Counting the Number of Matchings in Chordal and Chordal Bipartite Graph Classes

Yoshio Okamoto[1⋆], Ryuhei Uehara[2⋆⋆], and Takeaki Uno[3⋆⋆⋆]

[1] Graduate School of Information Science and Engineering, Tokyo Institute of Technology, Ookayama 2-12-1-W8-88, Meguro-ku, Tokyo 152-8552, Japan. E-mail: okamoto@is.titech.ac.jp
[2] School of Information Science, JAIST, Asahidai 1-1 , Nomi, Ishikawa 923-1292, Japan. E-mail: uehara@jaist.ac.jp
[3] National Institute of Informatics, Hitotsubashi 2-1-2, Chiyoda-ku, Tokyo 101-8430, Japan. E-mail: uno@nii.jp

Abstract. We provide polynomial-time algorithms for counting the number of perfect matchings and the number of matchings in chain graphs, cochain graphs, and threshold graphs. These algorithms are based on newly developed subdivision schemes that we call a recursive decomposition. On the other hand, we show the #P-completeness for counting the number of perfect matchings in chordal graphs, split graphs and chordal bipartite graphs. This is in an interesting contrast with the fact that counting the number of independent sets in chordal graphs can be done in linear time.

1 Introduction

The study of graph classes has been motivated by the fact that a lot of NP-hard problems can be solved in polynomial time when the input is restricted. While this research direction leads to many polynomial-time algorithms for decision problems and optimization problems, such results for counting problems seem rare. With this motivation, the authors studied problems to count the independent sets in chordal graphs [13], and refinement for interval graphs has been proposed by Lin [11] and Lin and Chen [12]. However, the current understanding for counting problems in graph classes is still poor. Counting algorithms may require properties of graphs that are not needed for solving decision and optimization problems.

This paper is concerned with perfect matchings. A perfect matching of a graph is one of the fundamental objects when we study counting problems. When Valiant [18] introduced the complexity class #P, he already proved that counting the perfect matchings in a bipartite graph is #P-complete. In another paper [19], he also proved that counting all matchings in a bipartite graph is #P-complete. His results were refined by Dagum and Luby [3] showing that counting the perfect matchings in a 3-regular bipartite graph is #P-complete, and by Vadhan [16] showing that counting all matchings in a bipartite graph of maximum degree 4 and in a planar bipartite graph of maximum degree 6 is #P-complete. There are also some results on the positive side, namely for polynomial-time algorithms. The perfect matchings in a planar graph can be counted in polynomial time [5, 9, 14] via the so-called Pfaffian orientations. A generalization of this approach yields a polynomial-time algorithm for graphs of bounded genus [7, 15]. Furthermore, we can count the perfect matchings in a graph of bounded treewidth [1]. Basically, all positive results are concerned with sparse graphs.

This paper concentrates on classes of chordal graphs and chordal bipartite graphs. An interesting phenomenon to be proven here is the #P-completeness for the counting problem of matchings in chordal graphs, while we can count the number of independent sets in chordal graphs in linear time [13]. We also prove that the matching counting is #P-complete even for chordal bipartite graphs. Therefore, we seek for subclasses of these graph classes for which the matchings can be counted in polynomial time. We give such polynomial-time algorithms for the following classes of graphs: Chain graphs, cochain

⋆ Supported by Global COE Program "Computationism as a Foundation for the Sciences" and Grant-in-Aid for Scientific Research from Ministry of Education, Science and Culture, Japan, and Japan Society for the Promotion of Science.
⋆⋆ Supported by Ministry of Education, Science and Culture, Japan, and Japan Society for the Promotion of Science.
⋆⋆⋆ Supported by Ministry of Education, Science and Culture, Japan, and Japan Society for the Promotion of Science.

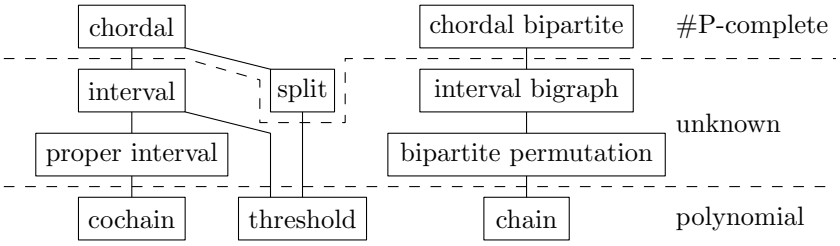

Fig. 1. Inclusion relationship among the graph classes in this paper and the summary of our results.

graphs, and threshold graphs. The definitions will be given later, but there is a relation among these classes as dipicted in Fig. 1. In the figure, we also have the classes of bipartite permutation graphs, proper interval graphs, interval graphs and interval bigraphs. For these four classes, complexity of counting the matchings is unsettled. It would be possible to design a quasi-polynomial-time algorithm by extending the technique developed in this paper, but polynomial-time algorithms are out of reach. This is a main open problem this paper leaves for us.

2 Preliminaries

We assume the reader is familiar with basic terminology on graphs. A graph is denoted by $G = (V, E)$ when V is the vertex set and E is the edge set of G. The *neighborhood* of a vertex $v \in V$ is the set $N_G(v) = \{u \in V \mid \{u, v\} \in E\}$. For a subset $U \subseteq V$, the *subgraph of G induced by U* is the graph (U, F), where $F = \{\{u, v\} \in E \mid u, v \in U\}$, and denoted by $G[U]$. For given two graphs $G = (V, E)$ and $G' = (V', E')$, we denote the graph $(V \cup V', E \cup E')$ by $G \cup G'$. A vertex set C is a *clique* if all pairs of vertices in C are joined by an edge. A vertex set I is *independent* if no pair of vertices in I is joined by an edge. An edge set M is a *matching* if no pair of edges in M shares an endpoint. Note that the empty set is a matching of size zero in any graph. An endpoint v of an edge e in a matching M is said to be *matched* by M. A matching is *perfect* if all vertices are matched by the matching.

