as the critical cliques of the graph. (The false twin relation is an equivalence relation as well.) In the critical-clique graph H of a graph G, every critical clique of G is represented by one vertex of H, and two vertices of H are adjacent if and only if some edge exists (and hence all possible edges exist) between the corresponding critical cliques of G. For brevity we refer to the critical cliques as *bags*, and we say that G is a "graph H of bags".

As stated earlier, an edge edit is an edge insertion or deletion. For every fixed graph H we define three edge modification problems as follows:

H-Bag Insertion / H-Bag Deletion / H-Bag Editing
Given: an input graph G and a parameter k.
Problem: Change G by at most k edge insertions / deletions / edits, such that the critical-clique graph of the resulting graph is H or an induced subgraph thereof.

We allow induced subgraphs of H in order to allow bags to be empty. Similarly we define the problems $H[0]$-Bag Deletion and $H[0]$-Bag Editing. The difference is that the target graph may additionally contain isolated vertices, that is, false twins with no edges attached. Thus, not all vertices are forced into the bags. More formally:

$H[0]$-Bag Deletion / $H[0]$-Bag Editing
Given: an input graph G and a parameter k.
Problem: Change G by at most k edge deletions / edits, such that the critical-clique graph of the resulting graph, after removal of all isolated vertices, is H or an induced subgraph thereof.

Problem $H[0]$-Bag Insertion is not mentioned above, as it easily reduces to H-Bag Insertion: As only insertions are permitted, the isolated vertices in an optimal solution are exactly the isolated vertices of G. Thus it remains to solve H-Bag Insertion on G without its isolated vertices.

We also consider problem variants where the bags have prescribed sizes. We sometimes refer to all the mentioned problems collectively as *bag modification problems*, for any fixed H. We say that editing an edge uv affects its end vertices u and v. A vertex is called unaffected if it is not affected by any edit.

Without loss of generality we can always assume that H has no true twins, because they could be merged, which leads to the same problems with a smaller graph in the role of H. For any fixed graph H understood from context, we use $\mathcal{H}$ to denote the class all graphs whose critical-clique graph is H or an induced subgraph thereof. Similarly, we use $\mathcal{H}[0]$ to denote the class of graphs consisting of all graphs from $\mathcal{H}$, but possibly with additional isolated vertices. All these classes are hereditary, and they are our classes of target graphs. (One could modify the problems by, e.g., allowing any target graphs with modular decompositions of some maximum size, but in this paper we also want a fixed structure given by H.)

We assume that the reader is familiar with fixed-parameter tractability (FPT) and basic facts, in particular with the notion of a branching vector. Otherwise we refer to [6, 14] for general introductions. A problem with input size n and an input parameter k is in FPT if some algorithm can solve it in $f(k) \cdot p(n)$ for some computable function f and some polynomial p. The $O^*(f(k))$ notation suppresses the polynomial factor $p(n)$. The subexponential parameterized tractable problems where $f(k) = 2^{o(k)}$ form the subclass SUBEPT. In our time analysis we will encounter branching vectors of a special form.

Lemma 1 *The branching vector $(1, r, \ldots, r)$ with q entries r has a branching number bounded by $1 + \frac{\log_2 r}{r}$, if r is large enough compared to the fixed q.*

Proof: Denoting the branching number by $1 + x$, we get the characteristic polynomial $(1+x)^{r+1} = (1+x)^r + q$, thus $x(1+x)^r = q$. Trying $x := \frac{\log_2 r}{r}$, the left-hand side becomes $\frac{\log_2 r}{r}(1 + \frac{\log_2 r}{r})^{\frac{r}{\log_2 r}\log_2 r}$. As r grows, $(1 + \frac{\log_2 r}{r})^{\frac{r}{\log_2 r}}$ tends to $e > 2$, thus, there is a threshold r_0 such that, for $r > r_0$, the left-hand side exceeds $\frac{\log_2 r}{r} 2^{\log_2 r} = \log_2 r > q$. Clearly, the latter inequality holds since q is fixed, and we can just make r_0 large enough. Next, as $x(1+x)^r$ is monotone in x, the true x is smaller than $x := \frac{\log_2 r}{r}$, for all $r > r_0$. It follows that $1 + \frac{\log_2 r}{r}$ is an upper bound on the branching number. □

3 Fixed-Parameter Tractability

Some of our bag modification problems (in different terminology) are known to be NP-complete, among them cases with very simple graphs H. Specifically, for $H = K_1$, problem $H[0]$-Bag Deletion can be stated as follows. Given a graph G, delete at most k edges so as to obtain a clique C and a set I of isolated vertices. Equivalently, delete a set I of vertices incident to at most k edges, and delete all these incident edges, so as to retain a clique. The problem is NP-complete due to an obvious reduction from Maximum Clique.

