Finding Highly Connected Subgraphs

Falk Hüffner, Christian Komusiewicz, and Manuel Sorge

Institut für Softwaretechnik und Theoretische Informatik, TU Berlin,
{falk.hueffner,christian.komusiewicz,manuel.sorge}@tu-berlin.de

Abstract. A popular way of formalizing clusters in networks are *highly connected* subgraphs, that is, subgraphs of k vertices that have edge connectivity larger than $k/2$ (equivalently, minimum degree larger than $k/2$). We examine the computational complexity of finding highly connected subgraphs. We show that the problem is NP-hard. Thus, we explore possible parameterizations, such as the solution size, number of vertices in the input, the size of a vertex cover in the input, and the number of edges outgoing from the solution (edge isolation). For some parameters, we find strong intractability results; among the parameters yielding tractability, the edge isolation seems to provide the best trade-off between running time bounds and a small value of the parameter in relevant instances.

1 Introduction

A popular method of analyzing complex networks is to identify *clusters* or *communities*, that is, subgraphs that have many interactions within themselves and fewer with the rest of the graph (e. g. [19, 20]). Hartuv and Shamir [11] proposed a prominent clustering algorithm producing *highly connected* clusters, formalized as follows: the *edge connectivity* $\lambda(G)$ of a graph G is the minimum number of edges whose deletion results in a disconnected graph; and a graph G with n vertices is called *highly connected* if $\lambda(G) > n/2$. An equivalent characterization is that a graph is highly connected if each vertex has degree at least $\lfloor n/2 \rfloor + 1$ [4]. Moreover, highly connected graphs have diameter at most two [11].

We study the following problem:

> Highly Connected Subgraph
> Input: An undirected graph $G = (V, E)$ and a nonnegative integer k.
> Question: Is there a vertex set S such that $|S| = k$ and $G[S]$ is highly connected?

In addition to the natural application in analyzing complex networks [20], Highly Connected Subgraph also occurs (with vertex weights) as subproblem in a column generation algorithm for partitioning into highly connected clusters [13].

Since Highly Connected Subgraph is NP-hard (Theorem 1), we explore the "parameter ecology" [9] of the problem. We are looking for fixed-parameter algorithms, that is, we try to find problem parameters p that allow for a running time of $f(p) \cdot |G|^{O(1)}$. The hope is that the function f grows not too fast (although

it has to be superpolynomial unless P = NP), and that the parameter value p can be expected to be relatively small in interesting instances. Clearly, there is a trade-off between these goals.

Results. We list the results going from the hardest parameters to the easiest, corresponding roughly to going from small expected parameter values to large ones. Let n be the number of vertices in G. For the parameter $\ell := n - k$ (the number of vertices to delete to obtain a highly connected subgraph), we obtain a strong hardness result: there is a trivial $n^{O(\ell)}$ time algorithm, but it is unlikely that $n^{o(\ell)}$ time can be achieved (Theorem 1). For the size of the solution k, a fixed-parameter algorithm is unlikely, even if we additionally consider the degeneracy of G as a parameter (Theorem 2). If we take the minimum size of a vertex cover τ for G as parameter, we obtain the first fixed-parameter algorithm: the problem can be solved in $O((2\tau)^\tau \cdot n^{O(1)})$ time (Corollary 1). Considering the number of vertices n, we can clearly solve the problem in $2^n \cdot n^{O(1)}$ time; however, it is unlikely that this can be improved to $2^{o(n)} \cdot n^{O(1)}$, that is, there is no subexponential-time algorithm (Corollary 2). If the parameter is the number γ of edges between $G[S]$ and the remaining vertices, then we can also find a fixed-parameter algorithm with running time $O(4^\gamma n^2)$ (Theorem 4). Finally, if we consider the number α of edges to delete to obtain a highly connected subgraph (plus singleton vertices), we even obtain a subexponential running time (Theorem 7).

Related work. The algorithm by Hartuv and Shamir [11] partitions a graph heuristically into highly connected components; another algorithm tries to explicitly minimize the number of edges that need to be deleted for this [13]. Highly connected graphs can be seen as *clique relaxation* [19], that is, a graph class that has many properties similar to cliques, without being as restrictive. Highly connected graphs are very similar to *0.5-quasi-complete graphs* [18], that is, graphs where every vertex has degree at least $(n-1)/2$. Recently, also the task of finding subgraphs with high *vertex* connectivity has been examined [21].

Preliminaries. We consider only undirected graphs $G = (V, E)$ with $n := |V|$ and $m := |E|$. We use $G - S$ as shorthand for $G[V \setminus S]$. A *cut* (A, B) in a graph $G = (V, E)$ is a vertex bipartition, that is, $A \cap B = \emptyset$ and $A \cup B = V$. The *cut edges* are the edges with one endpoint in A and one in B, and the *size* of a cut is the number of its cut edges. For the definitions of FPT, W[1], parameterized reduction, and problem kernelization refer to [8].

2 Vertex Deletion Parameter

For finding large cliques in a graph, one successful approach is to use fixed-parameter algorithms for the parameter "number of vertices in the graph that are not in the clique" [15, 16]. We show that for Highly Connected Subgraph this parameter does not lead to fixed-parameter tractability.

Theorem 1. Highly Connected Subgraph *is NP-hard and W[2]-hard parameterized by* $\ell := n - k$ *and it cannot be solved within* $n^{o(\ell)}$ *time unless FPT=W[1].*

Proof. We present a reduction from Hitting Set:

> Input: A set family $\mathcal{S} = \{S_1, \ldots, S_m\}$ over a ground set $U = \{1, \ldots, n\}$ and an integer k.
> Question: Is there a set $H \subseteq U$ of size k such that $S_i \cap H \neq \emptyset$ for each $S_i \in \mathcal{S}$?

Given an instance $(\mathcal{S}, k)$ of Hitting Set, construct an instance $(G = (V, E), k')$ of Highly Connected Subgraph as follows (we assume w.l.o.g. that $k < n-1$). Initially, set $V := U \cup V_S$ where $V_S := \{s_i \mid 1 \leq i \leq m\}$, that is, create one vertex for each element and each set of the Hitting Set instance. Next, make each vertex s_i adjacent to all vertices $u \in U \setminus S_i$, that is, to the vertices corresponding to elements *not* in S_i. This will encode the Hitting Set instance. Now, add three cliques V_X, V_Y, and V_Z to G, where V_X has size $k+1$, V_Y has size n, and V_Z has size m. These three cliques will enforce that at least k vertices are deleted, and that some of the deleted vertices are contained in U. The purpose of the edges between U and V_S is to make sure that for each s_i at least one deleted vertex from U is not a neighbor of s_i (and thus it is contained in S_i). To achieve these properties, add the following edges.

First, add all edges between the vertices in U, V_Y, and V_Z, that is, make $U \cup V_Y \cup V_Z$ a clique. Furthermore, add all possible edges between V_X and V_S and make each vertex in V_X adjacent to exactly $n-k+1$ vertices of V_Y. When adding these edges, ensure that every vertex in V_Y has at least one neighbor in V_X. Furthermore, add all edges between V_S and V_Z and make each vertex $s_i \in V_S$ adjacent to $|S_i| - 1$ vertices in V_Y.

To complete the construction, set $k' := |V| - k$. Note that this implies $\ell = k$. Before we show the soundness of the reduction, observe the following about the degrees of the vertices in G:

- Each vertex in $U \cup V_Y \cup V_Z$ has degree at least $m + 2n - 1$.
- Each vertex in V_X has degree exactly $m + n + 1$ in G.
- Each vertex in V_S has degree exactly $m + n + k$ in G.

It remains to show that

> $(\mathcal{S}, k)$ is a yes-instance of Hitting Set $\Leftrightarrow$ (G, k') is a yes-instance of Highly Connected Subgraph.

