A Generalization of Nemhauser and Trotter's Local Optimization Theorem

Michael R. Fellows, Jiong Guo, Hannes Moser, Rolf Niedermeier

Symposium on Theoretical Aspects of Computer Science 2009 (Freiburg), pp. 409–420
www.stacs-conf.org

A GENERALIZATION OF NEMHAUSER AND TROTTER'S LOCAL OPTIMIZATION THEOREM

MICHAEL R. FELLOWS [1] AND JIONG GUO [2] AND HANNES MOSER [2] AND ROLF NIEDERMEIER [2]

[1] PC Research Unit, Office of DVC (Research),
University of Newcastle, Callaghan, NSW 2308, Australia.
E-mail address: michael.fellows@newcastle.edu.au

[2] Institut für Informatik, Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2, D-07743 Jena, Germany.
E-mail address: (guo,moser,niedermr)@minet.uni-jena.de

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 present a framework that generalizes Nemhauser and Trotter's result to vertex deletion and graph packing problems, introducing novel algorithmic strategies based on purely combinatorial arguments (not referring to linear programming as the Nemhauser-Trotter result originally did).

We exhibit our framework using a generalization of Vertex Cover, called Bounded-Degree Deletion, that has promise to become an important tool in the analysis of gene and other biological networks. For some fixed $d \geq 0$, Bounded-Degree Deletion asks to delete as few vertices as possible 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 Deletion has a problem kernel with a linear number of vertices for every constant d. We also outline an application of our extremal combinatorial approach to the problem of packing stars with a bounded number of leaves. Finally, charting the border between (parameterized) tractability and intractability for Bounded-Degree Deletion, we provide a W[2]-hardness result for Bounded-Degree Deletion in case of unbounded d-values.

1998 ACM Subject Classification: F.2.2, G.2.1, G.2.2, G.2.3.

Key words and phrases: Algorithms, computational complexity, NP-hard problems, W[2]-completeness, graph problems, combinatorial optimization, fixed-parameter tractability, kernelization.

The first author was 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. The second author was supported by the DFG, Emmy Noether research group PIAF, NI 369/4, and project DARE, GU 1023/1. The third author was supported by the DFG, projects ITKO, NI 369/5, and AREG, NI 369/9.

1. Introduction

Nemhauser and Trotter [20] proved a famous theorem in combinatorial optimization. In terms of the NP-hard Vertex Cover[1] problem, it can be formulated as follows:

NT-Theorem [20, 4]. *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) Every vertex cover of the induced subgraph $G[V \setminus (A \cup B)]$ has size at least $|V \setminus (A \cup B)|/2$.

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 for finding a solution. 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. [7] 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 [10, 11, 21] because there is good reason to believe that there is a matching lower bound $2k$ for the kernel size unless P=NP [16].

Since its publication numerous authors have referred to the importance of the NT-Theorem from the viewpoint of polynomial-time approximation algorithms (e.g., [4, 17]) as well as from the viewpoint of parameterized algorithmics (e.g., [1, 7, 9]). The relevance of the NT-Theorem comes from both its practical usefulness in solving the Vertex Cover problem as well as its theoretical depth having led to numerous further studies and follow-up work [1, 4, 9]. In this work, our main contribution is to provide a more general and more widely applicable 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 completely different and are based on extremal combinatorial arguments. Vertex Cover can be formulated as the problem of finding a minimum-cardinality set of vertices whose deletion makes a graph edge-free, that is, the remaining vertices have degree 0. 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. Since the NP-hard Bounded-Degree 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 Deletion (or its dual problem) already appears in some theoretical work, e.g., [6, 18, 22], but so far it has received considerably less attention than Vertex Cover, one of the best studied problems in combinatorial optimization [17]. To advocate and justify more research on Bounded-Degree Deletion,

[1] Vertex Cover is the following problem: Given an undirected graph, find a minimum-cardinality set S of vertices such that each edge has at least one endpoint in S.

