Parameterized Complexity of Multi-Node Hubs

Saket Saurabh
University of Bergen, Norway, and Institute of Mathematical Science, HBNI, India
saket@imsc.res.in

Meirav Zehavi
Ben-Gurion University, Israel
meiravze@bgu.ac.il

Abstract

Hubs are high-degree nodes within a network. The examination of the emergence and centrality of hubs lies at the heart of many studies of complex networks such as telecommunication networks, biological networks, social networks and semantic networks. Furthermore, identifying and allocating hubs are routine tasks in applications. In this paper, we do not seek a hub that is a single node, but a hub that consists of k nodes. Formally, given a graph $G = (V, E)$, we a seek a set $A \subseteq V$ of size k that induces a connected subgraph from which at least p edges emanate. Thus, we identify k nodes which can act as a unit (due to the connectivity constraint) that is a hub (due to the cut constraint). This problem, which we call Multi-Node Hub (MNH), can also be viewed as a variant of the classic Max Cut problem. While it is easy to see that MNH is W[1]-hard with respect to the parameter k, our main contribution is the first parameterized algorithm that shows that MNH is FPT with respect to the parameter p.

Despite recent breakthrough advances for cut-problems like Multicut and Minimum Bisection, MNH is still very challenging. Not only does a *connectivity constraint* has to be handled on top of the involved machinery developed for these problems, but also the fact that MNH is a *maximization* problem seems to prevent the applicability of this machinery in the first place. To deal with the latter issue, we give non-trivial reduction rules that show how MNH can be preprocessed into a problem where it is necessary to delete a bounded-in-parameter number of vertices. Then, to handle the connectivity constraint, we use a novel application of the form of tree decomposition introduced by Cygan et al. [STOC 2014] to solve Minimum Bisection, where we demonstrate how *connectivity constraints* can be replaced by simpler *size constraints*. Our approach may be relevant to the design of algorithms for other cut-problems of this nature.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases hub, bisection, tree decomposition

Digital Object Identifier 10.4230/LIPIcs.IPEC.2018.8

1 Introduction

Hubs are high-degree nodes within a network. The examination of the emergence and centrality of hubs lies at the heart of many studies of complex networks such as telecommunication networks, biological networks, social networks and semantic networks [25, 5, 1, 19, 35]. For example, hubs have been examined in the context of complex phylogenetic diseases such as asthma [33]; indeed, deletion of a hub protein is more likely to be lethal than deletion of a non-hub protein [21, 24]. Furthermore, identifying and allocating hubs are routine tasks in a wide-variety of applications [6]. In fact, the concept of hubs is also of significant interest in studies of hyperlinked environments [27], viral marketing [30], outbreaks of epidemics [4] and the Blogosphere [41].

It is natural to ask whether one can not only efficiently find a hub that is merely a single node, but a hub that consists of at most k nodes or (more generally) exactly k nodes.[1] More precisely, given a threshold parameter p, a hub can be defined as a node of degree at least p. Here, we introduce the concept of a *multi-node hub*, which is a connected set of nodes of degree at least p. In other words, we aim to identify k nodes which can act as a unit (due to the connectivity constraint) that is a hub (due to the cut constraint). For example, in the context of biological networks, we might want to seek a module or a protein complex that acts as a hub, and in the context of social networks, we might want to identify a group of people that together act as a hub. Formally, we solve the following problem.

Multi-Node Hub (MNH)	**Parameters:** k, p
Input: An undirected graph G such that $\|V(G)\| = n$, and positive integers k and p.	
Question: Does there exist a subset $A \subseteq V(G)$ of size exactly k such that $G[A]$ is connected, and $E(G)$ contains at least p edges with exactly one endpoint in A?	

Multi-node hubs are of interest also for large values of k: when k is close to n, a multi-node hub allows us to "shave" a few nodes from the network while ensuring that the core stays connected and has a large number of links to the nodes that were shaved. Moreover, we believe that multi-node hubs deserve to be studied also because they are simple, elegant combinatorial objects. The MNH problem involves three different types of constraints, namely a cut constraint, a size constraint and a connectivity constraint. On the one hand, it is easily seen to be MNH is W[1]-hard with respect to k; that is, it is unlikely that MNH is FPT with respect to k. On the other hand, it can be shown to be FPT with respect to $k + p$ (by a simple application of color-coding [3]). This brings us to the parameter p. By relying on a preprocessing phase inspired by algorithms for the Max Leaf Spanning Tree problem along with a novel application of special tree decompositions, namely "unbreakable tree decompositions", we are able to establish that MNH is FPT with respect to the parameter p.

Related Work. MNH is closely related to the classical NP-hard Max-Cut problem. Here, the input consists of a graph G and a positive integer p, and the objective is to check whether there is a cut of size at least p. A cut of a graph is a partition of the vertices of the graph into two disjoint subsets. The size of the cut is the number of edges whose endpoints belong to different subsets of the partition. Max-Cut has been the focus of extensive study from the algorithmic perspective of computer science as well as the extremal perspective of combinatorics. Papadimitriou and Yannakakis showed that Max-Cut is APX-hard [37]. It is also well known that given an arbitrary partition of the vertex-set of G, by repeatedly moving vertices from one subset of the partition to the other as long as the size of the cut increases, one obtains a cut of size at least $|E(G)|/2$. A breakthrough result by Goemans and Williamson [20] gave a 0.878-approximation algorithm, which is optimal under the Unique Games Conjecture [26]. Max-Cut has also been well studied from the viewpoint of parameterized complexity [12, 13, 34, 38]. In this context, a notable work is the parameterized algorithm for an above-guarantee version of Max-Cut [13].

The $(k, n-k)$-Max Cut problem, which is a variant of Max Cut that has been extensively studied in the literature, is linked even more tightly to MNH. This variant asks whether there exists a subset A of the vertex-set of a given graph $G = (V, E)$ such that

[1] If we solve the problem with the condition "exactly", then we solve the version with the condition "at most" by trying every option to choose k. The condition "exactly" is of use when we have k tokens that should be placed on distinct nodes, which together should act as a hub.

$|A| = k$ and E contains at least p edges with exactly one endpoint in A. For this specific variant, Ageev and Sviridenko gave a 0.5-approximation algorithm [2], which has been slightly improved by Feige and Langberg [17]. It is also known that this variant is W[1]-hard with respect to the parameter k [9]. With respect to the parameter p, $(k, n-k)$-Max Cut can be solved in time $\mathcal{O}^*(p^p)$ [8]. This bound was improved to $\mathcal{O}^*(4^{p+o(p)})$ in [40], and to $\mathcal{O}^*(2^p)$ in [39]. The latter work also gives a polynomial kernel for $(k, n-k)$-Max Cut with respect to the parameter p. Bonnet et al. [8] also gave a parameterized approximation scheme for $(k, n-k)$-Max Cut with respect to the parameter k.

