A Generalization of Nemhauser and Trotter's Local Optimization Theorem[☆]

Michael R. Fellows[a,1], Jiong Guo[b,*,2], Hannes Moser[c,3], Rolf Niedermeier[d]

[a] *School of Engineering and Information Technology, Charles Darwin University, Darwin, Northern Territory 0909, Australia.*
[b] *Universität des Saarlandes, Campus E 1.7, D-66123 Saarbrücken, Germany.*
[c] *Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany.*
[d] *Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, D-10587 Berlin, Germany.*

Abstract

The Nemhauser-Trotter local optimization theorem applies to the NP-hard Vertex Cover problem and has applications in approximation as well as parameterized algorithmics. We generalize Nemhauser and Trotter's result to vertex deletion problems, introducing a novel algorithmic strategy based on purely combinatorial arguments (not referring to linear programming as the Nemhauser-Trotter result originally did). The essence of our strategy can be understood as a doubly iterative process of cutting away "easy parts" of the input instance, finally leaving a "hard core" whose size is (almost) linearly related to the cardinality of the solution set.

We exhibit our approach using a generalization of Vertex Cover, called Bounded-Degree Vertex Deletion. For some fixed $d \geq 0$, Bounded-Degree Vertex Deletion asks to delete at most k vertices from a graph in order to transform it into a graph with maximum vertex degree at most d. Vertex Cover is the special case of $d = 0$. Our generalization of the Nemhauser-Trotter theorem implies that Bounded-Degree Vertex Deletion, parameterized by k, admits an $O(k)$-vertex problem kernel for $d \leq 1$ and, for any $\epsilon > 0$, an $O(k^{1+\epsilon})$-vertex problem kernel for $d \geq 2$. Finally, we provide a W[2]-completeness result for Bounded-Degree Vertex Deletion in case of unbounded d-values.

Key words: Parameterized computational complexity, NP-hard problems, W[2]-completeness, graph algorithms, polynomial-time data reduction, kernelization

[☆] A preliminary version of this paper appeared in the proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science (STACS '09), held in Freiburg, Germany, February 26–28 [22]. Note that the preliminary version contained a flaw in the statement of the main result, which has been corrected in this paper. The main parts of the work were done while all authors were staying at the Friedrich-Schiller-Universität Jena.

[*] Corresponding author, Tel: +49 681 302-70792, Fax: +49 681 9325-199

Email addresses: michael.fellows@cdu.edu.au (Michael R. Fellows), jguo@mmci.uni-saarland.de (Jiong Guo), hannes.moser@uni-jena.de (Hannes Moser), rolf.niedermeier@tu-berlin.de (Rolf Niedermeier)

[1] Supported by the Australian Research Council. Work done while staying in Jena as a recipient of the Humboldt Research Award of the Alexander von Humboldt Foundation, Bonn, Germany.

[2] Supported by the DFG, Emmy Noether research group PIAF, NI 369/4, project DARE, GU 1023/1, and the DFG cluster of excellence "Multimodal Computing and Interaction".

[3] Supported by the DFG, projects ITKO, NI 369/5, and AREG, NI 369/9.

1. Introduction

Nemhauser and Trotter [36] proved a famous theorem in combinatorial optimization. In terms of the NP-hard Vertex Cover problem in graphs, where the task is to find a minimum-cardinality subset of vertices such that each edge has at least one endpoint in this subset, it can be formulated as follows:

NT-Theorem [36, 5]. *For an undirected graph $G = (V, E)$, one can compute in polynomial time two disjoint vertex subsets A and B such that the following three properties hold:*

1. *If S' is a vertex cover of the induced subgraph $G[V \setminus (A \cup B)]$, then $A \cup S'$ is a vertex cover of G.*
2. *There is a minimum-cardinality vertex cover S of G with $A \subseteq S$.*
3. *For every vertex cover S'' of the induced subgraph $G[V \setminus (A \cup B)]$,*

$$|S''| \geq \frac{|V \setminus (A \cup B)|}{2}.$$

In other words, the NT-Theorem provides a polynomial-time data reduction for Vertex Cover. That is, for vertices in A it can already be decided in polynomial time to put them into the solution set and vertices in B can be ignored when finding a solution. Hochbaum [26] first explained that the NT-Theorem is very useful for approximating Vertex Cover. The point is that the search for an approximate solution can be restricted to the induced subgraph $G[V \setminus (A \cup B)]$. The NT-Theorem directly delivers a factor-2 approximation for Vertex Cover by choosing $V \setminus B$ as the vertex cover. Chen et al. [10] first observed that the NT-Theorem directly yields a $2k$-vertex problem kernel for Vertex Cover, where the parameter k denotes the size of the solution set. Indeed, this is in a sense an "ultimate" kernelization result in parameterized complexity analysis [20, 23, 37] because there is good reason to believe that there is a matching lower bound of $2k$ vertices for the kernel assuming the unique games conjecture [28]. Moreover, Dell and van Melkebeek [17] recently showed that Vertex Cover has no $O(k^{2-\epsilon})$-edge kernel for $\epsilon > 0$ unless the polynomial hierarchy collapses to the third level. This also generalizes to Bounded-Degree Vertex Deletion.

Since its publication numerous authors have referred to the importance of the NT-Theorem from the viewpoint of polynomial-time approximation algorithms (see, e.g., [5, 27, 29]) as well as from the viewpoint of parameterized algorithmics (e.g., [2, 10, 13, 24]). The relevance of the NT-Theorem comes from both its practical usefulness in solving Vertex Cover [1] as well as its theoretical depth having led to numerous further studies and follow-up work [2, 5, 6, 13]. In this work, our main contribution is to provide a more general version of the NT-Theorem. The corresponding algorithmic strategies and proof techniques, however, are not achieved by a generalization of known proofs of the NT-Theorem but are based on extremal combinatorial arguments. Our main result is to prove a generalization of the NT-Theorem that helps in finding a minimum-cardinality set of vertices whose deletion leaves a graph of maximum degree d for arbitrary but fixed d. Clearly, $d = 0$ is the special case of Vertex Cover.

Motivation. As the NP-hard Bounded-Degree Vertex Deletion problem—given a graph and two positive integers k and d, find at most k vertices whose deletion leaves a graph of maximum vertex degree d—stands in the center of our considerations, some more explanations about its

relevance follow. Bounded-Degree Vertex Deletion (or its dual problem) already appears in some theoretical work [8, 9, 15, 30, 38, 39], but so far it has received considerably less attention than Vertex Cover, one of the best studied problems in combinatorial optimization [29]. To advocate and justify more research on Bounded-Degree Vertex Deletion (also see [34] for a more thorough discussion), we describe an application in computational biology. In the analysis of genetic networks based on micro-array data, recently a clique-centric approach has shown great success [4, 12]. Roughly speaking, finding cliques (that is, fully connected subgraphs) or near-cliques (called paracliques [12]) has been a central tool. Since finding cliques is computationally hard (also with respect to approximation), Chesler et al. [12, page 241] state that "cliques are identified through a transformation to the complementary dual Vertex Cover problem and the use of highly parallel algorithms based on the notion of fixed-parameter tractability." More specifically, in these Vertex Cover-based algorithms polynomial-time data reduction (such as the NT-Theorem) plays a decisive role [31] (also see [2]) for efficient solvability of the given real-world data. However, since biological and other real-world data typically contain errors, the demand for finding cliques (that is, fully connected subgraphs) often seems overly restrictive and somewhat relaxed notations of cliques are more appropriate. Chesler et al. [12] introduced paracliques, which are achieved by greedily extending the found cliques by vertices that are connected to almost all (para)clique vertices. An elegant mathematical concept of "relaxed cliques" is that of s-plexes where one demands that each s-plex vertex does not need to be connected to all other vertices in the s-plex but to all but $s-1$. Thus, cliques are 1-plexes. The s-plex concept was introduced in 1978 by Seidman and Foster [41] in the context of social network analysis. Recently, this concept has received considerable attention in various fields, see, e.g., [3, 16, 25, 33, 35]. The corresponding problem to find maximum-cardinality s-plexes in a graph is basically as computationally hard as clique detection is [3, 16, 30]. However, as Vertex Cover is the dual problem for clique detection, Bounded-Degree Vertex Deletion is the dual problem for s-plex detection: An n-vertex graph has an s-plex of size k if and only if its complement graph has a solution set for Bounded-Degree Vertex Deletion with $d = s-1$ of size $n-k$, and the solution sets can be directly computed from each other.

Our Results. A *bdd-d-set* for a graph $G = (V, E)$ is a vertex subset whose removal from G yields a graph in which each vertex has degree at most d. Our main theorem can be formulated as follows.

Theorem 1 (BDD-DR-Theorem). *For an undirected graph $G = (V, E)$, $|V| = n$, $|E| = m$ and for any constant $\epsilon > 0$, one can compute in $O(n^4 \cdot m)$ time two disjoint vertex subsets A and B such that the following three properties hold (the first two properties are referred as the* local optimality conditions *in the following and the third one is called the* size condition*):*

1. *If S' is a bdd-d-set of the induced subgraph $G[V \setminus (A \cup B)]$, then $A \cup S'$ is a bdd-d-set of G.*
2. *There is a minimum-cardinality bdd-d-set S of G with $A \subseteq S$.*
3. *For every bdd-d-set S'' of the induced subgraph $G[V \setminus (A \cup B)]$, for $d \leq 1$,*

$$|S''| \geq \frac{|V \setminus (A \cup B)|}{d^3 + 4d^2 + 6d + 4}$$

and for $d \geq 2$,

$$|S''|^{1+\epsilon} \geq \frac{|V \setminus (A \cup B)|}{c}$$

for some constant c depending on d and ϵ.[4]

As a direct application of Theorem 1 we obtain a problem kernel (by simply removing $A \cup B$ from the graph and setting $k := k - |A|$):

Corollary 1. *For $d \leq 1$,* Bounded-Degree Vertex Deletion *admits a problem kernel of at most $(d^3+4d^2+6d+4) \cdot k$ vertices, and for constant $d \geq 2$,* Bounded-Degree Vertex Deletion *admits a problem kernel of $O(k^{1+\epsilon})$ vertices for any constant $\epsilon > 0$. Both problem kernels can be computed in $O(n^4 \cdot m)$ time.*

There is a significant difference between Theorem 1 and the NT-theorem for Vertex Cover: for $d \geq 2$, with our approach we can only get arbitrarily close to a linear dependence of the minimum solution size on the number of vertices in $|V \setminus (A \cup B)|$. In terms of parameterized algorithmics, this yields a linear problem kernel for Bounded-Degree Vertex Deletion for $d \leq 1$, and an *almost* linear problem kernel for $d \geq 2$, that is, we can show that there exists a problem kernel of $O(k^{1+\epsilon})$ vertices for any constant $\epsilon > 0$. Very recently, Chen et al. [11] presented a $37k$-vertex problem kernel for Bounded-Degree Vertex Deletion in the special case of $d = 2$. Our general result specializes to a $4k$-vertex problem kernel for Vertex Cover (the NT-Theorem provides a $2k$ problem kernel), but applies to a larger class of problems. For instance, a slightly modified version of the BDD-DR-Theorem (with essentially the same proof) yields a $15k$ problem kernel for the problem of packing at least k vertex-disjoint length-2 paths of an input graph, giving the same bound as shown in work focussing on this problem [40].[5] We emphasize that our data reduction technique is based on extremal combinatorial arguments; the resulting combinatorial kernelization algorithm has practical potential [35]. Note that for $d = 0$ our algorithm computes the same type of structure as in the "crown decomposition" kernelization for Vertex Cover (see, for example, [1, 2]). However, for $d \geq 1$ the structure returned by our algorithm is much more complicated; in particular, unlike Vertex Cover crown decompositions, in the BDD-DR-Theorem the set A is not necessarily a separator and the set B does not necessarily form an independent set.