A graph $G = (V, E)$ is *bipartite* if V can be partitioned into two sets X and Y such that every edge joins a vertex in X and the other vertex in Y. We denote a bipartite graph by $G = (X, Y, E)$ when the partition is given. A bipartite graph G is *complete* if every vertex in X is adjacent to all vertices in Y.

For a graph G, the number of matchings in G is denoted by $\mu(G)$, the number of matchings of size i in G is denoted by $\mu_i(G)$, and the number of perfect matchings in G is denoted by $\pi(G)$.

3 Polynomial-time algorithm for chain graphs

In this section, we study chain graphs. A chain graph is also called a difference graph [8], a bisplit graph [6], and nonseparable bipartite graph [4].

To define a chain graph, we need to define monotonicity on vertex sets. Let $G = (X, Y, E)$ be a bipartite graph. An order $<$ on X in G is *increasing* if $x < x'$ implies $N(x) \subseteq N(x')$. Similarly, $<$ on X is *decreasing* if $x < x'$ implies $N(x) \supseteq N(x')$. An order is *monotone* if it is increasing or decreasing. A bipartite graph $G = (X, Y, E)$ is a *chain graph* if there exist monotone orders $<_X, <_Y$ on X, Y respectively [20, 10]. We assume that $<_X$ is decreasing and $<_Y$ is increasing. It is not hard to observe the following.

Proposition 1. *Let $G = (X, Y, E)$ be a connected chain graph with $|X| = n_x$ and $|Y| = n_y$. Then there exist a decreasing order $<_X$ and an increasing order $<_Y$ such that $y_{n_y} \in N(x_i)$ for every $i \in \{1, \ldots, n_x\}$ and $x_1 \in N(y_j)$ for every $j \in \{1, \ldots, n_y\}$, where $x_1 <_X x_2 <_X \cdots <_X x_{n_x}$ and $y_1 <_Y y_2 <_Y \cdots <_Y y_{n_y}$.*

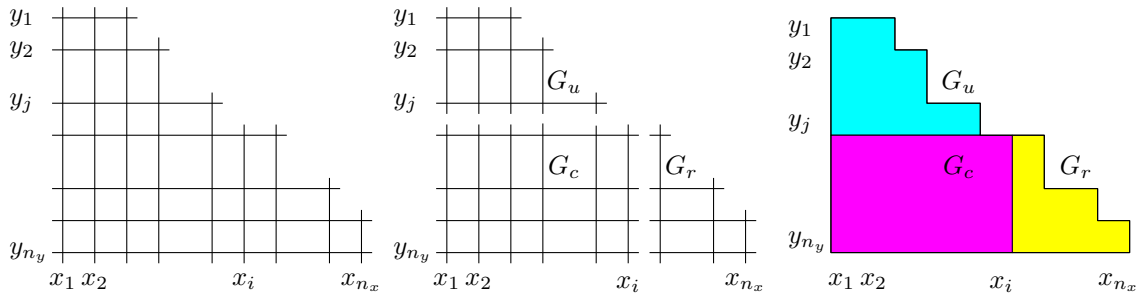

Fig. 2. The intersection model of a chain graph (left), a derived decomposition (middle) and its schematic representation (right).

For a given chain graph, monotone orders on X and Y can be found in linear time (e.g., using a PQ-tree), and from them, the orderings as in Proposition 1 can be computed in linear time (e.g., [17]). Hence, hereafter, we assume that a chain graph is given with two ordered vertex sets stated as in Proposition 1. Without loss of generality, we assume that a chain graph $G = (X, Y, E)$ is connected, and the vertex sets X and Y are ordered as stated in Proposition 1. That is, we have $N(x_i) = \{y_{j_i}, \ldots, y_{n_y}\}$ with $1 \leq i \leq n_x$ and $j_1 \leq j_2 \leq \cdots \leq j_{n_x}$, and $N(y_j) = \{x_1, \ldots, x_{i_j}\}$ with $1 \leq j \leq n_y$ and $i_1 \geq i_2 \geq \cdots \geq i_{n_y}$, where $|X| = n_x$ and $|Y| = n_y$. The main theorem in this section is as follows.

Theorem 1. *Given a connected chain graph $G = (X, Y, E)$, the number of perfect matchings in G can be computed in $O(n^2 \log n)$ time, where $n = |X \cup Y|$.*

We prove Theorem 1 by providing an algorithm. This is based on the following recursive subdivision structure $\mathcal{T}(G)$. The structure $\mathcal{T}(G)$ is a rooted tree, where each node possesses an induced subgraph of G and a node G' is a descendant of a node G'' only if G' is a subgraph of G''.

The structure $\mathcal{T}(G)$ is inspired by an intersection model of a chain graph (Fig. 2 (left)): The vertices x_i in X correspond to vertical line segments I_{x_i} which are located from left to right according to the ordering, and the bottom of the segments are on a horizontal line. The vertices y_j in Y correspond to horizontal line segments J_{y_j} from top to bottom and the left endpoints of the segments are on a vertical line. From Proposition 1, it is easy to see that G is a chain graph if and only if G can be represented by the intersection model of those horizontal line segments and vertical line segments such that I_{x_i} is longer than or equal to $I_{x_{i+1}}$ with $1 \leq i < n_x$ and $J_{y_{j+1}}$ is longer than or equal to J_{y_j} with $1 \leq j < n_y$.

Let $G = (X, Y, E)$ be a chain graph with two with two ordered vertex sets X, Y as in Proposition 1. An edge $e = \{x_i, y_j\} \in E$, $x_i \in X, y_j \in Y$, is *extremal* if $\{x_{i'}, y_j\} \notin E$ for any $i' > i$ or $\{x_i, y_{j'}\} \notin E$ for any $j' < j$. Fix an extremal edge $e = \{x_i, y_j\} \in E$. Then we partition X and Y into X_u, X_r, Y_u, and Y_r as follows; $X_u := \{x_{i'} \mid i' \leq i\}$, $X_r := \{x_{i'} \mid i' > i\}$, $Y_u := \{y_{j'} \mid j' \leq j\}$, and $Y_r := \{y_{j'} \mid j' > j\}$. Using these vertex sets, we define three graphs G_c, G_u, and G_r as follows; $G_c = G[X_u \cup Y_r]$, $G_r = G[X_r \cup Y_r]$, and $G_u = G[X_u \cup Y_u]$. Note that the edge sets of these three graphs form a partition of E, and furthermore, G_u and G_r are connected chain graphs (unless empty), and G_c is a complete bipartite graph. See Fig. 2 (middle, right).

