Random Generation and Enumeration of Bipartite Permutation Graphs☆

Toshiki Saitoh[a], Yota Otachi[b], Katsuhisa Yamanaka[c], Ryuhei Uehara[a]

[a]School of Information Science, JAIST, Asahidai 1-1, Nomi, Ishikawa 923-1292, Japan
[b]Department of Computer Science, Gunma University, 1-5-1 Tenjin-cho, Kiryu, Gunma 376-8515, Japan
[c]Graduate School of Information Systems, The University of Electro-Communications, Chofugaoka 1-5-1, Chofu, Tokyo 182-8585, Japan

Abstract

Connected bipartite permutation graphs without vertex labels are investigated. First, the number of connected bipartite permutation graphs of n vertices is given. Based on the number, a simple algorithm that generates a connected bipartite permutation graph uniformly at random up to isomorphism is presented. Finally an enumeration algorithm of connected bipartite permutation graphs is proposed. The algorithm is based on reverse search, and it outputs each connected bipartite permutation graph in O(1) time.

Keywords: Bipartite permutation graph, counting, Dyck path, enumeration, Motzkin path, random generation.

1. Introduction

Recently we have to process huge amounts of data in the area of data mining, bioinformatics, etc. In most cases, we have to use some certain structure to solve problems efficiently. We need three efficiencies to deal with the complex structure; it has to be represented efficiently, essentially different instances have to be enumerated efficiently, and its properties have to be checked efficiently. From the viewpoint of graph classes, the previously studied structures are relatively primitive. Although trees are widely investigated as a model of such structured data [6, 11, 13, 15], there are few results for more complex graph classes. Recently, distance-hereditary graphs [12] and proper interval graphs [18] are investigated from this viewpoint.

In this paper, we investigate counting, random generation, and enumeration of a graph class called *bipartite permutation graphs*. More precisely, we aim to count, generate, and enumerate unlabeled connected bipartite permutation graphs. From the practical point of view, "unlabeled" and "connected" are reasonable properties to avoid redundancy. On the other hand, however, they are also challenges to developing efficient algorithms. Especially, unlabeled property requires us to avoid generating isomorphic graphs. In other words, we have to recognize isomorphic graphs

☆A preliminary version of this article was presented at ISAAC 2009 [17].

Email addresses: toshikis@jaist.ac.jp (Toshiki Saitoh), otachi@comp.cs.gunma-u.ac.jp (Yota Otachi), yamanaka@is.uec.ac.jp (Katsuhisa Yamanaka), uehara@jaist.ac.jp (Ryuhei Uehara)

and suppress generating/counting/enumerating them twice or more. Roughly speaking, the graph isomorphism problem has to be solved efficiently for our target graph classes in this context. The graph isomorphism problem is one of well-known basic problems, and it is still hard on restricted graph classes [22]. There are two well-known graph classes that the graph isomorphism problem can be solved in polynomial time; interval graphs [14] and permutation graphs [3]. Hence, they are the final goal in this framework. We mention that these graph classes have been widely investigated since they are very basic graph classes from the viewpoint of graph theory. Therefore many useful properties have been revealed, and many efficient algorithms have been developed for them (see, e.g., [2, 7, 19]). From the practical point of view, when an efficient algorithm for a graph class is developed and implemented, we need many graphs belonging to the class to check the reliability of the algorithm. Hence, for such popular graph classes, efficient random generator and enumerator are required. On the other hand, the counting of such graphs is rather mathematical. From the viewpoint of combinatorics, the counting of graphs having a certain structure is an important issue. In combinatorics, the notion of Dyck path is one of basic tools, and it appears in a number of areas [20, 21]. One natural extension of the notion of Dyck path is known as Motzkin path; while a Dyck path is a sequence of +1 and −1, a Motzkin path is a sequence of +1, −1, and 0. We will show that an unlabeled connected bipartite permutation graph is strongly related to an extension of a Motzkin path, which is known as a 2-Motzkin path [5], that consists of +1, −1, +0, and −0. Our counting result also gives a new insight of this area.

Saitoh et al. have obtained such results for proper interval graphs which form a subclass of interval graphs [18]. We turn to bipartite permutation graphs that form a subclass of permutation graphs, and show the similar results for them. As we will see, bipartite permutation graphs have a certain structure, which can be seen as a generalization of the structure appearing in proper interval graphs implicitly. That is, developing some new nontrivial techniques based on the results in proper interval graphs, we advance the results in [18] to bipartite permutation graphs.

2. Preliminaries

Interval graph: A graph $G = (V, E)$ with $V = \{v_1, v_2, \dots, v_n\}$ is an *interval graph* if there is a finite set of intervals $\mathcal{I} = \{I_{v_1}, I_{v_2}, \dots, I_{v_n}\}$ on the real line such that $\{v_i, v_j\} \in E$ if and only if $I_{v_i} \cap I_{v_j} \neq \emptyset$ for each i and j with $0 < i, j \leq n$. We call the interval set $\mathcal{I}$ an *interval representation* of G. For each interval I, we denote by $L(I)$ and $R(I)$ the left and right endpoints of the interval, respectively. An interval representation is *proper* if no two distinct intervals I and J exist such that I properly contains J or vice versa. An interval graph is *proper* if it has a proper interval representation. If an interval graph G has an interval representation $\mathcal{I}$ such that every interval in $\mathcal{I}$ has the same length, G is said to be a *unit interval graph*. Such interval representation is called a *unit interval representation*. It is well-known that proper interval graphs coincide with unit interval graphs [16]. That is, given a proper interval representation, we can transform it to a unit interval representation. A simple constructive way of the transformation can be found in [1]. We can assume without loss of generality that $L(I) \neq L(J)$ (and hence $R(I) \neq R(J)$), and $R(I) \neq L(J)$ for any two distinct intervals I and J in a unit interval representation $\mathcal{I}$.

Let Σ be an alphabet {'[', ']'}. We encode a unit interval representation $\mathcal{I}$ of a unit interval graph G by a string $s(\mathcal{I})$ in Σ^* as follows; we sweep the interval representation from left to right,

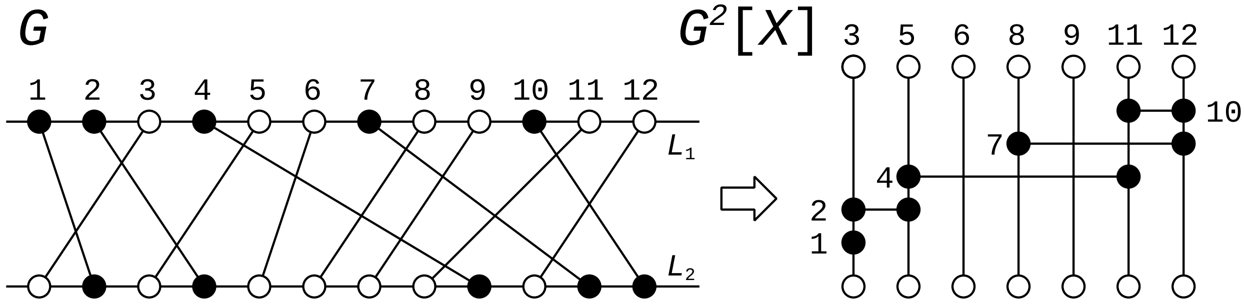

Figure 1: Proper interval graph from bipartite permutation graph

and for each $I \in \mathcal{I}$ encode $L(I)$ and $R(I)$ by '[' and ']', respectively. We call the encoded string a *string representation* of G. We say that string x in Σ^* is *balanced* if the number of '['s in x equals that of ']'s. Clearly $s(\mathcal{I})$ is a balanced string of $2n$ letters. Using the construction in [1], $s(\mathcal{I})$ can be constructed from a proper interval representation $\mathcal{I}$ in $O(n)$ time and vice versa since the ith '[' and the ith ']' give the left and right endpoints of the ith interval, respectively. (We assume that each interval representation is given by a list of the endpoints of intervals from left to right.)

We define '$\bar{[}$' = ']' and '$\bar{]}$' = '[' respectively. For two strings $x = x_1x_2\cdots x_n$ and $y = y_1y_2\cdots y_m$ in Σ^*, we say that x is *smaller* than y if (1) $n < m$, or (2) $n = m$ and there exists an index $i \in \{1,\ldots,n\}$ such that $x_{i'} = y_{i'}$ for all $i' < i$ and x_i = '[' and y_i = ']'. If x is smaller than y, we denote $x < y$. (This is so called "lexicographical order with length preferred.") For a string $x = x_1x_2\cdots x_n$ we define the *reverse* $\bar{x}$ of x by $\bar{x} = \bar{x}_n\bar{x}_{n-1}\cdots\bar{x}_1$. A string x is *reversible* if $x = \bar{x}$. A connected proper interval graph G is said to be *reversible* if its string representation is reversible.

