Fast Biclustering by Dual Parameterization

Pål Grønås Drange[1], Felix Reidl[2], Fernando Sánchez Villaamil[2], and Somnath Sikdar[2]

1 **Department of Informatics, University of Bergen, Norway**
`pal.drange@ii.uib.no`

2 **Theoretical Computer Science, RWTH Aachen University, Germany**
`reidl,fernando.sanchez,sikdar@cs.rwth-aachen.de`

Abstract

We study two clustering problems, Starforest Editing, the problem of adding and deleting edges to obtain a disjoint union of stars, and the generalization Bicluster Editing. We show that, in addition to being NP-hard, none of the problems can be solved in subexponential time unless the exponential time hypothesis fails.

Misra, Panolan, and Saurabh (MFCS 2013) argue that introducing a bound on the number of connected components in the solution should not make the problem easier: In particular, they argue that the subexponential time algorithm for editing to a fixed number of clusters (p-Cluster Editing) by Fomin et al. (J. Comput. Syst. Sci., 80(7) 2014) is an *exception* rather than the rule. Here, p is a secondary parameter, bounding the number of components in the solution.

However, upon bounding the number of stars or bicliques in the solution, we obtain algorithms which run in time $O(2^{3\sqrt{pk}} + n + m)$ for p-Starforest Editing and $O(2^{O(p\sqrt{k}\log(pk))} + n + m)$ for p-Bicluster Editing. We obtain a similar result for the more general case of t-Partite p-Cluster Editing. This is subexponential in k for a fixed number of clusters, since p is then considered a constant.

Our results even out the number of multivariate subexponential time algorithms and give reasons to believe that this area warrants further study.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases graph editing, subexponential algorithms, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.IPEC.2015.402

1 Introduction

Identifying clusters and biclusters has been a central motif in data mining research [22] and forms the cornerstone of algorithmic applications in, e.g., biology [25] and expression data analysis [7]. Cai [6] showed that clustering – among many other graph modification problems of similar flavor – is solvable in fixed-parameter tractable time. Parallel to these general results, some progress was made in the area of structurally sparse graphs: many problems are, when restricted to classes characterized by a finite set of forbidden minors, solvable in *subexponential parameterized time*, i.e. they admit algorithms with time complexity $2^{o(k)} \cdot \text{poly}(n)$.

The complexity class of problems admitting such an algorithm is called SUBEPT and was defined by Flum and Grohe in the seminal textbook on parameterized complexity [14]. They simultaneously noticed that most natural problems did, in fact, *not* live in this complexity class: The classical NP-hardness reductions paired with the *exponential time hypothesis* of Impagliazzo, Paturi, and Zane [20] is enough to show that no $2^{o(k)} \cdot \text{poly}(n)$ algorithm exists.

10th International Symposium on Parameterized and Exact Computation (IPEC 2015).
Editors: Thore Husfeldt and Iyad Kanj; pp. 402–413
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In this context, Jianer Chen posed the following open problem in the field of parameterized algorithms [5]: Are there examples of natural problems on graphs, that do not have such a topological constraint, and also have subexponential parameterized running time? Alon, Lokshtanov, and Saurabh [1] partially answered this question in the positive by providing a subexponential time algorithm for FEEDBACK ARC SET on tournament graphs. However, the aforementioned graph classes with topological constraints are sparse, and tournament graphs are extremely dense. Chen's question is therefore not fully answered – are there problems which are in SUBEPT on general graphs?

This is indeed the case. Fomin and Villanger [16] showed that MINIMUM FILL-IN was solvable in time $2^{O(\sqrt{k}\log k)} + \mathrm{poly}(n)$. MINIMUM FILL-IN is the problem of completing a graph into a chordal graph by adding as few edges as possible. Following this, a line of research was established investigating whether more graph modification problems admit such algorithms. It proved to be a fruitful area; Since the result by Fomin and Villanger (ibid.), we now know that several graph modification problems towards classes such as split graphs [17], threshold graphs [10], trivially perfect graphs [11], (proper) interval graphs [3, 4], and more admit subexponential time algorithms.

While these classes are rather "simple", they certainly are much more complex than simple cluster or bicluster graphs. Therefore, the problems CLUSTER EDITING and CLUSTER DELETION were a logical candidate for subexponential time algorithms. Surprisingly, we cannot expect that such algorithms exist. Komusiewicz and Uhlmann gave an elegant reduction proving that both parameterized and exact subexponential time algorithms were not achievable, unless ETH fails [21]. On the other hand, the problem p-CLUSTER EDITING, where the number of components in the target class is fixed to be at most p – rather surprisingly – does indeed admit a subexponential parameterized time algorithm; This was shown by Fomin et al. [15], who designed an algorithm solving this problem in time $2^{O(\sqrt{pk})} \cdot \mathrm{poly}(n)$.