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 [3, 8]. Roughly speaking, finding cliques or near-cliques (called paracliques [8]) has been a central tool. Since finding cliques is computationally hard (also with respect to approximation), Chesler et al. [8, 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 [19] (also see [1]) 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. For instance, Chesler et al. [8] 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[2] 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 corresponding problem to find maximum-cardinality s-plexes in a graph is basically as computationally hard as clique detection is [2, 18]. However, as VERTEX COVER is the dual problem for clique detection, BOUNDED-DEGREE DELETION is the dual problem for s-plex detection: An n-vertex graph has an s-plex of size k iff its complement graph has a solution set for BOUNDED-DEGREE DELETION with $d = s-1$ of size $n-k$, and the solution sets can directly be computed from each other. The VERTEX COVER polynomial-time data reduction algorithm has played an important role in the practical success story of analyzing real-world genetic and other biological networks [3, 8]. Our new polynomial-time data reduction algorithms for BOUNDED-DEGREE DELETION have the potential to play a similar role.

Our results. Our main theorem can be formulated as follows.

BDD-DR-Theorem (Theorem 2). *For an undirected n-vertex and m-edge graph $G = (V, E)$, we can compute two disjoint vertex subsets A and B in $O(n^{5/2} \cdot m + n^3)$ time, such that the following three properties hold:*

(1) If S' is a solution set for BOUNDED-DEGREE DELETION *of the induced subgraph $G[V \setminus (A \cup B)]$, then $S := S' \cup A$ is a solution set for* BOUNDED-DEGREE DELETION *of G.*

(2) There is a minimum-cardinality solution set S for BOUNDED-DEGREE DELETION *of G with $A \subseteq S$.*

(3) Every solution set for BOUNDED-DEGREE DELETION *of the induced subgraph $G[V \setminus (A \cup B)]$ has size at least*

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

In terms of parameterized algorithmics, this gives a $(d^3 + 4d^2 + 6d + 4) \cdot k$-vertex problem kernel for BOUNDED-DEGREE DELETION, which is linear in k for constant d-values, thus joining a number of other recent "linear kernelization results" [5, 12, 14, 15]. Our general result specializes to a $4k$-vertex problem kernel for VERTEX COVER (the NT-Theorem provides a size-$2k$ problem kernel), but applies to a larger class of problems.

[2]Introduced in 1978 by Seidman and Foster [24] in the context of social network analysis. Recently, this concept has again found increased interest [2, 18].

For instance, a slightly modified version of the BDD-DR-Theorem (with essentially the same proof) yields a $15k$-vertex 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 [23].[3] For the problem, where, given an undirected graph, one seeks a set of at least k vertex-disjoint stars[4] of the same constant size, we show that a kernel with a linear number of vertices can be achieved, improving the best previous quadratic kernelization [23]. We emphasize that our data reduction technique is based on extremal combinatorial arguments; the resulting combinatorial kernelization algorithm has practical potential and implementation work is underway. 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]). However, for $d \geq 1$ the structure returned by our algorithm is much more complicated; in particular, unlike for 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 Deletion for arbitrary values of the degree value d, we show the following.

Theorem 1. For unbounded d (given as part of the input), Bounded-Degree Deletion is $W[2]$-complete with respect to the parameter k denoting the number of vertices to delete.

In other words, there is no hope for fixed-parameter tractability with respect to the parameter k in the case of unbounded d-values. Due to the lack of space the proof of Theorem 1 and several proofs of lemmas needed to show Theorem 2 are omitted.

2. Preliminaries

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. The central problem of this paper is

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

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. 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 [10, 11, 21] is an approach to finding optimal solutions for NP-hard problems. A common method in parameterized algorithmics is to provide polynomial-time executable *data reduction rules* that lead to a *problem kernel* [13]. This is the most important concept for this paper. Given a parameterized problem instance (I, k), a

[3] Very recently, Wang et al. [25] improved the $15k$-bound to a $7k$-bound. We claim that our kernelization based on the BDD-DR-Theorem method can be easily adapted to also deliver the $7k$-bound.

[4] A *star* is a tree where all of the vertices but one are leaves.

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 $f(k)$ for a function f depending only on k. Roughly speaking, the kernel size $f(k)$ plays a similar role in the subject of problem kernelization as the approximation factor plays for approximation algorithms.