Furthermore, Connected Max-Cut (CMC), the variant of Max Cut where $G[A]$ should be a connected graph, has also been investigated in the literature. This variant is of interest, for example, in image segmentation [42]. Essentially, MNH can be defined as the variant of $(k, n-k)$-Max Cut that adopts the connectivity constraint of CMC. Hajiaghayi et al. [23] studied CMC from the view point of approximation, and obtained an $\Omega(\frac{1}{\log n})$-approximation algorithm for this problem. They explicitly note that since CMC has the flavors of both cut and connectivity problems simultaneously, well-known approaches used to develop algorithms for either connectivity problems or cut problems such as Minimum Bisection are *not* applicable to CMC. Recently, Lee et al. [29] developed a 0.5-approximation algorithm for the special case of CMC where the input graph has bounded treewidth. We remark that the study of the parameterized complexity of natural connectivity variants of well-known problems, such as Connected Vertex Cover [7, 14, 16, 18, 22, 36], is an active research area that has received considerable attention.

Our Contribution. In this paper, we initiate the study of multi-node hubs in general, and of algorithmic aspects of MNH in particular. Since the concept of multi-node hubs is a natural generalization of the concept of hubs, it can be of significant interest in various studies of complex networks where hubs play a central role. We believe that connectivity is the most fundamental condition to demand from multi-node hubs; in particular, it is the most basic requirement that can allow the nodes to act as a single unit. In specialized scenarios, one may want the hub to be "highly-connected" or to be structured in a manner compatible with some application-specific demands. We believe that our paper can lead to several follow-up works, not only from the viewpoint of Parameterized Complexity, on (possibly enriched) multi-node hubs.

While it is easy to observe that MNH is W[1]-hard with respect to the parameter k, our main contribution is the first parameterized algorithm that shows that MNH is FPT with respect to p (Section 4). Our algorithm is primarily a classification result and should be viewed as a proof of concept that the problem is FPT with respect to p. As a corollary to our algorithm we also get that CMC is FPT with respect to p.

Despite the recent breakthrough advances for cut-problems like Multicut and Minimum Bisection, MNH is still very challenging. Not only does a *connectivity constraint* has to be handled on top of the involved machinery developed for these problems, but also the fact that MNH is a *maximization* problem seems to prevent the applicability of this machinery in the first place. To deal with the latter issue, we give non-trivial reduction rules that show how MNH can be preprocessed into a problem where it is necessary to delete a bounded-in-parameter number of vertices. Then, to handle the connectivity constraint, we use a novel application of the form of tree decomposition introduced by Cygan et al. [15] to solve Minimum Bisection, where we demonstrate how *connectivity constraints* can be replaced by simpler *size constraints*. A more detailed description of this approach can be found in Section 3. We believe that our approach can be relevant to the design of algorithms for other cut-problems of this nature.

2 Preliminaries

Tree Decompositions. A *tree decomposition* of a graph G is a pair (D, β), where D is a rooted tree and $\beta : V(D) \to 2^{V(G)}$ is a mapping that satisfies the following conditions.

- For each vertex $v \in V(G)$, the set $\{d \in V(D) : v \in \beta(d)\}$ induces a nonempty and connected subtree of D.
- For each edge $\{v, u\} \in E(G)$, there exists $d \in V(D)$ such that $\{v, u\} \subseteq \beta(d)$.

The set $\beta(d)$ is called the *bag at* d, while sets $\beta(d) \cap \beta(t)$ for $\{d, t\} \in E(D)$ are called *adhesions*. Furthermore, given a node $d \in V(D)$, we denote $\sigma(d) = \beta(d) \cap \beta(\text{parent}(d))$ in case d is not the root, and $\sigma(d) = \emptyset$ otherwise. We also denote $\gamma(d) = (\bigcup_{t \in \mathsf{des}(d)} \beta(t)) \cup \beta(d)$.

Unbreakability. Given a graph G, we say that a pair (A, B) such that $A \cup B = V(G)$ is a *separation* if $E(A \setminus B, B \setminus A) = \emptyset$. The *order* of the separation is $|A \cap B|$.

Roughly speaking, a vertex-set $U \subseteq V(G)$ is unbreakable in the graph G if every separation of $V(G)$ of a small order must contain a large chunk of U in one of its two sets. Intuitively, this means that the set U is "highly-connected": any attempt to "break" it severely by using a separation of a small order is futile. Formally, an unbreakable set is defined as follows.

▶ **Definition 1.** Given a graph G, a set $U \subseteq V(G)$ is (q, p)-*unbreakable in* G if for any separation (A, B) of order at most p, we have that $|(A \setminus B) \cap U| \leq q$ or $|(B \setminus A) \cap U| \leq q$.

A (q, p)-*unbreakable tree decomposition*, a concept introduced in [15], is a tree decomposition which satisfies several properties relating to the unbreakability and connectedness of the "parts" to which it partitions the graph. These properties will be extremely valuable in Section 4, where we solve the MNH problem. In particular, such a decomposition consists of (q, p)-unbreakable bags; thus, for example, if we remove at most p vertices from the graph, and we want to ensure that there remains a path between two vertices that belong to the same bag, it is sufficient to ensure that each of these vertices is contained in a large enough connected component (consisting of vertices from the bag). Formally, an unbreakable tree decomposition is defined as follows.

▶ **Definition 2.** We say that a tree decomposition (D, β) of a graph G is (q, p)-*unbreakable* if $|V(D)| \leq |V(G)|$ and the following conditions hold.

1. For each $d \in V(D)$, the graph $G[\gamma(d) \setminus \sigma(d)]$ is connected and $N(\gamma(d) \setminus \sigma(d)) = \sigma(d)$.
2. For each $d \in V(D)$, the set $\beta(d)$ is (q, p)-unbreakable in $G[\gamma(d)]$.
3. For each non-root $d \in V(D)$, we have that $|\sigma(d)| \leq q$ and $\sigma(d)$ is $(2p, p)$-unbreakable in $G[\gamma(\mathsf{parent}(d))]$.

Cygan *et al.* [15] gave an efficient computation of unbreakable tree decompositions, summarized in the result below.

▶ **Theorem 3.** *For some constant $c > 0$, there is an $\mathcal{O}^*(2^{\mathcal{O}(p^2)})$-time algorithm that, given a connected graph G, computes a $(2^{cp}, p)$-unbreakable tree decomposition of G.*

3 Our Technique

To handle the cut constraint ($|E(A, V(G) \setminus A)| \geq p$), we employ the randomized contractions technique [10], for which an excellent high-level description is given in Section 1.1 of [11]. Since our problem also involves a size constraint ($|A| = k$), we also rely on Theorem 3 (Section 2). In short, Cygan et al. [15] showed how to replace the recursion scheme of

the randomized contractions technique with a dynamic programming computation over a static tree decomposition; this enabled a possibility for a DP state to depend on a few counters (like sizes of parts in the case of Minimum Bisection of [15]), unachievable in the pure randomized contractions framework. The randomized contractions framework, regardless of whether one uses the original recursion scheme or the tree decomposition of [15], always requires significant technical work, but in most so-far known cases, this work is not particularly novel – it follows the bumpy but well-paved way of [10]. This is not the case in the study of the MNH problem.

