# Parameterized Algorithms for Partitioning Graphs into Highly Connected Clusters*

**Ivan Bliznets[1] and Nikolai Karpov[2]**

1 **V.A. Steklov Institute of Mathematics of the Russian Academy of Sciences, St. Petersburg, Russia**
2 **V.A. Steklov Institute of Mathematics of the Russian Academy of Sciences, St. Petersburg, Russia**

## Abstract

Clustering is a well-known and important problem with numerous applications. The graph-based model is one of the typical cluster models. In the graph model, clusters are generally defined as cliques. However, such an approach might be too restrictive as in some applications, not all objects from the same cluster must be connected. That is why different types of cliques relaxations often considered as clusters.

In our work, we consider a problem of partitioning graph into clusters and a problem of isolating cluster of a special type where by cluster we mean highly connected subgraph. Initially, such clusterization was proposed by Hartuv and Shamir. And their HCS clustering algorithm was extensively applied in practice. It was used to cluster cDNA fingerprints, to find complexes in protein-protein interaction data, to group protein sequences hierarchically into superfamily and family clusters, to find families of regulatory RNA structures. The HCS algorithm partitions graph in highly connected subgraphs. However, it is achieved by deletion of not necessarily the minimum number of edges. In our work, we try to minimize the number of edge deletions. We consider problems from the parameterized point of view where the main parameter is a number of allowed edge deletions. The presented algorithms significantly improve previous known running times for the Highly Connected Deletion (improved from $O^*\left(81^k\right)$ to $O^*\left(3^k\right)$), Isolated Highly Connected Subgraph (from $O^*(4^k)$ to $O^*\left(k^{O\left(k^{2/3}\right)}\right)$), Seeded Highly Connected Edge Deletion (from $O^*\left(16^{k^{3/4}}\right)$ to $O^*\left(k^{\sqrt{k}}\right)$) problems. Furthermore, we present a subexponential algorithm for Highly Connected Deletion problem if the number of clusters is bounded. Overall our work contains three subexponential algorithms which is unusual as very recently there were known very few problems admitting subexponential algorithms.

**1998 ACM Subject Classification** G.2.2 Graph Theory, F.2.2 Nonnumerical Algorithms and Problems, I.5.3 Clustering, H.3.3 Information Search and Retrieval

**Keywords and phrases** clustering, parameterized complexity, highly connected

**Digital Object Identifier** 10.4230/LIPIcs.MFCS.2017.6

## 1 Introduction

Clustering is a problem of grouping objects such that objects in one group are more similar to each other than to objects in other groups. Clustering has numerous applications, including: machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics. Graph-based model is one of the typical cluster

* This research was supported by Russian Science Foundation (project 16-11-10123)


42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017).
Editors: Kim G. Larsen, Hans L. Bodlaender, and Jean-Francois Raskin; Article No. 6; pp. 6:1–6:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

models. In a graph-based model most commonly cluster is defined as a clique. However, in many applications, such definition of a cluster is too restrictive [17]. Moreover, clique model generally leads to computationally hard problems. For example clique problem is $W[1] - hard$ while $s$-club problem, with $s \geq 2$, is fixed-parameter tractable with respect to the parameters solution size and $s$ [19]. Because of the two mentioned reasons researchers consider different clique relaxation models [17, 20]. We mention just some of the possible relaxations: $s$-club(the diameter is less than of equal to $s$), $s$-plex (the smallest degree is at least $|G| - s$), $s$-defective clique (missing $s$ edges to complete graph), $\gamma$-quasi-clique ($|E|/\binom{|V|}{2} \geq \gamma$), highly connected graphs (smallest degree bigger than $|G|/2$) and others. With different degree of details all these relaxations were studied: $s$-club[19, 20], $s$-plex [14, 1], $s$-defective clique [21, 7], $\gamma$-quasi-clique [18, 16], highly connected graphs [12, 11, 9].

In this work, we study the clustering problem based on highly connected components model. A graph is *highly connected* if the edge connectivity of a graph(the minimum number of edges whose deletion results in a disconnected graph) is bigger than $\frac{n}{2}$ where $n$ is the number of vertices in a graph. An equivalent characterization is for each vertex has degree bigger than $\frac{n}{2}$, it was proved in [3]. One of the reasons for this choice is a huge success in applications of the Highly Connected Subgraphs(HCS) clustering algorithm proposed by Hartuv and Shamir and the second reason is the lack of research for this model compared with the standard clique model. HCS algorithm was used [11] to cluster cDNA fingerprints [8], to find complexes in protein-protein interaction data [10], to group protein sequences hierarchically into superfamily and family clusters [13], to find families of regulatory RNA structures [15].

Hüffner et al. [11] noted that while Hartuv and Shamir's algorithm partitions a graph into highly connected components, it does not delete the minimum number of edges required for such partitioning. That is why they initiated study of the following problem

Highly Connected Deletion
**Instance:** Graph $G = (V, E)$.
**Task:** Find edge subset $E' \subseteq E$ of the minimum size such that each connected component of $G' = (V, E \setminus E')$ is highly connected.

For this problem, Hüffner et al. [11] proposed an algorithm which is based on the dynamic programming technique with the running time bounded by $O^*(3^n)$ where $n$ is the number of vertices. For parameterized version of the problem they proposed an algorithm with the running time $O^*(81^k)$ where $k$ is an upper bound on the size of $E'$. Additionally, they proved that the problem admits a kernel with the size $O(k^{1.5})$. Moreover, they proved conditional lower bound on the running time of algorithms for Highly Connected Deletion , in particular, the problem cannot be solved in time $2^{o(k)} \cdot n^{O(1)}$, $2^{o(n)} \cdot n^{O(1)}$ , or $2^{o(m)} \cdot n^{O(1)}$ unless the exponential-time hypothesis (ETH) fails.

Moreover, in another work Hüffner et al. [12] studied a parameterized complexity of related problem of finding highly connected components in a graph.

Isolated Highly Connected Subgraph
**Instance:** Graph $G = (V, E)$, integer $k$, integer $s$.
**Task:** Is there a set of vertices $S$ such that $|S| = s$, $G[S]$ is highly connected graph and $|E(S, V \setminus S)| \leq k$.

Seeded Highly Connected Edge Deletion
**Instance:** Graph $G = (V, E)$, subset $S \subseteq V$, integer $a$, integer $k$.
**Task:** Is there a subset of edges $E' \subseteq E$ of size at most $k$ such that $G - E'$ contains only isolated vertices and one highly connected component $C$ with $S \subseteq V(C)$ and $|V(C)| = |S| + a$.

They proposed algorithms with the running time $O^*(4^k)$ and $O^*(16^{k^{3/4}})$ respectively.