3. A Local Optimization Algorithm for Bounded-Degree Deletion

The main result of this section is the following generalization of the Nemhauser-Trotter-Theorem [20] for Bounded-Degree Deletion with constant d.

Theorem 2 (BDD-DR-Theorem)**.** For an n-vertex and m-edge graph $G = (V, E)$, we can compute two disjoint vertex subsets A and B in $O(n^{5/2} \cdot m + n^3)$ time, such that the following three properties hold:

(1) If S' is a bdd-d-set of $G - (A \cup B)$, then $S := S' \cup A$ is a bdd-d-set of G.
(2) There is a minimum-cardinality bdd-d-set S of G with $A \subseteq S$.
(3) Every bdd-d-set of $G - (A \cup B)$ has size at least $\frac{|V \setminus (A \cup B)|}{d^3+4d^2+6d+4}$.

This first two properties are called the *local optimality conditions*. The remainder of this section is dedicated to the proof of this theorem. More specifically, we present an algorithm called **compute_AB** (see Figure 1) which outputs two sets A and B fulfilling the three properties given in Theorem 2. The core of this algorithm is the procedure **find_extremal** (see Figure 2) running in $O(n^{3/2} \cdot m + n^2)$ time. This procedure returns two disjoint vertex subsets C and D that, among others, satisfy the local optimality conditions. The procedure is iteratively called by **compute_AB**. The overall output sets A and B then are the union of the outputs of all applications of **find_extremal**. Actually, **find_extremal** searches for $C \subseteq V$, $D \subseteq V$, $C \cap D = \emptyset$ satisfying the following two conditions:

C1 Each vertex in $N_G[D] \setminus C$ has degree at most d in $G - C$, and
C2 C is a minimum-cardinality bdd-d-set for $G[C \cup D]$.

It is not hard to see that these two conditions are stronger than the local optimality conditions of Theorem 2:

Lemma 1. Let C and D be two vertex subsets satisfying conditions C1 and C2. Then, the following is true:

(1) If S' is a bdd-d-set of $G - (C \cup D)$, then $S := S' \cup C$ is a bdd-d-set of G.
(2) There is a minimum-cardinality bdd-d-set S of G with $C \subseteq S$.

Lemma 1 will be used in the proof of Theorem 2—it helps to make the description of the underlying algorithm and the corresponding correctness proofs more accessible. As a direct application of Theorem 2, we get the following corollary.

Corollary 1. Bounded-Degree Deletion with constant d admits a problem kernel with at most $(d^3 + 4d^2 + 6d + 4) \cdot k$ vertices, which is computable in $O(n^{5/2} \cdot m + n^3)$ time.

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" in G.

Algorithm: compute_AB (G)
Input: An undirected graph G.
Output: Vertex subsets A and B satisfying the three properties of Theorem 2.

1 $A := \emptyset, B := \emptyset$
2 Compute a witness X and the corresponding residual $Y := V \setminus X$ for G
3 **If** $|Y| \leq (d+1)^2 \cdot |X|$ **then return** (A, B)
4 $(C, D) \leftarrow$ **find_extremal** (G, X, Y).
5 $G \leftarrow G - (C \cup D); A \leftarrow A \cup C; B \leftarrow B \cup D;$ **goto** line 2

Figure 1: Pseudo-code of the main algorithm for computing A and B.

Definition 3.1. For $s \geq 1$, the graph $K_{1,s} = (\{u, v_1, \ldots, v_s\}, \{\{u, v_1\}, \ldots, \{u, v_s\}\})$ is called an *s-star*. The vertex u is called the *center* of the star. The vertices $v_1, \ldots, v_s$ are the *leaves* of the star. A *$\leq s$-star* is an s'-star with $s' \leq s$.

Due to this forbidden subgraph characterization of bounded-degree graphs, we can also derive a linear kernelization for the $(d+1)$-Star Packing problem. In this problem, given an undirected graph, one seeks for at least k vertex-disjoint $(d+1)$-stars for a constant d. With a slight modification of the proof of Theorem 2, we get the following corollary.

