An Efficient Algorithm for Enumerating Chordal Bipartite Induced Subgraphs in Graphs

Kazuhiro Kurita[1,a)] Kunihiro Wasa[2] Takeaki Uno[2] Hiroki Arimura[1]

Abstract: In this paper, we propose a characterization of chordal bipartite graphs and an efficient enumeration algorithm for chordal bipartite induced subgraphs. A chordal bipartite graph is a bipartite graph without induced cycles with length six or more. It is known that chordal bipartite graphs have several characterizations. One of them is as follows: A bipartite graph B is chordal bipartite if and only if a hypergraph corresponding to B is β-acyclic. By using the characterization of β-acyclic hypergraphs, we show that a graph is chordal bipartite if and only if it has a special vertex elimination ordering. We call this vertex ordering chordal bipartite elimination ordering (CBEO). Moreover, we propose an algorithm ECB which enumerates all chordal bipartite induced subgraphs in $O(kt\Delta^2)$ time per solution, where, k is the degeneracy, t is the maximum size of $K_{t,t}$ as a subgraph, and Δ is the degree.

Keywords: Output-sensitive enumeration, Chordal bipartite graph, Elimination ordering, Degeneracy, Biclique, Reverse search.

1. Intorduction

The chordality of graphs has been studied well. A graph G is *chordal* if any cycle with length four or more in G has a chord. *A chord of a cycle* is an edge which connects two vertices in a cycle and this edge is not included in a cycle. If G is a chordal, many NP-complete problems can be solved in polynomial time [10], e.g. minimum coloring, maximum clique, minimum clique cover, and maximum independent set problem. Moreover, chordality of a bipartite graph has been also studied. A chordal bipartite graph is a bipartite graph without any induced cycles with length six or more. There are many characterizations of chordal bipartite graphs [4, 11, 17]. In addition, the graph chordality is related to the hypergraph acyclicity [3, 4]. In particular, Ausiello *et. al.* show that a hypergraph is β-acyclic if and only if its bipartite incident graph is $(6, 1)$-chordal. We call a graph G is (a, b)-chordal if any cycle C which has the length a or more has at least b chords. A bipartite graph is $(6, 1)$-chordal if and only if a graph is chordal bipartite. In other words, a hypergraph is β-acyclic if and only if its bipartite incident graph is chordal bipartite.

In this paper, we address the chordal bipartite induced subgraph enumeration problem. We propose amortized $O(kt\Delta^2)$ time algorithm for the problem. To evaluate the efficiency of enumeration algorithms, we often measure in terms of the size of input and the number of output. An enumeration algorithm is *polynomial delay* if the maximum interval between two consecutive solutions is polynomial. Moreover, an enumeration algorithm is *amortized polynomial* if the total running time is $O(Mpoly(N))$ time, where M is the number of solutions and N is the input size. In enumeration area, there are some polynomial delay algorithms for chordal subgraphs and acyclic subhypergraphs enumeration [7, 13, 18, 20, 22]. Especially, Wasa *et. al.* proposed an enumeration algorithm for chordal bipartite induced subgraphs in bipartite graphs. In [22], Wasa *et. al.* proposed amortized $O(nk^3)$ time enumeration algorithm. However, Wasa points out that the time complexity of this algorithm is incorrect. Especially, Lemma 8 is incorrect in [22]. Correctly, it enumerates all solutions in amortized $O(nk^2\Delta)$ time. In this paper, we propose chordal bipartite induced subgraph enumeration algorithm ECB for general graphs. ECB enumerates amortized $O(kt\Delta^2)$ time, where k is the degeneracy, t is the size of a maximum biclique $K_{t,t}$ of G, and Δ is the degree of G. This algorithm achieves amortized $O(poly(\Delta))$ time enumeration. Since Δ and t are at most n and k, respectively, our algorithm is more efficient than the result of [22] in general cases.

In ECB, we use a similar technique as enumeration of chordal induced subgraphs. Kiyomi and Uno [13] use a special vertex ordering, called the *perfect elimination ordering* $(v_1, \ldots, v_n)$. In this ordering, any vertex v_i is simplicial in $G[V_{i\leq}]$, where $V_{i\leq} = \{v_j \in V \mid i \leq j\}$. A vertex is called *simplicial* if an induced subgraph of neighbors becomes a clique. Kiyomi and Uno developed a constant delay enumeration algorithm for chordal induced subgraphs [13] by using this ordering. Likewise a perfect elimination ordering of chordal graph, β-acyclic hypergraphs has a vertex elimination ordering [8]. We show that a graph is chordal bipartite if and only if a graph has a special vertex elimination ordering. We call this vertex ordering *a chordal bipartite elimination ordering* (CBEO). To define CBEO, we use *a weak-simplicial vertex* [17].

Main results: We show that a graph G is chordal bipartite if and only if G has a vertex elimination ordering CBEO. To define

[1] Hokkaido University, Sapporo, Hokkaido, Japan
[2] National Institute of Informatics, Hitotsubashi, Tokyo, Japan
[a)] k-kurita@ist.hokudai.ac.jp

CBEO, we use the notion of a weak-simplicial vertex. In this paper, we proved that any chordal bipartite graph has at least two weak-simplicial vertices. Hence, by removing a weak-simplicial vertex, we can eliminate all vertices in a graph. In addition, we show a new characterization of a weak-simplicial vertex. A vertex v is weak-simplicial if and only if an induced subgraph $G[\bigcup_{u\in N(v)} N[u]]$ is bipartite chain graph [21].

Using CBEO, we propose an enumeration algorithm ECB. This algorithm enumerates chordal bipartite induced subgraphs in amortized $O(kt\Delta^2)$ time, where k is the degeneracy of a graph, t is the maximum size of $K_{t,t}$ as a subgraph, and Δ is the degree of G. Note that t is bounded by k. Hence, ECB enumerates chordal bipartite induced subgraphs in constant amortized time for constant degree graphs. When ECB generates one solution, ECB checks only a local structure of an input graph, so it is efficient for sparse graphs. In enumeration algorithm area, there are efficient algorithms for sparse graphs [6, 9, 12, 14, 15, 18, 19]. Especially, *the degeneracy* of graphs is used for constructing efficient enumeration algorithms. In our proposed algorithm, we also use the degeneracy of graphs.

