Efficient Enumeration of Dominating Sets for Sparse Graphs

Kazuhiro Kurita
IST, Hokkaido University, Sapporo, Japan
k-kurita@ist.hokudai.ac.jp

Kunihiro Wasa
National Institute of Informatics, Tokyo, Japan
wasa@nii.ac.jp

Hiroki Arimura
IST, Hokkaido University, Sapporo, Japan
arim@ist.hokudai.ac.jp

Takeaki Uno
National Institute of Informatics, Tokyo, Japan
uno@nii.ac.jp

Abstract

A dominating set D of a graph G is a set of vertices such that any vertex in G is in D or its neighbor is in D. Enumeration of minimal dominating sets in a graph is one of central problems in enumeration study since enumeration of minimal dominating sets corresponds to enumeration of minimal hypergraph transversal. However, enumeration of dominating sets including non-minimal ones has not been received much attention. In this paper, we address enumeration problems for dominating sets from sparse graphs which are degenerate graphs and graphs with large girth, and we propose two algorithms for solving the problems. The first algorithm enumerates all the dominating sets for a k-degenerate graph in $O(k)$ time per solution using $O(n+m)$ space, where n and m are respectively the number of vertices and edges in an input graph. That is, the algorithm is optimal for graphs with constant degeneracy such as trees, planar graphs, H-minor free graphs with some fixed H. The second algorithm enumerates all the dominating sets in constant time per solution for input graphs with girth at least nine.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Enumeration algorithm, polynomial amortized time, dominating set, girth, degeneracy

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

One of the fundamental tasks in computer science is to enumerate all subgraphs satisfying a given constraint such as cliques [**?**], spanning trees [**?**], cycles [**?**], and so on. One of the approaches to solve enumeration problems is to design exact exponential algorithms, i.e., *input-sensitive* algorithms. Another mainstream of solving enumeration problems is to design *output-sensitive* algorithms, i.e., the computation time depends on the sizes of both of an input and an output. An algorithm $\mathcal{A}$ is *output-polynomial* if the total computation time is polynomial of the sizes of input and output. $\mathcal{A}$ is an *incremental polynomial time algorithm* if the algorithm needs $O(poly(n,i))$ time when the algorithm outputs the i^{th} solution after outputting the $(i-1)^{\text{th}}$ solution, where $poly(\cdot)$ is a polynomial function. $\mathcal{A}$ runs in *polynomial amortized time* if the the total computation time is $O(poly(n)N)$, where n

and N are respectively the sizes of an input and an output. In addition, $\mathcal{A}$ runs in *polynomial delay* if the maximum interval between two consecutive solutions is $O(poly(n))$ time and the preprocessing and postprocessing time is $O(poly(n))$. From the point of view of tractability, efficient algorithms for enumeration problems have been widely studied [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**]. On the other hands, Lawler *et al.* show that some enumeration problems have no output-polynomial time algorithm unless $P = NP$ [**?**]. In addition, recently, Creignou *et al.* show a tool for showing the hardness of enumeration problems [**?**].

A dominating set is one of a fundamental substructure of graphs and finding the minimum dominating set problem is a classical NP-hard problem [**?**]. A vertex set D of a graph G is a dominating set of G if every vertex in G is in D or has at least one neighbors in D. The enumeration of *minimal* dominating sets of a graph is closely related to the enumeration of *minimal hypergraph transversals* of a hypergraph [**?**]. Kanté *et al.* [**?**] show that the minimal dominating set enumeration problem and the minimal hypergraph transversal enumeration problem are equivalent, that is, the one side can be solved in output-polynomial time if the other side can be also solved in output-polynomial time. Several algorithms that run in polynomial delay have been developed when we restrict input graphs, such as permutation graphs [**?**], chordal graphs [**?**], line graphs [**?**], graphs with bounded degenerate [**?**], graphs with bounded tree-width [**?**], graphs with bounded clique-width [**?**], and graphs with bounded (local) LMIM-width [**?**]. Incremental polynomial-time algorithms have also been developed, such as chordal bipartite graphs [**?**], graphs with bounded conformality [**?**], and graphs with girth at least seven [**?**]. Kanté *et al.* [**?**] show that the conformality of the closed neighbourhood hypergraphs of line graphs, path graphs, and (C_4, C_5, claw)-free graphs is constant. However, it is still open whether there exists an output-polynomial time algorithm for enumerating minimal dominating sets from general graphs.

Main results: In this paper, we consider the relaxed problem, i.e., enumeration of all dominating sets that include non-minimal ones in a graph. Since the number of solutions exponentially increases compared to the minimal version, even if we can develop an enumeration algorithm that runs in constant time per solution, the algorithm becomes theoretically much slower than some enumeration algorithm for minimal dominating sets. However, when we consider the real-world problem, we sometimes use another criteria for enumerating solutions that form dominating sets in a graph. That is, enumeration algorithms for minimal dominating sets may not fit in with other variations of minimal domination problems. E.g., a tropical dominating set [**?**] and a rainbow dominating set [**?**] are such a dominating set. Thus, when we enumerate solutions of such domination problems, our algorithm becomes a base-line algorithm for these problems. Thus, our main goal is to develop an efficient enumeration algorithm for dominating sets.

By straightforwardly using an enumeration framework such as the reverse search technique [**?**], we can obtain an enumeration algorithm for the problem that runs in $O(n)$ or $O(\Delta)$ time per solution, where n and Δ are respectively the number of vertices and the maximum degree of an input graph. Although dominating sets are fundamental in computer science, no enumeration algorithm for dominating sets that runs in strictly faster than such a trivial algorithm has been developed so far. Thus, to develop efficient algorithms, we focus on the *sparsity* of graphs as being a good structural property and, in particular, on the *degeneracy* and *girth*, which are the measures of sparseness. As our contributions, we develop two optimal algorithms for enumeration of dominating sets in a sparse graph. We first focus on the degeneracy of an input graph. The degeneracy [**?**] of a graph is k if any subgraph of the graph has a vertex whose degree is at most k. Note that $k \leq \Delta$ always holds. It is known that some graph classes have constant degeneracy, such as forests, grid graphs,

outerplanar graphs, planer graphs, bounded tree width graphs, and H-minor free graphs for some fixed H [**?**, **?**]. A k-degenerate graph has a *good* vertex ordering, called a *degeneracy ordering* [**?**], as shown in Section **??**. So far, this ordering has been used to develop efficient enumeration algorithms [**?**, **?**, **?**]. By using this ordering and the reverse search technique [**?**], we show that our proposed algorithm `EDS-D` can solve the relaxed problem in $O(k)$ time per solution, where k is degeneracy of an input graph. This implies that `EDS-D` can optimally enumerate all the dominating sets in an input graph with constant degeneracy.

We next focus on the girth of a graph. Enumeration of minimal dominating sets can be solved efficiently if an input graph has no short cycles since its connected subgraphs with small diameter form a tree. Indeed, this local tree structure has been used in minimal dominating sets enumeration [**?**]. For relaxed problem, by using the reverse search technique, we can easily show that the delay of our proposed algorithm `EDS-G` for general graphs is $O(\Delta^3)$ time. However, if an input graph has the large girth, then each recursive call generates enough solutions, that is, we can amortize the complexity of `EDS-G`. Thus, by amortizing the time complexity using this local tree structure, we show that the problem can be solve in constant time per solution for graphs with girth at least nine.