Misra, Panolan, and Saurabh explicitly stated their surprise about this result: In their opinion, bounding the number of components in the target graph should in general *not* facilitate subexponential time algorithms [23]: *"We show that this sub-exponential time algorithm for the fixed number of cliques is rather an exception than a rule."*

We show that the related problem BICLUSTER EDITING and its generalization t-PARTITE p-CLUSTER EDITING as well as the special case STARFOREST EDITING also belong to this exceptional class of problems where a bound on the number of target components greatly improves their algorithmic tractability. Since BICLUSTER EDITING is an important tool in molecular biology and biological data analysis, and the necessary second parameter is not outlandish in these settings, we feel that this is a noteworthy insight. We complement these results with NP-completeness proofs for BICLUSTER EDITING and t-PARTITE p-CLUSTER EDITING on subcubic graphs and further show that, unless ETH fails, no parameterized or exact subexponential algorithm is possible without the secondary parameter. That a bound on the maximal degree does not contribute towards making these problems more tractable contrasts many other graph modification problems (like modifications towards split and threshold graphs [24]) which are polynomial-time solvable in this setting.

Previously, it was known BICLUSTER EDITING in general is NP-complete [2], and Guo, Hüffner, Komusiewicz, and Zhang [18] studied the problem from a parameterized point of view, giving a linear problem kernel with $6k$ vertices, and an algorithm solving the problem in time $O(3.24^k + m)$.

Our contribution. In this paper, we study both the very general t-Partite p-Cluster Editing as well as editing to the aforementioned special cases. On the positive side, we show that

- p-Starforest Editing is solvable in time $O(2^{3\sqrt{pk}} + n + m)$, and
- both p-Bicluster Editing and the more general t-Partite p-Cluster Editing are solvable in time $2^{O(p\sqrt{k}\log(pk))} + O(n + m)$ facilitated by a kernel of size $O(ptk)$, where $t = 2$ in the case of p-Bicluster Editing.

In many cases, p is considered a constant, and in this case our kernel has size linear in k. We supplement these algorithms with hardness results; Specifically, we show that

- assuming ETH, Starforest Editing and Bicluster Editing cannot be solved in time $2^{o(k)} \cdot \mathrm{poly}(n)$ and thus neither can t-Partite Cluster Editing, and finally,
- p-Starforest Editing is W[1]-hard if parameterized by p alone.

Organization of the paper. In Section 3 we give a subexponential time parameterized algorithm for the Starforest Editing problem when parameterized by the editing budget and the number of stars in the solution simultaneously. One ingredient for our subexponential algorithms is a polynomial kernel. A kernel for Bicluster Editing exists already [18] and we provide one for the t-partite case in Section 4. In Section 5 we show that p-Bicluster Editing is solvable in subexponential time in k; We give a $2^{O(p\sqrt{k}\log(pk))}+O(n+m)$ algorithm and generalize it to editing to t-partite p-cluster graphs. The parameter p is usually considered to be a fixed constant, hence the running time is truly subexponential, $2^{o(k)} + O(n + m)$ in the editing budget k. However, for a more fine-grained complexity analysis and for lower bounds, we treat p as a parameter.

In Section 6 we show that we cannot expect such an algorithm without an exponential dependency on p; The problem is not solvable in time $2^{o(k)}\,\mathrm{poly}(n)$ unless ETH fails. Further, we show that Starforest Editing is W[1]-hard if parameterized by p alone, before we conclude in Section 7. Due to page limits, some proofs have been deferred to the full version, available online [12].

2 Preliminaries

We consider only finite simple graphs $G = (V, E)$ and we use n and m to denote the size of the vertex set and edge set, respectively. We denote by $N_G(v)$ the set of neighbors of v in G, and let $\deg_G(v) = |N_G(v)|$. We omit subscripts when the graph in question is clear from context. We refer to the monograph by Diestel [9] for graph terminology and notation not defined here. For information on parameterized complexity, we refer to the textbook by Flum and Grohe [14]. We consider an edge in $E(G)$ to be a set of size two, i.e., $e \in E(G)$ is of the form $\{u, v\} \subseteq V(G)$ with $u \neq v$. We denote by $[V(G)]^2$ the set of all size two subsets of G. When $F \subseteq [V(G)]^2$, we write $G \vartriangle F$ to denote $G' = (V, E \triangle F)$, where $\triangle$ is the *symmetric difference*, i.e., $E \triangle F = (E \setminus F) \cup (F \setminus E)$. When the graph is clear from context, we will refer to F simply as a set of edges rather than $F \subseteq [V(G)]^2$.

Let us fix the following terminology: A *star graph* is a tree of diameter at most two (a graph isomorphic to $K_{1,\ell}$ for some ℓ). The degree-one vertices are called *leaves* and the vertex of higher degree the *center*. A *starforest* is a forest whose connected components are stars or, equivalently, a graph that does not contain $\{K_3, P_4, C_4\}$ as induced subgraphs. A biclique is a complete bipartite graph $K_{a,b}$ for some $a, b \in \mathbb{N}$, and a *bicluster graph* is a disjoint union of bicliques. A t-partite clique graph is a graph whose vertex set can be partitioned into at most t independent sets, all pairwise fully connected, and a t-partite cluster graph is a disjoint union of t-partite cliques. The problem of editing towards a starforest (resp. bicluster and t-partite cluster) is the algorithmic problem of adding and deleting as few edges as possible to convert a graph G to a starforest (resp. bicluster and t-partite cluster). We write $f(n) = \mathrm{poly}(n)$ to mean $f(n) = n^{O(1)}$, i.e., that there exists a $c \in \mathbb{N}$ such that $f(n) = O(n^c)$.