**Table 1** Results.

| Problem | Previous result | Our result |
|---|---|---|
| Highly Connected Deletion (exact) | $O^*\left(3^n\right)$ | $O^*\left(2^n\right)$ |
| Highly Connected Deletion (parameterized) | $O^*\left(81^k\right)$ | $O^*\left(3^k\right)$ |
| $p$-Highly Connected Deletion | - | $O^*\left(2^{O\left(\sqrt{pk}\right)}\right)$ |
| Isolated Highly Connected Subgraph | $O^*(4^k)$ | $O^*\left(k^{O\left(k^{2/3}\right)}\right)$ |
| Seeded Highly Connected Edge Deletion | $O^*\left(16^{k^{3/4}}\right)$ | $O^*\left(k^{\sqrt{k}}\right)$ |

**Our results.** We propose algorithms which significantly improve previous upper bounds. Running times of algorithms may be found in a Table 1. We would like to note that three of the algorithms have subexponential running time which is not common. Until very recently there were very few problems admitting subexponential running time. To our mind in algorithm for Isolated Highly Connected Subgraph problem we have an unusual branching procedure as in one branch parameter is not decreasing. However, the value of subsequent decrementation of parameter in this branch is increasing which leads to subexponential running time. We find the fact interesting as we have not met such behavior of branching procedures before. Presented analysis for this case might be useful in further development of subexponential algorithms.

# 2 Algorithms for partitioning

## 2.1 Highly Connected Deletion

In this section we present an algorithm for Highly Connected Deletion problem. Our algorithm is based on the fast subset convolution. Let $f, g : 2^X \to \{0, 1, \dots M\}$ be two functions and $|X| = n$. Björklund et al. in [2] proved that function $f * g : 2^X \to \{0, \dots, 2M\}$, where $(f * g)(S) = \min\limits_{T \subseteq S} (f(T) + g(S \setminus T))$, can be computed on all subsets $S \subseteq X$ in time $O(2^n \text{poly}(n, M))$.

▶ **Theorem 1.** *There is a $O^*(2^n)$ time algorithm for* Highly Connected Deletion *problem.*

**Proof.** Let define function $f$ in the following way

$$f(S) = \begin{cases} |E(S, V \setminus S)| & \text{if } G[S] \text{ is highly connected} \\ \infty & \text{otherwise} \end{cases}$$

Consider function $f^{*k}(V) = \underbrace{f * \dots * f}_{\text{k times}}$.

Note that $f^{*k}(V) = \min\limits_{S_1 \sqcup \dots \sqcup S_k = V} (f(S_1) + \dots + f(S_k))$. Hence, to solve the problem it is enough to find minimum of $f^{*k}(V)$ over all $1 \le k \le n$. Note that if $f^{*k}(V) = \infty$ then it is not possible to partition $V$ into $k$ highly connected components. So if the minimum value of $f^{*k}(V)$ is $\infty$ then there is no partitioning of $G$ into highly connected components.

Our algorithm contains the following steps.

1. Compute $f$, i.e. compute value $f(S)$ for all $S \subset V$. It takes $O(2^n(n + m))$ time.
2. Using Björklund et al.[2] algorithm iteratively compute $f^{*i}$ for all $1 \le i \le n$.
3. Find $k$ such that $f^{*k}(V)$ is minimal.

After we perform above steps we will know values of functions $f^{*i}$ on each subset $S \subseteq X$. Let $S_1 \sqcup S_2 \sqcup \cdots \sqcup S_k$ be an optimum partitioning of $X$ into highly connected components. Knowing values of function $f^{*k-1}$ and $f$ it is straightforward to restore $S_k$ in time $2^n$. Moreover, knowing $f^{*k-1}, S_k$ we can find value of $S_{k-1}$. Proceeding this way we obtain the optimum partitioning. As $k \leq n$, we spent at most $O(n2^n)$ time to find all $S_i$.

It is left to show how to compute all $f^{*i}$ within $O^*(2^n)$ time. The only obstacle why we cannot straightforwardly apply Björklund's algorithm is that $f$ sometimes takes infinite value. It is easy to fix the problem by replacing infinity value with $2m+1$. We know that each convolution require $O(2^n \text{poly}(n, M))$ time and above we show that we can put $M$ to be equal $2m+1$. As we need to perform $n$ subset convolutions. So, the running time of second step is $O^*(2^n)$. Hence, the overall running time is $O^*(2^n)$. ◀

Now we consider parameterized version of Highly Connected Deletion problem (one is asked whether it is possible to delete at most $k$ edges and get a vertex disjoint union of highly connected subgraphs).

▶ **Theorem 2.** *There is an algorithm for* Highly Connected Deletion *problem with running time $O^*(3^k)$.*

**Proof.** Before we proceed with the proof of the theorem we list several simplification rules and lemmas proved by Hüffner et al. in [11].

▶ **Rule 3.** If $G$ contains a connected component $C$ which is highly connected then replace original instance with instance $(G[V \setminus V(C)], k)$.

▶ **Lemma 4.** *Let $G$ be a highly connected graph and $u, v \in V(G)$ be two different vertices from $V(G)$. If $uv \in E$, then $|N(u) \cap N(v)| \geq 1$. If $uv \notin E$ then $|N(u) \cap N(v)| \geq 3$.*

▶ **Rule 5.** If $u, v \in E$ and $N(u) \cap N(v) = \varnothing$ then delete edge $uv$ and decrease parameter $k$ by 1. The obtained instance is $((V, E \setminus \{uv\}), k-1)$.

▶ **Definition 6.** Let us call vertices $u, v$ *$k$-connected* if any cut separating these two vertices has size bigger than $k$.

▶ **Rule 7.** Let $S$ be an inclusion maximal set of pairwise $k$-connected vertices and $|S| > 2k$. If the induced graph $G[S]$ is not highly connected then our instance is a NO-instance(it is not possible to delete $k$ edges and obtain vertex disjoint union of higly connected subgraphs). Otherwise, we replace original instance with an instance $(G[V \setminus S], k - |E(S, V \setminus S)|)$.

▶ **Lemma 8.** *If $G$ is highly connected then $diam(G) \leq 2$.*

It was shown in [11] that all of the above rules are applicable in polynomial time.

Without loss of generality assume that $G$ is connected. Otherwise, we consider several independent problems. One problem for each connected component. For each connected component we find minimum number of edges that we have to delete in order to partition this component into highly connected subgraphs. Note that in order to find a minimum number for each subproblem we simply consider all possible values of parameter starting from 0 to $k$.

From Lemma 8 follows that if $dist(u, v)$ (distance between two vertices $u, v$) is bigger than 2 then in optimal partitioning $u$ and $v$ belong to different connected components. Hence, if $dist(u, v) \geq 3$ then at least one edge from the shortest path between $u$ and $v$ belongs to $E'$. If $diam(G) > 2$ then it is possible to find two vertices $u, v$ such that $dist(u, v) = 3$. So given the shortest path $u, x, y, v$ we can branch to three instances $(G \setminus ux, k-1)$, $(G \setminus xy, k-1)$,

$(G \setminus yv, k-1)$. We apply such branching exhaustively. Finally, we obtain instance with a graph $G'$ of diameter 2.

Now, for our algorithm it is enough to consider a case when graph $G$ has the following properties: (i) $diam(G) \leq 2$; (ii) there are no subsets $S$ of pairwise $k$-connected vertices with $|S| > 2k$; (iii) $G$ is not highly connected.