2 Prelminaries

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 $E(G)$ is a set of pairs of vertices 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 $N_G[u] = N_G(u) \cup \{u\}$. 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 $d_G(v) = |N_G(v)|$ be the *degree* of u in G. We call the vertex v *pendant* if $d_G(v) = 1$. $\Delta(G) = \max_{v \in V} d(v)$ denotes the maximum degree of G. A set X of vertices is a *dominating set* if X satisfies $N[X] = V$.

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. 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 path and $v_k = v_1$. The length of a path and a cycle is defined by the number of its edges. The *girth* of G is the length of a shortest cycle in G. We now define the dominating set enumeration problem as follows:

▶ Problem 1. Given a graph G, then output all dominating sets in G without duplication.

3 A Basic Algorithm Based on Reverse Search

In this paper, we propose two algorithms `EDS-D` and `EDS-G`. These algorithms use the degeneracy ordering and the local tree structure, respectively. Before we enter into details of them, we first show the basic idea for them, called *reverse search method* that is proposed by Avis and Fukuda [**?**] and is one of the framework for constructing enumeration algorithms.

An algorithm based on reverse search method enumerates solutions by traversing on an implicit tree structure on the set of solution, called a *family tree*. For building the family tree,

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
6 |  |  Y ← X \ {v};
7 |  |  C(Y) ← {u ∈ C(X) | N[Y \ {u}] = V ∧ P(Y \ {u}) = Y};
8 |  |  AllChildren(Y, C(Y), G);
9 |  return;
```

we first define the parent-child relationship between solutions as follows: 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 arbitrarily number the vertices in G from 1 to n and call the number of a vertex the *index* of the vertex. If no confusion occurs, we identify a vertex with its index. We assume that there is a total ordering $<$ on V according to the indices. $pv(X)$, 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, thus, $\mathcal{P}(X)$ is also a dominating set of G. We call X is a *child* of Y if $\mathcal{P}(X) = Y$. We denote 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. We call $\mathcal{F}(G)$ the *family tree* for G and call V the *root* of $\mathcal{F}(G)$. Next, we show that $\mathcal{F}(G)$ forms a tree rooted at V.

▶ **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 **??**, $\mathcal{F}(G)$ is connected. Hence, the statement holds. ◀

Our basic algorithm `EDS` is shown in Algorithm **??**. We say $C(X)$ the *candidate set* of X and define $C(X) = \{v \in V \mid N[X \setminus \{v\}] = V \wedge \mathcal{P}(X \setminus \{v\}) = X\}$. Intuitively, the candidate set of X is the set of vertices such that any vertex v in the set, removing v from X generates another dominating set. We show a recursive procedure `AllChildren`$(X, C(X), G)$ actually generates all children of X on $\mathcal{F}(G)$. We denote by $ch(X)$ the set of children of X, and by $gch(X)$ the set of grandchildren 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 $X = Y \cup \{v\}$.*

Proof. The if part is immediately shown from the definition of a candidate set. We show the only if part by contradiction. Let Z be a dominating set in $ch(X)$ such that $Z = X \setminus \{v'\}$, where $v' \in Z$. We assume that $v' \notin C(X)$. From $v' \notin C(X)$, $N[\mathcal{P}(Z)] \neq V$ or $\mathcal{P}(Z) \neq X$. Since Z is a child of X, $\mathcal{P}(Z) = X$, and thus, $N[\mathcal{P}(Z)] = V$. This contradicts $v' \notin C(X)$. Hence, the statement holds. ◀

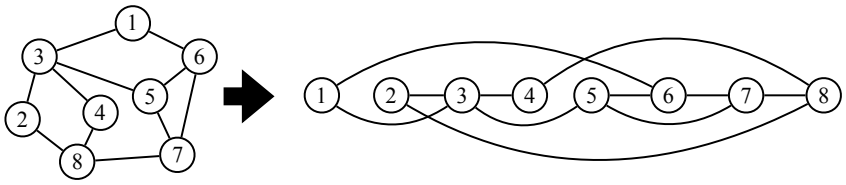

Figure 1 An example of a degeneracy ordering for a 2-degenerate graph G. In this ordering, each vertex v is adjacent to vertices at most two whose indices are larger than v.

From Lemma **??** and Lemma **??**, we can obtain the following theorem.

▶ **Theorem 4.** *By a DFS traversal of $\mathcal{F}(G)$,* `EDS` *outputs all dominating sets in G once and only once.*

4 Efficient Enumeration for Bounded Degenerate Graphs

The bottle-neck of `EDS` is the maintenance of candidate sets. Let X be a dominating set and Y be a child of X. We can easily see that the time complexity of `EDS` is $O(\Delta^2)$ time per solution since a removed vertex $u \in C(Y) \setminus C(X)$ has the distance at most two from v. In this section, we improve `EDS` by focusing on the degeneracy of an input graph G. G is a *k-degenerate graph* [**?**] 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 smallest k such that G is k-degenerate. A k-degenerate graph has a *good* ordering vertices. The definition of orderings of on vertices in G, called a *degeneracy ordering* of G, is as follows: 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. **??**. Matula and Beck show that the degeneracy and a degeneracy ordering of G can be obtained in $O(n+m)$ time [**?**]. Our proposed algorithm `EDS-D`, shown in Algorithm **??**, achieves amortized $O(k)$ time enumeration by using this good ordering. In what follows, we fix some degeneracy ordering of G and number the indices of vertices from 1 to n according to the degeneracy ordering. We assume that for each vertex v and each dominating set X, $N[v]$ and $C(X)$ are stored in a doubly linked list and sorted by the ordering. Note that the larger neighbors of v can be listed in $O(k)$ time. Let us denote by $V^{<v} = \{1, 2, \ldots, v-1\}$ and $V^{v<} = \{v+1, \ldots n\}$. Moreover, $A^{<v} := A \cap V^{v<}$ and $A^{v<} := A \cap V^{<v}$ for a subset A of V. We first show the relation between $C(X)$ and $C(Y)$.

▶ **Lemma 5.** *Let X be a dominating set of G and Y be a child of X. Then, $C(Y) \subset C(X)$.*

Proof. Let Z be a child of Y. Hence, $pv(Z) \in X$ and $pv(Z) \in C(Y)$. From the definition of $pv(Z)$, $pv(Z) = \min V \setminus Z$. Moreover, since $V \setminus X \subset V \setminus Z$, $pv(Z) \leq \min V \setminus X$. Therefore, $pv(Z) \in C(X)$. ◀

