On a Generalization of Nemhauser and Trotter's Local Optimization Theorem

Mingyu Xiao(✉)

School of Computer Science and Engineering,
University of Electronic Science and Technology of China, Chengdu, China
myxiao@gmail.com

Abstract. The Nemhauser and Trotter's theorem applies to the famous Vertex Cover problem and can obtain a 2-approximation solution and a problem kernel of $2k$ vertices. This theorem is a famous theorem in combinatorial optimization and has been extensively studied. One way to generalize this theorem is to extend the result to the Bounded-Degree Vertex Deletion problem. For a fixed integer $d \geq 0$, Bounded-Degree Vertex Deletion asks to delete at most k vertices of the input graph to make the maximum degree of the remaining graph at most d. Vertex Cover is a special case that $d = 0$. Fellows, Guo, Moser and Niedermeier proved a generalized theorem that implies an $O(k)$-vertex kernel for Bounded-Degree Vertex Deletion for $d = 0$ and 1, and for any $\varepsilon > 0$, an $O(k^{1+\varepsilon})$-vertex kernel for each $d \geq 2$. In fact, it is still left as an open problem whether Bounded-Degree Vertex Deletion parameterized by k admits a linear-vertex kernel for each $d \geq 3$. In this paper, we refine the generalized Nemhauser and Trotter's theorem. Our result implies a linear-vertex kernel for Bounded-Degree Vertex Deletion parameterized by k for each $d \geq 0$.

1 Introduction

Vertex Cover, to find a minimum set of vertices in a graph such that each edge in the graph is incident on at least one vertex in this set, is one of the most fundamental problems in graph algorithms, graph theory, parameterized algorithms, theories of NP-completeness and many others. Nemhauser and Trotter [22] proved a famous theorem (NT-Theorem) for Vertex Cover.

Theorem 1 [**NT – Theorem**]. *For an undirected graph $G = (V, E)$ of $n = |V|$ vertices and $m = |E|$ edges, there is an $O(\sqrt{n}m)$-time algorithm to compute two disjoint vertex subsets C and I of G such that for any minimum vertex cover K' of the induced subgraph $G[V \setminus (C \cup I)]$, $K' \cup C$ is a minimum vertex cover of G and*

$$|K'| \geq \frac{|V \setminus (C \cup I)|}{2}.$$

M. Xiao—Supported by NFSC of China under the Grant 61370071.

K. Elbassioni and K. Makino (Eds.): ISAAC 2015, LNCS 9472, pp. 442–452, 2015.
DOI: 10.1007/978-3-662-48971-0_38

This theorem provides a polynomial-time algorithm to reduce the size of the input graph by possibly finding partial solution. It turns out that NT-Theorem has great applications in approximation algorithms [5,17,19] and parameterized algorithms [2,7]. We can see that $V \setminus I$ is a 2-approximation solution and $G[V \setminus (C \cup I)]$ is a $2k$-vertex kernel of the problem taking the size of the solution as the parameter k. Lokshtanov et al. [21] also apply NT-Theorem to branching algorithms for Vertex Cover and some other related problems. Due to NT-Theorem's practical usefulness and theoretical depth in graph theory, it has attracted numerous further studies and follow-up work [2,4,9,14]. Bar-Yehuda, Rawitz and Hermelin [4] extended NT-Theorem for a generalized vertex cover problem, where edges are allowed not to be covered at a certain predetermined penalty. Fellows, Guo, Moser and Niedermeier [14] extended NT-Theorem for Bounded-Degree Vertex Deletion.

In this paper, we are interested in Bounded-Degree Vertex Deletion. A *d-degree deletion set* of a graph G is a subset of vertices, whose deletion leaves a graph of maximum degree at most d. For each fixed d, Bounded-Degree Vertex Deletion is to find a d-degree deletion set of minimum size in an input graph. Bounded-Degree Vertex Deletion and its "dual problem" to find maximum s-plexes have applications in computational biology [8,14] and social network analysis [3,24]. There is a substantial amount of theoretical work on this problem [20,23,24], specially in parameterized complexity [6,8,14].

Since Vertex Cover is a special case of Bounded-Degree Vertex Deletion, we are interested in finding a local optimization theorem similar to NT-Theorem for Bounded-Degree Vertex Deletion. Fellows, Guo, Moser and Niedermeier [14] made a great progress toward to this interesting problem by giving the following theorem.