2. Preliminaries

2.1 Hypergraphs

Let $\mathcal{H} = (V, \mathcal{E})$ be a *hypergraph*. V is a set of vertices and $\mathcal{E}$ is a set of subsets of V. We call an element of $\mathcal{E}$ a *hyperedge*. $\mathcal{H}(v)$ is the set of edges $\{e \in \mathcal{H} \mid v \in e\}$. A sequence of edges $C = (e_1, \ldots, e_k)$ is a *berge cycle* if there exists k distinct vertices $v_1, \ldots, v_k$ such that $v_k \in e_1 \cap e_k$ and $v_i \in e_i \cap e_{i+1}$ for each $1 \le i < k$. A berge cycle $C = (e_1, \ldots, e_k)$ is a *pure cycle* if $k \ge 3$ and $e_i \cap e_j \neq \emptyset$ hold for any distinct i and j, where i and j satisfy one of the following three conditions: (I) $|i - j| = 1$, (II) $i = 1$ and $j = k$, or (III) $i = k$ and $j = 1$. A cycle $C = (e_1, \ldots, e_k)$ is a *β cycle* if the sequence of $(e'_1, \ldots, e'_k)$ is a pure cycle, where $e'_i = e_i \setminus \bigcap_{1\le j\le k} e_j$. We call a hypergraph $\mathcal{H}$ *β-acyclic* if $\mathcal{H}$ has no β cycles. We call a vertex v a *β leaf (or nest point)* if $e \subseteq f$ or $e \supseteq f$ hold for any pair of edges $e, f \in \mathcal{H}(v)$. A bipartite graph $\mathcal{I}(\mathcal{H}) = (X, Y, E)$ is a *incidence graph* of a hypergraph $\mathcal{H} = (V, \mathcal{E})$ if $X = V$, $Y = \mathcal{E}$, and E includes an edge $\{v, e\}$ if $v \in e$, where $v \in V$ and $e \in \mathcal{E}$.

2.2 Graphs

Let $G = (V, E)$ be a simple graph, that is there is no self loops and multiple edges. $u, v \in V$ are *adjacent* if there is an edge $\{u, v\} \in E$. The sequence of vertices $\pi = (v_1, \ldots, v_k)$ is a *path* if v_i and v_{i+1} are adjacent for each $1 \le i \le k - 1$. If $v_1 = v_k$ holds in a path $C = (v_1, \ldots, v_k)$, we call C a *cycle*. *The distance* $dist(u, v)$ between u and v is the length of a shortest path between u and v. We call a graph $H = (U, F)$ a *subgraph* of $G = (V, E)$ if $U \subseteq V$ and $F \subseteq E$ hold. A subgraph $H = (U, F)$ is an *induced subgraph* of G if $F = \{\{u, v\} \in E \mid u, v \in U\}$ hold. In addition, we denote an induced subgraph as $G[U]$. *The neighbor of* v is the set of vertices $\{u \in V \mid \{u, v\} \in E\}$ and denoted by $N_G(v)$. In addition, we denote $N(v) \cap X$ as $N_X(v)$, where X is a subset of V. If there is no confusion, we denote $N_G(v)$ as $N(v)$. The set of vertices $N[v] = N(v) \cup \{v\}$ is called *the closed neighbor*. We define *the neighbor with distance k* and *the neighbor with distance at most k* as $N^k(v) = \{u \in V \mid dist(u, v) = k\}$ and $N^{\le k}(v) = \bigcup_{1\le i\le k} N(v)^k$, respectively. We say v is *incident* to an edge $e = \{u, v\}$ and e is the *incident edge* of u and v. For a vertex set X, we define the set of neighbors $N(X) = \bigcup_{v\in X} N(v) \setminus X$. The *degree of* v $d(v)$ is the size of $N(v)$. The degree of a graph G is the maximum size of $d(v)$ in V. Let U be a subset of V. For vertices $u, v \in V$, u and v are *comparable* if $N(v) \subseteq N(u)$ or $N(v) \supseteq N(u)$ hold. Otherwise, u and v are *incomparable*.

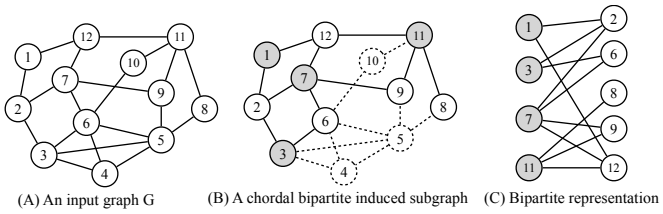


Fig. 1 (A) is an input graph G and (B) is one of the solutions $B = (X, Y, E)$. (C) is a graph drawn by dividing X and Y.

Let $B = (X, Y, E)$ be a bipartite graph. We call B is *a chordal bipartite* graph if there is no induced cycles with length four or more. Moreover, B is *biclique* if any pair of vertices $x \in X$ and $y \in Y$ are adjacent. We denote a biclique as $K_{a,b}$ if $|X| = a$ and $|Y| = b$. In this paper, we consider only the case $a = b$ and the size of a biclique $K_{t,t}$ is t.

Finally, we define our problem, chordal bipartite induced subgraph enumeration problem.

Problem 1 (Chordal bipartite induced subgraph enumeration problem)**.** *Output all chordal induced subgraphs in an input graph G without duplication.*

In Fig. 1, we show an input graph G and one of the solutions. In this paper, we propose an efficient enumeration algorithm ECB to solve chordal bipartite enumeration problem in amortized $O(tk\Delta^2)$ time.

3. A Characterization of Chordal Bipartite Graph

In this section, we propose a characterization of chordal bipartite graphs. In addition, we show that a vertex v is weak-simplicial in a bipartite graph B if and only if $B[N^{\le 2}(v)]$ is bipartite chain. To show the our characterization, we use the following two theorems.

Theorem 1. *(Theorem 1 of [1]) $\mathcal{I}(\mathcal{H})$ is chordal bipartite if and only if $\mathcal{H}$ is β-acyclic.*

From Theorem 1, an incidence graph $\mathcal{I}(\mathcal{H})$ is chordal bipartite if and only if $\mathcal{H}$ is β-acyclic. Next, we consider a property of β-acyclic hypergraphs. Brault showed that β-acyclic hypergraphs have at least two β leaves which are not adjacent [5].

