2017年12月22日開催の電子情報通信学会 コンピュテーション研究会で発表予定の本論文の結果がO(k) time per solutionだったが間違いだった．これは現在修正されて最終的にはO(k^2) time per solutionである．最終的な結果である定数縮退数グラフに対して，定数ならし時間であるという主結果に変更はない．

社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

k-縮退グラフ中の支配集合の効率良い列挙

栗田　和宏†　和佐　州洋††　宇野　毅明††　有村　博紀†

† 北海道大学 大学院情報科学研究科 情報理工学専攻, 北海道
†† 国立情報学研究所, 東京
E-mail: †{k-kurita,arim}@ist.hokudai.ac.jp, ††{wasa,uno}@nii.ac.jp

あらまし　支配集合はクリークや，独立点集合，s-t パス，カットのように基本的なグラフ構造の一つである．クリーク等のグラフ構造には極大や極小の構造を多項式遅延時間で列挙するアルゴリズムが知られている．しかし，極小支配集合を多項式遅延もしくはならし多項式時間で列挙するアルゴリズムが存在するかどうかはわかっていない．さらに，極小でない支配集合を含めた全ての支配集合の列挙についてはあまり研究が行われていない．本論文では極小な支配集合だけでなく，全ての支配集合の列挙を行う．主結果として，我々は効率の良い支配集合の列挙アルゴリズムを与える．このアルゴリズムは $O(nm)$ 領域を用いて，全ての支配集合を解 1 つあたり $O(k^2)$ 時間で列挙を行う．ここで，k とは入力グラフの縮退数である．さらに，平面的グラフは縮退数が定数であることが知られている．したがって，提案アルゴリズムは平面的グラフ中の支配集合を解 1 つあたり定数時間で列挙する．

キーワード　支配集合，列挙アルゴリズム，ならし多項式時間，縮退数

Efficient Enumeration of Dominating Sets in k-degenerate Graphs

Kazuhiro KURITA†, Kunihiro WASA††, Takeaki UNO††, and Hiroki ARIMURA†

† IST, Hokkaido University, Sapporo, Japan
†† National Institute of Informatics, Tokyo, Japan
E-mail: †{k-kurita,arim}@ist.hokudai.ac.jp, ††{wasa,uno}@nii.ac.jp

Abstract　A dominating set is one of the fundamental graph structure, like clique, independent set, s-t path, and cut. It is known that there is a polynomial delay algorithm to enumerate theses minimal or maximal graph structures. However, it is open problem that there is a polynomial delay or amortized polynomial time algorithm to enumerate minimal dominating sets. In addition, enumeration of all dominating sets has not been paid much attention. In this paper, we enumerate all dominating sets not only minimal dominating sets. As a main result, we propose an enumeration algorithm. This algorithm enumerates all dominating set in $O(k^2)$ time per solution using $O(nm)$ space, where k is degeneracy of an input graph. It is known that planar graphs have the constant degeneracy. Hence, the proposed algorithm enumerates all dominating set in constant time per solution.

Key words　dominating sets, enumeration algorithm, amorized polynomial time, degeneracy

1. Introduction

An *enumeration problem* is a problem to output all solutions satisfying a given constraint without duplication. In this paper, we address the enumeration of dominating sets in an input graph. The number of solutions of enumeration problems is often considerably smaller than exponential in the size of an input. Hence, if the estimation of the number of solutions is not tight, the evaluation of the time complexity according to the size of an input overestimates the actual behavior. We evaluate the efficiency of an enumeration algorithm by *not only* the size of input *but also* the number of outputs [10]. Let n and N be the size of the input and the number of outputs, respectively. We call an enumeration algorithm *amortized polynomial time algorithm* if the time complexity of the algorithm is $O(poly(n)N)$. In addition, we call an enumeration algorithm a *polynomial delay algorithm* if the followings are bounded by $O(poly(n))$: the time until the algorithm outputs the first solution, the interval between outputting two consecutive solutions, and the time until the algorithm stops after outputting the last solution. Moreover, we call an enumeration algorithm an *incremental*

polynomial time algorithm if delay is bounded by $poly(n, N')$, where N' is the number of solutions that are already generated by an algorithm, That is, when the algorithm generates i-th solution, $N' = i - 1$. We propose the amortized polynomial time algorithm that enumerates all dominating sets in a given graph G in $O\left(k^2\right)$ per solution time, where k is the *degeneracy* of G.

