数理解析研究所講究録 第2088巻 2018年 44-52

Efficient Enumeration Algorithm for Dominating Sets in Bounded Degenerate Graphs

Kazuhiro Kurita[1], Kunihiro Wasa[2], Hiroki Arimura[1], and Takeaki Uno[2]
[1] IST, Hokkaido University, Sapporo, Japan
[2] National Institute of Informatics, Tokyo, Japan

Abstract

Dominating sets are fundamental graph structures. However, enumeration of dominating sets has not received much attention. This study aims to propose an efficient enumeration algorithms for bounded degenerate graphs. The algorithm enumerates all the dominating sets for k-degenerate graphs in $O(k)$ time per solution using $O(n+m)$ space. Since planar graphs have a constant degeneracy, this algorithm can enumerate all such sets for planar graphs in constant time per solution.

1 Introduction

Dominating sets is one of the fundamental graph structure and finding the minimum dominating set problem is NP-hard. Moreover, it is still open that the minimal dominating set enumeration problem can be solved in output-polynomial time or not. Moreover, Kanté *et al.* have been proved if there is an output-polynomial time algorithm for the minimal dominating set enumeration problem, then we can enumerate the hypergraph transversals in output-polynomial time in [8]. Enumeration of hypergrpah transversal is also an interesting open problem. Several algorithms have been developed that can be enumerate the minimal dominating sets with polynomial delays for some graph classes [7–9], and incremental polynomial-time algorithms have also been found [6, 10]. In addition, Kanté *et al.* have proposed a polynomial-delay algorithm for the minimal edge dominating set enumeration problem [10].

In this paper, we consider the relax version problem, i.e. enumeration all dominating sets in an input graph. For this relaxation, the number of solution increase exponentially. Hence, the proposed algorithm may be not fast in practically in spite of the proposed algorithm is amortized polynomial time. When we use this algorithm in the real-world problem, we should use some pruning techniques. Recently, there is some variations of domination problem in vertex-colored graph, tropical and rainbow dominating set problem [2, 4]. Especially, the minimum tropical dominating set is not a minimal dominating set in a graph $G = (V, E)$. Thus, it is important to study non-minimal case problem. Our algorithm is base-line of these expansion problems.

Our main contribution is that we give the enumeration algorithm for dominating sets in $O(k)$ time per solution, where k is degeneracy of an input graph. It is known that some graph classes have constant degeneracy, such as forests, grid graphs, outerplaner graphs, planer graphs, bounded tree width graphs, and H-minor free graphs for some fixed H. The proposed algorithm enumerates all the dominating sets for such graphs in constant amortized time. Moreover, enumeration of minimal dominating sets in bounded degenerate graphs is studied. It can be enumerated in polynomial delay [7].

This study focuses on graph *sparsity* as being a good structural property and, in particular, on *degeneracy*, which are the measures of sparseness. While it is straightforward to enumerate dominating sets with a running time of $O(n)$ time per solution. we aim to develop an algorithm that is strictly faster than this trivial approach. A k-degenerate graph has a *good* vertex ordering called a *degeneracy ordering* [12], as shown in Section 4, and this has been used to develop some efficient enumeration algorithms [3, 5, 13].

2 Preliminaries

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 $N_G[u] = N_G(u) \cup \{u\}$. 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.

Let n and N be the size of the input and number of outputs, respectively. We call enumeration algorithm as an *amortized polynomial-time algorithm* if its time complexity is $O(poly(n)N)$. In addition, we call it a *polynomial-delay algorithm* if the following are

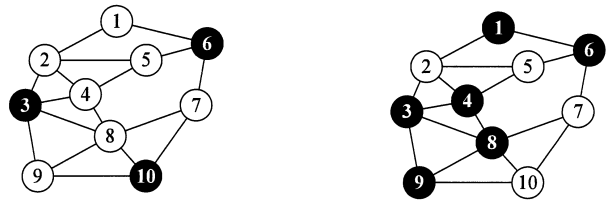

Figure 1: Examples for a dominating set of a graph. Black vertices indicate vertices in dominating sets.

bounded by $O(poly(n))$: the time taken by the algorithm to output the first solution, the interval between outputting two consecutive solutions, and the time taken for the algorithm to terminate after outputting the last solution. Finally, we call it as an *incremental polynomial-time algorithm* if these delays are bounded by $poly(n, N')$, where N' is the number of solutions that have already been generated by the algorithm. 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 an algorithm `EDS-D`. Before we enter into details of `EDS-D`, we first show the basic idea for them, called *reverse search method* that is proposed by Avis and Fukuda [1] 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, 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.

