Independent Set under a Change Constraint from an Initial Solution

Yuichi Asahiro[1], Hiroshi Eto[2,*], Kana Korenaga[2,†], Guohui Lin[3], Eiji Miyano[2,‡], and Reo Nonoue[2,§]

[1] Kyushu Sangyo University, Fukuoka, Japan
asahiro@is.kyusan-u.ac.jp

[2] Kyushu Institute of Technology, Iizuka, Japan
{*eto, ‡miyano}@ai.kyutech.ac.jp,
{†korenaga.kana518, §nonoue.reo265}@mail.kyutech.jp

[3] University of Alberta, Edmonton, Canada
guohui@ualberta.ca

Abstract. In this paper, we study a type of incremental optimization variant of the Maximum Independent Set problem (MaxIS), called Bounded-Deletion Maximum Independent Set problem (BD-MaxIS): Given an unweighted graph $G = (V, E)$, an initial feasible solution (i.e., an independent set) $S^0 \subseteq V$, and a non-negative integer k, the objective of BD-MaxIS is to find an independent set $S \subseteq V$ such that $|S^0 \setminus S| \leq k$ and $|S|$ is maximized. The original MaxIS is generally NP-hard, but, it can be solved in polynomial time for perfect graphs (and therefore, comparability, co-comparability, bipartite, chordal, and interval graphs). In this paper, we show that BD-MaxIS is NP-hard even if the input is restricted to bipartite graphs, and hence to comparability graphs. On the other hand, fortunately, BD-MaxIS on co-comparability, interval, convex bipartite, and chordal graphs can be solved in polynomial time. Finally, we study the computational complexity on very similar variants of the Minimum Vertex Cover and the Maximum Clique problems for graph subclasses.

1 Introduction

Background. Motivated by the practice-oriented research on the railroad blocking problem, the following general framework of *incremental optimization* problems with initial solutions was introduced [20]: Let P be an optimization problem with a starting feasible solution S^0, and let $\mathcal{F}$ be the set of all feasible solutions for P. For a new feasible solution $S \in \mathcal{F}$, the increment from S^0 to S is the amount of change given by a function $f(S, S^0) : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$, which we refer to as the increment function. Suppose that k is a given bound on the total amount of change permitted. We call S an incremental solution if it satisfies the inequality $f(S, S^0) \leq k$. The goal is to find an incremental solution S^* that results in the maximum improvement in the objective function value.

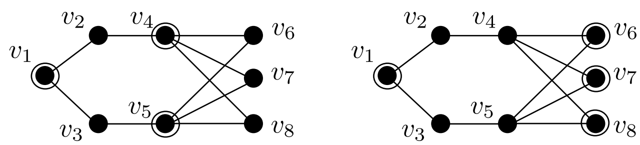

Fig. 1. Given a graph G, an initial solution $\{v_1, v_4, v_5\}$, and $k = 2$ as input (left), an optimal solution is $\{v_1, v_6, v_7, v_8\}$.

In this paper we study a type of incremental optimization of the Maximum Independent Set problem (MaxIS for short). The original MaxIS is one of the most important and most investigated combinatorial optimization problems in theoretical computer science. The input of MaxIS is an unweighted graph $G = (V, E)$, where V and E are the sets of vertices and edges in G, respectively. An *independent set* of G is a subset $S \subseteq V$ of vertices such that for every pair $u, v \in S$, the edge $\{u, v\}$ is not in E. The goal of MaxIS is to find an independent set of maximum cardinality. The problem MaxIS is a well-studied algorithmic problem, and actually it is one of the Karp's 21 fundamental NP-hard problems [14]. Furthermore, it is well known that MaxIS remains NP-hard even for substantial restricted graph classes such as cubic planar graphs [6], triangle-free graphs [19], and graphs with large girth [17]. Fortunately, however, it is also known that the problem can be solved in polynomial time if the input graph is restricted to, for example, graphs with constant treewidth [5] (and therefore, outerplanar, series-parallel, cactus graphs, and so on), perfect graphs [12] (and therefore, chordal [7], comparability [10], co-comparability, bipartite graphs, and so on), circular-arc graphs [8], and many other graph classes.

Our problem and contributions. Throughout this paper, we let S^0 and S denote an initial solution (i.e., an initial independent set) and a solution obtained by our algorithm. We define the increment function as $f(S, S^0) = |S^0 \setminus S|$, the number of vertices in S^0 but not in S, which is the number of deleted vertices from the initial solution S^0. The obtained solution S must satisfy the inequality $|S^0 \setminus S| \leq k$. That is, the number of vertices deleted from the initial solution S^0 is bounded by the given bound k. The function f can be seen as a "change-constraint" function. Now, we can define our problem as follows:

> Bounded-Deletion Maximum Independent Set (BD-MaxIS)
> **Input:** An unweighted graph $G = (V, E)$, an initial feasible solution (i.e., an independent set) $S^0 \subseteq V$, and a non-negative integer k.
> **Goal:** The goal is to find an independent set $S \subseteq V$ such that $|S^0 \setminus S| \leq k$ and $|S|$ is maximized.

See Figure 1 for an example. If a graph G of eight vertices, an initial solution $\{v_1, v_4, v_5\}$, and $k = 2$ are given as input, then $\{v_1, v_6, v_7, v_8\}$ is an optimal solution, which is obtained by deleting two vertices $\{v_4, v_5\}$ and adding three vertices $\{v_6, v_7, v_8\}$. If $k = 1$, then the initial solution $\{v_1, v_4, v_5\}$ is optimal

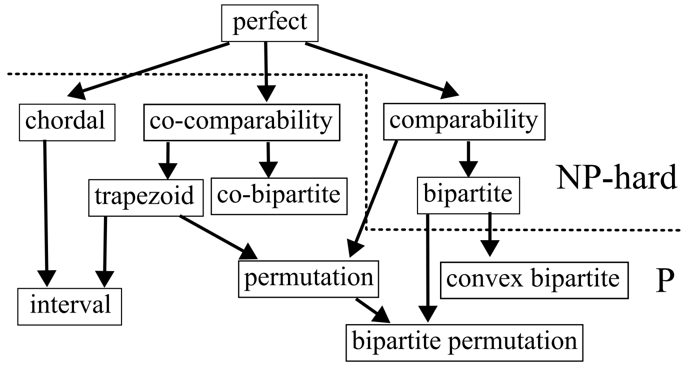

Fig. 2. Computational complexity of BD-MaxIS on graph classes. For example, "perfect $\rightarrow$ comparability" means that the perfect graph class is a superclass of the comparability graph class.

since one vertex-deletion does not make it possible to insert two or more new independent vertices.