First, note that the MNH problem asks for a cut of size *at least* p, whereas all known uses of randomized contractions focused on cuts of size bounded-in-parameter. Thus, directly, it seems that randomized contractions or the decomposition of [15] would not give much. Our first main insight is that if $n \geq k + p$, then a subgraph being a small enough tree with at least p leaves witnesses a YES-instance, in which case the problem becomes similar to the Max Leaf Spanning Tree problem [28], and can be solved using the method of bounded search trees. Thus we are left with the case where $|V(G) \setminus A| \leq p$, that is, we are to delete bounded-in-parameter number of vertices from the graph (delete $V(G) \setminus A$), such that the graph stays connected, but at least p edges "stick out" of the remaining graph. Now we think of the set $V(G) \setminus A$ as the "cut-set" for the randomized contractions technique, and finally we can start up the entire machinery. To this end, we give one more important problem-specific insight that greatly simplifies the use of randomized contractions: once we know $n < k + p$, then it is easy to check if a vertex v of degree $> 2p$ can be put outside A – any set A of size k not containing v needs to have $\geq p$ edges that touch v. Thus, after easy filtering, we can put all vertices of degree $> 2p$ into A. By replacing each of these vertices by a binary tree, we obtain a graph of degree bounded by $2p$. This degree bound makes the use of "flower cuts" (see [10]) unnecessary; intuitively, it makes the picture much closer to edge-cuts than vertex-cuts.

At this point, we finally employ the special form of tree decomposition of Cygan et al. [15]. We use this decomposition in a non-standard manner, which relies on a twist of the understanding of the meaning of a cut-set in the randomized contractions framework, that may be of independent interest. In particular, we demonstrate how this decomposition can be used to tackle difficult *connectivity constraints* by replacing them by simpler *size constraints*: since each bag in the tree decomposition is a "highly-connected" subgraph of G (see Section 2), removing a few vertices from G cannot create two (separate) connected components that contain many vertices from the bag. Thus, instead of using direct computations to ensure the existence of paths between certain vertices, which may be very inefficient in case the paths are long, we can sometimes merely verify that the vertices are contained in large connected components.

Each node of the tree decomposition will be associated with a problem reflecting the "work" that should be performed when handling its bag. The definition is quite technically involved, since it is has to capture *both connectivity and cut constraints*, and since it has to incorporate information provided by solutions for problems associated with the bags of the children (which also capture both connectivity and cut constraints), while providing enough information that will next allow us to solve the problem associated with the bag of the parent. By proving several observations that aim to bound the sizes of "problematic" sets of vertices (e.g., vertices in the "surroundings" of vertices that should not be put in A), we can develop a procedure that solves the problem associated with a single bag, based on the classic color coding method [3]. More precisely, by relying on the properties of an unbreakable tree decomposition as well as the fact that $n < k + p$ and $\Delta_G < 2p$, we implicitly

show that only for a bounded-in-parameter number of the problems associated with the bags of the children we may not simply take all of the vertices of the graph, and that it is sufficient to classify correctly (via color coding) only a bounded-in-parameter number of vertices in the "surroundings" of vertices that should not be put in A. We also bound the number of vertices that surround these "surroundings", and thus, essentially, we obtain connected components that can be handled in a partially independent manner that relies on dynamic programming. The solutions to the problems associated with the individual bags are combined by using a somewhat technically involved procedure that is also based on dynamic programming.

4 Multi-Node Hub is FPT with Respect to p

In this section we prove our main result:

▶ **Theorem 4.** *The* MNH *problem is solvable in time* $\mathcal{O}^*(2^{2^{\mathcal{O}(p)}})$.

We first solve simple cases of MNH (Section 4.1), where either the graph G contains many vertices (more precisely, $n \geq k+p$) or there exists a solution A such that $V(G) \setminus A$ contains a vertex of high-degree. Then, we turn to focus on the difficult instances whose handling relies on the properties of an unbreakable tree decomposition (Section 4.2).

4.1 The Preprocessing Phase

We first handle instances where G contains many vertices (Lemma 6). Then, we handle the remaining instances to which there is a solution A that excludes a high-degree vertex (Lemma 7). Thus, we overall define special instances (called useful instances) of a new problem (called Special MNH (SMNH)), which encapsulates the difficult instances of MNH (Lemma 8).

Assume WLOG that G is a connected graph. To solve instances where $n \geq k+p$, we need the following notation. Given a rooted tree T, $\mathsf{leaves}(T)$ and $\mathsf{internal}(T)$ denote the leaf-set and the set of internal vertices of T, respectively. Given two trees, T and T_e, we say that T_e *extends* T if T is a subtree of T_e. Observe that if T_e extends T, then the number of leaves of T_e is larger or equal to the number of leaves of T. This leads us to the next simple observation.

▶ **Observation 5.** *If a graph G on at least $k+p$ vertices contains a subtree T with at most k internal vertices and at least p leaves, then (G,k,p) is a yes-instance of* MNH.

Proof. Recall that G is a connected graph. Thus, since $n \geq k+p$, as long as $|\mathsf{internal}(T)| < k$ and $|V(T)| < k+p$, we can extend T to a tree T_e such that $|V(T_e)| = |V(T)|+1$. At the end of this process, we obtain a tree T' with at least p leaves (since the original tree T had at least p leaves) such that $|\mathsf{internal}(T')| \leq k$ and $|V(T')| \geq k+p$. We let $A = \mathsf{internal}(T') \cup U$ where U is an arbitrarily chosen subset of $\mathsf{leaves}(T')$ of size exactly $k - |\mathsf{internal}(T')|$. Thus, we obtain a solution A to the instance (G,k,p) of MNH. ◀

We are now ready to handle instances where $n \geq k+p$.

▶ **Lemma 6.** MNH *restricted to instances where $n \geq k+p$ is solvable in time* $\mathcal{O}^*(2^{\mathcal{O}(p)})$.

Proof. Let (G,k,p) be an instance of MNH where $n \geq k+p$. The main reason why such an instance is simple lies in Observation 5, which implies that once we find a tree with a small number of internal vertices and a large number of leaves, we are done. This observation is exploited by our bounded search tree-based procedure, called MNH. During its execution, we maintain a tree T, and make "difficult" decisions (that is, decisions made by using branching

rather than reduction rules) relating to the insertion of vertices into a solution A when handling vertices that should next be internal vertices with at least two children or fixed as leaves in the tree T. Since a tree with at most p leaves has at most p internal vertices with at least two children, Observation 5 bounds the number of difficult decisions we need to make (along each branch of the search tree) before we can conclude whether there is a solution.