"$\Rightarrow$": Let $H \subseteq U$ be a size-k hitting set. We show that $G - H$ is highly connected. Note that $|V| - k = 2(m+n)+1$ and thus $G - H$ is highly connected if all its vertices have degree at least $m + n + 1$. Since $k < n$, all vertices in $U \cup V_Y \cup V_Z$ have degree at least $m + n + 1$. Furthermore, each vertex in V_X has degree at least $m + n + 1$ in $G - H$ since there are no edges between U and V_X. Finally, every vertex $s_i \in V_S$ has degree at least $m + n + 1$: Since H is a

hitting set, there is one vertex in H that is not adjacent to s_i. Thus, the degree of s_i is at least $m + n + k - (k - 1) = m + n + 1$.

"$\Leftarrow$": Let H be a vertex set of size at most k such that $G - H$ is highly connected. First, note that $|V_X| = k + 1$ which implies that there is at least one vertex $v \in V_X$ that is not deleted. Since $G - H$ is highly connected this implies that v has more than $\lfloor(|V| - |H|)/2\rfloor$ neighbors in $G - H$. Since v has exactly $m + n + 1$ neighbors in G this implies that $|V| - |H| \leq 2(m + n) + 1$ and thus $|H| = k$. Note that this also implies that $N(v) \cap H = \emptyset$ which also implies $V_X \cap H = \emptyset$. Now, this means that all vertices in H are nonadjacent to *all* vertices in V_X, and thus $H \subseteq U \cup V_Z$. We show that $H \cap U$ is a hitting set for $\mathcal{S}$. Assume that this is not the case; then there is one vertex in $s_i \in V_S$ such that all deleted vertices are adjacent to s_i since s_i is adjacent to all vertices in $(U \cup V_Z) \setminus S_i$. This vertex has degree $m + n$ in $G - H$. This contradicts the fact that $G - H$ is highly connected.

Since the above reduction is a parameterized polynomial-time reduction, this shows that Highly Connected Subgraph is NP-hard and W[2]-hard with respect to ℓ. Moreover, the reduction is a *linear* parameterized reduction, that is, the new parameter is bounded in a linear function of the old parameter. Hence, an $n^{o(\ell)}$ algorithm for Highly Connected Subgraph implies an $n^{o(k)}$ algorithm for Hitting Set which implies FPT = W[1] [5]. □

3 Solution Size and Degeneracy

A graph has degeneracy d if every subgraph contains at least one vertex that has degree at most d. In many graphs from real-world applications, the degeneracy of a graph is very small compared to the network size. For yes-instances, the degeneracy of the input graph has to be at least $\lfloor k/2\rfloor + 1$. Therefore, Highly Connected Subgraph is polynomial-time solvable if the input graph has constant degeneracy. A straightforward algorithm decides the problem in $n^{2d} \cdot \text{poly}(n)$ time. This can be improved to the following running time.

Proposition 1. Highly Connected Subgraph *can be solved in* $2^d \cdot n^{d+O(1)}$ *time where* d *is the degeneracy of* G.

Proof. The algorithm is as follows. Pick a vertex v of degree at most d. Such a vertex exists since G has degeneracy d. Now check whether there is a $k \leq 2d$-vertex highly connected graph $G[S]$. Branch into all possible cases for $N(v) \cap S$. Since v has degree at most d, there are $O(2^d)$ many of these possibilities. Now find the at most d vertices of $S \setminus N(v)$ by brute-force. This means to consider $O(n^d)$ many possibilities. For each of these possibilities, check in polynomial time whether $G[S]$ is highly connected. If this is the case, then accept. If this is not the case for all $O(2^d \cdot n^d)$ possibilities, then remove v from G and restart the algorithm. □

Unfortunately, if we regard the degeneracy as a parameter instead of a constant, we obtain hardness, even if additionally the solution size k is a parameter.

Theorem 2. Highly Connected Subgraph *parameterized by the combined parameter* (d, k)*, where* d *is the degeneracy of* G*, is W[1]-hard.*

Proof. We reduce from the classic W[1]-hard Clique problem [7].

> Clique:
> Input: An undirected graph $G = (V = \{v_1, \ldots, v_n\}, E)$, a nonnegative integer k.
> Question: Does G contain a clique with at least k vertices?

Given an instance (G, k) of Clique, produce an instance (G', k') of Highly Connected Subgraph as follows. First, subdivide every edge $e = \{u, w\}$ in G, that is, remove e, insert a new *edge vertex* v^e and make v^e adjacent to both u and w. Call the set of edge vertices V_E. Next, add a clique with vertex set $X = \{x_1, \ldots, x_\ell\}$ where $\ell = \binom{k}{2} + 3k$. Then, choose $\binom{k}{2} + 2k - 1$ arbitrary vertices in X, denote this set by X_1 and make each v^e adjacent to each vertex in X_1. Furthermore, let $X_2 := X \setminus X_1$ and make each $v_i \in V$ adjacent to the $k+1$ vertices in X_2 and to $\binom{k}{2} + 1$ further arbitrary vertices of X_1, finishing the construction of G'. Note that each $v_i \in V$ is adjacent to $\binom{k}{2} + k + 2$ vertices in X. The construction of the instance is completed by setting the size of the desired highly connected subgraph to $k' := 2 \cdot \binom{k}{2} + 4k$.

Note that the graph G' is $\binom{k}{2} + 3k$-degenerate, which can be seen by the following order of vertex removals: First, remove the vertices from V_E; these have degree $\binom{k}{2} + 2k + 1$ in the graph. Then, remove the vertices from V, these have degree $\binom{k}{2} + k + 2$ in $G' - V_E$. Finally, the remaining set X has size $\binom{k}{2} + 3k$, thus each vertex in $G - (V_E \cup V)$ has degree $\binom{k}{2} + 3k - 1$. The overall degeneracy of G' follows.

We now show the equivalence of the constructed instances, that is,

> (G, k) is a yes-instance of Clique $\Leftrightarrow$ (G', k') is a yes-instance of Highly Connected Subgraph.

"$\Rightarrow$": Let $K \subseteq V$ be a clique of order k in G. Then, the set $K' := K \cup \{v^e \mid e \subseteq K\} \cup X$ induces a highly connected subgraph of order k' in G', which can be seen as follows. First, K' has size $k + \binom{k}{2} + \binom{k}{2} + 3k = k'$. Second, each vertex in $G'[K']$ has degree at least $k'/2 + 1 = \binom{k}{2} + 2k + 1$: The vertices in X have at least $|X| - 1 = \binom{k}{2} + 3k - 1 > k'/2 + 1$ neighbors in $G'[K']$ since X is a clique. The vertices in K have exactly $\binom{k}{2} + k + 2$ neighbors in X plus $k - 1$ neighbors in $K' \setminus (K \cup X)$ since they are adjacent to the $k - 1$ vertices corresponding to the edges incident with them in $G[K]$. Finally, each remaining vertex corresponds to an edge in $G[K]$ and thus it has two neighbors in K. Since it has $\binom{k}{2} + 2k - 1$ neighbors in X it has exactly $k'/2 + 1$ neighbors in $G'[K']$.

"$\Leftarrow$": Let K' be a vertex set of size k' such that $G'[K']$ is highly connected and let $y_1 = |V \cap K'|$ and $y_2 = |V_E \cap K'|$. Note that by the size of X, K' contains at least $\binom{k}{2} + k$ vertices from $V' \setminus X = V \cup V_E$, that is, $y_1 + y_2 \geq \binom{k}{2} + k$. Furthermore, note that if it contains a vertex v^e from V_E, then it must contain all neighbors of v^e in G', since v^e has degree exactly $k'/2 + 1$ in G'. Note that by the

above $K^* = K' \cap (V_E \cup V)$ directly corresponds to a subgraph of G: each vertex in V_E corresponds to an edge in G and both endpoints of this edge are in K'. Furthermore, since each vertex in $K' \cap V$ has at least $k-1$ neighbors in $K' \setminus X$, the corresponding graph has degree at least $k-1$. Finally, since K^* corresponds to a graph, we have $y_2 \leq \binom{y_1}{2}$. Since $y_1 + y_2 \geq \binom{k}{2} + k$ this implies $y_1 \geq k$. In the remainder of the proof we show $y_1 = k$ which directly implies that $K' \cap V$ is a clique in G since then K^* corresponds to a graph with k vertices and minimum degree $k-1$.