From now on we assume that $G$ has above mentioned properties. Suppose $C_1 \sqcup C_2 \sqcup \cdots \sqcup C_\ell$ is an optimum partitioning of $G$ into highly connected graphs and $E'$ is a subset of removed edges. We call vertex *affected* if it is incident with an edge from $E'$. Otherwise, it is *unaffected*. Denote by $U$ the set of all unaffected vertices and by $T$ the set of all affected vertices. By $C(v)$ we denote a cluster $C_i$ for which $v \in C_i$. Note that for affected vertex $u$ there is vertex $v$ such that $uv \in E(G)$ and $v \notin C(u)$.

▶ **Lemma 9.** *Let $G$ be a graph with diameter 2 then for any optimum partitioning $C_1 \sqcup C_2 \sqcup \cdots \sqcup C_\ell$ of $G$ into highly connected graphs there is an $i$ such that $U$ is contained in $C_i$.*

**Proof.** Assume that there are two unaffected vertices $u, v \in U$ and $C(v) \neq C(u)$. Note that any path between $u$ and $v$ must contain an edge from $E'$ and two different edges contained in $C(u), C(v)$ and incident to $u$ and $v$ correspondingly. So, the shortest path between $u$ and $v$ contains at least three edges which contradict our assumption that $diam(G) \leq 2$. Hence, there is an $i$ such that $U \subseteq C_i$. ◀

▶ **Lemma 10.** *Let $G$ be a graph with diameter 2 and optimum partitioning $C_1 \sqcup C_2 \sqcup \cdots \sqcup C_\ell$ into highly connected graphs. If $U$ is not empty then $|E'| \geq n - |C_i|$ where $U \subseteq C_i$.*

**Proof.** Consider an arbitrary unaffected vertex $u$. For any $v \in V$ we have $dist(v,u) \leq 2$. Hence, for any $v \notin C(u)$ there is an edge connecting component $C(u)$ with vertex $v$ as otherwise we have $dist(u,v) > 2$. So we have $|E'| \geq n - |C(u)|$. ◀

For any `YES`-instance we have $k \geq |E'| \geq \frac{|T|}{2}$, $n = |T| + |U|$, and $|U| \leq 2k$.The inequality $|U| \leq 2k$ follows from the simplification Rule 7 and Lemma 9. As otherwise highly connected component which contains $U$ is bigger than $2k$ and hence simplification Rule 7 can be applied which leads to contradiction. So, it means that $n = |T| + |U| \leq 4k$.

Below we present two algorithms. One of these algorithms solves the problem under assumption that optimum partitioning contains at least one unaffected vertex, the other one solves the problem under assumption that all vertices are affected in optimum partitioning. In order to estimate running time of the algorithms we use the following lemma.

▶ **Lemma 11.** *[5] For any non-negative integer $a$, $b$ we have $\binom{a+b}{b} \leq 2^{2\sqrt{ab}}$.*

At first, consider a case when there is at least one unaffected vertex in optimum partitioning.

▶ **Lemma 12.** *Let $G$ be a connected graph with diameter at most 2. If there is an optimum partitioning $C_1 \sqcup C_2 \sqcup \cdots \sqcup C_\ell$ of $G$ into highly connected graphs such that set of unaffected vertices is not empty then* Highly Connected Deletion *can be solved in $O^*(2^{\frac{3k}{2}})$ time.*

**Proof.** Let us fix some unaffected vertex $u$ (in algorithm we simply brute-force all $n$ possible values for unaffected vertex $u$). By Lemma 10 highly connected graph $C(u)$ contains at least $n-k$ vertices. As $u$ is unaffected then $N(u) \subset C(u)$ and $|N(u)| > \frac{|C(u)|}{2}$. Consider set $V \setminus N[u]$. And partition it into two subsets $W_{1,2} \sqcup W_{\geq 3}$, where $W_{1,2} = \{v | 1 \leq |N(u) \cap N(v)| \leq 2\}$, and $W_{\geq 3} = \{v | 3 \leq |N(u) \cap N(v)|\}$. From lemma 4 follows that $W_{1,2} \cap C(u) = \varnothing$. Note that knowing set $C_{part} = C(u) \cap W_{\geq 3}$ we can find set $C(u) = C_{part} \cup N[u]$ and after this simply run algorithm from Theorem 1 on set $V(G) \setminus C(u)$. We implement this approach.

We know that $N[u] \sqcup C_{part} = C(u)$ and $C(u) \leq 2k$. As $|C_{part}| \leq \frac{C(u)}{2}$ it follows that $|C_{part}| \leq k$. Brute-force over all possible values of $s = |C_{part}|$. Having fixed value of $s$ we enumerate all subsets of $W_{\geq 3}$ of size $s$. All such subsets are potential candidates for a $C_{part}$ role. It is possible to enumerate candidates with polynomial delay i.e. in $O^*(\binom{|W_{\geq 3}|}{|C_{part}|})$ time.

For each listed candidate we run algorithm from Theorem 1. Let $R = W_{\geq 3} \setminus C_{part}$. Hence, the overall running time for a fixed $|C_{part}|$ is bounded by $O^*(2^{|R \cup W_{1,2}|}\binom{|W_{\geq 3}|}{|C_{part}|}) = O^*(2^{|R \cup W_{1,2}|}\binom{|C_{part}|+|R|}{|C_{part}|})$. By Lemma 11 we have:

$O^*(2^{|R \cup W_{1,2}|}\binom{|C_{part}|+|R|}{|C_{part}|}) = O^*(2^{2\sqrt{|C_{part}||R|}+|R|+|W_{1,2}|})$.

We know that $|C_{part}| \leq k$, $3|R| + |W_{1,2}| \leq k$, hence $O^*(2^{2\sqrt{|C_{part}||R|}+|R|+|W_{1,2}|}) \leq O^*(2^{2\sqrt{k|R|}-2|R|+k})$. The function $g(t) = 2\sqrt{kt} - 2t + k$ attains it maximum when $t = \frac{k}{4}$. So the running time in the worst case is $O^*(2^{1.5k})$. ◀

It is left to construct an algorithm for a case in which all vertices are affected in optimum partitioning. First of all note that if $n \leq 1.57k \leq k \log_2 3$ we can simply run Algorithm 1 and it finds an answer in $O^*(2^n) = O^*(3^k)$ time. Taking into account that all vertices are affected we have that $n \leq 2k$. So we may assume that $1.57k \leq n \leq 2k$.