Theorem 2 [14]**.** *For an undirected graph $G = (V, E)$ of $n = |V|$ vertices and $m = |E|$ edges, any constant $\varepsilon > 0$ and any integer $d \geq 0$, there is an $O(n^4 m)$-time algorithm to compute two disjoint vertex subsets C and I of G such that for any minimum d-degree deletion set K' of the induced subgraph $G[V \setminus (C \cup I)]$, $K' \cup C$ is a minimum d-degree deletion set of G, and*

$$|K'| \geq \frac{|V \setminus (C \cup I)|}{d^3 + 4d^2 + 6d + 4} \qquad \text{for } \ d \leq 1, \qquad \text{and}$$

$$|K'|^{1+\varepsilon} \geq \frac{|V \setminus (C \cup I)|}{c} \qquad \text{for } \ d \geq 2,$$

where c is a function of d and ε.

In this theorem, for $d \geq 2$, the number of remaining vertices in $V \setminus (C \cup I)$ is not bounded by a constant times of the solution size $|K'|$ of $G[V \setminus (C \cup I)]$. This is a significant difference between this theorem and the NT-Theorem for Vertex Cover. In terms of parameterized algorithms, Theorem 2 cannot get a linear-vertex kernel for Parameterized Bounded-Degree Vertex Deletion (with parameter k being the solution size) for each $d \geq 2$. In fact, in an initial version [15] of Fellows, Guo, Moser and Niedermeier's paper, a better result was claimed, which can get a linear-vertex kernel for PARAMETERIZED BOUNDED-DEGREE VERTEX DELETION for each $d \geq 0$. Unfortunately, the proof in [15] is incomplete. We also note that Chen et al. [8] proved a $37k$-vertex kernel for BOUNDED-DEGREE VERTEX DELETION for $d = 2$. However, whether BOUNDED-DEGREE VERTEX DELETION for each $d \geq 3$ allows a linear-vertex kernel is not known. In this paper, based on Fellows, Guo, Moser and Niedermeier's work [15], we close the above gap by proving the following theorem for BOUNDED-DEGREE VERTEX DELETION.

Theorem 3 [**Our result**]**.** *For an undirected graph $G = (V, E)$ of $n = |V|$ vertices and $m = |E|$ edges and any integer $d \geq 0$, there is an $O(n^{5/2}m)$-time algorithm to compute two disjoint vertex subsets C and I of G such that for any minimum d-degree deletion set K' of the induced subgraph $G[V \setminus (C \cup I)]$, $K' \cup C$ is a minimum d-degree deletion set of G and*

$$|K'| \geq \frac{|V \setminus (C \cup I)|}{d^3 + 4d^2 + 5d + 3}.$$

From this version of the generalized Nemhauser and Trotter's theorem, we can get a $(d^3 + 4d^2 + 5d + 3)k$-vertex kernel for BOUNDED-DEGREE VERTEX DELETION parameterized by the size k of the solution, which is linear in k for any constant $d \geq 0$. There is no difference between the cases that $d \leq 1$ and $d \geq 2$ anymore. For the special case that $d = 0$, our theorem specializes a $3k$-vertex kernel for VERTEX COVER, while Theorem 2 provides a $4k$-vertex kernel and NT-Theorem provides a $2k$-vertex kernel. For the special case that $d = 1$, our theorem provides a $13k$-vertex kernel and Theorem 2 provides a $15k$-vertex kernel. For the special case that $d = 2$, our theorem obtains a $37k$-vertex kernel, the same result obtained by Chen et al. [8].

Recently, Dell and van Melkebeek [12] showed that unless the polynomial-time hierarchy collapses, PARAMETERIZED BOUNDED-DEGREE VERTEX DELETION does not have kernels consisting of $O(k^{2-\epsilon})$ edges for any constant $\epsilon > 0$, which implies that linear size would be the best possible bound on the number of vertices in any kernel for this problem. It has also been proved by Fellows, Guo, Moser and Niedermeier [14] that when d is not bounded, PARAMETERIZED BOUNDED-DEGREE VERTEX DELETION is W[2]-hard. Then unless FPT = W[2], it is impossible to remove d from the size function of any kernel of this problem. These two hardness results also imply that our result is 'tight' in some sense.