Lemma 1 (See, e.g., [4, Corollary 2.5])**.** *Let G be a connected proper interval graph, and $\mathcal{I}$ and $\mathcal{I}'$ be any two unit interval representations of G. Then either $s(\mathcal{I}) = s(\mathcal{I}')$ or $s(\mathcal{I}) = \overline{s(\mathcal{I}')}$ holds. That is, the string representation of a proper interval graph is unique up to isomorphism.*

Permutation graph: A graph $G = (V, E)$ with $V = \{1, 2, \ldots, n\}$ is said to be a *permutation graph* if there is a permutation π over V such that $\{i, j\} \in E$ if and only if $(i - j)(\pi(i) - \pi(j)) < 0$. Intuitively, each vertex i in a permutation graph corresponds to a line ℓ_i joining two endpoints on two parallel lines L_1 and L_2. Then two vertices i and j are adjacent if and only if the corresponding lines ℓ_i and ℓ_j intersect. The ordering of vertices gives the ordering of the endpoints on L_1, and the ordering by permutation π over V gives the ordering of the endpoints on L_2. We call the intersection model a *line representation* of the permutation graph. For two line representations $\mathcal{L}$ and $\mathcal{L}'$, suppose $\mathcal{L}$ contains (i, j) if and only if $\mathcal{L}'$ contains (i, j). Then we call them *isomorphic* and denote by $\mathcal{L} = \mathcal{L}'$.

When a permutation graph is bipartite, it is said to be a *bipartite permutation graph* (see Figure 1). Then the following lemma holds:

Lemma 2. *Let $G = (X, Y, E)$ be a connected bipartite permutation graph with $|X|, |Y| > 0$ and $\mathcal{L} = (L_1, L_2)$ its line representation. Without loss of generality, we assume that $v_1 \in X$ corresponds to $(1, i)$ for some i with $1 \leq i \leq n$. Then X and Y satisfy that $X = \{v_i \mid v_i$ corresponds to (i, j) with $i < j\}$ and $Y = \{v_i \mid v_i$ corresponds to (i, j) with $i > j\}$.*

Proof. If $v_1 \in X$ corresponds to $(1, 1)$, G is disconnected. Hence $v_1 = (1, i)$ with $i > 1$ and there is a vertex $v_{i'}$ corresponding to $(i', 1)$ with $i' > 1$. Clearly, ℓ_1 and $\ell_{i'}$ intersect. Hence $v_{i'} \in Y$, and v_1 and $v_{i'}$ satisfy the condition.

To derive a contradiction, we assume that there is a $v_j \in X$ that corresponds to (j, j') with $j \geq j'$ in G. Without loss of generality, every vertex corresponding to $\ell_k = (k, k')$ with $k < j$ satisfies the condition of the lemma. Then let x_j be the number of vertices in X placed before v_j on L_1, and y_j the number of vertices in Y placed before v_j on L_2, respectively. Moreover, let y'_j be the number of vertices in Y placed before v_j on L_1. If $j = j'$, we have $j - x_j = y'_j = y_j$. Hence G is disconnected, which is a contradiction. Thus assume $j > j'$. Then, we have $y_j + x_j = j' - 1 < j - 1 = x_j + y'_j$, equivalently, $y'_j > y_j$. Thus there exists $v_k \in Y$ with $\ell_k = (k, k')$ such that $k < j$ and $j' < k'$. We suppose that v_k is the leftmost one among such vertices. If $N(v_k) \cap X \cap \{v_1, \ldots, v_{k-1}\}$ is empty, it is not difficult to see that G is not connected (since v_j and v_k are the leftmost pair of the second connected component). Hence v_k has some neighbor, say v_x, in $X \cap \{v_1, \ldots, v_{k-1}\}$. By the assumption, for $\ell_j = (j, j')$, $\ell_k = (k, k')$, and $\ell_x(x, x')$, we have $x < k < j$ and $j' < k' < x'$. This implies that ℓ_j and ℓ_x intersect, which contradicts that v_j and v_x are in X. With a symmetric argument for Y, the lemma follows. □

Let $\mathcal{L} = (L_1, L_2)$ be a line representation of a bipartite permutation graph $G = (X, Y, E)$. For a connected bipartite permutation graph G, we can construct essentially equivalent representations by flipping $\mathcal{L}$. There are three operations that play important roles in this paper. On a *horizontal flip* $\mathcal{L}^H$ (H-flip for short) of $\mathcal{L}$, each line (i, j) on $\mathcal{L}$ is mapped to the line $(n - i + 1, n - j + 1)$. On a *vertical flip* $\mathcal{L}^V$ (V-flip for short) of $\mathcal{L}$, each line (i, j) on $\mathcal{L}$ is mapped to the line (j, i). For a line representation $\mathcal{L}$, $(\mathcal{L}^H)^V = (\mathcal{L}^V)^H$ gives us a *rotation* of $\mathcal{L}$. Hence we denote the line representation by $\mathcal{L}^R$ after this operation.

One important property is that they are unique up to isomorphism like Lemma 1:

Lemma 3. *Let $G = (V, E)$ be a connected bipartite permutation graph, and $\mathcal{L}$ and $\mathcal{L}'$ any two line representations of G. Then one of $\mathcal{L} = \mathcal{L}'$, $\mathcal{L} = \mathcal{L}'^H$, $\mathcal{L} = \mathcal{L}'^V$, and $\mathcal{L} = \mathcal{L}'^R$ holds. That is, the line representation of G is unique up to isomorphism.*

Proof. By Lemma 2, we can partition V to X and Y. Let $G^2[X] = (X, E_X)$ be a graph obtained from G by joining two vertices $x, x' \in X$ if and only if $N(x) \cap N(x') \neq \emptyset$. That is, two vertices x and x' are joined in $G^2[X]$ if the distance between them is 2. In other words, x and x' are joined by some vertex in Y. We first show that $G^2[X]$ is a connected proper interval graph. Intuitively, from a line representation of G, we can obtain the interval representation of $G^2[X]$ as follows (see Figure 1): we first rearrange the vertices in Y to vertical lines at regular intervals, and next make the vertices x in X be horizontal intervals spanning $N(x)$. Then the resultant intervals corresponding to the vertices x in X are proper, and this proper interval representation can be transformed to the unit interval representation in a straightforward way in [1]. The resultant graph $G^2[X]$ is also connected. Thus Lemma 1 implies that the resultant unit interval representation is unique up to reversal. $G^2[Y]$ can be defined in a symmetric way.

Now, we consider the rewind of this process. Given connected bipartite permutation graph $G = (V, E)$, X and Y are determined from G uniquely by Lemma 2. Then, by the discussion above, two proper interval graphs $G^2[X]$ and $G^2[Y]$ are uniquely determined. By Lemma 1, these unit interval graphs correspond to the unique interval representations. Thus, these unit interval representations give the unique orderings of X and Y in a natural way, respectively. Thus, combining these two orderings on X and Y with $G = (X, Y, E)$, we can construct the line representation of G uniquely as follows. First, we pick up the "leftmost" vertex x_1 in X according to the ordering of X. Then pick up the "leftmost" vertex y_1 from $N(x_1)$ according to the ordering of Y. Now all vertices in $N(x_1)$ are placed before x_1 on L_2 according to the ordering of Y, and all vertices in $N(y_1)$ are placed before y_1 on L_1 according to the ordering of X. Next we proceeds to x_2 and y_2, and so on. By a simple induction for the size of graph, we can show that the line representation of G is uniquely determined up to isomorphism. □

Let $G = (V, E)$ be a connected bipartite permutation graph, and $\mathcal{L}, \mathcal{L}^H, \mathcal{L}^V, \mathcal{L}^R$ its four line representations. Then some of them can be isomorphic; G is *H-symmetric*, *V-symmetric*, and *R-symmetric* if $\mathcal{L} = \mathcal{L}^H$, $\mathcal{L} = \mathcal{L}^V$, and $\mathcal{L} = \mathcal{L}^R$, respectively.

Here, we map each representation $\mathcal{L}$ to a string $s(\mathcal{L})$ in Σ^* as follows. We first sweep the endpoints from left to right on L_1, and construct a string $s_1(\mathcal{L})$ by adding '[' when the endpoint is in X, and ']' when the endpoint is in Y (e.g., $s_1(\mathcal{L})$ = [[][]][]][]][]] in Figure 1). Next we sweep the endpoints from left to right on L_2, and construct a string $s_2(\mathcal{L})$ by adding '[' when the endpoint is in Y, and ']' when the endpoint is in X (e.g., $s_2(\mathcal{L})$ = [][][[[[][]] in Figure 1). Finally, we concatenate $s_2(\mathcal{L})$ after $s_1(\mathcal{L})$ and obtain the resultant string (e.g., $s(\mathcal{L})$ = [[][]][]][]][]][][][[[[][]] in Figure 1).

Using the string, we define a *canonical* representation of G as follows. We first suppose that all strings $s(\mathcal{L}), s(\mathcal{L}^H), s(\mathcal{L}^V), s(\mathcal{L}^R)$ are distinct. Then the canonical representation is the one corresponding to the smallest string. When G satisfies exactly one symmetricalness with respect to H-flip, V-flip, or rotation, then four possible representations give two distinct strings. Then the canonical representation is the one corresponding to the smaller string. If G satisfies two symmetricalnesses, the last symmetricalness is also satisfied. Hence, in the case, four representations are isomorphic and this gives the unique canonical representation. By Lemma 3, this rule gives us a one-to-one mapping between bipartite permutation graphs and canonical representations.