Corollary 2. $(d+1)$-Star Packing admits a problem kernel with at most $(d^3 + 4d^2 + 6d + 4) \cdot k$ vertices, which is computable in $O(n^{5/2} \cdot m + n^3)$ time.

For $d \geq 2$, the best known kernelization result was a $O(k^2)$ kernel [23]. Note that the special case of $(d+1)$-Star Packing with $d = 1$ is also called P_3-Packing, a problem well-studied in the literature, see [23, 25]. Corollary 2 gives a $15k$-vertex problem kernel. The best-known bound is $7k$ [25]. However, the improvement from the formerly best bound $15k$ [23] is achieved by improving a properly defined witness structure by local modifications. This trick also works with our approach, that is, we can show that the NT-like approach also yields a $7k$-vertex problem kernel for 2-Star Packing.

3.1. The Algorithm

We start with an informal description of the algorithm. As stated in the introduction of this section, the central part is Algorithm **compute_AB** shown in Figure 1.

Using the characterization of bounded-degree graphs by forbidding large stars, in line 2 **compute_AB** starts with computing two vertex sets X and Y: First, 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 the 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 $Y := V \setminus X$. We call X the *witness* and Y the corresponding *residual*.

If the residual Y is too big (condition in line 3), the sets X and Y are passed in line 4 to the procedure **find_extremal** in Figure 2 which computes two sets C and D satisfying conditions C1 and C2. Computing X and Y represents the first step to find a subset pair satisfying condition C1: Since there is no vertex that has degree more than d in $G - X$ (due

Procedure: find_extremal (G, X, Y)
Input: An undirected graph G, witness X, and residual Y.
Output: Vertex subsets C and D satisfying the local optimality conditions.

1 $J \leftarrow$ bipartite graph with X and Y as its two vertex subsets and $E(J) \leftarrow \{\{u, v\} \in E(G) \mid u \in X \text{ and } v \in Y\}$
2 $F_0^X \leftarrow \emptyset$ ▷ *Initialize empty set of forbidden vertices*
3 start with $j = 0$ and **while** $F_j^X \neq X$ **do** ▷ *Loop while not all vertices in X are forbidden*
4 $F_j^Y \leftarrow N_G[N_J(F_j^X)] \setminus X$ ▷ *Determine forbidden vertices in Y*
5 $P \leftarrow$ **star-packing**$(J - (F_j^X \cup F_j^Y), X \setminus F_j^X, Y \setminus F_j^Y, d)$
6 $D_0 \leftarrow Y \setminus (F_j^Y \cup V(P))$ ▷ *Vertices in Y that are not forbidden and not in P*
7 start with $i = 0$ and **repeat** ▷ *Start search for C, D satisfying C2*
8 $C_i \leftarrow N_J(D_i)$
9 $D_{i+1} \leftarrow N_P(C_i) \cup D_i$
10 $i \leftarrow i + 1$
11 **until** $D_i = D_{i-1}$
12 $C \leftarrow C_i$, $D \leftarrow D_i$
13 **if** $C = X \setminus F_j^X$ **then** ▷ *C, D also satisfy C1*
14 **return** (C, D)
15 $F_{j+1}^X \leftarrow X \setminus C$ ▷ *Determine forbidden vertices in X for next iteration*
16 $j \leftarrow j + 1$
17 **end while**
18 $F_j^Y \leftarrow N_G[N_J(F_j^X)] \setminus X$ ▷ *Recompute forbidden vertices in Y (as in line 4)*
19 **return** $(\emptyset, V \setminus (X \cup F_j^Y))$

Procedure: star-packing (J, V_1, V_2, d)
Input: A bipartite graph J with two vertex subsets V_1 and V_2.
Output: A maximum-edge packing of stars that have their centers in V_1 and have at most $d + 1$ leaves in V_2.
See Lemma 2, the straightforward implementation details using matching techniques are omitted.

Figure 2: Pseudo-code of the procedure computing the intermediary vertex subset pair (C, D).

to the fact that X is a bdd-d-set), the search is limited to those subset pairs where C is a subset of the witness X and D is a subset of Y.