Each call to MNH has the form $\mathsf{MNH}(G, k, p, T, O, I)$, where T is a rooted subtree of G, $O \subseteq \mathsf{leaves}(T)$, and $I \subseteq \mathsf{leaves}(T) \setminus O$ such that $N_G(I) \subseteq V(T)$. Informally, $\mathsf{internal}(T) \cup I$ contains the vertices we decided to insert into the solution A that we attempt to construct, while O contains those we decided to leave outside A. We further ensure that for every vertex $v \in \mathsf{internal}(T)$, T contains the entire set $N_G(v)$ (i.e., $N_G(v) \subseteq V(T)$). In other words, given a vertex which we decided to insert into A (by inserting it into either I or $\mathsf{internal}(T)$), we can assume that all of its neighbors belong to T. That is, we rely on the following assumption:

For every vertex $v \in I \cup \mathsf{internal}(T)$, we have that $N_G(v) \subseteq V(T)$.

Since T, O and I are the only arguments changed during the execution, we use the abbreviation $\mathsf{MNH}(T, O, I)$. The goal of such a call is to accept if there is a solution A to the instance (G, k, p) of MNH such that $\mathsf{internal}(T) \cup I \subseteq A$ and $O \cap A = \emptyset$, and to reject if there is no solution A to this instance (in the other cases, we can either accept or reject). Clearly, we can use MNH to solve MNH in the following manner. We call it with every subtree T of G that consists of exactly one internal vertex along with all of its neighbors as children (which will be leaves), and $O = I = \emptyset$. Then, we accept if and only if at least one of the calls to MNH accepts.

To analyze the rules of MNH, we use the measure $3p - |E(\mathsf{internal}(T) \cup I, O)| - |\mathsf{leaves}(T)| - |I \cup O|$. The necessity of incorporating $|E(\mathsf{internal}(T) \cup I, O)|$ in the measure will be clear at the analysis of the branching vector of our last rule. Initially, the measure is at most $3p$. At the latest, when the measure drops to a non-positive value, we will ensure that MNH returns an answer in polynomial time. We will also ensure that the branching rules performed by MNH are associated with branching vectors whose roots are bounded by a *fixed* c. Thus, we get that the overall running time is bounded by $\mathcal{O}^*(2^{\mathcal{O}(p)})$. Now, it remains to give the list of rules of $\mathsf{MNH}(T, O, I)$, and prove that they satisfy the above mentioned properties. We note that although our problem resembles the Max Leaf Spanning Tree problem [28], there are important differences between the two problems, which forces us to devise a different, somewhat technically involved, set of rules (which is, nevertheless, inspired by the approach of [28]). In particular, recall that our goal is *not* to accept if and only if G contains a subtree with at least p leaves – however, if we do find a *small enough* such subtree, we can terminate the execution of our procedure.

Due to lack of space, the list of rules is omitted. ◀

We are only left with the case $n < k + p$. Next, we show a lemma that will help us in assuming that the maximum degree of the input graph is bounded by $2p$.

▶ **Lemma 7.** *Given an instance (G, k, p) of* MNH *where $n < k+p$, in polynomial time we can decide whether there exists a solution A that does not contain $\widetilde{R} = \{v \in V(G) : |N(v)| \geq 2p\}$.*

Proof. Suppose there exists a solution A such that $\widetilde{R} \setminus A \neq \emptyset$. Furthermore, let $v^* \in (\widetilde{R} \setminus A)$. Then, there is a connected component C^* in $G[V \setminus \{v^*\}]$ on at least k vertices. As long as $|V(C^*)| > k$, compute a tree that spans C^* and remove from C^* a vertex that is a leaf in this tree. At the end of this process, we obtain a connected subgraph A of $G[V \setminus \{v^*\}]$ on exactly k vertices. Since $n < k + p$ and $|N(v)| \geq 2p$, this subgraph contains at least p vertices from $N(v)$. Therefore, $|E(V(A), V(G) \setminus V(A))| \geq p$. Thus, we get that $V(A)$ is a

solution to (G, k, p) of the desired kind. Our argument shows that to decide whether there is a solution A such that $\widetilde{R} \setminus A \neq \emptyset$, we can simply check whether there is a vertex $v \in \widetilde{R}$ such that $G \setminus \{v\}$ has a connected component of size at least k. This can be done in polynomial time. ◀

When we will solve difficult instances, it will be useful to assume that we handle graphs G such that $\Delta_G < 2p$. To this end, we define the Special MNH (SMNH) problem as follows. The input (G, k, p, R) consists of an instance (G, k, p) of MNH and a subset $R \subseteq V(G)$. The goal is to decide if there is a solution A to the instance (G, k, p) of MNH such that $R \subseteq A$. We say that an instance (G, k, p, R) for SMNH is *useful* if $n < k + p$ and $\Delta_G < 2p$. Next, we show that we can focus on solving useful instances of SMNH rather than instances of MNH.

▶ **Lemma 8.** *Given an instance (G, k, p) of* MNH*, there is an $\mathcal{O}^*(2^{\mathcal{O}(p)})$-time algorithm that either solves (G, k, p) or returns a useful instance (G', k', p, R) of* SMNH *such that (1) $|V(G')| = \mathcal{O}(|E(G)|)$, and (2) (G, k, p) is a yes-instance if and only if (G', k', p, R) is a yes-instance.*

Proof. Let $\widetilde{R} = \{v \in V(G) : |N_G(v)| \geq 2p\}$. By Lemmas 6 and 7, we can assume that (a) $n < k + p$, and (b) there is no solution A for (G, k, p) such that $\widetilde{R} \setminus A \neq \emptyset$. For each vertex $v \in \widetilde{R}$, let T_v be a binary tree on $|N_G(v)|$ leaves and $|N_G(v)| - 1$ internal vertices. While G contains a vertex $v \in \widetilde{R}$, insert T_v into G such that each leaf of T_v is connected by an edge to a distinct vertex in $N_G(v)$, and then remove v (and the edges containing v) from G. Let the resulting graph be G'. Clearly, $\Delta_{G'} < 2p$ and property (1) holds. Let $k' = |V(G')| - (n - k)$, and $R = \bigcup_{v \in \widetilde{R}} V(T_v)$. By assumption (a), $|V(G')| = k' + (n - k) < k' + p$, and thus we conclude that (G', k', p, R) is useful. By assumption (b) and our definition of G', k' and R, A is a solution for (G, k, p) if and only if $(A \setminus \widetilde{R}) \cup R$ is a solution for (G', k', p, R). Thus, property (2) holds. ◀

4.2 An Algorithm for the Difficult Instances

First, we briefly discuss some of the details of the dynamic programming computation that combines solutions to problems associated with individual bags (Section 4.2.1). Then, we formally define the problem associated with a single bag (Section 4.2.2). Due to lack of space, the algorithm that solves it is omitted.