Assume towards a contradiction $y_1 \geq k+1$. We prove that we may also assume that $X \subseteq K'$. If this is not the case, then any vertex from $X \setminus K'$ belongs to X_2: we have that $K' \cap V_E \neq \emptyset$, because otherwise each v_i would have at most $\binom{k}{2} + 3k$ neighbors in K', and all vertices of X_1 are in K' because all neighbors of $K' \cap V_E$ are in K'. Hence, there is a vertex $x \in X_2 \setminus K'$. By construction, this vertex is a neighbor of all vertices in $K' \cap X$ and of all vertices in $K' \cap V$. We can thus pick an arbitrary vertex $v^e \in K' \cap V_E$, remove v^e from K' and add x to K'. In the graph that is induced by the modified K', the degree of every vertex except x has increased or remains the same: the removed vertex v_e has only neighbors in X and in V and x is in G' adjacent to all vertices in X and to all vertices in V. Furthermore, x has also degree at least $k'/2 + 1$ since it is adjacent to the $\binom{k}{2} + 2k - 1$ vertices in X_1 and to the at least $y_1 \geq k+1$ vertices in $V \cap K'$.

Note that this replacement can be performed without decreasing the minimum degree in $G'[K']$ below $k'/2 + 1$ until $X \subseteq K'$. Hence, if $y_1 \geq k+1$, then we can also assume that $X \subseteq K'$. But then $y_2 < \binom{k}{2}$. This implies that there is at least one $v_i \in K' \cap V$ that has less than $k-1$ neighbors in $K' \cap V_E$. Since v_i has only $\binom{k}{2} + k + 2$ neighbors in X, it has less than $k'/2 + 1$ neighbors in $G'[K']$. This contradicts the assumption that $G'[K']$ is highly connected.

Hence, if $G'[K']$ is highly connected, then $y_1 = k$. By the discussion above, K^* corresponds to a subgraph of G with k vertices and $\binom{k}{2}$ edges, that is, a clique of order k. □

4 Parameterization by Vertex Cover

We next consider the parameter "minimum size τ of a vertex cover" of G. This parameter is interesting because it can be smaller than the number of vertices of G. We show that Highly Connected Subgraph is solvable in $O^*((2\tau)^\tau)$ time. The algorithm first computes a vertex cover, then determines, via branching, its intersection with some fixed solution, and then adds suitable vertices of the independent set formed by the complement of the vertex cover. Note that the vertices of the independent set can be classified into 2^τ equivalence classes according to their neighbors in the vertex cover. Inspection shows that in each of these classes, only τ vertices can be in a highly connected graph. Hence, the instance can be reduced to $2^\tau\tau + \tau$ vertices. This already shows that Highly Connected Subgraph is fixed-parameter tractable with respect to τ. To obtain an efficient algorithm, we reduce each subproblem to an instance of Set Multicover.

Set Multicover
Input: A universe of elements U with covering demands $d : U \to \mathbb{N}$, a family $\mathcal{F}$ of subsets of the universe with multiplicity values $m : \mathcal{F} \to \mathbb{N}$, and a nonnegative integer p.
Question: Is there a multiset of subsets of size at most p such that each $u \in U$ occurs in at least $d(u)$ of them and no subset $F \in \mathcal{F}$ occurs more than $m(F)$ times?

Lemma 1. *A given instance of* Highly Connected Subgraph *with a vertex cover of size τ can be solved using the answers to at most 2^τ instances of* Set Multicover *with $|U| \leq \tau$ and $2\max_{u\in U} d(v) \leq \tau$. Furthermore, all these instances can be computed in $O(2^\tau(n + \tau n))$ time.*

Proof. Fix some highly connected subgraph $G[S]$ of order k if it exists. First compute a minimum vertex cover C for G in $O(1.274^\tau + \tau n)$ time [6]. Enumerate all 2^τ possibilities for $C' := C \cap S$. Clearly, one branch contains the correct C'. In each branch, delete the vertices from $C \setminus C'$. Then remove vertices from the independent set $V \setminus C'$ that have $k/2$ or less neighbors in C', since they cannot be part of S. Let V' be the thus reduced vertex set. It remains to find $k' := k - |C'|$ vertices in $V' \setminus C'$ such that each vertex v in C' has more than $d(v) := k/2 - |N(v) \cap C'|$ neighbors among these k' vertices.

This is an instance of Set Multicover. In our case, the universe is C', the covering demands are d as defined above, the family is $\mathcal{F} = \{N(v) \cap C' \mid v \in V' \setminus C'\}$, the multiplicity values count how many vertices in $V' \setminus C'$ have this neighborhood in C', and $p = k'$. If the solution to Set Multicover has less than k' sets, then we can add arbitrary further vertices from $V' \setminus C'$ to make the vertex subset large enough (if $|V' \setminus C'| < k'$, then we can reject this branch). □

Set Multicover with multiplicity constraints can be solved in $O((b+1)^{|U|}|\mathcal{F}|)$ time [12], where $b := \max_{u\in U} d(u)$. Note that $|C'| > k/2$, since the vertices outside of C' form an independent set and we cannot choose $k/2$ or more of them. Thus, $b < |C'| \leq \tau$. The size of $\mathcal{F}$ is at most n. Together with the enumeration of the instances, we obtain the following corollary.

Corollary 1. Highly Connected Subgraph *can be solved in $O((2\tau)^\tau \cdot \tau n)$ time.*

5 Number of Vertices

A trivial algorithm for Highly Connected Subgraph is to enumerate all vertex subsets S of size k and to check for each subset whether it is highly connected. This algorithm has running time $O(2^n \cdot m)$. We now show by a reduction from Clique that a running time improvement to $2^{o(n)} \cdot n^{O(1)}$ is unlikely. The idea of the reduction is to add to a Clique instance some new graph that is so large, that at least some vertex of it must be in a highly connected graph of order k. Moreover, the construction ensures that this means that all of this graph must be in this highly connected graph. The remaining vertices must form a clique in order to have sufficiently high degree. The following combinatorial observations are used in our construction.

Lemma 2. *For any $\ell \in \mathbb{N}$, the edges of the complete graph K_{2^ℓ} can be partitioned into $2^\ell - 1$ perfect matchings. Moreover, there is such a partition that includes two perfect matchings that together contain a spanning tree of K_{2^ℓ}, and the partition can be computed in polynomial time in 2^ℓ.*

Proof. The proof is by induction on ℓ. Clearly, the edge of K_{2^1} can be partitioned into a perfect matching in polynomial time. Now assume that the statement holds for all $\ell' < \ell$. Partition the edges of K_{2^ℓ} into three sets in the following manner. Two sets A, B of the partition are induced by two vertex-disjoint subgraphs K_A, K_B of K_{2^ℓ} each isomorphic to $K_{2^{\ell-1}}$. The third set C contains the edges with exactly one vertex from both of the subgraphs. By induction, we can compute edge-partitions into perfect matchings of K_A, K_B; call the partitions $\mathcal{P}_A, \mathcal{P}_B$. Now join $\mathcal{P}_A, \mathcal{P}_B$ into a partition $\mathcal{P}_{A\cup B}$ of $A \cup B$ by iteratively taking the union of an arbitrary part from $\mathcal{P}_A$ and an arbitrary part from $\mathcal{P}_B$ and deleting the parts from $\mathcal{P}_A$ and $\mathcal{P}_B$ respectively. Note that the obtained partition $\mathcal{P}_{A\cup B}$ contains $2^{\ell-1} - 1$ parts. Next, compute a partition $\mathcal{P}_C$ of C as follows. Denote $V(K_A) =: \{v_0, \ldots, v_{2^{\ell-1}-1}\}$, $V(K_B) =: \{u_0, \ldots, u_{2^{\ell-1}-1}\}$. Then $E_j := \{\{v_i, u_{i+j}\} \mid i \in \{0, \ldots, 2^{\ell-1} - 1\}\}$, and $\mathcal{P}_C := \{E_j \mid j \in \{0, \ldots, 2^{\ell-1} - 1\}\}$ where indices are taken modulo $2^{\ell-1}$. Clearly, $\mathcal{P}_C$ can be computed in polynomial time. Note that $\mathcal{P}_C$ is a partition of C into perfect matchings, and $|\mathcal{P}_C| = 2^{\ell-1}$. Finally, take the union $\mathcal{P}_{A\cup B} \cup \mathcal{P}_C$. Note that this is a partition of $E(K_{2^\ell})$ into perfect matchings and $|\mathcal{P}_{A\cup B} \cup \mathcal{P}_C| = 2^\ell - 1$.