From the Lemma **??**, for any $v \in C(X)$, what we need to obtain the candidate set of Y is to compute $Del(X, pv(Y)) := C(X) \setminus C(Y)$, where $Y = X \setminus \{v\}$. In addition, we can easily sort $C(Y)$ by the degeneracy ordering if $C(X)$ is sorted. In what follows, we denote by $Del_1(X,v) = \{u \in C(X) \mid N[u] \cap X = \{u, v\}\}$, $Del_2(X,v) = \{u \in C(X) \mid \exists w \in V \setminus X (N[w] \cap X = \{u, v\})\}$, and $Del_3(X,v) = C(X)^{v\leq}$. By the following lemmas, we show the time complexity for obtaining $Del(X, pv(Y))$.

▶ **Lemma 6.** *Suppose that $v \in C(X)$. Then, $Del(X,v) = Del_1(X,v) \cup Del_2(X,v) \cup Del_3(X,v)$.*

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

Procedure `EDS-D`$(G = (V, E))$ `// G: an input graph`
  foreach $v \in V$ **do** $D_v \leftarrow \emptyset$;
  `AllChildren`$(V, V, \mathcal{D}(V) := \{D_1, \ldots, D_{|V|}\})$;
Procedure `AllChildren`$(X, C, \mathcal{D})$
  Output X;
  $C' \leftarrow \emptyset$; $\mathcal{D}' \leftarrow \mathcal{D}$; `//` $\mathcal{D}' := \{D'_1, \ldots, D'_{|V|}\}$
  for $v \in C$ **do** `//` v `has the largest index in` C
    $Y \leftarrow X \setminus \{v\}$;
    $C \leftarrow C \setminus \{v\}$; `//` `Remove vertices in` $Del_3(X, v)$.
    $C(Y) \leftarrow$ `Cand-D`(X, v, C); `//` `Vertices larger than` v `are not in` C.
    $\mathcal{D}(Y) \leftarrow$ `DomList`$(v, Y, X, C(Y), C' \oplus C(Y), \mathcal{D}')$;
    `AllChildren`$(Y, C(Y), \mathcal{D}(Y))$;
    $C' \leftarrow C(Y)$; $\mathcal{D}' \leftarrow \mathcal{D}(Y)$;
    for $u \in N(v)^{v<}$ **do** $D'_u \leftarrow D'_u \cup \{v\}$;
Procedure `Cand-D`(X, v, C)
  $Y \leftarrow X \setminus \{v\}$; $Del_1 \leftarrow \emptyset$; $Del_2 \leftarrow \emptyset$;
  for $u \in N(v) \cap C$ **do**
    if $u < v$ **then**
      if $N(u)^{u<} \cap Y = \emptyset \wedge N(u)^{<u} \cap Y = \emptyset$ **then** $Del_1 \leftarrow Del_1 \cup \{u\}$;
    else
      if $N[u] \cap (X \setminus C) = \emptyset \wedge |N[u] \cap C| = 2$ **then** $Del_2 \leftarrow Del_2 \cup (N[u] \cap C)$;
  return $C \setminus (Del_1 \cup Del_2)$; `//` C `is` $C(X \setminus \{v\})$
Procedure `DomList` $(v, Y, X, C' \oplus C(Y), \mathcal{D}')$
  for $u \in C' \oplus C(Y)$ **do**
    for $w \in N'(u)^{u<}$ **do**
      if $u \notin D'_w(X)$ **then**
        if $u \notin C'$ **then** $D'_w \leftarrow D'_w \cup \{u\}$;
        else $D'_w \leftarrow D'_w \setminus \{u\}$;
  for $u \in N(v)^{v<}$ **do**
    if $u \in X$ **then** $D'_v \leftarrow D'_v \cup \{u\}$;
  return $\mathcal{D}'$; `//` $\mathcal{D}'$ `is` $\mathcal{D}(Y)$

Proof. $Del(X, v) \supseteq Del_1(X, v) \cup Del_2(X, v) \cup Del_3(X, v)$ is trivial since $X \setminus \{u, v\}$ is not dominating set for each $u \in Del_1(X, v) \cup Del_2(X, v)$ and the parent of $X \setminus \{u, v\}$ is not $X \setminus \{v\}$ for each $u \in Del_3(X, v)$. We next prove $Del(X, v) \subseteq Del_1(X, v) \cup Del_2(X, v)\, Del_3(X, v)$. Let u be a vertex in $Del(X, v)$. Suppose that $X \setminus \{u, v\}$ is a dominating set. Since $\mathcal{P}(X \setminus \{u, v\}) \neq X \setminus \{v\}$, $v < u$. Thus, $u \in Del_3(X, v)$. Suppose that $X \setminus \{u, v\}$ is not a dominating set, that is, $N[X \setminus \{u, v\}] \neq V$. This implies that there exists a vertex w in V such that w is not dominated by any vertex in $X \setminus \{u, v\}$. Note that w may be equal to u. Hence, $N[w] \cap X = \{u, v\}$ and the statement holds. ◀

We show an example of dominated list and a maintenance of $C(X)$ in Fig. **??**. To compute a candidate set efficiently, for each vertex u in V, we maintain the vertex lists $D_u(X)$ for X. We call $D_u(X)$ the *dominated list* of u for X. The definition of $D_u(X)$ is as follows: If $u \in V \setminus X$, then $D_u(X) = N(u) \cap (X \setminus C(X))$. If $u \in X$, then $D_u(X) = N(u)^{<u} \cap (X \setminus C(X))$.

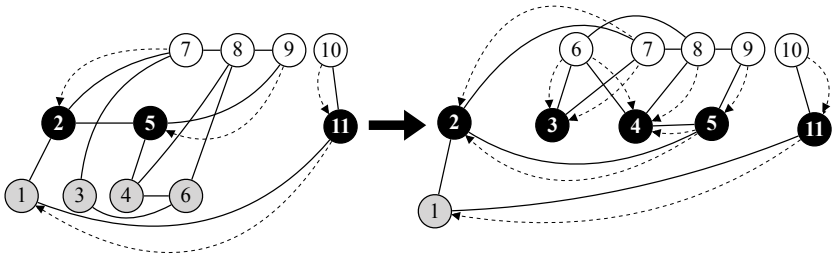

Figure 2 Let X be a dominating set $\{1, 2, 3, 4, 5, 6\}$. An example of the maintenance of $C(X)$ and $\mathcal{D}(X)$. Each dashed directed edge is stored in $\mathcal{D}(X)$, and each solid edge is an edge in G. A directed edge (u, v) implies $v \in D_u(X)$. The index of each vertex is according to a degeneracy ordering. White, black, and gray vertices belong to $V \setminus X$, $X \setminus C(X)$, and $C(X)$, respectively. When `EDS-D` removes vertex 6, $C(X \setminus \{6\}) = \{1\}$.

For brevity, we write D_u as $D_u(X)$ if no confusion arises. We denote by $\mathcal{D}(X) = \bigcup_{u \in V} \{D_u\}$. By using $\mathcal{D}(X)$, we can efficiently find $Del_1(X, v)$ and $Del_2(X, v)$.

▶ **Lemma 7.** *For each vertex $v \in C(X)$, we can compute $N(v) \cap C(X)$ and $N(v)^{v<} \cap X$ in $O(k)$ time on average over all children of X.*

Proof. Since G is k-degenerate, $G[C(X)]$ is also k-degenerate. Thus, the number of edges in $G[C(X)]$ is at most $k|C(X)|$. Remind that $C(X)$ is sorted by the degeneracy ordering. Hence, by scanning vertices of $C(X)$ from the smallest vertex to the largest one, for each v in $C(X)$, we can obtain $N(v) \cap C(X)$ in $O(k)$ time on average over all children of X. Since $N(v)^{v<}$ is the larger v's neighbors set, the size is at most k. Hence, the statement holds. ◀

▶ **Lemma 8.** *Let X be a dominating set of G. Suppose that for each vertex u in G, we can obtain the size of D_u in constant time. Then, for each vertex $v \in C(X)$, we can compute $Del_1(X, v)$ in $O(k)$ time on average over all children of X.*