Until now, researchers developed enumeration algorithms for minimal dominating sets with polynomial delay for some graph classes such as interval graphs [6], permutation graphs [6], and P_6-free chordal graphs [5]. In addition, there are incremental polynomial time algorithm for graphs with girth at least 7 [7] and chordal bipartite graphs [4]. On the other hands, the minimal edge dominating set enumeration problem is solved in polynomial delay [7]. However, to the best of our knowledge, there is no paper about enumeration of dominating sets for general graphs. It is actually easy to obtain an algorithm for a dominating set enumeration problem whose running time is $O(n)$ time per solution by using binary partition method. In this paper, we address to develop a strictly faster enumeration algorithm than the trivial algorithm. In addition, we focus on the *degeneracy* of a graph G. The degeneracy is one of measures of sparseness of a graph. It is known that graphs in some graph class have the constant degeneracy such as, forest, grid graphs, outerplaner graphs, planer graphs, bounded tree width graphs, H-minor free graphs for some fixed H, and so on. If a graph has the small degeneracy, a graph has a *good* vertex ordering, called a *degeneracy ordering* [9] shown in Sect. 4. By using this ordering, some efficient enumeration algorithms have been developed [2, 3, 11]. As the main result of this paper, we propose an efficient algorithm for enumerating dominating sets in a given graph. Our algorithm is based on *reverse search* method [1]. Moreover, by maintaining the number of neighbors of each vertex and using the degeneracy ordering, we achieve $O\left(k^2\right)$ time per solution to enumerate all solutions. This is the first non-trivial result for a dominating set enumeration problem.

The organization of this paper is as follows: in Sect. 2. we define some basic notation. In Sect. 3. we give the outline of our proposed algorithm EDS based on reverse search method, and in Sect. 4. we show some data structures using a degeneracy ordering for enumerating efficiently and prove the time complexity. In Sect. 5., we conclude this paper.

2. Preliminary

Let $G = (V(G), E(G))$ be a simple undirected graph, that is, G has no self loops and multiple edges, with vertex set $V(G)$ and edge set $E(G) \subseteq V(G) \times V(G)$. If G is clear from the context, we suppose that $V = V(G)$ and $E = E(G)$. Let u and v be vertices in G. An edge e with u and v is denoted by $e = \{u, v\}$. Two vertices $u, v \in V$ are *adjacent* if $\{u, v\} \in E$. We denote by $N_G(u)$ the set of vertices that are adjacent to u on G and by $N_G[u] = N_G(u) \cup \{u\}$ the set of closed neighbors of u. We say v is a *neighbor* of u if $v \in N_G(u)$. The *set of neighbors* of U is defined as $N(U) = \bigcup_{u \in U} N_G(u) \setminus U$. Similarly, let $N[U]$ be $\bigcup_{u \in U} N_G(u) \cup U$. Let $d_G(u) = |N_G(u)|$ be the *degree* of u in G. $\Delta(G) = \max_{x \in V} d(x)$ denotes the maximum degree of G. For any vertex subset $V' \subseteq V$, we call $G[V'] = (V', E[V'])$ an *induced subgraph* of G, where $E[V'] = \{\{u, v\} \in E(G) \mid u, v \in V'\}$. Since $G[V']$ is uniquely determined by V', we identify $G[V']$ with V'. We denote by $G \setminus \{e\} = (V, E \setminus \{e\})$ and $G \setminus \{v\} = G[V \setminus \{v\}]$. For simplicity, we will use $v \in G$ and $e \in G$ to refer to $v \in V(G)$ and $e \in E(G)$, respectively. A set D of vertices is a *dominating set* if D satisfies $N[D] = V$. v in D is the *private vertex* of u if $u \in N[v]$ and $u \notin N[D \setminus \{v\}]$ hold. Moreover, $v \in D$ is *isolated* or an *isolated vertex* if $N(v) \cap D = \emptyset$. We show an example of a dominating set and an isolated vertex in Figure 1. We now define the dominating set enumeration problem as follows:

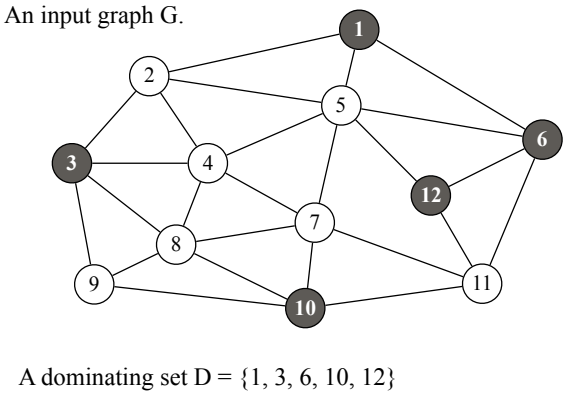

Fig 1 Black vertices indicate vertices of dominating set D in G. D is not minimal since $D \setminus \{6\}$ is also dominating set. 3 and 10 are a private vertex of 4 and 7, respectively. Moreover, 3 is isolated in D since $N(3) \cap D = \emptyset$.

