Editing the Simplest Graphs

Peter Damaschke and Olof Mogren

Department of Computer Science and Engineering
Chalmers University, 41296 Göteborg, Sweden
[ptr,mogren]@chalmers.se

Abstract. We study the complexity of editing a graph into a target graph with any fixed critical-clique graph. The problem came up in practice, in mining a huge word similarity graph 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.

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, into 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 is to use a minimum number k of edits. There is a rich literature on the complexity for a number of target graph properties, and on their various applications. Here we cannot survey them, we only refer to a few representative papers on hardness results [1, 10]. Ironically, results are missing on edge modification problems for some structurally very simple target graphs. Informally, "simple" means that the graph becomes small after identification of its twin vertices (see Section 2). For any fixed graph H, our target graphs are all graphs obtained from H by replacing vertices with bags of true twins.

Our motivation 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 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, 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. 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 [6]), 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 to define 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 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.

For any fixed H, our edge modification problems are (easily) in FPT. As our main result we get in Section 3 that they even belong to SUBEPT. (Not very many natural SUBEPT problems are known so far, as discussed in [7].) Every such problem has a $2^{\sqrt{k}\log k}$ time bound. The special case of p-Cluster Editing, where H is the graph with p vertices and no edges, was recently treated in [7], using techniques like enumeration of small cuts. Our result is more general, and the quite different algorithm looks conceptually simpler, 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 also be bounded by $O(1.6355^{\sqrt{k \ln k}})$. The base 1.2738 is due to the best known Vertex Cover algorithm from [3]. 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. Here, NP-completeness is an intriguing open question. However, in Section 6 we make some progress in proving NP-completeness 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. For a graph G, the complement graph $\bar{G}$ 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. 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. Notation $G - X$ is similarly defined for any subset $X \subseteq V$. The subgraph of G induced by $X \subseteq V$ is denoted $G[X]$.

By definition, a graph class $\mathcal{G}$ is 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 (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". (Similarly one could also consider target graphs with small modular decompositions.)

For every graph H we define three edge modification problems called H-Bag Insertion, H-Bag Deletion, H-Bag Editing, as follows: Given an input graph G and a parameter k, change G by at most k edge insertions, deletions, or edits, respectively, such that the resulting graph has H or an induced subgraph of H as its critical-clique graph. 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. Thus, not all vertices are forced into the bags. Problem $H[0]$-Bag Insertion 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.) We also consider problem variants where the bags have prescribed sizes. We sometimes refer to all the mentioned problems collectively as bag modification problems. 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 a fixed graph H understood from context, let $\mathcal{H}$ be the class all graphs whose critical-clique graph is H or an induced subgraph thereof. Let $\mathcal{H}[0]$ be the class of graphs consisting of all graphs of $\mathcal{H}$ with, possibly, additional isolated vertices. All these classes are hereditary.

We assume that the reader is familiar with the notion of fixed-parameter tractability (FPT) and basic facts, otherwise we refer to [5, 12]. 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. We use the $O^*(f(k))$ notation that suppresses $p(n)$. 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. The proof of the branching number, by standard algebra, is omitted due to lack of space.

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.*

3 Fixed-Parameter Tractability

Some 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. In other words, these are the bag modification problems where $H = \bar{K}_p$. It is known that p-Cluster Insertion is polynomial for every p, and so is p-Cluster Deletion for $p = 2$, but it is NP-complete for every $p \geq 3$, whereas p-Cluster Editing is NP-complete for every $p \geq 2$ [13].

The hardness results provoke the question on fixed-parameter tractability. By a well-quasi ordering argument based on Dickson's lemma [4] 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. 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 ($H = \bar{K}_p$) due to [8]; 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. For any class T of true twins, let $v \in T$ be some vertex with minimum edit degree. Consider any $u \in T \setminus \{v\}$. If u is put in a different bag than v, we undo all edits that affect u, and instead edit each 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 vertices $w \neq v$. Since we do not incur an additional edit of uv either, the new solution is no worse. We proceed in this way for all $u \in T \setminus \{v\}$, and also for all T. This proves the assertion for H-Bag Editing.

For $H[0]$-Bag Editing we treat the set of isolated vertices as yet another bag. The same arguments apply. What is not covered in the previous reasoning is the case when v is isolated and u is in a bag. But then u and v are not adjacent, neither before nor after the move, hence the number of edits does not increase. For the Insertion and Deletion problems, again the same arguments go through in all cases. Just replace "edit" with "insert" or "delete". The only change is that, in Insertion, the edge uv cannot have been deleted. □

We make another simple observation. In the following let p always denote the number of vertices of our fixed H.