Exploring the borders of parameterized tractability of Bounded-Degree Vertex Deletion for arbitrary values of the degree value d, we also show in Section 4 that the problem becomes W[2]-complete with respect to the parameter solution size (that is, the number of vertices to delete) for d being unbounded. In other words, there is no hope for fixed-parameter tractability with respect to the parameter k in the case of unbounded d-values.

2. Preliminaries

In this paper, all graphs are simple and undirected. The central problem of this paper is defined as follows.

Bounded-Degree Vertex Deletion
Input: An undirected graph $G = (V, E)$, and integers $d \geq 0$ and $k \geq 0$.
Question: Does there exist a bdd-d-set $S \subseteq V$ of size at most k for G?

[4] In the conference version [22] of this work, we erroneously claimed $|S''| \geq |V \setminus (A \cup B)|/c$ for some constant c for every $d \geq 0$.

[5] Recently, Wang et al. [42] improved the $15k$-bound to a $7k$-bound. Our kernelization based on the BDD-DR-Theorem method can be adapted to also deliver the $7k$-bound.

In this paper, for a graph $G = (V, E)$ and a vertex set $S \subseteq V$, let $G[S]$ be the subgraph of G induced by S and $G - S := G[V \setminus S]$. The *open neighborhood* of a vertex v or a vertex set $S \subseteq V$ in a graph $G = (V, E)$ is denoted as $N_G(v) := \{u \in V \mid \{u, v\} \in E\}$ and $N_G(S) := \bigcup_{v \in S} N_G(v) \setminus S$, respectively. The *closed neighborhood* is denoted as $N_G[v] := N_G(v) \cup \{v\}$ and $N_G[S] := N_G(S) \cup S$. We write $V(G)$ and $E(G)$ to denote the vertex and edge set of G, respectively. For $s \geq 1$, the graph $K_{1,s} := (\{u, v_1, \ldots, v_s\}, \{\{u, v_1\}, \ldots, \{u, v_s\}\})$ is an *s-star*, or simply *star*. The vertex u is the *center* of the star and the vertices $v_1, \ldots, v_s$ are the *leaves* of the star. A *$\leq s$-star* is an s'-star with $s' \leq s$ and a *$<s$-star* is an s'-star with $s' < s$. A *packing* P of a graph G is a set of pairwise vertex-disjoint subgraphs of G. A graph has maximum degree d when every vertex in the graph has degree at most d. A graph property is called *hereditary* if every induced subgraph of a graph with this property has the property as well.

Parameterized algorithmics [20, 23, 37] is an approach to finding optimal solutions for NP-hard problems. The idea is to accept the seemingly inevitable combinatorial explosion, but to confine it to one aspect of the problem, the *parameter*. More precisely, a problem is *fixed-parameter tractable* (FPT) with respect to a parameter k if there is an algorithm solving any problem instance of size n in $f(k) \cdot n^{O(1)}$ time for some computable function f. A common method in parameterized algorithmics is to provide polynomial-time executable *data reduction rules* that lead to a *problem kernel* [7, 24]. Given a parameterized problem instance (I, k), a data reduction rule replaces (I, k) by an instance (I', k') in polynomial time such that $|I'| \leq |I|$, $k' \leq k$, and (I, k) is a yes-instance if and only if (I', k') is a yes-instance. A parameterized problem is said to have a problem kernel, or, equivalently, *kernelization*, if, after the exhaustive application of the data reduction rules, the resulting reduced instance has size $g(k)$ for a function g depending only on k. Roughly speaking, the kernel size $g(k)$ plays a similar role in the subject of problem kernelization as the approximation factor plays for approximation algorithms. Analogously to classical complexity theory, Downey and Fellows [20] developed a framework providing reducibility and completeness notions. A *parameterized reduction* reduces a problem instance (I, k) in $f(k) \cdot n^{O(1)}$ time to an instance (I', k') (with k' depending only on k) such that (I, k) is a yes-instance if and only if (I', k') is a yes-instance. The first two levels of (presumable) parameterized intractability are W[1] and W[2]. We show a W[2]-completeness result for Bounded-Degree Vertex Deletion with unbounded d; it is commonly believed that W[2]-complete problems are not fixed-parameter tractable.

3. A Local Optimization Algorithm

In this section, we prove Theorem 1. Recall that the first two properties of Theorem 1 are called the local optimality conditions, since they guarantee that we can "locally" decide to take all vertices in A into an optimal solution set and vertices in B can be ignored when finding a solution. The third property of Theorem 1 is the size condition.

We begin to describe the main algorithm that computes two sets A and B as claimed in Theorem 1.

3.1. The Main Algorithm

The first step to prove Theorem 1 is to greedily compute a factor-$(d + 2)$ approximate bdd-d-set X for G. To this end, we use the following easy-to-verify forbidden subgraph characterization of bounded-degree graphs: A graph G has maximum degree d if and only if there is no $(d+1)$-star (a star with $d+1$ leaves) that is a subgraph of G. With a straightforward greedy algorithm, compute a *maximal* $(d + 1)$*-star packing* of G, that is, a set of vertex-disjoint $(d + 1)$-stars that cannot be

extended by adding another $(d+1)$-star. Let X be the set of vertices of this star packing. Since the number of stars in the packing is a lower bound for the size of a minimum bdd-d-set, X is a factor-$(d+2)$ approximate bdd-d-set. Greedily remove vertices from X such that X is still a bdd-d-set, and finally set $Y := V \setminus X$. These two vertex sets X and Y are the starting point for the search for the two vertex subsets A and B that fulfill the properties in Theorem 1; as we will see, we can restrict A to be a subset of X and B to be a subset of Y.

Since X is a factor-$(d+2)$ approximate bdd-d-set, every bdd-d-set S'' contains at least $|X|/(d+2)$ vertices, that is, $|S''| \geq |X|/(d+2)$. Thus, $|X| \leq |S''| \cdot (d+2)$. Roughly speaking, this shows that the size condition (third property) of Theorem 1 is fulfilled by choosing $A := \emptyset$ and $B := Y = V \setminus X$. However, this choice of A and B will in general not guarantee that the first two properties of Theorem 1 are fulfilled. To fulfill the first two properties, only a subset of Y can be chosen to be contained in B, but this subset should be as large as possible in order to fulfill the size condition, that is, one has to bound the size of $Y \setminus B$ with respect to $|X|$. The most important lemma, whose proof is deferred to the next subsections, shows that if $Y = V \setminus X$ is too big compared to X, then one can find two vertex sets $A' \subseteq X$ and $B' \subseteq Y$ that fulfill the local optimality conditions such that B' is not empty.

Lemma 1. *Let $G = (V, E)$ be an undirected graph with a bdd-d-set X and let $n := |V|$ and $m := |E|$. If $Y = V \setminus X$ contains more than $(d+1)^2 \cdot |X|$ vertices for $d \leq 1$ or more than $O(|X|^{1+\epsilon})$ vertices for $d \geq 2$, then one can find in $O(n^3 m)$ time two vertex subsets $A' \subseteq X$ and $B' \subseteq Y$ such that the following three properties hold:*

1. *If S' is a bdd-d-set of the induced subgraph $G - (A' \cup B')$, then $A' \cup S'$ is a bdd-d-set of G.*
2. *There is a minimum-cardinality bdd-d-set S of G with $A' \subseteq S$.*
3. *The subset B' is not empty.*

Note that the first two properties, that is, the local optimality conditions, are the same as the first two properties in Theorem 1. As the third property in Theorem 1, we also call the third property in Lemma 1 the *size condition*. The reason is as follows. The main algorithm iteratively applies the algorithm behind Lemma 1 and removes A' and B' from G and recomputes X, until the preconditions of Lemma 1 are not fulfilled anymore. The union of all A''s and B''s, respectively, then forms the sets A and B with the properties that are stated in Theorem 1. Since B' is never empty, we have a guarantee that B "grows bigger" in each iteration, which will eventually bound the size of $Y \setminus B$, and this bound will almost directly imply the size condition of Theorem 1.

In the following, we show the correctness of this approach. Let FindExtremal[6] be an algorithm that finds two subsets A' and B' as stated in Lemma 1. Figure 1 shows the pseudo-code of the main algorithm that will be used to show Theorem 1. The algorithm starts initializing A and B with empty sets in line 1, and then it computes a factor-$(d+2)$ approximate bdd-d-set X in line 2 and the remaining vertices Y in line 3. If the set Y is small compared to X (conditions in line 5 or line 7), then (A, B) is returned. If the set Y is too big compared to X (that is, the conditions in line 5 or line 7 are not fulfilled), then, in line 8, the graph G and the vertex set X are passed to the procedure FindExtremal, which computes two sets A' and B' satisfying the properties

[6]The name "FindExtremal" comes from extremal combinatorics arguments that we use in the proof. Such arguments are often used for parameterized algorithms and problem kernelization, see, e.g., [21].

Algorithm: ComputeAB (G)
Input: An undirected graph $G = (V, E)$.
Output: Vertex subsets A and B satisfying the three properties of Theorem 1.

```
1  A ← ∅, B ← ∅
2  Compute a (d + 2)-factor approximate bdd-d-set X for G.
3  Y ← V \ X
4  if d ≤ 1 then
5        if |Y| ≤ (d + 1)^2 · |X| then return (A, B)
6  if d ≥ 2 then
7        if |Y| ≤ c' · |X|^(1+ε) then return (A, B)
8  (A', B') ← FindExtremal (G, X).
9  G ← G − (A' ∪ B')
10 A ← A ∪ A'
11 B ← B ∪ B';
12 goto line 2
```

Figure 1: Pseudo-code of the main algorithm for computing A and B. The exact value of the constant c', which is depending on d and ϵ, is determined later in the proof of Proposition 4. The pseudo-code of FindExtremal is given in Figure 5.

in Lemma 1. The sets A' and B' are then added to A and B in lines 10 and 11, respectively. Finally, in line 12 the algorithm starts over and computes a factor-$(d + 2)$ approximate solution for the new graph G in line 2. Since B' is never empty due to Lemma 1, the conditions in line 5 or line 7 will eventually be fulfilled (because in each iteration at least one vertex is added to B, thus eventually $Y \setminus B$ will be "small"), and the algorithm returns the vertex subsets A and B. It remains to show that A and B fulfill the three properties of Theorem 1, and that the running time of ComputeAB is $O(n^4 \cdot m)$.

Lemma 2. *The sets A and B computed by* ComputeAB *fulfill the three properties given in Theorem 1.*

Proof. First, we prove that A and B fulfill the first two properties of Theorem 1. The proof is by a simple inductive argument: assume that in some iteration of ComputeAB the two vertex subsets A and B of a graph G fulfill the first two properties (local optimality conditions) of Theorem 1 with respect to G (call this assumption a), and that FindExtremal returns in line 8 two vertex subsets A' and B' of $G' := G - (A \cup B)$ that fulfill the first two properties (local optimality conditions) of Lemma 1 with respect to the graph G' (call this assumption b). We show that then $A \cup A'$ and $B \cup B'$ fulfill the first two properties of Theorem 1 with respect to G as well:

1. If S' is a bdd-d-set of the induced subgraph $G' - (A' \cup B') = G - (A \cup A' \cup B \cup B')$, then $A' \cup S'$ is a bdd-d-set of G' (due to assumption b), and therefore $A \cup A' \cup S'$ is a bdd-d-set of G (due to assumption a). This shows the first property.

2. There is a minimum-cardinality bdd-d-set S of G with $A \subseteq S$ (due to assumption a). Since the graph property "bounded degree d" is hereditary, $S \setminus (A \cup B)$ is a bdd-d-set for $G' = G - (A \cup B)$.