Dyck path, Motzkin path, and 2-Motzkin path: A path in the (x, y) plane from $(0, 0)$ to $(2n, 0)$ with steps $(1, 1)$ and $(1, -1)$ is called a *Dyck path* of length $2n$ if it never pass below the x-axis. It is well-known that the number of Dyck paths of length n is given by the nth *Catalan number* $C(n) := \frac{1}{n+1}\binom{2n}{n}$ (see [21, Corollary 6.2.3] for further details). We will use one of the generalized notions of Catalan number; $C(n, k) := \frac{k+1}{n+1}\binom{n+1}{(n-k)/2}$, which gives us the number of subpaths of Dyck paths from $(0, 0)$ to (n, k). This can be obtained by a generalized Raney's lemma about m-Raney sequences with letting $m = 2$; see [8, Equation (7.69), p. 349] for further details. A path in the (x, y) plane from $(0, 0)$ to $(n, 0)$ with steps $(1, 0)$, $(1, 1)$, and $(1, -1)$ is called a *Motzkin path* of length n if it never go below the x-axis (see [21, Exercise 6.38] for further details). The number of Motzkin paths of length n is called *Motzkin number* $\mathcal{M}(n)$; e.g., $\mathcal{M}(1) = 1, \mathcal{M}(2) = 2, \mathcal{M}(3) = 4, \mathcal{M}(4) = 9, \mathcal{M}(5) = 21, \mathcal{M}(6) = 51$. A *2-Motzkin* path is a Motzkin path that has two kinds of step $(1, 0)$. We distinguish them by $(1, +0)$ and $(1, -0)$. Deutsch and Shapiro show that 2-Motzkin paths have correspondences to ordered trees and others [5].

In paths above, each step consists of $(1, x)$ for some x in $\{+1, -1, +0, -0\}$. Hence we will denote a path by a sequence of such integers x in $\{+1, -1, +0, -0\}$.

Machine Model: Time complexity is measured by the number of arithmetic operations. Especially we assume that each binomial coefficient and each (generalized) Catalan number can be computed in O(1) time. Moreover we assume that the basic arithmetic operations of these numbers can be done in O(1) time. This assumption is out of the standard RAM model. We have to multiply the time complexity of calculation of these numbers to the complexities to obtain the time complexity in the standard RAM model. We employ the assumption only in Section 3 to simplify the discussion. The enumeration algorithm in Section 4 does not require the assumption, and all the results are valid on the standard RAM model.

3. Counting and Random Generation

Let $P(n)$ be the set of permutations corresponding to connected bipartite permutation graphs of n vertices, and $\mathcal{B}_n$ the set of distinct (up to isomorphism) connected bipartite permutation graphs of n vertices. We denote a (not necessarily canonical) line representation of a permutation π by $\mathcal{L}_\pi = (L_1, L_2)$, and the graph of π by $G_\pi = (X, Y, E)$. Without loss of generality, we assume that X contains the vertex corresponding to $(1, \pi(1))$ in $\mathcal{L}_\pi$ for $\pi(1) > 1$. Now, we construct a 2-Motzkin path as follows. For each i with $1 \leq i \leq n$, we see the endpoints at i on L_1 and L_2. Let p_i and q_i be the endpoints on L_1 and L_2, respectively. We say that p_i is in X (and Y) if p_i is the endpoint of a vertex corresponding to $(i, \pi(i))$ in X (and Y, respectively). Similarly, we say that q_i is in X (and Y) if q_i is the endpoint of a vertex corresponding to $(\pi^{-1}(i), i)$ in X (and Y, respectively). If G_π is not connected, in each connected component, we assume that the vertex corresponding to the leftmost endpoint on L_1 belongs to X. Then the value z_i is defined as follows;

$$z_i = \begin{cases} +1 & \text{if } p_i \text{ is in } X \text{ and } q_i \text{ is in } Y, \\ -1 & \text{if } p_i \text{ is in } Y \text{ and } q_i \text{ is in } X, \\ +0 & \text{if } p_i \text{ and } q_i \text{ are in } X, \\ -0 & \text{if } p_i \text{ and } q_i \text{ are in } Y. \end{cases}$$

That is, two values $+0$ and -0 are distinguished (for counting) but have the same value. From the sequence $z_1, \ldots, z_n$, we can consider a path $Z_\pi = (z_1, \ldots, z_n)$. (For example, $Z_\pi = (+1, +0, -0, +0, -0, -0, +1, -0, -1, +1, -1, -1)$ for the graph in Figure 1.) Note that $\pi = \pi'$ if and only if $Z_\pi = Z_{\pi'}$. For the path Z_π, we define its *height at point i* by $\sum_{j=1}^{i} z_j$. To simplify, we define that the height at point 0 is 0. We show that Z_π is a 2-Motzkin path that has positive height at point i, $1 < i < n$, if and only if $\pi \in P(n)$. To this end, we need a property of connected permutation graphs.

Lemma 4 ([9, Lemma 3.2])**.** *Let π be a permutation on $\{1, \ldots, n\}$. Then G_π is disconnected if and only if there exists $k < n$ such that $\{\pi(1), \pi(2), \ldots, \pi(k)\} = \{1, 2, \ldots, k\}$.*

Then we have the following lemma.

Lemma 5. *A sequence $Z = (z_1, \ldots, z_n)$ on the alphabet $\{+1, -1, +0, -0\}$ is constructed from $\pi \in P(n)$ in the above way if and only if Z is a 2-Motzkin path such that Z has height 0 at point 0 and n, and positive height at point i with $0 < i < n$.*

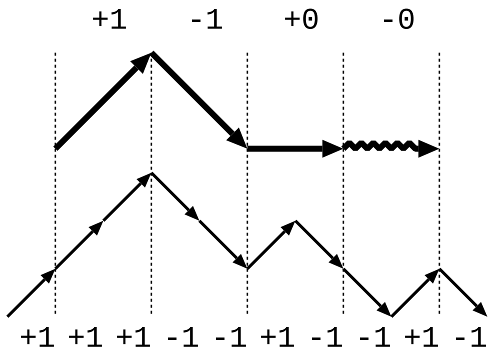

Figure 2: An example of the bijection

Proof. ($\Longrightarrow$) Clearly, $z_1 = +1$ and $z_n = -1$ since $G_\pi = (X, Y, E)$ is connected, and X and Y are nonempty. It is easy to see that the number of $+1$ is equal to the one of -1 in Z. Thus $\sum_{i=1}^{n} z_i = 0$. If Z has height 0 at some point k with $0 < k < n$, we have that $\pi(i) \in \{1, \dots, k\}$ for $1 \leq i \leq k$. From Lemma 4, we have that G_π is disconnected, which is a contradiction.
($\Longleftarrow$) We can construct a line representation $\mathcal{L} = (L_1, L_2)$ from Z as follows:

1. At point i ($1 \leq i \leq n$) on L_1, put x if $z_i \in \{+1, +0\}$, otherwise put y;
2. At point i ($1 \leq i \leq n$) on L_2, put x if $z_i \in \{-1, +0\}$, otherwise put y;
3. Draw a line segment from the ith x on L_1 to the ith x on L_2 for each i;
4. Draw a line segment from the ith y on L_1 to the ith y on L_2 for each i.

Then, we have a permutation π of $\mathcal{L}$. Thus, it suffices to show that $\pi \in P(n)$, that is, G_π is connected and bipartite. Clearly, two lines in $\mathcal{L}$ intersect only if one of them is a line from x to x and another line is from y to y. So, G_π is bipartite. If G_π is disconnected then there exists an index $k < n$ such that $\pi(i) \in \{1, \dots, k\}$ for $1 \leq i \leq k$ (Lemma 4). Obviously, this implies $\sum_{i=1}^{k} z_i = 0$, which contradicts the assumption. □

From the above characterization, we can count the number of elements in $P(n)$. Deutsch and Shapiro [5] have shown the following bijection between 2-Motzkin paths of length n and Dyck paths of length $2(n+1)$: In a 2-Motzkin path, we replace $+1$ by $(+1, +1)$, -1 by $(-1, -1)$, $+0$ by $(+1, -1)$, and -0 by $(-1, +1)$; Then add $+1$ before the obtained sequence, and add -1 after the sequence. Figure 2 shows an example. Note that a 2-Motzkin path has height k at point i if and only if the corresponding Dyck path has height $2k + 1$ at point $2i + 1$. The bijection gives the following lemma, which yields $|P(n)| = C(n-1)$.

Lemma 6 ([5])**.** *The number of 2-Motzkin paths of length n is $C(n+1)$.*

Corollary 1. $|P(n)| = C(n-1)$.