Proof. Since every vertex u in $Del_1(X, v)$ is adjacent to v, $Del_1(X, v) \subseteq N(v) \cap C(X)$. To compute $Del_1(X, v)$, we need to check whether $N[u] \cap X = \{u, v\}$ or not. We first consider smaller neighbors of u. Before computing $Del_1(X, v)$ for every vertex v, we record the size of D_u of $u \in C(X)$ in $O(|C(X)|)$ time. $D_u = \emptyset$ if and only if there are no smaller neighbors of u in $X^{<u} \setminus C(X)$. Moreover, the number of edges in $G[C(X)]$ is at most $k|C(X)|$ from the definition of the degeneracy. Thus, this part can be done in $O\left(\sum_{v \in C(X)} |N(v) \cap C(X)|\right)$ total time and in $O(k)$ time per each vertex in $C(X)$. We next consider larger neighbors. Again, before computing $Del_1(X, v)$ for every vertex v, from Lemma **??** and the degeneracy of G, we can check all of the larger neighbors of $u \in C(X)$ in $O(k|C(X)|)$ time. Thus, as with the smaller case, the checking for the larger part also can be done in $O(k)$ time on average over all children of X. Hence, the statement holds. ◀

▶ **Lemma 9.** *Suppose that for each vertex w in G, we can obtain the size of D_w in constant time. For each vertex $v \in C(X)$, we can compute $Del_2(X, v)$ in $O(k)$ time on average over all children of X.*

Proof. Let u be a vertex in $Del_2(X, v)$. Then, there exists a vertex w such that $N[w] \cap X = \{u, v\}$ and $w \in N(v) \cap (V \setminus X)$. In addition, for any vertex v' in $C(X)$, $pv(X \setminus \{v'\}) = v'$. Thus, $v < w$ and $u < w$ hold. Before computing $Del_2(X, v)$ for every vertex v, by scanning all larger neighbors w' of vertices of $C(X)$, we can list such vertices w' such that

$w' > \max\{C(X)\}$, $|N[w'] \cap C(X)| = 2$, and $w' \in V \setminus X$ in $O(k|C(X)|)$ time since G is k-degenerate. If $D_{w'} \neq \emptyset$, that is, w' has a neighbor in $X \setminus C(X)$, then $|N[w] \cap X| > 2$. Thus, since we can check the size of $D_{w'}$ in constant time, we can compute $Del_2(X, v)$ in $O(k)$ time on average over all children of X. ◀

In Lemma **??** and Lemma **??**, we assume that the dominated lists were computed when we compute $Del(X, v)$ for each vertex v in $C(X)$. We next consider how we maintain $\mathcal{D}$. Next lemmas show the transformation from $D_u(X)$ to $D_u(Y)$ for each vertex u in G. Due to limitations of space, we omit the proofs of them. See the appendix for the proofs.

▶ **Lemma 10.** *Let X be a dominating set, v be a vertex in $C(X)$, and $Y = X \setminus \{v\}$. For each vertex $u \in G$ such that $u \neq v$, $D_u(Y) = D_u(X) \cup (N(u)^{<u} \cap (Del_1(X, v) \cup Del_2(X, v))) \cup (N(u)^{<u} \cap (Del_3(X, v) \setminus \{v\}))$.*

▶ **Lemma 11.** *Let X be a dominating set, v be a vertex in $C(X)$, and $Y = X \setminus \{v\}$. $D_v(Y) = D_v(X) \cup (N(v)^{<v} \cap (Del_1(X, v) \cup Del_2(X, v))) \cup (N(v)^{v<} \cap X)$.*

We next consider the time complexity for obtaining the dominated lists for children of X. From Lemma **??** and Lemma **??**, a naïve method for the computation needs $O(k|Del(X, v)| + k)$ time for each vertex v of X since we can list all larger neighbors of any vertex in $O(k)$ time. However, if we already know $C(W)$ and $\mathcal{D}(W)$ for a child W of X, then we can easily obtain $\mathcal{D}(Y)$, where Y is the child of X immediately after W. The next lemma plays a key role in `EDS-D`. Here, for any two sets A, B, we denote by $A \oplus B = (A \setminus B) \cup (B \setminus A)$.

▶ **Lemma 12.** *Let X be a dominating set, v, u be vertices in $C(X)$ such that u has the minimum index in $C(X)^{v<}$, $Y = X \setminus \{u\}$, and $W = X \setminus \{v\}$. Suppose that we already know $C(Y) \oplus C(W)$, $\mathcal{D}(W)$, $Del(X, v)$, and $Del(X, u)$. Then, we can compute $\mathcal{D}(Y)$ in $O(k|C(Y) \oplus C(W)| + k)$ time.*

Proof. Suppose that z is a vertex in G such that $z \neq v$ and $z \neq u$. From the definition, $D_z(W) \setminus D_z(Y) = (Del(X, v) \setminus Del(X, u)) \cap N(z)^{<z}$ and $D_z(Y) \setminus D_z(W) = (Del(X, u) \setminus Del(X, v)) \cap N(z)^{<z}$. Hence, we first compute $A = Del(X, v) \oplus Del(X, u)$. Now, $A = (C(X) \setminus C(W)) \oplus (C(X) \setminus C(Y)) = C(W) \oplus C(Y)$. Next, for each vertex c in $C(W) \oplus C(Y)$, we check whether we add to or remove c from $D_z(Y)$ or not. Note that added or removed vertices from $D_z(Y)$ is a smaller neighbor of z. From the definition, if $c \notin D_z(Y)$ or $c \in D_z(X)$, then we add c to $D_z(Y)$. Otherwise, we remove c from $D_z(Y)$. Thus, since each vertex in $C(W) \oplus C(Y)$ has at most k larger neighbors, for all vertices other than u and v, we can compute the all dominated lists in $O(k|C(W) \oplus C(Y)|)$ time. Next we consider the update for $D_u(Y)$ and $D_v(Y)$. Note that from the definition, $D_v(W)$ and $D_u(Y)$ contain larger neighbors of v and u, respectively. However, the number of such neighbors is $O(k)$. Finally, since v belongs to Y, $v \in D_{u'}(Z)$ if $u' \in N(v)^{v<}$ for any vertex u'. Thus, as with the above discussion, we can compute $D_u(Y)$ and $D_v(Y)$ in $O(k|C(W) \oplus C(Y)| + k)$ time. Hence, the statement holds. ◀

Note that we can compute $C(Y) \oplus C(W)$ when we compute $C(Y)$ and $C(W)$. From the above discussion, we can obtain the time complexity of `AllChildren` in `EDS-D`.

▶ **Lemma 13.** *Let X be a dominating set. Then, `AllChildren`$(X, C(X), \mathcal{D}(X))$ of `EDS-D` other than recursive calls can be done in $O(k|ch(X)| + k|gch(X)|)$ time.*

Proof. We first consider the time complexity of `Cand-D`. From Lemma **??**, `Cand-D` correctly computes $Del_1(X, v)$ and $Del_2(X, v)$ in from line **??** to line **??** and from line **??** to line **??**,