The framework of our algorithm follows that of Fellows, Guo, Moser and Niedermeier's algorithm [14]. But we still need some new and nontrivial ideas to get our result. For the purpose of presentation, we will define a decomposition, called 'd-bounded decomposition' to prove Theorem 3 and construct our algorithms. This decomposition can be regarded as an extension of the crown decomposition for VERTEX COVER [1,10], but more sophisticated. To compute C and I in Theorem 3, we will change to compute a proper d-bounded decomposition. Some similar ideas in construction of crown decompositions as in Fellows, Guo, Moser and Niedermeier's algorithm for Theorem 2 [14] are used to construct our decomposition. The detailed differences between our and previous algorithms will be addressed in Sect. 4. Before introducing the decompositions, we first give the notation system in this paper. Proofs of some lemmas are omitted due to space limitation, which can be found in the full version of this paper.

2 Notation System

Let $G = (V, E)$ stand for a simple undirected graph with a set V of $n = |V|$ vertices and a set E of $m = |E|$ edges. For simplicity, we may denote a singleton set $\{v\}$ by v. For a vertex subset V', a vertex in V' is denoted by V'*-vertex*. The graph induced by V' is denoted by $G[V']$. We also use $N(V')$ to denote the set of vertices in $V \setminus V'$ adjacent to some vertices in V' and let $N[V'] = N(V') \cup V'$. The vertex set and edge set of a graph G' are denoted by $V(G')$ and $E(G')$, respectively. A bipartite graph with two parts of vertices A and B and edge set E_H is denoted by $H = (A, B, E_H)$.

For an integer $d' \geq 1$, a star with $d'+1$ vertices is called a d'*-star*. For $d' > 1$, the unique vertex of degree > 1 in a d'-star is called the *center* of the star and all other degree-1 vertices are called the *leaves* of the star. For a 1-star, any vertex can be regarded as a *center* and the other vertex as a *leaf*. A star with a center v is also called a star *centered at* v. For two disjoint vertex sets V_1 and V_2, a set of stars is *from* V_1 *to* V_2 if the centers of the stars are in V_1 and leaves are in V_2. A $_\leq d'$*-star* is a star with at most d' leaves. A d'*-star packing* (resp., $_\leq d'$*-star packing*) is a set of vertex-disjoint d'-stars (resp., $_\leq d'$-stars). We will use $\alpha(G)$ to denote the size of a minimum d-degree deletion set of a graph G.

3 The Decomposition Techniques

Crown decomposition is a powerful tool to obtain kernels for Vertex Cover. This technique was firstly introduced in [1,10] and found to be very useful in designing kernelization algorithms for Vertex Cover and related problems [2,9,26].

Definition 1 [**Crown Decomposition**]**.** *A crown decomposition of a graph G is a partition of the vertex set of G into three sets I, C and J such that*

(1) I is an independent set,
(2) there are no edges between I and J, and
(3) there is a matching M on the edges between I and C such that all vertices in C are matched.

See Fig. 1(a) for an illustration for crown decompositions. In some references, $I \neq \emptyset$ is also required in the definition of crown decompositions. Here we allow $I = \emptyset$ for the purpose of presentation. It is known that

Lemma 1 [1]**.** *Let (I, C, J) be a crown decomposition of G. Then (I, C) satisfies the local optimality condition in Theorem 1, i.e., $K' \cup C$ is a minimum vertex cover of G for any minimum vertex cover K' of the induced subgraph $G[V \setminus (I \cup C)]$.*

(a): A crown decomposition (I, C, J).

(b): A 3-bounded decomposition (I, C, T, J).

Fig. 1. Decompositions

By this lemma, we can reduce the instance of Vertex Cover by removing $I \cup C$ of a crown decomposition. There are some methods that find certain crown decompositions of a graph and result in a linear-vertex kernel for Vertex Cover [2].

In this paper, we will use *d-bounded decomposition*, which extends the definition of crown decompositions and Lemma 1. Let A and B be two disjoint vertex subsets of a graph G. A *full d'-star packing from A to B* is a set of $|A|$ vertex-disjoint d'-stars with centers in A and leaves in B. The third item in Definition 1 means that there is a full 1-star packing from C to I. We define the following decomposition.