Theorem 2. *(Theorem 3.9 of [5]) A β-acyclic hypergraph $\mathcal{H}$ with at least two vertices has two β leaves that are not neighbors in $\mathcal{H} \setminus \{V(\mathcal{H})\}$.*

Even if $\mathcal{H}$ has multiple edges, Theorem 2 holds since the set inclusion relation by edges is not changed.

A vertex v is *weak-simplicial* [17] if $N(v)$ is an independent set and any pair of neighbors of v are comparable. We show that the relation of a β leaf and a weak-simplicial vertex.

Let $\mathcal{V}$ be a set of vertex subsets. $\mathcal{V}$ is *totally ordered* if for any pair $X, Y \in \mathcal{V}$ of vertex subsets, either $X \subseteq Y$ or $X \supseteq Y$. Similarly,

a vertex v is *totally ordered* if any pair u, v of neighbors of v are comparable.

Lemma 3. *Let $v_{\mathcal{H}}$ be a vertex in $V(\mathcal{H})$ and $v_{\mathcal{I}(\mathcal{H})}$ be the vertex in $V(\mathcal{I}(\mathcal{H}))$ corresponding to $v_{\mathcal{H}}$. Then, $v_{\mathcal{H}}$ is a β-leaf if and only if $v_{\mathcal{I}(\mathcal{H})}$ is a weak-simplicial vertex in $\mathcal{I}(\mathcal{H})$.*

Proof. We assume that $v_{\mathcal{H}}$ is a β-leaf in $\mathcal{H}$. From the definition of a β-leaf, $v_{\mathcal{I}(\mathcal{H})}$ is also totally ordered. In addition, neighbors of $v_{\mathcal{I}(\mathcal{H})}$ form an independent set. Thus, $v_{\mathcal{I}(\mathcal{H})}$ is a weak-simplicial vertex.

We next assume that $v_{\mathcal{I}(\mathcal{H})}$ is weak-simplicial. From the definition, $v_{\mathcal{I}(\mathcal{H})}$ is totally ordered. Thus, $\mathcal{H}(v_{\mathcal{H}})$ is totally ordered. Therefore, $v_{\mathcal{H}}$ is a β-leaf in $\mathcal{H}$ and the statement holds. □

Lemma 4. *Let $B = (X, Y, E)$ be a chordal bipartite graph. If there is no vertices v in B such that $N(v) = X$ or $N(v) = Y$, then B has at least two weak-simplicial vertices which are not adjacent.*

Theorem 5. *A bipartite graph B is a chordal bipartite if and only if B has an elimination ordering $(v_1, \ldots, v_n)$ such that v_i is a weak-simplicial vertex in $B[V_{i\leq}]$ for any v_i.*

Proof. From Lemma 4, the only if part holds. We consider the contraposition of the if part. Since B is not chordal bipartite, B has an induced cycle C with length six or more. Since a vertex in C is not weak-simplicial, we cannot eliminate all vertices from B and the statement holds. □

Next, we show a characterization of a weak-simplicial vertex. A bipartite graph B is *bipartite chain* if for any $W \in \{X, Y\}$, any pair of vertices in W is comparable.

Lemma 6. *Let $B = (X, Y, E)$ be a chordal bipartite graph and v be a vertex in B. Then, v is weak-simplicial if and only if an induced subgraph $B[N^{\leq 2}[v]]$ is bipartite chain.*

Proof. We assume that $B[N^{\leq 2}[v]$ is bipartite chain. From the definition, any pair of vertices in $N(v)$ is comparable. Hence, v is weak-simplicial.

We next prove the other direction. We assume that v is weak-simplicial. Let x and y be vertices in $N^2(v)$. If x and y are incomparable, then there are two vertices $z \in N(x) \setminus N(y)$ and $z' \in N(y) \setminus N(x)$. Note that z and z' are neighbors of v. This contradicts that any pair of vertices in $N(v)$ is comparable. Hence, $x, y \in N^2(v)$ are comparable and $B[N^{\leq 2}[v]]$ is bipartite chain. □

To prove the time complexity of ECB in Sect. 4, we give the two upper bounds with respect to the number of vertices and edges in bipartite chain graph. Note that t is the maximum size of $K_{t,t}$ in G as a subgraph.

Lemma 7. *Let B be a bipartite chain graph and v be a vertex in B. Then, $\left|N^2(v)\right|$ is at most Δ.*

Proof. Since B is bipartite chain, there is a maximum vertex u in $N(v)$ with respect to inclusion of neighbors. Hence, for any vertex $w \in N(v) \setminus \{u\}$, $N(w) \subseteq N(u)$ holds. Since $N^2(v) = \bigcup_{w \in N(v)} N(w)$, $N^2(v)$ is equal to $N(u)$. Hence, the statement holds. □

Lemma 8. *Let $B = (X, Y, E)$ be a bipartite chain graph. Then, the number of edges in B is $O(t\Delta)$, where t is the maximum size of a biclique in B.*

Algorithm 1: ECB enumerates all chordal bipartite induced subgraphs in amortized polynomial time.

```
1 Procedure ECB(G)                    // G = (V, E): an input graph
2 |  RecECB((∅, V, G));
3 Procedure RecECB(X, C(X), G)
4 |  Output X;
5 |  for v ∈ C(X) do
6 |  |  if P(X ∪ {v}) = X then RecECB(X ∪ {v}, C(X ∪ {v}), G) ;
```

Proof. Let v be a maximum vertex in X with respect to inclusion of neighbors. If $d(v) \leq t$, then the statement holds since the size of $N^2(v)$ is at most Δ from Lemma 7.

