Vol.2009-AL-124 No.1
2009/5/11

Efficient Enumeration of All Pseudoline Arrangements

Katsuhisa Yamanaka,[†1] Shin-ichi Nakano,[†2]
Yasuko Matsui,[†3] Ryuhei Uehara[†4]
and Kento Nakada [†5]

Pseudoline arrangements are important and appealing objects in the area of geometory and combinatrics. In this paper we give an algorithm to enumerate all arrangements of n pseudolines. After $O(n^2)$ time preprocessing, our algorithm enumerates all arrangements in $O(1)$ time for each and uses $O(n^2)$ space.

1. Introduction

Arrangements of lines and pseudolines are one of important and appealing objects in the area of geometry and combinatorics.

Felsner[2] worked on the number of arrangements of pseudolines and showed that the number of arangements of n pseudolines is bounded by $2^{0.6974n^2}$. Arrangements of pseudolines are strongly related to oriented matroids, and there is a bijection between arrangements and oriented matroids of rank 3. Several results on enumeration of oriented matroids are known[3],[4].

In this paper we give an efficient algorithm to enumerate all arrangements of n pseudolines.

We have designed an algorithm to enumerate every "ladder lottery[10],[11]," which is a network of swaps as depicted in **Fig. 1**(c), for a given permutation. For the fixed permutation $\pi = (n, n-1, \ldots, 1)$ we can observe that each ladder lottery for π can be regarded as an arrangement of n pseudolines. See Fig. 1(a) and (c). In this paper, by simplifying the algorithm in the literatures[10],[11] to only work for the fixed permutation $\pi = (n, n-1, \ldots, 1)$, we design a simple but efficient algorithm to enumerate all arrangements of n pseudolines. After $O(n^2)$ time preprocessing, our algorithm computes each arrangement in $O(1)$ time for each.

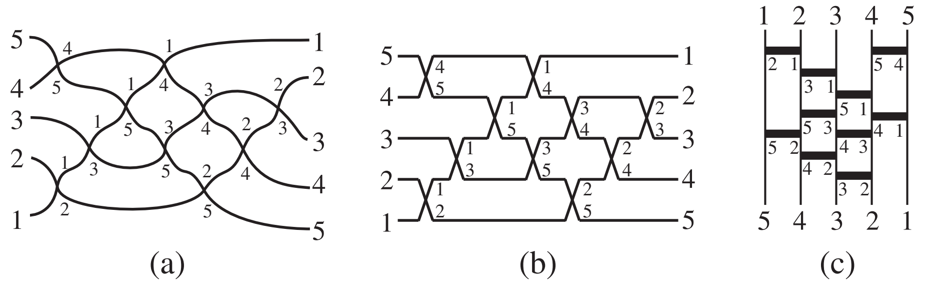

Fig. 1 (a) An arrangement of 5 pseudolines, (b) its wiring diagram and (b) the corresponding optimal ladder lottery.

The idea of our enumeration algorithm is as follows. Let S_n be the set of all combinatorial stuctures of arrangements of n pseudolines. We first define a tree structure T_n, called *the family tree*, among S_n, (see **Fig. 2**) in which each vertex of T_n corresponds to a combinatorial structure of an arrangement in S_n and each edge of T_n corresponds to a relation between two combinatorial structures of arrangements which can be transformed to the other by one *local swap operation*, as shown in **Fig. 3**. Then we design an efficient algorithm to generate all child vertices of a given vertex in T_n. Applying the algorithm recursively from the root of T_n, we can generate all vertices in T_n, and also corresponding all combinatorial structures of arrangements in S_n. Based on such a tree structure but with some other ideas a lot of efficient enumeration algorithms are designed[1],[7].

The rest of the paper is organized as follows. Section 2 gives some definitions. Section 3 defines the tree structure among S_n. Section 4 gives an efficient algorithm to enumerate all combinatorial structures of arrangements in S_n. Finally Section 5 is a conclusion.