Definition 2 [$d-$**BoundedDecomposition**]**.** *A d-bounded decomposition of a graph $G = (V, E)$ is a partition of the vertex set of G into four sets I, C, T and J such that*

(1) any vertex in $I \cup T$ is of degree $\leq d$ in the induced subgraph $G[V \setminus C]$,
(2) there are no edges between I and J, and
(3) there is a full $(d+1)$-star packing from C to I.

An illustration for d-bounded decompositions is given in Fig. 1(b). We have the following Lemma 2 for d-bounded decompositions. This lemma can be derived from the lemmas in [14], although d-bounded decomposition is not formally defined in [14].

Lemma 2. *Let (I, C, T, J) be a d-bounded decomposition of G. Then (I, C) satisfies the local optimality condition in Theorem 3, i.e., $K' \cup C$ is a minimum d-degree deletion set of G for any minimum d-degree deletion set K' of the induced subgraph $G[V \setminus (I \cup C)]$.*

By Lemma 2, we can reduce an instance by removing $I \cup C$ if the graph has a d-bounded decomposition (I, C, T, J). This is the main idea how we get Theorem 3 and kernels for our problem. Here arises a problem how to find a d-bounded decomposition (I, C, T, J) of a graph such that $I \neq \emptyset$ if it exists. First, we give a simple observation.

Observation 1. *Let R be a set of vertices v such that any vertex in $N[v]$ is of degree $\leq d$. Then $(I = R, C = \emptyset, T = N(R), J = V \setminus (I \cup T))$ is a d-bounded decomposition of G.*

By Lemma 2 and Observation 1, we can reduce an instance by removing from the graph the set B of vertices v such that any vertex in $N[v]$ is of degree $\leq d$. We will introduce an algorithm that can find more d-bounded decompositions.

4 Algorithms

We first introduce an algorithm to find d-bounded decompositions of graphs, based on which we can easily get an algorithm for the generalization of NT-theorem in Theorem 3.

4.1 The Algorithm for Decompositions

First of all, we give the main idea of our algorithm to find a d-bounded decomposition (I, C, T, J) of a graph $G = (V, E)$. It contains three major phases.
Phase 1: find a partition (X, Y) of the vertex set V such that the maximum degree in $G[Y]$ is at most d.
Phase 2: find two subsets $C' \subseteq X$ and $I' \subseteq Y$ satisfying *Basic Condition*: there is a full $(d+1)$-star packing from C' to I' and there is no edge between I' and $X \setminus C'$.
Phase 3: iteratively move some vertices out of I' and some vertices out of C' to make $(I', C', T' = N(I') \setminus C', J' = V \setminus (I' \cup C' \cup T'))$ a d-bounded decomposition.

In fact, the first two phases of our algorithm are almost the same as that of Fellows, Guo, Moser and Niedermeier's algorithm [14]. However, in Phase 3, our algorithm uses a different method to compute I' and C'. This is critical for us to get an improvement.

Phase 1. For Phase 1, we can find a maximal $(d+1)$-star packing S and let $X = V(S)$. By the maximality of S, we know that X is a d-degree deletion set and $G[Y]$ has no vertex of degree $> d$. Then the partition (X, Y) satisfies the condition in Phase 1. In order to obtain a good performance, our algorithm may not use an arbitrary maximal $(d+1)$-star packing S. When we obtain a new $(d+1)$-star packing S' such that $|S'| > |S|$ in our algorithm, we will update X by letting $X = V(S')$.

Phase 2. After obtaining (X, Y) in Phase 1, our algorithm finds two special sets $C' \subseteq X$ and $I' \subseteq Y$ in Phase 2. To find C' and I' satisfying Basic Condition, we need to find a special ${}_{\leq}(d+1)$-star packing from X to Y, which can be computed by the algorithms for finding maximum matchings in bipartite graphs. Note that the idea of computing ${}_{\leq}(d+1)$-stars from X and Y has been used to solve some other problems in references [11,16,25].