Exponential time hypothesis. To show that there is no subexponential time algorithm for Starforest Editing we give a linear reduction from 3Sat, that is, a reduction which constructs an instance whose parameter is bounded linearly in the size of the input formula. The constructed instance will also have *size* bounded linearly in the size of the formula, and we use this to also rule out an exact subexponential algorithm of the form $2^{o(n+m)}$. Pipelining such a reduction with an assumed subexponential parameterized algorithm for the problem would give a subexponential algorithm for 3Sat, contradicting the complexity hypothesis of Impagliazzo, Paturi, and Zane [20]. Their Sparsification Lemma shows that, unless the exponential time hypothesis (ETH) fails, 3Sat cannot be solved in time $2^{o(n+m)}$, where n and m here refer to the number of variables and the number of clauses, respectively.

3 Editing to starforests in subexponential time

A first natural step in handling modification problems related to bicluster graphs is modification towards the subclass of bicluster graphs called starforest. Recall that a graph is a starforest if it is a bicluster where every biclique has one side of size exactly one, or equivalently, every connected component is a star.

Starforest Editing parameterized by k

Input: A graph $G = (V, E)$ and a non-negative integer k.

Question: Is there a set of at most k edges F such that $G \triangle F$ is a disjoint union of stars?

The problem where we only allow to *delete* edges is referred to as Starforest Deletion. These two problems can easily be observed to be equivalent; Adding an edge to a forbidden induced subgraph will create one of the other forbidden subgraphs, or simply put, it never makes sense to add an edge.

In Section 6 we show that this problem is NP-hard, and that it is not solvable in time $2^{o(k)} \mathrm{poly}(n)$ unless the exponential time hypothesis fails.

Multivariate analysis. Since no subexponential algorithm is possible under ETH, we introduce a secondary parameter by p which bounds the number of connected components in a solution graph. This has previously been done with success in the Cluster Editing problem [15]. Hence, we define the following multivariate variant of the above problem.

p-Starforest Editing parameterized by p, k

Input: A graph $G = (V, E)$ and a non-negative integer k.

Question: Is there a set F of edges of size at most k such that $G \triangle F$ is a disjoint union of exactly p stars?

Observe that this problem is *not* the same as p-Starforest Deletion since we might need to merge stars to achieve the desired value p for the number of connected components. In Section 6 we show that the problem is W[1]-hard parameterized by p alone, and that we therefore need to parameterize on both p and k.

▶ **Lemma 1.** *Let* (G, k) *be input to* p-Starforest Editing*. If* (G, k) *is a yes-instance, there can be at most* $p + 2k$ *vertices with degree at least* 2*.*

The following bound will be key to obtain the subexponential running time.

▶ **Proposition 2** ([15])**.** *If* a *and* b *are non-negative integers, then* $\binom{a+b}{a} \leq 2^{2\sqrt{ab}}$*.*

▶ **Lemma 3.** *Given a graph* G *and a vertex set* S*, we can compute in linear time* $O(n + m)$ *an optimal editing set* F *such that* $G \triangle F$ *is a starforest, when restricted to have* S *as the set of centers in the solutions.*

We now describe an algorithm which solves p-Starforest Editing in time $O(2^{3\sqrt{pk}} + n + m)$.

The algorithm. Let (G, k) be an input instance for p-Starforest Editing. If the number of vertices of degree at least two is greater than $p + 2k$, we say no in accordance with Lemma 1. Otherwise we split the graph into G_1 and G_2 as follows: Let $X \subseteq V(G)$ be the collection of vertices contained in connected components of size one or two, i.e., $G[X]$ is a collection of isolated vertices and edges. Let $G_1 = G[X]$ and $G_2 = G[V(G) \setminus X]$. Clearly, there are no edges going out of X in G. We will treat G_1, G_2 as (almost) independent subinstances by guessing the budgets $k_1 + k_2 = k$ and the number of components in their respective solutions $p_1 + p_2 = p$. The only time we cannot treat them as independent instances is when p_1 or p_2 is zero; Let p_i^* be the number of stars completely contained in G_i in an optimal solution. If both $p_i^* > 0$, then there always exist an optimal solution that does not add any edge between G_1 and G_2.

Solving (G_1, k_1) with p_1 components: Assume G_1 contains s isolated edges and t isolated vertices, with $p_1 > 0$. If $|V(G_1)| < p_1$, we immediately say no, since we need exactly p_1 connected components. Depending on the values of s and t, we execute the following operations as long as the budget k_1 is positive. If $s \leq p_1$ and $s + t \leq p_1$, we have too few stars, and we arbitrarily delete edges to increase the number of connected components to p_1.

If $s = 0$ we turn the isolated vertices arbitrarily into p_1 stars. Otherwise, fix an arbitrary endpoint c of an isolated edge. Assume that $s \leq p_1$: then we connect enough isolated vertices to c such that the number of stars is p_1. Finally, if $s > p_1$, we first dissolve $s - p_1$ edges and continue as in the previous case. It is easy to check that the above solutions are optimal.