respectively. For each loop from line **??**, the algorithm picks the largest vertex in C. This can be done in $O(1)$ since C is sorted. The algorithm needs to remove vertices in $Del_3(X, v)$. This can be done in line **??** and in $O(1)$ time since v is the largest vertex. Thus, for each vertex v in $C(X)$, $C(X \setminus \{v\})$ can be obtained in $O(k)$ time on average. Hence, for all vertices in $C(X)$, the candidate sets can be computed in $O(k|ch(X)|)$ time. Next, we consider the time complexity of `DomList`. Before computing `DomList`, `EDS-D` already computed $C(Y) \oplus C(W)$, $\mathcal{D}(W)$, $Del(X, v)$, and $Del(X, v')$. Here, W is the previous dominating set, C' stores $C(W)$, and $\mathcal{D}'$ stores $\mathcal{D}(W)$. Thus, by using Lemma **??**, we can compute $\mathcal{D}(Y)$ in $O(k|C(Y) \oplus C(W)| + k)$ time. In addition, for all vertices in $C(X)$, the dominated lists can be computed in $O(k|gch(X)|)$ time since Y has a child at least $|C(W) \setminus C(Y)|$. When `EDS-D` copies data such as $\mathcal{D}$, `EDS-D` only copies the pointer of these data. By recording operations of each line, `EDS-D` restores these data when backtracking happens. These restoring can be done in the same time of the above update computation. Hence, 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.*

Proof. The correctness of `EDS-D` is shown by Theorem **??**. We next consider the space complexity of `EDS-D`. For any vertex v in G, if v is removed from a data structure used in `EDS-D` on a recursive procedure, v will never be added to the data structure on descendant recursive procedures. In addition, for each recursive procedure, the number of data structures that are used in the procedure is constant. Hence, the space complexity of `EDS-D` is $O(n + m)$. We finally consider the time complexity. Each recursive procedure needs $O(k|ch(X)| + k|gch(Y)|)$ time from Lemma **??**. Thus, 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. Now, $O\left(\sum_{X \in \mathcal{S}}(|ch(X)| + |gch(X)|)\right) = O(|\mathcal{S}|)$. Hence, the statement holds. ◀

5 Efficient Enumeration for Graphs with Girth at Least Nine

In this section, we propose an optimum enumeration algorithm `EDS-G` for graphs with girth at least nine. That is, the proposed algorithm runs in constant amortized time per solution for such graphs. The algorithm is shown in Algorithm **??**. To achieve constant amortized time enumeration, we focus on the *local structure* $G_v(X)$ for (X, v) of G defined as follows: $G_v(X) = G[(V \setminus N[X \setminus C(X)^{\leq v}]) \cup C(X)^{\leq v}]$. Fig. **??** shows an example of $G_v(X)$. $G_v(X)$ is a subgraph of G induced by vertices that (1) are dominated by vertices only in $C(X)^{\leq v}$ or (2) are in $C(X)^{\leq v}$. Intuitively speaking, we can efficiently enumerate solutions by using the local structure and ignoring vertices in $G \setminus G_v(X)$ since the number of solutions that are generated according to the structure is enough to reduce the *amortized* time complexity to constant. We denote by $G(X) = G[(V \setminus N[X \setminus C(X)]) \cup C(X)]$ the local structure for (X, v_*) of G, where v_* is the largest vertex in G. Due to space limitations, we will omit proofs of this section (see the appendix for the proofs).

We first consider the correctness of `EDS-G`. The parent-child relation between solutions used in `EDS-G` is the same as in `EDS`. Suppose that X and Y are dominating sets such that X is the parent of Y. Recall that, from Lemma **??**, $C(X) \setminus C(Y) = Del(X, v)$, where $X = Y \cup \{v\}$. We denote by $f_v(u, X) =$ `True` if there exists a neighbor w of u such that $w \in X \setminus C(X)^{\leq v}$; Otherwise $f_v(u, X) =$ `False`. Thus, `Cand-G` correctly computes $Del_1(X, v)$ and $Del_2(X, v)$ from line **??** to **??**. Moreover, in line **??**, vertices in $Del_3(X, v)$ are removed from $C(X)$ and hence, `Cand-G` also correctly computes $C(X \setminus \{v\})$. Moreover,

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

Procedure `EDS-G`$(G = (V, E))$ `// G: an input graph`
  foreach $v \in V$ **do** $f_v \leftarrow$ `False` ;
  `AllChildren` $(V, V, \{f_1, \ldots, f_{|V|}\}, G)$;
Procedure `AllChildren` (X, C, F, G)
  Output X;
  for $v \in C(X)$ **do** `// v is the largest vertex in C`
    $Y \leftarrow X \setminus \{v\}$;
    $(C(Y), F(Y), G(Y)) \leftarrow$ `Cand-G` (v, C, F, G);
    `AllChildren` $(Y, C(Y), F(Y), G(Y))$;
    for $u \in N_G(v)$ **do**
      if $u \in C$ **then** $f_u \leftarrow$ `True` ;
      else $G \leftarrow G \setminus \{u\}$;
    $G \leftarrow G \setminus \{v\}$;
    $C \leftarrow C \setminus \{v\}$; `// Remove vertices in` $Del_3(X, v)$.
Procedure `Cand-G` (v, C, F, G)
  $Del_1 \leftarrow \emptyset$; $Del_2 \leftarrow \emptyset$;
  foreach $u \in N_G(v)$ **do**
    if $N_G[u] \cap X = \{u, v\}$ *and* $f_u =$ `False` **then** $Del_1 \leftarrow Del_1 \cup \{u\}$;
    else if $\exists w (N_G[u] \cap X = \{w, v\})$ **then** $Del_2 \leftarrow Del_2 \cup \{w\}$;
  $C' \leftarrow C \setminus (Del_1 \cup Del_2 \cup \{v\})$;
  foreach $u \in N'[Del_1 \cup Del_2]$ **do** `// Lemma ??`
    $f_u \leftarrow$ `True`;
    if $u \notin C'$ **then** $G \leftarrow G \setminus \{u\}$;
  if $f_v =$ `True` **then** $G \leftarrow G \setminus \{v\}$;
  return (C', F, G);

for each vertex w removed from G during enumeration, w is dominated by some vertices in G. Hence, by the same discussion as Theorem **??**, we can show that `EDS-G` enumerates all dominating sets. In the remaining of this section, we show the time complexity of `EDS-G`. Note that $G_v(X)$ does not include any vertex in $N[Del_3(X, v) \setminus \{v\}] \setminus C(X)^{\leq v}$. Hence, we will consider only vertices in $Del_1(X, v) \cup Del_2(X, v) \cup \{v\}$. We denote by $Del'(X, v) = Del_1(X, v) \cup Del_2(X, v) \cup \{v\}$. We first show the time complexity for updating the candidate sets.

In what follows, if v is the largest vertex in $C(X)$, then we simply write $f(u, X)$ as $f_v(u, X)$. We denote by $N'_v(u) = N_{G_v(X)}(u)$, $N'_v[u] = N'_v(u) \cup \{u\}$, and $d'_v(u) = |N'_v(u)|$ if no confusion arises. Suppose that G and $G_v(X)$ are stored in an adjacency list, and neighbors of a vertex are stored in a doubly linked list and sorted in the ordering.

▶ **Lemma 15.** *Let X be a dominating set and v be a vertex in $C(X)$. Suppose that for any vertex u, we can check the number of u's neighbors in the local structure $G_v(X)$ and the value of $f_v(u, X)$ in constant time. Then, we can compute $C(X \setminus \{v\})$ from $C(X)^{\leq v}$ in $O(d'_v(v))$ time*

