Efficient Enumeration of Dominating Sets in Sparse Graphs

Kazuhiro Kurita* Kunihiro Wasa† Takeaki Uno†
Hiroki Arimura*

Abstract

A dominating set is one of the fundamental graph structures. However, enumeration of dominating sets has not been paid much attention. In this paper, we enumerate all dominating sets. As a main result, we propose two efficient enumeration algorithms for sparse graphs. The one algorithm enumerates all dominating sets in $O(k)$ time per solution using $O(n+m)$ space, where k is degeneracy of an input graph. It is known that planar graphs have the constant degeneracy. Hence, this algorithm enumerates all dominating set in constant time per solution for a planar graph. The other algorithm enumerates all dominating sets in constant time per solution if an input graph with girth at least 7.

1 Introduction

An *enumeration problem* is a problem to output all solutions satisfying a given constraint without duplication. We evaluate the efficiency of an enumeration algorithm by *not only* the size of input *but also* the number of outputs [11]. Because the number of solutions in an input is often exponentially smaller than the maximum number of solutions estimated only the size of an input(See Fig. 1). 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. The number of solutions, in other words graph structure, is important for alaysis of enumeration algorithms. 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. In this paper, We propose two algorithms. the one is an amortized polynomial time algorithm that enumerates all dominating sets in a given graph G in $O(k)$ time per solution, where k is the *degeneracy* of G. The other is a amortized $O(1)$ time

*Hokkaido University, Japan
†National Institute of Informatics, Tokyo, Japan

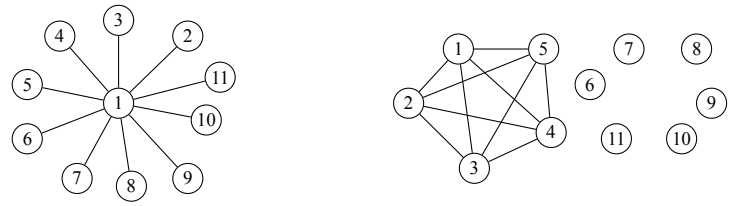

Figure 1: There are graphs with 11 vertices and 10 edges. G_1 has $2^{10}+1+1=1026$ dominating sets. On the other hands, G_2 has $2^5-1=31$ dominating sets. The number of solutions is exponentially different by graph structure even if the size of inputs are same.

algorithm that enumerates all solutions in a given graph G with *girth* at least 7.

Until now, researchers developed enumeration algorithms for minimal dominating sets with polynomial delay for some graph classes [6, 5]. In addition, incremental polynomial time algorithms are known [7, 4]. For the minimal edge dominating set enumeration problem, Kante et al. proposes a polynomial delay algorithm [7]. There results are obtained since these graph classes have a good property in graph structure. In this study, we focus on a *sparsity* of a graph as a good property of graph structure, in particular, the *degeneracy* and the *girth* of a graph G. These parameters are one of measures of sparseness of a graph. 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. We address to develop a strictly faster enumeration algorithm than the trivial algorithm. 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. 3.2. By using this ordering, some efficient enumeration algorithms have been developed [13, 2, 3]. On the other hands, the local structure for each vertex becomes tree if the girth of a graph G is sufficient large. By using this local tree structure, some W[1]-hard problems fall in FPT e.g. maximum induced matching and minimum dominating set [10, 12]. Moreover, an input graph has large girth, minimal dominating set can be enumerate in incremental polynomial time [7]. The organization of this paper is as follows: in Sect. 2 we define some basic notation. In Sect. 3 we give two algorithms to enumerate all dominating sets for sparse graphs. In Sect. 4, we conclude this paper and give future work.

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 $N_G(u)$

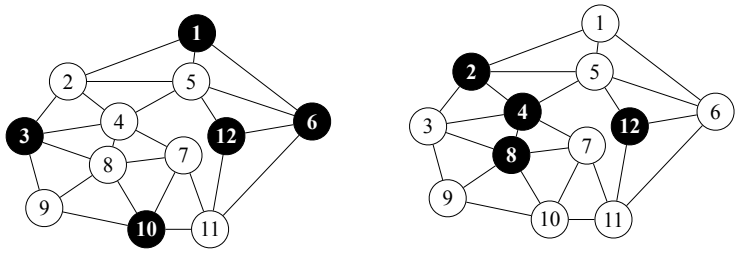

Figure 2: Black vertices indicate dominating set D in G.