Solving (G_2, k_2) with p_2 components: By Lemma 1, the number of vertices of degree at least two is bounded by $p_2 + 2k_2$. Every vertex of degree one in G_2 is adjacent to a vertex of larger degree, thus it never makes sense to choose it as a center (its neighbor will always be cheaper). Hence, it suffices to enumerate every set S_2 of p_2 vertices of degree larger than one and test in linear time, as per Lemma 3, whether a solution inside the budget k_2 is possible. Using Proposition 2 we can bound the running time by

$$\binom{p_2 + 2k_2}{p_2} \cdot pk + O(n + m) = O(2^{2\sqrt{2p_2k_2}} \cdot pk + n + m) = O(2^{3\sqrt{p_2k_2}} + n + m).$$

We are left with the cases where p_1 or p_2 are equal to zero: then the only possible solution is to remove *all* edges within G_1 or G_2, respectively, and connect all the resulting isolated vertices to an arbitrary center in the other instance. We either follow through with the operation, if within the respective budget, or deduce that the subinstance is not solvable. We conclude that the above algorithm will at some point guess the correct budgets for G_1 and G_2 and thus find a solution of size at most k. The theorem follows.

▶ **Theorem 4.** p-Starforest Editing *is solvable in time* $O(2^{3\sqrt{pk}} + n + m)$*.*

4 A polynomial kernel for t-partite p-cluster editing

We show a simple $O(ktp)$ kernel for the t-Partite p-Cluster Editing problem – which will be the foundation of the subsequent subexponential algorithms – with a single rule, Rule 1, which can be exhaustively applied in time $O(n+m)$. The problem at hand is the following generalization of p-Bicluster Editing:

> t-Partite p-Cluster Editing parameterized by p, k
>
> *Input:* A graph $G = (V, E)$ and a non-negative integer k.
>
> *Question:* Is there a set $F \subseteq [V]^2$ of edges of size at most k such that $G \triangle F$ is a disjoint union of exactly p complete t-partite graphs?

For our rule, we say that a set $X \subseteq V(G)$ is a *non-isolate twin class* if for every v and v' in X, $N_G(v) = N_G(v') \neq \emptyset$. Note that this is by definition a *false twin class*, i.e., $vv' \notin E(G)$, or in other words, a non-isolate twin class is an independent set.

▶ **Rule 1.** *If there is a non-isolate twin class $X \subseteq V(G)$ of size at least $2k+2$, then delete all but $2k+1$ of them.*

▶ **Lemma 5.** *Rule 1 is sound and can be exhaustively applied in linear time.*

The following result is an immediate consequence of the above rule and its correctness.

▶ **Theorem 6.** *The problem t-Partite p-Cluster Editing admits a kernel where the number of vertices is bounded by $pt(2k+1)+2k = O(ptk)$.*

Proof. We now count the number of vertices we can have in a yes instance after the rule above has been applied. We claim that if G has more than $pt(2k+1)+2k$ vertices, it is a no instance. Let (G, k) be the reduced instance according to Rule 1 and let F be a solution of size at most k. At most $2k$ vertices can be touched by F, so the rest of the graph remains as it is, and is a disjoint union of at most p complete t-partite graphs, each of which has at most t non-isolate twin classes. It follows that in a yes instance, G has at most $pt(2k+1)+2k = O(ptk)$ vertices. ◀

5 Editing to bicluster graphs in subexponential time

In this section we lift the result of Section 3 by showing that the following problem is solvable in time $2^{O(p\sqrt{k}\log(pk))} + O(n+m)$. Observe that we lose the subexponential dependence on p, however, contrary to the result of Misra et al. [23], for fixed (or small, relative to k) p, this still is truly subexponential parameterized by k.

> p-Bicluster Editing parameterized by p, k
>
> *Input:* A graph $G = (V, E)$ and a non-negative integer k.
>
> *Question:* Is there a set $F \subseteq [V]^2$ of edges of size at most k such that $G \triangle F$ is a disjoint union of p complete bipartite graphs?

We denote a biclique of G as $C = (A, B)$ and call the sets A, B the *sides* of C. Before describing the algorithm for the general problem, we show that the following simpler problem is solvable in linear time using a greedy algorithm:

Annotated Bicluster Editing

Input: A bipartite graph $G = (A, B, E)$, a partition $\mathcal{A} = \{A_1, A_2, \ldots, A_p\}$ of A and a non-negative integer k.

Question: Is there a set $F \subseteq [V]^2$ of edges of size at most k such that $G \triangle F$ is a disjoint union of p complete bipartite graphs with each one side in $\mathcal{A}$?

▶ **Lemma 7.** Annotated Bicluster Editing *is solvable in time* $O(n + m)$.

Proof. Let $G = (A, B, E)$, $\mathcal{A} = \{A_1, \ldots, A_p\}$, k be an instance of Annotated Bicluster Editing. Consider a vertex $v \in B$ and define $\mathrm{cost}_i(v)$ to be the cost of placing v in B_i where $C_i = (A_i, B_i)$ is the ith biclique of the solution, i.e.,

$$\mathrm{cost}_i(v) = |A_i| - 2\deg_{A_i}(v) + \deg(v),$$

where $\deg_{A_i}(v) = |N(v) \cap A_i|$. We prove the following claim which implies that we can greedily assign each vertex $v \in B$ to a biclique of minimum cost.