One sees that BD-MaxIS is generally NP-hard since if $k \geq |S^0|$, then we can completely change the solution, and thus BD-MaxIS includes the classical MaxIS as a special case (or simply, MaxIS is the case where S^0 is empty and $k = 0$). Hence, our work focuses on the computational complexity of BD-MaxIS on polynomial-time solvable graph classes such as perfect, comparability, co-comparability, bipartite, chordal graphs, and so on.

Our main results are summarized in the following list and Figure 2:

(1) BD-MaxIS is NP-hard even if the input is restricted to bipartite graphs. Since every bipartite graph is comparability and perfect, BD-MaxIS on comparability graphs, or perfect graphs is also NP-hard.
(2) BD-MaxIS can be solved in $O(k|V|^2)$ time for co-comparability graphs. If the input graph is an interval graph, then there is an $O(k|V| + |E|)$-time algorithm for BD-MaxIS.
(3) BD-MaxIS can be solved in $O(k|E|)$ time for convex bipartite graphs.
(4) BD-MaxIS can be solved in $O(k^2(|V| + |E|)^2)$ time for chordal graphs.

Other well-known graph classes including trapezoid, co-bipartite, permutation, and bipartite permutation are also polynomial-time solvable from the results (2), (3), and (4).

2 Preliminaries

Notation. Let $G = (V, E)$ be a simple (without multiple edge or self-loop edge), unweighted, and undirected graph, where V and E are sets of vertices and edges, respectively. We sometimes denote by $V(G)$ and $E(G)$ the vertex and the edge sets of G, respectively. Unless otherwise described, n and m denote the cardinality of V and the cardinality of E, respectively, for $G = (V, E)$. An edge between vertices u and v is denoted by $\{u, v\}$, and in this case vertices u and v are said to be adjacent. The graph $\overline{G}$ denotes the complement graph of G, i.e., $\overline{G} = (V, \overline{E})$, where $\{u, v\} \in \overline{E}$ if and only if $\{u, v\} \notin E$. Let $S \subseteq V$ be a set of vertices of G. Then, the cardinality of S is denoted by $|S|$ and the subgraph of G induced by S is denoted by $G[S]$. The set $N(u) = \{v \in V \mid \{u, v\} \in E\}$ is called the neighborhood of the vertex $u \in V$ in G.

Graph subclasses. A k-coloring of the vertices of a graph $G = (V, E)$ is a mapping $col : V \to \{1, \ldots, k\}$ such that $col(u) \neq col(v)$ whenever $\{u, v\}$ is an edge in G. The chromatic number of G, denoted by $\chi(G)$, is the least number k such that G admits a k-coloring. A *clique* in a graph G is a subset $S \subseteq V$ of vertices such that every two vertices in S are adjacent. The clique number of G, denoted by $\omega(G)$, is the number of vertices in a maximum clique of G. An independent set in a graph is a set of vertices no two of which are adjacent. The independence number of G, denoted by $\alpha(G)$, is the size of a largest independent set in G.

A graph G is called *perfect* if $\chi(H) = \omega(H)$ for every induced subgraph H of G. A graph is called *chordal* if every cycle of length at least four contains a chord, which is an edge that is not part of the cycle but connects two vertices of the cycle. A graph G is called *bipartite* if its chromatic number is at most two. Consider a bipartite graph G with the vertex set $V \cup W$ and its 2-coloring col, where V and W are the disjoint sets of vertices such that $col(V) = 1$ and $col(W) = 2$. The bipartite graph G is *convex* if the vertices in W can be ordered in such a way that, for each $v \in V$, the neighborhood $N(v)$ of v are consecutive in W. The ordering of the vertices in W is said to be *convex*, and G is said to be *convex with respect to* W. A graph G is called *co-bipartite* if its complement graph $\overline{G}$ is bipartite. A graph is called *comparability* if there exists a partial order $<_\sigma$ on its vertices such that two vertices u and v are adjacent in the graph if and only if $u <_\sigma v$ or $v <_\sigma u$. A graph G is called *co-comparability* if its complement graph $\overline{G}$ is a comparability graph. A graph is called *permutation* if it can be represented by a permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ in such a way that two vertices $i < j$ are adjacent if and only if $\pi(i) > \pi(j)$. A graph is called *bipartite permutation* if it is both bipartite and permutation.

3 NP-hardness of BD-MaxIS on bipartite graphs

Given an unweighted graph G, the goal of the Maximum Clique problem (MaxClique) is to find a clique $Q \subseteq V$ of maximum cardinality [14]. Let q-Clique be the decision version of MaxClique, i.e., given a graph G and an integer q, q-Clique is to determine if there is a clique of size q in G:

Theorem 1. *BD-MaxIS is NP-hard even if the input is restricted to bipartite graphs.*

Proof. We show that the NP-complete problem q-Clique is polynomial-time reducible to BD-MaxIS on bipartite graphs. Suppose that the input of q-Clique is

$G^0 = (V^0, E^0)$, where $V^0 = \{v_1^0, \ldots, v_n^0\}$ of n vertices and $E^0 = \{e_1^0, \ldots, e_m^0\}$ of m edges. Then, we construct the following bipartite graph $G = (V_v \cup V_e, E)$ of BD-MaxIS by subdividing every edge in E^0 to two edges:

$$\begin{aligned} V_v &= \{v_1, v_2, \ldots, v_n\}, \\ V_e &= \{e_1, e_2, \ldots, e_m\}, \text{ and} \\ E &= \{\{v_i, e_s\}, \{v_j, e_s\} \mid e_s^0 = \{v_i^0, v_j^0\} \in E^0\}. \end{aligned}$$

That is, the constructed graph G is so-called an *incidence graph* of G^0, and thus G must be bipartite. Then, we set an initial solution $S^0 = V_v$ and an integer $k = q$. This completes the reduction. One sees that each edge in E connects a vertex in V_v with a vertex in V_e. Therefore, $S^0 = V_v$ must be a (feasible) independent set. The reduction can be clearly executed in polynomial time.