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. An alternating sequence $\pi = (v_1, e_1, \ldots, e_k, v_k)$ of vertices and edges is a *path* if each edge and vertex in π appears at most once. We also call π an v_1-v_n *path*. An alternating sequence $C = (v_1, e_1, \ldots, e_k, v_k)$ of vertices and edges is a *cycle* if $(v_1, e_1, \ldots, v_{k-1})$ is a v_1-v_{k-1} path and $v_k = v_1$. The length of a path and a cycle is defined by the number of its edges. The *shortest path* between v_1 and v_n is the path π with minimum length in v_1-v_n path. The *girth* of a graph G is the length of a shortest cycle in G. 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. We show an example of a dominating set in Figure 2.

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

3 Enumeration of Dominating Sets in Sparse Graphs

In this section, we propose two algorithms EDS-D and EDS-G. These algorithms enumerate all dominating sets in sparse graphs. In this paper, we consider two parameter for sparsity of graphs, degeneracy and girth. EDS-D uses the degeneracy of a graph and EDS-G uses the girth of a graph. EDS-D and EDS-G are based on *reverse search method* [1]. Moreover, the correctness of EDS-D and EDS-G dependent on EDS. Hence, we explain EDS.

3.1 Basic Algorithm

To enumerate all solution by reverse search, we first define the *parent-child relation* between solutions. Let $G = (V, E)$ be an input graph with $V = \{v_1, \ldots, v_n\}$

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

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

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 X to Y on $\mathcal{F}(G)$ if X is the *parent* of Y. We call $\mathcal{F}(G)$ the *family tree* for G. By traversing on the family tree, we can obtain the all solutions.

We first define the parent-child relationship. 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* of X if $Y = X \cup \{pv(X)\}$. We denote by $\mathcal{P}(X)$ the parent of X. Note that since any superset of a dominating set also dominates G, $\mathcal{P}(X)$ is also 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. □

Our proposed algorithm EDS shown in Algorithm 1. We call $C(X) = \{v \in V \mid N[X \setminus \{v\}] = V \wedge \mathcal{P}(X \setminus \{v\}) = X\}$ in Algorithm 1 *a candidate set* of X. To prove Algorithm 1 outputs all solutions, we show a recursive procedure AllChildren generates all children on $\mathcal{F}(G)$. We denote $ch(X)$ is the set of children of X.

Lemma 3. *Let X and Y be distinct dominating sets in a graph G. $Y \in ch(X)$ if and only if there is a vertex $v \in C(X)$ such that $X = Y \cap \{v\}$.*

Proof. From the definition of $C(X)$, Y is a child of X. We assume that there is a dominating set Z such that $Z = X \setminus \{pv(Z)\}$ where $pv(Z) \notin C(X)$. From $pv(Z) \notin C(X)$, $N[\mathcal{P}(Z)] \neq V$ or $\mathcal{P}(X) \neq X$ hold. From the assumption Z is a child of X, $\mathcal{P}(Z) = X$ hold. Hence, $\mathcal{P}(Z) \neq X$ and $N[\mathcal{P}(Z)] \neq V$ do not hold since $\mathcal{P}(Z) = X$. It is contradict $pv(Z) \notin C(X)$. The statement holds. □

From Lemma 2 and Lemma 3, EDS traverses a family tree $\mathcal{F}(G)$ as a dfs-manner.

Theorem 4. *EDS outputs all dominating sets in G by using $O(n+m)$ space.*

To reduce the time complexity, the algorithm computes a candidate set efficiently. EDS-D and EDS-G maintain a candidate set efficiently by using the degeneracy and the local tree structure of a graph, respectively. In Sect. 3.2 and the Sect. 3.3, we use the following lemmas for the correctness of the candidate set maintenance.

Lemma 5. *Let X be a solution, v be a vertex in $C(X)$. For any vertex $u \in V \setminus X$, $v < u$ holds.*

Proof. We prove by contradiction. We assume that there is a vertex w such that $w \in V \setminus X$ and $w < v$. From the definition of parent-child relation, $v \neq pv(X \setminus \{v\})$. It is contradict $v \in C(X)$. The statement holds. □

Lemma 6. *Let X be a solution, v be a vertex in $C(X)$, and Y be a child solution of X, where $Y = X \setminus \{v\}$. Then, $C(Y) \subset C(X)$.*

Proof. We consider two conditions: (I)$N[Y] = V$ and (II)$\mathcal{P}(Y) = X$. Since Y is a child of X, if $u \in C(Y)$ holds, then u satisfy (I) in X. Moreover, $pv(Y) < pv(X)$ holds. Hence, u satisfy $u < pv(Y) < pv(X)$. In other words, u satisfy (II). Therefore, $u \in C(X)$. □

From the Lemma 6, when we computes $C(Y)$, we only consider removing vertices from $C(X)$. Let $Del(X, v)$ be the set of vertices $C(X) \setminus C(Y)$. $Del(X, v)$ satisfies the following lemma.