Algorithm 1: The algorithm 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;
```

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 basic algorithm EDS is shown in Algorithm 1. 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. □

From Lemma 2 and Lemma 3, 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.*

Algorithm 2: The algorithm 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 $u \in N(v)^{v<} \cup (N(v) \cap C)$ **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 $\mathcal{D}(Y)$

4 The proposed algorithm

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* [11] 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. Matula and Beck show that the degeneracy and a degeneracy ordering of G can be obtained in $O(n+m)$ time [12]. Our proposed algorithm `EDS-D`, shown in Algorithm 2, 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 $C(Y)$. Due to limitations of space, we omit the proofs of them.

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

From the Lemma 5, 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, $Del(X, v) = Del_1(X, v) \cup Del_2(X, v) \cup Del_3(X, v)$.*

We show an example of dominated list and a maintenance of $C(X)$ in Fig. 2. 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 $D_u(X) = N(u)^{<u} \cap (X \setminus C(X))$. 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 $Del_2(X, v)$. Before showing the efficient dominated lists maintenance, we consider the sets of neighbors $N(u) \cap C(X)$ and $N(u)^{u<} \cap X$ for each vertex $u \in C(X)$ We use these sets for computing dominated lists.

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.*

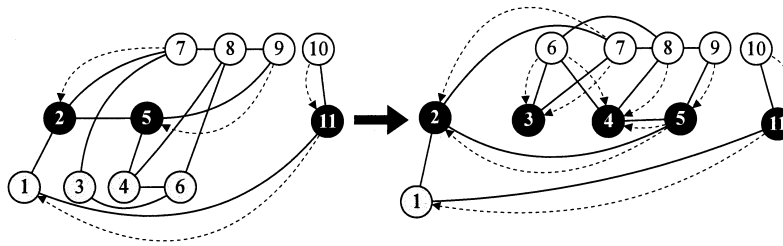

Figure 2: Let X be a dominating set $\{1,2,3,4,5,6\}$. An example of the maintenance of $C(X)$ and $\mathcal{D}(X)$. Each dashed directed edge is stored in $\mathcal{D}(X)$, and each solid edge is an edge in G. A directed edge (u,v) implies $v \in D_u(X)$. The index of each vertex is according to a degeneracy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

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.*

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.*

In Lemma 8 and Lemma 9, 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.

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 10 and Lemma 11, 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.*

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.*

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.*

5 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.

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.

References

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

[2] B. Brešar, M. A. Henning, D. F. Rall, et al. Rainbow domination in graphs. *Taiwanese Journal of Mathematics*, 12(1):213–225, 2008.

[3] A. Conte, R. Grossi, A. Marino, and L. Versari. Sublinear-Space Bounded-Delay Enumeration for Massive Network Analytics: Maximal Cliques. In *ICALP 2016*, volume 55 of *LIPIcs*, pages 148:1–148:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

[4] J.-A. A. d'Auriac, C. Bujtás, H. El Maftouhi, M. Karpinski, Y. Manoussakis, L. Montero, N. Narayanan, L. Rosaz, J. Thapper, and Z. Tuza. Tropical dominating sets in vertex-coloured graphs. In *WALCOM 2016*, pages 17–27. Springer, 2016.

[5] 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.

[6] P. A. Golovach, P. Heggernes, M. M. Kanté, D. Kratsch, and Y. Villanger. Enumerating minimal dominating sets in chordal bipartite graphs. *Discrete Appl. Math.*, 199(30):30–36, 2016.

[7] M. M. Kanté, V. Limouzy, A. Mary, and L. Nourine. Enumeration of minimal dominating sets and variants. *LNCS*, 6914 LNCS:298–309, 2011.

[8] M. M. Kanté, V. Limouzy, A. Mary, and L. Nourine. On the enumeration of minimal dominating sets and related notions. *SIAM J. Discrete Math.*, 28(4):1916–1929, 2014.

[9] 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 *ISAAC 2013*, volume 8283 of *LNCS*, pages 339–349. Springer Berlin Heidelberg, 2013.

[10] M. M. Kanté, V. Limouzy, A. Mary, L. Nourine, and T. Uno. Polynomial delay algorithm for listing minimal edge dominating sets in graphs. In *WADS 2015*, volume 9214 of *LNCS*, pages 446–457. Springer Berlin Heidelberg, 2015.

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

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

[13] K. Wasa, H. Arimura, and T. Uno. Efficient enumeration of induced subtrees in a k-degenerate graph. In *ISAAC 2014*, volume 8889 of *LNCS*, pages 94–102. Springer International Publishing, 2014.

Graduate School of Information Science and Technology
Hokkaido University
Hokkaido 060-0814
JAPAN
E-mail address: k-kurita@ist.hokudai.ac.jp

北海道大学 大学院情報科学研究科 栗田 和宏