We are now ready for defining $\mathcal{T}(G)$ for a connected chain graph G. The root of $\mathcal{T}(G)$ is G. Let G' be a node of $\mathcal{T}(G)$. If G' is complete bipartite, then G' has no child and hence it is a leaf of $\mathcal{T}(G)$. Otherwise, G' has two children G'_u and G'_r constructed by an appropriate choice of an extremal edge of G'. We call $\mathcal{T}(G)$ a *recursive decomposition* of G.

Let us describe how to choose an appropriate extremal edge when we construct a recursive decomposition. To do this, we look at the depth of each node in $\mathcal{T}(G)$. Namely, the depth of a root node is zero, and if a node has depth d, then its children have depth $d+1$. According to the parity of the depth, we make the choice. If d is even, then we let $k_x = n_x/2$ and $k_y = \min\{j' \mid \{x_i, y_{j'}\} \in E\}$. If d is odd, then we let $k_y = n_y/2$ and $k_x = \max\{i' \mid \{x_{i'}, y_j\} \in E\}$. In both cases, we see that $\{x_{k_x}, y_{k_y}\}$ is

Algorithm 1: $\mathsf{RD}(G, d)$

Input : A connected chain graph $G = (X, Y, E)$ and a non-negative integer d, where $n_x = |X|, n_y = |Y|$;
Output: A recursive decomposition $\mathcal{T}(G)$ of G;

add G as the root;
if G is complete bipartite **then return**;
if d is even **then** set $k_x := n_x/2$ and $k_y := \min\{j \mid \{x_{k_x}, y_j\} \in E\}$;
if d is odd **then** set $k_y := n_y/2$ and $k_x := \max\{i \mid \{x_i, y_{k_y}\} \in E\}$;
find G_u, G_r, G_c from k_x and k_y;
add $\mathsf{RD}(G_u, d{+}1)$ as the left subtree rooted at G;
add $\mathsf{RD}(G_r, d{+}1)$ as the right subtree rooted at G;
return;

an extremal edge of G. Algorithm 1 computes a recursive decomposition $\mathcal{T}(G)$ of G according to this choice of an extremal edge when $\mathsf{RD}(G, 0)$ is called.

Lemma 1. *Let $G = (X, Y, E)$ be a connected chain graph. Then, Algorithm 1 finds a recursive decomposition $\mathcal{T}(G)$ with at most $O(n)$ nodes and height at most $2\log_2 n$ in $O(n)$ time, where $n = |X|{+}|Y|$.*

Proof. The size of $\mathcal{T}(G)$ can easily be seen $O(n)$ since for every vertex v of G there exists exactly one leaf of $\mathcal{T}(G)$ containing v and $\mathcal{T}(G)$ is a rooted binary tree. For the running time, the discussion above implies that the intersection model can be found in $O(n)$ time. Since the connected components can be identified in $O(n)$ time and finding k_y from k_x can be done in $O(1)$ time for each k_x, the overall running time is $O(n)$. So it remains to show that the height is at most $2\log_2 n$. After the algorithm digs into two lower levels, G will be partitioned into four graphs $(G_u)_u$, $(G_u)_r$, $(G_r)_u$, and $(G_r)_r$. Then, we observe that $|X((G_i)_j)| \leq \frac{1}{2}|X(G)|$ and $|Y((G_i)_j)| \leq \frac{1}{2}|Y(G)|$ for each $i, j \in \{u, r\}$. Hence for a given connected chain graph $G = (X, Y, E)$ with $n = |X \cup Y|$ the maximum depth $d_{\max}$ of the algorithm satisfies $2^{2d_{\max}} \leq n$, and hence $d_{\max} \leq 2\log_2 n$. □

To describe the number of matchings, we denote by $\pi(G; a, b)$ the number of perfect matchings in a chain graph $G = (X, Y, E)$ with a vertices from X and b vertices from Y deleted. Namely,

$$\pi(G; a, b) = \left| \left\{ M \subseteq E(G) \;\middle|\; \begin{array}{l} M \text{ is a perfect matching of } G - (A \cup B), \\ \text{where } A \subseteq X, |A| = a, B \subseteq Y, |B| = b \end{array} \right\} \right|.$$

Intuitively, we will count the number of perfect mathings in G such that a vertices in X and b vertices in Y have been matched in the previous level. Since $\pi(G; 0, 0)$ is the number of perfect matchings of G, it suffices to compute $\pi(G; a, b)$ for all possible a and b.

Let us look at how we can decompose $\pi(G; a, b)$ into several independent parts. This gives a fundamental idea for our algorithm.

Lemma 2. *Let $G = (X, Y, E)$ be a connected chain graph. For $0 \leq a < |X|$ and $0 \leq b < |Y|$, it holds that*

$$\pi(G; a, b) = \sum_{a_u, b_r} \pi(G_u; a_u{+}i_c, b_u) \cdot \pi(G_r; a_r, b_r{+}i_c) \cdot \pi(G_c; k_x{-}i_c, n_y{-}k_y{-}i_c) \tag{1}$$

where the sum is taken over the ranges $0 \leq a_u \leq \min\{a, k_x\}$ and $0 \leq b_r \leq \min\{b, n_y{-}k_y\}$, and other symbols are defined as $b_u = b{-}b_r$, $a_r = a{-}a_u$, and $i_c = k_x{-}k_y{-}a_u{+}b{-}b_r$.