Overall, let $2^{c\cdot(\ell-1)}$ be an upper bound on the time needed to compute the edge partitions $\mathcal{P}_A$ and $\mathcal{P}_B$, and let $2^{c'\cdot(\ell-1)}$ be an upper bound on the time needed to compute $\mathcal{P}_C \cup \mathcal{P}_{A\cup B}$ from $\mathcal{P}_A$ and $\mathcal{P}_B$ where $c, c' \geq 2$ are universal constants. Then an overall time bound for the above procedure is $2^{c\cdot(\ell-1)+1} + 2^{c'\cdot(\ell-1)} = 2^{c\cdot\ell-c+1} + 2^{c'\cdot\ell-c'} \leq 2^{\max\{c,c'\}\cdot\ell+2-\min\{c,c'\}} \leq 2^{\max\{c,c'\}\cdot\ell}$, which is polynomial in 2^ℓ.

Let us now slightly modify the step of computing E_j in order to obtain two matchings that span K_{2^ℓ}. Fix an arbitrary matching $M \in \mathcal{P}_{A\cup B}$. Rename the vertices in P_A and P_B in such a way that

$$M = \{\{v_0, v_1\}, \{v_2, v_3\}, \ldots, \{v_{2^{2\ell-1}-2}, v_{2^{2\ell-1}-1}\},$$
$$\{u_1, u_2\}, \{u_3, v_4\}, \ldots, \{u_{2^{2\ell-1}-1}, u_0\}\}.$$

Then $M \cup E_0$ induces a connected graph containing all vertices of K_{2^ℓ}. The renaming of the vertices can be performed in linear time, thus the running time increases only by a constant factor. □

Lemma 3. *For any integer $\ell \in \mathbb{N}$, $\ell \geq 3$, there are two edge-disjoint Hamiltonian cycles in $K_{2\ell}$, computable in polynomial time.*

Proof. We describe a polynomial-time algorithm that computes the two cycles. First, fix an arbitrary permutation of the vertex set which directly defines a Hamiltonian cycle. Remove this cycle from the graph. The resulting degree of each vertex is $2\ell - 1 - 2$. We now use the following known result: If the closure of a graph is a complete graph, then this graph has a Hamiltonian cycle which

can be computed in polynomial time [3]. Here, the closure of a graph of order n is obtained by adding all edges $\{u, v\}$ if $\deg(u) + \deg(v) \geq n$ and iterating as long as possible. Since the sum of two vertex degrees in our graph is at least $2(2\ell-3) \geq 2\ell$, the closure of the resulting graph is $K_{2\ell}$ again. Thus, in polynomial time we can compute another Hamiltonian cycle in the remaining graph which is edge-disjoint from the first cycle. □

With these lemma at hand, we can show that we can build a graph whose construction will actually be the main part of our reduction.

Lemma 4. *For any two integers $a, b \in \mathbb{N}$ such that a is even, $b-3 \geq 8$ is a power of two, and $a-2 \geq 2b$, there is a graph $G = (X \cup W, E)$ on the disjoint vertex sets X and V, such that*
i) $G[X]$ is connected,
ii) $|X| = a-2$, $|W| = a-b+1$,
iii) $N_G(X) \setminus X = W$, and
iv) each vertex in X has degree a, each vertex in W has degree $a-b$.
Moreover, G can be constructed in time polynomial in a.

Proof. Begin with $G = (X \cup W, E)$ where $X \cup W$ is an independent set and where X and W have the prescribed sizes. Next, let $X_1, X_2 \subseteq X$ be arbitrary such that $|X_1| = |X_2| = a-b+1$ and $|X_1 \cap X_2|$ is of minimum size. That is, $|X_1 \cap X_2| = a-2b+4$, and $|X_1 \setminus X_2| = |X_2 \setminus X_1| = b-3$ (note that $a-2 \geq 2b$, hence, $|X_1 \cap X_2| \geq 6$). We add a set of matchings to G that saturate W in order to bring up the degree of the vertices in W to $a-b$. We first add a perfect matching to $G[W]$ (note that $a-b+1$ is even, because a is even and $b-3$ is a power of two). Then we add $a-b-1$ matchings that saturate W to $G[W \cup X]$; these matchings are divided evenly into matchings saturating X_1 and matchings saturating X_2. More formally, denote $W =: \{w_0, \ldots, w_{a-b}\}$, and $X =: \{x_0, \ldots, x_{a-3}\}$ such that $X_1 = \{x_0, \ldots, x_{a-b}\}$, and $X_2 = \{x_{b-3}, \ldots, x_{a-3}\}$. Let $i \in \{0, \ldots, a-b-2\}$. We define

$$M_i := \begin{cases} \{\{w_j, x_{(j+i) \bmod (a-b)}\} \mid j \in \{0, \ldots, a-b\}\}, & \text{if } i \text{ is even, and} \\ \{\{w_j, x_{b-3+((j+i+1) \bmod (a-b))}\} \mid j \in \{0, \ldots, a-b\}\}, & \text{otherwise.} \end{cases}$$

That is, if i is even, then M_i is a perfect matching in $G[W \cup X_1]$ and, otherwise M_i is a perfect matching in $G[W \cup X_2]$. Note that there is an even number of matchings M_i. Furthermore, $M_i, M_{i'}$, $i \neq i'$, are disjoint: this is clear if both i and i' are even, or both are odd. If i is odd and i' is even, then w_j is matched to some v_k with $k \equiv j \mod 2$ in M_i whereas $M_{i'}$ matches w_j to some v_k with $k \not\equiv j \mod 2$, because both $b-3$ and $i+1$ are even, and $a-b$ is odd. Hence, we may add all M_i, $i \in \{0, \ldots, a-b-2\}$, to G. Now each vertex in W has degree $a-b$, as required, and it remains to mend the degrees of vertices in X. Note that each vertex in $X_1 \cap X_2$ has degree $a-b-1$, and each vertex in $X \setminus (X_1 \cap X_2)$ has degree $(a-b-1)/2$. We divide $X_1 \cap X_2$ into two equal-sized parts A, B; note that this is possible, because $|X_1 \cap X_2| = a-2b+4$ is even and $a-2b+4 \geq 6$, since $a-2 \geq 2b$. We make each vertex from A adjacent to each vertex in $X_1 \setminus X_2$, and

each vertex in B adjacent to each vertex in $X_2 \setminus X_1$. Now each vertex in $X_1 \cap X_2$ has degree $a-b-1+b-3 = a-4$. Using Lemma 3 we add four perfect matchings to $G[X_1 \cap X_2]$ (recall that $|X_1 \cap X_2| \geq 6$).

It remains to lift the degree of the vertices in $X \setminus (X_1 \cap X_2)$ which currently have degree $(a-b+1)/2 + (a-2b+4)/2 = a - 3b/2 + 5/2$. Note that $2(b-3)$ is a power of two. Now we use Lemma 2 to add $3b/2 - 5/2 \leq 2(b-3) - 1$ perfect matchings to $G[X \setminus (X_1 \cap X_2)]$, including two perfect matchings that make $G[X \setminus (X_1 \cap X_2)]$ connected. Note that, indeed, $3b/2 - 5/2 \leq 2(b-3) - 1$, because $b - 3 \geq 8$. Hence, the degree of each vertex in $X \setminus (X_1 \cap X_2)$ is now a. Note also that $G[X]$ is connected. □

We call the graph G described in the above lemma an (a, b)-*equalizer* and the vertices in W are its *ports*.

Theorem 3. *There is a polynomial-time many-one reduction from* Clique *to* Highly Connected Subgraph *that is parameter-linear with respect to the number of vertices.*

Proof. Let (G, p) represent an instance of Clique. Without loss of generality, assume that $p - 3 \geq 8$ and $p - 3$ is a power of two. Otherwise, simply add a universal vertex and increase p by one, until $p - 3 \geq 8$ and it is a power of two. Note that this at most doubles p.