Problem 1. *Enumerate all dominating sets in a given graph G without duplication.*

3. Basic algorithm

In this section, we show the proposed algorithm EDS that is based on *reverse search method* [1]. The reverse search method is one of frameworks for constructing an enumeration algorithm. To enumerate all solution by reverse search, we first define the *parent-child relationship* between solutions.

Algorithm 1: EDS enumerates all dominating sets in amortized polynomial time.

```
1 Procedure EDS (G = (V, E))        // G: an input graph
2     AllChildren(V, G);
3 Procedure AllChildren (X, G = (V, E))        // X: the current solution
4     Output X ;
5     for v ∈ X do
6         if N[X \ {v}] = V ∧ P(X \ {v}) = X then
7             AllChildren(X \ {v}, G);
8     return;
```

The precise definition of the relationship will be given later. Let $G = (V, E)$ be an input graph with $V = \{v_1, \ldots, v_n\}$ and X and Y be dominating sets on G. We denote by $\mathcal{S}(G)$ the set of solutions, that is, the set of dominating sets of$G$, and by $\mathcal{F}(G)$ a digraph on the set of solutions $\mathcal{S}(G)$. Here, the vertex set of $\mathcal{F}(G)$ is $\mathcal{S}(G)$ and the edge set $\mathcal{E}(G)$ of $\mathcal{F}(G)$ is defined according to the parent-child relationship. For any pair of solutions X and Y, there is a directed edge from D_1 to D_2 on $\mathcal{F}(G)$ if D_1 is the *parent* of D_2. We call $\mathcal{F}(G)$ the *family tree* for G. By traversing on the family tree, we can obtain the all solutions.

We arbitrarily number the vertices in G from 1 to n and call the number of a vertex the *index* of the vertex. $pv(X)$, called the *parent vertex*, is the vertex in $V \setminus X$ with the minimum index. For any dominating set X such that $X \neq V$, Y is the *parent* $\mathcal{P}(X)$ of X if $Y = X \cup \{pv(X)\}$. Note that since any superset of a dominating set also dominates G, $\mathcal{P}(X)$ is a dominating set of G. We call X is a *child* of Y if $\mathcal{P}(X) = Y$. By using this parent-child relationship, we construct the family tree $\mathcal{F}(G)$ mentioned above. Next, we show that $\mathcal{F}(G)$ forms a tree rooted at V. We call V the *root* of $\mathcal{F}(G)$.

Lemma 1. *For any dominating set X, by recursively applying the parent function $\mathcal{P}(\cdot)$ to X at most n times, we obtain V.*

Proof. For any dominating set X, since $pv(v)$ always exists, there always exists the parent vertex for X. In addition, $|\mathcal{P}(X) \setminus X| = 1$. Hence, the statement holds. □

Lemma 2. *$\mathcal{F}(G)$ forms a tree.*

Proof. Let X be any solution in $\mathcal{S}(G) \setminus \{V\}$. Since X has exactly one parent and V has no parent, $\mathcal{F}(G)$ has $|V(\mathcal{F}(G))| - 1$ edges. In addition, since there is a path between X and V by Lemma 1, $\mathcal{F}(G)$ is connected. Hence, the statement holds. □

Algorithm 2: this algorithm shows improved AllChildren procedure.

```
1  Function AllChildren(X, C(X), Debt(X), SIV(X), G = (V, E))
       //X: the current solution, C(X): the candidate set of X, G: an input graph
2      Output X;
3      for v ∈ C(X) do
4          X' ← X \ {v};
5          Update(C(X), Debt(X), SIV(X), v);
6          AllChildren(X', C(X'), Debt(X'), SIV(X'), v, G);
7          Debt(X) ← Debt(X) \ N^<(v);
8      return;
9  Function Update(C(X), Debt(X), SIV(X), v)
10     X' ← X \ {v};
11     for u ∈ SIV(v, X) do
12         if N_X'(u) = {u} then
13             C(X') ← C(X') \ {u} ;   //Remove vertices in IV(v, X)
14     for u ∈ N^>(v) do
15         if u ∈ Debt(X) ∧ |N^>v(u)| = 1 then
16             w, w' ← arg min{N(u)}, arg min{N(w)};
17             C(X') ← C(X') \ w;
18             SIV(w', X') ← SIV(w', X') \ {w};
19             Debt(X') ← Debt(X') \ N^>v[w];
20     if N_X(v) ∩ X^<v = ∅ then
21         Debt(X') ← Debt(X') ∪ {v};
```

Our proposed algorithm EDS shown in Algorithm 1 outputs solutions by naively traversing on $\mathcal{F}(G)$ in the depth-first manner.