Proof. Let $\pi \in P(n)$. Since π bijectively corresponds to Z_π, it suffices to count the elements of Z_π. Lemma 5 and its proof imply that Z_π bijectively corresponds to a 2-Motzkin path of length $n-2$ (as the first and the last steps in Z_π are removed). The corollary follows from Lemma 6. □

We can show that the bijection is also a bijection for restricted paths. For $z \in \{+1, -1, +0, -0\}$, we define $-z$ naturally; $-z = \pm b$ if and only if $z = \mp b$ for $b \in \{0, 1\}$. A Dyck path $D = (d_1, \ldots, d_{2n})$ is *symmetric* if $d_i = -d_{2n-i+1}$ for $1 \le i \le 2n$.

Lemma 7 ([18])**.** *The number of symmetric Dyck paths of length* $2n$ *is* $\binom{n}{\lfloor n/2 \rfloor}$.

A 2-Motzkin path $Z = (z_1, \ldots, z_n)$ is *semi-symmetric* if $z_i = -z_{n-i+1}$ for $1 \le i \le n$, and Z is *symmetric* if $z_i = -z_{n-i+1}$ for $z_i \in \{+1, -1\}$ and $z_i = z_{n-i+1}$ for $z_i \in \{+0, -0\}$. A 2-Motzkin path can be semi-symmetric only if its length is even. Obviously, the bijection is also a bijection between symmetric 2-Motzkin paths of length n and symmetric Dyck paths of length $2(n+1)$. Furthermore, if n is even, there is a bijection between semi-symmetric 2-Motzkin paths of length n and symmetric Dyck paths of length $2(n+1)$, since a semi-symmetric 2-Motzkin path can be bijectively transformed to a symmetric 2-Motzkin path by flipping the signs of 0s in the right half. From the above observation and Lemma 7, we have the following corollary.

Corollary 2. *The number of symmetric 2-Motzkin paths of length n is* $\binom{n+1}{\lfloor (n+1)/2 \rfloor}$. *If n is even, the number of semi-symmetric 2-Motzkin paths of length n is also* $\binom{n+1}{\lfloor (n+1)/2 \rfloor}$.

Any given $\pi \in P(n)$, Lemma 3 implies that there exist at most four line representations $\mathcal{L}_\pi$, $\mathcal{L}_\pi^H$, $\mathcal{L}_\pi^V$, and $\mathcal{L}_\pi^R$ for a graph G_π. We define four subsets of $P(n)$ as follows: (1) $P^H(n) = \{\pi \in P(n) \mid \mathcal{L}_\pi$ is H-symmetric$\}$, (2) $P^V(n) = \{\pi \in P(n) \mid \mathcal{L}_\pi$ is V-symmetric$\}$, (3) $P^R(n) = \{\pi \in P(n) \mid \mathcal{L}_\pi$ is R-symmetric$\}$, and (4) $P^F(n) = P^H(n) \cap P^R(n) \cap P^V(n)$.

Proposition 1. *If n is odd,* $P^H(n)$ *and* $P^V(n)$ *are empty.*

Proof. Both H-flip and V-flip exchange X and Y, which are determined uniquely by Lemma 2. Thus $P^H(n)$ and $P^V(n)$ can be nonempty only if $|X| = |Y|$. Therefore, they are empty if $|X| + |Y|$ is odd. □

Proposition 2. $P^F(n) = P^H(n) \cap P^V(n) = P^V(n) \cap P^R(n) = P^R(n) \cap P^H(n)$.

Proof. Let $\pi \in P^H(n) \cap P^V(n)$. Then $\mathcal{L}_\pi = \mathcal{L}_\pi^H = \mathcal{L}_\pi^V$. Since $\mathcal{L}_\pi^R = \left(\mathcal{L}_\pi^H\right)^V$ for any π, we have that $\mathcal{L}_\pi^R = \left(\mathcal{L}_\pi^H\right)^V = \mathcal{L}_\pi^V = \mathcal{L}_\pi$. Hence $\pi \in P^R(n)$. The remaining two cases are similar. □

Lemma 8. $|\mathcal{B}_n| = \frac{1}{4}\left(|P(n)| + |P^H(n)| + |P^V(n)| + |P^R(n)|\right)$.

Proof. From Lemma 3 and Proposition 2, each connected bipartite permutation graph corresponds to four, two, and one permutations if it has no, one, and three symmetricalness, respectively. According to the number of corresponding permutations, we can partition $\mathcal{B}_n$ into three sets $\mathcal{B}_n^4$, $\mathcal{B}_n^2$, and $\mathcal{B}_n^1$. Each element of $\mathcal{B}_n^i$ corresponds to exactly i permutations in $P(n)$: For $G \in \mathcal{B}_n^1$, there exists $\pi \in P^F(n)$ such that $G \simeq G_\pi$; For $G \in \mathcal{B}_n^2$, there exist two permutations π_1 and π_2 in $(P^H(n) \cup P^V(n) \cup P^R(n)) \setminus P^F(n)$ such that $G \simeq G_{\pi_1} \simeq G_{\pi_2}$; For $G \in \mathcal{B}_n^4$, there exist four permutations π_i, $1 \le i \le 4$, in $P(n) \setminus \left(P^H(n) \cup P^V(n) \cup P^R(n)\right)$ such that $G \simeq G_{\pi_i}$ for $1 \le i \le 4$. Combining the inclusion-exclusion principle with Proposition 2 implies that

$$|P^H(n) \cup P^V(n) \cup P^R(n)| = |P^H(n)| + |P^V(n)| + |P^R(n)| - 2|P^F(n)|.$$

So, we have that

$$\begin{aligned}|\mathcal{B}_n| &= |\mathcal{B}_n^1| + |\mathcal{B}_n^2| + |\mathcal{B}_n^4| \\ &= |P^F(n)| + \frac{1}{2}\left(|P^H(n)| + |P^V(n)| + |P^R(n)| - 3|P^F(n)|\right) \\ &\qquad + \frac{1}{4}\left(|P(n)| - |P^H(n)| - |P^V(n)| - |P^R(n)| + 2|P^F(n)|\right) \\ &= \frac{1}{4}\left(|P(n)| + |P^H(n)| + |P^V(n)| + |P^R(n)|\right),\end{aligned}$$

as required. □

Lemma 8 implies that it suffices to count the elements of $P(n)$, $P^H(n)$, $P^V(n)$, and $P^R(n)$ to show the size of $\mathcal{B}_n$. For our random generation, we also count the elements in $P^F(n)$.

Lemma 9. $|P^V(n)| = C(n/2-1)$ *for even n.*

Proof. Let $\pi \in P^V(n)$. We claim that $Z_\pi = (z_1, \ldots, z_n)$ contains neither $+0$ nor -0. If $z_i = +0$ for some i, $1 \le i \le n$, $\mathcal{L}_\pi$ contains the lines (i, j) and (k, i) for some j and k, $k < i < j$. However, since $\mathcal{L}_\pi$ is V-symmetric, $\mathcal{L}_\pi$ contains (j, i) as well. This implies that $j = k$, a contradiction. The proof of $z_i \neq -0$ is almost the same. Thus Z_π bijectively corresponds to a Dyck path of length $n-2$, as required. □

Lemma 10. $|P^R(n)| = \binom{n-1}{\lfloor (n-1)/2 \rfloor}$.

Proof. From Corollary 2, it suffices to show that $\pi \in P^R(n)$ if and only if the 2-Motzkin path Z_π is symmetric and has positive height at point i with $1 < i < n$.
($\Longrightarrow$) Suppose $z_i = +1$. Then the lines (i, j) and (k, i), $i < j$ and $i < k$, are in $\mathcal{L}_\pi$. Since $\pi \in P^R(n)$, we have that $(n-j+1, n-i+1)$ and $(n-i+1, n-k+1)$ are also in $\mathcal{L}_\pi$. Therefore, $z_{n-i+1} = -1$ since $i < j$ and $i < k$. The case $z_i = -1$ is similar.