Denote $|V(G)| = n$. We construct the instance (G', k) of Highly Connected Subgraph where $k = 4n-1$. Note that the minimum degree in a highly connected graph with k vertices is $2n$. Graph G' is constructed as follows. First, copy G into G'. Then add a disjoint $(2n, p)$-*equalizer*. By Lemma 4, a $(2n, p)$-*equalizer* exists and is computable in polynomial time, because $2n$ is even, $p - 3 \geq 8$ is a power of two, and $2n - 2 \geq 2p$ (the last property holds without loss of generality). Denote the ports of the equalizer by W and its remaining vertices by X. Add an edge between each port and each vertex in $V(G)$; this finishes the construction. The graph G' has less than $5n$ vertices, since the $(2n, p)$-*equalizer* has less than $4n$ vertices. It remains to show equivalence of the instances, that is,

$$(G, p) \text{ is a yes-instance} \Leftrightarrow (G', k = 4n - 1) \text{ is a yes-instance.}$$

"⇒": Let $G[S]$ be a clique of order p in G. Then, $G'[S \cup X \cup W]$ is highly connected: Each vertex in S has is adjacent to $p-1$ vertices in S and to $2n-p+1$ vertices in W. Hence, each vertex in S has $2n$ neighbors in $S \cup X \cup W$, as required. Each port has $2n - p$ neighbors in $X \cup W$ and p neighbors in S. Finally, each vertex in X has $2n$ neighbors in $X \cup W$.

"⇐:" Let $G'[S]$ be a highly connected graph of order k in G'. There are at most n vertices in $V(G) \cap S$, thus there is at least one vertex in $S \cap X$. Since $G'[X]$ is connected and each vertex in X has degree exactly $2n$ (the minimum degree in $G'[S]$), we have $X \subseteq S$. Furthermore, since $\{v \mid X \cap N(v) \neq \emptyset\} \setminus X = W$, also $W \subseteq S$, leaving $4n - 1 - |X| - |W| = p$ vertices in $S \cap V(G)$. Since $N_{G'}(V(G)) \setminus V(G) = W$ and $|W| = 2n - p + 1$, each vertex in $S \cap V(G)$ has at least $p - 1$ neighbors in $S \cap V(G)$. Thus $G[S \cap V(G)]$ is a clique. □

Using Theorem 3, we can connect the parameterized running time for parameter n with the Exponential Time Hypothesis (ETH) [17].

Corollary 2. *If the Exponential Time Hypothesis (ETH) is true, then* Highly Connected Subgraph *does not admit a* $2^{o(n)} \cdot n^{O(1)}$*-time algorithm.*

6 Edge Isolation Parameter

We now present a single-exponential FPT algorithm for the number γ of edges between the desired highly connected subgraph $G[S]$ and the remaining graph. In this case, S is called "γ-isolated". More formally, if $G = (V, E)$ is a graph, we call a set $S \subseteq V$ *γ-isolated* if $(S, V \setminus S)$ is a cut of size at most γ. To our knowledge, Ito et al. [15] where the first to consider a formal notion of isolation in the context of dense subgraph identification. There is the following slight difference between the isolation definitions: we count the total size of the cut $(S, V \setminus S)$, whereas previous definitions count the size of $(S, V \setminus S)$ divided by the size of S [14, 15] or the minimum of the number of outgoing edges per vertex [16]. Our isolation definition leads to the following problem.

Isolated Highly Connected Subgraph
Input: An undirected graph $G = (V, E)$, nonnegative integers k and γ.
Question: Is there a k-vertex γ-isolated highly connected subgraph contained in G?

The notion of isolation is not only motivated from the algorithmic point of view but also from the application. Ideally, communities in a network have fewer connections to the rest of the network [19]. Thus, putting an additional constraint on the number of outgoing edges may yield better communities than merely demanding high edge connectivity.

In the following, it will be useful to consider an augmented version of Isolated Highly Connected Subgraph: we place integer labels on the vertices which imply that these vertices are harder to isolate. We thus additionally equip each instance of Isolated Highly Connected Subgraph with a labeling $f\colon V \to \mathbb{N}$ and we call $V' \subseteq V$ *γ-isolated under f* if there are at most $\gamma - \sum_{v \in V'} f(v)$ edges between V' and $V \setminus V'$ in G. Without loss of generality, assume $k \geq 2$ in the following.

The algorithm first performs three reduction rules. The first simple rule removes connected components that are too small.

Rule 1. *If G contains a connected component C with less than k vertices, then remove C.*

The next rule identifies connected components which are either trivial solutions or cannot contain any other solution since induced proper subgraphs violate the isolation condition.

Rule 2. *If there is a connected component $C = (V', E')$ of G that has minimum cut size at least $\gamma + 1$, then accept if C is highly connected, $|V'| = k$, and V' is γ-isolated under f. Otherwise remove C from G.*

Proof (Correctness of Rule 2). The rule is clearly correct if it accepts. If the rule removes C, then C has a minimum cut of size at least $\gamma + 1$. Thus, for every induced subgraph $C[S]$ of C that does not contain all of its vertices we have that S is not γ-isolated. Hence, no subgraph of C is a solution and we can safely remove C from G. □

Rule 3. *If G has a connected component C with a minimum cut (A, B) of size at most $k/2$, then do the following. For each $v \in A$ redefine $f(v) := f(v)+|N(v)\cap B|$ and for each $v \in B$ redefine $f(v) := f(v) + |N(v) \cap A|$. Then, delete all edges between A and B.*

Proof (Correctness of Rule 3). Any k-vertex subgraph of C with nonempty intersection with both sides of (A, B) is not highly connected as it has a minimum cut of size at most $k/2$. Hence, any highly connected induced subgraph $C[S]$ of C is either contained in $C[A]$ or in $C[B]$. If S is γ-isolated under f in G, then it is also γ-isolated under the modified f in the modified graph (and vice-versa) by the way we have redefined f. □

Exhaustive application of these rules yields instances in which γ and $k/2$ are related.

Lemma 5. *If Rules 1 to 3 are not applicable, then $\gamma > k/2$.*

Proof. Assume the contrary. Each connected component has a minimum cut cutting at least one edge because Rule 1 is not applicable and $k \geq 2$. Further, each connected component has a cut of size at most γ because Rule 2 is not applicable. By assumption $\gamma \leq k/2$ and, hence, each connected component has a cut of size at most $k/2$ which contradicts the inapplicability of Rule 3. □

As shown by the following lemma, the reduction rules can be applied efficiently.

Lemma 6. *Rules 1 to 3 can be exhaustively applied in $O((kn + \gamma)nm)$ time.*

Proof. We first apply Rule 3 exhaustively. To this end, we determine for each connected component of G whether it contains a minimum cut of size at most $k/2$ by fixing an arbitrary vertex v and for each vertex u running $k/2+1$ rounds of the Ford-Fulkerson algorithm to find a flow from v to u. If at some round the flow does not increase, we find a corresponding cut by considering the strongly connected components in the residual graph and apply Rule 3. We repeat the procedure if it was applicable. Finding the connected components in the residual graph, and the rounds of Ford-Fulkerson can each be implemented to run in $O(n+m)$ time. Hence if $C_1, \ldots, C_\ell$ are the connected components of G then one iteration of Rule 3 takes $\sum_{i=1}^{\ell} O(k|C_i|(|C_i| + |E(G[C_i])|)) = O(knm)$-time. The overall number of rounds for applying Rule 3 is n since the number of connected components

increases by one in each round, thus the overall running time for exhaustively applying Rule 3 is $O(kn^2m)$.

Then we apply Rule 2 by computing for each connected component C whether its minimum cuts have size at least γ. If this is true, then we check whether C is highly connected and compute the sum of the f-values of each vertex to decide whether the vertex set of C is γ-isolated. By the same arguments as above, the computation of the minimum cut size can be performed in $O(\gamma nm)$ time overall. The summation of the f-values can be performed in $O(n+m)$ time overall and for all connected components we can decide in $O(n+m)$ time whether they are highly connected. Hence, Rule 2 can be exhaustively applied in $O(\gamma nm)$ time.