Theorem 1. *EDS outputs all dominating sets in G.*

4. Data structures and analysis

In this section, we show the details of the proposed algorithm EDS. We particularly focus on the *degeneracy* of an input graph. G is a *k-degenerate graph* [8] if for any induced subgraph H of G, the minimum degree in H is less than or equal to k. The degeneracy of G is the such minimum value k. If G is a k-degenerate graph, then there always exists an ordering among vertices in G, called a *degeneracy ordering* of G, satisfies the following condition: for any vertex v in G, the number of vertices that are larger than v and adjacent to v is at most k. We show an example of a degeneracy ordering of a graph in Fig. 2. Matula and Beck show that the degeneracy of G and a degeneracy ordering of G can be obtained in $O(n + m)$ time [9]. Note that there are some degeneracy orderings for a graph. In what follows, we fix a degeneracy ordering of G and number the indices of vertices from 1 to n

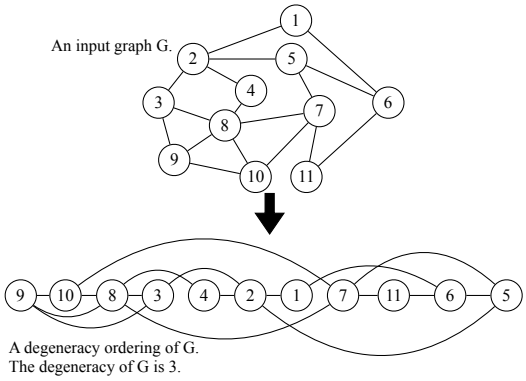

Fig 2 An example of a degeneracy ordering. In this ordering, each vertex v satisfies $\left|N_{<}(v)\right| \leqq 3$. Hence, G is a 3-degenerate graph and the degeneracy of G is 3.

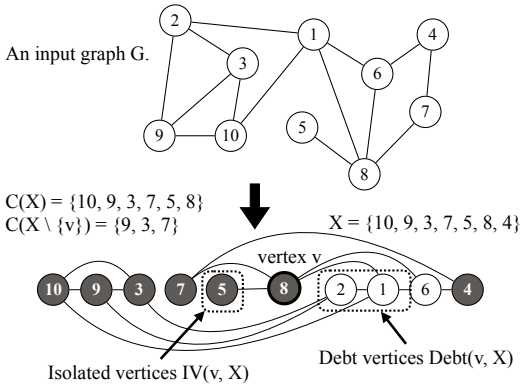

Fig 3 This figure shows an example of $IV(v, X)$ and $Debt(v, X)$. X is a dominating set of G. Vertices in G are sorted in the degeneracy ordering. Let $v = 8$ and $X = \{10, 9, 3, 7, 5, 8, 4\}$. Here, $IV(8, X) = \{5\}$ and $Debt(8, X) = \{2, 1\}$. When we remove 8 from X, then 5 and 10 can not be removed from $X \setminus \{8\}$ since $X \setminus \{8, 5\}$ does not dominate 5 and $X \setminus \{8, 10\}$ does not dominate 1. Then, 10 does not belong to $IV(3, X \setminus \{8\})$, where $3 = \arg\min\{N(10)\}$. Here, 7 is an isolated vertex. However, 7 is only dominated by 4. Hence, $C(X)$ does not include 7.

according to the degeneracy ordering. If no confusion occurs, we identify a vertex with its index, that is, for any pair of vertices u and v, we write $u < v$ if u is smaller than v in the degeneracy ordering. $V^{<v}$ denotes the set of vertices that are in V and smaller than v in the ordering. In addition, we assume that a graph is represented by an adjacency list and these lists are sorted by a degeneracy order in preprocess using $O(\Delta n)$ time.

We first define some notations that will be used in this section. Let X be a dominating set of G and $N_X^{\leqq u}(v) = \{w \in N(v) \mid w < u \wedge w \in X\}$ be the set of neighbors of v that are smaller than u and in X. To simplify the notation, We denote $N_X^{\geqq v}(v)$ and $N_X^{\leqq v}(v)$ by $N_X^{\geqq}(v)$ and $N_X^{\leqq}(v)$. We define the *candidate set* $C(X)$ of X as $C(X) = \{v \in X \mid N[X \setminus \{v\}] = V \wedge v < pv(X)\}$. Note that $v \in C(X)$ is smaller than any vertex in $V \setminus X$.