We consider the bipartite graph $H = (X, Y, E_H)$ with edge set E_H being the set of edges between X and Y in G, and are going to find a ${}_{\leq}(d+1)$-star packing from X to Y in H. Note that a Y-vertex no adjacent to any vertex in X will become a degree-0 vertex in H. We construct an auxiliary bipartite graph $H' = (X_1 \cup X_2 \cup \dots X_{d+1}, Y, E'_H)$, where each X_i $(i = 1, 2, \dots, d+1)$ is a copy of X and a vertex $v_i \in X_i$ is adjacent to a vertex $u \in Y$ if and only if the corresponding vertex $v \in X$ is adjacent to u in H. For a vertex $v \in X$, we may use v_i to denote its corresponding vertex in X_i.

We find a maximum matching M' in H' by using a $O(n^{1/2}m)$-time algorithm [13,18]. Let M be the set of edges in H corresponding to the matching M', i.e., an edge uv ($u \in Y$ and $v \in X$) of H is in M if and only if uv_i is in M' for some v_i corresponding to v. Edges in M are called *marked* and others are called *unmarked*. Observe that since M' is a matching in H', we have that $|M| = |M'|$. The set of marked edges in H forms a $_{\leq}(d+1)$-star packing $S_{\leq d+1}$. This is the $_{\leq}(d+1)$-star packing we are seeking for. It is also easy to observe that

Lemma 3. *Graph H has a $_{\leq}(d+1)$-star packing containing t edges if and only if H' has a matching of size t.*

Next, we analyze some properties of $S_{\leq d+1}$ and find C' and I' satisfying Basic Condition based on these properties.

Let S_{d+1} denote the set of $(d+1)$-stars in $S_{\leq d+1}$. An X-vertex in a star in S_{d+1} is *fully tagged*. Then $X \cap V(S_{d+1})$ is the set of fully tagged vertices. A Y-vertex is *untagged* if it is adjacent to at least one vertex in X in H but not contained in any star in $S_{\leq d+1}$. A path P in H that alternates between edges not in M and edges in M is called an *M-alternating path*.

Lemma 4. *If there is an M-alternating path P from an untagged vertex $u \in Y$ to a vertex $v \in X$ in H, then v is fully tagged.*

Next, we are going to set C' and I'. If there is no untagged vertex, let $C' = \emptyset$. Otherwise let C' be the set of X-vertices connected with at least one untagged vertex by an M-alternating path in H. Let $X' = X \setminus C'$. Let Y' be the set of Y-vertices that is a leaf of a $_{\leq}(d+1)$-star in $S_{\leq d+1}$ that is centered at a vertex in X', and $I' = Y \setminus Y'$.

Lemma 5. *The two sets C' and I' obtained above satisfy Basic Condition.*

We describe the above progress to compute C' and I' as an algorithm `basic`(G, X, Y) in Fig. 2, which will be used as a subalgorithm in our main algorithm.

Lemma 6. *Algorithm* `basic`(G, X, Y) *runs in $O(n^{1/2}m)$ time.*

Note that all untagged vertices will be in I'. So if the size of Y is large, for example $|Y| > (d+1)|X|$, we can guarantee that there is always some untagged vertices and the set I' returned by `basic`(G, X, Y) is not an empty set.

Phase 3. After obtaining (C', I') from Phase 2, we look at the partition $\mathcal{P} = (I', C', T' = N(I') \setminus C', J' = V \setminus (I' \cup C' \cup T'))$. Since there is no edge between I' and $X' = X \setminus C'$, we know that $T' \subseteq Y$ and $X' \subseteq J'$. Then there is no edge between I' and J'. The partition $\mathcal{P}$ satisfies Conditions (2) and (3) in Definition 2 for d-bounded decompositions. Next, we consider Condition (1). Let $G^* = G[V \setminus C']$. Any vertex in I' is of degree $\leq d$ in G^*, because $G[Y] = G[V \setminus X]$ has maximum degree $\leq d$ and I'-vertices are not adjacent to any vertex in $X \setminus C'$.

Input: A graph $G = (V, E)$ and a partition (X, Y) of the vertex set V.
Output: Two sets $C' \subseteq X$ and $I' \subseteq Y$ satisfying the Basic Condition.