Next, suppose $z_i = +0$. Then the lines (i, j) and (k, i), $k < i < j$, are in $\mathcal{L}_\pi$. Since $\pi \in P^R(n)$, we have that $(n-j+1, n-i+1)$ and $(n-i+1, n-k+1)$ are also in $\mathcal{L}_\pi$. Therefore, $z_{n-i+1} = +0$ since $k < i < j$. The case $z_i = -0$ is similar.
($\Longleftarrow$) Clearly, $\pi \in P(n)$. Let $(i, j) \in \mathcal{L}_\pi$. We show that $(n-j+1, n-i+1)$ is also in $\mathcal{L}_\pi$. Without loss of generality, we assume that $i < j$, namely $(i, j) \in X$. Let i and j be the kth endpoints of lines in X, on L_1 and L_2, respectively. For $1 \le \ell < i$, the number of indices ℓ such that $z_\ell \in \{+1, +0\}$ is $k-1$. Since Z_π is symmetric, for $n-i+1 < \ell \le n$ the number of indices ℓ such that $z_\ell \in \{-1, +0\}$ is also $k-1$. This implies that the point $n-i+1$ on L_2 is the $(|X|-k+1)$th endpoint of a line in X. Similarly, we can show that the point $n-j+1$ on L_1 is the $(|X|-k+1)$th endpoint of a line in X. Therefore, $(n-j+1, n-i+1) \in \mathcal{L}_\pi$. □

Lemma 11. $|P^H(n)| = \binom{n-1}{\lfloor (n-1)/2 \rfloor}$ *for even n.*

Proof. The idea of proof is almost the same as the one of Lemma 10. From Corollary 2, it suffices to show that $\pi \in P^H(n)$ if and only if the 2-Motzkin path Z_π is semi-symmetric and has positive height at point i with $1 < i < n$.
($\Longrightarrow$) Let $(i, j), (k, i) \in \mathcal{L}_\pi$. Since $\pi \in P^H(n)$, we have that $(n-i+1, n-j+1)$ and $(n-k+1, n-i+1)$ are also in $\mathcal{L}_\pi$. It is easy to see that (i, j) is positive if and only if $(n-i+1, n-j+1)$ is negative. In the same way, we can see that (k, i) is positive if and only if $(n-k+1, n-i+1)$ is negative. Thus, $z_i = -z_{n-i+1}$.
($\Longleftarrow$) Clearly, $\pi \in P(n)$. Let $(i, j) \in \mathcal{L}_\pi$. We show that $(n-i+1, n-j+1)$ is also in $\mathcal{L}_\pi$. Without loss of generality, we assume that $i < j$, namely $(i, j) \in X$. Let i and j be the kth endpoints of lines in X, on L_1 and L_2, respectively. For $1 \leq \ell < i$, the number of indices ℓ such that $z_\ell \in \{+1, +0\}$ is $k-1$. Since Z_π is semi-symmetric, for $n-i+1 < \ell \leq n$ the number of indices ℓ such that $z_\ell \in \{-1, -0\}$ is also $k-1$. This implies that the point $n-i+1$ on L_1 is the $(|X|-k+1)$th endpoint of a line in Y. Similarly, we can show that the point $n-j+1$ on L_2 is the $(|X|-k+1)$th endpoint of a line in Y. Therefore, $(n-i+1, n-j+1) \in \mathcal{L}_\pi$. □

Lemma 12. $|P^F(n)| = \binom{(n-2)/2}{\lfloor (n-2)/4 \rfloor}$ *for even n.*

Proof. From Lemma 7, it suffices to show that $\pi \in P^F(n)$ if and only if the 2-Motzkin path Z_π is a symmetric Dyck path and has positive height at point i with $1 < i < n$. This is implied by the proofs of Lemmas 9 and 11. □

Lemmas 8, 9, and Proposition 1 together show the number of elements of $\mathcal{B}_n$. We use a well-known relation $2\binom{2m-1}{m-1} = \binom{2m}{m}$ for the even case.

Theorem 13. *For $n \geq 2$, the number of connected bipartite permutation graphs of n vertices is given by*

$$|\mathcal{B}_n| = \begin{cases} \frac{1}{4}\left(C(n-1) + C(n/2-1) + \binom{n}{n/2}\right) & \text{if } n \text{ is even,} \\ \frac{1}{4}\left(C(n-1) + \binom{n-1}{(n-1)/2}\right) & \text{if } n \text{ is odd.} \end{cases}$$

Theorem 14. *For any given positive integer n, a connected bipartite permutation graph with n vertices can be generated uniformly at random in $O(n)$ time and $O(n)$ space.*

Proof. Basically, using the same idea in [18] with Lemma 6, the algorithm generates a 2-Motzkin path uniformly at random, and outputs the corresponding graph. However, this straightforward algorithm does not generate a connected bipartite permutation graph uniformly at random since it does not consider symmetricalness of the graph. That is, comparing to an asymmetric graph, the chances of graphs with one symmetricalness and three symmetricalness are only a half and a quarter, respectively. Hence the algorithm adapts the probability as follows. The algorithm first chooses one of three sets $\mathcal{B}_n$, $\mathcal{B}_n^2 \cup \mathcal{B}_n^1$, and $\mathcal{B}_n^1$ with probabilities $|\mathcal{B}_n|/B$, $|\mathcal{B}_n^2 \cup \mathcal{B}_n^1|/B$, and $2|\mathcal{B}_n^1|/B$, respectively, where $B = |\mathcal{B}_n| + |\mathcal{B}_n^2 \cup \mathcal{B}_n^1| + 2|\mathcal{B}_n^1| = |\mathcal{B}_n^4| + 2|\mathcal{B}_n^2| + 4|\mathcal{B}_n^1|$.

Next, in each case, the algorithm generates each element uniformly at random from the chosen set S. This is a natural extension of [18], and we can show the correctness in a similar way. In each case, the algorithm selects as follows.

When $S = \mathcal{B}_n$, the algorithm simply picks up an element by generating a 2-Motzkin path.

If $S = \mathcal{B}_n^2 \cup \mathcal{B}_n^1$, it meets three subcases; H-symmetric case, V-symmetric case, and R-symmetric case (note that these cases are not disjoint). These subcases are chosen with probabilities proportional to their sizes given by Lemmas 9, 10, and 11. In H-symmetric case, the algorithm first constructs the left half of the graph. To do that, the algorithm generates a *nonnegative* 2-Motzkin path of half length uniformly at random. Here, a nonnegative 2-Motzkin path is defined in a similar way to the nonnegative Dyck path in [18]; it is a subpath of a 2-Motzkin path that ends at (n, i) for some $i \geq 0$. A nonnegative 2-Motzkin path of length n can be generated by adding each consecutive pair in a nonnegative Dyck path of length $2(n-1)$ after "+1" (Figure 2). Thus the algorithm can generate a nonnegative 2-Motzkin path by modifying the algorithm in [18], that generates the path backwardly. Then the right half can be constructed from the left half since the resultant 2-Motzkin path has to be semi-symmetric. In V-symmetric case, the algorithm generates a 2-Motzkin path that consists of only $+1$ and -1, or consequently, a Dyck path. Hence we can use the same algorithm in [18]. The R-symmetric case is similar to H-symmetric case. The algorithm first generates a nonnegative 2-Motzkin path of half length, and extends it to be symmetric.

In the last case, the algorithm picks up an element from $\mathcal{B}_n^1$. This case is a combination of the three subcases above. Thus the algorithm has to generate a symmetric 2-Motzkin path that only contains $+1$ and -1, which is a symmetric Dyck path. Thus we can use the same algorithm in [18] again. □

In the RAM model, binomial coefficient $\binom{n}{k}$ can be computed in $O(k^2 + k \log k)$ time and $O(k)$ space with Iriyama's algorithm [10]. Thus Catalan number and its generalization can be computed in $O(n^2)$ time. Since we compute the generalized Catalan number $n/2$ times in the R-symmetric and H-symmetric cases, our random generation algorithm can be performed in $O(n^3)$ time. Note that $|\mathcal{B}_n|$ is exponentially larger than $|\mathcal{B}_n^2 \cup \mathcal{B}_n^1|$ and $2|\mathcal{B}_n^1|$ so the probability of selecting the case $S = \mathcal{B}_n$ is close to 1. Therefore our algorithm runs in $O(n^2)$ expected time on the RAM model.

4. Enumeration

In this section we give an efficient algorithm to enumerate all bipartite permutation graphs of n vertices. Our algorithm can enumerate such graphs in $O(1)$ time for each.

Our approach is to repeatedly enumerate all bipartite permutation graphs of the specified number of vertices. If we can enumerate all bipartite permutation graphs with $p = |X|$ and $q = |Y|$, such graphs of n vertices can be enumerated by repeating the method for each pair of $(p, q) = (\lceil \frac{n}{2} \rceil, \lfloor \frac{n}{2} \rfloor), (\lceil \frac{n}{2} \rceil + 1, \lfloor \frac{n}{2} \rfloor - 1), \ldots, (n-1, 1)$. By the above observation and Lemma 3, it is sufficient to enumerate all canonical representations of bipartite permutation graphs with $p = |X|$ and $q = |Y|$.

We first define a tree structure, *family tree*, among the set of canonical representations. The algorithm traverses the family tree efficiently. As a result, we can enumerate all canonical representations.

Let $S_{p,q}$ be the set of canonical representations of bipartite permutation graphs of p vertices in X and q vertices in Y. We assume $p \geq q$ without loss of generality. The *root* $R_{p,q}$ in $S_{p,q}$ is the smallest representation in $S_{p,q}$; $s(R_{p,q}) = [[\cdots[]]\cdots][[\cdots[]]\cdots]$ (Figure 3). As we will see, the root corresponds to the root vertex in a tree structure among $S_{p,q}$.

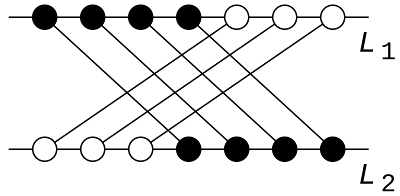

Figure 3: $R_{4,3}$ in $S_{4,3}$.