We then assume that $d(v) > t$. We consider the number of edges in $G[N(v) \cup N^2(v)]$. Let $(u_1, \ldots, u_{d(v)})$ be a sequence of vertices in $N(v)$ such that $N(u_i) \subseteq N(u_{i+1})$ for $1 \leq i < d(v)$. For each $d(v) - t + 1 \leq i \leq d(v)$, since $|N(u_i)|$ is at most Δ, the sum of $|N(u_i)|$ is at most $O(t\Delta)$. We next consider the case for $1 \leq i \leq d(v) - t$. Since $N(u_i)$ is a subset of $N(u_j)$ for any $i < j$, $|N(u_i)|$ is at most t. If $|N(u_i)|$ is greater than t, then B has a biclique $K_{t+1,t+1}$. Hence, the number of edges in B is $O(t\Delta)$ and the statement holds. □

4. Enumeration of Chordal Bipartite Induced Subgraphs

In this section, we propose an amortized $O(kt\Delta^2)$ time enumeration algorithm ECB. ECB is based on reverse search [2]. ECB enumerates all solutions by traversing on a tree structure $\mathcal{F}(G) = (\mathcal{S}(G), \mathcal{E}(G))$, called the *family tree*, where $\mathcal{S}(G)$ is a set of solutions in an input graph G. Note that $\mathcal{F}(G)$ is directed. To define $\mathcal{F}(G)$, Let X be a solution. we first define the parent-child relationship on solutions by using Theorem 5. We denote the set of weak-simplicial vertices in $G[X]$ as $WS(X)$. In what follows, we number the vertex index from 1 to n and compare the vertices with their indices. The *parent* of X is defined as $\mathcal{P}(X) = X \setminus \max\{WS(X)\}$ and a solution X is a *child* of Y if $\mathcal{P}(X) = Y$. Let $ch(X)$ be the set of children of X. We define the *parent vertex* $pv(X)$ as $\max\{WS(X)\}$. For any pair of solutions X and Y, $(X, Y) \in \mathcal{E}(G)$ if $Y = \mathcal{P}(X)$. From Theorem 5, any solution reaches an empty set by recursively removing the parent vertex from the solution. Hence, the following lemma holds.

Lemma 9. *The family tree forms a tree.*

Next, we show that ECB enumerates all solutions. For any vertex subset $X \subset V$, We denote $X_{\leq v} = X \cap V_{\leq v}$, where $V_{\leq v} = \{u \in V \mid u \leq v\}$. An *addible weak-simplicial vertex set* is $AWS(X) = \{v \in V \setminus X \mid v \in WS(X \cup \{v\})\}$, that is, any vertex v in $AWS(X)$ of a solution X generates new solution $X \cup \{v\}$. We define a *candidate set* $C(X)$ as follows: $C(X) = AWS_{pv(X)<}(X) \cup (AWS(X) \cap N^{\leq 2}(pv(X)))$. Note that $C(X)$ is a subset of $AWS(X)$. We show that the relation between $ch(X)$ and $C(X)$.

Lemma 10. *Let X and Y be distinct solutions. If Y is a child of X, then $pv(Y) \in C(X)$.*

Proof. Suppose that Y is a child of X. Let $v = pv(Y) = \max\{WS(Y)\}$ and $u = pv(X) = \max\{WS(X)\}$. Note that v belongs to $AWS(X)$. If $u < v$, then $v \in AWS_{pv(X)<}(X)$ and thus $v \in C(X)$. Otherwise, u is not included in $WS(Y)$ since v has the maximum

Algorithm 2: ECB enumerates all chordal bipartite graphs in amortized $O(kt\Delta^2)$ time.

```
1  Procedure RecECB(X, C(X), G)
2      Output X;
3      for v ∈ C(X) do
4          WS ← UpdateWS(X, v, G);
5          if P(X ∪ {v}) = X then
6              AWS ← UpdateAWS(X, v, G);
7              Computes C and L(X ∪ {v}) from AWS and L(X),
                 respetively;
8              RecECB(X ∪ {v}, C, G);
9  Procedure UpdateWS(X, v, G)
10     for u ∈ N_X(v) do
11         if u ∈ WS ∧ there is a vertex w ∈ N_X(u) is incomparable to u.
             then WS ← WS \ {u};
12         for w ∈ N_X(u) ∩ WS do
13             if w and v are incomparable then WS ← WS \ {w};
14     return WS;
15 Procedure UpdateAWS(X, v, G)
16     AWS ← AWS(X);
17     for u ∈ N(v) do
18         if u ∈ AWS then
19             if There is a vertex w ∈ N_X(u) which is incomparable to u.
                 then AWS ← AWS \ {u};
20         else if u ∈ X then
21             for w ∈ N(u) ∩ AWS do
22                 if w and v are incomparable. then
                       AWS ← AWS \ {w};
23     return AWS;
```

index in $WS(Y)$. From the definition of a weak-simplicial vertex, there are two vertices in $N_Y(u)$ which are incomparable in $G[Y]$. Since u is totally ordered in $G[X]$, v must be in $N^{\leq 2}(u)$. Hence, the statement holds. □

In what follows, we call a vertex $v \in C(X)$ *generates a child* if $X \cup \{v\}$ is a child of X. From Lemma 9 and Lemma 10, ECB can do a DFS traversal on $\mathcal{F}(G)$.

Theorem 11. ECB *enumerates all solutions.*

In the remaining of this paper, we will show the time complexity of ECB. There are two bottlenecks of ECB. (1) Some vertices in $C(X)$ does not generate a child and (2) the maintenance of $WS(X)$, $AWS(X)$, and $C(X)$ consumes time. A trivial bound of redundant vertices in $C(X)$ is $O(\Delta^2)$ since only vertices in $(AWS(X) \cap N^{\leq 2}(pv(X))$ may not generate a child. To evaluate the number of such redundant vertices precisely, we use a *degeneracy ordering*. A graph G is *k-degenerate* if any induced subgraph of G has a vertex with degree k or less. The *degeneracy* of a graph is the smallest such number k. Note that $k \leq \Delta$. Matula *et. al.* [16] showed that a k-degenerate graph G has a following vertex ordering: For each vertex v, the number of neighbors smaller than v is at most k. This ordering is called a *degeneracy ordering* G (See Fig. 2). By using a degeneracy ordering, we show that the number of redundant verticdes is at most $k\Delta$. In what follows, we fix a degeneracy ordering and $WS(X)$ and $AWS(X)$ are sorted by a degeneracy ordering.

Lemma 12. *Let v be a vertex in G. Then, $\left|\{u < v \mid u \in N^{\leq 2}(v)\}\right| \leq k\Delta$.*

Proof. We consider $N_{<v}(v)$. Since $|N_{<v}(v)|$ is less than k, the number of neighbors of $N_{<v}(v)$ is at most $k\Delta$.