For the above construction of G, we show that G contains an independent set S such that $|S^0 \setminus S| \leq k$ and $|S| \geq |V| - k + k(k-1)/2$ if and only if G^0 contains a clique Q^0 such that $|Q^0| \geq q$.

(1) Suppose that G^0 contains a clique Q^0 of size q, and $Q^0 = \{v_1^0, \ldots, v_q^0\}$, without loss of generality. Then, let $R = \{v_1, \ldots, v_q\}$ be the subset of the corresponding q vertices in the initial independent set V_v. Since there must be an edge between every pair of v_i^0 and v_j^0 in Q^0 of G^0, we can find a set, say, A, of $q(q-1)/2$ isolated vertices in V_e by deleting all the vertices in R corresponding to Q^0. Let $S = (S^0 \setminus R) \cup A$. One can see that (i) $S^0 \setminus S = R$, and thus $|S^0 \setminus S| = q = k$, and (ii) $S \setminus S^0 = A$ and $|S \setminus S^0| = q(q-1)/2 = k(k-1)/2$. Namely, $|S| = |V| - k + k(k-1)/2$.

(2) Suppose that the size of a maximum clique in G^0 is at most $q-1$. Let $R = \{v_1, \ldots, v_q\}$ be an arbitrary subset of q vertices in the initial independent set V_v. Then, we consider the corresponding set $R^0 = \{v_1^0, \ldots, v_q^0\}$ of q vertices in G^0 of q-Clique and the subgraph $G[R^0]$ induced by R^0 in G^0. Since the size of the maximum clique in G is at most $q-1$, $G[R^0]$ contains at most $q(q-1)/2-1$ edges. It follows that we can only obtain the new independent set of at most $q(q-1)/2 - 1 = k(k-1)/2 - 1$ vertices by deleting any subset of $q = k$ vertices from V_v, i.e., the size of any independent set is at most $|V| - k + k(k-1)/2 - 1$. This completes the proof. □

Since comparability graphs and perfect graphs are superclasses of bipartite graphs [11], we obtain the following corollary:

Corollary 1. *BD-MaxIS is NP-hard even if the input is restricted to comparability graphs, or perfect graphs.*

4 Polynomial-time solvable graph subclasses of BD-MaxIS

4.1 Co-comparability graphs

In this section, for BD-MaxIS on co-comparability graphs, we design a polynomial-time algorithm, while BD-MaxIS on perfect graphs is NP-hard as shown in the

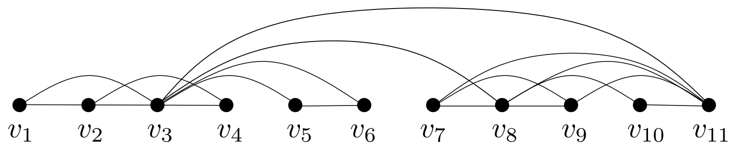

Fig. 3. Umbrella-free vertex ordering. For example, consider three vertices v_3, v_7, and v_8. Since there is an edge between v_3 and v_8, there is an edge $\{v_7, v_8\}$.

previous section. Before the detailed description of our algorithm ALG_CoC, we give the *vertex ordering characterization* of co-comparability graphs.

Vertex ordering characterization. A *vertex ordering* of $G = (V, E)$ is a bijection $\sigma : V \leftrightarrow \{1, 2, \ldots, n\}$, i.e., for $v \in V$, $\sigma(v)$ denotes the unique position of v in σ, $\sigma(u) \neq \sigma(v)$ for $u \neq v$. For two vertices u and v, we write that $u <_\sigma v$ if and only if $\sigma(u) < \sigma(v)$. For two vertices $u, v \in V$, we say that u is left (resp., right) to v in σ if $u <_\sigma v$ (resp., $v <_\sigma u$). A *vertex ordering characterization* is an ordering on the vertices of a graph that satisfies certain properties. If every $G \in \mathcal{G}$ has a total ordering of its vertices that satisfies some property, then we say that the graph class $\mathcal{G}$ has a vertex ordering characterization on the property, which is often used to design polynomial-time algorithms. The co-comparability graph has the following vertex ordering characterization:

Proposition 1 ([15]). *A graph $G = (V, E)$ is a* co-comparability *graph if and only if there exists a vertex ordering σ of its vertices such that for every triple of vertices u, v, and w such that if $u <_\sigma v <_\sigma w$ and $\{u, w\} \in E$, then $\{u, v\} \in E$ or $\{v, w\} \in E$ (or both).*

The vertex ordering σ that satisfies the above proposition is called an *umbrella-free ordering* since σ does not contain an *umbrella*, which is a triple of vertices $u <_\sigma u <_\sigma w$ with $\{u, w\} \in E$ but $\{u, v\}, \{v, w\} \notin E$. For example, see Figure 3. McConnell and Spinrad presented an algorithm to compute such a vertex ordering in $O(n + m)$ time [16].

Algorithm. Our algorithm ALG_CoC for BD-MaxIS on co-comparability graphs is based on a dynamic programming along the vertex ordering of co-comparability graphs. Given a co-comparability graph $G = (V, E)$, we first compute an umbrella-free vertex ordering σ of V in $O(n + m)$ time. Suppose that the ordering σ is $v_1 <_\sigma v_2 <_\sigma \cdots <_\sigma v_n$. In order to make the description of our algorithm easier, we add an isolated dummy vertex v_0 so that $v_0 <_\sigma v_1$ into the leftmost position (i.e., the 0th position). Let $V_{i..j} = \{v_i, v_{i+1}, \ldots, v_j\}$ be the set of the $j - 1 + 1$ consecutive vertices, v_i through v_j. Also, let $N_L(v_i) = N(v_i) \cap V_{0..(i-1)} = \{v_j \in V_{0..(i-1)} \mid \{v_i, v_j\} \in E\}$ is called the left neighborhood of v_i. Let δ_i be the subscript of the leftmost vertex in $N_L(v_i)$. If $N_L(v_i) = \emptyset$, then $\delta_i = i$. Let $\overline{N}_L(v_i) = \{v_j \in V_{\delta_i..(i-1)} \mid \{v_j, v_i\} \notin E\}$. See Figure 3 again. For example, $N_L(v_{11}) = \{v_3, v_7, v_8, v_9, v_{10}\}$, $\overline{N}_L(v_{11}) = \{v_4, v_5, v_6\}$, and $\delta_{11} = 3$.