Lemma 7. *We define two sets of vertices $Del_1(X, v) = \{u \in C(X) \mid N[u] \cap X = \{u, v\}\}$ and $Del_2(X, v) = \{u \in C(X) \mid \exists w \in V(N[w] \cap X = \{u, v\})\}$. $Del(X, v) = Del_1(X, v) \cup Del_2(X, v)$ holds.*

Proof. $Del(X, v) \supseteq Del_1(X, v) \cup Del_2(X, v)$ is trivial since $X \setminus \{u, v\}$ is not dominating set. We prove $Del(X, v) \subseteq Del_1(X, v) \cup Del_2(X, v)$. Let u be a vertex in $Del(X, v)$. From the definition of $Del(X, v)$, $N[X \setminus \{u, v\}] \neq V$. Hence, there is a vertex w such that $N[w] \cap (X \setminus \{v\}) = \{u\}$. If $w = u$, then u is included by $Del_1(X, v)$. Otherwise, $N[w] \cap X = \{u, v\}$ holds. The statement holds. □

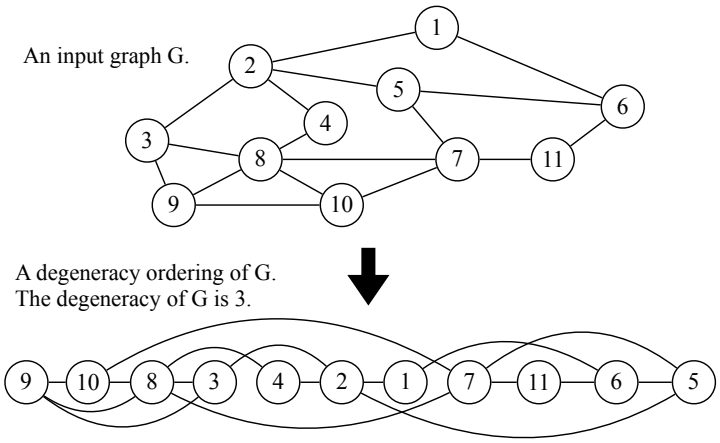

Figure 3: An example of a degeneracy ordering. In this ordering, each vertex v adjacent to vertices at most 3 with larger index than v Hence, G is a 3-degenerate graph and the degeneracy of G is 3.

3.2 Enumeration for Bounded Degenerate Graphs

We consider a candidate set maintenance and time complexity of EDS-D. To compute a candidate set efficiently, we maintain the adjacency list such that each vertex v has a list $N(v) \cap (X \setminus C(X))$ if $v \in V \setminus C(X)$. Otherwise, we maintain $N(v) \cap (X \setminus C(X)) \cap V^{<v}$. We denote the set of this adjacency list $\mathcal{D}(X)$ and each adjacency list for u D_u. We call this adjacency list *the dominated list* of u. To maintain a candidate set and the dominated lists, 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. 3. 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 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.

We consider two parts. In the first part, we find the set of vertices $C(X) \setminus C(Y)$. In the latter part, we maintain the dominated lists.

Lemma 8. *EDS-D computes $N(v) \cap C(X)$ and $N(v) \cap V^{v<} \cap X$ in $O(k|C(X)|)$ time for each $v \in C(X)$ in each recursive procedure.*

Proof. From the definition of k-degenerate graphs, the number of edges such that $e \cap C(X) \neq \emptyset$ is at most $k|C(X)|$. Hence, EDS-D can compute $N(v) \cap C(X)$