▶ **Lemma 16.** *Let X be a dominating set, v be a vertex in $C(X)$, and $Y = X \setminus \{v\}$. Then, we can compute $G(Y)$ from $G_v(X)$ in $O\left(\sum_{u \in Del'(X,v)} d'_v(u) + \sum_{u \in G_v(X) \setminus G(Y)} d'_v(u)\right)$ time.*

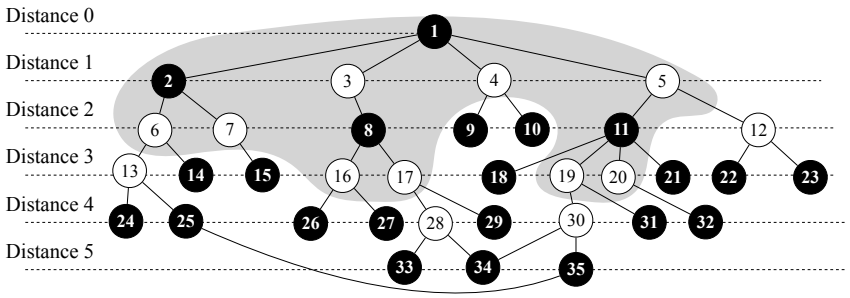

Figure 3 An example of $G_v(X)$. The vertices in the grey area are $Del'(X,v) \cup (G_v(X) \setminus G(Y)) \cup (N'_v(v) \setminus X)$. Each horizontal line represents the distance between 1 and any vertex.

Note that $N'_v(u) = N_{G_v(X)}(u)$ *and* $d'_v(u) = |N'_v(u)|$.

From Lemma **??** and Lemma **??**, we can compute the local structure and the candidate set of Y from those of X in $O\left(\sum_{u \in Del'(X,v)} d'_v(u) + \sum_{u \in G_v(X) \setminus G(Y)} d'_v(u)\right)$ time. We next consider the time complexity of the loop in line **??**. In this loop procedure, `EDS-G` deletes the all the neighbors u of v from $G_v(X)$ if $u \notin C(X)^{\leq v}$ because for each descendant W of dominating set Y', $v \in W \setminus C(W)$, where Y' is a child of X and is generated after Y. Thus, this needs $O\left(d'_v(v) + \sum_{u \in N'(v) \setminus X} d'_v(u)\right)$ time. Hence, from the above discussion, we can obtain the following lemma:

▶ **Lemma 17.** *Let X be a dominating set, v be a vertex in $C(X)$, and $Y = X \setminus \{v\}$. Then,* `AllChildren` *other than a recursive call runs in the following time bound:*

$$O\left(\sum_{u \in Del'(X,v)} d'_v(u) + \sum_{u \in G_v(X) \setminus G(Y)} d'_v(u) + \sum_{u \in N'_v(v) \setminus X} d'_v(u)\right). \tag{1}$$

Before we analyze the number of descendants of X, we show the following lemmas.

▶ **Lemma 18.** *Let us denote by $Pen_v(X) = \{u \in Del'(X,v) \mid d'_v(u) = 1\}$. Then, $\sum_{v \in C(X)} |Pen_v(X)|$ is at most $|C(X)|$.*

From Lemma **??**, $|Pen(X)|$ is at most $|C(X)|$. Let v be a vertex in $C(X)$ and a pendant in $G_v(X)$. Since the number of such pendants is at most $|C(X)|$, the sum of degree of such pendants is at most $|C(X)|$ in each execution of `AllChildren` without recursive calls. Hence, the cost of deleting such pendants is $O(|C(X)|)$ time. Next, we consider the number of descendants of X. From Lemma **??**, we can ignore such pendant vertices. Hence, for each $u \in Del'(X,v)$, we will assume that $d'_v(u) \geq 2$ below.

▶ **Lemma 19.** *Let X be a dominating set, v be a vertex in $C(X)$, and Y be a dominating set $X \setminus \{v\}$. Then, $|C(Y)|$ is at least $|(N'_v(v) \cap X) \setminus Del'(X,v)|$.*

▶ **Lemma 20.** *Let X be a dominating set, v be a vertex in $C(X)$, and Y be a dominating set $X \setminus \{v\}$. Then, $|C(Y)|$ is at least $\sum_{u \in N'_v(v) \setminus X} (d'_v(u) - 1)$.*

▶ **Lemma 21.** *Let X be a dominating set, v be a vertex in $C(X)$, and Y be a dominating set $X \setminus \{v\}$. Then, $|C(Y)|$ is at least $\sum_{u \in Del'(X,v) \setminus \{v\}} (d'_v(u) - 1)$.*

▶ **Lemma 22.** *Let X be a dominating set v be a vertex in $C(X)$, and Y be a dominating set $X \setminus \{v\}$. Then, the number of children and grandchildren of Y is at least $\sum_{u \in G_v(X) \setminus (G(Y) \cup Del'(X,v) \cup N'_v(v))} (d'_v(u) - 1)$.*

Note that for any pair of candidate vertices v and v', $X \setminus \{v\}$ and $X \setminus \{v'\}$ do not share their descendants. Thus, from Lemma **??**, Lemma **??**, Lemma **??**, and Lemma **??**, we can obtain the following lemma:

▶ **Lemma 23.** *Let X be a dominating set. Then, the sum of the number of X's children, grandchildren, and great-grandchildren is bounded by the following order:*

$$\Omega\left(|C(X)| + \sum_{v \in C(X)} \left(\sum_{u \in Del'(X,v)} d'_v(u) + \sum_{u \in G_v(X) \setminus G(Y)} d'_v(u) + \sum_{u \in N'_v(v) \setminus X} d'_v(u) \right) \right). \quad (2)$$

From Lemma **??**, Lemma **??**, and Lemma **??**, each iteration outputs a solution in constant amortized time. Hence, by the same discussion of Theorem **??**, we can obtain the following theorem.

▶ **Theorem 24.** *For an input graph with at least nine,* `EDS-G` *enumerates all dominating sets in $O(1)$ time per solution by using $O(n+m)$ space.*

6 Conclusion

In this paper, we proposed two enumeration algorithms. `EDS-D` 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, `EDS-G` solves this problem in constant time per solution if an input graph have girth at least nine.

Our future work includes to develop efficient dominating set enumeration algorithms for dense graphs. If a graph is dense, then k is large and G has many dominating sets. For example, in the case of complete graphs, k is equal to $n-1$ and every nonempty subset of V is a dominating set. That is, the number of solutions for a dense graph is much larger than that for a sparse graph. This allows us to spend more time in each recursive call. However, `EDS-D` is not efficient for dense graphs although the number of solutions is large. Moreover, if G is small girth, that is, G is dense then `EDS-G` does not achieve constant amortized time enumeration. Hence, the dominating set enumeration problem for dense graphs is interesting.

A Proofs for Section ?? (Efficient Enumeration for Bounded Degenerate Graphs)

Proof for Lemma ??

Proof. Let $X_{\bar{C}} = X \setminus C(X)$. Suppose that $u \in Y$. From the definition, $D_u(X) = N(u)^{<u} \cap X_{\bar{C}}$. From the distributive property,

$$\begin{aligned} L &= D_u(X) \cup (N(u)^{<u} \cap (Del_1(X,v) \cup Del_2(X,v))) \cup (N(u)^{<u} \cap (Del_3(X,v) \setminus \{v\})) \\ &= N(u)^{<u} \cap (X_{\bar{C}} \cup (Del(X,v) \setminus \{v\})) \\ &= N(u)^{<u} \cap (Y \setminus C(Y)) \end{aligned}$$

Since $X_{\bar{C}} \cup (Del(X,v) \setminus \{v\}) = Y \setminus C(Y)$. Suppose that $u \in V \setminus X$. From the parent-child relation, $pv(Y) < u$ holds. Since $Del(X,v) \subseteq V^{<u}$, $N(u)^{<u} \cap (Del_1(X,v) \cup Del_2(X,v)) = N(u) \cap (Del_1(X,v) \cup Del_2(X,v))$, and $N(u)^{<u} \cap (Del_3(X,v) \setminus \{v\}) = N(u) \cap (Del_3(X,v) \setminus \{v\})$. From the definition, $D_u(X) = N(u) \cap X_{\bar{C}}$,