Proof. We first show the inequality "the left-hand side $\leq$ the right-hand side." Every matching M counted in the left-hand side can be partitioned into three parts $M = M_u \cup M_r \cup M_c$, where M_u, M_r, M_c is a matching of G_u, G_r, G_c, respectively. Since M is a perfect matching of $G{-}(A \cup B)$

Algorithm 2: $\#\mathsf{M}(G, a, b)$

Input : A connected chain graph $G = (X, Y, E)$ and two integers a, b together with a recursive decomposition $\mathcal{T}(G)$;
Output: $\pi(G; a, b)$;

if $n_x - a \neq n_y - b$ **then return** 0;
else if G is complete bipartite **then return** $(n_x - a)!$;
else
  $sum := 0$; $a_r := a - a_u$; $b_u = b - b_r$; $i_c = k_x - k_y - a_u + b - b_r$;
  foreach $a_u = 0, 1, \ldots, \min\{a, k_x\}$ and $b_r = 0, 1, \ldots, \min\{b, n_y - k_y\}$ **do**
    $sum := sum + i_c! \cdot \binom{k_x}{i_c} \cdot \binom{n_y - k_y}{i_c} \cdot \#\mathsf{M}(G_u, a_u + i_c, b_u) \cdot \#\mathsf{M}(G_r, a_r, b_r + i_c)$;
  end
  return sum.
end

for some $A \subseteq X, B \subseteq Y$ with $|A| = a, |B| = b$, it holds that $|M| = n_x - a = n_y - b$. Let $|M_u| = i_u, |M_r| = i_r, |M_c| = i_c$. Let $A_c \subseteq X_u$ and $B_c \subseteq Y_r$ be the set of vertices matched by M_c. Note that $|A_c| = i_c = |B_c|$. Then, M_u is a perfect matching of $G_u - (A_u \cup A_c \cup B_u)$ for some $A_u \subseteq X_u \setminus A_c$ and $B_u \subseteq Y_u$, and M_r is a perfect matching of $G_r - (A_r \cup B_r \cup B_c)$ for some $A_r \subseteq X_r$ and $B_r \subseteq Y_r \setminus B_c$. It is important to observe that such A_u, B_u, A_r, B_r are unique. For example, A_u is determined as the set of vertices in $X_u \setminus A_c$ that are not matched by M_u. Therefore, if we let $|A_u| = a_u$ and $|B_r| = b_r$, then M_u is counted exactly once in $\pi(G_u; a_u + i_c, b - b_r)$ and similarly M_r is counted exactly once in $\pi(G_r; a - a_u, b_u + i_c)$. Then, we see that $k_x - (a_u + i_c) = i_u = k_y - (b - b_r)$ and $n_x - k_x - (a - a_u) = i_r = n_y - k_y - (b_u + i_c)$, and therefore, $i_c = k_x - k_y - a_u + b - b_r$. Then, it suffices to note that M_c is counted exactly once in $\pi(G_c; k_x - i_c, n_y - k_y - i_c)$.

Conversely, We show the inequality "the left-hand side $\geq$ the right-hand side." Let M_u and M_r be perfect matchings counted in $\pi(G_u; a_u + i_c, b_u)$ and $\pi(G_r; a_r, b_r + i_c)$ respectively, for some a_u, b_r in the appropriate ranges and a_r, b_u, i_c as defined in the statement of the lemma. Specifically, let M_u be a perfect matching of $G_u - (A_u \cup A_c \cup B_u)$ for some $A_u \cup A_c \subseteq X_u$ and $B_u \subseteq Y_u$, and M_r is a perfect matching of $G_r - (A_r \cup B_r \cup B_c)$ for some $A_r \subseteq X_r$ and $B_r \cup B_c \subseteq Y_r$, where $|A_u| = a_u, |A_r| = a_r, |B_u| = b_u, |B_r| = b_r, |A_c| = |B_c| = i_c$. Consider constructing a perfect matching M of $G - (A \cup B)$ for some A, B with $|A| = a, |B| = b$ as $M = M_u \cup M_r \cup M_c$ with some matching M_c of G_c. Then, such M_c should be a perfect matching of $G_c - ((X_u - A_c) \cup (Y_r - B_r))$. This completes the proof. □

Since G_c is a complete bipartite graph, it is not difficult to see that $\pi(G_c; k_x - i_c, n_y - k_y - i_c) = i_c! \binom{k_x}{i_c} \binom{n_y - k_y}{i_c}$. Hence Lemma 2 readily gives Algorithm 2. To compute the number of perfect matchings in a given chain graph G, we first call $\#\mathsf{M}(G, 0, 0)$.

Proof (of Theorem 1). Algorithm 2 can be implemented by dynamic programming on $\mathcal{T}(G)$. For each node G' of $\mathcal{T}(G)$, we store the values returned by calls $\#\mathsf{M}(G', a, b)$ for all possible a and b. If the depth of G' is d, then the number of such possibilities is at most $n_x / 2^{\lceil d/2 \rceil} \cdot n_y / 2^{\lfloor d/2 \rfloor} \leq n_x n_y / 2^d$. Therefore, the number of values stored for each node of $\mathcal{T}(G)$ is $O(n^2 / 2^d)$. At the call to $\#\mathsf{M}(G', a, b)$ we need to look up at most $n_x n_y / 2^d$ values. Note that the values of factorials and binomial coefficients can be computed beforehand and stored as well in $O(n^2)$ time and space. Since the number of nodes at depth d is at most 2^d, the overall running time is at most $\sum_{d=0}^{2 \log_2 n} 2^d O(n^2 / 2^d) = O(n^2 \log n)$. The space requirement is also $O(n^2 \log n)$. □

Note that if $n_x - a = n_y - b$, then $\pi(G; a, b)$ is the number of matchings of size $n_x - a$. Hence, Algorithm 2 computes the number of matchings of each possible size. This implies the following corollary.

Corollary 1. *Given a chain graph $G = (X, Y, E)$, the number of matchings and the number of matchings of fixed size in G can be computed in $O(n^2 \log n)$ time, where $n = |X \cup Y|$.*

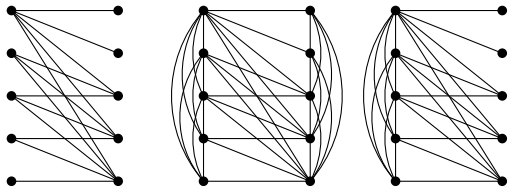

Fig. 3. Comparison of three graph classes. (Left) A chain graph. (Center) A cochain graph. (Right) A threshold graph.