Algorithm 2: EDS-D enumerates all dominating sets in $O(k)$ time per solution.

```
1  Procedure EDS-D (G = (V, E))                          // G: an input graph
2      AllChildren(V, V, {∅_1, ..., ∅_|V|}, G);
3  Procedure AllChildren(X, C(X), D(X), G = (V, E))
4      Output X;
5      Z, W, D ← ∅, C(X), D(X);
6      for v ∈ C(X) do                                   // Y = X \ {v}
7          C(Y) ← Cand-D(X, v, W, G);
8          D(Y) ← DomList(Y, v, W, C(Y), Z, D, G);
9          AllChildren(Y, C(Y), D(Y), G);
10         Z, W ← C(Y), W \ {v};
11         for u ∈ N(v) ∩ V^{v<} do
12             D_u ← D_u ∪ {v};
13 Procedure Cand-D(X, v, C(X), G = (V, E))              // Y = X \ {v}
       //For each vertex u ∈ C(X), we obtain the set of vertex
         N(u) ∩ C(X) and N(u) ∩ V^{u<} ∩ X in preprocess.
14     Del_1, Del_2 ← ∅, ∅;
15     for u ∈ N(v) ∩ C(X) do
16         if u < v then
17             if N(u) ∩ V^{u<} ∩ Y = ∅ ∧ N(u) ∩ V^{<u} ∩ Y = ∅ then
18                 Del_1 ← Del_1 ∪ {u};
19         else
20             if N[u] ∩ (X \ C(X)) = ∅ ∧ |N[u] ∩ C(X)| = 2 then
21                 Del_2 ← Del_2 ∪ (N[u] ∩ C(X));
22     return C(X) \ (Del_1 ∪ Del_2 ∪ {v});
23 Procedure DomList(Y, v, W, C(Y), Z, D, G = (V, E))
24     Del, Ins ← W \ (C(Y) ∪ Z), Z \ C(Y);
25     for u ∈ Del do
26         for w ∈ N(u) ∩ V^{u<} do D_w ← D_w ∪ {u};
27     for u ∈ Ins do
28         for w ∈ N(u) ∩ V^{u<} do D_w ← D_w \ {u};
29     return D;
```

in $k|C(X)|$ time. Moreover, $|N(v) \cap V^{v<}|$ is at most k for each $v \in V$. The statement holds. □

From Lemma 8 and the definition of $\mathcal{D}$, we obtain the following corollary.

Corollary 9. *EDS-D has $N(v) \cap C(X)$ and $N(v) \cap V^{v<} \cap X$ for each $v \in C(X)$.*

Lemma 10. *For any vertex $v \in C(X)$, EDS-D computes $Del_1(X, v)$ in $O(k|C(X)|)$ in one recursion procedure.*

Proof. From the parent-child relation, $N(v) \cap C(X) \cap V^{v<} = \emptyset$. Hence, $Del_1(X, v) \subseteq N(v) \cap C(X)$. For each $w \in N(u) \cap C(X)$, EDS-D computes $N[w] \cap X = \{u, w\}$

or not in constant time from Corollary 9 and D_w. From the definition of $Del_1(X, v)$, $Del_1(X, v)$ can be computed in $|N(v) \cap C(X)|$. From the definition of k-degenerate graph, $\sum_{v \in C(X)} |N(v) \cap C(X)| \leq k|C(X)|$. The statement holds. □

Lemma 11. *For any vertex $v \in C(X)$, EDS-D computes $Del_2(X, v)$ in $O(k|C(X)|)$ time in one recursion procedure.*

Proof. From the parent-child relation, any $u \in N(v) \cap (V \setminus X)$ satisfies $v < u$. Since the definition of k-degenerate graph, $|Del_2(X, v)|$ is at most k for each $v \in C(X)$. Moreover, from the definition of $Del_2(X, v)$, if there is a vertex $w \in Del_2(X, v)$, then $N[u] \cap X = \{w, v\}$ holds. We assume that $u \notin X$. From the definition of $\mathcal{D}$, if $D_u \neq \emptyset$, then $N[u] \cap X = \{w, v\}$ does not hold. Otherwise, $N[u] \cap X \subseteq C(X)$. Hence, we can check $N[u] \cap X = \{w, v\}$ in constant time since $N(u) \cap C(X)$ is sorted by degeneracy ordering. Hence, for each $v \in C(X)$, EDS-D computes $Del_2(X, v)$ in $O(k)$ time. The statement holds. □

We consider maintenance of the dominated lists. When the algorithm removes a vertex v in $C(X)$, the dominated lists are maintained. From the definition of parent-child relation, when v is removed from X, $X \cap V^{v<}$ is not changed. Moreover, EDS-D makes a dominated list $N(v) \cap (X \setminus C(X)) \cap V^{v<}$ in Line 12. Hence, the following lemma holds.