Lemma 2. *In any bag modification problem, 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 isolated vertices it suffices to decide how many of them we put in each bag, which are $O(n^p)$ options. Hence the time for this naive branching is $O^*((p+1)^{2k+p})$. Instead of this poor bound we now show:

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

Proof. First we focus on H-Bag Editing. 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.

One difficulty is that the sizes of the p bags are not known in advance. A *preprocessing phase* takes care of that. Initially all bags are *open*. In 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 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 base of log. Every open bag has now k^a vertices. We will not add any further vertices to closed bags. Vertices that are not yet added to bags are called *undecided*. We also do all necessary edits of edges between different bags, to stick to the given graph H, and reduce k accordingly.

In the *main phase* we do branchings that further reduce the parameter k by edits. The branching rules are applied exhaustively in the given order. In the

following we first consider the special case that all bags are open. Later we show how to handle closed bags, too.

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 and 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 and 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, 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 $\mathcal{H}$.

The difficulty with closed bags is that they do not guarantee at least k^a edits. 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. Trivially, each superbag is larger than k^a. On $H[U]$ and the superbags we perform exactly the same branching rules as above. 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. This latter choice does not change the adjacency relations within the union of open bags and undecided vertices any more. 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.

The worst branching vector we encounter 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. We must multiply this with the bound $2^{k^a \log k}$ from the first phase. Choosing $a = 1/2$ yields the product $2^{\sqrt{k} \log k}$.

For H-Bag Deletion and H-Bag Insertion we proceed similarly. Since only one type of edits is permitted, 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 a bag; some necessary adjustments are straightforward. □

4 Clique Deletion

If H is the one-vertex graph, then the $H[0]$ edge modification problems aim at a single clique plus isolated vertices. Instead of "$H[0]$-Bag ..." we speak in this case of Clique Insertion, Clique Deletion, and Clique Editing, which is more suggestive. Clique Insertion is a trivial problem. In this section we study Clique Deletion: given a graph G, delete at most k edges so as to obtain a clique C and a set I of isolated vertices. An equivalent formulation is to 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 vertex-deletion interpretation is sometimes more convenient. The problem is NP-complete due to an obvious reduction from Clique or Vertex Cover, and in SUBEPT by Theorem 1.

Besides the generic time bound with unspecified constants, we are now aiming at an FPT algorithm with a tight time bound, as a function of k and $c = |C|$. We remark that the smallest possible c can be calculated from the number m of edges in the input graph. Clearly, we must have $m - k \leq \frac{1}{2}c(c-1)$, thus $c \geq \frac{1}{2} + \sqrt{\frac{1}{4} + 2(m-k)}$. We may even guess the exact clique size c above this threshold and try all possible sizes, which adds at most a factor $n - c$ to the time bound.

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.*

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 $|I|$. □

Before we turn to an upper complexity bound, we first give an implicit lower bound. Let us join our input graph G with a clique K, and define $c^* := |K|$. Joining means that all possible edges between K and G are created. Observe that an optimal solution for the joined graph consists of an optimal solution for G, 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, the size of the solution clique is c^*+c. Furthermore, the size of the vertex cover in $\bar{G}$ is $n-c$. If we choose c^* "large" compared to n, but still polynomial in n, then the number of deleted edges and the clique size are essentially $(n-c)c^*$ and c^*, respectively. Their ratio is the vertex cover size $n-c$. Back to the original notations k and c for these numbers, it follows that any FPT algorithm for Clique Deletion, that runs in time bounded by some function $f(k/c)$, could be used to solve also Vertex Cover on $\bar{G}$ within time $f(n-c)$. 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 can be rewritten as $c(k/c^2)$, where the second factor has a natural interpretation: Since the number of edges in C is roughly $c^2/2$, we can view $2k/c^2$ as an "edit density", the ratio of deleted edges and remaining edges in the target graph. It will be convenient to define $c' := c - 1$, and to define the edit density slightly differently as $d := 2k/c'^2$. 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.

Our algorithm for Clique Deletion preprocesses the input graph with a single reduction rule: Remove each vertex v of degree smaller than c', along with all incident edges. After exhaustive application, there remains a graph with minimum degree c'. From now on we can suppose without loss of generality that G has already minimum degree c'. This also bounds $i := |I|$ as follows.