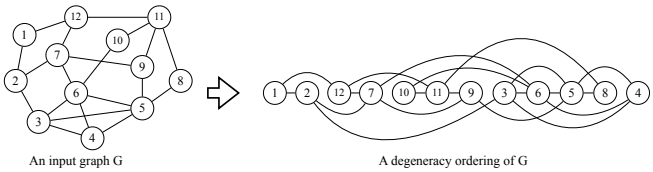


Fig. 2 It is a degeneracy ordering of G. The degeneracy of G is three. In this ordering, $\left|N^{\leq 2}_{<v}(v)\right|$ is at most $k\Delta$ for any vertex v.

We next consider $N_{v<}(v)$. The number of $N_{v<}(v)$ is at most Δ. Since $u \in N_{v<}(v)$ is larger than v, a vertex in $N_{u<}(u)$ is larger than v. Hence, we consider vertices $N_{<u}(u)$. Since $|N_{<u}(u)|$ is at most k for each $u \in N_{v<}(v)$ and v is smaller than u, $\sum_{u \in N_{v<}(v)} |N_{<v}(u)|$ is at most $k\Delta$ and the statement holds. □

Lemma 13. *Let X be a solution. The number of vertices in $C(X)$ which do not generate a child is at most $k\Delta$.*

Proof. Let v be a vertex in $C(X)$. If $pv(X) < v$, then v generates a child. We assume that $v < pv(X)$. Since v is in $C(X)$, $v \in N^{\leq 2}(pv(X))$ and v is smaller than $pv(X)$. From Lemma 12, the number of such vertices is at most $k\Delta$. Hence, the statement holds. □

Next, we show how to update a candidate set. From the definition of $C(X)$, we can compute $C(Y)$ in $O(|C(Y)| + k\Delta)$ time if we have $AWS(Y)$. Moreover, if we have $WS(X \cup \{v\})$, then we can determine whether $X \cup \{v\}$ is a child of X or not in constant time since $WS(X \cup \{v\})$ is sorted. Hence, to obtain children of X, computing $AWS(X)$ and $WS(X)$ dominate the computation time of each iteration.

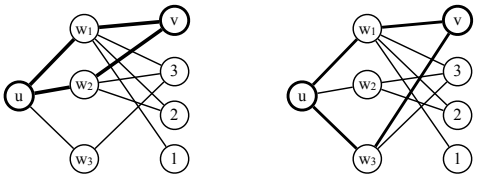
Here, we define some notations. For each $v \in AWS(X) \cup WS(X)$, $L(X, v)$ is a vertex sequence $(u_1, u_2, \ldots, u_k)$ such that each u_i is a smaller neighbor of v in $G[X]$ and $N_X(u_i) \subseteq N_X(u_j)$ holds for any $i < j$. In addition, $\bigcup_{i=1}^{k} u_i$ is equal to $N_X(v)$ and $L(X, u)$ stores each difference between $N(u_i)$ and $N(u_{i+1})$. We call $L(X, v)$ a *neighbor inclusion list* of v. We denote the set of neighbor inclusion lists as $\mathcal{L}(X)$. We define two vertex sets, $Del_{\mathrm{W}}(X, v) = \{u \in N(v)^{\leq 2} \cap WS(X) \mid u \text{ is not weak-simplicial in } G[X \cup \{v\}]\}$ and $Del_{\mathrm{A}}(X, v) = \{u \in N(v)^{\leq 2} \cap AWS(X) \mid u \text{ is not weak-simplicial in } G[X \cup \{u, v\}]\}$, that is, these vertex sets are sets of vertices that are removed from $WS(X)$ and $AWS(X)$ after adding v to X.

Lemma 14. *Let X be a solution, $v = pv(X)$, and u be a vertex in $N_X(v) \cap WS(X)$. Then, $u \in Del_W(X, v)$ if and only if u has a neighbor w in X which is not comparable to v.*

Proof. If part: If u has a neighbor w in X which is incomparable to v, then from the definition, u is not weak-simplicial.

Only if part: Let u be a vertex in $Del_{\mathrm{W}}(X, v)$. Thus, there is a pair of vertices w_1 and w_2 in $N_{X \cup \{v\}}(u)$ which are incomparable. If w_1 or w_2 is equal to v, then u has a neighbor w which is incomparable with v. Hence, we assume that both w_1 and w_2 are not equal to v. Since $G[X \cup \{v\}]$ is bipartite, w_1 and w_2 are not adjacent to v. Hence, w_1 and w_2 are comparable in $X \cup \{v\}$ since after adding v to X, the neighbors of w_1 and w_2 are not changed. However, this contradicts w_1 and w_2 are incomparable. □

Let $L(X, v, i)$ be the i-th vertex u_i in $L(X, v)$ and $L(X, v, i_<)$ be



(A) When a vertex v is added, u is a weak-simplicial

(B) When a vertex v is added, u is not weak-simplicial

Fig. 3 Let X be a set of vertices $\{u, w_1, w_2, w_3, 1, 2, 3\}$. In $G[X]$, $L(X, u) = (w_1, w_2, w_3)$. In case (A), a vertex u is still weak-simplicial. In case (B), however, u is not weak-simplicial since $N(v)$ includes only w_1 and w_3.

the set of vertices from u_1 to u_i in $L(X, v)$.

Lemma 15. *Let X be a solution, $v = pv(X)$, and u be a vertex in $N_X^2(v) \cap WS(X)$. Then, $u \in Del_W(X, v)$ if and only if there exist two integers i and j which satisfy the following three conditions: $i < j$, $L(X, u, i) \in N_{X\cup\{v\}}(v)$, and $L(X, u, j) \notin N_{X\cup\{v\}}(v)$.*

Proof. Let i and j be integers that satisfy the three condition. Since $L(X, u, i) \in N_{X\cup\{v\}}(v)$ and $L(X, u, j) \notin N_{X\cup\{v\}}(v)$ hold, $L(X, u, i)$ and $L(X, u, j)$ are incomparable in $G[X \cup \{v\}]$. Hence, $u \in Del_{\mathrm{W}}(X, v)$.