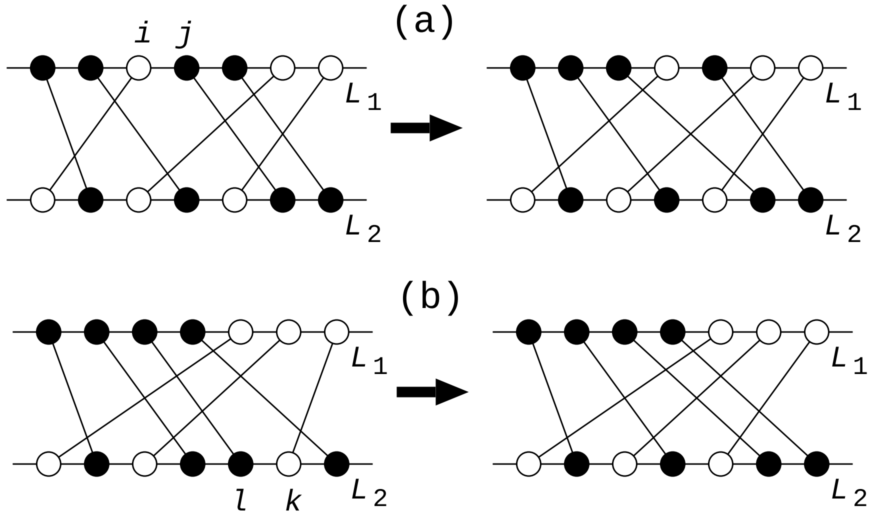

Figure 4: Examples of the parents.

Let $\mathcal{L} = (L_1, L_2)$ be a representation in $S_{p,q} \setminus \{R_{p,q}\}$. Let $s(\mathcal{L}) = x_1x_2 \cdots x_{2n}$, $s_1(\mathcal{L}) = x_1x_2 \cdots x_n$, and $s_2(\mathcal{L}) = x_{n+1}x_{n+2} \cdots x_{2n}$. Now we define "the parent" $P(\mathcal{L})$ of the representation $\mathcal{L}$ in $S_{p,q}$ as follows. We have two cases.

Case (a): $s_1(\mathcal{L}) \neq s_1(R_{p,q})$. Let i be the index of $s_1(\mathcal{L})$ such that x_i = ']' and $x_{i'}$ = '[' for all $i' < i$, and j be the index of $s_1(\mathcal{L})$ such that x_j = '[' and $x_{j'}$ = ']' for all $i \leq j' < j$. Then j is the *swappable point* of $\mathcal{L}$. $P(\mathcal{L})$ is the representation obtained from $\mathcal{L}$ by swapping two endpoints at $j - 1$ and j on L_1 (Figure 4(a)).

Case (b): $s_1(\mathcal{L}) = s_1(R_{p,q})$. In this case we define $P(\mathcal{L})$ by swapping two endpoints on L_2. Let k be the index of $s_2(\mathcal{L})$ such that x_k = '[' and $x_{k'}$ = ']' for all $k < k'$, and l be the index of $s_2(\mathcal{L})$ such that x_l = ']' and $x_{l'}$ = '[' for all $l < l' \leq k$. Then l is called *the swappable point* of $\mathcal{L}$. $P(\mathcal{L})$ is the representation obtained from $\mathcal{L}$ by swapping two endpoints at l and $l + 1$ on L_2. See Figure 4(b).

$P(\mathcal{L})$ is called the *parent* of $\mathcal{L}$ and $\mathcal{L}$ is called a *child* of $P(\mathcal{L})$. We can observe that $s(P(\mathcal{L}))$ is smaller than $s(\mathcal{L})$, and the parent $P(\mathcal{L})$ of $\mathcal{L}$ in $S_{p,q} \setminus \{R_{p,q}\}$ is always defined, since there exists the swappable point of $\mathcal{L}$. Since $\mathcal{L}$ is canonical, so is $P(\mathcal{L})$. The next lemma shows we finally obtain the root in $S_{p,q}$ by repeatedly finding the parent.

Lemma 15. *Let $\mathcal{L}$ be a representation in $S_{p,q} \setminus \{R_{p,q}\}$. The sequence obtained by repeatedly finding the parent ends up with the root $R_{p,q}$.*

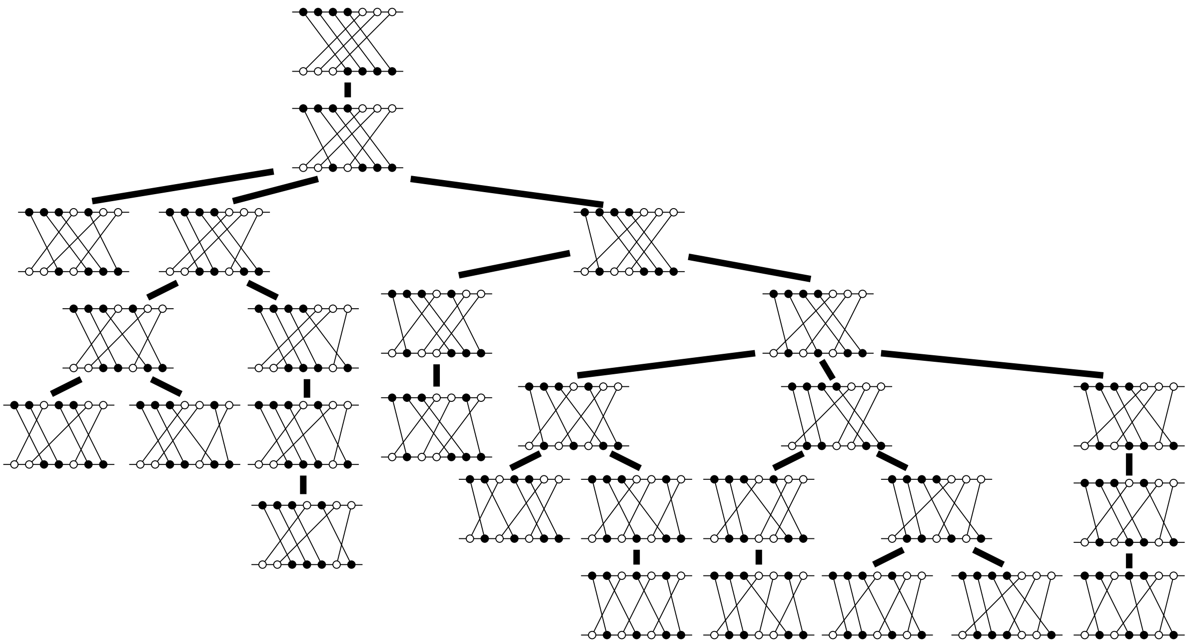

Figure 5: Family tree of $S_{4,3}$.

Proof. For a representation $\mathcal{L}$ with $s(\mathcal{L}) = x_1x_2\cdots x_{2n}$, we define a potential function $f(\mathcal{L}) = \Sigma_{i=1}^{2n} 2^{2n-i} g(x_i)$, where $g($'['$) = 0$ and $g($']'$) = 1$. $f(\mathcal{L})$ is a mapping from $\mathcal{L}$ into a non-negative integer. We can observe that $f(R_{p,q})$ is the smallest among values of representations in $S_{p,q}$.

Let j be the swappable point of $\mathcal{L}$. In Case 1, we have $f(P(\mathcal{L})) = f(\mathcal{L}) - 2^{2n-(j-1)} + 2^{2n-j} = f(\mathcal{L}) - 2^{2n-j} < f(\mathcal{L})$ by the definition of the parent and the potential function. Similarly, in Case 2, we have $f(P(\mathcal{L})) = f(\mathcal{L}) - 2^{2n-(j+n)} + 2^{2n-(j+n+1)} = f(\mathcal{L}) - 2^{2n-(j+n)-1} < f(\mathcal{L})$. Therefore $f(P(\mathcal{L})) < f(\mathcal{L})$ holds. Since the parent of $\mathcal{L}$ is always defined for $\mathcal{L}$ in $S_{p,q} \setminus \{R_{p,q}\}$, we eventually obtain $R_{p,q}$ by repeatedly finding the parent of the derived representation, which completes the proof. □

By merging all these sequences we have the *family tree* $T_{p,q}$ of $S_{p,q}$; the root of $T_{p,q}$ corresponds to $R_{p,q}$, the vertices of $T_{p,q}$ correspond to representations in $S_{p,q}$, and each edge corresponds to a relation between a representation in $S_{p,q} \setminus \{R_{p,q}\}$ and its parent. See Figure 5 for an example.

Now we give an algorithm that enumerates all representations in $S_{p,q}$. The algorithm traverses a family tree and enumerates canonical representations corresponding to the vertices of the family tree. To traverse a family tree, we design finding all children of a given canonical representation.