Lemma 4. *With the above denotations we have $i \leq 2k/c'$, and in the case $d < 1$ 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 $i \leq c' - \sqrt{c'^2 - 2k}$. (We had excluded the case $i > c'$.) By simple algebra this can be rewritten as $i \leq \frac{2}{1+\sqrt{1-d}} \cdot k/c'$. □

Note that the factor in front of k/c' grows 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 the reduction phase, 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.*

Proof. After applying our reduction rule, due to Lemma 3 it suffices to compute a vertex cover of minimum size in $\bar{G}$. As for the time bound, the base comes from [3] and the exponent comes from Lemma 4. □

For large edit densities we may also express the time bound as a function of k only, as in the previous section, but with a specific base. The algorithm of Theorem 2 with the simpler bound from Lemma 4 has the running time $O^*(1.2738^{2k/c})$. (We replace c' with c, which does not make a difference asymptotically.) If c is small, we can instead use a brute-force approach and check all subsets of c vertices for being cliques. This runs in $O^*(2k^c)$ time, since at most $2k + c$ non-isolated vertices exist, and k is large compared to c in the considered case. The two expressions decrease and increase, resepctively, as c grows. Hence their minimum is maximized if, approximately, $c = 0.492\sqrt{k/\ln k}$. Plugging in this c yields $1.6355^{\sqrt{k \ln k}}$ time. The naive $O^*(2k^c)$ bound can certainly be improved by excluding most c-vertex subsets as candidates for the final clique.

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.

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 optimization versions 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: Simply 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 fixed, this also minimizes the number of edge insertions needed to make C a clique. Due to [9], finding at least s vertices that cover at most t edges, known as Minimum Partial Vertex Cover, is W[1]-complete in parameter s, thus our assertion follows with $s := n - c$. □

Note that Theorem 3 does not immediately imply NP-completeness of Clique Editing with free size c, since 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, albeit this is hard to imagine. We conjecture that Clique Editing is NP-complete. Another equivalent formulation of Clique Editing 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 becomes also interesting in a weighted version. For a given real number $w > 0$, maximize $m(G[C]) - w \cdot m(\bar{G}[C])$. This 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). What happens in between? For any constant $w > 0$? In particular, for $w = 1$? We must leave this question open.

Next we propose an FPT algorithm Clique Editing when k is the parameter. It works if c is part of the input (cf. Theorem 3), and hence also for free c, by trying all values. 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 that uses similar ideas as the earlier ones is omitted due to lack of space.

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

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 give a general construction that "lifts" NP-completeness from some H to larger graphs H'. To be specific, suppose that H-Bag Editing is already known to

be NP-complete. We will reduce it in polynomial time to H'-Bag Editing, for certain graphs H' specified later on.

Let the graph G and parameter k be an instance of H-Bag Editing. 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 the same graph may have several occurrences as induced subgraph, hence we must fix some S.) Let S_0 and S_1 be some set of vertices of $H' - S$ being adjacent to no vertices of S, and to all vertices of S, respectively. We construct a graph G' as follows, in polynomial time. We replace every vertex of $S_0 \cup S_1$ with a bag of size $c > 2k$. Two bags are joined by all possible edges (by no edges) if the corresponding vertices of H' are adjacent (not adjacent). Then we add G and insert all possible edges between S_1 and the vertices of G.

If G with parameter k is a yes-instance of H-Bag Editing, then we can mimic the same, at most k, edits also in the subgraph G of G', which implies that G' with parameter k is a yes-instance of H'-Bag Editing. Our aim in the following is to show the converse, under some conditions on H and H'. The equivalence will then establish the desired reduction. Specifically, suppose that the following technically looking condition is fulfilled. Here, an embedding of a graph into another graph means that edges are mapped to edges, and non-edges are mapped to non-edges.

(*) 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, of those vertices coming from S_0 and S_1, respectively. For any 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, and our condition must hold for each of them. Also, T may contain some vertices of U_1.

Now suppose that at most k edits of edges in G' have produced a graph in $\mathcal{H}'$. Since k edits affect at most $2k$ vertices, but $c > 2k$, clearly every bag in the edited graph corresponding to a vertex of S_0 or S_1 still has at least one unaffected vertex. We select one from each bag and obtain a set U of unaffected vertices. The subgraph induced by U is isomorphic to $H'[S_0 \cup S_1]$. Let U_0 and U_1 be the subset of vertices of U corresponding to vertices of S_0 and S_1, respectively. Then we have $U = U_0 \cup U_1$, and all vertices of G are still adjacent (non-adjacent) to all vertices of U_1 (U_0). Thus (*) implies 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.

Condition (*) looks more complicated than it is, when it comes to specific graphs H. In the following we give some examples. We refer to 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 with some partite set of 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 $p \geq 2$ [13]. We reduce the case $H = \bar{K}_p$ for a suitable $p \geq 2$ to the case H'.