Next, for any fixed p, the p-Cluster Editing problem asks to turn a graph, by editing at most k edges, into a disjoint union of at most p cliques. p-Cluster Insertion and p-Cluster Deletion are similarly defined. Observe that these are the bag modification problems for $H = \bar{K}_p$. It is known that p-Cluster Insertion is polynomial for every p, and so is p-Cluster Deletion for $p = 2$,

but it remains NP-complete for every $p \geq 3$, whereas p-Cluster Editing remains NP-complete even for every $p \geq 2$ [16].

These hardness results provoke the question of fixed-parameter tractability of bag modification problems. By a well-quasi ordering argument based on Dickson's lemma [5] one can show that $\mathcal{H}$ and $\mathcal{H}[0]$ have only finitely many induced subgraphs, and then the general result from [2] implies that the bag modification problems are in FPT, for every fixed graph H. Although the argument is neat, we omit the details, because we will prove a stronger statement: membership in SUBEPT.

The following observation is known for Cluster Editing (that is, $H = \bar{K}_p$) due to [10]; here we show it for general H.

Proposition 1 *Any bag modification problem has an optimal solution where any two true twins of the input graph belong to the same bag (or both are isolated) in the target graph.*

Proof: First we consider H-Bag Editing. For a vertex v, an input graph, and a solution, we define the edit degree of v to be the number of edits that affect v. Consider any solution. For any equivalence class T of true twins, let $v \in T$ be some vertex with minimum edit degree. Consider any $u \in T \setminus \{v\}$. If the solution puts u in a different bag than v, then we undo all edits that affect u, and instead proceed as follows: We edit every edge uw, $w \neq v$, if and only if vw is edited. We also move u to the bag of v and undo the deletion of edge uv (if it happened). – Clearly, this yields a valid solution and does not increase the number of edits between u and the vertices $w \neq v$. Since we do not incur an additional edit of uv either, the new solution is no worse. Doing these changes for all $u \in T \setminus \{v\}$, and also for all T, we get a solution where any true twins end up in the same bag. This proves the assertion for H-Bag Editing.

For $H[0]$-Bag Editing we treat the set of isolated vertices as yet another bag. Then the same arguments apply. What is, however, not covered in the previous reasoning is the case when v is isolated and u is in a bag of true twins. But then u and v are not adjacent, neither before nor after the move, hence again the number of edits does not increase.

Finally, for the Insertion and Deletion problems, again the same arguments go through in all cases. Just replace "edit" with "insert" or "delete". □

We make another simple observation.

Lemma 2 *In any bag modification problem, for a fixed graph H with p vertices, the input graph has at most $2k+p$ critical cliques (isolated vertices not counted), or the instance has no solution.*

Proof: The unaffected vertices induce a subgraph that belongs to $\mathcal{H}$ or $\mathcal{H}[0]$, respectively, hence it has at most p bags. Any affected vertex is adjacent to either all or none of the vertices of any of these bags (since the latter ones are

unaffected). In the worst case, k edits affect $2k$ vertices, and each of them becomes a critical clique of its own. Together this yields the bound. □

Lemma 2 implies again that all bag modification problems for fixed H are in FPT: Assign every critical clique in the input graph to some bag of the target graph (or make its vertices isolated, in the $H[0]$ case). These are at most $p+1$ options for every critical clique. For the isolated vertices it suffices to decide how many of them we put in each bag, which are $O(n^p)$ options in total. Hence the time for this naive branching algorithm is $O^*((p+1)^{2k+p})$. Instead of this poor bound we will now show:

Theorem 1 *Any bag modification problem with a fixed graph H can be solved in $O^*(2^{\sqrt{k}\log k})$ time, hence it belongs to SUBEPT.*

Proof: First we focus on H-Bag Editing. The other problem variants can then be treated in the same way, with minor modifications.

Let a, $0 < a < 1$, be some fixed number to be specified later. To avoid bulky notation, we omit rounding brackets and work with terms like k^a as if they were integers. Let p denote the number of vertices of our fixed graph H. One difficulty is that the sizes of the p bags are not known in advance. Our preprocessing phase takes care of that.

Preprocessing phase: Initially all bags are declared *open*. For every bag we create k^a *places* that we successively treat as follows. At every place we branch: Either we *close* the bag and leave it, or we decide on a critical clique of the input graph and put any of its vertices in the bag. (Clearly, the latter choice of a vertex from a critical clique is arbitrary. By Proposition 1 we can even immediately fill further places with the entire critical clique, but our analysis will not take advantage of that.) Due to Lemma 2 these are at most $2k+p+1$ branches, hence the total number of branches is $(2k+p+1)^{pk^a} = O(k)^{pk^a} = 2^{k^a \log k}$. Note that p is fixed, and constant factors are captured by the unspecified base of log in the exponent.

Every open bag has now k^a vertices (where k is the initially given parameter value). We will not add any further vertices to closed bags. Vertices that are not yet added to bags are called *undecided*. Finally we do all necessary edits of edges between different bags, to stick to the structure of the target graph given by H, and subtract the number of these edits from k, the number of remaining edits.

Main phase: In every branch obtained in the preprocessing phase we apply further branching rules that will further reduce the parameter k by edits. The branching rules are applied exhaustively in the order described below. In the following we first consider the special case that all bags are open. Later we show how to handle the presence of closed bags, too.