4.2.1 Combining Individual Bags (Overview)

Let (G, k, p, R) be a useful instance of SMNH (i.e., $n < k + p$ and $\Delta_G < 2p$). Recall that we assume WLOG that G is a connected graph. Then, by Theorem 3, we can construct a $(2^{\mathcal{O}(p)}, p)$-unbreakable tree decomposition (D, β) in time $\mathcal{O}^*(2^{\mathcal{O}(p^2)})$. We let c be the constant such that (D, β) is $(2^{cp}, p)$-unbreakable. Let $r \in V(D)$ be the root of D, and recall that $\sigma(r) = \emptyset$. As usual for tree decompositions, we will use dynamic programming. Unlike dynamic programming over normal tree decompositions where the size of the bag is bounded, we have that for all $d \in V(D)$, $\beta(d)$ is $(2^{cp}, p)$-unbreakable. Thus, the problem we need to solve even for one bag becomes a problem in itself.

The Definition of $\mathcal{M}$. At every node d of the tree decomposition we keep a matrix (table) that stores how an optimal solution A^* when restricted to $G[\gamma(d)]$ could potentially look like. Towards this we use a matrix $\mathcal{M}$ that has an entry $[d, T, q, P^*]$ for all $d \in V(D)$, $R \cap \beta(d) \subseteq T \subseteq (R \cap \beta(d)) \cup \sigma(d)$, $q \in \{\max\{0, k - |(V(G) \setminus \gamma(d)) \cup (T \cap \sigma(d))|\}, \ldots, \min\{|\gamma(d) \setminus$

$\sigma(d)|, k - |T \cap \sigma(d)|\}\}$, and $P^* \subseteq \{\{v, u\} : v, u \in \sigma(d) \cap T\}$. The set T is used to guess the intersection of A^* with $\sigma(d)$, and the edge-set P^* is used to determine the partition of T into connected components. That is, if we look at the graph, say T^*, induced by $T \cap \sigma(d)$ together with edges from P^*, we get a set of connected components, and the idea is that when we restrict A^* to $G[\gamma(d)]$, say A^*_d, then every connected component of A^*_d must contain exactly one connected component of T^*. In other words, we use T^* to remember the connected components of A^*_d in $G[\gamma(d)]$. Next, consider some entry $\mathcal{M}[d, T, q, P^*]$.

We say that a subset $A \subseteq \gamma(d)$ is (d, T, q, P^*)-*valid* if the following conditions hold.

1. $T \subseteq A$, $(\sigma(d) \setminus T) \cap A = \emptyset$ and $R \cap \gamma(d) \subseteq A$.
2. If $\sigma(d) \cap T \neq \emptyset$, then every connected component of $G[A]$ contains a vertex from $\sigma(d)$, and otherwise $G[A]$ is a connected graph.
3. For all $\{v, u\} \in P^*$: $G[A]$ has a connected component containing both v and u.
4. $|A \setminus \sigma(d)| = q$.
5. If $A \neq \emptyset$, then $A \cap \beta(d) \neq \emptyset$.

For a (d, T, q, P^*)-*valid* set A, we define $\widetilde{p}(A) = |E(A, \gamma(d) \setminus A) \setminus E(A \cap \sigma(d), \sigma(d) \setminus A)|$. The entry $\mathcal{M}[d, T, q, P^*]$ is meant to store $\widetilde{p}_{d,T,q,P^*}$, the maximum value such that there exists a (d, T, q, P^*)-*valid* set A satisfying $\widetilde{p}(A) = \widetilde{p}_{d,T,q,P^*}$.

Having computed $\mathcal{M}$, we can solve the useful instance (G, k, p, R) of SMNH. Indeed, (G, k, p, R) is a yes-instance if and only if a value that is at least p is stored in at least one entry of the form $\mathcal{M}[d, T, k, \emptyset]$ where $\sigma(d) \cap T = \emptyset$. Since $\mathcal{M}$ contains $\mathcal{O}^*(2^{2^{\mathcal{O}(p)}})$ entries, it is sufficient to show that each entry of $\mathcal{M}$ can be computed in time $\mathcal{O}^*(2^{2^{\mathcal{O}(p)}})$.

Constructing an Instance of Single Bag. We compute the entries of $\mathcal{M}$ by using a postorder traversal; the order of computation between entries having the same argument d is arbitrary. Now, consider some entry $\mathcal{M}[d, T, q, P^*]$, where it is possible that d is a leaf, and assume that all of the preceding entries of $\mathcal{M}$ have been already computed correctly. To compute the entry $\mathcal{M}[d, T, q, P^*]$, we construct an instance $(H, k, p, T, U^*, P^*, \mathcal{U}, \mathsf{edges}, \mathsf{size}, \mathsf{cut})$ of a problem that we call Single Bag, and let $\mathcal{M}[d, T, q, P^*]$ store the result W_q obtained by calling the algorithm we shall develop for Single Bag, which runs in the desired time $\mathcal{O}^*(2^{2^{\mathcal{O}(p)}})$. Here, edges, size and cut are appropriate functions that are specified using the previously computed matrices at the children nodes. The parameters k and p are the same as those in the original instance of SMNH, and T and P^* are specified by the entry of $\mathcal{M}$. We let $H = G[\beta(d)]$ and $U^* = \sigma(d)$. Since $\Delta_G < 2p$, we have that $\Delta_H < 2p$. Moreover, we let $\mathcal{U}$ denote the *multiset* $\{\sigma(d') : d' \in \mathsf{children}_D(d)\}$. For each occurrence $U \in \mathcal{U}$, we let d_U denote a different child such that $\sigma(d_U) = U$. The functions are defined as follows.

- For each $U \in \mathcal{U}$, $\mathsf{edges}(U)$ contains every edge-set in $2^{\{\{v,u\}:v,u\in U\}}$. This is used to represent how a partial solution is connected in the graph $G[\sigma(d_U)]$.
- For each $U \in \mathcal{U}$, let $\mathsf{size}(U) = |\gamma(d_U) \setminus \sigma(d_U)|$. Moreover, let $\mathsf{size}(U^*) = |V(G) \setminus \gamma(d)|$.
- $\mathsf{cut} : \{(U, A_U, E_U, s_U) : U \in \mathcal{U}, A_U \subseteq U, E_U \in \mathsf{edges}(U), s_U \in \{0, 1, \ldots, \mathsf{size}(U)\}\} \to \{-\infty, 0, 1, 2, \ldots\}$. The function cut assigns a value to every tuple (U, A_U, E_U, s_U). Such a tuple represents a problem already solved that is associated with the bag $\beta(d_U)$. The tuple implies that in this problem, it is required that A_U is contained in the partial solution, connectivity of the partial solution is similar to the one given by edges in E_U, and exactly s_U vertices from $\gamma(d_U) \setminus \beta(d)$ are part of this partial solution. The value assigned by cut is the maximum number of edges that could emanate from a partial solution satisfying the connectivity and the size requirements given by A_U, E_U and s_U.