We prove the other direction. We assume that $u \in Del_{\mathrm{W}}(X, v)$. Hence, There is a pair of neighbors w_1 and w_2 of u such that they are incomparable in $X \cup \{v\}$. Without loss of generality, $N_X(w_1) \subseteq N_X(w_2)$ holds in X since u is in $WS(X)$. Since w_1 and w_2 are incomparable in $X \cup \{v\}$, w_1 is adjacent to v and w_2 is not adjacent to v. Here, let i and j be the positions of w_1 and w_2 in $L(X, u)$, respectively. From the definition of neighbor inclusion list, j is larger than i since $N_X(w_1) \subseteq N_X(w_2)$. Thus, the statement holds. □

Lemma 16. *Let X be a solution, $v = pv(X)$, and u be a vertex in $N(v) \cap AWS(X)$. Then, $u \in Del_A(X, v)$ if and only if u has a neighbor w in $X \cup \{u, v\}$ which is incomparable to v.*

Proof. We assume that u has a neighbor w which is incomparable to v. From the definition of weak-simplicial, u is not a weak-simplicial vertex in $X \cup \{u, v\}$. We prove the other direction. We assume that $u \in Del_{\mathrm{A}}(X, v)$ hold. Since $u \in Del_{\mathrm{A}}(X, v)$, There exists a pair w_1 and w_2 of neighbors of u which are incomparable in $X \cup \{u, v\}$. If w_1 or w_2 is equal to v, then u has a neighbor w which is incomparable to v. We next assume that w_1 and w_2 are distinct from v. If v is adjacent to w_1, w_2, or both of them, then at least one of them is incomparable to v in $X \cup \{v\}$ and the statement holds. Otherwise, w_1 and w_2 are comparable in $X \cup \{u, v\}$ since w_1 and w_2 are comparable in $G[X \cup \{v\}]$. It is contradiction to w_1 and w_2 are incomparable and the statement holds. □

Lemma 17. *Let X be a solution, $v = pv(X)$, and u be a vertex in $N^2(v) \cap AWS(X)$. Then, $u \in Del_A(X, v)$ if and only if there are two integers i and j which satisfies the following conditions: $i < j$, $L(X \cup \{v\}, u, i) \in N_{X\cup\{v\}}(v)$, and $L(X \cup \{v\}, u, j) \notin N_{X\cup\{v\}}(v)$.*

Proof. We assume that there are two integers i and j. Since $L(X \cup \{v\}, u, i)$ is adjacent to v and $L(X \cup \{v\}, u, j)$ is not adjacent to v, $L(X \cup \{v\}, u, i)$ and $L(X \cup \{v\}, u, j)$ are incomparable. Hence, $u \in Del_{\mathrm{A}}(X, v)$. We prove the other direction. We assume that $u \in Del_{\mathrm{A}}(X, v)$. Hence, u has a pair of vertices w_1 and w_2 which are incomparable in $X \cup \{v\}$. Without loss of generality, $N_X(w_1) \subseteq N_X(w_2)$ hold. Since w_1 and w_2 are incomparable, w_1 is adjacent to v. Hence, a pair of integers i and j corresponding to w_1 and w_2 satisfies all conditions and the statement holds. □

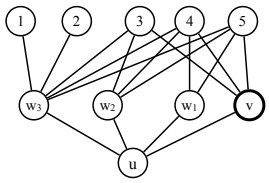


Fig. 4 A comparison between two neighbors of a weak-simplicial vertex. In this case, we compare w_1 with v. It can be done in $O(|N(w_1)|)$ time. Next, we compare w_2 with v. Since $N(w_1)$ is included in $N(w_2)$, this comparison is done in $O(|N(w_2) \setminus N(w_1)|)$ time.

In the following lemmas, we show that $WS(X)$ and $AWS(X)$ can be update if we have $Del_{\mathrm{W}}(X, v)$ and $Del_{\mathrm{A}}(X, v)$.

Lemma 18. *Let X be a solution, Y be a child of X, and $v = pv(Y)$. Then, $WS(Y) = (WS(X) \setminus Del_W(X, v)) \cup \{v\}$.*

Proof. If a vertex is weak-simplicial in Y, then this is also weak-simplicial in any induced subgraph of Y. Thus, $WS(Y) \subseteq (WS(X) \setminus Del_{\mathrm{W}}(X, v)) \cup \{v\}$ holds. We next show $WS(Y) \supseteq (WS(X) \setminus Del_{\mathrm{W}}(X, v)) \cup \{v\}$. It is easily see that $v \in WS(Y)$. Let u be a vertex in $WS(X) \setminus Del_{\mathrm{W}}(X, v)$. Since $u \notin Del_{\mathrm{W}}(X, v)$, u is a weak-simplicial in $X \cup \{v\}$. Hence, $u \in WS(Y)$ and the statement holds. □

Lemma 19. *Let X be a solution, Y be a child of X, and v be a vertex $pv(Y)$. Then, $AWS(Y) = AWS(X) \setminus Del_A(X, v)$.*

Proof. Let u be a vertex in $AWS(Y)$. We prove u is included in $AWS(X) \setminus Del_{\mathrm{A}}(X, v)$. From the definition of an addible candidate set, $u \in AWS(X)$ holds. If $u \in Del_{\mathrm{A}}(X, v)$, then u is not weak-simplicial in $X \cup \{u, v\}$. Since $u \in AWS(Y)$, $u \notin Del_{\mathrm{A}}(X, v)$. We prove the other direction. Let u be a vertex in $AWS(X) \setminus Del_{\mathrm{A}}(X, v)$. From the definition of $AWS(X)$ and $Del_{\mathrm{A}}(X, v)$, u is weak-simplicial in $X \cup \{v, u\}$. Hence, $u \in AWS(Y)$ and the statement holds. □

From Lemma 18 and Lemma 19, we can obtain $WS(X \cup \{v\})$ and $AWS(X \cup \{v\})$ by removing $Del_{\mathrm{W}}(X, v)$ and $Del_{\mathrm{A}}(X, v)$ from $WS(X)$ and $AWS(X)$ and adding v to $WS(X)$, respectively. Note that by just removing redundant vertices, $WS(Y)$ and $AWS(Y)$ can be easily sorted if $WS(X)$ and $AWS(X)$ were already sorted. Next, we consider the time complexity of computing $Del_{\mathrm{W}}(X, v)$ and $Del_{\mathrm{A}}(X, v)$.