All bags are open: We describe the branching rules and the situations after their exhaustive application.

- If there exists an undecided vertex u and a bag B such that u is adjacent to some but not all vertices of B, then we branch as follows: either insert all missing edges between u and B, or delete all edges between u and B. (But for now, u is not yet added to any bag.) The branching vector is some $(i, k^a - i)$ with two positive entries, or a better vector if already more than k^a vertices got into B.

- Now every undecided vertex u is either completely adjacent or completely non-adjacent to each bag B. We say that u fits in B, if u is adjacent to exactly those bags that belong to $N[B]$. Remember that H has no true twins. It follows that every vertex u fits in at most one bag.

- If there exists an undecided vertex u that fits in no bag, we branch as follows: we decide on a bag for u, put u in this bag, and do the necessary edits. Since u does not fit anywhere, we need at least k^a edits, thus the branching vector, of length p, is $(k^a, \ldots, k^a)$ or better.

- After that, every undecided vertex u fits in exactly one bag $B(u)$. Suppose that two undecided vertices u and v have the wrong adjacency relation. That is, either uv is an edge but $B(u)$ and $B(v)$ are not adjacent, or uv is not an edge but $B(u)$ and $B(v)$ are adjacent or $B(u) = B(v)$. We branch as follows: either we edit uv or not. If we don't, then u and v cannot be both added to their designated bags. Then we also decide on u or v and put that vertex in one of the other $p-1$ bags, which again costs at least k^a edits. Thus, the worst-case branching vector is $(1, k^a, \ldots, k^a)$ with $2p-2$ entries k^a.

- Finally, all undecided vertices have their correct adjacency relations, hence the graph belongs to $\mathcal{H}$.

Some bags are closed: We cannot treat closed bags like the open bags, since closed bags can be small, hence the above branching rules would not guarantee at least k^a edits. Therefore we modify the above procedure. Remember that all edits between bags were already done in the preprocessing phase, and undecided vertices can be put in open bags only.

Let U be the set of vertices of H corresponding to the open bags. Note that $H[U]$ may have true twins. In that case we merge every critical clique of $H[U]$ into one superbag. Since every open bag entered the main phase with k^a vertices, trivially, each superbag is larger than k^a.

To place the undecided vertices we perform exactly the same branching rules as above, but only on $H[U]$ (where superbags have the role of bags). Since we have fewer branches, the branching vectors do not get worse. A new twist is needed only when we actually add a vertex u to a superbag S. In every such event we also decide on the bag within S that will host u. Since every S came from a critical clique of $H[U]$, these latter choices do not change any more the adjacency relations of u with other vertices in the open bags and with other

undecided vertices. Therefore we can take these decisions independently for all u, and always choose some bag in S that causes the minimum number of edits of edges between u and the closed bags.

Complexity result: The worst branching vector we encounter (see above) is $(1, k^a, \ldots, k^a)$ with $2p-2$ entries k^a. From Lemma 1 we obtain the bound $(1 + \frac{a \log_2 k}{k^a})^k = 2^{k^{1-a} \log k}$ for some suitable logarithm base. Since the main phase is applied to every branch resulting from the prepocessing phase, we must multiply the two bounds: $2^{k^a \log k} 2^{k^{1-a} \log k}$. Choosing $a = 1/2$ yields the product $2^{\sqrt{k} \log k}$. (Note that the base is, arbitrarily, 2 because of the unspecified logarithm base.)

For H-Bag Deletion and H-Bag Insertion we proceed similarly. The only difference is that only one type of edits is permitted, hence some of the branches are disabled, which cannot make the branching vectors worse. In $H[0]$-Bag Deletion and $H[0]$-Bag Editing we can treat the set of isolated vertices like another bag; some necessary adjustments are straightforward. □

4 Clique Deletion

If H is the one-vertex graph, then the $H[0]$ edge modification problems have as target graphs a single clique plus isolated vertices. Instead of the $H[0]$-Bag terminology we speak in this case of Clique Insertion, Clique Deletion, and Clique Editing, which is more suggestive. Clique Insertion is a trivial problem: Since only edge insertions are permitted, all vertices except the isolated ones must be connected to a clique, thus there is no choice. In this section we study:

Clique Deletion
Given: an input graph G and a parameter k.
Problem: Change G by at most k edge deletions so as to obtain a clique C and a set I of isolated vertices. Equivalently: Delete a set I of vertices incident to at most k edges, and delete all these incident edges as well, so as to retain a clique.

This should not be confused with problems having a split graph as target graph, as split graphs can have additional edges between C and I.