There is a minimum-cardinality bdd-d-set S' of G' with $A' \subseteq S'$ (due to assumption b). Since S' has minimum cardinality, $|S'| \leq |S \setminus (A \cup B)|$. The set $S' \cup A$ is a bdd-d-set of G (due to assumption a), and because $A \subseteq S$ we know that $|S' \cup A| \leq |S|$. Since S has minimum cardinality, $S' \cup A$ has minimum-cardinality, and thus $S' \cup A$ is a minimum-cardinality bdd-d-set that contains $A' \cup A$. This shows the second property.

The sets $A = \emptyset$ and $B = \emptyset$ (line 1) trivially fulfill the first two properties of Theorem 1, and by the above inductive argument the sets A and B returned by ComputeAB fulfill these properties as well.

It remains to show that the sets A and B fulfill the third property.

3. Let $V' := V \setminus (A \cup B)$. Clearly $V' = X \cup Y$ (line 3). Since the condition in line 5 (for $d \leq 1$) or line 7 (for $d \geq 2$) is true, we know that either $|Y| \leq (d+1)^2 \cdot |X|$ and therefore

$$|V'| = |X| + |Y| \leq (1 + (d+1)^2) \cdot |X| \quad \text{(for } d \leq 1\text{), or}$$

$|Y| = O(|X|^{1+\epsilon})$ and therefore

$$|V'| = |X| + |Y| = O(|X|^{1+\epsilon}) \quad \text{(for } d \geq 2\text{).}$$

Recall that X is a factor-$(d+2)$ approximate bdd-d-set for $G' := G - (A \cup B)$. Thus, $|X| \leq (d+2) \cdot |S|$ for an arbitrary bdd-d-set S.

For $d \leq 1$, one obtains $|V'| \leq (1 + (d+1)^2) \cdot |X| \leq (1 + (d+1)^2)(d+2) \cdot |S|$ and therefore

$$|S| \geq \frac{|V'|}{(1 + (d+1)^2)(d+2)} = \frac{|V'|}{d^3 + 4d^2 + 6d + 4}.$$

For $d \geq 2$, one obtains $|V'| = O(|X|^{1+\epsilon}) = O(|S|^{1+\epsilon})$ and therefore

$$|S|^{1+\epsilon} \geq \frac{|V'|}{c}$$

for some constant c. This shows the third property. □

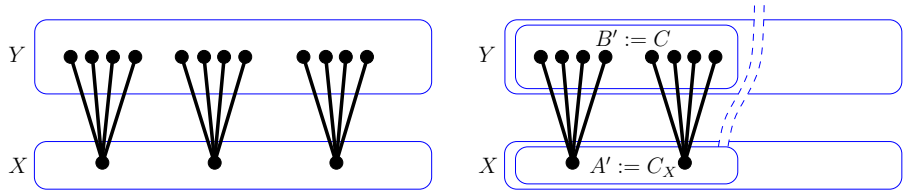
Next, we show the running time of ComputeAB.

Lemma 3. *Algorithm* ComputeAB *runs in* $O(n^4 \cdot m)$ *time.*

Proof. With the described simple greedy approach, computing a factor-$(d+2)$ approximate solution in line 2 takes $O(n+m)$ time. Each call of FindExtremal in line 8 takes $O(n^3 \cdot m)$ time. FindExtremal always returns two sets A' and B' such that B' is not empty (Lemma 1), hence after at most n iterations of ComputeAB, Y must be small compared to X and ComputeAB returns in line 5 (for $d \leq 1$) or line 7 (for $d \geq 2$). Thus, in total, we find the sets A and B in $O(n^4 \cdot m)$ time. □

With Lemmas 2 and 3, the proof of Theorem 1 is completed.

The remaining part of this section is dedicated to the proof of Lemma 1 by providing a description of the algorithm FindExtremal. The description is divided into an outline (Subsection 3.2), the description of a method to show the local optimization conditions (Subsection 3.3), the description of an important subroutine (Subsection 3.4), and finally a description of FindExtremal together with the correctness proofs (Subsection 3.5).



(a) Each vertex in X is the center of a star with four leaves in Y.

(b) Each vertex in A' is the center of a star with four leaves in $B' \subseteq Y$. The dashed lines illustrate that there are no edges between B' and the rest of the graph in $G - A'$.

Figure 2: Some simple cases for FindExtremal, assuming $d = 3$.

3.2. The Ingredients of FindExtremal

We prove Lemma 1 by describing an algorithm called FindExtremal that, given an undirected graph G and a bdd-d-set X such that $Y := V \setminus X$ is "large" compared to X, finds two subsets $A' \subseteq X$ and $B' \subseteq Y$ such that they fulfill the local optimality conditions and such that B' is not empty (see Lemma 1). We first focus on the local optimality conditions:

1. If S' is a bdd-d-set of the induced subgraph $G - (A' \cup B')$, then $A' \cup S'$ is a bdd-d-set of G.
2. There is a minimum-cardinality bdd-d-set S of G with $A' \subseteq S$.

Informally speaking, these two properties guarantee that one can always assume that there exists a minimum-cardinality bdd-d-set that contains all vertices in A' and no vertex in B'. We use this informal interpretation for the following step-by-step explanation of the main obstacles that FindExtremal has to bypass.

How to Fulfill the Local Optimality Conditions. The fundamental idea to show the local optimality conditions is to use the forbidden subgraph characterization of bounded-degree graphs: a graph G has maximum degree d if and only if there is no $(d+1)$-star (a star with $d+1$ leaves) that is a subgraph of G. To illustrate the idea, let us first assume that there is a packing of vertex-disjoint $(d+1)$-stars in G such that each vertex in the bdd-d-set X is the center of such a star (thus, all leaves are in Y). Hence, each vertex in X is "covered" by the star packing. See Figure 2a for an example. Then, due to the forbidden subgraph characterization, a minimum-cardinality bdd-d-set has to contain at least one vertex of each star, thus a minimum-cardinality bdd-d-set contains at least $|X|$ vertices, and X (that is, the set of all centers) is therefore a minimum-cardinality bdd-d-set of G. Thus, for $A' := X$ and $B' := Y$ there exists a minimum-cardinality bdd-d-set that contains all vertices in A' and no vertex in B'.

Obviously, in general there might not exist a vertex-disjoint packing of $(d+1)$-stars whose centers cover all vertices in X; rather, it can happen that one is only able to find a subset C_X of X whose vertices are centers of $(d+1)$-stars with leaves in Y. Now suppose that the subset $C_X \subseteq X$ is a separator in G such that the leaves of these $(d+1)$-stars are contained in a component C that is "separated" from the rest of the graph, that is, every path from C to a vertex neither in C nor in C_X passes through C_X. See Figure 2b for an example. Then, due to the forbidden subgraph characterization, a minimum-cardinality bdd-d-set has to contain at least one vertex for each $(d+1)$-star with center in C_X, and taking all vertices in C_X into a solution is always optimal,

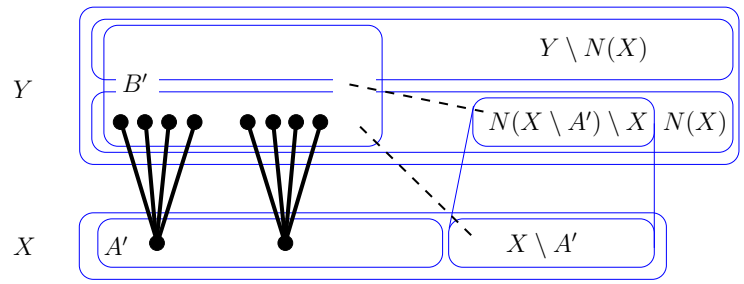


Figure 3: Illustration of the structure of the graph with bdd-d-set X (assuming $d = 3$), its neighborhood $N(X)$, and all remaining vertices $Y \setminus N(X)$. The sets A' and B' fulfill the A'-star cover property and the restricted neighborhood properties, because there is a $(d+1)$-star with center in A' and four leaves in B' for each vertex in A' and there are no edges between B' and $X \setminus A'$ and no edges between B' and $N(X \setminus A') \setminus X$ (illustrated by dashed lines).

since each vertex in C has degree at most d in $G - C_X$. Thus, for $A' := C_X$ and $B' := V(C)$ there exists a minimum-cardinality bdd-d-set that contains all vertices in A' and no vertex in B'. If C_X is not a separator as described, then this approach does not work directly; however, as we will see, it is not necessary that A' is a separator that completely separates B' from the rest of the graph; A' and B' only have to fulfill the following three properties:

A'-star cover property: There exists a packing of vertex-disjoint stars in $G[A' \cup B']$, each star having at least $d+1$ leaves, such that each vertex in A' is the center of such a star.

Restricted X-neighborhood property: There are no edges between B' and $X \setminus A'$.

Restricted Y-neighborhood property: There are no edges between B' and $N(X \setminus A') \setminus X$.

See Figure 3 for an illustration. Note the difference from the case illustrated in Figure 2b: with these three properties, the set A' is not necessarily a separator. Intuitively, the A'-star cover property is needed to prove that there exists an optimal bdd-d-set containing A', the restricted X-neighborhood property is needed to avoid that a vertex in B' has degree more than d in $G - A'$, and the restricted Y-neighborhood property is needed to avoid that a *neighbor* of a vertex in B' has degree more than d in $G - A'$ (because, since X is a bdd-d-set of G, the only vertices of degree more than d in $G - A'$ can be in $X \setminus A'$ or in $N(X \setminus A') \setminus X$). Roughly speaking, then, similarly to the ideas outlined above, it is always optimal to take all vertices in A' into the solution, and the vertices in B' do not have to be considered for an optimal solution, since they and their neighbors have degree at most d in $G - A'$; the formal correctness proof is given in Subsection 3.3. With the A'-star cover property and the restricted neighborhood properties we are now ready to sketch how FindExtremal works.

FindExtremal in a Nutshell. The algorithm FindExtremal guarantees the A'-star cover property and the restricted X-neighborhood property as follows. FindExtremal computes a packing of $(d+1)$-stars between X and Y such that the centers C_X of the stars are in X and the leaves in Y, and such that the leaves of the stars are not adjacent to vertices in $X \setminus C_X$. This is accomplished by a procedure called StarPacking based on maximum flow techniques, which is described in detail in Subsection 3.4. Roughly speaking, by setting $A' := C_X$ and by choosing B' such that it contains

all leaves of the $(d+1)$-stars, A' and B' fulfill the A'-star cover and the restricted X-neighborhood property. The more difficult part is to fulfill the restricted Y-neighborhood property. There might be edges between leaves of the $(d+1)$-stars and vertices in $N(X \setminus A') \setminus X$. If there are no such edges, then the A'-star cover and restricted neighborhood properties are fulfilled. If there are such edges, then the trick is to "forbid" the vertices in $X \setminus A'$, $N(X \setminus A') \setminus X$, and their neighbors in Y (that is, all vertices in Y within distance at most two in $G - A'$ to a vertex in $X \setminus A'$) from being used for recomputing the star packing. These forbidden vertices will contain some leaves of the packing, thus the star packing between X and Y is recomputed, but excluding the forbidden vertices. The point is, as we will see, that if the recomputed packing can be used to construct A' and B' fulfilling the A'-star cover property and the two restricted neighborhood properties in the subgraph excluding the forbidden vertices, then they also fulfill these properties in G. This approach of computing a packing and forbidding vertices is then iterated until the algorithm finds two vertex subsets A' and B' fulfilling the local optimality conditions. In summary, FindExtremal works roughly as follows:

1. Call StarPacking to compute a packing of $(d+1)$-stars (in order to fulfill the A'-star cover property and the restricted X-neighborhood property), excluding forbidden vertices.
2. If the restricted Y-neighborhood property can be fulfilled, then construct A' and B' and return.
3. Forbid vertices that prevent the restricted Y-neighborhood property from being fulfilled.
4. Goto 1.