▶ **Claim 8.** *An optimal solution will always have $v \in B$ in a biclique $C_i = (A_i, B_i)$ which minimizes* $\mathrm{cost}_i(v)$.

Suppose that $\mathrm{cost}_i(v)$ is minimal but v is placed by a solution F in a biclique $C_j = (A_j, B_j)$ with $\mathrm{cost}_j(v) > \mathrm{cost}_i(v)$. Deleting from F all edges E_j between v and A_j and adding all edges E_i between v and A_i creates a new solution $F' = (F \setminus E_j) \cup E_i$. Since $\mathrm{cost}_j(v) > \mathrm{cost}_i(v)$, we have that $|F| > |F'|$ hence F is not optimal. This concludes the proof of the claim and the lemma. ◀

5.1 Subexponential time algorithm

We now show that the problem p-Bicluster Editing is solvable in subexponential time by using the kernel from Theorem 6, guessing the annotated sets and applying the polynomial time algorithm for the annotated version of the problem. The important ingredient will be *cheap* vertices, by which we mean vertices that are known to receive very few edits. Intuitively, a cheap vertex is a "pin" that in subexponential time reveals for us its neighborhood in the solution, and thus can be leveraged to uncover parts of said solution.

We adopt the following notation and vocabulary. For an instance (G, k) of p-Bicluster Editing, and a solution F, we call $H = G \triangle F$ the *target* graph. A vertex v is called *cheap* with respect to F if it receives at most $\sqrt{k}$ edits. Observe that any set X of size larger than $2\sqrt{k}$ has a cheap vertex. We call such a set *large* and all sets that contain at most $2\sqrt{k}$ vertices *small*. We will further classify the bicliques in the target graph into two different classes: A biclique is small if its vertex set is small and large otherwise.

The algorithm now works as follows. Given an input instance (G, k) of p-Bicluster Editing, we try all combinations of $p_s + p_\ell = p$, with the intended meaning that p_s is the number of small bicliques and p_ℓ is the number of large bicliques in the target graph.

Handling small bicliques. We enumerate a set of p_s sets $\mathcal{A}_s \subseteq 2^V$ with the property that they are pairwise disjoint, and each of size at most $2\sqrt{k}$. Furthermore, $G[\bigcup \mathcal{A}_s]$ contains at most k edges. Delete all edges in $\mathcal{A}_s$ and reduce the budget accordingly. These are going to be all the left sides in small bicliques. This enumeration takes time

$$(2\sqrt{k})^{p_s} \binom{n}{2\sqrt{k}}^{p_s} \leq (2\sqrt{k})^{p} \binom{pk + k^2}{2\sqrt{k}}^{p} = 2^{O(p\sqrt{k}\log(pk))}.$$

Handling large bicliques. The large bicliques have the following nice property. Since the vertex set of each such biclique is large, every biclique contains a cheap vertex. We guess a set $\mathcal{B}_\ell$ of size p_ℓ. For the biclique C_i, the vertex v_i of $\mathcal{B}_\ell$ will be a cheap vertex in B_i. Now, we enumerate all combinations of p_ℓ sets $\mathcal{N} = \langle N_1, N_2, \ldots, N_{p_\ell}\rangle$, each of size at most $2\sqrt{k}$ which will be the edited neighborhood of each cheap vertex, and we conclude that $A_i = N_H(v_i) = N_G(v_i) \triangle N_i$. The enumeration of this asymptotically takes time

$$\binom{n}{p_\ell} \cdot (2\sqrt{k})^{p_\ell} \binom{n}{2\sqrt{k}}^{p_\ell} \leq \binom{pk+k^2}{p} \cdot (2\sqrt{k})^p \binom{pk+k^2}{2\sqrt{k}}^p = 2^{O(p\sqrt{k}\log(pk))}.$$

Putting things together. With the above two steps, in time $2^{O(p\sqrt{k}\log(pk))}$ we obtained all the left sides $\mathcal{A}$, partitioned into $\mathcal{A}_s$ and $\mathcal{A}_\ell$. Using this information, we can in polynomial time compute whether the Annotated Bicluster Editing instance $(G, k, \mathcal{A})$ is a yes-instance. If so, we conclude yes, otherwise, we backtrack.

▶ **Theorem 9.** *p-Bicluster Editing is solvable in time $2^{O(p\sqrt{k}\log(pk))} + O(n+m)$.*

Proof. We now show that the algorithm described above correctly decides p-Bicluster Editing given an instance (G, k). Suppose that the algorithm above concludes that (G, k) is a yes instance. The only time it outputs yes, is when Annotated Bicluster Editing for a given set $\mathcal{A}$ and a given budget k' outputs yes. Since this budget is the leftover budget from making A an independent set, it is clear that any Annotated Bicluster Editing solution of size at most k' gives a yes instance for p-Bicluster Editing.