Lemma 3. *Let X be a dominating set of G and v be a vertex in X. Then, $v \in C(X)$ if and only if $X \setminus \{v\}$ is dominating set and $X = \mathcal{P}(X \setminus \{v\})$.*

Proof. The only if part is obvious from the definition of $C(X)$ and $\mathcal{P}(\cdot)$. We show the other direction. Since $X = \mathcal{P}(X \setminus \{v\})$, v is smaller than any vertex in $V \setminus (X \setminus \{v\})$. Hence, $v < pv(X)$ holds. From the assumption, $X \setminus \{v\}$ dominates X. Hence, $v \in C(D)$. The statement holds. □

Lemma 3 implies that we can generate children of X by removing a vertex in the candidate set of X. That is, we can avoid to check whether a genereated dominating set is actually a child of X or not. We next consider how to compute the children of X, that is, how to compute the candidate of X. To compute $C(X)$ efficiently, EDS maintains the following two data structures: $IV(v, X)$ and $Debt(v, X)$, where $v \in C(X)$. The definition of the *isolated vertices* $IV(v, X)$ for v and the *debt vertices* $Debt(v, X)$ are as follows:

$$IV(v, X) = \left\{u \in N_{C(X)}^{<v}(v) \;\middle|\; N_X(u) = \{v\}\right\}, \text{ and}$$
$$Debt(v, X) = \left\{u \in V^{>v} \;\middle|\; N_X[u] \subseteqq C^{\leqq v}(X) \wedge |N_X[u]| \geqq 2\right\}.$$

Also, we denote by $IV(X) = \bigcup_{v \in C(X)} IV(v, X)$. Roughly speaking, $u \in IV(v, X)$ becomes the private vertex of itself in $X \setminus \{v\}$ after removing v from X, and $u \in Debt(v, X)$ is only dominated by vertices smaller than v in X. In Fig. 3, we show an example of the isolated vertices and debt vertices.

Lemma 4. *Let $v \in C(X)$. If $V^{<v} \subseteqq X$, then $C(X \setminus \{v\}) = C(X) \setminus \left(V^{>v} \cup IV(v, X) \cup N_{C(X)}^{<v}(\Phi(v, X))\right)$, where $\Phi(v, X) = \left\{u \in N_{Debt(v,X)}(v) \;\middle|\; \left|N_{C(X)}^{<v}(u)\right| = 1\right\}$.*

Proof. We show $C(X) \setminus \left(V^{>v} \cup IV(v, X) \cup N_{C(X)}^{<v}(\Phi(v, X))\right)$ is a subset or equal of $C(X \setminus \{v\})$. Let $w \in C(X) \setminus \left(V^{>v} \cup IV(v, X) \cup N_{C(X)}^{<v}(\Phi(v, X))\right)$. Hence, $w < v$. Suppose that $w \notin C(X \setminus \{v\})$. This implies that $X' = X \setminus \{v, w\}$ is not a dominating set of G or $\mathcal{P}(X') \neq X \setminus \{v\}$. (I) We assume that X' is not a dominating set. Since $X \setminus \{v\}$ is a dominating set, there exists a vertex w' in $N[w]$ that is not dominated by X'. Since $w \in C(X)$, w' is only dominated by both v and w. If $w' = w$, then $w \in IV(v, X)$. If $w' \neq w$, then since $w < v$, $w \in N_{C(X)}^{<v}(\Phi(v, X))$ holds. However, these contradict the assumption. (II) We assume that X' is a dominating set but $\mathcal{P}(X') \neq X \setminus \{v\}$. This implies that $pv(X') \neq w$. Since $pv(X \setminus \{v\}) = v$, there is no smaller vertex than v in $V \setminus (X \setminus \{v\})$. From the assumption $w < v$, w is smaller than any vertex in $V \setminus (X \setminus \{v, w\})$. However, this contradicts the assumption,

and thus, $w \in C(X \setminus \{v\})$. We next show $C(X \setminus \{v\}) \subseteq C(X) \setminus \left(V^{>v} \cup IV(v,X) \cup N^{\leq v}_{C(X)}(\Phi(v,X))\right)$. From the definition, $C(X \setminus \{v\}) \subseteq C(X)$ holds. Let $w \in C(X \setminus \{v\})$. Then, w is not included by $V^{>v}$ since $w < v$. Moreover, w is not included by $IV(v,X) \cup N^{\leq v}_{C(X)}(\Phi(v,X))$ since $X \setminus \{v,w\}$ is a dominating set. Hence, the statement holds. □

Lemma 5. *Let $v, w \in C(X)$ such that $v < w$. If $V^{<v} \subseteq X$, then $IV(v, X \setminus \{w\}) = IV(v,X) \cup \Gamma(v,w,X)$, where $\Gamma(v,w,X) = \left\{u \in N^{\leq}_X(v) \mid N_X(u) = \{v,w\}\right\}$.*