Lemma 3 *A partitioning of the vertex set of a graph G into sets C and I is a valid solution to* Clique Deletion *if and only if I is a vertex cover of $\bar{G}$. Moreover, a minimum vertex cover I of $\bar{G}$ also yields a minimum number of edge deletions in G. Consequently,* Clique Deletion *is NP-complete.*

Proof: The first assertion is evident. For the second assertion, note that Clique Deletion requests a vertex cover I of $\bar{G}$ being incident to the minimum number of edges of G. Since C is a clique, and every edge of G is either in C or incident to I, we get the following chain of equivalent optimization

problems: minimize the number of edges incident to I, maximize the number of edges in C, maximize $|C|$, minimize $|I|$. The final assertion follows from the NP-completeness of Vertex Cover. □

Clique Deletion is also in SUBEPT by Theorem 1, but besides the generic time bound with unspecified constants in the exponent, we are now aiming at an FPT algorithm with a tight time bound, as a function of k and $c := |C|$. With m being the number of edges in the input graph, clearly, c must satisfy $m - k \leq \frac{1}{2}c(c-1)$, thus $c \geq \frac{1}{2} + \sqrt{\frac{1}{4} + 2(m-k)}$. In an algorithm for Clique Deletion we may guess the exact clique size c above this threshold and try all possible sizes c, which adds a factor smaller than n to the time bound. Therefore we may assume in the following that c is already prescribed.

Before we turn to an upper complexity bound, we first give an implicit lower bound, relative to Vertex Cover parameterized by the solution size.

Proposition 2 *Any* Clique Deletion *algorithm with a time bound $O^*(b^{k/c})$, where $b > 1$ is some constant base, yields a* Vertex Cover *algorithm with a time bound $O^*(b^v)$, where v is the vertex cover size.*

Proof: We join our input graph $G = (V, E)$ with a clique K, and define $c^* := |K|$. Joining means that all possible edges between K and V are created. Observe that an optimal solution for the joined graph consists of an optimal solution for G (a partitioning of V into some C and I), with K added to C. Thus, if k edges are deleted in G, then $k + (n-c)c^*$ edges are deleted in the joined graph, and the size of the solution clique is $c^* + c$. Furthermore, the size of the vertex cover I in $\bar{G}$ is $n - c$.

The above reasoning holds for every size c^*. If we choose c^* "large" compared to n, but still polynomial in n, then the number of deleted edges and the clique size are $(n-c)c^*$ and c^*, respectively, subject to lower-order terms. Their ratio is the vertex cover size $v := n - c$. Using the original notations k and c for the number of deleted edges and the clique size, respectively, it follows that any FPT algorithm for Clique Deletion that runs in time bounded by some function $O^*(f(k/c))$ could be used to solve also Vertex Cover in $\bar{G}$ within $O^*(f(n-c)) = O^*(f(v))$ time. □

Therefore, the best we can hope for is a Clique Deletion algorithm with a time bound $O^*(b^{k/c})$, with some constant base $b > 1$ that cannot be better than in the state-of-the-art Vertex Cover algorithm. This bound is also tight in a sense, as we will see below.

The exponent k/c is not an arbitrary measure, rather, it has a natural interpretation: It can be rewritten as $c\frac{k}{c^2}$, where the second factor can be viewed as an "edit density"; note that $\frac{2k}{c(c-1)}$ is the ratio of deleted edges and remaining edges in the target graph. For technical reasons it will be convenient to define the edit density slightly differently as $d := 2k/c'^2$ where $c' := c - 1$. Furthermore, in applications we are mainly interested in instances that are already nearly cliques, thus we keep a special focus on the case $d < 1$ in the following.

We start our algorithm for Clique Deletion with a single reduction rule: Consider any vertex v of degree smaller than c'. Clearly, v cannot be in a clique C of size c, hence it must be in I, and we can remove it without changing the problem. It also follows that it is correct to iterate this process.

Reduction rule: Remove any vertex v of degree smaller than c', along with all incident edges, and subtract the degree of v from the parameter.

After exhaustive application of this rule there remains a graph where all vertex degrees are at least c'. In order words, we compute the c'-core. From now on we can suppose without loss of generality that, already in G, all vertices have degree at least c'.

Lemma 4 *If a graph where all vertex degrees are at least c' admits a solution to* Clique Deletion *with at most k edge deletions, $|C| = c' + 1$, and $|I| = i$, then we have $i \leq 2k/c'$, and in the case $d := 2k/c'^2 < 1$ this can be improved to $i \leq \frac{2}{1+\sqrt{1-d}} \cdot k/c'$.*

Proof: Let h be the number of edges in I. Since at most k edge deletions are permitted, we have $ic' - h \leq k$. Since $h \leq k$ (or we must delete too many edges already in I), it follows $i \leq 2k/c' = dc'$.