4.2.2 The Problem Associated with a Single Bag

In this section we define the problem, called Single Bag, which reflects the "work" that should be performed when handling each bag individually.

Input. The input is of the form $(H, k, p, T, U^*, P^*, \mathcal{U}, \mathsf{edges}, \mathsf{size}, \mathsf{cut})$, where

1. (H, k, p, T) is an instance of SMNH, where $\Delta_H < 2p$.
2. $U^* \subseteq V(H)$, $P^* \subseteq \{\{v, u\} : v, u \in U^* \cap T\}$.
3. $\mathcal{U}$ is a multiset of subsets of $V(H)$.
4. edges is a function that assigns to each set $U \in \mathcal{U}$ a subfamily of the family of edge-sets $2^{\{\{v,u\}:v,u\in U\}}$. The family $2^{\{\{v,u\}:v,u\in U\}}$ includes the edge-set $E^U_{\mathsf{com}} = \{\{v, u\} : v, u \in U\}$ such that the graph on the vertex-set U and the edge-set E^U_{com} is a clique.
5. $\mathsf{size} : \{U^*\} \cup \mathcal{U} \to \{0, 1, \ldots, n\}$. Recall that $n = |V(G)|$.
6. $\mathsf{cut} : \{(U, A_U, E_U, s_U) : U \in \mathcal{U}, A_U \subseteq U, E_U \in \mathsf{edges}(U), s_U \in \{0, 1, \ldots, \mathsf{size}(U)\}\} \to \{-\infty, 0, 1, 2, \ldots\}$.

We now attempt to give informal explanations of the intuition underlying the choice of the arguments of the input. Roughly speaking, they relate to a bag as follows. Let $\beta(d)$ be the bag associated with the node d of the tree decomposition that is currently being examined, $\beta(d^p)$ be the bag associated with the parent d_p, $\beta(d^c_1), \beta(d^c_2), \ldots, \beta(d^c_t)$ be the bags associated with the children, and $\gamma(d^c_i)$ be the union of $\beta(d^c_i)$ and the bags of the descendants of the child d^c_i for all $i \in \{1, 2, \ldots, t\}$.

Recall that G is the graph in the original instance of SMNH. In that context, we will let the graph H be the subgraph of G induced by the set of vertices in the current bag $\beta(d)$, i.e., $H = G[\beta(d)]$. The parameters k and p are the same as those in the original instance of SMNH, while T may contain, in addition to R, vertices from U^*. The arguments U^*, P^* and $\mathsf{size}(U^*)$ allow using solutions to the current problem (defined by d) when we next solve the problem defined by d_p, while the remaining arguments allow using solutions to the problems defined by $d^c_1, d^c_2, \ldots, d^c_t$ when solving the current problem. More precisely, U^* is the adhesion between d and d^p (i.e., $U^* = \beta(d) \cap \beta(d^p)$), while each set $U \in \mathcal{U}$ is the adhesion between $\beta(d)$ and the child $\beta(d^c_i)$. The function size assigns U^* the size of $V(G) \setminus \gamma(d)$ (which contains vertices yet to be handled), while it assigns each set $U \in \mathcal{U}$ the size of $\beta(d^c_i) \setminus U$ for the appropriate i (which contains vertices already handled). The set P^* contains pairs of vertices that should be connected in the solution for the current problem, to ensure the connectivity of the the graph induced by the solution A that we ultimately attempt to construct (upon handling all of the bags). When we solve the problem associated with d^p, it might not be possible/worthy to connect these vertices, and thus we need to have a set that specifies connectivity requirements that must already be addressed.

For each set $U \in \mathcal{U}$, edges assigns a family of edge-sets. Exactly one edge-set can be added to the graph H (see "Valid Triples"), and it is aimed to allow finding solutions that cannot be found without it (since a connectivity constraint will be broken). Each edge in such an edge-set represents a path that exists in a solution already computed (for a problem associated with a bag of a descendant). For example, when we next solve the problem associated with d^p, we will be able to use P^* as such an edge-set. Note that each edge-set is associated with a certain value, specified by cut, and thus it is not true that it is always best to choose a large, or even a non-empty, edge-set from $\mathsf{edges}(U)$. Finally, the function cut assigns a value to each tuple (U, A_U, E_U, s_U). Such a tuple represents a problem already solved, associated with d^c_i for the appropriate i (in particular, $\beta(d) \cap \beta(d^c_i) = U$). The tuple implies that in this problem, it was required that A_U is contained in the solution, paths

between every two vertices in E_U are located, and exactly s_U vertices from $\gamma(d_i^c) \setminus \beta(d)$ are inserted to the solution. The value assigned by cut is the number of edges between vertices in the solution (to the problem already solved) and the other vertices in $\gamma(d_i^c)$ (excluding edges between vertices in U).

Restrictions. Next, we denote $tSize = |V(H)| + \sum_{U \in \{U^*\} \cup \mathcal{U}} \mathsf{size}(U)$. For the sake of efficiency of our algorithm for Single Bag, we impose several restrictions on an input instance.

1. For all $U \in \mathcal{U} \cup \{U^*\}$: $|U| \leq 2^{cp}$, and each vertex in $V(H)$ is contained in at most $2p$ (not necessarily distinct) sets in the multiset $\mathcal{U}$.
2. $tSize = n < k + p$.
3. For all $(U, A_U, E_U, s_U) \in \mathsf{domain}(\mathsf{cut})$ such that $E_U \neq \{\{v, u\} : v, u \in U\}$, $U = A_U$ and $s_U = \mathsf{size}(U)$: $\mathsf{cut}(U, A_U, E_U, s_U) = -\infty$.

The first restriction bounds the size of sets that are part of an input instance, and thus it is clear that this restriction can potentially make the instance easier to solve efficiently. The second restriction will allow us to assume that there are not too many sets $U \in \mathcal{U}$ for which we may not use the default options $A_U = U$ and $s_U = \mathsf{size}(U)$, while the third restriction will overall allow us to conclude that there are not too many sets $U \in \mathcal{U}$ for which we may not use the default option $E_U = \{\{v, u\} : v, u \in U\}$. The claims are made precise in "Goal".