Of course it has to be shown in detail that this approach always terminates and returns a correct solution. Moreover, it still remains to consider the size condition of Lemma 1.

How to Fulfill the Size Condition. Next, consider the size condition of Lemma 1:

> The subset B' is not empty.

Recall the preconditions of Lemma 1: FindExtremal is only called if $Y := V \setminus X$ contains more than $(d+1)^2 \cdot |X|$ vertices for $d \leq 1$ or more than $O(|X|^{1+\epsilon})$ vertices for $d \geq 2$. The problem is that in the iterative process of computing a star packing and forbidding vertices, all vertices in X and Y might become forbidden, and hence FindExtremal would not be able to return a non-empty vertex subset B'. However, one can show that not too many vertices become forbidden in this process, and that there will be always some vertices in B' left. To make this possible, it is necessary that the StarPacking procedure finds a star packing such that the set C_X of centers of $(d+1)$-stars is "as large as possible" and that the remaining vertices in $X \setminus C_X$ have only few neighbors in Y. Then, roughly speaking, since $X \setminus C_X$ contains few vertices, there are only few neighbors in $N(X \setminus C_X) \setminus X$, and since X is a bdd-d-set, each vertex in $N(X \setminus C_X) \setminus X$ has only d neighbors in Y. Hence, there are only few forbidden vertices, and summing up the number of all forbidden vertices over all iterations will show that there can be only $(d+1)^2 \cdot |X|$ forbidden vertices in Y for $d \leq 1$ or $O(|X|^{1+\epsilon})$ forbidden vertices for $d \geq 2$.

Remarks. Note that the above description of FindExtremal is somewhat simplified; for the case $d \leq 1$ it works as described, but for the case $d \geq 2$ we actually compute a packing of stars with more than $d+1$ leaves. The adapted number of leaves depends on ϵ and $|X|$ and guarantees that FindExtremal iterates only a constant number of times (where the constant depends on ϵ). Moreover, for $d \geq 2$ it is possible that the algorithm returns two subsets A' and B' that do not fulfill the A'-star cover and restricted neighborhood properties, but nevertheless fulfill the local optimality conditions. However, we emphasize that the main concept is the same as for $d \leq 1$, the difference becomes important in the formal proof of the correctness of FindExtremal.

Proof Structure of the Remainder of this Section. In the following, we shortly outline the structure of the remaining description of FindExtremal. The main parts are as follows.

1. A proof that if A' and B' fulfill the A'-star cover property and the restricted neighborhood properties, then they also fulfill the local optimality conditions (Subsection 3.3).
2. A description of the StarPacking algorithm (used by FindExtremal in order to fulfill the A'-star cover property and the restricted X-neighborhood property) and its correctness proof (Subsection 3.4).
3. A description of the FindExtremal algorithm and its correctness proof (Subsection 3.5). This part is organized as follows, roughly ordered by increasing technical difficulty.
   (a) A pseudo-code formulation of FindExtremal.
   (b) A proof of the running time of FindExtremal.
   (c) A proof that FindExtremal always outputs A' and B' fulfilling the local optimality conditions.
   (d) A proof of the size condition, that is, that FindExtremal always returns a nonempty set B'.

3.3. Star Cover and Restricted Neighborhood Properties

Recall that FindExtremal tries to find two subsets A' and B' that fulfill the local optimality conditions and the size condition. We repeat the definition of the A'-star cover property and the restricted neighborhood properties, and show that these properties suffice to show the local optimality conditions.

A'-star cover property: There exists a packing of vertex-disjoint stars in $G[A' \cup B']$, each star having at least $d+1$ leaves, such that each vertex in A' is the center of such a star.

Restricted X-neighborhood property: There are no edges between B' and $X \setminus A'$.

Restricted Y-neighborhood property: There are no edges between B' and $N(X \setminus A') \setminus X$.

See Figure 3 for an illustration of these properties.

First, we show that these three properties are at least as strong as the local optimality conditions, that is, the first two properties of Lemma 1:

Lemma 4. *Let A' and B' be two vertex subsets satisfying the A'-star cover property and the restricted neighborhood properties. Then, the following two properties hold:*

(1) If S' is a bdd-d-set of the induced subgraph $G - (A' \cup B')$, then $A' \cup S'$ is a bdd-d-set of G.
(2) There is a minimum-cardinality bdd-d-set S of G with $A' \subseteq S$.

Proof. To prove (1), suppose that S' is a bdd-d-set of $G' := G - (A' \cup B')$. To prove that $S'' := S' \cup A'$ is a bdd-d-set of G, we have to consider the vertices in $N_G[B'] \setminus S''$. For these vertices we have to show that their degree is at most d in $G - S''$. To this end, we show that each vertex in $N_G[B'] \setminus A' \supseteq N_G[B'] \setminus S''$ has degree at most d in $G - A'$. Since X is a bdd-d-set of G and since $A' \subseteq X$, the only vertices that can have degree more than d in $G - A'$ are in $X \setminus A'$ and in $N(X \setminus A') \setminus X$, but these vertices are neither in B' nor are they neighbors of vertices in B' due to the restricted neighborhood properties, and hence each vertex in $N_G[B'] \setminus A'$ has degree at most d in $G - A'$ and, therefore, also in $G - S''$.

Before proving (2), one needs to show that A' is a minimum-cardinality bdd-d-set of $G[A' \cup B']$. Since X is a bdd-d-set of G, the vertex subset A' is a bdd-d-set of $G[A' \cup B']$; moreover, due to the A'-star cover property, for each vertex $v \in A'$ there is a star with at least $d + 1$ leaves in B' with center v. Since each star has at least $d + 1$ leaves, it has to contain at least one vertex of a minimum-cardinality bdd-d-set of $G[A' \cup B']$, and, therefore, every bdd-d-set of $G[A' \cup B']$ contains at least $|A'|$ vertices, showing that A' is a minimum-cardinality bdd-d-set of $G[A' \cup B']$.

To prove (2), suppose that S' is a minimum-cardinality bdd-d-set of G. If $A' \subseteq S'$, then we are done. Therefore, assume that $A' \not\subseteq S'$. We show that we can transform S' into a bdd-d-set S with $|S| = |S'|$ and $A' \subseteq S$. Let $A'' := A' \setminus S'$. As shown above, the set A' is a minimum-cardinality bdd-d-set of $G[A' \cup B']$. Since the bounded-degree property is hereditary, $S' \cap (A' \cup B')$ is a bdd-d-set of $G[A' \cup B']$. Since A' is a minimum-cardinality bdd-d-set of $G[A' \cup B']$, for the vertex subset $B'' := B' \cap S'$ we know that $|A''| \leq |B''|$. We claim that the set $S := (S' \setminus B'') \cup A''$ (thus, $A' \subseteq S$) is also a bdd-d-set of G. Since the vertices in B'' are the only vertices in $S' \setminus S$, it suffices to show that these vertices and their neighbors have degree at most d in $G - S$. As shown in the proof of (1), each vertex in $N_G[B'] \setminus A'$ has degree at most d in $G - A'$ and thus each vertex in $N_G[B''] \setminus A'$ has degree at most d in $G - A'$ and, therefore, S is a bdd-d-set of G. Due to $|A''| \leq |B''|$, S is a minimum-cardinality bdd-d-set of G with $A' \subseteq S$. □

Lemma 4 will be used in the proof of the correctness of FindExtremal—it helps to make the description of the underlying algorithm and the corresponding correctness proofs more accessible.

As described in the outline of FindExtremal (Subsection 3.2), the search for the subsets $A' \subseteq X$ and $B' \subseteq Y$ will be driven by the search for a packing of vertex-disjoint stars with centers in X and at least $(d + 1)$ leaves in $N(X)$. Roughly speaking, the centers of such stars with at least $d + 1$ leaves will be in A' and their leaves will be in B', which fulfills the A'-star cover property, but in order to fulfill the restricted neighborhood properties the vertices for A' and B' have to be selected carefully. To fulfill the third property of Lemma 1, which says that the returned vertex set $B' \subseteq Y$ is not empty, it is necessary that the packing of stars with centers in X and leaves in $N(X)$, which is used to compute A' and B', contains "as many stars with at least $d + 1$ leaves as possible". This is described in more detail in the next subsection.

3.4. Star Packing

As outlined in the preceding subsections, given a graph G and a bdd-d-set X, the task is to compute a star packing P with the centers of the stars being from X and the leaves being from $N(X) \subseteq Y = V \setminus X$. The stars in the packing shall have at most r leaves, where r depends

on d and is set to

$$r := \begin{cases} d+1, & \text{if } d \leq 1, \\ d+1+\lceil |X|^{\epsilon} \rceil, & \text{otherwise.} \end{cases}$$

The reason for this distinction between $d \leq 1$ and $d \geq 2$ will become clear later in the analysis of the algorithm. For the moment it is only important that $r \geq d+1$, which implies that an r-star is a forbidden subgraph. To compute the star packing P, we relax, on the one hand, the requirement that the stars in the packing have exactly r leaves, that is, the packing P might contain $<r$-stars. On the other hand, P shall have a maximum number of edges. The rough idea behind this requirement for a maximum number of edges is to maximize the number of r-stars in P, and to guarantee that the leaves of many r-stars are not adjacent to centers of $<r$-stars. Based on P, it is possible to "separate" many r-stars, whose centers will be in A' and whose leaves will be in B', such that their leaves are not adjacent to the center of any $<r$-star. This will guarantee that there are no edges between B' and $X \setminus A'$ (restricted X-neighborhood property). For computing such a star packing, we can restrict our attention to the bipartite graph J induced by the edges between X and $N(X)$, that is, $V(J) = X \cup N(X)$ and $E(J) = \{\{u,v\} \in E(G) \mid u \in X \text{ and } v \in N(X)\}$. The following lemma is a precise statement of the properties of the star packing. In the lemma, the centers of the "separated" r-stars are contained in a vertex set $C_X \subseteq X$ and the leaves of the remaining stars are contained in a vertex set $C_Y \subseteq Y$. The fact that there are no edges between the leaves of the "separated" r-stars and the centers of the remaining stars is expressed by saying that $C_X \cup C_Y$ is a vertex cover in J. Since FindExtremal will call the star packing algorithm for subsets of X and Y, we state the lemma with respect to $X' \subseteq X$ and $Y' \subseteq Y$.

Lemma 5. *In a bipartite graph J with vertex sets X' and Y', one can find in $O(n^2 \cdot m)$ time a $\leq r$-star packing P and a vertex cover $C_X \cup C_Y$ of J, where $C_X \subseteq X'$ and $C_Y \subseteq Y'$ such that*

1. *every vertex of C_X is the center of an r-star in P and the leaves of the r-stars in P (with center in C_X) are not in C_Y, and*
2. *every vertex of C_Y is a leaf in the star packing (of some $\leq r$-star with center in $X' \setminus C_X$).*

See Figure 4 for an example of such a packing P with vertex cover $C_X \cup C_Y$. Let ComputePacking(J,X',Y') be an algorithm that computes such a packing P and two vertex subsets C_X and C_Y as stated in Lemma 5.

Proof. From the given bipartite graph J, construct a flow network as follows (see Figure 4). Introduce two new vertices s and t, and add an edge with capacity r from s to v for every $v \in X'$, add an edge with capacity 1 from w to t for every $w \in Y'$, and add an edge with infinite capacity from $v \in X'$ to $w \in Y'$ if $\{v,w\} \in E(J)$. A maximum flow f corresponds to a packing P of $\leq r$-stars in J. Let (S,T) be the corresponding cut of capacity f with $s \in S$ and $t \in T$. The set $C_X \cup C_Y$ with $C_X := X' \cap T$ and $C_Y := Y' \cap S$ is a vertex cover for J; otherwise, there would be an edge with infinite capacity that leaves S, contradicting the fact that (S,T) has capacity f. Moreover, one can observe that the vertices in C_X must be centers of vertex-disjoint r-stars, whose leaves are in T, and the vertices in C_Y must be leaves of stars in the corresponding packing P (otherwise, the cut (S,T) would have higher capacity than the maximum flow).