For $d < 1$, this further implies $i \leq c'$. Using $h < i^2/2$, the previous inequality $ic' - h \leq k$ yields $ic' - i^2/2 \leq k$, thus $i^2 - 2c'i + 2k \geq 0$ with the solution $i \leq c' - \sqrt{c'^2 - 2k}$. (Recall that $i \leq c'$, thus the other solution is already excluded.) By using simple algebra this can be rewritten as

$$i \leq c' - \sqrt{c'^2 - 2k} = \frac{c'^2 - (c'^2 - 2k)}{c' + \sqrt{c'^2 - 2k}} = \frac{2k}{c' + \sqrt{c'^2 - 2k}} = \frac{2}{1 + \sqrt{1 - 2k/c'^2}} \cdot k/c'.$$

Finally remember $2k/c'^2 = dc$. □

Note that the factor in front of k/c' grows only from 1 to 2 when d grows from 0 to 1. To make this factor more comprehensible, we may also simplify it to a slightly worse upper bound: Since $\sqrt{1-d} > 1-d$, we have $i \leq \frac{2}{2-d} \cdot k/c'$. We also remark that Clique Deletion is trivial if $k < c'$, because, after reduction to the c'-core, either there remains a clique, or the instance has no solution.

Theorem 2 Clique Deletion *can be solved in $O^*(1.2738^{\frac{2}{1+\sqrt{1-d}} \cdot k/c'})$ time if $d < 1$, and in general in $O^*(1.2738^{2k/c'})$ time.*

Proof: First we apply our reduction rule, in polynomial time. Let G be the remaining graph. Due to Lemma 3 it suffices then to compute a vertex cover of minimum size in $\bar{G}$. As for the time bound, the base comes from the Vertex Cover algorithm in [4], and the exponent comes from the bounded size i in Lemma 4. For large edit densities we may still use the algorithm with the simpler bound from Lemma 4. □

In the time bound from Theorem 2 we may replace c' with c, which does not make a difference asymptotically, and write $O^*(1.2738^{2k/c})$. The following

result gives a time bound as a function of k only, as in the previous section, but with a specified base.

Corollary 1 Clique Deletion *can be solved in* $O^*(1.6355^{\sqrt{k \ln k}})$ *time.*

Proof: Depending on c, either we run the algorithm from Theorem 2 in $O^*(1.2738^{2k/c})$ time, or we check in a brute-force manner all subsets of c vertices for being cliques. The latter method runs in $O^*((2k+c)^c)$ time, since at most $2k+c$ non-isolated vertices exist due to Lemma 2. For any fixed k, compare the expressions $1.2738^{2k/c}$ and $(2k+c)^c$. They decrease and increase, respectively, as functions of c. Hence their minimum is maximized if they are equal. This happens at approximately $c = 0.492\sqrt{k/\ln k}$. Plugging in this c yields the asserted time bound. □

One may wish to improve the naive $O^*(2k+c^c)$ bound, and hence the Corollary, by fast exclusion of most c-vertex subsets as candidates for the clique C. However, the maximum clique problem cannot be solved in $O(f(c) \cdot n^{o(c)})$ time for any function f, under the Exponential Time Hypothesis [3].

5 Clique Editing

Recall that Clique Editing is the problem of editing at most k edges so as to obtain a clique C, say of size c, and a set I of $n-c$ isolated vertices. In the following theorem, c is part of the input.

Theorem 3 Clique Editing *with prescribed size* c *of the target clique is W[1]-complete in parameter* $n-c$*, hence also NP-complete.*

Proof: We argue with the (non-parameterized) optimization version and show that minimizing the number of edited edges is equivalent to finding a set I of $n-c$ vertices being incident to the minimum number of edges. This claim is verified as follows. Note that the edges incident to I are exactly those to be deleted, and minimizing deletions means maximizing the number of remaining edges. Since c is presecribed, this also minimizes the number of edge insertions needed to make C a clique. Due to [11], finding at least s vertices that cover at most t edges, known as Minimum Partial Vertex Cover, is W[1]-complete in the parameter s. Thus our assertion follows by letting $s := n-c$. □

Note that we cannot simply use Theorem 3 to conclude NP-completeness of Clique Editing when the size c is arbitrary. The catch is that the prescribed clique sizes c in the reduction graphs may be different from c in optimal solutions to Clique Editing on these graphs, and our problem might still be polynomial for the "right" c. Actually, NP-completeness has been proved in [12].

Another equivalent way to state the Clique Editing problem is: Given a graph G, find a subset C of vertices that induces a subgraph that maximizes the number of edges minus the number of non-edges. Denoting the number of edges by $m(G)$, the objective can be written as $m(G[C]) - m(\bar{G}[C])$. This

formulation also gives rise to an extension to a weighted version: For a given real number $w > 0$, maximize $m(G[C]) - w \cdot m(\bar{G}[C])$. Now the effect of w becomes interesting. The problem is trivial for $w = 0$ (the whole vertex set is an optimal C), and NP-complete if w is part of the input (since a maximum clique is an optimal C if w is large enough). But what happens in between? What is the complexity for any constant $w > 0$? We must leave this question open.

Next we propose an FPT algorithm for Clique Editing when k is the parameter. It works if c is part of the input, and hence also for free c, as we can try all, at most n, values of c. Membership in SUBEPT follows from Theorem 1, but as earlier we are also interested in the dependency of the time bound on c. The following algorithm uses similar ideas as the earlier ones.

Theorem 4 Clique Editing *can be solved in* $O^*(2^{\log c \cdot k/c})$ *time.*

Proof: We have to decide for every vertex whether to put it in C or in I.