4 Polynomial-time algorithm for cochain graphs and threshold graphs

Similarity among chain graphs, cochain graphs and threshold graphs allows us to provide polynomial-time algorithms to count the number of perfect matchings in cochain graphs and threshold graphs.

A *cochain graph* is simply defined as the complement of a chain graph. From Proposition 1, we can immediately see that a cochain graph has the following property.

Proposition 2. *Let $G = (V, E)$ be a cochain graph and $\overline{G} = (X, Y, \overline{E})$ be a chain graph that is the complement of G. with $|X| = n_x$ and $|Y| = n_y$. Then X and Y are cliques of G, and there exist a decreasing order $<_X$ and an increasing order $<_Y$ such that $y_{n_y} \in N(x_i) \setminus X$ for every $i \in \{1, \ldots, n_x\}$ and $x_1 \in N(y_j) \setminus Y$ for every $j \in \{1, \ldots, n_y\}$, where $x_1 <_X x_2 <_X \cdots <_X x_{n_x}$ and $y_1 <_Y y_2 <_Y \cdots <_Y y_{n_y}$.* □

Namely, a cochain graph can be constructed from a chain graph by filling up both of the color classes to cliques. See Fig. 3.

A graph $G = (V, E)$ is a *threshold* graph if there exist a weight assignment $w \colon V \to \mathbb{R}$ such that $\{u, v\} \in E$ if and only if $w(u) + w(v) > 0$. The following is a well-known property (or actually a characterization) of thereshold graphs.

Proposition 3 (Chvátal and Hammer [2]). *For a threshold graph $G = (V, E)$, a partition $\{X, Y\}$ of V with the following properties can be found in $O(|V|+|E|)$ time. First, $X = \{x_1, \ldots, x_{n_x}\}$ is a clique of G, $Y = \{y_1, \ldots, y_{n_y}\}$ is an independent set of G, and there exist a decreasing order $<_X$ and an increasing order $<_Y$ such that $y_{n_y} \in N(x_i) \setminus X$ for every $i \in \{1, \ldots, n_x\}$ and $x_1 \in N(y_j)$ for every $j \in \{1, \ldots, n_y\}$, where $x_1 <_X x_2 <_X \cdots <_X x_{n_x}$ and $y_1 <_Y y_2 <_Y \cdots <_Y y_{n_y}$.*

Namely, a threshold graph can be constructed from a chain graph by filling up one of the color classes to a clique. See Fig. 3.

Since a cochain graph and a threshold graph possess a structure similar to a chain graph, we may define a recursive decomposition for them analogously. An important difference is that G_c is not an induced subgraph of G, but G_c will be a subgraph of G with vertex set $X_u \cup Y_r$ and edge set consisting of those edges between X_u and Y_r. Namely, G_c is complete bipartite. Then, the equation similar to (1) holds. The whole arguments are verbatim. Hence we obtain the following theorem.

Theorem 2. *The number of perfect matchings in a cochain graph and a threshold graph can be computed in $O(n^2 \log n)$ time.* □

5 Hardness results

In this section, we prove the #P-completeness of counting the perfect matchings in split graphs and chordal bipartite graphs. The #P-completeness for split graphs immediately implies that for chordal graphs.

5.1 Hardness for split graphs

A graph is a *split graph* if the vertex set can be partitioned into two parts such that one part is a clique and the other is an independent set. In other words, a split graph is constructed from a bipartite graph by filling up one color class to a clique. A graph is *chordal* if every induced cycle has length three. It is easy to see that every split graph is chordal.

Theorem 3. *Counting the number of perfect matchings in a split graph is* #P*-complete.*

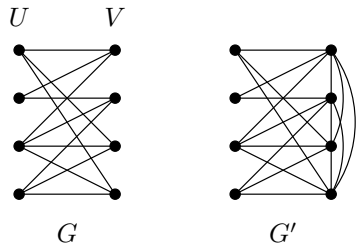

Fig. 4. Reduction for Theorem 3.

Proof. We use a reduction from the problem to count the number of perfect matchings in a bipartite graph, which is known to be #P-complete [18].

Let $G = (U, V, E)$ be a bipartite graph with $|U| = |V| = n$. We construct another graph $G' = (V', E')$ out of G by making V a clique; that is, $V' := U \cup V$ and $E' := E \cup \binom{V}{2}$. Fig. 4 illustrates the construction. We can see that G' is a split graph.

Now, we claim that an edge subset $M \subseteq E'$ of G' is a perfect matching of G' if and only if M is a perfect matching of G. (This will imply that the numbers of perfect matchings in G and G' are the same. So, the reduction is persimonous.) Since G is a subgraph of G' with the same vertex set, when M is a perfect matching of G it is also a perfect matching of G'. Conversely, assume that M is a perfect matching of G'. Since every vertex in U is matched through M with some vertex in V and $|U| = |V|$, M only uses edges from G. Therefore, M is a perfect matching of G. □

From Theorem 3, we can immediately see that counting the number of maximum matchings in a split graph is #P-complete.

By utilizing an interpolation technique, we are able to show the following.

Theorem 4. *Counting the number of matchings in a split graph is* #P*-complete.*

Proof. For our reduction, we again use the problem to count the number of perfect matchings in a bipartite graph, which is known to be #P-complete [18].

Let $G = (U, V, E)$ be a bipartite graph with $|U| = |V| = n$. We construct a graph $G_i = (V_i, E_i)$ for every $i \in \{1, \ldots, n+1\}$ out of G as follows. Let $V := \{v_1, \ldots, v_n\}$. For each vertex $v_\ell \in V$, we use a set $V_i^{(\ell)}$ of i vertices for G_i where $V_i^{(1)}, \ldots, V_i^{(n)}$ are all disjoint. We set $V_i := U \cup V \cup \bigcup_{\ell=1}^{n} V_i^{(\ell)}$, and $E_i := E \cup \binom{V}{2} \cup \bigcup_{\ell=1}^{n} F_i^{(\ell)}$, where $F_i^{\ell} := \{\{v_\ell, v\} \mid v \in V_i^{(\ell)}\}$ for every $\ell \in \{1, \ldots, n\}$. Fig. 5 illustrates the construction.