Algorithm **compute_AB** calls **find_extremal** iteratively until the sets A and B, which are constructed by the union of the outputs of all applications of **find_extremal** (see line 5), satisfy the third property in Theorem 2. In the following, we intuitively describe the basic ideas behind **find_extremal**.

To construct the set C from X, we compute again a star packing P with the centers of the stars being from X and the leaves being from Y. We relax, on the one hand, the requirement that the stars in the packing have exactly $d + 1$ leaves, that is, the packing P might contain $\leq d$-stars. On the other hand, P should have a maximum number of edges.

The rough idea behind the requirement for a maximum number of edges is to maximize the number of $(d+1)$-stars in P in the course of the algorithm. Moreover, we can observe that, by setting C equal to the center set of the $(d+1)$-stars in P and D equal to the leaf set of the $(d+1)$-stars in P, C is a minimum bdd-d-set of $G[C \cup D]$ (condition C2). We call such a packing a *maximum-edge X-center $\leq (d+1)$-star packing*. For computing P, the algorithm constructs an auxiliary bipartite graph J with X as one vertex subset and Y as the other. The edge set of J consists of the edges in G with exactly one endpoint in X. See line 1 of Figure 2. Obviously, a maximum-edge X-center $\leq (d+1)$-star packing of G corresponds one-to-one with a maximum-edge packing of stars in J that have their centers in X and have at most $d+1$ leaves in the other vertex subset. Then, the star packing P can be computed by using techniques for computing maximum matchings in J (in the following, let **star-packing**(J,V_1,V_2,d) denote an algorithm that computes a maximum-edge V_1-center $\leq (d+1)$-star packing P on the bipartite graph J).

The most involved part of **find_extremal** in Figure 2 is to guarantee that the output subsets in line 4 fulfill condition C1. To this end, one uses an iterative approach to compute the star packing P. Roughly speaking, in each iteration, if the subsets C and D do not fulfill condition C1, then exclude from further iterations the vertices from D that themselves or whose neighbors violate this condition. See lines 2 to 15 of Figure 2 for more details of the iterative computation. Herein, for $j \geq 0$, the sets $F_j^X \subseteq X$ and $F_j^Y \subseteq Y$, where F_j^X is initialized with the empty set, and F_j^Y is computed using F_j^X, store the vertices excluded from computing P. To find the vertices that themselves cause the violation of the condition, that is, vertices in D that have neighbors in $X \setminus C$, one uses an augmenting path computation in lines 7 to 11 to get in line 12 subsets C and D such that the vertices in D do *not* themselves violate the condition. Roughly speaking, the existence of an edge e from some vertex in D to some vertex in $X \setminus C$ would imply that the $\leq (d+1)$-star packing is not maximum (witnessed by an augmenting path beginning with e—in principle, this idea is also used for finding crown decompositions, cf. [1]). The vertices whose neighbors cause the violation of condition C1 are all vertices in D with neighbors in $Y \setminus D$ that themselves have neighbors in $X \setminus C$. These neighbors in $Y \setminus D$ and the corresponding vertices in D are excluded in line 4 and line 18. We will see that the number of all excluded vertices is $O(|X \setminus C|)$, thus, in total, we do not exclude too many vertices with this iterative method. The formal proof of correctness is given in the following subsection.

3.2. Running Time and Correctness

Now, we show that **compute_AB** in Figure 1 computes in the claimed time two vertex subsets A and B that fulfill the three properties given in Theorem 2.

3.2.1. *Running Time of **find_extremal**.* We begin with the proof of the running time of the procedure **find_extremal** in Figure 2, which uses the following lemmas.

Lemma 2. Procedure **star-packing**(J, V_1, V_2, d) in Figure 2 runs in $O(\sqrt{n} \cdot m)$ time.

The next lemma is also used for the correctness proof; in particular, it guarantees the termination of the algorithm.

Lemma 3. If the condition in line 13 of Figure 2 is false for a $j \geq 0$, then $F_j^X \subsetneq F_{j+1}^X$.