Finally, we check whether Rule 1 is applicable which can be easily performed in $O(n+m)$ time. Altogether, we arrive at the claimed running time bound. □

Using the above, we can now present the branching algorithm.

Theorem 4. Isolated Highly Connected Subgraph *can be solved in* $O(4^\gamma n^2 + (kn+\gamma)nm)$ *time.*

Proof. We first reduce the instance with respect to Rules 1 to 3. By Lemma 6 this can be done in $O((kn+\gamma)nm)$ time. Next, we guess one vertex v that is in the solution S (by branching into n cases according to the n vertices). We start with $S' := \{v\}$ and try to extend S' to a solution. More precisely, we choose a vertex v' from the neighborhood of S' (that is, from $\bigcup_{u\in S'} N(u) \setminus S'$), and branch into two cases: add v' to S', or exclude v', that is, delete v' and increase $f(u)$ by one for all $u \in N(v')$. In the first case, we increase $|S'|$ by one. In the second case, we increase $\sum_{u\in S'} f(u)$ by at least one. Branching is performed until $|S'| = k$ or $\sum_{u\in S'} f(u)$ exceeds γ or the neighborhood of S' is empty. When $|S'|$ reaches k, we check whether S' is highly connected and γ-isolated under f, and if this is the case, we have found a solution. Otherwise, when $\sum_{u\in S'} f(u)$ exceeds γ or no branching is possible because the neighborhood of S' is empty, we abort the branch; in this case, clearly no superset of S' can be a solution. The height of the search tree is bounded by $k+\gamma$, and each branch can be executed in $O(n)$ time, yielding an overall running time bound of $O(n \cdot 2^{k+\gamma} \cdot n)$.

We now distinguish two cases: $k \leq \gamma$ and $k > \gamma$. In the first case, the claimed bound already holds; if $k > \gamma$, there is at least one vertex in S that has no neighbors outside of S. Thus, we can start with $S' := \{v\} \cup N(v)$; since v has more than $k/2$ neighbors in S, we have $|S'| > k/2+1$, and thus need to take less than $k/2$ branches of adding a vertex. Together with Lemma 5, we obtain the claimed bound. □

We now present a further way of analyzing the presented data reduction rules by showing that they can be used to obtain a *Turing kernelization* [1] for Isolated Highly Connected Subgraph parameterized by γ. Informally, in Turing kernelization one may create *many* problem kernels instead of just one problem kernel. Then, the solution to the parameterized problem can be computed by solving the problem separately on each of these problem kernels.

To motivate the Turing kernelization result we first observe that Isolated Highly Connected Subgraph does not admit a regular problem kernel: The

disjoint union of a set of graphs has an isolated highly connected subgraph if and only if at least one of the graphs has one. Hence, Isolated Highly Connected Subgraph has a trivial OR-composition which implies the following [2].

Proposition 1. Isolated Highly Connected Subgraph *does not admit a polynomial-size problem kernel with respect to γ unless NP $\subseteq$ coNP/poly.*

Before describing the Turing kernelization, we give a formal definition of this kernelization concept.

Definition 1. *Let L be a parameterized problem and let $g : \mathbb{N} \to \mathbb{N}$ be a computable function. A* Turing kernelization *for L is an algorithm that, for each instance (x, k), decides whether $(x, k) \in L$ in polynomial time using an oracle for $\{(x', k') \mid |x'| + k' \leq g(k) \wedge (x', k') \in L\}$. The sequence of queries posed to the oracle is called* Turing kernel. *We call $g(k)$ the* size *of the Turing kernel.*

We now describe the algorithm in detail. The first step is to reduce to the augmented version of Isolated Highly Connected Subgraph in which we introduce the vertex labeling f. Then, apply Rules 2 and 3 exhaustively. Afterwards, apply the following reduction rule which removes high-degree vertices.

Rule 4. *Let (G, k, γ) be an instance of* Isolated Highly Connected Subgraph *that is reduced with respect to Rules 2 and 3. If G contains a vertex v of degree at least $3\gamma - f(v)$, then remove v from G, and for each $u \in N(v)$ increase $f(u)$ by one.*

Proof (Correctness of Rule 4). Since G is reduced respect to Rules 2 and 3, we have $\gamma > k/2$ (due to Lemma 5). Thus, v has less than 2γ neighbors in any solution $G[S]$. This implies that v has more than $\gamma - f(u)$ neighbors in $V \setminus S$. Consequently, if $v \in S$, then S is not γ-isolated under f. Hence, v is not contained in any solution. □

Now we construct n instances of Isolated Highly Connected Subgraph that have $O(\gamma^3)$ vertices each. The original instance is a yes-instance if and only if one of these instances is a yes-instance. The idea is to exploit the fact that highly connected graphs have diameter two [11]. Thus, to find highly connected graphs, it is sufficient to explore the two-neighborhood of each vertex. More precisely, the instances are constructed as follows.

For each vertex $v \in V$, construct the graph $G_v := G[N_2[v]]$ where $N_2[v]$ is the set of all vertices that have distance at most two from v (including v). When solving the Isolated Highly Connected Subgraph instances we need to determine whether a subgraph is γ-isolated. Thus, the graph G_v has to contain information on the original vertex degrees. Note that for each $u \in V$, $f(u)$ denotes the number of edge deletions performed during the data reduction that are incident with u. Moreover, for each vertex u in G_v, let $g(u)$ denote the number of neighbors of u in G in $V \setminus N_2[v]$. To obtain instances of Isolated Highly Connected Subgraph one may not use vertex labelings. Thus, for each u of G_v add $g(u) + f(u)$ new vertices and make them adjacent to u. This

completes the construction of G_v. In this way we obtain n instances (G_v, k, γ). The following lemma shows that it is sufficient to solve these instances in order to determine whether the original Isolated Highly Connected Subgraph instance $(G = (V, E), k, \gamma)$ is a yes-instance. We use G' to denote the labeled graph of the augmented instance that is obtained from reducing (G, k, γ) to the augmented problem and applying Rules 2 to 4 exhaustively.

Lemma 7. *Let $(G = (V, E), k, \gamma)$ be an instance of* Isolated Highly Connected Subgraph *and, for each $v \in V$, let (G_v, k, γ) denote the instance as constructed above. Then, (G, k, γ) is a yes-instance if and only if there is a $v \in V$ such that (G_v, k, γ) is a yes-instance.*

Proof. If (G, k, γ) is a yes-instance, then there is a γ-isolated vertex set S of size k such that $G[S]$ is highly connected. Consider some $v \in S$. Then, $S \subseteq N_2[v]$ since $G[S]$ has diameter two. Thus, the subgraph $G_v[S]$ is highly connected. Moreover, S is γ-isolated in G_v. Let $\deg(u)$ denote the degree of a vertex u in G. In G' every vertex has degree $\deg(u) - v$ and before adding the degree-one vertices in the construction of G_v, every vertex has degree $\deg(u) - f(u) - g(u)$. Thus, every vertex has degree $\deg(u) - f(u) - g(u) + f(u) + g(u) = \deg(u)$ in the constructed G_v. Consequently, S is γ-isolated in G_v since for each $u \in S$ the vertex degrees in G_v and in G are the same.

For the converse, assume that there is a vertex $v \in V$ such that (G_v, k, γ) is a yes-instance. Then, let S be a γ-isolated vertex set of size k such that the subgraph of $G_v[S]$ is highly connected. First, observe that $S \subseteq V$: All vertices in G_v that are not from V have degree one. Since we assume that $k \geq 2$ these vertices cannot be in a highly connected subgraph with k vertices. Thus, $G[S]$ is a highly connected subgraph of G (since G_v is a subgraph of G). By the discussion above, the vertices of S have the same degree in G and G_v. Thus, S is γ-isolated in G and (G, k, γ) is a yes-instance. □

We now show that the instances have bounded size.