For a graph H with n vertices, remember that we denote by $\mu_j(H)$ the number of matchings in H of size j, by $\mu(H)$ the number of matchings in H. Furthermore, let $F_i := \bigcup_{\ell=1}^{n} F_i^{(\ell)}$.

Let us consider $\mu(G_i)$. Each matching M of G_i potentially uses some edges from E and some edges from F_i. Let M use j edges from E and k edges from F_i. Consider constructing M by first choosing j edges from E, then k edges from F_i, and finally the rest of edges from $\binom{V}{2}$. Since M is a matching,

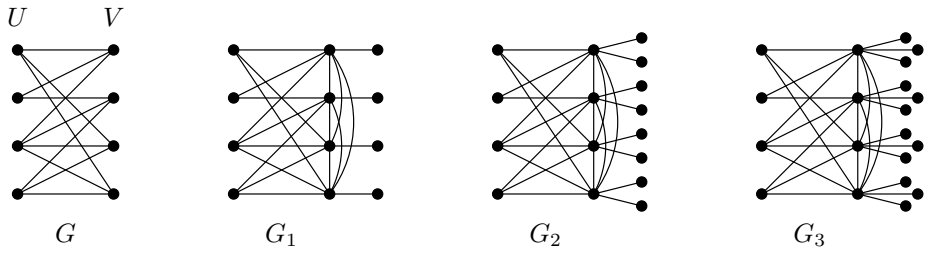

Fig. 5. Construction in the proof of Theorem 4.

we have $\mu_j(G)$ ways to choose j edges from E for M. Then j vertices in V are already matched, so there are $n-j$ vertices left unmatched in V. Therefore, the number of ways to choose k edges from F_i for M is $\binom{n-j}{k} i^k$. Then there are $n-j-k$ vertices left unmatched in V. Among them we choose some edges for M. Therefore, the number of choices is $\mu(K_{n-j-k})$, where K_{n-j-k} is a complete graph with $n-j-k$ vertices. This way, we obtain the following formula: for every $i \in \{1,\dots,n+1\}$ it holds that $\mu(G_i) = \sum_{j=0}^{n} \mu_j(G) \left(\sum_{k=0}^{n-j} \left(\binom{n-j}{k} i^k \right) \mu(K_{n-j-k}) \right)$. In a matrix form, this can be written as

$$\begin{bmatrix} \mu(G_1) \\ \mu(G_2) \\ \vdots \\ \mu(G_{n+1}) \end{bmatrix} = A \begin{bmatrix} \mu_0(G) \\ \mu_1(G) \\ \vdots \\ \mu_n(G) \end{bmatrix},$$

where A is a matrix with row index set $\{1,\dots,n+1\}$ and column index set $\{0,\dots,n\}$ defined as $A_{i,j} := \sum_{k=0}^{n-j} \left(\binom{n-j}{k} i^k \right) \mu(K_{n-j-k})$ for every row index $i \in \{1,\dots,n+1\}$ and every column index $j \in \{0,\dots,n\}$.

Claim. The matrix A defined above is non-singular (namely, the inverse of A exists).

We would like to notice that the claim finishes the proof of the theorem. If we are able to know $\mu(G_i)$ for every $i \in \{1,\dots,n+1\}$, then by computing the inverse of A, we are also able to know $\mu_j(G)$ for all $j \in \{0,\dots,n\}$. (Note that each entry of A can be computed efficiently since $\mu(K_m) = \sum_{j=0}^{\lfloor m/2 \rfloor} \mu_j(K_m) = \sum_{j=0}^{\lfloor m/2 \rfloor} \binom{m}{2j} \frac{(2j)!}{j! 2^j}$ holds.) In particular we obtain $\mu_n(G)$, the number of perfect matchings in G. This completes the reduction.

Therefore, it suffices to prove the claim. To do that, first observe that for each row index $i \in \{1,\dots,n+1\}$ and each column index $j \in \{0,\dots,n\}$ the i,j-entry $A_{i,j}$ can be written as $A_{i,j} = \sum_{k=0}^{n-j} i^k \left(\binom{n-j}{k} \mu(K_{n-j-k}) \right)$. Let B be a matrix with row index set $\{1,\dots,n+1\}$ and column index set $\{0,\dots,n\}$ defined as $B_{i,k} := i^k$ for each $i \in \{1,\dots,n+1\}$ and $k \in \{0,\dots,n\}$, and C be a matrix with row index set $\{0,\dots,n\}$ and column index set $\{0,\dots,n\}$ defined as

$$C_{k,j} := \begin{cases} \binom{n-j}{k} \mu(K_{n-j-k}) & \text{if } 0 \le k \le n-j, \\ 0 & \text{otherwise,} \end{cases}$$

for each $k \in \{0,\dots,n\}$ and $j \in \{0,\dots,n\}$. Then, we can see that for every $i \in \{1,\dots,n+1\}$ and $j \in \{0,\dots,n\}$ it holds that $A_{i,j} = \sum_{k=0}^{n} B_{i,k} C_{k,j}$. In other words, $A = BC$ as a matrix. The matrix B is a famous Vandermonde matrix, which is known to be non-singular. How about the non-singularity of C? Since $\binom{n-j}{k} \mu(K_{n-j-k}) \neq 0$ when $0 \le k \le n-j$, the upper-left half of C is occupied with non-zero entries, and the lower-right half of C is occupied with zero entries. So, the matrix C is also non-singular. (See Fig. 6 for the situation.) Thus, A is non-singular. (In particular, $A^{-1} = (BC)^{-1} = C^{-1}B^{-1}$.) $\square$

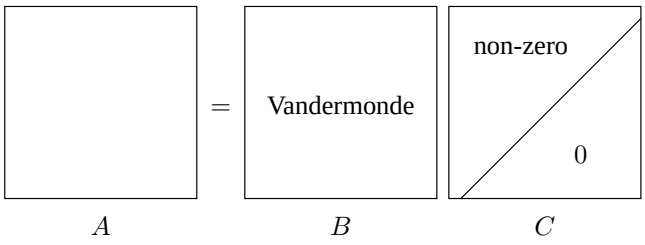

Fig. 6. The matrices in the proof of Theorem 4.