▶ **Lemma 13.** *Let $G$ be a graph with diameter 2 and $|V(G)| \geq 1.57k$. Moreover, $(G, k)$* Highly Connected Deletion *problem admits correct partitioning into highly connected components $C_1 \sqcup C_2 \sqcup \dots \sqcup C_\ell$ such that all vertices are affected in this partitioning. Then there are two highly connected components $C_i, C_j$ such that $|C_i| + |C_j| \geq n - k$.*

**Proof.** Let $E'$ be set of deleted edges for partitioning $C_1 \sqcup C_2 \sqcup \dots \sqcup C_\ell$. From $n \geq 1.57k$ follows that in graph $(V(G), E')$ there is a vertex $s$ of degree 1, let $st \in E'$ be the edge. We prove that $C(s), C(t)$ are desired highly connected components. As $diam(G) \leq 2$ then for any vertex $v \in V(G) \setminus C(s) \setminus C(t)$ there is path of length at most 2 from $s$ to $v$. Hence, any vertex $v \in V(G) \setminus C(s) \setminus C(t)$ should be connected with $C(s) \cup C(t)$ in graph $G$. As $|E'| \leq k$ then $V(G) \setminus (C(s) \cup C(t)) \leq k$. So $|C(s)| + |C(t)| \geq n - k$. ◀

Now we brute-force all vertices as candidates for a role of vertex $s$, i.e. vertex of degree 1 in solution $E'$. Consider two possibilities either $|C(s)| > 2n - 3.14k$ or $|C(s)| \leq 2n - 3.14k$.

Consider the first case, if $|C(s)| > 2n - 3.14k$, then we find solution in $O^*(2^{n - \frac{|C(s)|}{2}}) = O^*(3^k)$ time. In order to do this we consider $deg_G(s)$ cases. Each case correspond to a different edge $st$ incident with $s$. Such an edge we treat as the only edge incident with $s$ from $E'$. Having fixed an edge $st$ being from $E'$ we know that all other edges incident with $s$ belong to $E(C(s))$. Denote the set of endpoints of these edges to be $U$. So we can identify at least $\frac{|C(s)|}{2}$ vertices from $C(s)$. Now we can apply the same technique as in proof of Theorem 1.

We define three functions $f, g, h$ over subsets of $W = V \setminus U$.

- $f(S) = |E(S, W \setminus S)|$ if $G[S]$ is highly connected, otherwise it is equal to $\infty$.
- $h(S) = \min_i(f^{*i}(S))$.
- $g(S) = 2|E(W \setminus S, U)| + |E(S, W \setminus S)|$ if $G[U \cup S]$ is highly connected otherwise it is $\infty$.

Let us provide some intuition standing behind the formulas. Value $f(S)$ indicate number of vertices that we have to delete in order to separate highly connected graph $G[S]$. $h(S)$ is a number of edges needed to be deleted in order to separate $G[S]$ into highly connected components. $g(S)$ in some sense is a number of edge deletion needed to create a highly connected component $U \cup S$ which contains vertex $s$. We show that to solve the problem it is enough to compute $(g * h)(W)$. In similar way to Theorem 1 $(g * h)(W)/2$ equals to a number of optimum edge deletions. Note that all deleted edges not having endpoints in $C(s)$ will be calculated two times, one for each of its incident highly connected component, see definition of function $h$. Each edge of $E'$ having an endpoint in $U$ is counted twice in first term of function $g$. And finally each edge from $E'$ having endpoint in $C(s) \setminus U$ is counted twice, once in second term of the formula of $g$, and once in the formula of $h$. So $(g * h)(W)/2$ is required number of edge deletions.

Second case, if $|C(s)| \leq 2n - 3.14k$ then $n - k \leq |C(s)| + |C(t)| \leq 2n - 3.14k + |C(t)|$.

It follows that $|C(t)| + 2n - 3.14k \geq n - k$. Hence, $C(t) \geq 2.14k - n \geq 0.14k$. It means that in $C(t)$ there is a vertex of degree at most 7 in graph $(V(G), E')$. We brute-force all candidates for such vertex and for such edges from $E'$. Having fixed the candidates, vertex $t'$ and at most seven edges, we identify more than a half vertices from $C(t') = C(t)$ in the following way. All edges incident to $t'$ except just fixed set of candidates belong to $C(t)$. Denote the endpoints of these edges as $U_t$. In the same way, all edges incident with $s$ except $st$ belong to $C(s)$. Denote by $U_s$ endpoints of edges incident with $s$ except the edge $st \in E'$. Let $U = U_s \cup U_t$. Below we show how to solve obtained problem in $O^*\left(2^{n-\frac{1}{2}(|C(s)|+|C(t)|)}\right)$ time. As in previous case we apply idea similar to algorithm from Theorem 1. Now we present only functions which convolution give an answer. As the further details are identical to Theorem 1.

Our functions are defined over subsets of a set $W = V \setminus U$.

- $f(S) = |E(S, W \setminus S)|$ if $G[S]$ is highly connected, otherwise $\infty$.
- $h(S) = \min\limits_i \left(f^{*i}(S)\right)$.
- $g_s(S) = 2|E(S, U_t)| + |E(S, W \setminus S)|$ if $G[S \cup U_s]$ is highly connected, otherwise $\infty$.
- $g_t(S) = 2|E(S, U_s)| + |E(S, W \setminus S)|$ if $G[S \cup U_t]$ is highly connected, otherwise $\infty$.

The only difference from previous case is that we constructed two functions $g_s, g_t$ instead of just one function $g$ as now we know two halves of two guessed highly connected components. Minimum number of edge deletions in YES-instance separating clusters $C(s), C(t)$ ($U_s \subseteq C(s), U_t \subseteq C(t)$) is $(h * g_s * g_t)(W)/2$. So in this case we need $O^*(2^{|W|})$ running time which is $O^*\left(2^{n-\frac{(n-k)}{2}}\right) = O^*\left(2^{\frac{3k}{2}}\right)$. ◀

## 2.2 $p$-Highly Connected Deletion

| |
|---|
| $p$-HIGHLY CONNECTED DELETION |
| **Instance:** Graph $G = (V, E)$, integer numbers $p$ and $k$. |
| **Task:** Is there a subset of edges $E' \subset E$ of size at most $k$ such that $G - E'$ contains at most $p$ connected components and each component is highly connected? |

Our algorithm for $p$-HIGHLY CONNECTED DELETION is insipired by algorithm for $p$-CLUSTER EDITING by Fomin et al. [5].

First of all, we prove an upper bound on the number of small cuts in highly connected graph.

▶ **Lemma 14.** *Let $G = (V, E)$ be highly connected graph, $X = \operatorname*{arg\,min}\limits_{\substack{S \subset V \\ \frac{|V|}{4} \leq |S| \leq \frac{3|V|}{4}}} |E(S, V \setminus S)|$, and $Y = V \setminus X$, then*

**(i)** *If $|E(X, Y)| \geq \frac{|V|^2}{100}$ then for any partition of $V = A \sqcup B$ we have $|E(A, B)| \geq \frac{|A| \cdot |B|}{100}$.*

**(ii)** *If $|E(X, Y)| < \frac{|V|^2}{100}$ then for any partition of $V = A \sqcup B$ we have:*
$|E(A \cap X, B \cap X)| \geq \frac{|X \cap A| \cdot |X \cap B|}{100}$, $|E(A \cap Y, B \cap Y)| \geq \frac{|Y \cap A| \cdot |Y \cap B|}{100}$,
$|E(A, B)| \geq \frac{|X \cap A| \cdot |X \cap B|}{100} + \frac{|Y \cap A| \cdot |Y \cap B|}{100}$.