Now, consider the following two values *pick* and j: The former value $\textit{pick} \in \{0, 1\}$ indicates whether v_i is picked into a (partial) solution S or not. The latter value $j \in \{0, 1, \ldots, k\}$ indicates the number of deleted vertices from the initial solution S^0 in order to construct S. For the ith vertex v_i, we define $IS(i, pick, j)$ to be the value of a maximum independent set in the induced subgraph $G[V_{1..i}]$ satisfying the following: (i) If $pick = 1$, then a partial solution S for $G[V_{1..i}]$ includes the ith vertex v_i; otherwise, S does not include v_i. (ii) The number $|(V_{1..i} \cap S^0) \setminus S|$ of deleted vertices so far is exactly j.

Let $\#S(i_1, i_2)$ be the number of vertices in $S^0 \cap V_{i_1..i_2}$, i.e., the number of vertices in $\{v_{i_1}, \ldots, v_{i_2}\}$ which are picked into the initial solution S^0. Initially we set $IS(0, pick, j) = 0$ for $pick = 0, 1$, and $j = 0, 1, \ldots k$. The recursive formula of our DP-based algorithm `ALG_CoC` is divided into the following two cases, (Case 1) v_i is not in the initial solution S^0, i.e, $v_i \notin S^0$, and (Case 2) v_i is in S^0, i.e., $v_i \in S^0$.

(Case 1) Suppose that $v_i \notin S^0$. The recursive formula is defined as follows:

$$IS(i, pick, j) = \begin{cases} \max\{IS(i-1, 0, j), IS(i-1, 1, j)\} \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } pick = 0; \\ 1 + \max\Big\{ \displaystyle\max_{v_\ell \in \overline{N}_L(v_i)} \big\{IS(\ell, 1, j - \#S(\ell+1, i-1))\big\}, \\ \qquad\qquad\qquad IS(\delta_i, 0, j - \#S(\delta_i + 1, i-1))\Big\} \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } pick = 1 \text{ and } \delta_i \neq i; \\ 1 + \max\{IS(i-1, 0, j), IS(i-1, 1, j)\} \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } pick = 1 \text{ and } \delta_i = i. \end{cases}$$

(1) Consider the case where v_i is not picked into the solution S. Then, the number $|(V_{1..i} \cap S^0) \setminus S|$ of deleted vertices at v_i is equal to the number $|(V_{1..(i-1)} \cap S^0) \setminus S|$ at v_{i-1}. Furthermore, one sees that the value of the maximum independent set in the induced subgraph $G[V_{1..i}]$ is equal to the value of a maximum independent set in the induced subgraph $G[V_{1..(i-1)}]$.
(2) Suppose that v_i is picked into the solution S. Then, the value of the maximum independent set in $G[V_{1..i}]$ increases by one. One sees that all the left neighborhood of v_i cannot be picked into S, but $v_\ell \in \overline{N}_L(v_i)$ can be possibly picked into S since v_ℓ is not adjacent to v_i. (i) If all the vertices $v_\ell \in \overline{N}_L(v_i)$ are not picked into S, then $IS(i, 1, j)$ (now $pick = 1$) is equal to the value of a maximum independent set in the induced subgraph $G[V_{1..\delta_i}]$ which is stored into $IS(\delta_i, 0, j - \#S(\delta_i, i-1))$ since all the vertices of $S \cap V_{\delta_i, i-1}$ must *not* be included in the solution S. (ii) For ease of exposition, take a look at five vertices v_3, v_4, v_5, v_6, and v_{11} in Figure 3. If v_{11} is in S, then v_3 is not in S. Suppose that v_4 and v_6 in $\overline{N}_L(v_{11})$ is picked into S and v_5 is not in S. Since v_5 is not in S, $IS(5, 0, j)$ can be obtained from $\max\{IS(4, 0, j), IS(4, 1, j)\}$ if v_5 is in the initial solution S^0, and from $\max\{IS(4, 0, j-1), IS(4, 1, j-1)\}$ if v_5 is not in S^0. That is, if v_5 is not in S, then the current information of v_5 can be obtained from the information of the left vertex v_4. Therefore, it is enough to verify the information of $v_\ell \in \overline{N}_L(v_i)$ only when v_ℓ is picked into

S. This is the main reason why our DP-based algorithm works in polynomial time if the vertex ordering characterization is umbrella-free.
(3) Suppose that v_i is picked into the solution S, and $N_L(v_i) = \emptyset$. Then, $IS(i, pick, j)$ can be computed from the two values $IS(i-1, 0, j)$ and $IS(i-1, 1, j)$ of the left vertex v_{i-1}.

(Case 2) Suppose that $v_i \in S^0$. One sees that "v_i is not picked" means that v_i must be deleted from the initial solution S^0. The recursive formula is almost the same as the formula in (Case 1), but, the number of deleted vertices is different if v_i is not picked into the solution S:

$$IS(i, pick, j) = \begin{cases} \max\{IS(i-1, 0, j-1), IS(i-1, 1, j-1)\} \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } pick = 0; \\ 1 + \max\Big\{ \max\limits_{v_\ell \in \overline{N}_L(v_i)} \{IS(\ell, 1, j - \#S(\ell+1, i-1))\}, \\ \qquad\qquad\qquad IS(\delta_i, 0, j - \#S(\delta_i + 1, i-1))\Big\} \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } pick = 1 \text{ and } \delta_i \neq i; \\ 1 + \max\{IS(i-1, 0, j), IS(i-1, 1, j)\} \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } pick = 1 \text{ and } \delta_i = i. \end{cases}$$

Our algorithm `ALG_CoC` computes the value of $IS(i, pick, j)$ and stores it into a three-dimensional table IS of size $(n+1) \times 2 \times (k+1) = O(kn)$. Then, finally, `ALG_CoC` returns $\max_{0 \leq j \leq k} \{IS(n, 0, j), IS(n, 1, j)\}$.