†1 Graduate School of Information Systems, The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan. yamanaka@is.uec.ac.jp
†2 Department of Computer Science, Gunma University, 1-5-1 Tenjin-cho, Kiryu, Gunma 376-8515, Japan. nakano@cs.gunma-u.ac.jp
†3 Department of Mathematical Sciences, Tokai University, 1117, Kitakaname, Hiratsuka, Kanagawa, 259-1292, Japan. yasuko@ss.u-tokai.ac.jp
†4 School of Information Science, Japan Advanced Institute of Science and Technology, 1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan. uehara@jaist.ac.jp
†5 Research Institute for Mathematical Sciences, Kyoto University, Kyoto 606-8502, Japan, Partially supported by GCOE, Kyoto University. nakada@kurims.kyoto-u.ac.jp

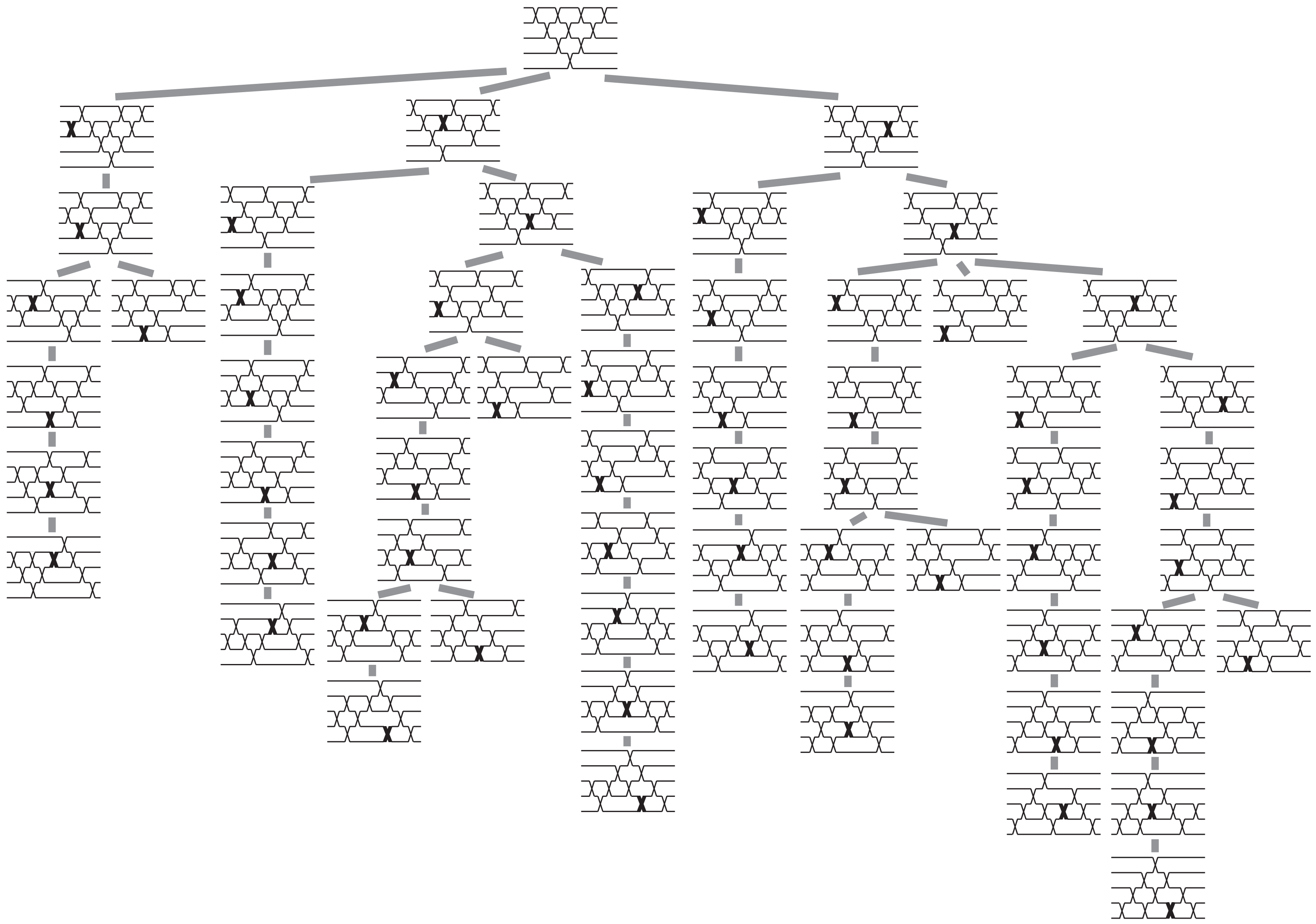

Fig. 2 The family tree T_5.

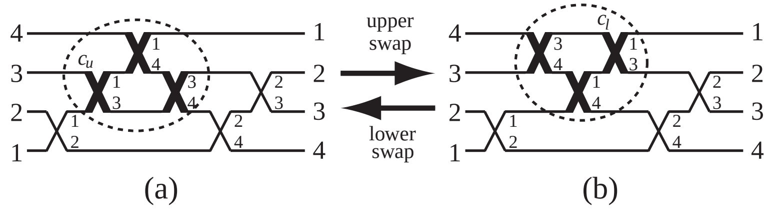

Fig. 3 A local swap operation.

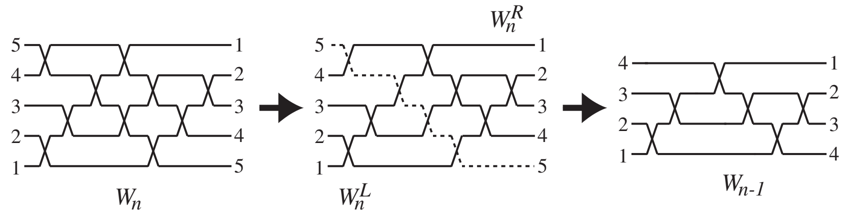

Fig. 4 The removal of the line n.

2. Preliminary

A *pseudoline* is an x-monotone curve in the Euclidean plane. An *arrangement* of pseudolines is a set of pseudolines in which every pair intersects exactly once. See Fig. 1(a). An arrangement is *simple* if no three pseudolines share a common point. Throughout this paper, the term arrangement always denotes a simple arrangement of pseudolines.

A *wiring diagram*, introduced in the literature5), of an arrangement of n pseudolines is a network with n lines and $\binom{n}{2}$ intersections. See Fig. 1(b). The left ends correspond to the reverse permutation $(n, n-1, \ldots, 1)$ in top to bottom order. The right ends correspond to the identical permutation $(1, 2, \ldots, n)$ in top to bottom order. Each line i starts at the i-th left end from the bottom, then goes right, however at every intersection the line goes up or down to cross other line, then finally line i reaches the i-th right end from the top. Each such path corresponds to a pseudoline. Note that each path has exactly $n-1$ intersections.

The combinatorial structure of each pseudoline arrangement can be modeled as a wiring diagram, and each wiring diagram models the combinatorial structure of a set of pseudolines arrangements. Note that applying some perturbation to a pseudoline arrangement still results in the same corresponding wiring diagram. We say two pseudoline arrangements are *isomorphic* if there is a bijection between their faces of corresponding wiring diagrams preserving their neighbor relation. In this paper we enumerate all combinatorial structures of pseudoline arrangements by enumerating all distinct wiring diagrams.