**Proof.** *i)* Let $V = A \sqcup B$. Without loss of generality $|A| < |B|$.

If $\frac{|V|}{4} \le |A|$ then $|E(X,Y)| \le |E(A,B)|$. Hence, $|E(A,B)| \ge |E(X,Y)| \ge \frac{|V|^2}{100} \ge \frac{|A|\cdot|B|}{100}$.

If $|A| < \frac{|V|}{4}$ then $|E(A,B)| \ge \sum\limits_{v \in A} (\deg(v) - |A|)$. As $\deg(v) > \frac{|V|}{2}$ for all $v \in V(G)$, we have $|E(A,B)| \ge |A|\left(\frac{|V|}{2} - |A|\right) \ge \frac{|A|\cdot|V|}{4} \ge \frac{|A|\cdot|B|}{4} \ge \frac{|A|\cdot|B|}{100}$.

*ii)* Note that $|E(A,B)| \ge |E(A\cap X, B\cap X)| + |E(A\cap Y, B\cap Y)|$. So it is enough to prove that $|E(A\cap X, B\cap X)| \ge \frac{|A\cap X|\cdot|B\cap X|}{50}$, as the proof of $|E(A\cap Y, B\cap Y)| \ge \frac{|A\cap Y|\cdot|B\cap Y|}{50}$ is analogous. The sum of these two inequalities gives the proof of the theorem.

Without loss of generality $|B\cap X| \le |A \cap X|$. Hence, $\frac{|V|}{8} \le |A\cap X|$ and $|B\cap X| \le \frac{3|V|}{8}$. Consider two cases: $|A\cap X| \ge \frac{|V|}{4}$ and $|A\cap X| < \frac{|V|}{4}$.

Consider case when $|A\cap X| \ge \frac{|V|}{4}$. At first we prove $|E(A\cap X, B\cap X)| \ge |E(B\cap X, Y)|$. It is known that:

$$|E(A\cap X, V\setminus(A\cap X))| = |E(X,Y)| - |E(B\cap X, Y)| + |E(A\cap X, B\cap X)|, \tag{1}$$

$|A\cap X| \ge \frac{|V|}{4}$, and $|V\setminus(A\cap X)| \ge |Y| \ge \frac{|V|}{4}$, it means $|E(A\cap X, V\setminus(A\cap X))| \ge |E(X,Y)|$. The last inequality and (1) imply $|E(A\cap X, B\cap X)| \ge |E(B\cap X, Y)|$. It follows that $2|E(A\cap X, B\cap X)| \ge |E(B\cap X, A\cap X)| + |E(B\cap X, Y)| = |E(B\cap X, V\setminus(B\cap X)|$.

As $\frac{3|V|}{8} \ge |B\cap X|$ and $|E(B\cap X, V\setminus(B\cap X))| \ge |B\cap X|\left(\frac{|V|}{2} - |B\cap X|\right)$ we have $|E(B\cap X, V\setminus(B\cap X))| \ge \frac{|B\cap X|\cdot|V|}{8}$. Hence, $|E(A\cap X, B\cap X)| \ge \frac{|B\cap X|\cdot|V|}{16} \ge \frac{|B\cap X|\cdot|V|}{100}$.

It is left to consider case $|A\cap X| < \frac{|V|}{4}$. Note that $|E(A\cap X, B\cap X)| = |E(A\cap X, V\setminus(A\cap X))| - |E(A\cap X, Y)|$. As $\frac{|V|}{4} > |A\cap X|$ we have $|E(A\cap X, V\setminus(A\cap X))| \ge |A\cap X|\left(\frac{|V|}{2} - |A\cap X|\right) \ge \frac{|V|}{8}\cdot\frac{|V|}{4} \ge \frac{|V|^2}{32}$. We know that $|E(A\cap X, Y)| \le |E(X,Y)| \le \frac{|V|^2}{100}$, hence $|E(A\cap X, B\cap X) \ge \frac{|V|^2}{32} - \frac{|V|^2}{100} > \frac{|V|^2}{50} \ge \frac{|A\cap X|\cdot|B\cap X|}{100}$. ◀

▶ **Definition 15.** A partition of $V = V_1 \sqcup V_2$ is called a $k$-cut of $G$ if $|E(V_1,V_2)| \le k$.

The following lemma limits number of $k$-cuts in a disjoint union of highly connected graphs.

▶ **Lemma 16.** *If $G = (V,E)$ is a union of $p$ disjoint highly connected components and $p \le k$ then the number of $k$-cuts in $G$ is bounded by $2^{O\left(\sqrt{pk}\right)}$.*

**Proof.** Let $G$ be a disjoint union of highly connected components $C_1,\ldots,C_p$. For each $C_i$ we consider sets $X_i, Y_i$ where $E(X_i,Y_i)$ is a minimum cut of $C_i$ and $C_i = X_i \sqcup Y_i$. We construct a new partition $C'_1,\ldots,C'_q$ of $V(G)$. The new partition is obtained from partition $C_1 \sqcup \ldots \sqcup C_p$ in the following way: if $|E(X_i,Y_i)| < |C_i^2|/100$ then we split $C_i$ into two sets $X_i, Y_i$ otherwise we take $C_i$ without splitting. Note that $p \le q \le 2p$ as we either split $C_i$ into to parts or leave it as is.

We bound number of $k$-cuts of graph $G$ in two steps. In first step we bound number of cuts $V_1, V_2$ such that $|V_1 \cap C'_i| = x_i$ and $|V_2 \cap C'_i| = y_i$ where $x_i, y_i$ are some fixed integers. In second step we bound number of tuples $(x_1,\ldots,x_q,y_1,\ldots,y_q)$ for which there is at least one $k$-cut $V_1,V_2$ satisfying conditions $|V_1\cap C'_i| = x_i$, $|V_2 \cap C'_i| = y_i$.

If $x_i, y_i$ are fixed and $x_i + y_i = |C'_i|$ the number of partitions of $C'_i$ is equal to $\binom{x_i+y_i}{x_i}$. Note that by Lemma 11 we have $\binom{x_i+y_i}{x_i} \le 2^{\sqrt{x_i y_i}}$. Observe that there are at least $\frac{x_i y_i}{100}$ edges between $V_1\cap C'_i$ and $V_2 \cap C'_i$ by Lemma 14. So if $V_1 \sqcup V_2$ is partition of $V$ then $\sum\limits_{i=1}^{q} x_i y_i \le 100k$.

Applying Cauchy–Schwarz inequality we infer that $\sum_{i=1}^{q} \sqrt{x_i y_i} \le \sqrt{q} \cdot \sqrt{\sum_{i=1}^{q} x_i y_i} \le \sqrt{200pk}$. Therefore, the number of considered cuts is at most $\prod_{i=1}^{q} \binom{x_i+y_i}{x_i} \le 2^{2\sum_{i=1}^{q}\sqrt{x_i y_i}} \le 2^{\sqrt{800pk}}$.