A maximum flow can be computed in $O(n^2 \cdot m)$ time using, e.g., "Dinic's algorithm" (cf. [18]). □

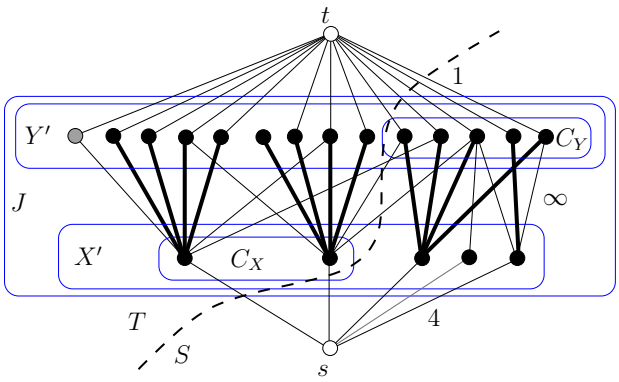


Figure 4: Example of a bipartite graph J, a packing P (bold edges and black vertices) in J that fulfills the properties stated in Lemma 5 for $r = 4$, and the additions to the graph that are used to prove Lemma 5 by maximum flow / minimum cut duality (s, t, and their incident edges). The corresponding flow network has the source s and the sink t (assuming that all edges are directed from bottom to top), where each edge incident to s has capacity $r = 4$, each edge in J has capacity ∞, and each edge incident to t has capacity 1. The dashed line shows a minimum s-t-cut (S, T), which can be used to compute C_X and C_Y.

We mention in passing that the structure that is found by Lemma 5 can be interpreted as a generalization of *crown structures* for Vertex Cover (cf. [14, 2]) to Bounded-Degree Vertex Deletion.

The algorithm ComputePacking is used by FindExtremal to find A' and B' that fulfill the local optimality conditions. The FindExtremal algorithm is described next.

3.5. The FindExtremal Algorithm

As described in the outline of FindExtremal (Subsection 3.2), the approach of the FindExtremal algorithm is to iteratively call ComputePacking and use the returned packing P and the vertex sets C_X and C_Y to try to obtain two sets A' and B' satisfying the A'-star cover property and the restricted neighborhood properties, or, if that fails, to forbid parts of graph and to try again. An important addition for $d \geq 2$, which has not been mentioned so far, is that if FindExtremal fails too many times to find A' and B' satisfying the A'-star cover property and the restricted neighborhood property, then one can directly return two vertex sets A' and B' satisfying the local optimality conditions (the first two properties of Lemma 1).

The description of FindExtremal is divided into four parts. First, we give a pseudo-code description of FindExtremal, implementing the "trial-and-error" strategy outlined above. Then, we show the running time of FindExtremal. After that, we show that FindExtremal is correct in the sense that it returns two subsets A' and B' that fulfill the local optimality conditions of Lemma 1. For the output of FindExtremal there are two different cases to consider, for one of them we show that the A'-star cover property and the restricted neighborhood properties are fulfilled, and for the other case we directly show the validity of the local optimality conditions. Note that the first case applies for all $d \geq 0$, but the latter only applies for $d \geq 2$. Finally, we address the last property in Lemma 1, that is, we show that the vertex subset B' is never empty.

Procedure: FindExtremal (G, X)
Input: An undirected graph G and a bdd-d-set X of G.
Output: A vertex subset pair (A', B') satisfying the local optimality conditions.

```
1 J ← bipartite graph with X and N(X) as its two vertex subsets and
        E(J) ← {{u, v} ∈ E(G) | u ∈ X and v ∈ N(X)}
2 F_X ← ∅; F_Y ← ∅
3 j ← 1
4 (P, C_X, C_Y) ← ComputePacking(J − (F_X ∪ F_Y), X \ F_X, Y \ F_Y)
5 if C_X = X \ F_X then return(X \ F_X, Y \ F_Y)
6 F_X ← X \ C_X; F_Y ← N_G[N_J(F_X)] \ X
7 if d ≥ 2 and j ≥ ⌈1/ε⌉ + 1 then return(X \ F_X, Y \ F_Y)
8 j ← j + 1
9 goto 4
```

Figure 5: Pseudo-code of FindExtremal.

3.5.1. Pseudo-Code for FindExtremal

The pseudo-code in Figure 5 shows the algorithm FindExtremal. The input to FindExtremal is a graph G and a bdd-d-set X. It starts with computing the bipartite graph J induced by the edges between X and $N(X)$ (line 1). Vertices in X that are forbidden in the course of the algorithm execution are stored in the set F_X, which is initialized with an empty set (line 2). Vertices in Y that are forbidden in the course of the algorithm execution are stored in the set F_Y, which is also initialized with an empty set (line 2). The variable j counts the number of calls of ComputePacking (line 4) and is initialized with "1" (line 3). The algorithm always returns a vertex pair (A', B'), where $A' = X \setminus F_X$ and $B' = Y \setminus F_Y$ (lines 5 and 7), that is, it returns all the vertices that are not forbidden. There are two possible cases when (A', B') is returned:

1. either the vertex set C_X contains all vertices in X that are not in F_X (line 5) or
2. the algorithm has iterated $\lceil 1/\epsilon \rceil + 1$ times (line 7).

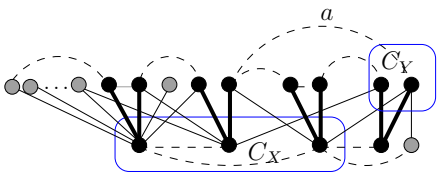
We will show that for each of these cases the pair (A', B') fulfills the local optimality conditions. If the first case does not apply, then the algorithm computes the new set F_X of forbidden vertices in X and updates the set F_Y of forbidden vertices in Y (line 6). After that, it is checked whether the second case applies (line 7). If not, the counter j is increased by one (line 8) and the algorithm starts over (line 9) by recomputing the star packing in line 4. See Figure 6 for an example of how FindExtremal works.

In the remainder of this section, we will show the following three statements.

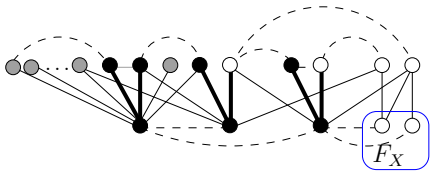
Statement 1. *Algorithm* FindExtremal *in Figure 5 runs in* $O(n^3 \cdot m)$ *time.*

Statement 2. *Algorithm* FindExtremal *in Figure 5 returns two vertex subsets* (A', B') *that fulfill the local optimality conditions:*

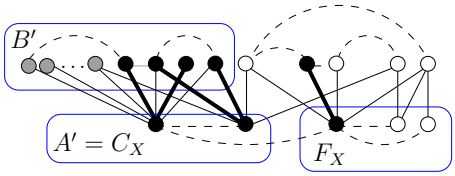
1. *If* S' *is a bdd-d-set of the induced subgraph* $G - (A' \cup B')$*, then* $A' \cup S'$ *is a bdd-d-set of* G*.*



(a) An initial star packing P (bold edges) computed in line 4 with no vertex being forbidden. The A'-star cover property and the restricted X-neighborhood property could be fulfilled by choosing $A' = C_X$ and $B' = Y \setminus C_Y$. However, the restricted Y-neighborhood property cannot be fulfilled due to, for example, the edge a.



(b) Forbidding vertices. In line 6, the set F_X is computed, which is used to compute the vertices of Y excluded from the next iteration, namely, the set F_Y.



(c) Computing a new star packing P and F_X, and, then, two sets A' and B' that fulfill the A'-star cover property and the restricted neighborhood properties.

Figure 6: Example for $d = 1$ of how the algorithm FindExtremal in Figure 5 finds the two vertex subsets A' and B'. Solid lines (bold and non-bold) denote the edges in J, dashed lines denote the edges in $E(G) \setminus E(J)$. White vertices are forbidden and excluded from further iterations for computing the star packing P. Black vertices and bold edges are in the star packing P.

2. There is a minimum-cardinality bdd-d-set S of G with $A' \subseteq S$.

Statement 3. *Let $G = (V, E)$ be an undirected graph and let X be a bdd-d-set of G. If $Y = V \setminus X$ contains more than $(d+1)^2 \cdot |X|$ vertices for $d \leq 1$ or more than $c' \cdot |X|^{1+\epsilon}$ vertices for $d \geq 2$ (for some c' depending on d and ϵ), then algorithm* FindExtremal *in Figure 5 returns two vertex subsets (A', B') such that B' is not empty.*

With Statements 1, 2, and 3, Lemma 1 follows immediately.

3.5.2. Running Time of FindExtremal

In order to show the running time, we have to show that an updated F_X (line 6) is always a superset of an old one.

Lemma 6. *Assume that* FindExtremal *does not return in line 5, that is, $C_X \neq X \setminus F_X$. Let F'_X be the set $X \setminus C_X$ computed in line 6. Then it holds that $F_X \subsetneq F'_X$.*

Proof. In line 4 of FindExtremal, the packing P and the vertex sets C_X and C_Y are computed for the bipartite subgraph $J' = J - (F_X \cup F_Y)$ with $X' := X \setminus F_X$ and $Y' := Y \setminus F_Y$ as the two vertex subsets. Thus, since $C_X \neq X \setminus F_X = X'$, $C_X \subsetneq X'$, and, therefore, $F'_X = X \setminus C_X \supsetneq X \setminus X' = F_X$. □

Lemma 6 shows that FindExtremal will eventually terminate (for all $d \geq 0$), and we can prove the running time, which shows Statement 1.

Proposition 1. *Algorithm* FindExtremal *in Figure 5 runs in $O(n^3 \cdot m)$ time.*

Proof. In each iteration FindExtremal either returns in line 5 or line 7, or at least one vertex from X is added to F_X due to Lemma 6. Thus, after at most $|X| < n$ iterations, $F_X = X$. Then, $X \setminus F_X$ is empty and hence C_X returned by ComputePacking is empty, and, therefore, the condition $C_X = X \setminus F_X = \emptyset$ is true and FindExtremal returns in line 5. In each iteration, FindExtremal calls ComputePacking, which runs in $O(n^2 \cdot m)$ time (Lemma 5); all the other operations are simple assignments, if-instructions, and neighborhood computations, which take $O(n + m)$ time. □

3.5.3. Fulfillment of Local Optimality Conditions

In order to show Statement 2, that is, that FindExtremal returns two vertex subsets (A', B') fulfilling the local optimality conditions, it is important to note that

$$F_Y = N_G[N_J(F_X)] \setminus X$$

is an invariant of FindExtremal (see line 6). We will need this invariant for the proofs of several lemmas and propositions. The next proposition corresponds to the case that FindExtremal returns in line 5.

Proposition 2. *Let (P, C_X, C_Y) be the output of* ComputePacking*$(J - (F_X \cup F_Y), X \setminus F_X, Y \setminus F_Y)$ (line 4). If $C_X = X \setminus F_X$, then $A' = X \setminus F_X$ and $B' = Y \setminus F_Y$ fulfill the local optimality conditions.*

Proof. We show that A' and B' fulfill the A'-star cover property and the restricted neighborhood properties. Due to Lemma 4, the sets A' and B' then also fulfill the local optimality conditions.