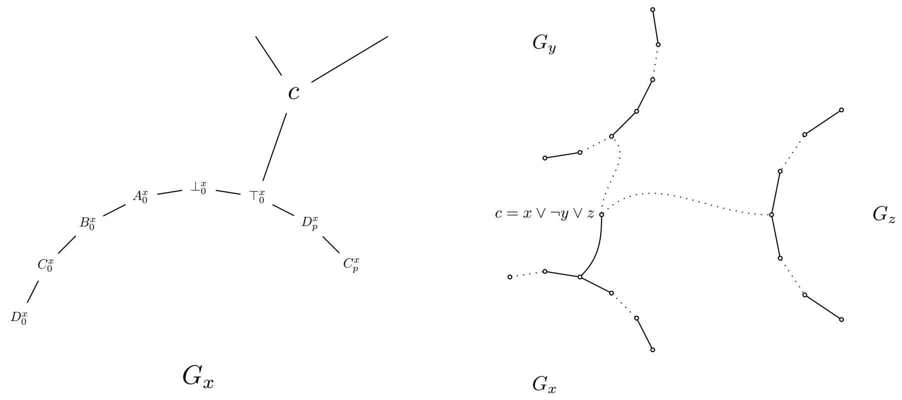
Suppose now for the other direction that (G, k) is a yes instance for p-Bicluster Editing and let F be a solution. Consider the left sides $A_1, \ldots, A_p$ of $G \triangle F$ with the restriction that the smaller of the two sides in C_i is named A_i. First we observe that during our subexponential time enumeration of sets, all the A_is that are of size at most $2\sqrt{k}$ will be enumerated in one of the branches where p_s is set to the number of small bicliques. Furthermore, if A_j is large, then both are large, and then, for each of the large bicliques, there is a branch where we selected exactly one cheap vertex for each of the largest sides. Given these cheap vertices, there is a branch where we guess exactly the edits affecting each of the cheap vertices, hence we can conclude that in some branch, we know the entire partition $\mathcal{A}$. From Lemma 7, we can conclude that the algorithm described above concludes correctly that we are dealing with a yes-instance. ◀

5.2 The t-partite case

We can in fact obtain similar (we treat t here as a constant so the results are up to some constant factors in the exponents) results for the more general case of t-Partite p-Cluster Editing. Again we need the polynomial kernel described in Theorem 6. The only difference now to the bicluster case is that we define a cluster to be small if *every side* is small. In this case, we can enumerate $\binom{n}{\sqrt{k}}^{tp}$ sets, which will form the small clusters.

In the other case a cluster $C = (A_1, A_2, \ldots, A_t)$ is divided into $A_1, A_2, \ldots, A_{t_s}$ small sides and $A_{t_s+1}, A_{t_s+2}, \ldots, A_t$ large sides. For this case, we guess *all* the small sides and for each of the large sides we guess a cheap vertex. Guessing the neighborhoods $N_{t_s+1}, N_{t_s+2}, \ldots, N_t$ for the cheap vertices $v_{t_s+1}, v_{t_s+2}, \ldots, v_t$ gives us complete information on C; To compute what A_j is, if $j > t_s$, we simply take the intersection $\bigcap_{t_s<i\leq t, i\neq j} N_i$ and remove $\bigcup_{i\leq t_s} A_i$. We arrive at the following lemma whose proof is directly analogous to that of Theorem 9.

▶ **Theorem 10.** *The problem t-Partite p-Cluster Editing is solvable in subexponential time $2^{O(p\sqrt{k}\log(pk))} + O(n+m)$.*



(a) Parts of a variable gadget x and its connection when occurring positively in c.

(b) Deletion when x is chosen to satisfy the clause. If we chose not to delete the edge connecting the clause vertex with G_y we would have gotten an induced P_4.

Figure 1 Reduction from 3SAT to starforest editing on subcubic graphs.

6 Lower bounds

We show that (a) STARFOREST EDITING is NP-hard and that we cannot expect a subexponential algorithm unless the ETH fails; and (b) that p-STARFOREST EDITING is W[1]-hard parameterized only by p.

6.1 Starforest editing

In the following we describe a linear reduction from 3SAT to STARFOREST EDITING. Furthermore, the instance we reduce to has maximal degree three, thus not only showing that STARFOREST EDITING is NP-hard on graphs of bounded degree, but also not solvable in subexponential time on subcubic graphs.

▶ **Theorem 11.** *The problem* STARFOREST EDITING *is* NP-*complete and, assuming ETH, does not admit a subexponential parameterized algorithm when parameterized by the solution size k, i.e., it cannot be solved in time $2^{o(k)} \cdot \mathrm{poly}(n)$, nor in exact exponential time $2^{o(n+m)}$, even when restricted to subcubic graphs.*

To prove the theorem above we will reduce from 3SAT. But to obtain the result, it is crucial that in our reduction, both the parameter k, and the size of the instance G are bounded in linearly in n and m. Such results have been shown earlier, in particular by Komusiewicz and Uhlmann for CLUSTER EDITING [21] and Drange and Pilipczuk for TRIVIALLY PERFECT EDITING [13]. Thus we resort to similar reductions as used there, however, the reductions here are tweaked to work for the problem at hand. We also achieve lower bounds for subcubic graphs. See Figures 1a and 1b for figures of the gadgets.