Lemma 8. *Let (G_v, k, γ) be an instance of* Isolated Highly Connected Subgraph *constructed from G as described above. Then G_v has less than $(3\gamma)^3$ vertices and less than $3\gamma^4$ edges.*

Proof. The graph G' from which we construct (G_v, k, γ) is reduced with respect to Rule 4. Thus, the maximum degree in G' is at most $3\gamma - 1 - f(u)$ which implies $|N_2[v]| < 1 + 3\gamma - 1 + (3\gamma - 1)^2$. For each vertex in $N_2[v]$ we then add further degree-one neighbors, but the difference between the degree of u in G' and G is exactly $f(u)$. Thus, each vertex in G_v has degree at most $3\gamma - 1$. Consequently, G_v has at most $(1 + 3\gamma - 1 + (3\gamma - 1)^2) \cdot (3\gamma - 1) < (3\gamma)^3$ vertices. Moreover, as each vertex in G_v has degree at most $3\gamma - 1$, G_v has also less than $(3\gamma)^4$ edges. □

Combining Lemmas 7 and 8 leads to the following.

Theorem 5. Isolated Highly Connected Subgraph *admits a Turing kernel of size $O(\gamma^4)$ which has less than $(3\gamma)^3$ vertices.*

7 Edge Deletion Parameter

We now show that there is a subexponential fixed-parameter algorithm for Highly Connected Subgraph with respect to the number of edges α we are allowed to delete in order to obtain a highly connected graph of order k. The algorithm is a search tree algorithm which branches on whether or not a given vertex is part of the highly connected graph. Repeated application of two reduction rules (similar to Rules 2 and 3 above) ensures that the branches are effective in reducing the remaining search space. To give a precise presentation of the branching step and the reduction rules, we define the problem with an additional *seed* S, a set of vertices which have to be in the highly connected graph.

> Seeded Highly Connected Edge Deletion
> Input: An undirected graph $G = (V, E)$, a vertex set $S \subseteq V$, and nonnegative integers k and α.
> Question: Is there a set $E' \subseteq E$ of at most α edges such that $G - E'$ consists only of degree-zero vertices and a $(k + |S|)$-vertex highly connected subgraph containing S?

If we set $S = \emptyset$, we obtain the plain edge deletion problem. The reduction rules are as follows.

Rule 5. *If there is a connected component $C = (V', E')$ of G that has minimum cut size at least $\alpha + 1$, then accept if C is highly connected, $S \subseteq V'$, $|V' \setminus S| = k$, and the remaining connected components of G contain at most α edges. Otherwise reject.*

Proof (Correctness of Rule 5). The rule is clearly correct if it accepts. If it rejects, then the instance is a no-instance: If there is a highly connected graph in $G[V \setminus V']$, then the $\alpha + 1$ or more edges of C are not in this graph. Thus, any solution is contained in C. Hence, if the number of edges in the remaining components is more than α or if $S \setminus V' \neq \emptyset$, then the instance is a no-instance. Otherwise, either C is not highly connected or $|V' \setminus S| \neq k$. In both cases a highly connected graph of order $|S| + k$ that is contained in C has less than $|V'|$ vertices and thus it needs to be cut from the rest of C. This needs at least $\alpha + 1$ edges. Consequently, there is no solution and the instance is a no-instance. □

Rule 6. *If there is a connected component of G that has a minimum cut of size at most $(k + |S|)/2$, then delete all cut edges and reduce α by their number.*

Proof (Correctness of Rule 6). Every graph G' with non-empty intersection with both "sides" of the minimum cut has a cut of size $(k + |S|)/2$. Thus, if G' has order $(k + |S|)$, then it is not highly connected. Hence, for each deleted edge at least one of its endpoints is not in any solution. □

Similarly to the edge isolation parameter, after using the reduction rules k, $|S|$, and α are related.

Lemma 9. *If Rules 5 and 6 are not applicable, then* $\alpha > (k + |S|)/2$.

Proof. Assume Rules 5 and 6 are not applicable but $\alpha \leq (k+|S|)/2$. Without loss of generality we may assume that there are no connected components consisting of singleton vertices. Otherwise, simply remove them. Hence, each connected component has a minimum cut cutting at least one edge. Further, there is a connected component with minimum cut of size at most α because Rule 5 is not applicable. By assumption $\alpha \leq (k + |S|)/2$ and hence, there is a connected component with a minimum cut of size at most $(k + |S|)/2$ which contradicts the inapplicability of Rule 6. □

The running time of the rules can be bounded in a similar way as it was done in Section 6.

Lemma 10. *Rules 5 and 6 are exhaustively applicable in* $O(\alpha^2 nm)$ *time.*

Proof. We first decide for each connected component of G whether it contains a minimum cut of size at most $\alpha + 1$ by fixing an arbitrary vertex v and for each vertex u running $\alpha + 2$ rounds of the Ford-Fulkerson algorithm to find a flow from v to u. If at some round the flow does not increase, we find a corresponding cut by considering the strongly connected components in the residual graph and apply the rules. We iterate the procedure if Rule 6 was applicable.

Both rules, finding the connected components in the residual graph, and the rounds of Ford-Fulkerson can each be implemented to run in $O(n + m)$ time. Hence if $C_1, \ldots, C_\ell$ are the connected components of G then one iteration takes $\sum_{i=1}^{\ell} O(\alpha|C_i|(|C_i| + |E(G[C_i])|)) = O(\alpha nm)$-time. Since if Rule 5 applies we are finished and if Rule 6 applies both the number of connected components increases and α decreases, the whole procedure takes $O(\min\{n, \alpha\}\alpha nm)$ time. □

Exhaustively applying the reduction rules lets us bound the number of the remaining vertices linearly in α. This will be useful in the branching algorithm below.

Theorem 6. Seeded Highly Connected Edge Deletion *admits a problem kernel with at most* $2\alpha + 4\alpha/k$ *vertices and* $\binom{2\alpha}{2} + \alpha$ *edges computable in* $O(\alpha^2 nm)$ *time.*

Proof. The kernelization algorithm first exhaustively applies Rules 5 and 6. By Lemma 10 this needs $O(\alpha^2 nm)$ time. Let $(G = (V, E), k, \alpha)$ be a yes-instance output by this procedure and let $E' \subseteq E$ be of minimum size such that $G - E'$ consists of degree-zero vertices and a highly connected subgraph $G[S']$ such that $S \subseteq S'$ and $|S'| = |S| + k$. We first bound $|V \setminus S'|$. Because the instance is reduced with respect to Rule 6, the graph G has minimum vertex degree $k/2$. Hence, the number of edges incident with at least one vertex in $V \setminus S'$ is at least $|V \setminus S'| \cdot k/4$. This number is at most α and thus $|V \setminus S'| \cdot k/4 \leq \alpha$ which implies $|V \setminus S'| \leq 4\alpha/k$. Thus G contains at most $k + |S| + 4\alpha/k$ vertices. By Lemma 9, $\alpha > (k + |S|)/2$. This implies $|S'| < 2\alpha$ and thus also that the number of edges within a solution $G[S']$ is less than $\binom{2\alpha}{2}$. Hence, every instance with at least $2\alpha + 4\alpha/k$ vertices or $\binom{2\alpha}{2} + \alpha$ edges is a no-instance and can be rejected immediately. □

In the subexponential branching algorithm, we use the following simple branching rule. It simply takes a vertex and branches on whether or not it should be added to the seed S for the desired highly connected graph.

Branching Rule 1. *If $\alpha + k \geq 0$, then choose an arbitrary vertex $v \in V \setminus S$ and branch into the cases of adding v to S or removing v from G. That is, create the instances $I_1 = (G, S \cup \{v\}, k-1, \alpha)$ and $I_2 = (G - v, S, k, \alpha - \deg_G(v))$. Accept if I_1 or I_2 is accepted.*

It is clear that Branching Rule 1 is correct. We now describe the complete algorithm and bound its running time.

Theorem 7. *There is an $O(2^{4 \cdot \alpha^{0.75}} + \alpha^2 nm)$-time algorithm for* Highly Connected Edge Deletion.

Proof. We first apply the kernelization from Theorem 6, which entails applying Rules 5 and 6 exhaustively. Then, if $k \leq 2\sqrt{\alpha}$ we check whether $S \cup V'$ induces a highly connected subgraph, for every vertex subset $V' \subseteq V \setminus S$ of size k. We accept or reject accordingly. If $k > 2\sqrt{\alpha}$, then we apply Branching Rule 1 and recurse on the two created instances.