Output. For each $q \in \{\max\{0, k - \mathsf{size}(U^*) - |U^* \cap T|\}, \ldots, \min\{tSize - \mathsf{size}(U^*) - |U^*|, k - |U^* \cap T|\}\}$, we will need to find the "best" triple (A, f, g) that satisfies certain conditions. Thus, we first work towards defining which triples are valid with respect to a given q.

Valid Triples. A triple (A, f, g) is *potentially valid* if $T \subseteq A \subseteq V(H) \setminus (U^* \setminus T)$, f is a function such that $f(U) \in \mathsf{edges}(U)$ for all $U \in \mathcal{U}$, and g is a function such that $g(U) \in \{0, 1, \ldots, \mathsf{size}(U)\}$ for all $U \in \mathcal{U}$. We say that a set $U \in \mathcal{U}$ is *A-damaged*, *f-damaged* and *g-damaged* if $U \setminus A \neq \emptyset$, $f(U) \neq \{\{v, u\} : v, u \in U\}$ and $g(U) \neq \mathsf{size}(U)$, respectively. Roughly speaking, a set $U \in \mathcal{U}$ is A-damaged if we do not "take" all of its vertices, *f-damaged* if we do not "take" all edges between its vertices, and g-damaged if we do not "take" all of the vertices "below" U. Overall, a set $U \in \mathcal{U}$ is *damaged* if it is *A-damaged*, *f-damaged* or *g-damaged*. We will show below (in "Goal") that only few sets are damaged; the efficiency of our algorithm crucially relies on this observation, which essentially says that only few sets need "special attention". Moreover, we say a vertex $v \in V(H)$ is *damaged* if $v \in A$ and at least one of the two following conditions hold: (1) $v \in U$ for a damaged set U; (2) $v \in N_H(u)$ for a vertex $u \in V(H) \setminus A$. Finally, we define the *damage control* graph, $H(A, f)$, as the graph $H[A]$ to which we add the edges in $\bigcup_{U \in \mathcal{U}} f(U)$. Formally, $V(H(A, f)) = A$ and $E(H(A, f)) = E(H[A]) \cup (\bigcup_{U \in \mathcal{U}} f(U) \cap \{\{v, u\} : v, u \in A\})$. We say that a connected component is *huge* if it contains at least 2^{cp} vertices.

We are now ready to define which triples are valid with respect to a given $q \in \{\max\{0, k - \mathsf{size}(U^*) - |U^* \cap T|\}, \ldots, \min\{tSize - \mathsf{size}(U^*) - |U^*|, k - |U^* \cap T|\}\}$.

▶ **Definition 9.** A potentially valid (A, f, g) is *q-valid* if the following conditions hold.

1. For each damaged vertex v such that the connected component C of $H(A, f)$ containing v is not huge, at least one of the following conditions holds.
   a. $V(C) \cap (U^* \cap A) \neq \emptyset$.
   b. $|V(H)| \leq 2^{cp} + p$ and $H(A, f)$ is a connected graph.

2. If there exists a damaged vertex v such that the connected component C of $H(A, f)$ containing v is huge, then at least one of the following conditions holds.
   a. $H(A, f)$ has a huge connected component C' such that $V(C') \cap (U^* \cap A) \neq \emptyset$.
   b. $H(A, f)$ has no connected component that contains a damaged vertex and is not huge.
3. For all $\{v, u\} \in P^*$, either $H(A, f)$ has a connected component containing both v and u or each of its connected components containing v or u is huge.
4. $q = |A \setminus U^*| + \sum_{U \in \mathcal{U}} g(U)$.

Each q-valid triple is a potential solution for Single Bag. The first three conditions are related to the connectivity demand of a solution to MNH, and the fourth to the size demand. The *value* of a q-valid triple (A, f, g) is

$$W(A, f, g) = |E(A, V(H) \setminus A) \setminus E(U^* \cap A, U^* \setminus A)| + \sum_{U \in \mathcal{U}} \mathsf{cut}(U, A \cap U, f(U), g(U)).$$

Informally, $W(A, f, g)$ is the sum of the number of edges between vertices in A and vertices outside A (excluding edges with both endpoints in U^*), to which we add the sum of the values of previously computed solutions.

Goal. For each $q \in \{\max\{0, k - \mathsf{size}(U^*) - |U^* \cap T|\}, \ldots, \min\{tSize - \mathsf{size}(U^*) - |U^*|, k - |U^* \cap T|\}\}$, we need to compute the maximum value W_q for which there exists a q-valid triple (A, f, g) such that $W_q = W(A, f, g)$. In case W_q is not defined, since there does not exist a q-valid tuple, we set $W_q = -\infty$.

The following observation bounds the number of damaged sets in $\mathcal{U}$ and the number of damaged vertices corresponding to a q-valid triple. The main intuition is that if the current bag has several children then the size of the graph (which is less than $k + p$) forces a solution to select all vertices of all the subgraphs induced by the children, except p of them. This observation is used to allow us to apply the color coding method efficiently. Due to lack of space, the proof is omitted.

▶ **Observation 10.** *If (A, f, g) is a q-valid triple such that $W_q = W(A, f, g)$, then*
- *$|V(H) \setminus A| \leq p$ and (A, f, g) damages at most $2p^2$ sets in $\mathcal{U}$,*
- *(A, f, g) damages at most $2p^2 \cdot (2^{cp} + 1)$ vertices, and*
- *$g(U) \geq \mathsf{size}(U) - p$ for all $U \in \mathcal{U}$.*

5 Conclusion

In this paper, we initiated the study of multi-node hubs in general, and of algorithmic aspects of MNH in particular. The concept of multi-node hubs is a natural generalization of the concept of hubs, therefore it might play a central role in the analysis of complex networks. We showed that although MNH is W[1]-hard parameterized by k, it is FPT parameterized by p. Our algorithm is primarily a classification result and should be viewed as a proof of concept that the problem is FPT. While our approach may lead to an algorithm with running time $\mathcal{O}^*(2^{p^{\mathcal{O}(1)}})$ (or even $\mathcal{O}^*(2^{\mathcal{O}(p \log p)})$), developing an algorithm with running time $\mathcal{O}^*(2^{\mathcal{O}(p)})$ seems challenging. It may also be interesting to consider other natural restrictions on the set A such as properties expressible in monadic second order logic, or to study MNH with respect to parameters other than k and p.

To the best of our knowledge, besides our algorithm, only algorithms for Bisection and a problem called Judicious Partition [32] are based on the decomposition of [15]. Exploring the power of this decomposition more deeply might be fruitful; in this context, it is noteworthy to mention [31]. In our application, we showed how an instance of a maximization problem

can be preprocessed to a form that enables using known tools to solve cut-problems and in particular the decomposition of [15], as well as a manner in which connectivity constraints can be handled on top of cut and size constraints. We believe that our approach can be relevant to the design of algorithms for other cut-problems of this nature.