Now we show bound for a second step i.e. number of possible tuples $(x_1, \ldots, x_q, y_1, \ldots, y_q)$ generating at least one $k$-cut. Note that $\min\{x_i, y_i\} \le \sqrt{x_i y_i}$. Hence, $\sum_{i=1}^{q} \min(x_i, y_i) \le \sqrt{100qk}$. Tuple $(x_1, \ldots, x_q, y_1, \ldots, y_q)$ can be generated in the following way: at first we choose which value is smaller $x_i$ or $y_i$. Then we express $\sqrt{\lfloor 100qk \rfloor}$ as a sum of $q+1$ non-negative numbers: $\min\{x_i, y_i\}$ for $1 \le i \le q$ and the rest $\sqrt{\lfloor 100qk \rfloor} - \sum_{i=1}^{q} \min(x_i, y_i)$.

The number of choices in the first step of generation is equal to $2^q \le 2^{\sqrt{2qk}}$, and number of ways to express $\sqrt{100qk}$ as a sum of $q+1$ number is at most $\binom{\sqrt{100qk}+q+1}{q} \le 2^{\sqrt{100qk}+q+1} \le 2^{\sqrt{100qk}+\sqrt{2qk}+1}$. Therefore, the total number of partitions is bounded by $2^{c\sqrt{pk}}$ for some constant $c$. ◀

The last ingredient for our algorithm is the following lemma proved by Fomin et al.[5]

▶ **Lemma 17.** *[5] All cuts $(V_1, V_2)$ such that $|E(V_1, V_2)| \le k$ of a graph $G$ can be enumerated with polynomial time delay.*

Now we are ready to present a final theorem.

▶ **Theorem 18.** *There is a $O^*(2^{O(\sqrt{pk})})$ time algorithm for $p$-*Highly Connected Deletion *problem.*

**Proof.** First of all we solve the problem in case of connected graph. Denote by $\mathcal{N}$ set of all $k$-cuts in graph $G$. All elements of set $\mathcal{N}$ can be enumerated with a polynomial time delay. If $G$ is a union of $p$ clusters plus some edges then the size of $\mathcal{N}$ is bounded by $2^{c\sqrt{pk}}$ by Lemma 16 (as additional edges only decrease number of $k$-cuts). Thus, we enumerate $\mathcal{N}$ in time $O^*(2^{O(\sqrt{pk})})$. If we exceed the bound $2^{c\sqrt{pk}}$ given by Lemma 16 we know that we can terminate our algorithm and return answer `NO`. So we may assume that we enumerate the whole $\mathcal{N}$ and it contains at most $2^{c\sqrt{pk}}$ elements.

We construct a directed graph $D$, whose vertices are elements of a set $\mathcal{N} \times \{0, 1, \ldots, p\} \times \{0, 1, \ldots, k\}$, note that $|V(D)| = 2^{O(\sqrt{pk})}$. We add arcs going from $((V_1, V_2), j, l)$ to $((V_1', V_2'), j+1, l')$, where $V_1 \subset V_1'$, $G[V_1' \setminus V_1]$ is highly connected graph, $j \in \{0, 1, \ldots, p-1\}$, and $l' = l + |E(V_1, V_1' \setminus V_1)|$. The arcs can be constructed in $2^{O(\sqrt{pk})}$ time. We claim that the answer for an instance $(G, p, k)$ is equivalent to existence of path from a vertex $((V, \varnothing), 0, 0)$ to a vertex $((\varnothing, V), p', k')$ for some $p' \le p, k' \le k$.

In one direction, if there is a path from $((\varnothing, V), 0, 0)$ to $((V, \varnothing), p', k')$ for some $k' \le k$ and $p' \le p$, then the consecutive sets $V_1' \setminus V_1$ along the path form highly connected components. Moreover, number of deleted edges from $G$ is equal to last coordinate which is smaller than $k$.

Let us prove the opposite direction. Let assume that we can delete at most $k$ edges and get a graph with highly connected components $C_1, \ldots, C_p$. Let us denote $T_i = \cup_{j<i} V(C_i)$, $l_{i+1} = l_i + |E(T_{i+1} \setminus T_i, T_i)|$ then the vertices $((T_i, V \setminus T_i), i-1, l_i)$ constitute desired path in graph $D$.

Reachability in a graph can be tested in a linear time with respect to the number of vertices and arcs. To concude the algorithm we simply test the reachability in the graph $D$.

It is left co consider a case when $G$ is not connected. Let assume that $G$ consist of $q$ connected components $C_1, \dots, C_q$ then for each connected component $C_i$ we find all $p' \leq p$ and $k' \leq k$ such that $(C_i, p', k')$ is YES-instance. After this we construct auxiliary directed graph $Q$ with a set of vertices $\{0, \dots, q\} \times \{0, \dots, p\} \times \{0, \dots, k\}$. We add arcs going from $(i, a, b)$ to $(i + 1, a + p', b + k')$ if $(C_i, p', k')$ is a YES-instance. Using similar arguments as before it could be shown that reachability of vertex $(q, p', k')$ from vertex $(0, 0, 0)$ is equivalent to possibility delete $k'$ edges and get $p'$ highly connected components. ◀

# 3 Algorithms for finding a subgraph

## 3.1 Seeded Highly Connected Edge Deletion

> SEEDED HIGHLY CONNECTED EDGE DELETION
> **Instance:** Graph $G = (V, E)$, subset $S \subseteq V$ and integer numbers $a$ and $k$.
> **Task:** Is there a subset of edges $E' \subseteq E$ of size at most $k$ such that $G - E'$ contains only isolated vertices and one highly connected component $C$ with $S \subseteq V(C)$ and $|V(C)| = |S| + a$.

Hüffner et al. [12] constructed an algorithm with running time $O(16^{k^{0.75}} + k^2nm)$ for SEEDED HIGHLY CONNECTED EDGE DELETION problem. We improve the result to $O^*\left(2^{O(\sqrt{k}\log k)}\right)$ time algorithm.

▶ **Theorem 19.** *There is $O^*(2^{O(\sqrt{k}\log k)})$ time algorithm for* SEEDED HIGHLY CONNECTED EDGE DELETION *problem.*

## 3.2 Isolated Highly Connected Subgraph

> ISOLATED HIGHLY CONNECTED SUBGRAPH
> **Instance:** Graph $G = (V, E)$, integer $k$, integer $s$.
> **Task:** Is there a set of vertices $S$ such that $|S| = s$, $G[S]$ is highly connected graph and $|E(S, V \setminus S)| \leq k$.

Hüffner et al. [12] proposed $O^*(4^k)$ algorithm for ISOLATED HIGHLY CONNECTED SUBGRAPH problem, in this work we construct subexponential algorithm for the same problem with running time $O^*(k^{O(k^{2/3})})$.

In order to solve ISOLATED HIGHLY CONNECTED SUBGRAPH problem Hüffner et al. in [12] constructed algorithm for a more general problem:

> $f$-ISOLATED HIGHLY CONNECTED SUBGRAPH
> **Instance:** Graph $G = (V, E)$, integer $k$, integer $s$, function $f : V \to \mathbb{N}$.
> **Task:** Is there a set of vertices $S$ such that $|S| = s$, $G[S]$ is highly connected and $|E(S, V \setminus S)| + \sum_{v \in S} f(v) \leq k$.