Theorem 2. *Given an n-vertex co-comparability graph G and a non-negative integer k, BD-MaxIS can be solved in $O(kn^2)$ time.*

Proof. Given the co-comparability graph G, we can obtain its umbrella-free ordering in $O(n^2)$ time by using the method proposed in [16]. Clearly, each table entry takes $O(n)$ time to compute. Since the table size is $O(kn)$, the running time of `ALG_CoC` is $O(kn^2)$. $\square$

4.2 Interval graphs

Since every interval graph is co-comparability, BD-MaxIS on interval graphs can be solved in $O(kn^2)$ time by `ALG_CoC`. Fortunately, however, we can provide a faster algorithm `ALG_Int` if the following vertex ordering characterization of interval graphs, known as an *interval ordering*, is given:

Proposition 2 ([18]). *A graph $G = (V, E)$ is an interval graph if and only if there exists an ordering σ of its vertices such that for every triple of vertices u, v, and w such that if $u <_\sigma v <_\sigma w$ and $\{u, w\} \in E$, then $\{u, v\} \in E$.*

Theorem 3. *Suppose that we are given the interval ordering of an n-vertex interval graph G and a non-negative integer k as input. Then, BD-MaxIS can be solved in $O(kn)$ time. (The proof will appear in the full version of this paper.)*

Since the interval ordering of interval graphs with n vertices and m edges can be obtained in $O(n+m)$ [3], we obtain the following corollary:

Corollary 2. *Given an interval graph with n vertices and m edges, and a non-negative integer k, BD-MaxIS can be solved in $O(kn+m)$ time.*

4.3 Convex bipartite graphs

As shown in Section 3, BD-MaxIS on bipartite graphs is NP-hard. One of the famous subclasses of bipartite graphs is the convex bipartite graph class. In this section we show that BD-MaxIS on convex bipartite graphs can be solved in polynomial-time. Here, we give our notation and additional terminology.

Let $G=(V,W,E)$ be a convex bipartite graph with respect to W. Suppose that V and W have n_1 and n_2 vertices, $V=\{v_1,v_2,\ldots,v_{n_1}\}$ and $W=\{w_1,w_2,\ldots,w_{n_2}\}$, where the convex vertex ordering σ is $w_1 <_\sigma w_2 <_\sigma \cdots <_\sigma w_{n_2}$. The vertex ordering of vertices in V is given later. See Figure 4. For example, the neighborhood $N(v_3)=\{w_3,w_4,w_5,w_6,w_7\}$ of v_3 contains five consecutive vertices. Let w_i^ℓ and w_i^r be the leftmost and the rightmost vertices in $N(v_i)$ of v_i, respectively. Assume that n_1 vertices in $V=\{v_1,\ldots,v_{n_1}\}$ are sorted such that $w_1^r <_\sigma \cdots <_\sigma w_{n_1}^r$ holds by the vertex ordering σ, with ties broken arbitrarily. The ordering can be computed in $O(n_1 \log n_1)$. For the convex bipartite graph in Figure 4, $w_1^r=w_2^r=w_5$, $w_3^r=w_7$, $w_4^r=w_9$, and $w_5^r=w_{10}$. As for the ith vertex v_i, $e_v^\ell(i)=\{v_i,w_i^\ell\}$ and $e_v^r(i)=\{v_i,w_i^r\}$ are called the leftmost and the rightmost edges of v_i, respectively. The other edges are called middle edges of v_i. If v_i is incident to one edge only, then the edge is also regarded as the rightmost edge. Now we define a mapping $right_v : E \to \{0,1\}$ such that if an edge e is the rightmost edge, then $right_v(e)=1$; otherwise, $right_v(e)=0$. Similarly, let v_i^ℓ and v_i^r be the leftmost and the rightmost vertices in $N(w_i)$ of w_i, respectively. As for the ith vertex w_i, $e_w^\ell(i)=\{v_i^\ell,w_i\}$ and $e_w^r(i)=\{v_i^r,w_i\}$ are called the leftmost and the rightmost edges of w_i, respectively. The other edges are called middle edges of w_i. If w_i is incident to one edge only, then the edge is regarded as the leftmost edge. Again, we define a mapping $left_w : E \to \{0,1\}$ such that if an edge e is the leftmost edge, then $left_w(e)=1$; otherwise, $left_w(e)=0$. Take a look at w_5 in Figure 4. One sees that the neighborhood $N(w_5)$ of w_5 is v_1, v_2, v_3, and v_4. Then, for example, $right_v(\{v_1,w_5\})=1$ and $right_v(\{v_2,w_5\})=1$, but, $right_v(\{v_3,w_5\})=0$. Also, $left_w(\{v_1,w_5\})=1$.

Algorithm. Our algorithm `ALG_CB` for BD-MaxIS on convex bipartite graphs with respect to W follows the convex ordering of W, roughly from the leftmost edge to the rightmost edge. More precisely, `ALG_CB` uses the *edge* ordering σ_e such that $\{w_1,v_{1_1}\} <_{\sigma_e} \{w_1,v_{1_2}\} <_{\sigma_e} \ldots <_{\sigma_e} \{w_1,v_{1_{|N(w_1)|}}\} <_{\sigma_e} \{w_2,v_{2_1}\} <_{\sigma_e} \ldots <_{\sigma_e} \{w_2,v_{2_{|N(w_2)|}}\} <_{\sigma_e} \cdots <_{\sigma_e} \{w_{n_2},v_{n_{2_{|N(w_{n_2})|}}}\}$, where $N(w_i)=\{v_{i_1},\ldots,v_{i_{|N(w_i)|}}\}$ and $v_{i_1} <_\sigma \cdots <_\sigma v_{i_{|N(w_i)|}}$ for $1 \le i \le n_2$. That is, the leftmost $|N(w_1)|$ edges of the edge ordering are incident to w_1, the next $|N(w_2)|$ edges are incident to w_2, and so on.