In a complete multipartite graph H', let $b \geq 3$ be the size of some largest partite set. We choose $p = b - 1 \geq 2$. We let S_1 be empty, and 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 induce a $\bar{K}_{b-1} = \bar{K}_p$. No matter where else we embed our 0-vertex in a graph H' of bags, the set T as defined in (*) forms a $\bar{K}_{b-1}$ of bags (note that bags are allowed to be empty), hence (*) is satisfied.

Consider $H' = K_{2,2} = C_4$. We choose $p = 2$. We let S_1 consist of two non-adjacent vertices, while S_0 is empty. Clearly, their common neighbors induce $H = \bar{K}_2$. The only possible embedding of our two non-adjacent 1-vertices in a C_4 of bags is to put them in two non-adjacent bags, such that the set T forms a graph $H = \bar{K}_2$ of bags, hence (*) is satisfied. For $H' = K_{2,\ldots,2}$ we proceed by induction on the number of partite sets. Let $H = K_{2,\ldots,2}$ with two vertices less. Then the same choice of S_1 and S_0 and the same arguments establish the induction step.

Let H' be K_3-free, v a vertex of maximum degree $d \geq 3$, and u some neighbor of v. We choose $p = d - 1$, $S_1 = \{v\}$ and $S_0 = \{u\}$. The vertices adjacent to v and non-adjacent to u induce $\bar{K}_{d-1} = \bar{K}_p$. For any embedding of an adjacent pair of a 1-vertex and a 0-vertex into an H' of bags, the set T forms a graph $H = \bar{K}_p$ of bags, since in H' every vertex has at most $d - 1$ neighbors, and they are pairwise non-adjacent. □

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, however, note that we need an NP-complete case to start with. For edge deletions we can use $\bar{K}_p$ with $p \geq 3$. Remember that $\bar{K}_p$-Bag Insertion is polynomial [13] for every p. However, we can start from P_3 instead:

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

Proof. P_3-Bag Insertion in G means to delete in $\bar{G}$ a minimum number of edges so as to reach a complete bipartite graph (biclique) and isolated vertices. This is equivalent to finding a biclique with maximum number of edges. The latter problem is NP-complete (even in bipartite graphs and hence in general graphs) due to [11]. 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 easy to verify condition (*) in the equivalence proofs of the reductions. □

These Theorems are only illustrations of a few cases. The conditions on H' can be weakened, and even more cases H' proved to be NP-complete, however we want to avoid a tedious list of applications of one particular technique. On the negative side, the current construction fails for other graphs. The "smallest" open cases are $K_1[0]$-Bag Editing and P_3-Bag Editing. We also remark that P_3-Bag Deletion is polynomial: consider the complement graph and proceed similarly as in [13].

Acknowledgments. This work has been done within the projects "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 discussing the applications, and Henning Fernau for intensive discussions of some open problems during a short visit at Chalmers.

References

1. Burzyn, P., Bonomo, F., Durán, G.: NP-completeness Results for Edge Modification Problems. Discr. Appl. Math. 154, 1824–1844 (2006)
2. Cai, L.: Fixed-Parameter Tractability of Graph Modification Problems for Hereditary Properties. Inf. Proc. Lett. 58, 171–176 (1996)
3. Chen, J., Kanj, I.A., Xia, G.: Improved Upper Bounds for Vertex Cover. Theor. Comp. Sci. 411 3736–3756 (2010)
4. Dickson, L.E.: Finiteness of the Odd Perfect and Primitive Abundant Numbers with n Distinct Prime Factors. Am. J. Math. 35, 413–422 (1913)
5. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, New York (1999)
6. Fellows, M.R., Guo, J., Komusiewicz, C., Niedermeier, R., Uhlmann, J.: Graph-Based Data Clustering with Overlaps. Discr. Optim. 8, 2–17 (2011)
7. Fomin, F.V., Kratsch, S., Pilipczuk, M., Pilipczuk, M., Villanger, Y.: Tight Bounds for Parameterized Complexity of Cluster Editing. In: Portier, N., Wilke, T. (eds.) STACS 2013. LIPIcs, vol. 20, pp. 32–43, Dagstuhl (2013)
8. Guo, J.: A More Effective Linear Kernelization for Cluster Editing. In: Chen, B., Paterson, M., Zhang, G. (eds.) ESCAPE 2007. LNCS, vol. 4614, pp. 36–47, Springer, Heidelberg (2007)
9. Guo, J., Niedermeier, R., Wernicke, S.: Parameterized Complexity of Vertex Cover Variants. Theory Comput. Syst. 41, 501–520 (2007)
10. Natanzon, A., Shamir, R., Sharan, R.: Complexity Classification of some Edge Modification Problems. Discr. Appl. Math. 113, 109–128 (2001)
11. Peeters, R.: The Maximum Edge Biclique Problem is NP-complete. Discr. Appl. Math. 131, 651–654 (2003)
12. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford Lecture Series in Math. and its Appl., Oxford Univ. Press (2006)
13. Shamir, R., Sharan, R., Tsur, D.: Cluster Graph Modification Problems. Discr. Appl. Math. 144, 173–182 (2004)

Appendix

Proof of Lemma 1

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.