Proof. In lines 4 and 5 of Figure 2, all vertices in F_j^X and their neighbors $N_J(F_j^X)$ are excluded from the star packing P in the jth iteration of the outer loop. Moreover, the vertices in $N_J(F_j^X)$ are excluded from the set D_0 (line 6). Therefore, a vertex in F_j^X cannot be added to C in line 12. Thus F_{j+1}^X (set to $X \setminus C$ in line 15) contains F_j^X. Moreover, this containment is proper, as otherwise the condition in line 13 would be true. ∎

Lemma 4. Procedure **find_extremal** runs in $O(n^{3/2} \cdot m + n^2)$ time.

3.2.2. *Correctness of **find_extremal**.* The correctness proof for **find_extremal** in Figure 2 is more involved than its running time analysis. The following lemmas provide some properties of (C, D) which are needed.

Lemma 5. For each $j \geq 0$ the following properties hold after the execution of line 12 in Figure 2:

(1) every vertex in C is a center vertex of a $(d+1)$-star in P, and
(2) the leaves of every star in P with center in C are vertices in D.

Proof. (Sketch) To prove (1), first of all, we show that $v \in C$ implies $v \in V(P)$, since, otherwise, we could get a P-augmenting path from some element in D_0 to v. A P-augmenting path is a path where the edges in $E(P)$ and the edges not in $E(P)$ alternate, and the first and the last edge are not in $E(P)$. This P-augmenting path can be constructed in an inductive way by simulating the construction of C_i in lines 6 to 11 of Figure 2. From this P-augmenting path, we can then construct a X-center $\leq (d+1)$-star packing that has more edges than P, contradicting that $E(P)$ has maximum cardinality. Second, every vertex in C is a center of a star due to the definition of P and Procedure **star-packing**. Finally, if a vertex $v \in C$ is the center of a star with less than $(d+1)$ leaves, then again we get a P-augmenting path from some element in D_0 to v.

The second statement follows easily from Procedure **star-packing** and the pseudo-code in lines 6 to 12. ∎

Lemma 6. For each $j \geq 0$ there is no edge in G between D and $N_J(F_j^X)$.

Proof. The vertices in F_j^X and the vertices in $N_G[N_J(F_j^X)] \setminus X$ are excluded from the computation of P and are not contained in D_0 (lines 4 to 6 in Figure 2). Thus, $N_J[F_j^X] \cap D = \emptyset$ and therefore there are no edges in G between D and $N_J(F_j^X)$. ∎

The next lemma shows that the output of **find_extremal** fulfills the local optimality conditions.

Lemma 7. Procedure **find_extremal** returns two disjoint vertex subsets fulfilling conditions C1 and C2.

Proof. Clearly, the output consists of two disjoints sets. The algorithm returns in lines 14 or 19 of Figure 2. If it returns in line 19, then the output C is empty and D contains only vertices that have a distance at least 3 to the vertices in X: The condition in line 3 implies $F_j^X = X$ and, therefore, F_j^Y contains all vertices in $G \setminus X$ that have distance at most 2 to the vertices in X. Since X is a bdd-d-set of G, all vertices in D and their neighbors in G have a degree at most d. This implies that both conditions hold for the output returned in this line. It remains to consider the output returned in line 14.

To show that condition C1 holds, recall that $G-X$ has maximum degree d and that $C \subseteq X$. Therefore, if for a vertex v in $V \setminus X$ we have $N_J(v) \subseteq C$, then v has degree at most d in $G - C$. Thus, to show that each vertex in $N_G[D] \setminus C$ has degree at most d in $G - C$, it suffices to prove that $N_J(N_G[D] \setminus C) \subseteq C$. We show separately that $N_J(D) \subseteq C$ and that $N_J(N_G(D) \setminus C) \subseteq C$.

The assignment in line 8 and the until-condition in line 11 directly give $N_J(D) \subseteq C$. Due to Lemma 6 there is no edge in G between D and $N_J(F_j^X)$, where $F_j^X = X \setminus C$ (the if-condition in line 13, which has to be satisfied for the procedure to return in line 14). From this it follows that the vertices in $N_G(D) \setminus C$ have no vertex in F_j^X as neighbor and, thus, $N_J(N_G(D) \setminus C) \cap F_j^X = \emptyset$. Therefore, $N_J(N_G(D) \setminus C) \subseteq C$.