Let $[pick_i, pick_{i_q}] \in \{[0,0],[0,1],[1,0]\}$ be a status of the edge $\{w_i,v_{i_q}\}$ such that if $pick_i=1$ (resp., $pick_i=0$), then w_i is picked (resp, not picked) into

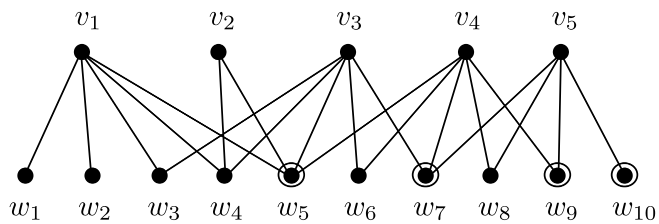

Fig. 4. Convex bipartite

the solution S and if $pick_{i_q} = 1$ (resp., $pick_{i_q} = 0$), then v_{i_q} is picked (resp, not picked) into the solution S for $1 \le i \le n_2$ and $1 \le q \le |N(w_i)|$. For the ith vertex w_i, we define $IS([i, i_q], [pick_i, pick_{i_q}], j)$ to be the value of a maximum independent set in the induced subgraph $G[\{w_1, \ldots, w_i\} \cup N(w_1) \cup \cdots \cup N(w_{i-1}) \cup \{v_{i_1}, \ldots, v_{i_q}\}]$ satisfying that the number of deleted vertices is exactly j, where $[pick_i, pick_{i_q}]$ is $[0, 0]$, $[0, 1]$, or $[1, 0]$.

In order to make the description of our algorithm easier, we add two dummy vertices v_0 and w_0 into the leftmost positions in V and W, respectively. Initially we set $IS([0, 0], [0, 0], j) = IS([0, 0], [0, 1], j) = IS([0, 0], [1, 0], j) = 0$ for $j = 0, 1, \ldots, k$.

Consider a vertex v_i and its neighbor vertices in $N(v_i)$. If v_i is in S, then any vertex in $N(v_i)$ cannot be picked into S. Conversely, if at least one vertex in $N(v_i)$ is in S, then v_i cannot be picked into S. See Figure 4 again. Consider six vertices $W_{1..6} = \{w_1, w_2, w_3, w_4, w_5, w_6\}$ and their four neighbor vertices $V_{1..4} = \{v_1, v_2, v_3, v_4\}$, and also 13 edges between $W_{1..6}$ and $V_{1..4}$. If every vertex in $W_{1..6}$ is fixed to be picked or not into S, then the status $v_1 \in S$ or $v_1 \notin S$, and $v_2 \in S$ or $v_2 \notin S$ can be fixed since $\{v_1, w_5\}$ and $\{v_2, w_5\}$ are the rightmost edges and $w_5 <_\sigma w_6$. On the other hand, for example, the status $v_3 \in S$ or $v_3 \notin S$ cannot be fixed since it depends on whether w_7 is picked or not into S. Therefore, roughly speaking, as for vertices in W, (i) if w_i is picked into S, then the size of S increases by one; on the other hand, as for vertices in V, (ii) if any neighbor vertex in $N(v_i) \setminus \{w_i^r\}$ are not picked into S, then the size of S is incremented when $w_i^r \in S$ is determined.

The recursive formula of our DP-based algorithm `ALG_CB` is divided into the following three cases, (Case 1) $w_i, v_{i_q} \notin S^0$, (Case 2) $w_i \notin S^0$ but $v_{i_q} \in S^0$, and (Case 3) $w_i \in S^0$ but $v_{i_q} \notin S^0$. Furthermore, each of the three cases (Case 1), (Case 2), and (Case 3) has four sub-cases $(right_v(\{w_i, v_{i_q}\}), left_w(\{w_i, v_{i_q}\})) = (0, 0)$, $(0, 1)$, $(1, 0)$, and $(1, 1)$. Note that if an edge $\{i, i_q\} \notin E$, then we set $IS([i, i_q], [pick_i, pick_{i_q}], j) = 0$ in the right-hand side of the recursive formula. Here we show only (Case 1) since (Case 2) and (Case 3) are very similar to (Case 1); (Case 2) and (Case 3) will appear in the full version of this paper.

(Case 1) $w_i, v_{i_q} \notin S^0$.

(i) Suppose that $right_v(\{w_i, v_{i_q}\}) = 0$ and $left_w(\{w_i, v_{i_q}\}) = 0$.

$$
IS([i,i_q],[pick_i,pick_{i_q}],j)
$$
$$
=\begin{cases}
\max\{IS([i,i_{q-1}],[0,0],j), IS([i,i_{q-1}],[0,1],j), \\
\qquad IS([i-1,i_q],[0,0],j), IS([i-1,i_q],[1,0],j)\} \\
\hfill \text{if } pick_i = 0 \text{ and } pick_{i_q} = 0 \\
\max\{IS([i,i_{q-1}],[1,0],j), IS([i-1,i_q],[0,0],j), \\
\qquad IS([i-1,i_q],[1,0],j)\} \\
\hfill \text{if } pick_i = 1 \text{ and } pick_{i_q} = 0 \\
\max\{IS([i,i_{q-1}],[0,0],j), IS([i,i_{q-1}],[0,1],j), \\
\qquad IS([i-1,i_q],[0,1],j)\} \\
\hfill \text{if } pick_i = 0 \text{ and } pick_{i_q} = 1
\end{cases}
$$

(ii) Suppose that $right_v(\{w_i, v_{i_q}\}) = 1$ and $left_w(\{w_i, v_{i_q}\}) = 0$.

$$
IS([i,i_q],[pick_i,pick_{i_q}],j)
$$
$$
=\begin{cases}
\max\{IS([i,i_{q-1}],[0,0],j), IS([i,i_{q-1}],[0,1],j), \\
\qquad IS([i-1,i_q],[0,0],j), IS([i-1,i_q],[1,0],j)\} \\
\hfill \text{if } pick_i = 0 \text{ and } pick_{i_q} = 0 \\
\max\{IS([i,i_{q-1}],[1,0],j), IS([i-1,i_q],[0,0],j), \\
\qquad IS([i-1,i_q],[1,0],j)\} \\
\hfill \text{if } pick_i = 1 \text{ and } pick_{i_q} = 0 \\
1+\max\{IS([i,i_{q-1}],[0,0],j), IS([i,i_{q-1}],[0,1],j), \\
\qquad IS([i-1,i_q],[0,1],j)\} \\
\hfill \text{if } pick_i = 0 \text{ and } pick_{i_q} = 1
\end{cases}
$$