Consider the following reduction rule: Put every vertex v of degree at most $(c-1)/2$ in I, and delete the incident edges. The correctness is seen as follows. Assume that $v \in C$ in the final solution. Since v has degree at most $(c-1)/2$, at least $(c-1)/2$ edges between v and the rest of C have been inserted. If we had instead decided $v \in I$, we would have inserted no edges incident to v, but deleted the at most $(c-1)/2$ incident edges, which is not more expensive.

After exhaustive application of the reduction rule, there remains a graph of minimum degree at least $c/2$. We can assume without loss of generality that already the input graph has minimum degree at least $c/2$. We begin with branching. A vertex is called undecided if it is not yet put in C or I. Initially we guess one vertex of C, which adds only a linear factor to the time bound. All other vertices are undecided in the beginning.

As long as there exists an undecided vertex v which is not adjacent to all of C, we branch on v. In the $I := I \cup \{v\}$ branch we delete the, at least $c/2$, incident edges. (Whenever some vertex degrees fall below $c/2$ because of the deletions, we first apply the reduction rule again.) In the $C := C \cup \{v\}$ branch we insert at least one edge that is missing in C. After exhaustive application, all undecided vertices are adjacent to all vertices in C. If the undecided vertices form a clique, we are done, as we can add the undecided vertices to C, and if we get $|C| > c$, some surplus vertices are moved to I without branching. Suppose that the other case holds: two non-adjacent undecided vertices u and v exist. Then we branch by setting $C := C \cup \{u, v\}$ or $I := I \cup \{u\}$ or $I := I \cup \{v\}$. In the first branch we must insert an edge, and otherwise delete at least $c/2$ edges.

Our rules have, obviously, the worst-case branching vectors $(1, c/2)$ and $(1, c/2, c/2)$, and Lemma 1 yields the time bound. □

6 Some Hardness Results

All bag modification problems are trivially in NP. In this section we prove the NP-completeness of bag modification problems for many target graphs H. We

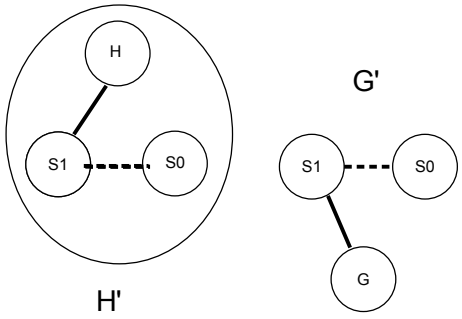

Figure 2: Thick lines mean that all possible edges between the induced subgraphs exist, dotted lines mean that some of the edges may exist. Inflation of the vertices to bags is not shown here.

give a general construction that "lifts" NP-completeness from a considered H to larger graphs H'. That is, we will in polynomial time reduce H-Bag Editing to H'-Bag Editing, for certain graphs H and H' specified later on.

The general situation is as follows. Let the graph G and parameter k be any instance of H-Bag Editing, and let H' be a graph that contains H as an induced subgraph.

We choose a particular subset S of vertices of H' such that $H'[S]$ is isomorphic to H. Note that H may have several occurrences as induced subgraph in H', but we fix some set S. Let S_0 be some set of vertices of $H' - S$ which are adjacent to no vertices of S. (But S_0 is not necessarily the set of all such vertices.) Similarly, let S_1 be some set of vertices of $H' - S$ which are adjacent to all vertices of S. From G we construct a graph G' as follows, in polynomial time. We take $S_0 \cup S_1$ from H' and replace every vertex of $S_0 \cup S_1$ with a bag of size c, for some number $c > 2k$. Two bags are joined by all possible edges (by no edges) if the corresponding vertices in H' are adjacent (not adjacent). Then we add G and insert all possible edges between S_1 and the vertices of G. Figure 2 is a simple sketch of the construction that shall help remember the role of S_0 and S_1.

If G with parameter k is a yes-instance of H-Bag Editing, then we can take the at most k edits that yield a solution, and mimic them in the subgraph G of G'. This immediately implies that G' with parameter k is a yes-instance of H'-Bag Editing: we can map the vertices of G onto S to obtain an edited graph whose critical-clique graph is an induced subgraph of H'. (Assumption $c > 2k$ is not needed for this direction.)

However the converse does not hold in general: "If G' with parameter k is a yes-instance of H'-Bag Editing, then G' with parameter k is a yes-instance of H-Bag Editing." Our aim in the following is to show this converse under certain conditions on H and H'. The equivalence will then establish the desired

reduction.

Specifically, suppose that the following technically looking condition is satisfied. Here, an embedding of a graph into another graph means that edges are mapped to edges, and non-edges are mapped to non-edges, i.e., the embedded graph is an induced subgraph of the host graph. Remember that H is isomorphic to $H'[S]$.