Lemma 20. *Let X be a solution and v be a vertex in $C(X)$. Then, we can compute $Del_W(X, v)$ in $O(t\Delta)$ time.*

Proof. We first compute vertices in $WS(X) \cap N_X(v)$ that remain in $WS(X \cup \{v\})$. From Lemma 14, u is included in $Del_{\mathrm{W}}(X, v)$ if and only if u has a neighbor w which is incomparable to v. We consider a pair of vertices w_1 and v, where $w_1 = L(X, u, 1)$. We can decide whether w_1 and v are comparable in $O(|N_X(w_1)|)$ time. Moreover, we can decide whether w_2 and v are comparable in $O(|N_X(w_2) \setminus N_X(w_1)|)$ time since $N_X(w_1) \subseteq N_X(w_2)$ holds (See Fig. 4). Hence, by scanning from the head to the tail of $L(X, u)$, we can check whether $u \in Del_{\mathrm{W}}(X, v)$ or not in $O(\left|N(w_{|N_X(u)|})\right|)$

time in total. Since $O(\sum_{u \in N_X(v)} |N(w_{|N_X(u)|})|) = O(t\Delta)$ holds from 8, we can find $N_X(v) \cap Del_W(X, v)$ in $O(t\Delta)$ total time.

We next compute vertices in $WS(X) \cap N_X^2(v)$ that remain in $WS(X \cup \{v\})$. From Lemma 18 and Lemma 15, $w \notin WS(X \cup \{v\})$ if and only if the ℓ smallest vertices of $L(X, v)$ are contained in $N_X(w) \cap N_X(v)$, where ℓ is the number of neighbors of w in $X \cup \{v\}$. By scanning vertices with distance two from v, and from Lemma 8, this can be done in linear time in the size of $\sum_{w \in N_X^2(v)} |N_X(w) \cap N_X(v)|$, that is, $\sum_{u \in N_X(v)} |N_X(u)| = O(t\Delta)$. □

Lemma 21. *Let X be a solution and v be a vertex in $C(X)$. Then, we can compute $Del_A(X, v)$ in $O(\Delta^2)$ time.*

Proof. In the same fashion as Lemma 20, we can decide $u \in AWS(X \cup \{v\})$ in $O(\Delta)$ time. By applying the above procedure for all vertices distance two from v, we can obtain all vertices in $Del_A(X, v)$ in $O(\Delta^2)$ time since the number of edges can be bounded in $O(\Delta^2)$. □

From Lemma 20 and Lemma 21, we can compute $Del_W(X, v)$ and $Del_A(X, v)$ in $O(t\Delta)$ and $O(\Delta^2)$ time for each $v \in C(X)$, respectively. Next, we show an update procedure for $\mathcal{L}(X \cup \{v\})$ from $\mathcal{L}(X)$.

Lemma 22. *Let X be a solution, Y be a child of X, and $v = pv(Y)$. Then, we can compute $\mathcal{L}(Y)$ in $O(\Delta^2)$ time from $\mathcal{L}(X)$.*

Proof. We first consider an update of $L(Y, u)$ for each vertex in $u \in N(v) \cap (AWS(Y) \cup WS(Y))$. From the definition of $\mathcal{L}(X)$, neighbor inclusion lists that are modified in $\mathcal{L}(X)$ to obtain $\mathcal{L}(Y)$ are contain vertices in $N^{\leq 2}(v)$. Since only the vertex added to X is v, the neighbor inclusion lists of neighbors of v are modified. Let $w_i = L(X, u, i)$ for $1 \leq i \leq |N_X(u)|$. We can decide if $N_Y(w_1) \subseteq N_Y(v)$ or not in $O(|w_1|)$ time. As the same strategy in Lemma 21, we can compare $N_Y(v)$ and w_{i+1} in $O(|N(w_{i+1}) \setminus N(w_i)|)$ time. Hence, we can obtain $L(Y, u)$ in $O(\Delta)$ time from $L(X, u)$.

We next consider update of $L(Y, u)$ for each $u \in N^2(v) \cap (AWS(Y) \cup WS(Y))$. We separate $N_Y(u)$ into two parts S_0 and S_1, such that S_0 is the set of neighbors of v and S_1 is the remains. We sort the vertices in S_0 and S_1 in the same ordering of $L(X, u)$, respectively. By concatenating S_0 onto the end of S_1, we can get $L(Y, u)$ in linear in the size of S_0, that is, $|N(v) \cap N(u)|$. Hence, the statement holds. □

From Lemma 10, Lemma 19, Lemma 18, and Lemma 22, we can enumerate all children in $O(|C(X)|\, t\Delta + |ch(X)|\, \Delta^2)$ time. In the following theorem, we show the amortized time complexity and space usage of ECB.

Theorem 23. ECB *enumerates all solutions in amortized $O(kt\Delta^2)$ time by using $O(n\Delta)$ space, where n is the number of vertices and m is the number of edges in an input graph.*

Proof. In ECB, we use $AWS(X)$, $WS(X)$, and $\mathcal{L}(X)$ as data structures. $WS(X)$ and $AWS(X)$ demand linear space. In addition, a list $L(X, u)$ demands $O(\Delta)$ space since $L(X, u)$ store each difference between $N(u_i)$ and $N(u_{i+1})$, where $u_i = L(X, u, i)$. Hence, the total space usage of ECB is $O(n\Delta)$ space. We consider the amortized time complexity of ECB. From Lemma 11, ECB enumerates all solutions. From Lemma 20, Lemma 21, and Lemma 22, ECB computes all children and updates all data structures in $O(|C(X)|\, t\Delta + |ch(X)|\, \Delta^2)$ time. From Lemma 13, $|C(X)|$ is at most $|ch(X)| + k\Delta$. Hence, we need $O((|ch(X)| + k\Delta)t\Delta + |ch(X)|\, \Delta^2)$ time to generate all children. Note that $O((|ch(X)| + k\Delta)t\Delta + |ch(X)|\, \Delta^2)$ is bounded by $O((|ch(X)| + kt)\Delta^2)$. We consider the total time to enumerate all solution. Since each iteration X needs $O((|ch(X)| + kt)\Delta^2)$ time, the total time is $O(\sum_{X \in \mathcal{S}}(|ch(X)| + kt)\Delta^2)$ time. Since $O(\sum_{X \in \mathcal{S}} |ch(X)|\, \Delta^2)$ is bounded by $O(|\mathcal{S}|\, \Delta^2)$, the total time is $O(|\mathcal{S}|\, kt\Delta^2)$ time. Therefore, ECB enumerates all solution in amortized $O(kt\Delta^2)$ time and the statement holds. □

Corollary 24. ECB *enumerates all solutions in amortized constant time by using $O(n)$ space for constant degree graphs.*

5. Conclusion

In this paper, we propose a vertex ordering CBE0 by relaxing a vertex ordering proposed by Uehara [17]. A bipartite graph B is chordal bipartite if and only if B has CBE0, that is, this vertex ordering characterize chordal bipartite graphs. This ordering comes from hypergraph acyclicity and the relation between β-acyclic hypergraphs and chordal bipartite graphs. In addition, we also show that a vertex v is weak-simplicial if and only if $G[N^{\leq 2}[v]]$ is bipartite chain. By using these facts, we propose an amortized $O(kt\Delta^2)$ time algorithm ECB based on the reverse search [2].