Our algorithm uses reduction rules proposed in [12]. Here, we state the reduction rules without proof, as the proofs can be found in [12].

▶ **Rule 20.** If $G$ contains connected component $C$ of size smaller than $s$ then delete $C$ i.e. solve instance $(G \setminus C, f, k)$.

▶ **Rule 21.** Let $G$ contains connected component $C = (V', E')$ with minimal cut bigger than $k$. If $C$ is highly connected graph, $|V'| = s$ and $\sum_{s \in V'} f(s) \leq k$ then output a trivial

YES-instance otherwise remove $C$, i.e. consider instance $(G \setminus C, f, k)$ of $f$-Isolated Highly Connected Subgraph problem.

▶ **Rule 22.** Let $G$ contains connected component $C$ with minimal cut $(A, B)$ of size at most $\frac{s}{2}$. We define function $f'$ in the following way: for each vertex $v \in A$ $f'(v) := f(v) + |N(v) \cap B|$ and for each $v \in B$ we let $f'(v) := f(v) + |N(v) \cap A|$. Replace original instance with an instance $(G \setminus\ E(A, B), f', k)$.

▶ **Lemma 23.** *Rules 20, 21, 22 can be exhaustively applied in time $O((sn + k)m)$. If rules 20, 21, 22 are not applicable then $k > \frac{s}{2}$.*

We also use following Fomin and Villanger's result.

▶ **Proposition 24.** [6] For each vertex $v$ in graph $G$ and integers $b, f \geq 0$ number of connected induced subgraphs $B \subseteq V(G)$ satisfying the following properties $v \in B$, $|B| = b + 1$, $|N(B)| = f$; is at most $\binom{b+f}{b}$. Moreover, all these sets can be enumerated in time $O\left(\binom{b+f}{b}(n + m)b(b + f)\right)$.

Now we have all ingredients for out algorithm.

▶ **Theorem 25.** *$f$-Isolated Highly Connected Subgraph can be solved in time $2^{O(k^{2/3} \log k)}$.*

**Proof.** First of all we exhaustively apply reduction rules 20, 21, 22. From Lemma 23 follows that we may assume $2k > s$. We consider two cases either $k^{2/3} < s$ or $k^{2/3} \geq s$.

**Case 1:** $s \leq k^{2/3}$. Enumerate all induced connected subgraphs $G' = (V', E')$ such that $|V'| = s$ and $N(V') \leq k$. If desired $S$ exists than it is among enumerated sets. From Proposition 24 follows that number of such sets is at most $nkO^*(\binom{s+k}{s})$. As $s < 2k$ and $s < k^{2/3}$ we have $nkO^*(\binom{s+k}{s}) \leq O^*((s + k)^s) \leq O^*(2^{k^{2/3} \log k})$. Hence, in time $O^*(2^{k^{2/3} \log k})$ we can enumerate all potential candidates $S'$. For each candidate we check in polynomial time whether $G[S']$ is highly connected and $|E(S', V \setminus S')| + \sum\limits_{v \in S'} f(v) \leq k$.

**Case 2:** $k^{\frac{2}{3}} < s$. Let set $S$ be a solution. Define edge set $E' = E(S, V \setminus S)$. Consider function $d : S \to \mathbb{N}$ where $d(v) = |N(v) \cap (V \setminus S)|$. As $\sum\limits_{v \in S} d(v) = |E(S, V \setminus S)| \leq k$ then there is a vertex $v \in S$ such that $d(v) \leq \frac{k}{s} < k^{\frac{1}{3}}$. Note that for such $v$ we have $|N(v)| = |N(v) \cap S| + |N(v) \setminus S| \leq s + k^{\frac{1}{3}}$. We branch on possible values of such vertex and a set of its neighbors that do not belong to $S$. In order to do this we have to consider at most $n \sum\limits_{i \leq k^{1/3}} \binom{s+k^{1/3}}{i} \leq nk^{1/3} 2^{2\sqrt{(s+k^{1/3}-i)i}} \leq nk^{1/3} 2^{2\sqrt{3k^{4/3}}} = n2^{O(k^{2/3})}$ cases. Knowing vertex $v \in S$ and $N(v) \setminus S$ we find $N(v) \cap S$. So we already identified at least $\frac{s}{2} + 1$ vertices from $S$, let denote this set by $W$. Now we start branching procedure that in right branch extend set $W$ into a solution set $S$. Branching procedure takes as an input tuple $(G, k, s', W, B)$ where $W$ is a set of vertices determined to be in solution $S$, $B$ is a set of vertices determined to be not in solution, $k$ number of allowed edge deletions, $s' = s - |W|$ number of vertices that is left to add. The procedure pick a vertex $w \notin W \cup B$ and consider two cases either $w \in S, w \notin B$ or $w \notin S, w \in B$. The first call of the procedure is performed on tuple $(G, k - |E(W, N(v) \setminus W)|, s - |W|, W, \varnothing)$.

Consider arbitrary vertex $x \in V \setminus (W \cup B)$. If $x \in S$ then $|N(x) \cap S| \geq \frac{s}{2}$. Hence, $|N(x) \cap W| \geq |N(x) \cap S| - |S \setminus W| \geq \frac{s}{2} - (s - |W|) = |W| - \frac{s}{2}$. So any vertex $x$ such that

$|N(x) \cap W| < |W| - \frac{s}{2}$ cannot belong to solution $S$ and we safely put $x$ to $B$. Otherwise, we run our procedure on tuples $(G, k - |N(x) \cap B|, s' - 1, W \cup x, B)$ and $(G, k - |N(x) \cap W|, s', W, B \cup x)$. Note that we stop computation in a branch if $k' \leq 0$ or $s' = 0$. It is easy to see that the algorithm is correct.

It is left to determine the running time of the algorithm. Note that procedure contains two parameters $k$ and $s'$. In one branch we decrease value of $s'$ by one in the other branch we decrease value of $k$ by $E(x, W)$. Note that in first branch we not only decrease value of $s'$ but we also increase a lower bound on $|N(x) \cap W|$ by 1 as $|N(x) \cap W| \geq |W| - \frac{s}{2}$.

Let us consider a path $(x_1, x_2, \dots x_l)$ from root to leaf in our branching tree. To each node we assign a vertex $x_i$ on which we are branching at this node. For each such path we construct unique sequence $a_1, a_2, \dots, a_m$ and a number $b$. We put $b$ equal to the number of vertices from set $\{x_1, x_2, \dots, x_l\}$ that was assigned to solution $S$. And $a_i - 1$ is a number of vertices that was assigned to $W$ in a sequence $x_1, x_2, \dots x_j$ where $x_j$ is an $i-$th vertex assigned to $B$ in this sequence. Note that $|N(x_j) \cap W| \geq a_i$, so $\sum_i a_i \leq k$. Note that for any path from root to leaf we can construct a corresponding sequence $a_i$ and number $b$. Moreover, any sequence $a_1, a_2, \dots a_m$ and number $b$ correspond to at most one path from root to node.

▶ **Proposition 26.** Given number $b$ and non-decreasing sequence $a_1, a_2, \dots, a_m$ we can uniquely determine a corresponding path in a branching tree.