A modification of the proof of Theorem 4 shows the following. The proof is postponed to Appendix.

Theorem 5. *Counting the number of maximal matchings in a split graph is* #P*-complete.* □

5.2 Hardness for chordal biparitite graphs

Next, we switch to chordal bipartite graphs. A bipartite graph is *chordal bipartite* if every induced cycle is of length four.

The #P-completeness for chordal bipartite graphs will be proven via the interpolation technique.

Theorem 6. *The problem to count the number of perfect matchings in a chordal bipartite graph is* #P*-complete.*

Proof. We use a reduction from the problem to count the number of perfect matchings in a bipartite graph, which is known to be #P-complete [18].

Given a bipartite graph $G = (X, Y, E)$ with $|X| = |Y| = n$, we construct the following chordal bipartite graph $cb_i(G)$ for each $i \in \{1, \ldots, n+1\}$. The vertex set of $cb_i(G)$ is defined as $V(G) = X \cup Y \cup \{p_{j,v,e} \mid 1 \leq j \leq i, v \in X \cup Y, e \in E\} \cup \{q_{j,v,e} \mid 1 \leq j \leq i, v \in X \cup Y, e \in E\}$. The edge set of $cb_i(G)$ is defined as $E(G) = \{\{x, y\} \mid x \in X, y \in Y\} \cup \{\{x, p_{j,x,e}\} \mid x \in X, e \in E, x \in e, 1 \leq j \leq i\} \cup \{\{y, q_{j,y,e}\} \mid y \in Y, e \in E, y \in e, 1 \leq j \leq i\} \cup \{\{p_{j,x,e}, q_{j,y,e}\} \mid x \in X, y \in Y, e = \{x, y\} \in E\}$. Namely, to construct $cb_i(G)$ from G, we replace each edge of G by i paths of length three, and join the vertices of X and Y by edges to make them complete bipartite. It is not difficult to see that $cb_i(G)$ is chordal bipartite. Fig. 7 shows an example.

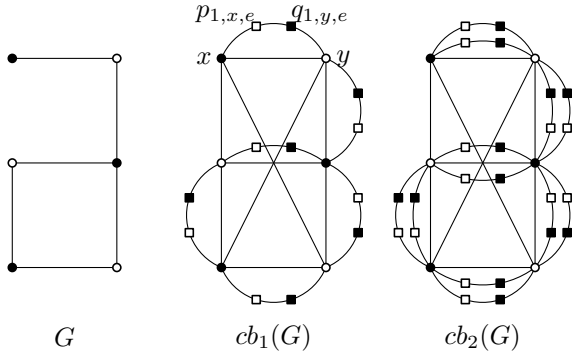

Fig. 7. Hardness for chordal bipartite graphs.

Consider a perfect matching M of $cb_i(G)$. We map M to a matching M' of G if and only if the following conditions are satisfied.

– When $e = \{x, y\} \notin M'$, it holds that $\{p_{j,x,e}, q_{j,y,e}\} \in M$ for all $j \in \{1, \ldots, i\}$.

– When $e = \{x, y\} \in M'$, it holds that $\{x, p_{j,x,e}\}, \{y, q_{j,y,e}\} \in M$ for exactly one $j \in \{1, \ldots, i\}$. (Then, it must hold that $\{p_{j,x,e}, q_{j,y,e}\} \in M$ for all other $j \in \{1, \ldots, i\}$.)

There are several perfect matchings M that corresponds to M'. We can count the number of such matchings M from M'. In the second condition, we have i choices for each edge of M'. Let $|M'| = k$. Then, this gives rise do i^k choices. Moreover, there are $n-k$ vertices in both X and Y that do not appear in these conditions, and they are supposed to be matched. Since these vertices induce a complete bipartite subgraph of G, the number of ways to match them is exactly $(n-k)!$.

In this way, we obtain $\pi(cb_i(G)) = \sum_{k=0}^{n} \mu_k(G) i^k (n-k)!$ for each $i \in \{1, \ldots, n+1\}$, where $\pi(cb_i(G))$ means the number of perfect matchings in $cb_i(G)$. In the matrix form, this can be written as

$$\begin{bmatrix} \pi(cb_1(G)) \\ \pi(cb_2(G)) \\ \vdots \\ \pi(cb_{n+1}(G)) \end{bmatrix} = A \begin{bmatrix} \mu_0(G) \\ \mu_1(G) \\ \vdots \\ \mu_n(G) \end{bmatrix},$$

where A is a matrix with row index set $\{1, \ldots, n+1\}$ and column index set $\{0, \ldots, n\}$ defined as $A_{i,k} := i^k (n-k)!$ for every row index $i \in \{1, \ldots, n+1\}$ and every column index $k \in \{0, \ldots, n\}$. We can see that the determinant of A is the determinant of a non-singular Vandermonde matrix times $\prod_{k=0}^{n} (n-k)!$, thus non-zero. Therefore, from the equality above, we can recover $\mu_n(G)$, the number of perfect matchings of the given bipartite graph G, in polynomial time. □

Again, more elaboration proves the #P-completeness of counting the matchings and the maximal matchings in chordal bipartite graphs. See Appendix.