In line 4 of FindExtremal, the packing P and the vertex sets C_X and C_Y are computed for the bipartite subgraph $J' = J - (F_X \cup F_Y)$ with $X' := X \setminus F_X$ and $Y' := Y \setminus F_Y$ as the two vertex sets of J'. Thus, since $C_X = X'$, due to Lemma 5 we know that $C_Y = \emptyset$, since there are no stars with centers in $X' \setminus C_X = \emptyset$. Hence, due to Lemma 5 the vertices in C_X are centers of r-stars with leaves in Y', thus $A' = X'$ and $B' = Y'$ fulfill the A'-star cover property. We emphasize that this is also correct for the case $F_X = X$; in this case, the set A' is empty, and the A'-star cover property is trivially fulfilled. The restricted X-neighborhood property, that is, that there is no edge between B' and $X \setminus A' = F_X$, is fulfilled, since all the neighbors of F_X in Y are in $F_Y = N_G[N_J(F_X)] \setminus X$ and therefore not in $B' = Y \setminus F_Y$. The restricted Y-neighborhood property, that is, that there is no edge between B' and $N(X \setminus A') \setminus X = N_J(F_X)$, is fulfilled, since all the neighbors of $N_J(F_X)$ are in F_Y and therefore not in B'. □

We mention in passing that for the case $F_X = X$ all vertices in B' have distance at least three to vertices in X, as all vertices of distance at most two from vertices in X are in F_Y. Since X is a bdd-d-set of G, the vertices in B' thus have degree at most d, and all their neighbors have degree at most d as well. This is the reason why the vertices in B' do not have to be considered for a solution even when A' is empty.

Next, we deal with the more involved case that FindExtremal returns in line 7. We need to define some notation in order to be able to refer to the variables in FindExtremal in some particular

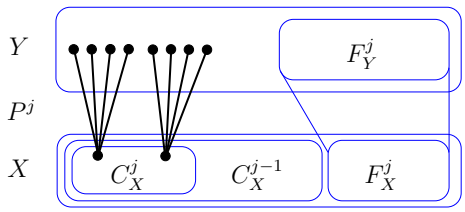


Figure 7: Illustration of the relation of the sets F_X^j, F_Y^j, C_X^j, C_X^{j-1}, and a packing P^j for $r = 4$. Only the r-stars with center in C_X^j in P_j are shown (there might exist $\leq r$-stars with center in $C_X^{j-1} \setminus C_X^j$).

iteration. Let F_X^j and F_Y^j be the sets F_X and F_Y, respectively, in the jth call of ComputePacking (line 4). Furthermore, let (P^j, C_X^j, C_Y^j) be the output of the jth call of ComputePacking$(J - (F_X^j \cup F_Y^j), X \setminus F_X^j, Y \setminus F_Y^j)$. Since $F_X^j = X \setminus C_X^{j-1}$ (line 6) it holds that $C_X^{j-1} = X \setminus F_X^j$. See Figure 7 for an illustration.

The key for the proof that FindExtremal returns two vertex sets A' and B' satisfying the local optimality conditions is the following result that, if FindExtremal is iterated sufficiently many times, then every minimum-cardinality solution contains C_X.

Lemma 7. *For $j \geq \lceil 1/\epsilon \rceil + 1$ and $d \geq 2$, the set C_X^j is contained in every minimum-cardinality bdd-d-set of G.*

The proof of Lemma 7 is given below. Before that, we use Lemma 7 to show that the two sets (A', B') returned by FindExtremal in line 7 fulfill the local optimality conditions.

Proposition 3. *If $j \geq \lceil 1/\epsilon \rceil + 1$ for $d \geq 2$, then the following properties hold for the vertex subsets (A', B') returned by* FindExtremal *in line 7:*

(1) If S' is a bdd-d-set of the induced subgraph $G - (A' \cup B')$, then $A' \cup S'$ is a bdd-d-set of G.
(2) There is a minimum-cardinality bdd-d-set S of G with $A' \subseteq S$.

Proof. We first show the second property. Since F_X^j is set to $X \setminus C_X^j$ in line 6, $A' = X \setminus F_X^j = C_X^j$, and hence by Lemma 7 there exists a minimum-cardinality bdd-d-set S such that $A' \subseteq S$.

Next, we show the first property. Let S' be a bdd-d-set of $G-(A' \cup B')$. To show that $A' \cup S'$ is a bdd-d-set of G, it suffices to show that all vertices in $N[B'] \setminus A'$ have degree at most d in $G-(A' \cup S')$. Since X is a bdd-d-set of G, $X \setminus A' = F_X^j$ is a bdd-d-set for $G - A'$. Therefore, in $G - A'$, the only vertices that possibly have degree more than d are in F_X^j or adjacent to vertices in F_X^j. Since F_Y^j is set to $N_G[N_J(F_X^j)] \setminus X$ (line 6), neither the vertices in $B' = Y \setminus F_Y^j$ nor their neighbors $N(B')$ can be in F_X^j or $N_J(F_X^j)$, and thus the vertices in $N[B'] \setminus A'$ have degree at most d in $G - A'$ and therefore also in $G - (A' \cup S')$. Hence, $A' \cup S'$ is a bdd-d-set of G, which shows the first property. □

With Proposition 2 and Proposition 3, Statement 2 follows immediately.

It remains to show Lemma 7. To this end, we first prove the following lemma.

Lemma 8. *Suppose that $d \geq 2$ and $C_X^j \neq X \setminus F_X^j$. If for a minimum-cardinality bdd-d-set S it holds that $|C_X^j \setminus S| = l$, then $|C_X^{j-1} \setminus S| \geq l \cdot \lceil |X|^\epsilon \rceil$.*

Proof. The proof approach is to assume that $|C_X^{j-1} \setminus S| < l \cdot \lceil |X|^\epsilon \rceil$ and to show that then there exists a bdd-d-set of G that is smaller than S, contradicting the assumption that S is a minimum-cardinality bdd-d-set.

First, for a simpler presentation of the main argument, assume that v is the only vertex in C_X^j that is not in S, that is, $C_X^j \setminus S = \{v\}$. Due to Lemma 5, each vertex of C_X^j is the center of an r-star in the star packing P^j. Since P^j is a star packing for $J - (F_X^j \cup F_Y^j)$, all the leaves of these stars are in $Y \setminus F_Y^j$. Recall that $r = d + 1 + \lceil |X|^\epsilon \rceil$ (cf. Subsection 3.4). Since $v \notin S$, at least $r - d = \lceil |X|^\epsilon \rceil + 1$ leaves of the r-star in P^j with center v must be in S, since otherwise v would have degree more than d in $G - S$. Let S_v be the set of these leaves. If $C_X^{j-1} \setminus S$ contains less than $\lceil |X|^\epsilon \rceil$ vertices, then one obtains a smaller bdd-d-set S' by setting $S' := (S \setminus S_v) \cup (C_X^{j-1} \setminus S)$, contradicting the assumption that S is minimum; the set S' is clearly smaller than S, and one can show that S' is a bdd-d-set as follows. We only have to verify that each vertex in $N[S_v] \setminus S'$ has degree at most d in $G - S'$. Clearly, $C_X^{j-1} = X \setminus F_X^j \subseteq S'$. Since X is a bdd-d-set of G, the only vertices in $G - S'$ that could have degree more than d are in F_X^j and $N_J(F_X^j)$. Since F_Y^j is set to $N_G[N_J(F_X^j)] \setminus X$ (line 6), neither the vertices in $S_v \subseteq Y \setminus F_Y^j$ nor their neighbors $N(S_v)$ can be in F_X^j or $N_J(F_X^j)$. Thus, the vertices in $N[S_v] \setminus S'$ have degree at most d in $G - S'$. This shows that S' is a bdd-d-set of G.

Thus, $C_X^{j-1} \setminus S$ contains at least $\lceil |X|^\epsilon \rceil$ vertices. One can show in complete analogy that if $C_X^j \setminus S$ contains l vertices, then $C_X^{j-1} \setminus S$ contains at least $l \cdot \lceil |X|^\epsilon \rceil$ vertices. □

Now, we are ready to prove Lemma 7, that is, that for $j \geq \lceil 1/\epsilon \rceil + 1$ and $d \geq 2$, the set C_X^j is contained in every minimum-cardinality bdd-d-set of G.

Proof (of Lemma 7). In order to show the lemma, that is, that C_X^j is contained in every minimum-cardinality bdd-d-set, we assume that there exists a minimum-cardinality bdd-d-set S such that $C_X^j \not\subseteq S$, and show a contradiction by proving that S cannot have minimum cardinality.

By assumption, $C_X^j \setminus S$ contains at least one vertex. By Lemma 8, then $C_X^{j-1} \setminus S$ contains at least $\lceil |X|^\epsilon \rceil$ vertices. By a repeated application of Lemma 8, we obtain that $C_X^1 \setminus S$ contains at least $\lceil |X|^\epsilon \rceil^{j-1} \geq |X|^{\lceil 1/\epsilon \rceil \cdot \epsilon} \geq |X|$ vertices. However, for each vertex in $C_X^1 \setminus S$ there is a vertex-disjoint r-star (Lemma 5), where $r = d + 1 + \lceil |X|^\epsilon \rceil$ (cf. Subsection 3.4), and hence S would have to contain more than $|X|$ vertices in order to be a bdd-d-set. This is a contradiction to the assumption that S has minimum cardinality, since X is a bdd-d-set of G and therefore $|X|$ is a trivial upper bound of the size of a minimum-cardinality bdd-d-set of G. This shows that every minimum-cardinality bdd-d-set S contains C_X^j. □

In summary, FindExtremal always returns two sets A' and B' satisfying the local optimality conditions. It remains to show Statement 3, that is, that the returned set B' is not empty. The key to showing this is to prove that there cannot be too many forbidden vertices in F_Y compared to F_X.

3.5.4. Number of Forbidden Vertices

Recall that one of the preconditions of Statement 3 is that Y contains more than $(d+1)^2 \cdot |X|$ vertices for $d \leq 1$ or more than $O(|X|^{1+\epsilon})$ vertices for $d \geq 2$. The point is, as we will show, that the set F_Y in FindExtremal always contains at most $(d+1)^2 \cdot |F_X|$ vertices for $d \leq 1$ or at most $O(|F_X|^{1+\epsilon})$ vertices for $d \geq 2$. Hence, the set $B' := Y \setminus F_Y$ returned by FindExtremal can never be empty.

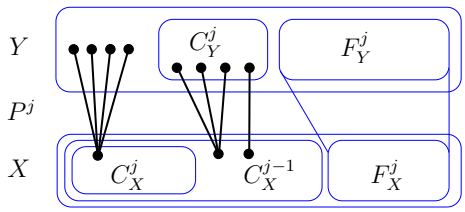


Figure 8: Illustration of the relation of the sets F_X^j, F_Y^j, C_X^j, C_Y^j, C_X^{j-1}, and the packing P^j for $r = 4$.

Lemma 9. *For each $j \geq 1$, the set F_Y^j has size at most $r(1 + d + dj(d-1)^j) \cdot |F_X^j|$.*

Proof. We recall the definitions of some important notations. Let F_X^j and F_Y^j be the sets F_X and F_Y, respectively, in the jth call of ComputePacking (line 4 of FindExtremal). Furthermore, let (P^j, C_X^j, C_Y^j) be the output of the jth call of ComputePacking$(J - (F_X^j \cup F_Y^j), X \setminus F_X^j, Y \setminus F_Y^j)$. Note that $F_X^1 = F_Y^1 = \emptyset$ (line 2 of FindExtremal). Since $F_X^j = X \setminus C_X^{j-1}$ (line 6 of FindExtremal) it holds that $C_X^{j-1} = X \setminus F_X^j$. See Figure 8 for an illustration that shows the variables that are important in this proof. Recall that $F_Y^j = N_G[N_J(F_X^j)] \setminus X$ (line 6 of FindExtremal).