(iii) Suppose that $right_v(\{w_i, v_{i_q}\}) = 0$ and $left_w(\{w_i, v_{i_q}\}) = 1$.

$$
IS([i,i_q],[pick_i,pick_{i_q}],j)
$$
$$
=\begin{cases}
\max\{IS([i-1,i_q],[0,0],j), IS([i-1,i_q],[1,0],j)\} \\
\hfill \text{if } pick_i = 0 \text{ and } pick_{i_q} = 0 \\
1+\max\{IS([i-1,i_q],[0,0],j), IS([i-1,i_q],[1,0],j)\} \\
\hfill \text{if } pick_i = 1 \text{ and } pick_{i_q} = 0 \\
IS([i-1,i_q],[0,1],j) \\
\hfill \text{if } pick_i = 0 \text{ and } pick_{i_q} = 1
\end{cases}
$$

(iv) Suppose that $right_v(\{w_i, v_{i_q}\}) = 1$ and $left_w(\{w_i, v_{i_q}\}) = 1$.

$$IS([i, i_q], [pick_i, pick_{i_q}], j) = \begin{cases} \max\{IS([i-1, i_q], [0,0], j), IS([i-1, i_q], [1,0], j)\} \\ \qquad\qquad\qquad\qquad \text{if } pick_i = 0 \text{ and } pick_{i_q} = 0 \\ 1 + \max\{IS([i-1, i_q], [0,0], j), IS([i-1, i_q], [1,0], j)\} \\ \qquad\qquad\qquad\qquad \text{if } pick_i = 1 \text{ and } pick_{i_q} = 0 \\ 1 + IS([i-1, i_q], [0,1], j) \\ \qquad\qquad\qquad\qquad \text{if } pick_i = 0 \text{ and } pick_{i_q} = 1 \end{cases}$$

Theorem 4. *Given an m-edge convex bipartite graph G and a non-negative integer k, BD-MaxIS can be solved in $O(km)$ time.*

Proof. Our algorithm `ALG_CB` computes the value of $IS([i, i_q], [pick_i, pick_{i_q}], j)$ and stores it into a two-dimensional table IS of size $(m+1)\times 3\times(k+1) = O(km)$. Then, finally, `ALG_CB` returns

$$\max_{0\le j\le k} \Big\{IS([n_2, n_1], [0,0], j), IS([n_2, n_1], [0,1], j), IS([n_2, n_1], [1,0], j)\Big\}.$$

Since each table entry takes $O(1)$ time to compute, the running time of `ALG_CB` is $O(km)$. □

4.4 Chordal graphs

The class of chordal graphs is one of the important subclasses of perfect graphs. Indeed, chordal graphs have attracted interest in graph theory since several combinatorial optimization problems that are intractable turn to be tractable on chordal graphs. In this section we provide a polynomial-time algorithm for BD-MaxIS on chordal graphs, which is again based on a dynamic programming for the *clique tree* representation of chordal graphs.

Clique tree. Let $\mathcal{Q}_G$ be the set of all maximal cliques in a graph G, and let $\mathcal{Q}_v \subseteq \mathcal{Q}_G$ be the set of all maximal cliques that contain a vertex $v \in V(G)$. It is known [4, 9] that G is chordal if and only if there exists a tree $T = (\mathcal{Q}_G, E(T))$ such that each node[1] of T corresponds to a maximal clique in $\mathcal{Q}_G$ and T has the *induced subtree property*, i.e., the subtree $T[\mathcal{Q}_v]$ induced by $\mathcal{Q}_v$ is connected for every vertex $v \in V(G)$. Such a tree is called a *clique tree* of G, and it can be constructed in linear time [1]. Given a chordal graph $G = (V, E)$, we construct a clique tree T and then a *rooted* clique tree $T(Q_r)$ of G by selecting an arbitrary node in T as a root Q_r. For the rooted clique tree $T(Q_r)$ of G and a node Q_i in $T(Q_r)$, $T(Q_i)$ represents the subtree rooted at Q_i. See Figure 5, where the left is a chordal graph G of 11 vertices, and the right is its rooted clique tree $T(Q_r)$.

[1] We will refer to a node in a tree in order to distinguish it from a vertex in a graph.

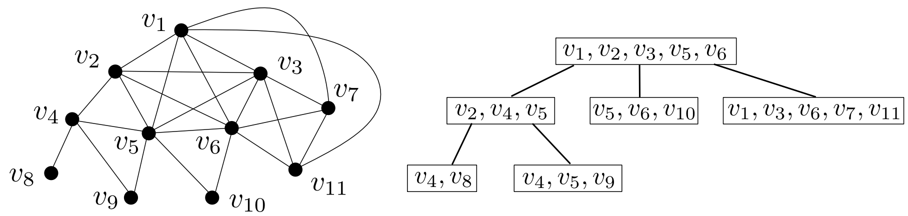

Fig. 5. (Left) Chordal graph G, and (Right) its rooted clique tree T_r with a root $G[\{v_1, v_2, v_3, v_4, v_5\}]$

Let $V(Q_i)$ and $V(T(Q_i))$ be the set of vertices in the node Q_i and the union of vertices in all nodes of the subtree $T(Q_i)$ rooted at Q_i, respectively.

In this paper, we consider a *weak clique tree* representation of a chordal graph [13]. Each node of the original clique tree must be a maximal clique, but each node of the weak clique tree is just a clique. It is known [13] that every chordal graph $G = (V, E)$ has a weak clique tree T such that T is a *binary* tree with $O(n)$ nodes and the sum of all cardinalities of its nodes is $O(n + m)$. Furthermore, every weak clique tree of a chordal graph is still a *tree decomposition*, i.e., satisfies the induced subtree property. Therefore, the dynamic programming using the weak clique tree works well.

Algorithm. Given a chordal graph $G = (V, E)$, we first compute a rooted weak clique tree $T(Q_r)$ of G in $O(n+m)$ time. For the sake of notational convenience, let $T = T(Q_r)$, $T_i = T(Q_i)$, and let $V_T = \{Q_1, Q_2, \ldots, Q_{|V_T|}\}$ be the set of nodes in T. Suppose that Q_{i_ℓ} and Q_{i_r} in V_T respectively are the left and the right children of Q_i, if exist. Recall that $V(T_i)$ $(= V(T(Q_i)))$ is the union of vertices in all nodes in the subtree T_i rooted at Q_i, and $V(T) = V(G)$.