Variable gadget. Let φ be an input instance of 3SAT, and denote its variable set and clause set as $\mathcal{V}(\varphi)$ and $\mathcal{C}(\varphi)$, respectively. We construct for $x \in \mathcal{V}(\varphi)$ a graph $G_x \cong C_{6p_x}$ where p_x is the number of clauses in φ which x appears in. The vertices of G_x are labeled, consecutively, $\top_i^x, \bot_i^x, A_i^x, B_i^x, C_i^x, D_i^x$ for $i \in [0, p_x - 1]$.

There are exactly three ways of deleting G_x into a starforest using at most $k_x = 6p_x$ edges. Clearly a collection of P_3s is a starforest and is our target graph. We will define the $\top$-deletion for G_x as the deletion set $S_\top^x = \{C_i^x D_i^x, \perp_i^x A_i^x \mid i \leq p_x - 1\}$ and the $\perp$-deletion for G_x as the deletion set $S_\perp^x = \{A_i^x B_i^x, D_i^x \top_{i+1}^x \mid i \leq p_x - 1\}$, taking the $i+1$ in the index of $\top_{i+1}^x$ modulo p_x. In other words, in the gadget G_x, we are *keeping* the edges

- $D_{i-1}^x \top_i^x \perp_i^x, A_i^x B_i^x C_i^x$, when x is set to true, and
- $\top_i^x \perp_i^x A_i^x, B_i^x C_i^x D_i^x$, when x is set to false.

Observe that when x is set to true, we will have paths on three vertices, where $\top_i^x$ is the middle vertex, and if x is set to false, we will have paths on three vertices with $\perp_i^x$ being the middle vertex. Later, we will see that if x satisfies a clause c, the ith clause x appears in, then either $\top_i^x$ or $\perp_i^x$ will be the middle vertex of a claw, depending on whether x appears positively or negatively in c.

▶ **Observation 12.** *In an optimal edge edit of a cycle of length divisible by 6, no edge is added and exactly every third consecutive edge is deleted.*

Clause gadget. A clause gadget simply consists of one vertex, i.e., for a clause $c \in \mathcal{C}(\varphi)$, we construct the vertex v_c. This vertex will be connected to G_x, G_y and G_z, for x, y, z being its variables, in appropriate places, depending on whether or not the variable occurs negated in c. In fact, it will be connected to $\top_i^x$ if c is the ith clause x appears in, and x appears positively in c, and it is connected to $\perp_i^x$ if c is the ith clause x appears in, and x appears negatively in c.

Let $k_\varphi = 2|\mathcal{C}| + 2\sum_x p_x = 2|\mathcal{C}| + 3 \cdot 2|\mathcal{C}| = 8|\mathcal{C}|$ be the budget for a formula φ. We now observe that the budget is tight.

▶ **Lemma 13.** *The graph G_φ has no starforest editing set of size less than k_φ, and if the editing set has size k_φ it contains only deletions.*

We now continue to the main lemma, from which Theorem 11 follows.

▶ **Lemma 14.** *A* 3SAT *instance φ is satisfiable if and only if (G_φ, k_φ) is a yes instance for* STARFOREST EDITING.

Observing that the maximum degree of G_φ is three – the clause vertices have exactly degree three, and the variable gadgets are cycles with some vertices connected to at most one clause vertex – this concludes the proof of Theorem 11. From the discussions above, the following result is an immediate consequence:

▶ **Corollary 15.** *The problem* STARFOREST DELETION *is* NP*-complete and not solvable in subexponential time under ETH, even on subcubic graphs.*

Before going into parameterized lower bounds of STARFOREST EDITING, we show that the exact same reduction above simultaneously proves similar results for the bicluster case. We note that the NP-hardness was shown by Amit [2], but their reduction suffers a quadratic blowup and is therefore not suitable for showing subexponential lower bounds.

▶ **Corollary 16.** *The problems* BICLUSTER EDITING *and* BICLUSTER DELETION *are* NP*-complete and not solvable in subexponential time under ETH, even on subcubic graphs.*

6.2 W[1]-hardness parameterized by p

In this section we show that the parameterization by k is necessary, even for the case of p-Starforest Editing. That is, we show that when we parameterize by p alone, the problem becomes W[1]-hard, and we can thus not expect any algorithms of the form $f(p)\cdot \text{poly}(n)$ for any function f solving p-Starforest Editing. We reduce from the problem Multicolored Regular Independent Set. An instance of this problem consists of a regular graph colored into p color classes, each color class inducing a complete graph, and we are asked to find an independent set of size p.

▶ **Proposition 17** ([8]). *The problem* Multicolored Regular Independent Set *is* W[1]*-complete.*

Since each color class is complete, any independent set will be of size at most p and any independent set of size p is maximum. The reduction is direct; In fact we have that given a budget $k = (n-p)(d-1)$, where d is the regularity degree, the following direct translation between the two problems holds:

▶ **Lemma 18.** *Let G be a d-regular graph on n vertices, $p \leq n$ and $k = (n-p)(d-1)$. Then (G,p) is a yes instance for* Multicolored Regular Independent Set *if and only if (G,k) is a yes instance for p-*Starforest Editing*.*

Combining Proposition 17 with Lemma 18 yields the following result:

▶ **Theorem 19.** *p-*Starforest Editing *is* W[1]*-hard when parameterized by p alone.*

7 Conclusion

We presented subexponential time algorithms for editing problems towards bicluster graphs, and more generally, t-partite cluster graphs when the number of connected components in the target graph is bounded. We supplemented these findings with lower bounds, showing that this dual parameterization is indeed necessary.