As future work, the following two enumeration problems are interesting: Enumeration of bipartite induced subgraph for dense graphs and enumeration of bipartite subgraph enumeration. For dense graphs, ECB does not achieve an amortized linear time enumeration. If an input graph is biclique, then the time complexity of ECB becomes $O(nm)$ time. Hence, it is still open that there is an amortized linear time enumeration algorithm for chordal bipartite induced subgraph enumeration problem.

For chordal bipartite subgraph enumeration, it is not easy to apply the strategy of ECB for chordal bipartite subgraph enumeration since ECB is based on CBE0. However, in subgraph enumeration problem, we may be allowed to take much time generating all children since the number of chordal bipartite subgraph is larger than the number of induced chordal bipartite subgraph. Hence, an amortized linear time enumeration of chordal bipartite subgraphs is interesting.

References

[1] Ausiello, G., D'Atri, A. and Moscarini, M.: Chordality properties on graphs and minimal conceptual connections in semantic data models, *J. Comput. Syst. Sci.*, Vol. 33, No. 2, pp. 179–202 (1986).
[2] Avis, D. and Fukuda, K.: Reverse search for enumeration, *Discrete Appl. Math.*, Vol. 65, No. 1, pp. 21–46 (1996).
[3] Beeri, C., Fagin, R., Maier, D. and Yannakakis, M.: On the desirability of acyclic database schemes, *J. ACM*, Vol. 30, No. 3, pp. 479–513 (1983).
[4] Brandstadt, A., Spinrad, J. P. et al.: *Graph classes: a survey*, Vol. 3, Siam (1999).
[5] Brault-Baron, J.: Hypergraph acyclicity revisited, *ACM Comput. Surv.*, Vol. 49, No. 3, p. 54 (2016).
[6] Conte, A., Grossi, R., Marino, A. and Versari, L.: Sublinear-Space Bounded-Delay Enumeration for Massive Network Analytics: Maximal Cliques, *Proc. ICALP 2016*, LIPIcs, Vol. 55, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 148:1–148:15 (online), DOI: 10.4230/LIPIcs.ICALP.2016.148 (2016).
[7] Daigo, T. and Hirata, K.: On generating all maximal acyclic subhypergraphs with polynomial delay, *In Proc. SOFSEM 2009*, Springer, pp. 181–192 (2009).

[8] Duris, D.: Some characterizations of γ and β-acyclicity of hypergraphs, *Inf. Process. Lett.*, Vol. 112, No. 16, pp. 617–620 (2012).
[9] Eppstein, D., Löffler, M. and Strash, D.: Listing All Maximal Cliques in Large Sparse Real-World Graphs, *J. Exp. Algorithmics*, Vol. 18, pp. 3.1:3.1–3.1:3.21 (online), DOI: 10.1145/2543629 (2013).
[10] Gavril, F.: Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph, *SIAM J. Comput.*, Vol. 1, No. 2, pp. 180–187 (1972).
[11] Huang, J.: Representation characterizations of chordal bipartite graphs, *J. Comb. Theory, Series B*, Vol. 96, No. 5, pp. 673–683 (2006).
[12] Kanté, M. M., Limouzy, V., Mary, A. and Nourine, L.: Enumeration of Minimal Dominating Sets and Variants, *Proc. FCT 2011*, Springer, pp. 298–309 (2011).
[13] Kiyomi, M. and Uno, T.: Generating chordal graphs included in given graphs, *IEICE Trans. Inf. & Syst.*, Vol. 89, No. 2, pp. 763–770 (2006).
[14] Kurita, K., Wasa, K., Arimura, H. and Uno, T.: Efficient Enumeration of Dominating Sets for Sparse Graphs, *Proc. ISAAC 2018*, pp. 8:1–8:13 (online), DOI: 10.4230/LIPIcs.ISAAC.2018.8 (2018).
[15] Kurita, K., Wasa, K., Uno, T. and Arimura, H.: Efficient enumeration of induced matchings in a graph without cycles with length four, *IEICE Trans. Fundamentals*, Vol. 101, No. 9, pp. 1383–1391 (2018).
[16] Matula, D. W. and Beck, L. L.: Smallest-last ordering and clustering and graph coloring algorithms, *J. ACM*, Vol. 30, No. 3, pp. 417–427 (1983).
[17] Uehara, R.: Linear time algorithms on chordal bipartite and strongly chordal graphs, *Proc. ICALP 2002*, Springer, pp. 993–1004 (2002).
[18] Wasa, K., Arimura, H. and Uno, T.: Efficient Enumeration of Induced Subtrees in a K-Degenerate Graph, *Proc. ISAAC 2014*, LNCS, Vol. 8889, Springer, pp. 94–102 (online), DOI: 10.1007/978-3-319-13075-0_8 (2014).
[19] Wasa, K. and Uno, T.: Efficient Enumeration of Bipartite Subgraphs in Graphs, *Proc. COCOON 2018* (Wang, L. and Zhu, D., eds.), Cham, Springer International Publishing, pp. 454–466 (2018).
[20] Wasa, K., Uno, T., Hirata, K. and Arimura, H.: Polynomial delay and space discovery of connected and acyclic sub-hypergraphs in a hypergraph, *Proc. DS*, Springer, pp. 308–323 (2013).
[21] Yannakakis, M.: The complexity of the partial order dimension problem, *SIAM J. on Algebraic Discrete Methods*, Vol. 3, No. 3, pp. 351–358 (1982).
[22] 和佐州洋, 有村博紀, 宇野毅明 and 平田耕一: 二部グラフ中に含まれる弦二部誘導グラフの列挙, 研究報告アルゴリズム *(AL)*, Vol. 2015, No. 5, pp. 1–8 (2015).