First, we bound the size of F_Y^2 with respect to F_X^2. Due to Lemma 5, $C_X^1 \cup C_Y^1$ is a vertex cover for J and thus there are no edges between $F_X^2 = X \setminus C_X^1$ and $Y \setminus C_Y^1$. Hence, since due to Lemma 5 every vertex in C_Y^1 is the leaf of a $\leq r$-star with center in F_X^2, we have $N_J(F_X^2) = C_Y^1$, and $|N_J(F_X^2)| \leq r \cdot |F_X^2|$. Since X is a bdd-d-set of G, it follows that $|F_Y^2| = |N_G[N_J(F_X^2)] \setminus X| \leq r(d+1) \cdot |F_X^2|$.

Two important observations for the above size bound are that F_Y^2 contains all vertices that have distance at most one to a vertex in C_Y^1 in $G - X$, and that the vertices in C_Y^1 are leaves of $\leq r$-stars with center in F_X^2. We generalize these observations and show the general size bound of F_Y^j with respect to F_X^j. To this end, define

$$D_X^j := C_X^{j-1} \setminus C_X^j (\text{for } j \geq 2).$$

Hence, D_X^j is exactly the set $F_X^{j+1} \setminus F_X^j$ (informally speaking, the set of vertices that are added to F_X in the jth iteration of FindExtremal), and $N_G[N_J(D_X^j)] \setminus X$ contains all vertices in $F_Y^{j+1} \setminus F_Y^j$.

By Lemma 5, $C_X^j \cup C_Y^j$ is a vertex cover of $J - (F_X^j \cup F_Y^j)$, and the vertices in C_Y^j are the leaves of $\leq r$-stars with center in D_X^j. Thus, there are no edges between D_X^j and $Y \setminus (F_Y^j \cup C_Y^j)$ and therefore $N_J(D_X^j)$ contains the vertices in C_Y^j and possibly also vertices in F_Y^j, but no vertices in $Y \setminus (F_Y^j \cup C_Y^j)$ (cf. Figure 8). The number of vertices in C_Y^j is easy to bound: since the vertices in C_Y^j are the leaves of a $\leq r$-star packing with centers in D_X^j, we have

$$|C_Y^j| \leq r \cdot |D_X^j|.$$

For a vertex v of $N_J(D_X^j) \cap F_Y^j$, observe that v is in F_Y^j because either it is in $C_Y^{j'}$ for some $j' < j$, or there is a path in $G - X$ of length at most $j - j'$ from v to a vertex in $C_Y^{j'}$ for some $j' < j$. Hence, for $1 \leq j' < j$, in j iterations the algorithm FindExtremal can only add vertices to F_Y^j that are at distance at most $j - j'$ from a vertex in $C_Y^{j'}$ in $G - X$. To simplify the analysis, for

each $1 \leq j' < j$, we bound the number of vertices at distance at most $j+1$ from $C_Y^{j'}$ in $G - X$. Since $G - X$ has bounded degree d, the number of all vertices at distance at most $j+1$ from $C_Y^{j'}$ in $G - X$ (including the vertices in $C_Y^{j'}$) can be bounded by

$$\begin{aligned} & r \cdot |D_X^{j'}| + rd \cdot |D_X^{j'}| + rd(d-1) \cdot |D_X^{j'}| + rd(d-1)^2 \cdot |D_X^{j'}| + \cdots + rd(d-1)^j \cdot |D_X^{j'}| \\ &= |D_X^{j'}| \cdot r \left((1+d) + d((d-1) + (d-1)^2 + \cdots + (d-1)^j)\right) \\ &\leq |D_X^{j'}| \cdot r \left((1+d) + dj(d-1)^j\right). \end{aligned}$$

In total, since $F_X^j = \bigcup_{1 \leq j' < j} D_X^{j'}$ and $D_X^{j'} \cap D_X^{j''} = \emptyset$ for $j' \neq j''$ (by the definition of D_X^j), we obtain

$$\begin{aligned} |F_Y^j| &= \sum_{1 \leq j' < j} |D_X^{j'}| \leq \sum_{1 \leq j' < j} |D_X^{j'}| \cdot r \left(1 + d + dj(d-1)^j\right) \\ &= |F_X^j| \cdot r \left(1 + d + dj(d-1)^j\right). \end{aligned}$$

□

With Lemma 9 one can now show the following proposition, which proves Statement 3.

Proposition 4. *Let $G = (V, E)$ be an undirected graph and let X be a bdd-d-set of G. If $Y = V \setminus X$ contains more than $(d+1)^2 \cdot |X|$ vertices for $d \leq 1$ or more than $c' \cdot |X|^{1+\epsilon}$ vertices for $d \geq 2$ (for some c' depending on d and ϵ), then algorithm* FindExtremal *in Figure 5 returns two vertex subsets (A', B') such that B' is not empty.*

Proof. For $d \leq 1$ (recall that $r = d + 1$ in this case, cf. Subsection 3.4), if FindExtremal returns (A', B') in line 5, then by Lemma 9 one knows that $|F_Y| \leq |F_X| \cdot (d+1)^2$. Since Y contains more than $(d+1)^2 \cdot |X|$ vertices and since $F_X \subseteq X$, $B' = Y \setminus F_Y$ cannot be empty.

For $d \geq 2$ (recall that $r = d + 1 + \lceil |X|^\epsilon \rceil$ in this case), if FindExtremal returns (A', B') (in line 5 or line 7), then $j \leq \lceil 1/\epsilon \rceil + 1$ ($j \leq \lceil 1/\epsilon \rceil + 1$ in line 5 and $j = \lceil 1/\epsilon \rceil + 1$ in line 7). Since $F_X \subseteq X$, one knows by Lemma 9 that

$$\begin{aligned} |F_Y| &\leq (d + 1 + \lceil |X|^\epsilon \rceil) \left(1 + d + d(\lceil 1/\epsilon \rceil + 1)(d-1)^{\lceil 1/\epsilon \rceil + 1}\right) \cdot |X| \\ &\leq c' \cdot |X|^{1+\epsilon} \text{ (for some } c' \text{ depending on } d \text{ and } \epsilon). \end{aligned}$$

Hence, since Y contains more than $c' \cdot |X|^{1+\epsilon}$ vertices, $B' = Y \setminus F_Y$ cannot be empty. □

This finishes the proof of the local optimization theorem for Bounded-Degree Vertex Deletion (Theorem 1).

Remarks. Note that the entire local optimization algorithm is based on packing stars. For example, for $d = 1$, it is based on a packing of stars with two leaves (P_3). We can use our local optimization algorithm also for the problem of packing at least k copies of P_3 in a given graph G, called P_3-Packing. First, we compute again a packing of stars with two leaves. If we find at least k stars, then we abort returning "yes-instance". Otherwise, let the set X contain the vertices of the less than k stars, and proceed with the kernelization as described. To show that the algorithm returns two vertex subsets A' and B' satisfying the A'-star cover property for $d = 1$, we used the fact that

there is a packing of 2-stars in $G[A' \cup B']$ such that each vertex in A' is the center of one such star (and the leaves of these 2-stars are therefore in B'). Moreover, the restricted neighborhood properties also imply that there is no P_3 using vertices from B' and $G \setminus A'$, thus using the stars in $G[A' \cup B']$ is always optimal. The size bound and the remaining proof then are exactly the same. Thus, we obtain the following result.

Corollary 2. *P_3-Packing admits a problem kernel of $15k$ vertices.*

The currently best-known problem kernel for P_3-Packing has $7k$ vertices [42]. This improvement stems basically from some local modification of an initial maximal P_3-packing and would also work with our technique.

The main point we want to make here is that there seems to be a close relationship between the kernelizations for star packing problems and Bounded-Degree Vertex Deletion, and similar observations also hold for other packing/deletion problem pairs. Note that the problem of packing at least k stars of more than two leaves ($K_{1,l}$-Packing for constant l) admits a problem kernel of $O(k^2)$ vertices [40]. It is conceivable that our technique also works for this problem. However, to this end, one would have to provide a new proof of Proposition 3.

4. Parameterized Hardness for Unbounded d

Our results in Section 3 show that Bounded-Degree-d Vertex Deletion is fixed-parameter tractable with respect to the parameter k if d is a constant. However, as we will prove in this section, the problem becomes presumably fixed-parameter intractable for unbounded d—in other words, we show it to be W[2]-complete.

A parameterized problem L is contained in W[2] if there is a parameterized reduction from L to the Weighted Satisfiability problem for polynomial-size weft-two circuits of constant depth [19]. Herein, the *weft* of a circuit C is the maximum number of "large" gates on an input-output path in C. In a Boolean circuit, a gate ($\neg$, $\wedge$, $\vee$) is *small* if it has fan-in bounded by a function of the parameter k, whereas *large* gates have unbounded fan-in. The *depth* of a circuit C is defined as the maximum number of gates on an input-output path in C. The *weight* of a truth assignment is the number of variables that are set true. To show W[2]-hardness, we employ the W[2]-complete Dominating Set problem (see, e.g., [20]).

> Dominating Set
> **Input:** An undirected graph $G = (V, E)$ and an integer $k \geq 0$.
> **Question:** Does there exist a vertex subset $S \subseteq V$ of size at most k such that every vertex of V belongs to S or has a neighbor in S?

Theorem 2. *For d being unbounded,* Bounded-Degree Vertex Deletion *is W[2]-complete with respect to the parameter k.*

Proof. The W[2]-hardness of Bounded-Degree Vertex Deletion can be easily shown by a parameterized reduction from the W[2]-complete Dominating Set problem: Pad the vertices in the Dominating Set instance with degree-one neighbors such that every vertex has the same degree. Let $d + 1$ be the degree of the resulting regular graph. For each original vertex, at least one neighbor or the vertex itself has to be removed in order to obtain maximum degree d (we assume without loss of generality that no newly added degree-one vertex is removed by an optimal solution), which directly corresponds to a dominating set in the Dominating Set instance.

Second, we show the membership of Bounded-Degree Vertex Deletion in W[2]. Let $(G = (V, E), k, d)$ be an instance of Bounded-Degree Vertex Deletion. We construct a Boolean circuit of weft two and constant depth, where small gates have fan-in bounded by an arbitrary fixed function of k. This shows membership in W[2].

The Boolean circuit is given by a Boolean expression E that is satisfiable by a weight-k truth assignment if and only if G has a k-vertex solution to Bounded-Degree Vertex Deletion.

The informal idea of the construction is as follows: We have k choices to select vertices in V to be in the solution S. For each choice, we introduce a *block* of $|V|$ Boolean variables. A Boolean subexpression E_1 will ensure that only one Boolean variable of each block can be set to true; the variable set to true in a block corresponds directly to the choice of the corresponding vertex to be in S. To avoid that a single vertex appears twice in a solution, we introduce a second subexpression E_2. Furthermore, we need Boolean subexpressions to express that a vertex $v \in V$ is in S (subexpression $E_3(v)$) or has to have at least $\deg(v) - d$ neighbors in S (subexpression $E_4(v)$). The complete Boolean expression can then be written as

$$E := E_1 \wedge E_2 \wedge \bigwedge_{v \in V^+} (E_3(v) \vee E_4(v)),$$

where V^+ is the set of vertices in V with degree at least $d + 1$. Next, we formally describe the corresponding Boolean expressions:

The set of Boolean variables for E is

$$X := \{c[i, u] : 1 \leq i \leq k, u \in V\},$$

where $c[i, u]$ means that the ith choice of a vertex of S is vertex u. We define

$$E_1 := \bigwedge_{i=1}^{k} \bigwedge_{u,u' \in V, u \neq u'} \neg(c[i, u] \wedge c[i, u']),$$

meaning that no two variables in the same block can be set true,

$$E_2 := \bigwedge_{u \in V} \bigwedge_{1 \leq i < j \leq k} \neg(c[i, u] \wedge c[j, u]),$$

meaning that no two variables corresponding to the same vertex are set true, and

$$E_3(v) := \bigvee_{1 \leq i \leq k} c[i, v],$$

meaning that at least one variable corresponding to vertex v is set true in some block. Let $R(k, r)$ denote the set of size-r subsets of $\{1, \ldots, k\}$. Finally, we define

$$E_4(v) := \bigvee_{R' \in R(k, \deg(v) - d)} \bigwedge_{i \in R'} \bigvee_{u \in N(v)} c[i, u].$$

Informally speaking, this subexpression examines, for a given vertex v, every possible subset of blocks that is large enough to witness that sufficiently many neighbors (that is, at least $\deg(v) - d$) of v are chosen to be in the solution. Subexpression $E_4(v)$ checks for every block B in each such

subset whether at least one variable of a neighbor of v is set true in B and returns true if this is the case for all blocks in the subset. Due to expression E_1 we know that then there are at least $\deg(v) - d$ neighbors of v chosen to be in the solution S.