Lemma 12. *Let v be a vertex picked in Line 6. For each vertex $u \in V$, D_u has $N(u) \cap (X \setminus (C(X) \cap V^{<v}))$ if $u \in V \setminus (C(X) \cap V^{v<})$ holds. Otherwise, D_u has $N(u) \cap (X \setminus (C(X) \cap V^{v<})) \cap V^{<u}$ holds.*

Proof. From the definition of $\mathcal{D}$, $D_u = N(v) \cap (X \setminus C(X))$. In Line 12, D_u becomes $D_u \cup \{v\}$. Moreover, a vertex does not removed from $C(X)$ otherwise in Line 12. Hence, the statement holds. □

From Lemma 12, when the algorithm translates Y, $\mathcal{D}$ is equal to $\mathcal{D}(Y)$, where Y is a child of X since $C(Y) \cap V^{<pv(Y)} = C(Y)$. Finally, we consider the total time complexity of EDS-D in one recursion procedure. We define exlusive or of A and B by $A \oplus B$.

Lemma 13. *EDS-D computes all children in $O(k|ch(X)| + k|gch(X)|)$ time in each recursion procedure.*

Proof. For each $v \in C(X)$, the total computation time of finding $Del(X, v)$ is $O(k|C(X)|)$. We consider the total time of maintenance of $\mathcal{D}$. Let v be a vertex picked in Line 6 and u be a vertex picked in Line 6 just before v. Then, Z be a set of vertices $C(X \setminus \{v\}) \oplus C(X \setminus \{u\})$. EDS-D needs $O(k|Z|)$ time to maintain $\mathcal{D}$. However, the number of children of $X \setminus \{v\}$ is at least $|C(X \setminus \{u\}) \setminus C(X \setminus \{v\})|$. Hence, the number of grand children of X is at least $|C(X) \setminus C(X \setminus \{w\})|$, where w is a minimum vertex in $C(X)$. $|C(X \setminus \{w\})|$ is at most $|C(X)|$. Hence, EDS-D computes all children in $O(k|C(X)| + k|gch(X)|)$ time. Therefore, The total time of EDS-D in each recursion procedure is $O(k|ch(X)| + k|gch(X)|)$ time. The statement holds. □

Theorem 14. *EDS-D enumerates all dominating sets in $O(k)$ time per solution in a k-degenerate graph by using $O(n + m)$ space.*

Algorithm 3: EDS-G enumerates all dominating sets in $O(1)$ time per solution for a graph with girth at least 7.

```
1  Procedure EDS-G (G = (V, E))                          // G: an input graph
2      AllChildren(V, V, {False, ..., False}, G);
3  Procedure AllChildren(X, C(X), F(X), G(X) = (V, E))
4      Output X;
5      W, F, H ← C(X), F(X), G(X);
6      for v ∈ C(X) do                                   // Y = X \ {v}
7          C(Y), F(Y), G(Y) ← Cand-G(v, W, F, H);
8          AllChildren(Y, C(Y), F(Y), G(Y));
9          for u ∈ N_H(v) do
10             if u ∈ W then f_u ← Ture ;
11             else H ← H \ {u} ;
12         H ← H \ {v};
13 Procedure Cand-G(v, W, F, H = (V, E)))
14     Del_1, Del_2 ← ∅, ∅;
15     for u ∈ N_H(v) do
16         if u ∈ W then
17             if N_H(u) ∩ W = {v} then Del_1, H ← Del_1 ∪ {u}, H \ N_H[u] ;
18         else if |N_H(u) ∩ W| = 2 then
19             Del_2 ← Del_2 ∪ {w} ;                     // w ∈ (N_H(u) ∩ W) \ {v}
20             for x ∈ N_H(w) do
21                 if x ∈ V \ W then H ← H \ {x};
22     if f_v = True then H ← H \ {v};
23     return W \ (Del_1 ∪ Del_2), F, H;
```

Proof. The correctness of EDS-D is shown by Theorem 4. We consider the space complexity of EDS-D. In each recursive procedure, EDS-D stores $\mathcal{D}$. However, the size of $\mathcal{D}$ is at most m. Hence, the space complexity of EDS-D is $O(n+m)$.