Proof. Since for any $u \in IV(v,X)$, u is adjacent to only v, $IV(v,X) \subseteq IV(v, X \setminus \{w\})$. If $u \in \Gamma(v,w,X)$, from the definition of $IV(v, X \setminus \{w\})$ and $\Gamma(v,w,X)$, $u \in IV(v, X \setminus \{w\})$. Hence, $IV(v,X) \cup \Gamma(v,w,X) \subseteq IV(v, X \setminus \{w\})$. We next show $IV(v, X \setminus \{w\}) \subseteq IV(v,X) \cup \Gamma(v,w,X)$. If $u \in IV(v, X \setminus \{w\})$, from the definition of $IV(v, X \setminus \{w\})$, $N_{X \setminus \{w\}}(u) = \{v\}$. Hence, $N_X(u) = \{v\}$ or $N_X(u) = \{v,w\}$. This implies that $u \in IV(v,X) \cup \Gamma(v,X)$. Thus, the statement holds. □

Lemma 6. *Let $v, w \in X$ such that $v < w$. Moreover, there is no vertex $x \in X$ such that $v < x < w$. If $V^{<v} \subseteq X$, then $Debt(v,X) = Debt(w,X) \setminus N^{>v}(w)$.*

Proof. If $u \in Debt(w,X) \setminus N^{>v}(w)$, since u is not adjacent to w and from the assumption, all neighbors of u in X are smaller than v and $X \setminus \{u\}$ also dominates G. Hence, $u \in Debt(v,X)$. If $u \in Debt(v,X)$, u does not belong to $N^{>v}(w)$. In addition, u is also in $Debt(w,X)$ since $w \in X$. Thus, the statement holds. □

Lemma 7. *Let $v, w \in C(X)$ such that $v < w$. Moreover, there is no vertex $x \in X$ such that $v < x < w$. If $V^{<v} \subseteq X$, then $Debt(v, X \setminus \{w\}) = (Debt(w,X) \cup U(v,w,X)) \setminus (W(v,w,X) \cup T(v,w,X))$, where*

$$W(v,w,X) = \left\{u \in Debt(w,X) \;\middle|\; \left|N_{X\setminus\{w\}}(u)\right| = 1\right\} \text{ and,}$$

$$T(v,w,X) = \left\{u \in Debt(w,X) \;\middle|\; u \in N\left(N^{\leq v}_{C(X)}(\Phi(v,X))\right)\right\} \text{ and,}$$

$$U(v,w,X) = \left\{u \in N^{>v}(w) \middle| N_{X\setminus\{w\}}[u] \subseteq C^{\leqq v}(X \setminus \{w\}) \wedge \left|N_{X\setminus\{w\}}[u]\right| \geqq 2\right\}.$$

Proof. (I) Suppose that $z \in (Debt(w,X) \cup U(v,w,X)) \setminus (W(v,w,X) \cup T(v,w,X))$. If $z \in Debt(w,X)$, then since $z \notin W(v,w,X) \cup T(v,w,X)$, the number of neighbors of z in $C^{\leqq v}(X)$ is at least two. Moreover, $w \in C(X)$ and z is not included by $N\left(N^{\leq v}_{C(X)}(\Phi(v,X))\right)$. This implies z is not dominated by $(X \setminus \{w\}) \setminus C(X \setminus \{w\})$ Hence, $z \in Debt(v, X \setminus \{w\})$. On the other hands, from the definition of debt vertices, $U(v,w,X) \subseteq Debt(v, X \setminus \{w\})$. Thus, $z \in Debt(v, X \setminus \{w\})$. (II) Suppose that $z \in Debt(v, X \setminus \{w\})$. Since $\left|N_{X\setminus\{w\}}(z)\right| \geqq 2$, $z \notin W(v,w,X)$. Moreover, $z \notin T(v,w,X)$ since z is only dominated by $C(X \setminus \{w\})$. If $z \in N(w)$, then since $z \in Debt(v, X \setminus \{w\})$, $z \in U(v,w,X)$. If $z \notin N(w)$, then since $z \in Debt(v, X \setminus \{w\})$, $N_{X\setminus\{w\}}[z] = N_X[z]$. Hence, $z \in Debt(w,X)$. Thus, the statement holds. □

The algorithm constructs the following data structures in $O(n+m)$ time in preprocess: $Debt(v,V) = \emptyset$, $IV(u,V)$ for each $u \in V$, and $C(V)$, where v has the maximum index in V. When the algorithm computes the children of a current solution X, the algorithm also updates data structures of them. In the remaining of this section, we consider the time complexity for updating these data structures. We first show that computing $Debt(w, X \setminus \{v\})$ from $Debt(v,X)$ can be done in $O(k^2)$ time. During the execution, for each vertex w, EDS maintains the number $\delta(w)$ of neighbors of w that belong to $X^{>v}$, where X and v are a dominating set and a vertex in $C(X)$, respectively.