1. Compute the bipartite graph H and the auxiliary bipartite graph H'.
2. Compute a maximum matching M' in H' and the corresponding edge set M and the $_{\leq}(d+1)$-star packing $S_{\leq d+1}$ in H.
3. Let C' be $\emptyset$ if there is no untagged vertex, and the set of X-vertices connected with at least one untagged vertex by an M-alternating path in H otherwise. Let $X' \leftarrow X \setminus C'$. Let Y' be the set of Y-vertices each of which is a leaf of a $_{\leq}(d+1)$-star centered at a vertex in X' and let $I' \leftarrow Y \setminus Y'$.
4. Return (C', I').

Fig. 2. Algorithm `basic`(G, X, Y)

Although $T' = N(I') \setminus C' \subseteq Y$, vertices in T' is possible to be of degree $> d$ in G^*. In fact, we only know that each vertex in T' is of degree $\leq d$ in $G[Y]$. But in G^*, every T'-vertex is adjacent to some vertices in $X' = X \setminus C'$ and thus can be of degree $> d$. So Condition (1) may not hold. We will move some vertices out of C' and I' to make the decomposition satisfying Condition (1).

Let B be the set of T'-vertices that are of degree $> d$ in G^*. Note that any vertex in B is adjacent to some vertices in X. We call vertices in $N_{I'}(B) = N(B) \cap I'$ *bad* vertices. Note that B is not an empty set if and only if $N_{I'}(B)$ is not an empty set. If $B = \emptyset$, then Condition (1) holds directly. For the case that $B \neq \emptyset$, i.e., $N_{I'}(B) \neq \emptyset$, our idea is to update I' by removing $N_{I'}(B)$ out of I'. However, after moving some vertices out of I', there may not be a full $(d+1)$-star packing from C' to I' anymore. So after moving $N_{I'}(B)$ out of I' we invoke the algorithm `basic`$(G[C' \cup I'], C', I')$ for Phase 2 on the subgraph $G[C' \cup I']$ to find new C' and I', and then check whether there are new bad vertices or not. We do these iteratively until we find a d-bounded decomposition, where no bad vertex exists. In the returned d-bounded decomposition, I' and C' may become empty. However, we can guarantee $I' \neq \emptyset$ when the size of the graph satisfies some conditions. We analyze this after describing the whole algorithm.

The Whole Algorithm for Decomposition. Our algorithm `decomposition` (G) presented in Fig. 3 is to compute two subsets of vertices C and I of the input graph G such that $(I, C, T = N(I) \setminus C, J = V \setminus (I \cup C \cup T))$ is a d-bounded decomposition of G.

Steps 3, 4 and 6 in `decomposition`(G) are the same steps in `basic`(G, X, Y). Here we add Step 5 into these steps, which is used to update the $(d+1)$-star packing S. In `decomposition`(G), Steps 1, 2 and 5 are corresponding to Phase 1, Steps 3, 4 and 6 are corresponding to Phase 2, and Steps 7 and 8 are corresponding to Phase 3. Note that Step 8 will also invoke `basic`(G, X, Y).

Input: A graph $G = (V, E)$.
Output: Two subsets of vertices C and I such that $(I, C, T = N(I) \setminus C, J = V \setminus (I \cup C \cup T))$ is a d-bounded decomposition.

1. Find a maximal $(d+1)$-star packing S in G.
2. $X \leftarrow V(S)$ and $Y \leftarrow V \setminus X$.
3. Compute the bipartite graph H and the auxiliary bipartite graph H'.
4. Compute a maximum matching M' in H' and the corresponding edge set M and the $_{\leq}(d+1)$-star packing $S_{\leq d+1}$ in H.
5. Let S_{d+1} be the set of $(d+1)$-stars in $S_{\leq d+1}$.
   If $\{|S_{d+1}| > |S|\}$,
   then $S \leftarrow S_{d+1}$ and **goto** Step 2.
6. Let C' be $\emptyset$ if there is no untagged vertex, and be the set of X-vertices connected with at least one untagged vertex by an M-alternating path in H otherwise. Let $X' \leftarrow X \setminus C'$. Let Y' be the set of leaves of $_{\leq}(d+1)$-stars in $S_{\leq d+1}$ centered at vertices in X' and let $I' \leftarrow Y \setminus Y'$.
7. Compute the set $N_{I'}(B)$ of bad vertices based on C' and I'.
8. **If** $\{N_{I'}(B) \neq \emptyset\}$,
   then $I' \leftarrow I' \setminus N_{I'}(B)$, $(C', I') \leftarrow \texttt{basic}(G[C' \cup I'], C', I')$, and **goto** Step 7.
9. **Return** $(C = C', I = I')$.