From the correctness of the rules it is clear that this algorithm finds a solution if there is one. Let us analyze its running time. Note that in each recursive call, except the first one, the input instance has $O(\alpha)$ vertices according to Theorem 6. Thus applying Rules 5 and 6 in a recursive call amounts to $O(\alpha^5)$ time except in the first one where it is $O(\alpha^2 nm)$ time. Next, in each recursive call we may have to check whether $S \cup V'$ is highly connected for all k-vertex subsets V'. This is done only after Rules 5 and 6 have been exhaustively applied and only if $k \leq 2\sqrt{\alpha}$. Thus, the graph G is of order at most $2\alpha + 4\alpha/k \leq 4\alpha$ (note that $k \geq 2$ without loss of generality). Hence, testing the subgraphs amounts to $O((4\alpha)^{2\sqrt{\alpha}+2})$ time. In total, the time spent per search tree node is $O((4\alpha)^{\max\{5, 2\sqrt{\alpha}+2\}})$.

Now let us bound the number of leaves C of the search tree. Note that the total number of search tree nodes is within a constant factor of C. For an instance $I = (G, S, k, \alpha)$ of Highly Connected Subgraph, consider the value $\mu(I) = k + \alpha$ in the root of the search tree, after applying Rules 5 and 6. Then, $\mu(I) \leq 3\alpha$ by Lemma 9. Let $C(\mu(I))$ denote the number of leaves that a search tree with a root with value $\mu(I)$ can have. Whenever we apply Branching Rule 1, μ is reduced by a certain amount. More precisely, $C(\mu(I))$ fullfills $C(0) = 1$, and $C(\mu(I)) \leq C(\mu(I_1)) + C(\mu(I_2))$. Hence, $C(\mu(I))$ is monotone. Further, since Rule 6 is not applicable, $\deg_G(v) \geq (|S| + k)/2 \geq k/2 \geq \sqrt{\alpha}$ in the application of Branching Rule 1. This implies $C(\mu(I)) \leq C(\mu(I) - 1) + C(\mu(I) - \sqrt{\alpha})$. Hence $C(\mu(I))$ is at most the number of lattice paths from the origin to some point (x, y) that take only steps $(1, 0)$ or $(0, \sqrt{\alpha})$, where $x + y = \mu(I)$. Scaling the y-axis by a factor of $1/\sqrt{\alpha}$, computing $C(\mu(I))$ reduces to the problem of counting the lattice paths from the origin to some (x, y') taking only steps $(1, 0)$ or $(0, 1)$ such that $x + \sqrt{\alpha}y' = \mu(I)$. We now bound the number of these paths.

The number of $(0, 1)$ steps is at most $3\sqrt{\alpha}$. If the path contains i $(0, 1)$-steps, then the total number of steps in the path is $i + 3\alpha - \sqrt{\alpha}i$. Hence,

there are $\binom{i+3\alpha-\sqrt{\alpha}i}{i}$ paths with exactly i steps $(0,1)$. This implies $C(\mu(I)) \leq \sum_{i=0}^{3\sqrt{\alpha}} \binom{i+3\alpha-\sqrt{\alpha}i}{i}$. To bound this number we use the fact that $\binom{a+b}{a} \leq 2^{2\sqrt{ab}}$ [10, Lemma 9]. Hence $C(\mu(I)) \leq \sum_{i=0}^{3\sqrt{\alpha}} 2^{2\sqrt{i\cdot(3\alpha-\sqrt{\alpha}i)}}$. Consider the derivative $f(i)$ of $2\sqrt{i\cdot(3\alpha-\sqrt{\alpha}i)}$ with respect to i. We have

$$f(i) = \frac{\sqrt{\alpha}(3\sqrt{\alpha}-2i)}{\sqrt{\sqrt{\alpha}i(3\sqrt{\alpha}-i)}}.$$

Inspecting $f(i)$ shows that $\sqrt{i\cdot(3\alpha-\sqrt{\alpha}i)}$ is maximized over $0 \leq i \leq 3\sqrt{\alpha}$ if $i = 3\sqrt{\alpha}/2$. This gives $C(\mu(I)) \leq 3\sqrt{\alpha}\cdot 2^{3\sqrt{\alpha\sqrt{\alpha}}}$. Finally,

$$3\sqrt{\alpha}\cdot 2^{3\sqrt{\alpha\sqrt{\alpha}}}\cdot(4\alpha)^{\max\{5,2\sqrt{\alpha}+2\}} \in O(2^{4\alpha^{0.75}}),$$

giving the overall running time bound of $O(2^{4\alpha^{0.75}} + \alpha^2 nm)$. □

Although the presented algorithm is a subexponential-time algorithm with relatively small constants in the exponential functions, it is unclear whether it can be useful in practice. This is because the parameter α is likely to be large in real-world instances. With further substantial running time improvements, however, one might obtain practical algorithms. For instance, an algorithm with running time $2^{\alpha^{0.5}}\cdot n^{O(1)}$ should perform well on many real-word instances.

References

[1] D. Binkele-Raible, H. Fernau, F. V. Fomin, D. Lokshtanov, S. Saurabh, and Y. Villanger. Kernel(s) for problems with no kernel: On out-trees with many leaves. *ACM Transactions on Algorithms*, 8(4):38, 2012.

[2] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009.

[3] J. Bondy and V. Chvátal. A method in graph theory. *Discrete Mathematics*, 15(2):111–135, 1976.

[4] G. Chartrand. A graph-theoretic approach to a communications problem. *SIAM J. Appl. Math.*, 14(4):778–781, 1966.

[5] J. Chen, B. Chor, M. Fellows, X. Huang, D. W. Juedes, I. A. Kanj, and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. *Information and Computation*, 201(2):216–231, 2005.

[6] J. Chen, I. A. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40–42), 2010.

[7] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.

[8] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity.* Texts in Computer Science. Springer, 2013.

[9] M. R. Fellows, B. M. P. Jansen, and F. A. Rosamond. Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *Eur. J. Combinatorics*, 34(3):541–566, 2013.

[10] 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.

[11] E. Hartuv and R. Shamir. A clustering algorithm based on graph connectivity. *Inf. Process. Lett.*, 76(4–6):175–181, 2000.

[12] Q.-S. Hua, Y. Wang, D. Yu, and F. C. M. Lau. Dynamic programming based algorithms for set multicover and multiset multicover problems. *Theor. Comput. Sci.*, 411(26-28):2467–2474, 2010.

[13] F. Hüffner, C. Komusiewicz, A. Liebtrau, and R. Niedermeier. Partitioning biological networks into highly connected clusters with maximum edge coverage. *IEEE/ACM Trans. Comput. Biol. Bioinf.*, 11(3):455–467, 2014.

[14] H. Ito and K. Iwama. Enumeration of isolated cliques and pseudo-cliques. *ACM Trans. Algorithms*, 5(4):Article 40, 2009.

[15] H. Ito, K. Iwama, and T. Osumi. Linear-time enumeration of isolated cliques. In *Proc. 13th ESA*, volume 3669 of *LNCS*, pages 119–130. Springer, 2005.

[16] C. Komusiewicz, F. Hüffner, H. Moser, and R. Niedermeier. Isolation concepts for efficiently enumerating dense subgraphs. *Theor. Comput. Sci.*, 410(38-40):3640–3654, 2009.

[17] D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the EATCS*, 105:41–71, 2011.

[18] H. Matsuda, T. Ishihara, and A. Hashimoto. Classifying molecular sequences using a linkage graph with their pairwise similarities. *Theor. Comput. Sci.*, 210(2):305–325, 1999.

[19] J. Pattillo, N. Youssef, and S. Butenko. On clique relaxation models in network analysis. *Eur. J. Operational Research*, 226(1):9–18, 2013.

[20] R. Sharan, I. Ulitsky, and R. Shamir. Network-based prediction of protein function. *Molecular Systems Biology*, 3:88, 2007.

[21] A. Veremyev, O. A. Prokopyev, V. Boginski, and E. L. Pasiliao. Finding maximum subgraphs with relatively large vertex connectivity. *Eur. J. Operational Research*, 239(2):349–362, 2014.