From Lemma 3, EDS-D enumerates all dominating sets. Moreover, each recursive procedure need $O(k|ch(X)| + k|gch(Y)|)$ time from Lemma 13. We analyze the time complexity of EDS-D. The time complexity of EDS-D is $O\left(k\sum_{X\in\mathcal{S}}(|ch(X)| + |gch(X)|)\right)$, where $\mathcal{S}$ is the set of solutions. Hence, $O\left(\sum_{X\in\mathcal{S}}(|ch(X)| + |gch(X)|)\right) = O(|\mathcal{S}|)$. Since each solution in $\mathcal{F}(G)$ is counted at most 2. the time complexity of EDS-D is amortized $O(k)$ time. □

3.3 Enumeration for Graphs with Girth at Least 7

We propose an optimum enumeration algorithm EDS-G for a graph with girth at least 7. EDS-G is based on EDS. the parent-child relation of EDS-G is same EDS. To achieve constant amortized time enumeration, EDS-G maintains a graph structure. This graph structure is an induced subgraph of an input graph. In EDS-G, we use an induced subgraph $G(X) = G[(V \setminus N[X \setminus C(X)]) \cup C(X)]$. In this subsedction, $N(v)$ implies $N_{G(X)}(v)$.

From Theorem 4, EDS-G enumerates all dominating sets. We consider the

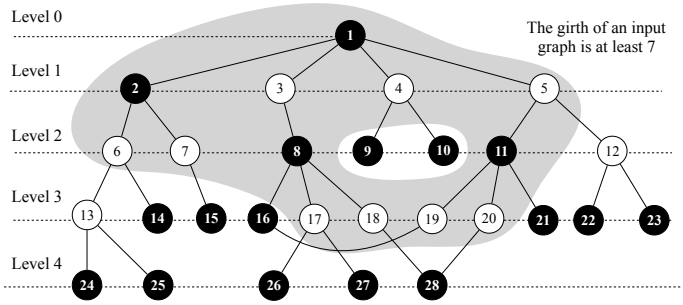

Figure 4: A example of $G(X)$. The vertices in the grey area are checked the neighbor.

time complexity of EDS-G. In EDS-G, we maintain an induced subgraph $G(X)$. The following lemma holds in $G(X)$.

Lemma 15. *Let X be a dominating set and v be a vertex in $G(X)$. Then, $|N[v] \cap C(X)|$ is at least 2.*

Proof. From the definition of $G(X)$, $N[v] \cap (X \setminus C(X)) = \emptyset$. We prove by contradiction, We assume that $|N[v] \cap C(X)|$ is 0 or 1. (I)$|N[v] \cap C(X)| = 0$: Since $N[v] \cap (X \setminus C(X)) = \emptyset$ from the definition of $G(X)$, it is contradict that X is a dominating set. (II)$|N[v] \cap C(X)| = 1$: Let u be a vertex in $N[v] \cap C(X)$ In this case, $X \setminus \{u\}$ is not a dominating set since $v \notin N[X \setminus \{u\}]$. Hence, it is contradict $u \in C(X)$. The statement holds. □

Let the set of vertices Z be $Del(X, v) \cup N[v]$.

Lemma 16. *In a translation from X to $Y = X \setminus \{v\}$, EDS-G finds the set of vertices Z in $O\left(\sum_{u \in Z} d(u)\right)$ time.*

Proof. We consider maintenance of the candidate set. From the definition of $Del_1(X, v)$, for each vertex $u \in N(v)$, we can compute satisfying $N[u] \cap X = u, v$ or not in $O(d(u))$ time. Moreover, from the definition of $Del_2(X, v)$, we can compute $Del_2(X, v)$ in $O\left(|Del_2(X, v)| + \sum_{u \in N(v)} d(u)\right)$ time by using bfs search. Hence, we can maintain the candidate set in $O\left(|Del(X, v)| + \sum_{u \in N(v)} d(u)\right)$ time. Moreover, we consider maintenance of the induced subgraph $G(X)$. It maintains in $\sum_{u \in Del(X,v)} d(u)$ time. The statement holds. □

In each translation, EDS-G needs $O\left(\sum_{u \in Z} d(u)\right)$ time. Next, we analyze the number of children and grand children of Y. We assume that if there is a vertex $u \in C(X)$ such that $N_G(u) \cap X = \emptyset$ and $N(u) = \{v\}$, then the cost of touching $d(u)$ is ignored since the sum of costs for touching such vertices is at most $O(C(X))$. Hence, we assume that $d(u) \geq 2$ for each $u \in C(X)$.