A *local swap operation* is a local modification of a wiring diagram, as shown in Fig. 3. Note that the dashed circle contains exactly three intersections. Also note that applying this modification to a wiring diagram of a pseudline arrangement results in other wiring diagram of other pseudoline arrangement, since only the "location" of the three intersections has changed. A local swap operation (a) to (b) in Fig. 3 is called an *upper swap operation to intersection* c_u. Similarly, a local swap operation (b) to (a) in Fig. 3 is called a *lower swap operation to intersection* c_l.

3. The Family Tree

In this section we design a tree structure T_n among wiring diagrams. See Fig. 2.

Let S_n be the set of all wiring diagrams corresponding to the set of all combinatorial structures of arrangements of n pseudolines, and $W = W_n$ be a wiring diagram in S_n. The line in the wiring diagram starting at the i-th left end from the bottom is called *line* i. The line n starts the uppermost left end, then crosses each of other lines exactly once, finally reaches the lowermost right end. Thus the line n contains exactly $n-1$ "downward" intersection, and line n is y-monotone in the wiring diagram. Note that this property does not hold for other line $i \neq n$. See Fig. 1(b).

The line n partitions W_n into the left part W_n^L and the right part W_n^R. Removing the line n from W_n as shown in **Fig. 4**, results in a wiring diagram W_{n-1} of the remaining $n-1$ lines. We say W_n is *n-clean* if W_n^L has no intersection. If the wiring diagram of W_n is n-clean then the line $n-1$ is also y-monotone in W_{n-1}, and we can define W_{n-2} similarly, and we say W_{n-1} is $(n-1)$-clean if W_{n-1}^L has no intersection. We repeat this process until some non-clean wiring diagram appears or W_2 is derived. If the wiring diagram of W_k is k-clean for each $k = n, n-1, \ldots, 3$, then W is called *the root*, denoted by R. See the rightmost

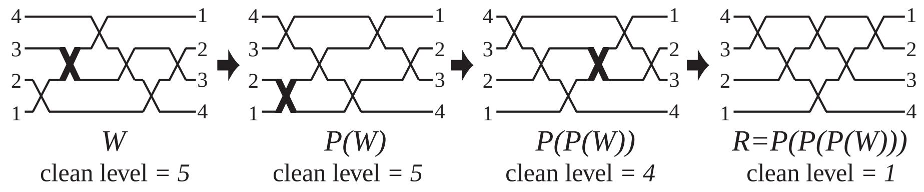

Fig. 5 The sequence of a wiring diagram W.

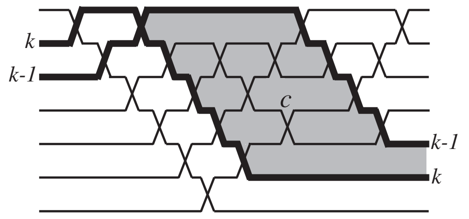

Fig. 6 An active region.

diagram in **Fig. 5**. Otherwise, there exists some k such that W_i is i-clean for each $i = n, n-1, \ldots, k$, and W_{k-1} is not $(k-1)$-clean. Then we say *the clean level of W is k*. Especially if W_n^L has an intersection, then W has the clean level $n+1$, and the root R has the clean level 3. Note that if the clean level is k then the lines $n, n-1, \ldots, k$ form so called "brick structure" in the wiring diagram, in which each line $i \geq k-1$ first goes up $n-i$ times, crossing the lines $n, n-1, \ldots, i+1$, in this order, then pass through an uppermost horizontal segment, then go down $i-1$ times. Note that "the region" below line $i \geq k$ contains no intersection of two lines both less than k. Also the region above the line k and below the line $k-1$ contains at least one intersection of two lines both less than k. See **Fig. 6**. The region is called *the active region* of the wiring diagram of A. Especially, we define the active region of R is ϕ for convenience (in the proof of Lemma 3.2).