$$\begin{aligned} L &= D_u(X) \cup (N(u)^{<u} \cap (Del_1(X,v) \cup Del_2(X,v))) \cup (N(u)^{<u} \cap (Del_3(X,v) \setminus \{v\})) \\ &= (N(u) \cap X_{\bar{C}}) \cup (N(u) \cap (Del_1(X,v) \cup Del_2(X,v))) \cup (N(u) \cap (Del_3(X,v) \setminus \{v\})) \\ &= N(u) \cap (X_{\bar{C}} \cup (Del_1(X,v) \cup Del_2(X,v)) \cup (Del_3(X,v) \setminus \{v\})) \\ &= N(u) \cap (X_{\bar{C}} \cup (Del(X,v) \setminus \{v\})) \\ &= N(u) \cap (Y \setminus C(Y)) \end{aligned}$$

Hence, the statement holds. ◀

Proof for Lemma ??

Proof. Since $Del_1(X,v) \cup Del_2(X,v) \subseteq V^{<v}$ and $Del_3(X,v) \cap V^{<v} = \emptyset$, $N(v)^{<v} \cap (Del_1(X,v) \cup Del_2(X,v)) = N(v)^{<v} \cap Del(X,v)$. By the same discussion as Lemma **??**, $L = D_v(X) \cup (N(v)^{<v} \cap Del(X,v)) = N(v)^{<v} \cap (Y \setminus C(Y))$. Since $Y = X \setminus \{v\}$, $N(v)^{v<} \cap Y = N(v)^{v<} \cap X$. Moreover, since $X^{<v} = Y^{<v}$ and $C(Y)^{v<} = \emptyset$, $N(v)^{v<} \cap (Y \setminus C(Y)) = N(v)^{v<} \cap X$. Since $L = (N(v)^{v<} \cup N(v)^{<v}) \cap (Y \setminus C(Y)) = D_v(Y)$, the statement holds. ◀

B Proofs for Section ?? (Efficient Enumeration for Graphs with Girth at Least Nine)

▶ **Lemma 25.** *Let X be a dominating set, v be a vertex in $C(X)$, and u be a vertex in G. Then, $u \in Del_1(X,v)$ if and only if $N'_v[u] \cap X = \{u,v\}$ and $f_v(u,X) =$* `False`*.*

Proof. The only if part is obvious since $u, v \in C(X)^{\leq v}$ and $N[u] \cap X = \{u,v\}$. We next prove the if part. Since $f_v(u,x) =$ `False`, $N[u] \cap (X \setminus C(X)^{\leq v}) = \emptyset$. Moreover, since $N'_v[u] \subseteq C(X)^{\leq v}$, $N[u] \cap X = N'_v[u] \cup (N[u] \cap (X \setminus C(X)^{v<})) = \{u,v\}$. Hence, the statement holds. ◀

▶ **Lemma 26.** *Let X be a dominating set, v be a vertex in $C(X)$, and u be a vertex in G. Then, $u \in Del_2(X,v)$ if and only if there is a vertex w in $G_v(X)$ such that $N'_v[w] \cap X = \{u,v\}$.*

Proof. The only if part is obvious since $u, v \in C(X)^{\leq v}$ and there is a vertex w such that $N[w] \cap X = \{u,v\}$. We next show the if part. Since $w \in G_v(X)$, $w \in C(X)^{\leq v}$ or $w \notin X \cup N[X \setminus C(X)^{\leq v}]$. Moreover, since $N'[w] = \{u,v\}$, $w \notin X$, that is, $w \notin C(X)$.

Hence, $w \notin N[X \setminus C(X)^{\leq v}]$. Therefore, $N[w] \cap X = N'_v[w] \cup (N[w] \cap (X \setminus C(X)^{v<})) = \{u, v\}$ and the statement holds. ◀

Proof for Lemma ??

Proof. Since $Del_3(X, v) \cap C(X \setminus \{v\}) = \emptyset$, $C(X \setminus \{v\}) \subseteq C(X)^{\leq v}$. Thus, we do not need to remove vertices in $Del_3(X, v)$ from $C(X)^{\leq v}$. From Lemma **??**, for each vertex $u \in N'_v(v)$, we can check whether $u \in Del_1(X, v)$ or not in constant time by confirming that $f_v(u, X) =$ `False` and $|N'_v(u)| = 2$. Moreover, from Lemma **??**, for each vertex $w \in N'_v(v)$, we can compute $Del_2(X, v)$ by listing vertices in $u \in C(X)^{\leq v}$ such that $N'[w] \cap X = \{u, v\}$ or not. Note that since any vertex in $X^{<v}$ belongs to X, $N'[w] \cap X = \{u, v\}$ if $f_v(w, X) =$ `False`, $|N'[w]| = 2$, and u and v are adjacent to w. Hence, the statement holds. ◀

Proof for Lemma ??

Proof. From the definition, $V(G(Y)) \subseteq V(G_v(X))$. Let us denote by u a vertex in $G_v(X)$ but not in $G(Y)$ such that $u \neq v$. This implies that (A) u is dominated by some vertex in $Y \setminus C(Y)$ and (B) $u \notin C(Y)$. Thus, for any vertex $u' \notin N'_v[Del'(X, v) \setminus \{v\}]$, $u' \in G_v(X)$ if and only if $u' \in G(Y)$. Hence, we can find such vertex u by checking whether for each vertex $w \in N'_v[Del'(X, v)]$, w satisfies (A) and (B). Before checking, we first update the value of f. This can be done by checking all the vertices in $N'_v[Del'(X, v)]$ and in $O(1)$ time per vertex. Hence, this update needs $O\left(\sum_{w \in Del'(X,v)} d'_v(w)\right)$ time. If w satisfies these conditions, that is, $f_v(w, X) =$ `False`, $f(w, Y) =$ `True`, and (B), then we remove w and edges that are incident to w from $G_v(X)$. This needs $O\left(\sum_{w \in G_v(X) \setminus G(Y)} d'_v(w)\right)$ total time for removing vertices. Thus, the statement holds. ◀

Proof for Lemma ??

Proof. Let u be the largest vertex in $C(X)^{<v}$ and w be a vertex in $G_v(X) \cap Del'(X, v)$. If $w \in Del_1(X, v)$, then $d'_u(w) = 0$ since $w \in N'_v(v)$. Otherwise, $w \in Del_2(X, v)$, then $d'_u(w) = 0$ since a vertex x such that $N'_v[x] = \{w, v\}$ is removed from $G_v(X)$. Hence, $Pen_v(X) \cap Pen_u(X) = \emptyset$. Moreover, for each $v \in C(X)$, $Pen_v(X)$ is a subset of $C(X)$. Hence, the union of $Pen_v(X)$ is a subset of $C(X)$ for each $v \in C(X)$. ◀

▶ **Lemma 27.** *Let X be a dominating set, v be a vertex in $C(X)$, and u be a vertex in $G_v(X)$. Then, $\left|N'_v[u] \cap C(X)^{\leq v}\right| \geq 2$ if $u \notin C(X)$. Otherwise, $\left|N'_v[u] \cap C(X)^{\leq v}\right| \geq 1$.*

Proof. If $u \in C(X)$, then $u \in N'[u] \cap C(X)$. We assume that $u \notin C(X)$. Thus, $N'[u] \cap (X \setminus C(X)) = \emptyset$ from the definition of $G(X)$. If $|N'[u] \cap C(X)| = 0$, then u is not dominated by any vertex. This contradicts X is dominating set. If $|N'[u] \cap C(X)| = 1$, then u is dominated only by the neighbor w of u in $C(X)$. This contradicts $w \in C(X)$. Hence, $|N[v] \cap C(X)| \geq 2$ if $v \notin C(X)$. ◀

Proof for Lemma ??

Proof. Let u be a vertex in $(N'_v(v) \cap X) \setminus Del'(X, v)$. If $u \in C(X)$, then u is also a candidate vertex in $C(Y)$ since $u \notin Del'(X, v)$. Suppose that $u \notin C(X)$. Since $u \in G_v(X)$, u is dominated by only candidate vertices of X. However, since $u \in X$, u dominates it self and thus, this contradicts. Hence, the statement holds. ◀

Proof for Lemma ??