Embedding condition: Let J be any induced subgraph of H' isomorphic to $H'[S_0 \cup S_1]$. Accordingly, we embed J into any graph of $\mathcal{H}'$ and divide the vertex set of J in two sets U_0 and U_1. These are the sets of those vertices which come from S_0 and S_1, respectively. For every such embedding, let T be the set of vertices t such that $N[t]$ contains all vertices of U_1 and no vertex of U_0. Then the subgraph induced by T is always in $\mathcal{H}$.

Note that there may exist many possible embeddings of J into a host graph from $\mathcal{H}'$, and our condition must hold for each of them. We also remark that T may contain some vertices of U_1.

Now suppose that G' with parameter k is a yes-instance of H'-Bag Editing, that is, at most k edge edits in G' have produced a graph in $\mathcal{H}'$. Since k edits affect at most $2k$ vertices, but $c > 2k$, we conclude that every bag in the edited graph corresponding to a vertex of $S_0 \cup S_1$ still has at least one unaffected vertex. Accordingly, let U be some set of unaffected vertices, containing one such vertex from each of the bags. The subgraph induced by U in the edited graph is then isomorphic to $H'[S_0 \cup S_1]$. Let U_0 and U_1 be the subset of those vertices of U corresponding to vertices of S_0 and S_1, respectively. Then we have $U = U_0 \cup U_1$. Furthermore, since U is unaffected, all vertices of G are still adjacent (non-adjacent) to all vertices of U_1 (U_0).

Recall that H and H' are assumed to satisfy our embedding condition. Take as T the vertex set of G. It follows that, after editing, the vertices of G form a graph in $\mathcal{H}$. Since at most k edits have been done in the whole graph, we get that G with parameter k is a yes-instance of H-Bag Editing. To summarize:

Lemma 5 *If the embedding condition holds for graphs H and H' and some vertex sets S, S_0, S_1 (as specified above), then H-Bag Editing is reducible to H'-Bag Editing in polynomial time.*

The embedding condition looks more complicated than it is. When it comes to specific graphs H and H', it is often easy to apply, as we will see in the examples below. We only have to find suitable sets S_0 and S_1. For convenience we refer to the vertices in S_0 and S_1 as 0-vertices and 1-vertices, respectively, and we call any graph in $\mathcal{H}$ a *graph H of bags*.

Theorem 5 *H'-Bag Editing is NP-complete for, at least, the following graphs H':*
complete multipartite graphs where some partite set has at least 3 vertices;
the complete multipartite graph with partite sets of exactly 2 vertices;
K_3-free graphs with maximum degree at least 3.

Proof: H-Bag Editing for $H = \bar{K}_p$ is p-Cluster Editing, which is known to be NP-complete for every $p \geq 2$ [16]. By virtue of Lemma 5 we reduce H-Bag Editing for $H = \bar{K}_p$, with a suitable $p \geq 2$, to H'-Bag Editing for the mentioned graphs H'.

In a complete multipartite graph H', let b denote the size of some largest partite set, and assume $b \geq 3$. We choose $p = b - 1 \geq 2$. We let S_1 be empty, and we let S_0 consist of a single vertex in a partite set of size b. The vertices of H' being non-adjacent to this 0-vertex are in the same partite set, hence they induce a graph $\bar{K}_{b-1} = \bar{K}_p$. No matter where we embed our 0-vertex in a graph H' of bags, the set T as defined in the embedding condition forms a graph $H = \bar{K}_{b-1}$ of bags. (For partite sets smaller than b, note that bags are allowed to be empty.) Hence the embedding condition is satisfied.

Next consider $H' = K_{2,2} = C_4$. We choose $p = 2$, and we let S_1 consist of two non-adjacent vertices, while S_0 is empty. Clearly, the common neighbors of the two 1-vertices induce the graph $\bar{K}_2 = H$. The only possible embedding of our two non-adjacent 1-vertices in a graph $H' = C_4$ of bags is to put them in two non-adjacent bags. Then the set T forms again a graph $\bar{K}_2 = H$ of bags, thus the embedding condition is satisfied.

To prove NP-completeness of H'-Bag Editing for $H' = K_{2,\ldots,2}$, let $H = K_{2,\ldots,2}$ but with two vertices less. Then literally the same arguments as in the previous paragraph show that the embedding condition is satisfied. Using this observation as induction step and the case $H' = K_{2,2}$ as the induction base, this proves the claim by induction on the number of partite sets.

Finally, consider any K_3-free graph H' of maximum degree $d \geq 3$. (That is, H' is K_3-free but not merely a disjoint union of paths and cycles.) Fix some vertex v of degree d, and some neighbor u of v. We choose $p = d - 1$, $S_1 = \{v\}$, and $S_0 = \{u\}$. The vertices adjacent to v and non-adjacent to u obviosuly induce a graph $\bar{K}_{d-1} = \bar{K}_p$. For any embedding of an adjacent pair of a 1-vertex and a 0-vertex into a graph H' of bags, the set T forms a graph $H = \bar{K}_p$ of bags. This holds because every vertex in H' has at most d neighbors, they are pairwise non-adjacent, and one of the d bags neighbored to the 1-vertex is already occupied by the 0-vertex. Once more, the embedding condition is satisfied. □