References

1 L A Adamic, R M Lukose, A R Puniyani, and B A Huberman. Search in power-law networks. *CoRR cs.NI/0103016*, 2001.

2 A A Ageev and M Sviridenko. Approximation algorithms formaximum coverage andmax cutwith given sizes of parts. In *IPCO*, pages 17–30, 1999.

3 N Alon, R Yuster, and U Zwick. Color coding. *J. ACM*, 42(4):844–856, 1995.

4 D Balcan, H Hu, B Goncalves, P Bajardi, C Poletto, J J Ramasco, D Paolotti, N Perra, M Tizzoni, W Van den Broeck, V Colizza, and A Vespignani. Seasonal transmission potential and activity peaks of the new influenza A(H1N1): a Monte Carlo likelihood analysis based on human mobility. *BMC Medicine*, 7(45):29, 2009.

5 A Barabasi and R Albert. Emergence of scaling in random networks. *Science*, 286:509, 1999.

6 M Berlingerio, M Coscia, F Giannotti, A Monreale, and D Pedreschi. The pursuit of hubbiness: Analysis of hubs in large multidimensional networks. *J. Comput. Science*, 2:223–237, 2011.

7 D Binkele-Raible. Amortized analysis of exponential time and parameterized algorithms: Measure and conquer and reference search trees. *PhD Thesis, University of Trier*, 2009.

8 E Bonnet, B Escoffier, V T Paschos, and E Tourniaire. Multi-parameter analysis for local graph partitioning problems: using greediness for parameterization. *Algorithmica*, 71(3):566–580, 2015.

9 L Cai. Parameter complexity of cardinality constrained optimization problems. *Comput. J.*, 51(1):102–121, 2008.

10 R H Chitnis, M Cygan, M T Hajiaghayi, M Pilipczuk, and M Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. In *FOCS*, pages 460–469, 2012.

11 R H Chitnis, M Cygan, M T Hajiaghayi, M Pilipczuk, and M Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. In *http://arxiv.org/pdf/1207.4079v1.pdf*, 2012.

12 R Crowston, G Gutin, M Jones, and G Muciaccia. Maximum balanced subgraph problem parameterized above lower bound. *Theor. Comput. Sci.*, 513:53–64, 2013.

13 R Crowston, M Jones, and M Mnich. Max-Cut Parameterized Above the Edwards-Erdős Bound. *Algorithmica*, 72(3):734–757, 2015.

14 M Cygan. Deterministic parameterized connected vertex cover. In *SWAT*, pages 95–106, 2012.

15 M Cygan, D Lokshtanov, M Pilipczuk, M Pilipczuk, and S Saurabh. Minimum bisection is fixed parameter tractable. In *STOC*, pages 323–332, 2014.

16 M Cygan, J Nederlof, M Pilipczuk, M Pilipczuk, J M M van Rooij, and J O Wojtaszczyk. Deterministic parameterized connected vertex cover. In *FOCS*, pages 150–159, 2012.

17 U Feige and M Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *J. Algorithms*, 41(2):174–211, 2001.

18 H Fernau and D Manlove. Vertex and edge covers with clustering properties: Complexity and algorithms. *J. Discrete Algorithms*, 7(2):149–167, 2009.

19 D W Franks, J Noble, P Kaufmann, and S Stagl. Extremism propagation in social networks with hubs. *Adaptive Behavior – Animals, Animats, Software Agents, Robots, Adaptive Systems*, 16:264–274, 2008.

20 M X Goemans and D P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

21 P Goymer. Network biology: Why do we need hubs? *Nature Reviews Genetics*, 9:650–651, 2008.

22 J Guo, R Niedermeier, and S Wernicke. Parameterized complexity of generalized vertex cover problems. In *WADS*, pages 36–48, 2005.

23 M T Hajiaghayi, G Kortsarz, R MacDavid, M Purohit, and K K Sarpatwar. Approximation algorithms for connected maximum cut and telated problems. In *ESA*, pages 693–704, 2015.

24 X He and J Zhang. Why Do Hubs Tend to Be Essential in Protein Networks? *PLOS Genetics*, 2(6):e88, 2006.

25 H Jeong, B Tombor, R Albert, Z N Oltvai, and A L Barabasi. The large-scale organization of metabolic networks. *Nature*, 407:651–654, 2000.

26 S Khot, G Kindler, E Mossel, and R O'Donnell. Optimization, approximation, and complexity classes. *SICOMP*, 37(1):319–357, 2007.

27 J M Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46:604–632, 1999.

28 J Kneis, A Langer, and P Rossmanith. A new algorithm for finding trees with many leaves. *Algorithmica*, 61:882–897, 2011.

29 J Lee, V Nagarajan, and X Shen. Max-Cut Under Graph Constraints. In *IPCO*, pages 50–62, 2016.

30 J Leskovec, L A Adamic, and Huberman B A. The dynamics of viral marketing. In *EC*, pages 228–237, 2006.

31 D Lokshtanov, M S Ramanujan, S Saurabh, and M Zehavi. Reducing CMSO Model Checking to Highly Connected Graphs. In *ICALP*, pages 135:1–135:14, 2018.

32 D Lokshtanov, S Saurabh, R Sharma, and M Zehavi. Balanced Judicious Bipartition is Fixed-Parameter Tractable. In *FSTTCS*, pages 40:40–40:15, 2017.

33 X Lu, V J Vipul, P W Finn, and D L Perkins. Hubs in biological interaction networks exhibit low changes in expression in experimental asthma. *Molecular Systems Biology*, 3(98), 2008.

34 M Mahajan and V Raman. Parameterizing above Guaranteed Values: MaxSat and MaxCut. *J. Algorithms*, 31(2):335–354, 1999.

35 A S Maiya and T Y Berger-Wolf. Online sampling of high centrality individuals in social networks. In *PAKDD*, pages 91–98, 2010.

36 D Molle, S Richter, and P Rossmanith. Enumerate and expand: Improved algorithms for connected vertex cover and tree cover. *Theory Comput. Syst.*, 43(2):234–253, 2008.

37 C H Papadimitriou and M Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991.

38 V Raman and S Saurabh. Improved fixed parameter tractable algorithms for two "edge" problems: MAXCUT and MAXDAG. *Inf. Process. Lett.*, 104(2):65–72, 2007.

39 S Saurabh and M Zehavi. $(k, n-k)$-Max-Cut: an $O^*(2^p)$-time algorithm and a polynomial kernel. In *LATIN*, pages 686–699, 2016.

40 H Shachnai and M Zehavi. Parameterized algorithms for graph partitioning problems. In *WG*, pages 384–395, 2014.

41 X Shi, B Tseng, and L Adamic. Looking at the Blogosphere Topology through Different Lenses. In *ICWSM*, 2007.

42 S Vicente, V Kolmogorov, and C Rother. Graph cut based image segmentation with connectivity priors. In *CVPR*, pages 1–8, 2008.