Lemma 17. *The number of children of Y is $O\left(\sum_{u \in N[v]} d(u)\right)$.*

Proof. From Lemma 3, the number of children of Y is equal to the size of $C(Y)$. Hence, we analyze the size of $C(Y)$. Let u be a vertex in $N(v)$. Each pair of $N(u) \setminus \{v\}$ and $N(u') \setminus \{v\}$ are disjoint for each u and u' since $G(Y)$ has the

girth at least 7. Moreover, for each vertex $w \in N(u) \setminus \{v\}$ and $w' \in N(u') \setminus \{v\}$, $N(w)$ and $N(w')$ are disjoint and $|N(w) \cap C(Y)| \geq 2$ from Lemma 15. Hence, the size of $C(Y)$ is at least $\sum_{u \in N(v)}(d(u) - 1)$. Moreover, $d(u) - 1 \geq 1$ since $d(u) \geq 2$ from the assumption. Hence, $\sum_{u \in N(v)}(d(u) - 1) \geq d(v)$. Therefore, $2\sum_{u \in N(v)}(d(u) - 1) \geq \sum_{u \in N[u]}(d(u))$. The statement holds. □

Lemma 18. *The total number of children and grand children of Y is $O\left(\sum_{u \in Del_2(X,v)} d(u)\right)$.*

Proof. Let u be a vertex in $N(v)$ and w be a vertex $N(u) \setminus \{v\}$. Then, there is a vertex $x \in C(Y) \cap N(w) \setminus \{u\}$ from Lemma 15. Moreover, for each u and w, x is mutually distinct since the girth of $G(X)$ is at least 7. If $x \in C(X)$, then $x \in C(Y)$. Let x' be a vertex in $N(w)$ such that $x' \notin C(Y)$ There is a vertex $y \in N(x) \cap C(Y) \setminus \{x\}$. Since there is a path (x', w, y), in other words $dist(x', y) = 3$, there is a grand child $Y \setminus \{x, y\}$. Hence, Y has children and grand children at least $|N(w)|$. Moreover, for each w and w', $N(w)$ and $N(w)$ are disjoint. The statement holds. □

From Lemma 17 and Lemma 18, the following lemma holds.

Lemma 19. *The number of children and grand children of Y is at least $\sum_{u \in Z} d(u)$.*

From Lemma 16 and Lemma 19, we show the following theorem.

Theorem 20. *For an input graph with at least 7, EDS-G enumerates all dominating sets in $O(1)$ time per solution by using $O(n + m)$ space.*

Proof. The correctness of EDS-G is shown by Theorem 4. The space complexity of EDS-G is $O(n + m)$ since the size of $G(X)$ is at most $n + m$.

From Lemma 16 and Lemma 19, in each translation from X to Y, EDS-G needs $O(|ch(Y)| + |gch(Y)| + c)$ steps. Hence, the time complexity of EDS-G is $O\left(\sum_{X \in \mathcal{S}}(|ch(X)| + |gch(X)| + c)\right)$. The total number of children and grand children in each node in $\mathcal{T}$ is $O(|\mathcal{T}|)$. Since the number of solutions is $|\mathcal{T}| = |\mathcal{S}|$, EDS-G enumerates all solutions amortized $O(1)$ time. The statement holds. □

4 Conclusion

In this paper, we proposed two enumeration algorithms It solves the dominating set enumeration problem in $O(k)$ time per solution by using $O(n + m)$ space, where k is a degeneracy of an input graph G. Moreover, it solves this problem in constant time per solution if an input graph have girth at least 7.

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. If an input graph has many solutions, then we may enumerate all solutions efficiently even if the time complexity of each recursive procedure is large. Also, k is equal to Δ when a graph is dense EDS-D is not efficient for dense graphs nevertheless the number of solution is large. Moreover, if G is small girth, then EDS-G does not achieve constant amortized time enumeration. Hence, we are interested in more efficient enumeration algorithm for dense graphs.

References

[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] H. Moser and S. Sikdar. The parameterized complexity of the induced matching problem. *Discrete Applied Mathematics*, 157(4):715–727, 2009.

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

[12] V. Raman and S. Saurabh. Short cycles make w-hard problems hard: Fpt algorithms for w-hard problems in graphs with no short cycles. *Algorithmica*, 52(2):203–225, 2008.

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