By Properties 1 and 2 of Lemma 5, there are exactly $|C|$ many vertex-disjoint $(d+1)$-stars in $G[C \cup D]$. Moreover, there is no $(d+1)$-star in $G[D]$, since X is a bdd-d-set of G. Thus, C is a minimum-cardinality bdd-d-set of $G[C \cup D]$. ∎

3.2.3. *Running Time and Correctness of* ***compute_AB***. To prove the running time and correctness of **compute_AB**, we have to show that the output of **find_extremal** contains sufficiently many vertices of Y. To this end, the following lemma plays a decisive role.

Lemma 8. For all $j \geq 0$, the set F_j^Y in line 4 and line 18 of Figure 2 has size at most $(d+1)^2 \cdot |F_j^X|$.

Proof. The proof is by induction on j. The claim trivially holds for $j = 0$, since $F_0^Y = \emptyset$. Assume that the claim is true for $j > 0$. Since $F_j^X \subsetneq F_{j+1}^X$ (Lemma 3), we have

$$F_{j+1}^Y = F_j^Y \cup N_{G-X}[N_{J-F_j^Y}(F_{j+1}^X \setminus F_j^X)].$$

We first bound the size of $N_{J-F_j^Y}(F_{j+1}^X \setminus F_j^X)$. Since F_{j+1}^X was set to $X \setminus C$ at the end of the jth iteration of the outer loop (line 15), the vertices in $N_{J-F_j^Y}(F_{j+1}^X \setminus F_j^X)$ were not excluded from computing the packing P (line 5) of the jth iteration. Moreover, $N_{J-F_j^Y}(F_{j+1}^X \setminus F_j^X) \subseteq V(P)$ for the star packing P computed in the jth iteration, since, otherwise, the set D_0 in line 6 would contain a vertex v in $N_{J-F_j^Y}(F_{j+1}^X \setminus F_j^X)$ and, then, line 8 would include $N_J(v)$ into C, which would contradict the fact that $C \cap F_{j+1}^X = \emptyset$ (line 15). Due to property 2 in Lemma 5 the leaves of every star in P with center in C are vertices in D and, thus, the vertices in $N_{J-F_j^Y}(F_{j+1}^X \setminus F_j^X)$ are leaves of stars in P with centers in $F_{j+1}^X \setminus F_j^X$. Since each star has at most $(d+1)$ leaves, the set $N_{J-F_j^Y}(F_{j+1}^X \setminus F_j^X)$ has size at most $(d+1) \cdot |F_{j+1}^X \setminus F_j^X|$. The remaining part is easy to bound: since all the vertices in $V \setminus X$ have degree at most d, we get

$$\begin{aligned}\left|N_{G-X}[N_{J-F_j^Y}(F_{j+1}^X \setminus F_j^X)]\right| &\leq (d \cdot (d+1) + (d+1)) \cdot |F_{j+1}^X \setminus F_j^X| \\ &= (d+1)^2 \cdot |F_{j+1}^X \setminus F_j^X|.\end{aligned}$$

With the induction hypothesis, we get that

$$\begin{aligned}|F_{j+1}^Y| &\leq |F_j^Y| + |N_{G-X}[N_{J-F_j^Y}(F_{j+1}^X \setminus F_j^X)]| \\ &= (d+1)^2 \cdot |F_j^X| + (d+1)^2 \cdot |F_{j+1}^X \setminus F_j^X| = (d+1)^2 \cdot |F_{j+1}^X|.\end{aligned}$$

∎

Lemma 9. Procedure **find_extremal** always finds two sets C and D such that $|Y \setminus D| \leq (d+1)^2 \cdot |X \setminus C|$.