One can easily verify that E is satisfiable by a weight-k truth assignment if and only if G has a k-vertex solution to Bounded-Degree Vertex Deletion. Moreover, the depth of the circuit is constant and the weft is two, as the only large gates (that is, with fan-in that is not bounded by a function of k) correspond to the outermost conjunction of E (over all $v \in V^+$), the inner conjunction $\bigwedge$ of E_1, the outermost conjunction of E_2, and the innermost disjunction of $E_4(v)$. All other gates have fan-in bounded by some function of k. □

5. Conclusion

Our main result in this paper is a generalization of the Nemhauser-Trotter-Theorem, which applies to the Bounded-Degree Vertex Deletion problem with $d = 0$ (that is, Vertex Cover), to the general case with arbitrary $d \geq 0$. In particular, in this way we contribute an almost linear-vertex problem kernel for Bounded-Degree-d Vertex Deletion with respect to the parameter k for any fixed d, that is, a kernel of $O(k^{1+\epsilon})$ vertices for any fixed d. For $d \leq 1$, the same method even gives a linear-vertex problem kernel. To this end, we developed a new algorithmic strategy that is based on extremal combinatorial arguments. The original NT-Theorem [36] has been proven using linear programming relaxations—we see no way how this could have been generalized to Bounded-Degree Vertex Deletion. By way of contrast, we presented a purely combinatorial data reduction algorithm which is also completely different from known combinatorial data reduction algorithms for Vertex Cover (see [1, 2, 5, 13]). Finally, Baldwin et al. [4, page 175] remarked that, with respect to practical applicability in the case of Vertex Cover kernelization, combinatorial data reduction algorithms are more powerful than "slower methods that rely on linear programming relaxation". Some recent experimental results [35] partially exploiting our data reduction for Bounded-Degree Vertex Deletion confirm the relevance of the proposed kernelization.

Our W[2]-completeness result for Bounded-Degree Vertex Deletion with unbounded degree value d gives a complementing negative result that shows the fundamental limitations concerning fixed-parameter algorithms for Bounded-Degree Vertex Deletion with respect to the parameter "solution size".

As to challenges for future research, first of all, does there exist a linear-vertex problem kernel for Bounded-Degree-d Vertex Deletion for every constant d? Moreover, it would be worthwhile to study whether our methods can also be successfully applied to vertex-weighted problem variants (such as does the Nemhauser-Trotter-Theorem). Finally, applicability to the problem of generating *regular* graphs (that is, degree exactly d for all vertices) should be investigated as well (cf. [32]).

6. Acknowledgements

We are grateful to Jiří Sgall for spotting a flaw in a previous version [22] of this paper, and to Dániel Marx for helpful remarks improving the presentation.

References

[1] F. N. Abu-Khzam, R. L. Collins, M. R. Fellows, M. A. Langston, W. H. Suters, and C. T. Symons. Kernelization algorithms for the Vertex Cover problem: Theory and experiments. In *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments (ALENEX '04)*, pages 62–69. ACM/SIAM, 2004.

[2] F. N. Abu-Khzam, M. R. Fellows, M. A. Langston, and W. H. Suters. Crown structures for vertex cover kernelization. *Theory Comput. Syst.*, 41(3):411–430, 2007.

[3] B. Balasundaram, S. Butenko, and I. V. Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Oper. Res.*, 2010. To appear.

[4] N. E. Baldwin, E. J. Chesler, S. Kirov, M. A. Langston, J. R. Snoddy, R. W. Williams, and B. Zhang. Computational, integrative, and comparative methods for the elucidation of genetic coexpression networks. *J. Biomed. Biotechnol.*, 2005(2):172–180, 2005.

[5] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Ann. of Discrete Math.*, 25:27–45, 1985.

[6] R. Bar-Yehuda, D. Rawitz, and D. Hermelin. An extension of the Nemhauser & Trotter theorem to generalized vertex cover with applications. *SIAM J. Discrete Math.*, 24(1):287–300, 2010.

[7] H. L. Bodlaender. Kernelization: New upper and lower bound techniques. In *Proceedings of the 4th International Workshop on Parameterized and Exact Computation (IWPEC '09)*, volume 5917 of *LNCS*, pages 17–37. Springer, 2009.

[8] H. L. Bodlaender and B. van Antwerpen-de Fluiter. Reduction algorithms for graphs of small treewidth. *Inform. and Comput.*, 167(2):86–119, 2001.

[9] R. Boliac, K. Cameron, and V. V. Lozin. On computing the dissociation number and the induced matching number of bipartite graphs. *Ars Combin.*, 72:241–253, 2004.

[10] J. Chen, I. A. Kanj, and W. Jia. Vertex cover: Further observations and further improvements. *J. Algorithms*, 41(2):280–301, 2001.

[11] Z.-Z. Chen, M. Fellows, B. Fu, H. Jiang, Y. Liu, L. Wang, and B. Zhu. A linear kernel for co-path/cycle packing. In *Proceedings of the 6th International Conference on Algorithmic Aspects in Information and Management (AAIM '10)*, volume 6124 of *LNCS*, pages 90–102. Springer, 2010.

[12] E. J. Chesler, L. Lu, S. Shou, Y. Qu, J. Gu, J. Wang, H. C. Hsu, J. D. Mountz, N. E. Baldwin, M. A. Langston, D. W. Threadgill, K. F. Manly, and R. W. Williams. Complex trait analysis of gene expression uncovers polygenic and pleiotropic networks that modulate nervous system function. *Nat. Genet.*, 37(3):233–242, 2005.

[13] M. Chlebík and J. Chlebíková. Crown reductions for the minimum weighted vertex cover problem. *Discrete Appl. Math.*, 156:292–312, 2008.

[14] B. Chor, M. R. Fellows, and D. W. Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In *Proceedings of the 30th International Workshop on Graph-Theoretic Conecpts in Computer Science (WG '04)*, volume 3353 of *LNCS*, pages 257–269. Springer, 2004.

[15] A. Coja-Oghlan. Solving NP-hard semirandom graph problems in polynomial expected time. *J. Algorithms*, 62 (1):19–46, 2007.

[16] V. J. Cook, S. J. Sun, J. Tapia, S. Q. Muth, D. F. Argüello, B. L. Lewis, R. B. Rothenberg, P. D. McElroy, and the Network Analysis Project Team. Transmission network analysis in tuberculosis contact investigations. *J. Infect. Dis.*, 196:1517–1527, 2007.

[17] H. Dell and D. van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC '10)*, pages 251–260. ACM Press, 2010.

[18] Y. Dinitz. Dinitz' algorithm: The original version and Even's version. In *Theoretical Computer Science, Essays in Memory of Shimon Even*, volume 3895 of *LNCS*, pages 218–240. Springer, 2006.

[19] R. G. Downey and M. R. Fellows. Threshold dominating sets and an improved characterization of $W[2]$. *Theor. Comput. Sci.*, 209(1-2):123–140, 1998.

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

[21] V. Estivill-Castro, M. R. Fellows, M. A. Langston, and F. A. Rosamond. FPT is P-time extremal structure I. In *Proceedings of the 1st Algorithms and Complexity in Durham (ACiD '05) Workshop*, volume 4 of *Texts in Algorithmics*, pages 1–41. College Publications, 2005.

[22] M. R. Fellows, J. Guo, H. Moser, and R. Niedermeier. A generalization of Nemhauser and Trotter's local optimization theorem. In *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science (STACS '09)*, pages 409–420. IBFI Dagstuhl, Germany, 2009.

[23] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[24] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1): 31–45, 2007.

[25] J. Guo, C. Komusiewicz, R. Niedermeier, and J. Uhlmann. A more relaxed model for graph-based data clustering: s-plex editing. In *Proceedings of the 5th International Conference on Algorithmic Aspects in Information and Management (AAIM '09)*, volume 5564 of *LNCS*, pages 226–239. Springer, 2009. Long version to appear in *SIAM Journal on Discrete Mathematics*.

[26] D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM J. Comput.*, 11(3):555–556, 1982.

[27] D. S. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company, 1997.

[28] S. Khot and O. Regev. Vertex cover might be hard to approximate to within $2-\epsilon$. *J. Comput. System Sci.*, 74 (3):335–349, 2008.

[29] S. Khuller. The Vertex Cover problem. *SIGACT News*, 33(2):31–33, 2002.

[30] 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.

[31] M. A. Langston, 2008. Personal communication.

[32] L. Mathieson and S. Szeider. The parameterized complexity of regular subgraphs problems and generalizations. In *Proceedings of the 14th Computing: The Australasian Theory Symposium (CATS '08)*, volume 77 of *CRPIT*, pages 79–86. Australian Computer Society, 2008.

[33] N. Memon, K. C. Kristoffersen, D. L. Hicks, and H. L. Larsen. Detecting critical regions in covert networks: A case study of 9/11 terrorists network. In *Proceedings of the 2nd International Conference on Availability, Reliability and Security (ARES '07)*, pages 861–870. IEEE Computer Society Press, 2007.

[34] H. Moser. *Finding Optimal Solutions for Covering and Matching Problems*. PhD thesis, Institut für Informatik, Friedrich-Schiller Universität Jena, 2009.

[35] H. Moser, R. Niedermeier, and M. Sorge. Algorithms and experiments for clique relaxations—finding maximum s-plexes. In *Proceedings of the 8th International Symposium on Experimental Algorithms (SEA '09)*, volume 5526 of *LNCS*, pages 233–244. Springer, 2009.

[36] G. L. Nemhauser and L. E. Trotter. Vertex packings: Structural properties and algorithms. *Math. Program.*, 8: 232–248, 1975.

[37] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[38] N. Nishimura, P. Ragde, and D. M. Thilikos. Fast fixed-parameter tractable algorithms for nontrivial generalizations of Vertex Cover. *Discrete Appl. Math.*, 152(1–3):229–245, 2005.

[39] M. Okun and A. Barak. A new approach for approximating node deletion problems. *Inf. Process. Lett.*, 88(5): 231–236, 2003.

[40] E. Prieto and C. Sloper. Looking at the stars. *Theor. Comput. Sci.*, 351(3):437–445, 2006.

[41] S. B. Seidman and B. L. Foster. A graph-theoretic generalization of the clique concept. *J. Math. Sociol.*, 6: 139–154, 1978.

[42] J. Wang, D. Ning, Q. Feng, and J. Chen. An improved parameterized algorithm for a generalized matching problem. In *Proceedings of the 5th Annual Conference on Theory and Applications of Models of Computation (TAMC '08)*, volume 4978 of *LNCS*, pages 212–222. Springer, 2008.