Let S_{T_i} be an independent set in the subtree induced by $V(T_i)$, i.e., S_{T_r} is an independent set S of G. For a node Q_i in T, $S_i \subseteq V(Q_i)$, and $j_i \in \{0, \ldots, k\}$, we define $IS(i, S_i, j_i)$ to be the maximum size of the independent set S_{T_i} in T_i satisfying that $S_{T_i} \cap V(Q_i) = S_i$ and the number $|S^0 \setminus S_{T_i}|$ of deleted vertices from T_i is exactly j_i. A high level description of the recursive formula used by our algorithm `ALG_Cho` is very similar to the formula given in [2], although we have to count the number of deleted vertices $|S^0 \setminus S_{T_i}|$: The algorithm `ALG_Cho` computes the values of $IS(i, S_i, j_i)$ for all nodes Q_i in T. This can be done in a typical bottom-up manner in the weak clique tree. Then, finally, `ALG_Cho` returns a maximum independent set satisfying the deletion constraint at the root node Q_r. Each table value $IS(i, S_i, j_i)$ is computed after the table values of the two children are obtained. Note that each node Q_i in T is a clique, and thus we can pick at most one vertex from Q_i. It follows that for every node Q_i, the number of possible choices as S_i is at most $|V(Q_i)| + 1$ including $S_i = \emptyset$. Further details are omitted here, but, we can obtain the following theorem:

Theorem 5. *Given an n-vertex chordal graph G and a non-negative integer k, BD-MaxIS can be solved in $O(k^2(n + m)^2)$ time.*

5 Concluding remarks

The Minimum Vertex Cover problem is also known as one of the Karp's 21 fundamental NP-hard problems [14]. We can similarly define a bounded-deletion variant of the Minimum Vertex Cover problem:

> Bounded-Deletion Minimum Vertex Cover (BD-MinVS)
> **Input:** An unweighted graph $G = (V, E)$, an initial feasible solution (i.e., a vertex cover) $S^0 \subseteq V$, and a non-negative integer k.
> **Goal:** The goal is to find a vertex cover $S \subseteq V$ such that $|S^0 \setminus S| \leq k$ and $|S|$ is minimized.

Although details are omitted here, we can show the following results by a similar polynomial-time reduction in the proof of Theorem 1:

Corollary 3. *BD-MinVS is NP-hard even if the input is restricted to bipartite graphs, comparability graphs, or perfect graphs.*

Similarly, we can consider a deletion-bounded variant of the classical and famous NP-hard MaxClique [14]:

> Bounded-Deletion Maximum Clique (BD-MaxClique)
> **Input:** An unweighted graph $G = (V, E)$, an initial feasible solution (i.e., a clique) $S^0 \subseteq V$, and a non-negative integer k.
> **Goal:** The goal is to find a clique set $S \subseteq V$ such that $|S^0 \setminus S| \leq k$ and $|S|$ is maximized.

Every independent set S in the complement graph $\overline{G}$ of a graph G forms a clique induced by S in G. Therefore, we can show the following:

Corollary 4. *BD-MaxClique is NP-hard even if the input is restricted to co-bipartite graphs, co-comparability graphs, or perfect graphs.*

Future work is to show the tractability/intractability of BD-MaxClique on graph classes, such as chordal graphs, interval graphs, and permutation graphs.

Acknowledgments. This work is partially supported by NSERC Canada, and JSPS KAKENHI Grant Numbers JP17K00024, JP21K11755 and JP22K11915.

References

1. J.R.S. Blair and B. Peyton. An introduction to chordal graphs and clique trees. In: Graph Theory and Sparse Matrix Computation, IMA Vol.56, pp. 1–29 (1993).
2. H.L. Bodlaender. A tourist guide through treewidth. Acta Cybern, Vol.11, pp.1–21 (1993).

3. K.S. Booth and G.S. Lueker. Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithm. J. Comput. System Sci., Vol.13, pp.335–379 (1976).
4. P. Buneman. A characterization of rigid circuit graphs. Discrete Mathematics, Vol.9, pp.205–212 (1974).
5. M. Cygan, F.V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. Parameterized Algorithms. Springer International Publishing (2015).
6. M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. Theoretical Computer Science, Vol.1, pp.237–267 (1976).
7. F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of chordal graph. SIAM J. Comput. Vol.1, pp.180–187 (1972).
8. F. Gavril. Algorithms on circular-arc graphs. Networks, Vol.4, pp.357–369 (1974).
9. F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. Journal of Combinatorial Theory, Series B, Vol.16, pp.47–56 (1974).
10. M.C. Golumbic. The complexity of comparability graph recognition and coloring. Computing, Vol.18, pp.199–208 (1977).
11. M.C. Golumbic. Algorithmic Graph Theory and Perfect Graphs. Annals of Discrete Mathematics, Vol.57, North-Holland Publishing Co, Amsterdam, The Netherlands (2004).
12. M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. Combinatorica, Vol.1(2), pp.169–197 (1981).
13. F. Kammer. Treelike and chordal graphs: algorithms and generalizations. University of Augsburg (2012).
14. R.M. Karp. Reducibility among combinatorial problems. Complexity of Computer Computations, pp.85–103 (1972).
15. D. Kratsch and L. Stewart. Domination on cocomparability graphs. SIAM J. Discrete Math., Vol.6(3), pp.400–417 (1993).
16. R.M. McConnell and J.P. Spinrad. Modular decomposition and transitive orientation. Discrete Math. Vol.201(1), pp.189–241 (1999).
17. O.J. Murphy. Computing independent sets in graphs with large girth. Discrete Applied Mathematics, Vol.35, pp.167–170 (1992).
18. S. Olariu. An optimal greedy heuristic to color interval graphs. Information Processing Letters, Vol.37(1), pp.21–25 (1991).
19. S. Poljak. A note on stable sets and coloring of graphs. Comment. Math. Univ. Carolin., Vol.15, pp.307–309 (1974).
20. O. Seref, R.K. Ahuja, and J.B. Orlin. Incremental network optimization: theory and algorithms. Operations Research, Vol.57(3), pp.586–594 (2009).