We need some definitions. $\mathcal{L}_1[i]$ is the line representation obtained from $\mathcal{L}$ by swapping two endpoints at i and $i+1$ on L_1, and similarly $\mathcal{L}_2[i]$ is the line representation obtained from $\mathcal{L}$ by swapping two endpoints at $i-1$ and i on L_2. If $\mathcal{L} = P(\mathcal{L}_1[i])$ (and $\mathcal{L} = P(\mathcal{L}_2[i])$), we say i is a *nominated point* on L_1 (and L_2, respectively). $\mathcal{L}_1[i]$ (and $\mathcal{L}_2[i]$) is a child of $\mathcal{L}$ only if i is a nominated point on L_1 (and L_2) and $\mathcal{L}_1[i]$ (and $\mathcal{L}_2[i]$, respectively) is connected and canonical.

For a string $s(\mathcal{L}) = x_1x_2\cdots x_{2n}$, we define the *connectivity value* $c(i)$ for $i = 0, 1, \ldots, 2n$ as follows: $c(0) = c(n) = 0$, and

$$c(i) = \begin{cases} c(i-1)+1 & \text{if } (x_i = \text{`['} \text{ and } i < n) \text{ or } (x_i = \text{`]'} \text{ and } i > n) \\ c(i-1)-1 & \text{if } (x_i = \text{`]'} \text{ and } i < n) \text{ or } (x_i = \text{`['} \text{ and } i > n) \end{cases}$$

Intuitively, $c(i)$ for $i < n$ is the number of '['s minus the number of ']'s in $x_1 x_2 \cdots x_i$, and $c(i)$ for $i > n$ is the number of ']'s minus the number of '['s in $x_{n+1} x_{n+2} \cdots x_i$. A bipartite permutation graph is connected if and only if we have $c(i) \neq c(n+i)$ for each $i = 1, 2, \ldots, n-1$. We say $\mathcal{L}$ is *connected* if $c(i) \neq c(n+i)$ for each $i = 1, 2, \ldots, n-1$.

All children can be enumerated as follows. We construct $\mathcal{L}_1[i]$ for each $i = 1, 2, \ldots, n-1$, then check whether or not (1) i is a nominated point on L_1, (2) $\mathcal{L}_1[i]$ is connected and (3) $\mathcal{L}_1[i]$ is canonical. If all conditions are satisfied, $\mathcal{L}_1[i]$ is a child. Similarly, we check whether or not $\mathcal{L}_2[i]$ is a child for each $i = 2, 3, \ldots, n$. This method takes much running time.

To improve the running time, We show that (1) the list of nominated points can be maintained efficiently, and (2) efficient way to check if a representation is connected and canonical.

Lemma 16. *Let $\mathcal{L} = (L_1, L_2)$ be a representation in $S_{p,q}$. There exist at most 3 nominated points on L_1 and L_2.*

Proof. Let $s(\mathcal{L}) = x_1 x_2 \cdots x_{2n}$. We consider the following two cases.

Case 1: $s_1(\mathcal{L}) \neq s_1(R_{p,q})$. Let i be the index of $s_1(\mathcal{L})$ such that x_i = ']' and $x_{i'}$ = '[' for all $i' < i$. Then $i-1$ is a nominated point on L_1. Let j be the index of $s_1(\mathcal{L})$ such that x_j = '[' and $x_{j'}$ = ']' for all $i \leq j' < j$. If x_{j+1} = ']' holds, then j is a nominated point. Other points on L_1 are not nominated points and there is no nominated point on L_2.

Case 2: $s_1(\mathcal{L}) = s_1(R_{p,q})$. Clearly we have one nominated point p on L_1, where p is equal to the number of '['s in $x_1 x_2 \cdots x_n$. Now we consider nominated points on L_2. Let k be the index of $s_2(\mathcal{L})$ such that x_k = '[' and $x_{k'}$ = ']' for all $k < k'$. Then $k+1$ is a nominated point on L_2. Let l be the index of $s_2(\mathcal{L})$ such that x_l = ']' and $x_{l'}$ = '[' for all $l < l' \leq k$. If x_{l-1} = '[' holds, then l is a nominated point on L_2. Other points on L_2 are not nominated. □

We have the following lemma.

Lemma 17. *Given $\mathcal{L}$ and its nominated points, we can construct the list of nominated points of each child in $O(1)$ time.*

Proof. We first consider the nominated points on L_1. Let n_1, n_2 ($n_1 < n_2$) be two nominated points on L_1. We consider each case of $\mathcal{L}_1[n_1]$ and $\mathcal{L}_1[n_2]$.

Case 1: $\mathcal{L}_1[n_1]$. If x_{n_1+2} = '[' then $n_2 = n_1 + 2$ holds or $\mathcal{L}$ has only one nominated point n_1. In this case $\mathcal{L}_1[n_1]$ has one nominated point $n_1 - 1$ on L_1. Otherwise, x_{n_1+2} = ']', $\mathcal{L}_1[n_1]$ has two nominated points $n_1 - 1$ and $n_1 + 1$ on L_1. $\mathcal{L}_1[n_1]$ has no nominated point on L_2.

Case 2: $\mathcal{L}_1[n_2]$. If x_{n_2+2} = '[', then $\mathcal{L}_1[n_2]$ has one nominated point n_1. Otherwise, x_{n_2+2} = ']', $\mathcal{L}_1[n_2]$ has two nominated points n_1 and $n_2 + 1$.

Therefore each nominated point of $\mathcal{L}_1[n_2]$ and $\mathcal{L}_2[n_2]$ (1) appears in the previous or next point of n_1 or n_2, (2) disappears from the list, or (3) is identical to one of $\mathcal{L}$'s.

The case on L_2 is symmetric and hence omitted. □

Algorithm 1: find-all-children($\mathcal{L}$)

```
1 begin
2     for each nominated point i on L_1 do
3         if ℒ_1[i] is connected and canonical then find-all-children(ℒ_1[i])
4     end
5     for each nominated point i on L_2 do
6         if ℒ_2[i] is connected and canonical then find-all-children(ℒ_2[i])
7     end
8 end
```

Now we have **Algorithm** 1, that generates all children of a given representation $\mathcal{L}$. For each nominated point i on L_1 (and L_2), it first checks if $\mathcal{L}_1[i]$ (and $\mathcal{L}_2[i]$) is connected and canonical, and next recursively calls it for $\mathcal{L}_1[i]$ (and $\mathcal{L}_2[i]$, respectively) if it satisfies the conditions. By calling the algorithm recursively at $R_{p,q}$ in $S_{p,q}$, we can traverse the family tree $T_{p,q}$ and enumerate all representations in $S_{p,q}$.

By Lemma 17, steps 2 and 5 can be done in O(1) time in each recursive call. The remaining task is checking whether or not $\mathcal{L}$ is connected and canonical efficiently.

We first consider the check of connectivity of a representation. By symmetry we only consider $\mathcal{L}_1[i]$ without loss of generality. Assume $\mathcal{L}$ is connected. Then $\mathcal{L}_1[i]$ is connected only if $c(i) \neq c(n+i)$ and $c(i+1) \neq c(n+i+1)$. We can check such conditions in O(1) time using an array of size $2n$ to maintain the sequences of connectivity values of $\mathcal{L}_1[i]$. Update of the array also can be done in O(1) time. Therefore, the connectivity of $\mathcal{L}_1[i]$ can be checked in O(1) time.

Next we check whether or not $\mathcal{L}$ is canonical. When $p \neq q$, $s(\mathcal{L})$ is canonical if $s(\mathcal{L})$ is the smallest string among $s(\mathcal{L}^V)$, $s(\mathcal{L}^H)$ and $s(\mathcal{L}^R)$. If $p = q$, we need more discussions. Let $\mathcal{L}$ be a representation in $S_{p,q}$ and G be the bipartite permutation graph corresponding to $\mathcal{L}$. Then there exists a line representation $\mathcal{L}'$ obtained from $\mathcal{L}$ by swapping lines corresponding to vertices in X and ones in Y. Similarly, we denote by $\mathcal{L}^{V'}$, $\mathcal{L}^{H'}$, $\mathcal{L}^{R'}$ the representations obtained from $\mathcal{L}^V$, $\mathcal{L}^H$, $\mathcal{L}^R$ by swapping lines corresponding to vertices in X and ones in Y, respectively. Then $\mathcal{L}$ is canonical if and only if $s(\mathcal{L})$ is the smallest string among $s(\mathcal{L}^V)$, $s(\mathcal{L}^H)$, $s(\mathcal{L}^R)$, $s(\mathcal{L}')$, $s(\mathcal{L}^{V'})$, $s(\mathcal{L}^{H'})$ and $s(\mathcal{L}^{R'})$. Using a similar idea in [18], we have the following lemma.