Proof. If **find_extremal** terminates, then $V' = F_j^X \cup F_j^Y$ for the graph $G' = (V', E')$ resulting by removing $C \cup D$ from G. Since $C \subseteq X$ and $D \subseteq Y$, we have $X \setminus C = F_j^X$ and $Y \setminus D = F_j^Y$, and by Lemma 8 it follows immediately that $|Y \setminus D| \leq (d+1)^2 \cdot |X \setminus C|$. ∎

Therefore, if $|Y| > (d+1)^2 \cdot |X|$, then **find_extremal** always returns two sets C and D such that D is not empty.

Lemma 10. Algorithm **compute_AB** runs in $O(n^{5/2} \cdot m + n^3)$ time.

Lemma 11. The sets A and B computed by **compute_AB** fulfill the three properties given in Theorem 2.

Proof. Since every (C, D) output by **find_extremal** in line 4 of **compute_AB** in Figure 1 fulfills conditions C1 and C2 (Lemma 7), the pair (A, B) output in line 3 of **compute_AB** fulfills conditions C1 and C2, and, therefore, also the local optimality conditions (Lemma 1). It remains to show that (A, B) fulfills the size condition.

Let X and Y be the last computed witness and residual, respectively. Since the condition in line 3 is true, we know that $|Y| \leq (d+1)^2 \cdot |X|$. Recall that X is a factor-$(d+2)$ approximate bdd-d-set for $G' := G - (A \cup B)$. Thus, every bdd-d-set of G' has size at least $|X|/(d+2)$. Since the output sets A and B fulfill the local optimality conditions and the bounded-degree property is hereditary, every bdd-d-set of G' has size at least

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

The inequality (*) follows from the fact that Y is small, that is, $|Y| \leq (d+1)^2 \cdot |X|$ (note that $V' = X \cup Y$). ∎

With Lemmas 10 and 11, the proof of Theorem 2 is completed.

4. Conclusion

Our main result is to generalize the Nemhauser-Trotter-Theorem, which applies to the Bounded-Degree 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 problem kernels with a number of vertices linear in the solution size k for all constant values of d for Bounded-Degree Deletion. To this end, we developed a new algorithmic strategy that is based on extremal combinatorial arguments. The original NT-Theorem [20] has been proven using linear programming relaxations—we see no way how this could have been generalized to Bounded-Degree 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, 4, 9]). Finally, Baldwin et al. [3, 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". Hence, we expect that benefits similar to those derived from Vertex Cover kernelization for biological network analysis (see the motivation part of our introductory discussion) may be provided by Bounded-Degree Deletion kernelization.

References

[1] 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.

[2] B. Balasundaram, S. Butenko, I. V. Hicks, and S. Sachdeva. Clique relaxations in social network analysis: The maximum k-plex problem. Manuscript, 2008.

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

[4] 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.

[5] H. L. Bodlaender and E. Penninkx. A linear kernel for planar feedback vertex set. In *Proc. 3rd IWPEC*, volume 5018 of *LNCS*, pages 160–171. Springer, 2008.

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

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

[8] 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. *Nature Genetics*, 37(3):233–242, 2005.

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

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

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

[12] J. Guo. A more effective linear kernelization for cluster editing. *Theor. Comput. Sci.*, 2008. To appear.

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

[14] J. Guo and R. Niedermeier. Linear problem kernels for NP-hard problems on planar graphs. In *Proc. 34th ICALP*, volume 4596 of *LNCS*, pages 375–386. Springer, 2007.

[15] I. A. Kanj, M. J. Pelsmajer, G. Xia, and M. Schaefer. On the induced matching problem. *J. Comput. System Sci.*, 2009. To appear.

[16] 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.

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

[18] C. Komusiewicz, F. Hüffner, H. Moser, and R. Niedermeier. Isolation concepts for enumerating dense subgraphs. In *Proc. 13th COCOON*, volume 4598 of *LNCS*, pages 140–150. Springer, 2007.

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

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

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

[22] 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.

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

[24] S. B. Seidman and B. L. Foster. A graph-theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, 6:139–154, 1978.

[25] J. Wang, D. Ning, Q. Feng, and J. Chen. An improved parameterized algorithm for a generalized matching problem. In *Proc. 5th TAMC*, volume 4978 of *LNCS*, pages 212–222. Springer, 2008.