The same construction also lifts NP-completeness results from $H[0]$ to $H'[0]$, whenever we can choose $S_1 = \emptyset$ and a suitable S_0. Our construction also works for H'-Bag Deletion and H'-Bag Insertion; the only modification is that only one type of edge edits is permitted. However, note that we need an NP-complete case to start with. For H'-Bag Deletion we can use $\bar{K}_p$ with $p \geq 3$. As for H'-Bag Insertion, remember that $\bar{K}_p$-Bag Insertion is polynomial [16] for every p. Still we can start from P_3 instead and get, for instance, the following results:

Theorem 6 *H'-Bag Insertion is NP-complete for, at least, the graphs $H' = P_3$, and $H' = P_n$ and $H' = C_n$ for each $n \geq 6$.*

Proof: P_3-Bag Insertion in G means to delete in $\bar{G}$ a minimum number of

edges so as to obtain a graph consisting of a complete bipartite graph (biclique) and isolated vertices. This in turn is equivalent to the problem of finding a biclique with a maximum number of edges. The latter problem is NP-complete (even in bipartite graphs and hence in general graphs) due to [15]. Finally we reduce P_3-Bag Insertion to P_n-Bag Insertion for each $n \geq 6$ by setting $S_1 = \emptyset$ and S_0 isomorphic to P_{n-4}. Similarly, we reduce P_3-Bag Insertion to C_n-Bag Insertion for each $n \geq 6$ by setting $S_1 = \emptyset$ and S_0 isomorphic to P_{n-5}. It is straightforward to verify the embedding condition. □

These results are applications of only one reduction technique. We may get out more NP-complete cases, but the construction also fails for some other graphs. We also remark that P_3-Bag Deletion is polynomial: consider the complement graph and proceed similarly as in [16]. It would be interesting to achieve a complexity dichotomy for all target graphs.

Acknowledgements

This work has been partially supported by the following project grants: "Generalized and fast search strategies for parameterized problems", grant 2010-4661 from the Swedish Research Council (Vetenskapsrådet), and "Data-driven secure business intelligence", grant IIS11-0089 from the Swedish Foundation for Strategic Research (SSF). The authors would also like to thank Gabriele Capannini for sharing the data and for initial discussions of the applications, and Henning Fernau for intensive discussions of some open problems during a short visit at Chalmers.

References

[1] P. Burzyn, F. Bonomo, and G. Durán. Np-completeness results for edge modification problems. *Discrete Applied Mathematics*, 154(13):1824–1844, 2006. doi:10.1016/j.dam.2006.03.031.

[2] L. Cai. 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.

[3] J. Chen, X. Huang, I. A. Kanj, and G. Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006. doi:10.1016/j.jcss.2006.04.007.

[4] J. Chen, I. A. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010. doi:10.1016/j.tcs.2010.06.026.

[5] L. E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *Amer. J. Math.*, 35(4):413–422, 1913. doi:10.2307/2370405.

[6] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.

[7] M. R. Fellows, J. Guo, C. Komusiewicz, R. Niedermeier, and J. Uhlmann. Graph-based data clustering with overlaps. *Discrete Optimization*, 8(1):2–17, 2011. doi:10.1016/j.disopt.2010.09.006.

[8] F. V. Fomin, S. Kratsch, M. Pilipczuk, M. Pilipczuk, and Y. Villanger. Tight bounds for parameterized complexity of cluster editing with a small number of clusters. *J. Comput. Syst. Sci.*, 80(7):1430–1447, 2014. doi:10.1016/j.jcss.2014.04.015.

[9] P. A. Golovach, D. Paulusma, and I. A. Stewart. Graph editing to a fixed target. In *Combinatorial Algorithms - 24th International Workshop, IWOCA 2013, Rouen, France, July 10-12, 2013, Revised Selected Papers*, pages 192–205, 2013. doi:10.1007/978-3-642-45278-9_17.

[10] J. Guo. A more effective linear kernelization for cluster editing. *Theor. Comput. Sci.*, 410(8-10):718–726, 2009. doi:10.1016/j.tcs.2008.10.021.

[11] J. Guo, R. Niedermeier, and S. Wernicke. Parameterized complexity of vertex cover variants. *Theory Comput. Syst.*, 41(3):501–520, 2007. doi:10.1007/s00224-007-1309-3.

[12] I. Kovác, I. Selecéniová, and M. Steinová. On the clique editing problem. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 469–480, 2014. doi:10.1007/978-3-662-44465-8_40.

[13] A. Natanzon, R. Shamir, and R. Sharan. Complexity classification of some edge modification problems. *Discrete Applied Mathematics*, 113(1):109–128, 2001. doi:10.1016/S0166-218X(00)00391-7.

[14] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Univ. Press, 2006.

[15] R. Peeters. The maximum edge biclique problem is np-complete. *Discrete Applied Mathematics*, 131(3):651–654, 2003. doi:10.1016/S0166-218X(03)00333-0.

[16] R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1-2):173–182, 2004. doi:10.1016/j.dam.2004.01.007.