Lemma 8. *Let $v, w \in V$ such that $v < w$. Moreover, there is no vertex $x \in X$ such that $v < x < w$. If $V^{<v} \subseteq X$, then the time complexity for updating debt vertices both (1) from $Debt(w,X)$ to $Debt(v,X)$ and (2) from $Debt(w,X)$ to $Debt(v, X \setminus \{w\})$ is $O(k^2)$ time.*

Proof. Case (1): From Lemma 6, this can be done in $O(k)$ time. Case (2): We first consider how to obtain $W(v,w,X)$. For any vertex $u \in N^{>v}(w)$, we can compute $\left|N_{X\setminus\{w\}}[u]\right| = 1$ in constant time by checking the smallest and the second smallest neighbors of u since $V^{<v} \subseteq X$ holds and adjacency lists are sorted in the degeneracy ordering. If u also belongs to $Debt(w,X)$, then $u \in W$. This check can be done in constant time. Thus, $W(v,w,X)$ can be computed in $O(k)$ time. We next consider how to obtain $U(v,w,X)$. For any vertex $u \in N^{>v}(w)$, we can compute $\left|N_{X\setminus\{w\}}[u]\right| \geqq 2$ in constant time by the same fashion as $W(v,w,X)$. By checking whether u is in $Debt(w,X)$ or not, we can also compute $N_{X\setminus\{w\}}[u] \subseteq C^{\leqq v}(X)$ in constant time. Thus, $U(v,w,X)$ can be computed in $O(k)$ time. Finally, we consider how to obtain $T(v,w,X))$. From the definition, $\left|N^{\leq v}_{C(X)}(\Phi(v,X))\right| \leqq |\Phi(v,X)| \leqq k$ and u has the larger index than v. Hence, the size of $N\left(N^{\leq v}_{C(X)}(\Phi(v,X))\right)$ is at most k^2. Since we can decide a vertex z is included by $Debt(w,X)$ or not in constant time, the statement holds. □

Next, we consider how to find $IV(v,X)$, where v is a vertex such that $v < pv(X)$ and $v \in C(X \setminus \{pv(X)\})$. EDS maintains a super set $SIV(v,X)$ of $IV(v,X)$ for finding $IV(v,X)$. The definition of $SIV(v,X)$ is $SIV(v,X) = \{u \in C(X) \mid \arg\min\{N(u)\} = v\}$. We denote by $SIV(X) = \bigcup_{v \in C(X)} SIV(v,X)$.

Lemma 9. *Let v and u be a vertex in $C(X)$ such that $u < v$*

holds. If $V^{<u} \subseteqq X$*, then* $IV(v,X) \subseteqq SIV(v,X)$ *holds.*

Proof. Let w be a vertex in $IV(v,X)$. From the definition of $IV(v,X)$, $w < v < u$ and $N_X(w) = \{v\}$ holds. If w has a neighbor $x \notin X$, then $v < x$. Hence, $\arg\min\{N(w)\} = v$. Moreover, $IV(v,X) \subseteqq C(X)$. Thus, the statement holds. □

From Lemma 9, we can easily obtain $IV(v,X)$ from $SIV(v,X)$ by removing redundant vertices. From Lemma 8 and Lemma 9, we show the time complexity of each procedure `AllChildren` is $O\left(k^2|C(X)|\right)$

Lemma 10. `AllChildren` *except for recursive calls can be done in* $O\left(k^2|C(X)|\right)$.

Proof. Let v be a vertex in $C(X)$. We first consider the time complexity for computing debt vertices of a child $X \setminus \{v\}$ of X. Let u be a vertex satisfying $u < v$. From Lemma 8, the time complexity of Line 15 and Line 19 is $O\left(k^2\right)$ time. If a vertex $w \in C(X)$ is not included by $C(X \setminus \{v\})$, the algorithm maintains $SIV(\arg\min w, X \setminus \{v\})$ to dominate a vertex in $Debt(v,X)$. From Lemma 9, it can be done in constant time. Hence, the time complexity of from Line 14 to Line 19 is $O\left(k^2\right)$. Therefore, since each child needs $O(k)$ time to compute debt vertices, the time complexity for computing sets of debt vertices of all children of X is $O\left(k^2|C(X)|\right)$ times. Next, we consider the time complexity for updating $SIV(X)$. $|SIV(X)|$ is at most the size of $C(X)$ since each pair of sets in $SIV(X)$ have not same vertices. For each vertex $u \in SIV(v,X)$, $u \in IV(v,X)$ if $N_X[u] = \{u\}$. Hence, it can be done in $O(k)$ time for all sets of isolated vertices. Therefore, the time complexity of Line 11 to Line 13 is $O(k|SIV(X)|) = O(k|C(X)|)$ in `AllChildren`. Finally, Line 7 and Line 20 can be done in $O(k|C(X)|)$ time from Lemma 6. Hence, the time complexity of `AllChildren` is $O\left(k^2|C(X)|\right)$ except for recursive calls. The statement holds. □

We show EDS enumerates all dominating set in $O(k)$ time per solution.