Lemma 18. *One can determine whether or not $\mathcal{L} = (L_1, L_2)$ is canonical in $O(1)$ time.*

Proof. Let $s(\mathcal{L}) = x_1 x_2 \cdots x_{2n}$ and $s(\mathcal{I}) = y_1 y_2 \cdots y_{2n}$ for any $\mathcal{I} \in \{\mathcal{L}^H, \mathcal{L}^V, \mathcal{L}^R, \mathcal{L}', \mathcal{L}^{V'}, \mathcal{L}^{H'}, \mathcal{L}^{R'}\}$. We maintain a doubly linked list L in order to check $s(\mathcal{L}) < s(\mathcal{I})$ in O(1) time. The list L maintains the indices of different characters in $s(\mathcal{L})$ and $s(\mathcal{I})$. L is empty if and only if $s(\mathcal{L}) = s(\mathcal{I})$. We can check whether $s(\mathcal{L}) < s(\mathcal{I})$ by comparing $x_{L[1]}$ and $y_{L[1]}$, where $L[i]$ is the ith element in L.

The update of L is as follows. Let n_1, n_2 be the nominated points on L_1 of $\mathcal{L}$. We maintain i such that $L[i] \leq n_1 < L[i+1]$ and j such that $L[j] \leq n_2 < L[j+1]$. It is easy to see we can update L using i and j in O(1) time. Since the nominated point n_1 (and n_2) is updated by n_1 or $n_1 - 1$ (and $n_1 + 1$ or $n_2 + 1$, respectively) by Lemma 16, i and j can be updated in O(1) time. The case on L_2 is similar and hence omitted. □

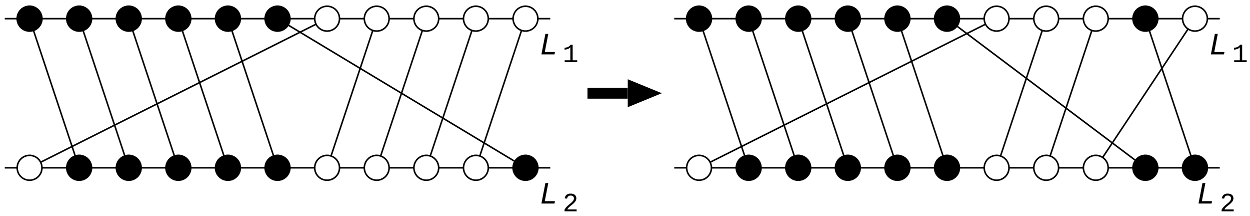

Figure 6: Construction of a representation in $S_{7,4}$ from the jump representation in $S_{6,5}$.

Therefore steps 3 and 6 in **Algorithm** 1 can be computed in O(1) time.

Lemma 19. *Our algorithm uses $O(n)$ space and runs in $O(|S_{p,q}|)$ time.*

By Lemma 19, our algorithm generates each representation in O(1) time "on average". **Algorithm** 1 may return from the deep recursive calls without outputting any representation after generating a representation corresponding to the leaf of a large subtree in a family tree. This delay can be canceled by outputting the representations in the "prepostorder" in which representations are outputted in the preorder (and postorder) at the vertices of odd (and even, respectively) depth of a family tree (see [13] for the further details). Thus we have the following lemma.

Lemma 20. *After outputting the root in $O(n)$ time, our algorithm enumerates every representation in $S_{p,q}$ in $O(1)$ time in worst case.*

Now we turn to enumerate all canonical representations corresponding to bipartite permutation graphs of n vertices. By applying Lemma 20 for each $(p, q) = (\lceil \frac{n}{2} \rceil, \lfloor \frac{n}{2} \rfloor), (\lceil \frac{n}{2} \rceil + 1, \lfloor \frac{n}{2} \rfloor - 1), \ldots, (n-1, 1)$ in this order, we can enumerate all representations; every non-root representation is generated in O(1) time. However, $R_{p,q}$ in $S_{p,q}$ is not constructed from the last outputted representation in $S_{p-1,q+1}$ in O(1) time.

This delay can be canceled as follows. Let $\mathcal{L} = (L_1, L_2)$ be a representation in $S_{p,q}$. Then $\mathcal{L}$ is *jump representation* if $s_1(\mathcal{L}) = s_1(R_{p,q})$ and $s_2(\mathcal{L}) = []] \cdots][[\cdots []$ (see Figure 6). When jump representation in $S_{p,q}$ is generated, we construct a representation $\mathcal{K}$ in $S_{p+1,q-1}$ by swapping the three lines (p, n), $(n-1, n-2)$, $(n, n-1)$ to $(p, n-1)$, $(n-1, n)$, $(n, n-2)$, respectively. We note that the line $(n-1, n-2)$ is switched to a line corresponding to a vertex in X, and $\mathcal{K}$ can be generated from $\mathcal{L}$ in O(1) time. Then we enumerate all representations in $S_{p+1,q-1}$ by traversing $T_{p+1,q-1}$ as follows. After $\mathcal{K}$ is generated, the descendants of $\mathcal{K}$ in $T_{p+1,q-1}$ are enumerated by **Algorithm** 1, and we construct $P(\mathcal{K})$. Then we traverse the descendants of $P(\mathcal{K})$ except the subtree rooted at $\mathcal{K}$ and construct $P(P(\mathcal{K}))$. We repeat this process until the root is generated. We note that $P(\mathcal{K})$ can be generated in O(1) time by maintaining the swappable point and its data structure can be updated in O(1) time.

We note that (1) swapping two endpoints of a canonical representation corresponds to adding or removing one edge in the corresponding graph and (2) a graph can be constructed from the graph corresponding to a jump representation by a constant number of operations to add and remove edges. Therefore we have the following theorem.

Theorem 21. *(1) After outputting the root in $S_{\lceil \frac{n}{2} \rceil, \lfloor \frac{n}{2} \rfloor}$, one can enumerate every canonical representation of a bipartite permutation graph of n vertices in $O(1)$ time. (2) The algorithm enumerates every connected bipartite permutation graph of n vertices in $O(1)$ time.*

Reference

[1] K. P. Bogart and D. B. West. A Short Proof that 'Proper=Unit'. *Disc. Math.*, 201:21–23, 1999.

[2] A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph Classes: A Survey*. SIAM, 1999.

[3] C.J. Colbourn. On Testing Isomorphism of Permutation Graphs. *Networks*, 11:13–21, 1981.

[4] X. Deng, P. Hell, and J. Huang. Linear-time Representation Algorithms for Proper Circular-arc Graphs and Proper Interval Graphs. *SIAM J. on Comp.*, 25(2):390–403, 1996.

[5] E. Deutsch and L. W. Shapiro. A Bijection Between Ordered Trees and 2-Motzkin Paths and Its Many Consequences. *Disc. Math.*, 256(3):655–670, 2002.

[6] R. Geary, N. Rahman, R. Raman, and V. Raman. A Simple Optimal Representation for Balanced Parentheses. Theoretical Computer Science, 368(3):231–246, 2006.

[7] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Annals of Disc. Math. 57. Elsevier, 2nd edition, 2004.

[8] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley Publishing Company, 1989.

[9] Y. Koh and S. Ree. Connected Permutation Graphs. *Disc. Math.*, 307(21):2628–2635, 2007.

[10] Y. Komaki, and M. Arisawa. *Nano Piko Kyoushitsu* (in Japanese). Kyouritsu shuppan, 1990.

[11] S.-i. Nakano. Efficient Generation of Plane Trees. *IPL*, 84(3):167–172, 2002.

[12] S.-i. Nakano, R. Uehara, and T. Uno. A New Approach to Graph Recognition and Applications to Distance-hereditary Graphs. *J. of Computer Science and Technology*, 24(3):517–533, 2009.

[13] D.E. Knuth. *Generating All Trees, History of Combinatorial Generation*, Vol. 4, Fascicle 4 of *The Art of Computer Programming*. Addison-Wesley Publishing Company, 2005.

[14] G.S. Lueker and K.S. Booth. A Linear Time Algorithm for Deciding Interval Graph Isomorphism. *J. of the ACM*, 26(2):183–195, 1979.

[15] J. I. Munro and V. Raman. Succinct Representation of Balanced Parentheses and Static Trees. *SIAM J. on Comp.*, 31:762–776, 2001.

[16] F. S. Roberts. Indifference Graphs. In F. Harary, editor, *Proof Techniques in Graph Theory*, pages 139–146. Academic Press, 1969.

[17] T. Saitoh, Y. Otachi, K. Yamanaka, and R. Uehara. Random Generation and Enumeration of Bipartite Permutation Graphs. In *ISAAC 2009*, pages 1104–1113. LNCS Vol. 5878, Springer-Verlag, 2009.

[18] T. Saitoh, K. Yamanaka, M. Kiyomi, and R. Uehara. Random Generation and Enumeration of Proper Interval Graphs. In *WALCOM 2009*, pages 177–189. LNCS Vol. 5431, Springer-Verlag, 2009.

[19] J.P. Spinrad. *Efficient Graph Representations*. AMS, 2003.

[20] R.P. Stanley. *Enumerative Combinatorics*, Vol. 1. Cambridge, 1997.

[21] R.P. Stanley. *Enumerative Combinatorics*, Vol. 2. Cambridge, 1999.

[22] R. Uehara, S. Toda, and T. Nagoya. Graph Isomorphism Completeness for Chordal Bipartite Graphs and Strongly Chordal Graphs. *Disc. App. Math.*, 145(3):479–482, 2004.