As an interesting open problem, we pose the question of whether t-Partite p-Cluster Editing can be solved in time $2^{O(\sqrt{pk})}\text{poly}(n)$, i.e., in subexponential time with respect to both parameters. It is known that Bicluster Editing admits a linear kernel, but when introducing the secondary parameter, we only obtain a kernel whose size is bounded by the product of both parameters; Recall that we got a $tp(2k+1)+2k$ kernel, which in the bicluster case is $p(4k+2)+2k$. Does Bicluster Editing admit a truly linear kernel, i.e., a kernel with $O(p+k)$ vertices?

Finally, in many practical applications of biclustering problems, the input can often be considered bipartite. The proof of the NP-completeness and subexponential algorithm lower bounds is highly non-bipartite, hence a natural question is whether it is possible to get similar lower bounds for the problem Bipartite Bicluster Editing, the problem where you are given a bipartite graph and asked to respect the bipartition.

Acknowledgments. We thank Daniel Lokshtanov for pointing us in the direction of the W[1]-completeness of the problem Regular Independent Set [8].

We would like to thank the anonymous reviewers for their feedback which greatly improved the quality of this paper.

Pål Grønås Drange has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 267959, *Rigorous Theory of Preprocessing*.

Fernando Sánchez Villaamil is supported by DFG Project RO 927/13-1, *Pragmatic Parameterized Algorithms*.

References

1 N. Alon, D. Lokshtanov, and S. Saurabh. Fast FAST. In *ICALP*, 2009.
2 N. Amit. The bicluster graph editing problem. Master's thesis, Tel Aviv University, 2004.
3 I. Bliznets, F. V. Fomin, M. Pilipczuk, and M. Pilipczuk. Subexponential parameterized algorithm for interval completion. *SODA*. ACM-SIAM, 2016. To appear.
4 I. Bliznets, F. V. Fomin, M. Pilipczuk, and M. Pilipczuk. A subexponential parameterized algorithm for proper interval completion. In *ESA*. Springer, 2014.
5 H. L. Bodlaender, L. Cai, J. Chen, M. R. Fellows, J. A. Telle, and D. Marx. Open problems in parameterized and exact computation, 2006. IWPEC.
6 L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Proc. Lett.*, 58(4):171–176, 1996.
7 Y. Cheng and G. M. Church. Biclustering of expression data. In *ISMB*. AAAI Press, 1999.
8 M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.
9 R. Diestel. *Graph theory (Graduate texts in mathematics)*. Springer, 2005.
10 P. G. Drange, M. Dregi, D. Lokshtanov, and B. D. Sullivan. On the threshold of intractability. In *ESA*. Springer, 2015.
11 P. G. Drange, F. V. Fomin, M. Pilipczuk, and Y. Villanger. Exploring subexponential parameterized complexity of completion problems. In *STACS*. Schloss Dagstuhl, 2014.
12 P. G. Drange, F. Reidl, F. Sánchez Villaamil, and S. Sikdar. Fast biclustering by dual parameterization. *CoRR*, abs/1507.08158, 2015.
13 P. G. Drange and M. Pilipczuk. A polynomial kernel for trivially perfect editing. In *ESA*. Springer, 2015.
14 J. Flum and M. Grohe. *Parameterized complexity theory*. Springer, 2006.
15 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.
16 F. V. Fomin and Y. Villanger. Subexponential parameterized algorithm for minimum fill-in. *SIAM J. Comput.*, 42(6):2197–2216, 2013.
17 E. Ghosh, S. Kolay, M. Kumar, P. Misra, F. Panolan, A. Rai, and M. S. Ramanujan. Faster parameterized algorithms for deletion to split graphs. *Algorithmica*, 71(4):989–1006, 2015.
18 J. Guo, F. Hüffner, C. Komusiewicz, and Y. Zhang. Improved algorithms for bicluster editing. In *TAMC*. Springer, 2008.
19 M. Habib and C. Paul. A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59, 2010.
20 R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.
21 C. Komusiewicz and J. Uhlmann. Cluster editing with locally bounded modifications. *Discrete Appl. Math.*, 160(15):2259–2270, 2012.
22 S. C. Madeira and A. L Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Trans. Comput. Biol. Bioinform*, 1(1):24–45, 2004.
23 N. Misra, F. Panolan, and S. Saurabh. Subexponential algorithm for d-cluster edge deletion: Exception or rule? In *MFCS*. Springer, 2013.
24 A. Natanzon, R. Shamir, and R. Sharan. Complexity classification of some edge modification problems. *Discrete Appl. Math.*, 113(1):109–128, 2001.
25 A. Tanay, R. Sharan, and R. Shamir. Biclustering algorithms: A survey. *Handbook of Comp. Mol. Biol.*, 9(1-20):122–124, 2005.