Proof of Theorem 4

We decide for every vertex whether to put it into C or I.

Observe the following reduction rule: Put every vertex v of degree at most $(c-1)/2$ into I, and delete the incident edges. The correctness is seen as follows. Assume $v \in C$ 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 in total.

After exhaustive application of the reduction rule, there remains a graph of minimum degree $c/2$. We can assume without loss of generality that already the input graph has minimum degree $c/2$. 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. All other vertices are undecided.

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. Hence suppose that 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.

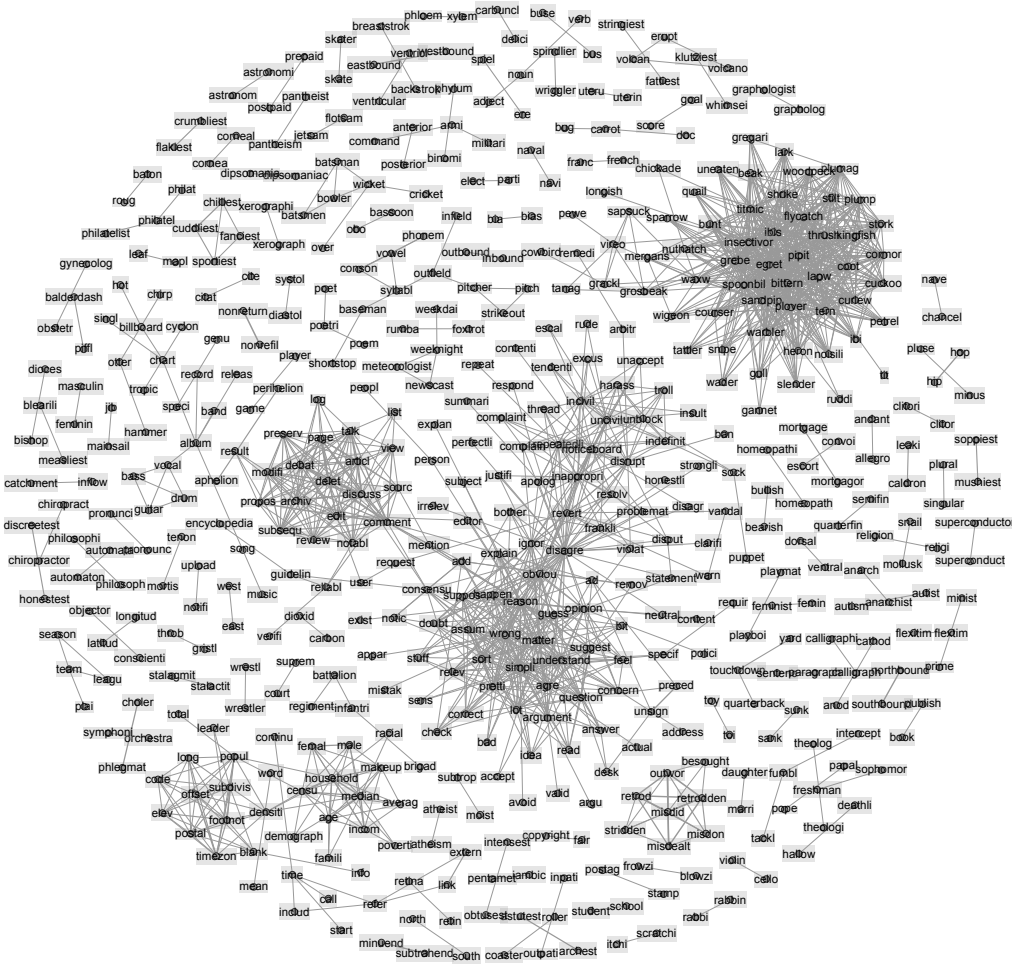

Fig. 1. This Gephi visualization shows a small part of our word similarity graph, for some similarity threshold. Words have been stemmed prior to the calculations. One can clearly recognize the "almost cliques" structure, and in the middle we see an example of two overlapping cliques (the $H = P_3$ case). 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.