**Proof.** For a notation convenience we let $a_0 = 1$. For $1 \leq i \leq m$ we perform the following operation: we make $a_i - a_{i-1}$ steps of assigning vertices to a solution set, i.e. to set $W$ and make one step in branch assigning vertex to a set $B$. After $m$ such iterations we perform $b - m$ steps of assigning vertices to solution. As $a_1, a_2, \dots a_m$ is non-decreasing sequence we have constructed a unique path in branching tree. It is easy to see that the original sequence $a_1, \dots, a_m$ and number $b$ correspond to a constructed path. So for each path from root to leaf there is a corresponding sequence and for each sequence with a number there is at most one corresponding path from root to node in a tree. ◀

▶ **Lemma 27.** *The number of tuples $(a_1, \dots, a_m, b)$ where $0 \leq b \leq s$, $1 \leq a_i \leq a_{i+1}$ for $i < m$, and $\sum_i a_i \leq k$ is bounded by $O^*\left(2^{O(\sqrt{k})}\right)$*

**Proof.** For fixed $l$, tuples $(a_1, \dots, a_m)$ such that $\sum_i a_i = l$ are well-known and are called partitions of $l$. Pribitkin [4] gave a simple upper bound $e^{2.57\sqrt{l}}$ on the number of partitions of $l$. Hence, number of tuples $(a_1, \dots, a_m)$ is bounded by $\sum_{i=0}^{k} e^{2.57\sqrt{i}} \leq (k+1)e^{2.57\sqrt{k}}$. Moreover, we know that $0 \leq b \leq s$. It means that the number of tuples $(a_1, \dots, a_m, b)$ is bounded by $(s+1)(k+1)2^{O(\sqrt{k})}$. ◀

From Proposition 26 and Lemma 27 follows that the number of nodes in a branching tree is at most $s2^{O(\sqrt{k})}$. Hence, the running time of the procedure is at most $s2^{O(\sqrt{k})}$.

Now, we compute required time for algorithm in this case(case 2). At first, we branch on a vertex and its neighbors from solution set $S$. We did it by creating at most $O^*\left(2^{O\left(k^{2/3}\right)}\right)$ subcases. In each subcase we run a procedure with running time $O^*\left(2^{O(\sqrt{k})}\right)$. So, the overall runnning time equals to $O^*\left(2^{O(\sqrt{k})}2^{O\left(k^{2/3}\right)}\right) = O^*\left(2^{O\left(k^{2/3}\right)}\right)$.

The worst running time has **Case 1**, so the running time of the whole algorithms is $O^*\left(k^{O\left(k^{2/3}\right)}\right)$. ◀

## References


1 Balabhaskar Balasundaram, Sergiy Butenko, and Illya V Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, 59(1):133–142, 2011.

2 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets möbius: fast subset convolution. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 67–74, 2007. doi:10.1145/1250790.1250801.

3 Gary Chartrand. A graph-theoretic approach to a communications problem. *SIAM Journal on Applied Mathematics*, 14(4):778–781, 1966.

4 Wladimir de Azevedo Pribitkin. Simple upper bounds for partition functions. *The Ramanujan Journal*, 18(1):113–119, 2009. doi:10.1007/s11139-007-9022-z.

5 Fedor V. Fomin, Stefan Kratsch, Marcin Pilipczuk, Michal Pilipczuk, and Yngve 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.

6 Fedor V. Fomin and Yngve Villanger. Treewidth computation and extremal combinatorics. *Combinatorica*, 32(3):289–308, 2012. doi:10.1007/s00493-012-2536-z.

7 Jiong Guo, Iyad A Kanj, Christian Komusiewicz, and Johannes Uhlmann. Editing graphs into disjoint unions of dense clusters. *Algorithmica*, 61(4):949–970, 2011.

8 Erez Hartuv, Armin O Schmitt, Jörg Lange, Sebastian Meier-Ewert, Hans Lehrach, and Ron Shamir. An algorithm for clustering cdna fingerprints. *Genomics*, 66(3):249–256, 2000.

9 Erez Hartuv and Ron Shamir. A clustering algorithm based on graph connectivity. *Inf. Process. Lett.*, 76(4-6):175–181, 2000. doi:10.1016/S0020-0190(00)00142-3.

10 Wayne Hayes, Kai Sun, and Nataša Pržulj. Graphlet-based measures are suitable for biological network comparison. *Bioinformatics*, 29(4):483–491, 2013.

11 Falk Hüffner, Christian Komusiewicz, Adrian Liebtrau, and Rolf Niedermeier. Partitioning biological networks into connected clusters with maximum edge coverage. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 11(3):455–467, 2014. doi:10.1109/TCBB.2013.177.

12 Falk Hüffner, Christian Komusiewicz, and Manuel Sorge. Finding highly connected subgraphs. In *SOFSEM 2015: Theory and Practice of Computer Science - 41st International Conference on Current Trends in Theory and Practice of Computer Science, Pec pod Sněžkou, Czech Republic, January 24-29, 2015. Proceedings*, pages 254–265, 2015. doi:10.1007/978-3-662-46078-8_21.

13 Antje Krause, Jens Stoye, and Martin Vingron. Large scale hierarchical clustering of protein sequences. *BMC bioinformatics*, 6(1):15, 2005.

14 Hannes Moser, Rolf Niedermeier, and Manuel Sorge. Algorithms and experiments for clique relaxations—finding maximum s-plexes. In *International Symposium on Experimental Algorithms*, pages 233–244. Springer, 2009.

15 Brian J Parker, Ida Moltke, Adam Roth, Stefan Washietl, Jiayu Wen, Manolis Kellis, Ronald Breaker, and Jakob Skou Pedersen. New families of human regulatory rna structures identified by comparative analysis of vertebrate genomes. *Genome research*, 21(11):1929–1943, 2011.

16 Jeffrey Pattillo, Alexander Veremyev, Sergiy Butenko, and Vladimir Boginski. On the maximum quasi-clique problem. *Discrete Applied Mathematics*, 161(1):244–257, 2013.

17 Jeffrey Pattillo, Nataly Youssef, and Sergiy Butenko. On clique relaxation models in network analysis. *European Journal of Operational Research*, 226(1):9–18, 2013.

18 Jeffrey Pattillo, Nataly Youssef, and Sergiy Butenko. On clique relaxation models in network analysis. *European Journal of Operational Research*, 226(1):9–18, 2013. doi:10.1016/j.ejor.2012.10.021.

19 Alexander Schäfer. *Exact algorithms for s-club finding and related problems.* PhD thesis, Friedrich-Schiller-University Jena, 2009.

20 Shahram Shahinpour and Sergiy Butenko. Distance-based clique relaxations in networks: s-clique and s-club. In *Models, algorithms, and technologies for network analysis*, pages 149–174. Springer, 2013.

21 Haiyuan Yu, Alberto Paccanaro, Valery Trifonov, and Mark Gerstein. Predicting interactions in protein networks by completing defective cliques. *Bioinformatics*, 22(7):823–829, 2006.