Fig. 3. Algorithm $\texttt{decomposition}(G)$

Lemma 7. *The two vertex sets C and I returned by* `decomposition`(G) *make* $(I, C, T = N(I) \setminus C, J = V \setminus (I \cup C \cup T))$ *a d-bounded decomposition.*

We can prove the following two important lemmas.

Lemma 8. *Algorithm* `decomposition`(G) *runs in $O(n^{3/2}m)$ time and returns (C, I) such that (I, C, T, J) is a d-bounded decomposition of G, where $T = N(I) \setminus C$ and $J = V(G) \setminus (I \cup C \cup T)$.*

Lemma 9. *Algorithm* `decomposition`(G) *returns (C, I) such that*

$$|V \setminus (C \cup I)| \leq (d^3 + 4d^2 + 5d + 3)\alpha(G).$$

4.2 The Algorithm for Theorem 3

Lemma 9 can get the size condition in Theorem 3 directly. We use the following algorithm in Fig. 4 for Theorem 3.

From the second iteration of Step 2 in `BDD`(G), each execution of $I \leftarrow I \cup I'$ will include at least one new vertex to I. So `decomposition`$(G[V \setminus (C \cup I)])$ will be called for at most $n + 1$ times. Algorithm `BDD`(G) runs in $O(n^{5/2}m)$ time. Furthermore, if `decomposition`$(G' = G[V \setminus (C \cup I)])$ returns two empty sets, then by Lemma 9 we have $|V(G')| = |V(G') \setminus (C \cup I)| \leq (d^3 + 4d^2 + 5d + 3)\alpha(G')$. These together with Lemmas 8 and 9 imply Theorem 3.

Input: A graph $G = (V, E)$.
Output: Two subsets of vertices C and I satisfying the conditions in Theorem 3.

1. $C, I \leftarrow \emptyset$.
2. **Do** { $(C', I') \leftarrow$ `decomposition`$(G[V \setminus (C \cup I)])$, $C \leftarrow C \cup C'$ and $I \leftarrow I \cup I'$ } **while** $I' \neq \emptyset$.
3. **Return** (C, I).

Fig. 4. Algorithm `BDD`(G)

5 Concluding Remarks

In this paper, we provide a refined version of the generalized Nemhauser-Trotter-Theorem, which applies to Bounded-Degree Vertex Deletion and for any $d \geq 0$ can get a linear-vertex problem kernel for the problem parameterized by the solution size. This is the first linear-vertex kernel for the case that $d \geq 3$. Our algorithms and proofs are based on extremal combinatorial arguments, while the original NT-Theorem uses linear programming relaxations [22]. It seems no way to generalize the linear programming relaxations used for the original NT-Theorem to Bounded-Degree Vertex Deletion [14]. A crucial technique in this paper is the d-bounded decomposition. To find such kinds of decompositions, we follow the ideas to find crown decompositions [2] and the algorithmic strategy in [14]. However, we use more ticks and can finally obtain the linear size condition.

As pointed out by Fellows et al. [14], the results for Bounded-Degree Vertex Deletion in this paper can be modified for the problem of packing stars. We believe that the new decomposition technique can be used to get local optimization properties and kernels for more deletion and packing problems.

References

1. Abu-Khzam, F.N., Collins, R.L., Fellows, M.R., Langston, M.A., Suters, W.H., Symons, C.T.: Kernelization algorithms for the vertex cover problem: theory and experiments. In: ALENEX 2004, pp. 62–69. ACM/SIAM (2004)
2. Abu-Khzam, F.N., Fellows, M.R., Langston, M.A., Suters, W.H.: Crown structures for vertex cover kernelization. Theor. Comput. Syst. **41**(3), 411–430 (2007)
3. Balasundaram, B., Butenko, S., Hicks, I.V.: Clique relaxations in social network analysis: the maximum k-plex problem. Oper. Res. **59**(1), 133–142 (2011)
4. Bar-Yehuda, R., Rawitz, D., Hermelin, D.: An extension of the Nemhauser & Trotter theorem to generalized vertex cover with applications. SIAM J. Discrete Math. **24**(1), 287–300 (2010)
5. Bar-Yehuda, R., Even, S.: A local-ratio theorem for approximating the weighted vertex cover problem. Ann. Discrete Math. **25**, 27–45 (1985)
6. Betzler, N., Bredereck, R., Niedermeier, R., Uhlmann, J.: On bounded-degree vertex deletion parameterized by treewidth. Discrete Appl. Math. **160**(1–2), 53–60 (2012)