Proof. Let u be a vertex in $N'_v(v) \setminus X$. That is, $u \notin C(X)$ and $N'_v(u) \subseteq C(X)$. Thus, from Lemma **??**, there is a vertex $w \in N'_v(u)$ such that $w < v$. We consider the following two cases: (A) If $N'_v(u) = \{v, w\}$, then $w \in Del'(X, v)$. From the assumption, w has at least one neighbor x such that $x \neq u$. If $x \notin C(X)$, then there is a neighbor $y \in C(X)$ such that $y \neq w$. Suppose that $y \in Del'(X, v)$. This implies that there is a cycle with length at most six. This contradicts the girth of G. Hence, $y \notin Del'(X, v)$ and $Y \setminus \{y\}$ is a dominating set. If $x \in C(X)$, then $x \notin Del'(X, v)$ from the definition of $Del'(X, v)$ and the girth of G. Hence, $Y \setminus \{x\}$ is a dominating set. (B) Suppose $N'_v(u)$ has a vertex $z \in C(X)$ such that $z \neq v$ and $z \neq w$. If both z and w are in $Del'(X, v)$, then from the definition of $Del'(X, v)$ and the girth of G, G has a cycle with length at most five. Thus, without loss of generality, we can assume that $z \notin Del'(X, v)$. This allows us to generate a child $Y \setminus \{z\}$ of Y. Since the girth of G is at least nine, all children of Y generated above are mutually distinct. Hence, the statement holds. ◀

Proof for Lemma ??

Proof. Let u be a vertex in $Del'(X, v) \setminus \{v\}$. From the assumption, there is a neighbor w of u in $G(X)$. We consider the following two cases: (A) Suppose that w is in $G(Y)$. Since u is in $Y \setminus C(Y)$, $w \in C(Y)$. Hence, $Y \setminus \{w\}$ is a child of Y. Suppose that for any two distinct vertices x, y in $Del'(X, v) \setminus \{v\}$, they have a common neighbor w' in $G(Y)$. If both x and y are in $Del_2(X, v)$, then there exist two vertex z_x, z_y such that $N'_v[z_x] \cap X = \{x, v\}$ and $N'_v[z_y] \cap X = \{y, v\}$, respectively. Therefore, there is a cycle $(v, z_x, x, w', y, z_y, v)$ with length six. As with the above, if x or y in $Del_1(X, v)$, then there exists a cycle with length less than six since $\{x, v\} \in G$ or $\{x, v\} \in G$. This contradicts of the assumption of the girth of G. Hence, any pair vertices in $Del'(X, v)$ has no common neighbors. Thus, in this case, all grandchildren of X are mutually distinct. (B) Suppose that w is not in $G(Y)$. Thus, if $w \in C(X)$, then $w \in Del'(X, v)$. This implies that there is a cycle including w and u whose length is less than six. Hence, w is not in $C(X)$. Then, from Lemma **??**, there is a vertex z in $N'_v(w) \cap C(X)$ such that $z \neq u$. Since $u \in Del'(X, v) \setminus \{v\}$, there is an edge between u and v, or there is a vertex c such that $\{u, c\}$ and $\{v, c\}$ are in $G_v(X)$. Again, if z is in $Del'(X, v)$, then there is a cycle with length less than seven. Thus, z still belongs to $C(Y)$ and $X \setminus \{v, z\}$ is a dominating set. Next, we consider the uniqueness of $X \setminus \{v, z\}$. If there is a vertex w' such that $w' \in N'_v(u)$, $w' \neq w$, w and w' share a common neighbor u' other than u, then (u, w, u', w') is a cycle. Hence, any pair neighbors of u has no common neighbors. As with the above, any two distinct vertices in $Del'(X, v) \setminus \{v\}$ also has no common vertex like z. If there are two distinct vertex $u, u' \in Del'(X, v)$ such that u and u' has a common vertex like z, then there is a cycle with length at most eight even if $u, u' \in Del_2(X, v)$. This contradicts the assumption of the girth, and thus, the statement holds. ◀

Proof for Lemma ??

Proof. Let u be a vertex in $G_v(X) \setminus (G(Y) \cup Del'(X, v) \cup N'_v(v))$. Since $u \notin Del'(X, v)$ and $u \in G_v(X) \setminus G(Y)$, u is not in X. Since $\left|N'_v(u) \cap C(X)^{\leq v}\right|$ is greater than or equal to two from Lemma **??**, there are two distinct vertices w, w' in $N'_v(u)$. We assume that $w, w' \in Del'(X, v)$. From Lemma **??**, the distance between w and v is at most two. Similarly, the distance between w' and v is at most two. Hence, there is a cycle with the length at most six since $w \neq v$ and $w' \neq v$. Thus, without loss of generality, we can assume that $w \notin Del'(X, v)$.

(A) Suppose that $|N'_v(u)| = 2$. If there is a vertex $u' \in G_v(X) \setminus (G(Y) \cup Del'(X, v) \cup N'_v(v))$ such that $u' \neq u$ and $w \in N'(u)$, then as with Lemma **??**, there is a short cycle. Hence, for each vertex such as u, there is a corresponding dominating set $X \setminus \{v, w\}$. (B) Suppose that there is a neighbor $w'' \in N'_v(u) \cap C(X)$. Then, as mentioned in above, there is a dominating set $X \setminus \{v, w, w''\}$. In addition, by the same discussion as Lemma **??**, such generated dominating sets are mutually distinct. (C) Suppose that there is a neighbor $w'' \in N'_v(u) \setminus C(X)$. From Lemma **??**, there are two vertices $z, z' \in N'(w'') \cap C(X)$. Then, $z \notin Del'(X, v)$ or $z' \notin Del'(X, v)$, and thus, we can assume that $z \notin Del'(X, v)$. Therefore, there is a dominating set $X \setminus \{v, w, z\}$. Next, we consider the uniqueness of grandchildren of Y. Moreover, if there is a vertex u' such that $w, y \in N'(u')$ holds, such that $z \in N'(y)$. Then, there is a cycle (u, w, u', y, z, w'') with the length six. Hence, grandchildren of Y are mutually distinct for each $u \in G(X) \setminus G(Y) \setminus Del'(X, v)$. Thus, from (A), (B), and (C), the statement holds. ◀

Proof for Theorem ??

Proof. The correctness of `EDS-G` is shown by Theorem **??**, Lemma **??**, and Lemma **??**. By the same discussion with Theorem **??**, the space complexity of `EDS-G` is $O(n + m)$. We next consider the time complexity of `EDS-G`. From Lemma **??**, Lemma **??**, and Lemma **??**. we can amortize the cost of each recursion by distributing $O(1)$ time cost to the corresponding descendant discussed in the above lemmas. Thus, the amortized time complexity of each recursion becomes $O(1)$. Moreover, each recursion outputs a solution. Hence, `EDS-G` enumerates all solutions in $O(1)$ amortized time per solution. ◀