Theorem 2. *EDS enumerates all dominating sets in a given graph* G *in* $O\left(k^2\right)$ *time per solution with* $O(\Delta n + m)$ *preprocess time and* $O(nm)$ *space.*

Proof. EDS constructs a degeneracy ordering and an adjacency list sorted by a degeneracy ordering in preprocess. From [9], we can compute a degeneracy ordering in $O(n+m)$ time and construct an adjacency list in $O(\Delta n)$ time by using bucket sort. In addition, EDS stores an input graph, a current solution, and three data structures. These data costs $O(n+m)$ space. Since the depth of $\mathcal{F}(G)$ is at most n. Hence, this algorithm needs $O(nm)$ space.

We consider the time complexity of EDS. From Lemma 10, each recursive call `AllChildren` needs $O\left(k^2|C(X)|\right)$ time. Hence, the total time of EDS is $O\left(\sum_{X \in \mathcal{S}(G)}(k^2|C(X)|)\right)$. Since $\mathcal{F}(G)$ forms tree from Lemma 2, each solution X except for the root solution V is a child of its parent. Hence, the sum of candidate sets X in $\mathcal{S}(G)$ is $O(|\mathcal{S}(G)|)$ and $O\left(\sum_{X \in \mathcal{S}(G)} k^2|C(X)|\right) = O\left(k^2|\mathcal{S}(G)|\right)$. Therefore, the time complexity of EDS is $O\left(\frac{k^2|\mathcal{S}(G)|}{|\mathcal{S}(G)|}\right)$ and the statement holds. □

From Theorem 2, we can easily obtain the following corollary.

Corollary 1. *If the degeneracy of an input graph is constant, e.g., an input graph is a planer graph, then EDS solves Problem 1 in constant time per solution.*

5. Conclusion

In this paper, we proposed the enumeration algorithm EDS. It solves the dominating set enumeration problem in $O\left(k^2\right)$ time per solution by using $O(nm)$ space, where k is a degeneracy of an input graph G. As a future work, we are interested in an efficient enumeration algorithm of dominating sets for dense graphs. If a graph is dense, then k is large and G has many dominating sets. For example, if an input graph is a complete graph, then k is equal to $n-1$ and every subset of a vertex set is a dominating set without $\emptyset$. Hence, G has $2^{|V|}-1$ dominating sets. When a graph is dense, the number of solution is larger than the number of solution in a sparse graph. Also, k is equal to Δ when a graph is dense EDS is not efficient for dense graphs nevertheless the number of solution is large. Hence, we are interested in more efficient enumeration algorithm for dense graphs.

6. Acknowledgements

This work was partially supported by JST CREST, Grant Number JPMJCR1401, Japan. The authors express their appreciation to Dr. Alessio Conte for fruitful discussions.

Reference

[1] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Appl. Math.*, 65(1):21–46, 1996.

[2] A. Conte, R. Grossi, A. Marino, and L. Versari. Sublinear-space bounded-delay enumeration for massive network analytics: Maximal cliques. In *ICALP 2016*, volume 148, pages 1–148, 2016.

[3] D. Eppstein, M. Löffler, and D. Strash. Listing all maximal cliques in large sparse real-world graphs. *J. Exp. Algorithmics*, 18:3.1:3.1–3.1:3.21, Nov. 2013.

[4] P. A. Golovach, P. Heggernes, M. M. Kanté, D. Kratsch, and Y. Villanger. Enumerating minimal dominating sets in chordal bipartite graphs. *Discrete Applied Mathematics*, 199:30–36, 2016.

[5] M. M. Kanté, V. Limouzy, A. Mary, and L. Nourine. On

the enumeration of minimal dominating sets and related notions. *SIAM Journal on Discrete Mathematics*, 28(4):1916–1929, 2014.

[6] M. M. Kanté, V. Limouzy, A. Mary, L. Nourine, and T. Uno. On the enumeration and counting of minimal dominating sets in interval and permutation graphs. In *ISSAC 2013*, pages 339–349. Springer, 2013.

[7] M. M. Kanté, V. Limouzy, A. Mary, L. Nourine, and T. Uno. Polynomial delay algorithm for listing minimal edge dominating sets in graphs. In *Workshop on Algorithms and Data Structures*, pages 446–457. Springer, 2015.

[8] D. R. Lick and A. T. White. k-degenerate graphs. *Canadian J. of Mathematics*, 22:1082–1096, 1970.

[9] D. W. Matula and L. L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM*, 30(3):417–427, 1983.

[10] C. H. Papadimitriou. Np-completeness: A retrospective. In *ICALP 1997*, pages 2–6, 1997.

[11] K. Wasa, H. Arimura, and T. Uno. Efficient enumeration of induced subtrees in a k-degenerate graph. In *ISAAC*, pages 94–102. Springer, 2014.