7. Chen, J., Kanj, I.A., Jia, W.: Vertex cover: further observations and further improvements. J. Algorithms **41**(2), 280–301 (2001)
8. Chen, Z.-Z., Fellows, M., Fu, B., Jiang, H., Liu, Y., Wang, L., Zhu, B.: A linear kernel for co-path/cycle packing. In: Chen, B. (ed.) AAIM 2010. LNCS, vol. 6124, pp. 90–102. Springer, Heidelberg (2010)
9. Chlebík, M., Chlebíková, J.: Crown reductions for the minimum weighted vertex cover problem. Discrete Appl. Math. **156**, 292–312 (2008)
10. Chor, B., Fellows, M., Juedes, D.W.: Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In: Hromkovič, J., Nagl, M., Westfechtel, B. (eds.) WG 2004. LNCS, vol. 3353, pp. 257–269. Springer, Heidelberg (2004)
11. Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: An improved FPT algorithm and a quadratic kernel for pathwidth one vertex deletion. Algorithmica **64**(1), 170–188 (2012)
12. Dell, H., van Melkebeek, D.: Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. J. ACM **61**(4), 23:1–23:27 (2014)
13. Even, S., Tarjan, R.E.: An $O(n^{2.5})$ algorithm for maximum matching in general graphs. In: FOCS 1975, pp. 100–112 (1975)
14. Fellows, M.R., Guo, J., Moser, H., Niedermeier, R.: A generalization of Nemhauser and Trotter's local optimization theorem. J. Comput. Syst. Sci. **77**, 1141–1158 (2011)
15. Fellows, M.R., Guo, J., Moser, H., Niedermeier, R.: A generalization of Nemhauser and Trotter's local optimization theorem. In: STACS 2009, pp. 409–420. IBFI Dagstuhl (2009)
16. Fomin, F.V., Lokshtanov, D., Misra, N., Philip, G., Saurabh, S.: Hitting forbidden minors: approximation and kernelization. In: STACS 2011, pp. 189–200 (2011)
17. Hochbaum, D.S.: Approximation algorithms for the set covering and vertex cover problems. SIAM J. Comput. **11**(3), 555–556 (1982)
18. Hopcroft, J., Karp, R.M.: An $O(n^{2.5})$ algorithm for maximum matching in bipartite graphs. SIAM J. Comput. **2**(4), 225–231 (1973)
19. Khuller, S.: The vertex cover problem. SIGACT News **33**(2), 31–33 (2002)
20. Komusiewicz, C., Huffner, F., Moser, H., Niedermeier, R.: Isolation concepts for efficiently enumerating dense subgraphs. Theoret. Comput. Sci. **410**(38–40), 3640–3654 (2009)
21. Lokshtanov, D., Narayanaswamy, N.S., Raman, V., Ramanujan, M.S., Saurabh, S.: Faster parameterized algorithms using linear programming. ACM T. Algorithms **11**(2), 15 (2014)
22. Nemhauser, G.L., Trotter, L.E.: Vertex packings: structural properties and algorithms. Math. Program. **8**, 232–248 (1975)
23. Nishimura, N., Ragde, P., Thilikos, D.M.: Fast fixed-parameter tractable algorithms for nontrivial generalizations of vertex cover. Discrete Appl. Math. **152**(1–3), 229–245 (2005)
24. Seidman, S.B., Foster, B.L.: A graph-theoretic generalization of the clique concept. J. Math. Sociol. **6**, 139–154 (1978)
25. Thomassé, S.: A $4k^2$ kernel for feedback vertex set. ACM T. Algorithms **6**(2), 32:1–32:28 (2010)
26. Xiao, M.: A note on vertex cover in graphs with maximum degree 3. In: Thai, M.T., Sahni, S. (eds.) COCOON 2010. LNCS, vol. 6196, pp. 150–159. Springer, Heidelberg (2010)