References

1. S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. Journal of Algorithms **12** (1991) 308–340.
2. V. Chvátal and P.L. Hammer. Set-packing and threshold graphs. Research Report, Computer Science Department, University of Waterloo, CORR 73-21, 1973.
3. P. Dagum and M. Luby. Approximating the permanent of graphs with large factors. Theoretical Computer Science **102** (1992) 283–305.
4. D. Ding. Covering the edges with consecutive sets. Journal of Graph Theory **15** (1991) 559–562.
5. M.E. Fisher. Statistical mechanics of dimers on a plane lattice. Physical Review Series 2 **124** (1961) 1664–1672.
6. H. Frost, M. Jacobson, J. Kabell, and F.R. Morris. Bipartite analogues of split graphs and related topics. Ars Combinatoria **29** (1990) 283–288.
7. A. Galluccio and M. Loebl. On the theory of Pfaffian orientations I. Perfect matchings and permanents. Electronic Journal of Combinatorics **6** (1999) Research Paper 6.
8. P.L. Hammer, U.N. Peled, and X. Sun. Difference graphs. Discrete Applied Mathematics **28** (1990) 35–44.
9. P.W. Kasteleyn. Dimer statistics and phase transitions. Journal of Mathematical Physics **4** (1963) 287–293.
10. T. Kloks, D. Kratsch, and H. Müller. Bandwidth of chain graphs. Information Processing Letters **68** (1998) 313–315.
11. M.-S. Lin. Fast and simple algorithms to count the number of vertex covers in an interval graph. Information Processing Letters **102** (2007) 143–146.
12. M.-S. Lin and Y.-J. Chen. Linear time algorithms for counting the number of minimal vertex covers with minimum/maximum size in an interval graph. Information Processing Letters **107** (2008) 257–264.
13. Y. Okamoto, T. Uno, and R. Uehara. Counting the independent sets in a chordal graph. Journal of Discrete Algorithms **6** (2008) 229–242.
14. H.N.V. Temperley and M.E. Fisher. Dimer problem in statistical mechanics — an exact result. Philosophical Magazine, Series 8 **6** (1961) 1061–1063.
15. G. Tesler. Matchings in graphs on non-orientable surfaces. Journal of Combinatorial Theory, Series B **78** (2000) 198–231.
16. S.P. Vadhan. The complexity of counting in sparse, regular, and planar graphs. SIAM Journal on Computing **31** (2001) 398–427.
17. R. Uehara and Y. Uno. On computing longest paths in small graph classes. International Journal of Foundations of Computer Science **18** (2007) 911–930.
18. L.G. Valiant. The complexity of computing the permanent. Theoretical Computer Science **8** (1979) 189–201.
19. L.G. Valiant. The complexity of enumeration and reliability problems. SIAM Journal on Computing **8** (1979) 410–421.
20. M. Yannakakis. Node-deletion problems on bipartite graphs. SIAM Journal on Computing **10** (1981) 310–327.

A Postponed hardness proofs

Proof (of Theorem 5). We use the same reduction as in the proof of Theorem 4. Denote by $\tilde{\mu}(H)$ the number of maximal matchings in a graph H. Then, for G and the G_i's in the proof of Theorem 4 it holds that

$$\tilde{\mu}(G_i) = \sum_{j=0}^{n} \mu_j(G) \left(\sum_{\substack{k=0, \\ n-j-k \text{ even}}}^{n-j} \left(\binom{n-j}{k} i^k \right) \pi(K_{n-j-k}) \right)$$

for every $i \in \{1, \ldots, n+1\}$. We can show that the corresponding coefficient matrix is non-singular in the same way. □

Next we will prove the #P-completeness of counting the matchings and the maximal matchings of chordal bipartite graphs. This will be done via the interpolation technique again.

Theorem 7. *Counting the number of matchings in chordal bipartite graphs is* #P*-complete.*

Proof. We reduce the problem to count the number of perfect matchings in chordal bipartite graphs. The proof uses a reduction similar to the proof of Theorem 4.

Let $G = (U, V, E)$ be a given chordal bipartite graph with $|U| = |V| = n$. For each $i \in \{1, \ldots, n+1\}$, we construct a chordal bipartite graph $G_i = (U_i, V_i, E_i)$ as follows. Let $V = \{v_1, \ldots, v_n\}$. For each vertex $v_\ell \in V$, we use a set $V_i^{(\ell)}$ of i vertices for G_i where $V_i^{(1)}, \ldots, V_i^{(n)}$ are all disjoint. Let $U_i = U \cup \bigcup_{\ell=1}^{n} V_i^{(\ell)}$, $V_i = V$, and $E_i := E \cup \bigcup_{\ell=1}^{n} F_i^{(\ell)}$, where $F_i^{\ell} := \{\{v_\ell, v\} \mid v \in V_i^{(\ell)}\}$ for every $\ell \in \{1, \ldots, n\}$. Then, we can see that G_i is a chordal bipartite graph for every $i \in \{1, \ldots, n+1\}$.

It holds that $\mu(G_i) = \sum_{j=0}^{n} \mu_j(G) \sum_{k=0}^{n-j} \binom{n-j}{k} i^k$. for every $i \in \{1, \ldots, n+1\}$. In a matrix form, this can be written as

$$\begin{bmatrix} \mu(G_1) \\ \mu(G_2) \\ \vdots \\ \mu(G_{n+1}) \end{bmatrix} = A \begin{bmatrix} \mu_0(G) \\ \mu_1(G) \\ \vdots \\ \mu_n(G) \end{bmatrix},$$

where A is a matrix with row index set $\{1, \ldots, n+1\}$ and column index set $\{0, \ldots, n\}$ defined as $A_{i,j} := \sum_{k=0}^{n-j} \binom{n-j}{k} i^k$ for every row index $i \in \{1, \ldots, n+1\}$ and every column index $j \in \{0, \ldots, n\}$. Then, by the same argument as in the proof of Theorem 4, we can conclude that A is non-singular, and thus $\mu_n(G) = \pi(G)$ can be recovered in polynomial time. □

Theorem 8. *Counting the number of maximal matchings in chordal bipartite graphs is* #P*-complete.*

Proof. We use the same reduction as the proof of Theorem 7. For the constructed graph G_i from a given chordal bipartite graph G, let $\tilde{\mu}(G_i)$ be the number of maximal matchings in G_i. Then, it holds that

$$\tilde{\mu}(G_i) = \sum_{j=0}^{n} \mu_j(G) i^{n-j}.$$

We can see that the corresponding coefficient matrix is Vandermonde, thus non-singular. □