Now we assign a wiring diagram in S_n for each wiring diagram W in $S_n \setminus \{R\}$ as follows. Assume that W has the clean level k. Thus the active region of W has at least one intersection. We say an intersection c in the active region is *visible from line $k-1$* if the two lines, say i and j, crossing at c, both next cross to line $k-1$. Among the visible intersections from line $k-1$, the lowermost intersection is called *the active intersection* of W. In Fig. 6, intersection c is the active intersection. Applying an upper swap operation to the active intersection of $W \in S_n \setminus \{R\}$ results in other wiring diagram, denoted by $P(W)$, in S_n. We say $P(W)$ is *the parent* of W, and W is a *child* of $P(W)$. Note that the parent of W is unique, while $P(W)$ may have many children. Also note that the clean level of $P(W)$ is smaller or equal to W, and $P(W)$ has less intersections in the active region of W.

We have the following lemma.

Lemma 3.1 *For any $W \in S_n \setminus \{R\}$, $P(W) \in S_n$ holds.*

Given a wiring diagram W in $S_n \setminus \{R\}$, by repeatedly finding the parent of the derived wiring diagram, we can have the unique sequence $W, P(W), P(P(W)), \ldots$ of wiring diagrams in S_n, which eventually ends up with the root R in S_n. See Fig. 5. The active intersections are depicted by thick lines.

We also have the following lemma.

Lemma 3.2 *The sequence $W, P(W), P(P(W)), \ldots$ of $W \in S_n \setminus \{R\}$ ends with $R \in S_n$.*

Proof. For each $W \in S_n$ we define *the clean potential* $C(W) = (s, t)$, where s is the clean level of W and t is the number of intersections in the active region of W. For $W_1, W_2 \in S_n$ with $C(W_1) = (s_1, t_1)$ and $C(W_2) = (s_2, t_2)$, we say W_1 *is cleaner than* W_2 if (1) $s_1 < s_2$ or (2) $s_1 = s_2$ and $t_1 < t_2$. For any $W \in S_\pi$ we can observe $P(W)$ is cleaner than W. Now R is the cleanest among S_n since $C(R) = (1, 0)$. Thus for any $W \in S_n$ the sequence of clean potentials $C(W), C(P(W)), C(P(P(W))), \ldots$ always ends at $C(R)$. □

By merging all these sequences we can have *the family tree* of S_n, denoted by T_n, in which the root of T_n corresponds to R, the vertices of T_n correspond to the wiring diagrams in S_n and each edge corresponds to a relation between a wiring diagram in S_n and its parent. See Fig. 2.

4. Enumerating

In this section we give an efficient algorithm to enumerate all wiring diagrams in S_n.

If we have an algorithm to enumerate all children of a given wiring diagram in S_n, then by recursively applying the algorithm starting at the root R of S_n, we can enumerate all wiring diagrams in S_n, and all corresponding combinatorial structures of arrangements of n pseudolines. Now we design such an algorithm.

We need some definitions. Let $W \neq R$ be a wiring diagram in S_n. Assume W has the clean level k. So each intersection below line $i \geq k$ contains no intersection of two lines both less than k, but in the active region (see Fig. 6) there is at least one intersection of two lines, say x and y, with $x, y < k-1$. Each line $i \geq k-1$ goes up $n-i$ times, "turns" at the top, then goes down $i-1$ times. For each line $i = n-1, n-2, \ldots, k-1$, if c is the first intersection to go down after intersections to go up, then c is called *the turn intersection* of line i. Note that line n contains only intersections to go down, so has no turn intersection. Also note that the turn intersection is defined only for line $i = n-1, n-2, \ldots, k-1$ since otherwise it may have many "turns."

Lemma 4.1 *Let W be a wiring diagram with the clean level k. Every turn intersection of line $i = n-1, n-2, \ldots, k$ can be lower swapped, however any other intersection on line $i = n-1, n-2, \ldots, k$ cannot.*

Proof. An intersection c_l can be lower swapped only if the dashed circle in Fig. 3(b) contains exactly three intersections. Because of the brick structure, other lower swaps are interfered by a fourth intersection. □

Let $W[c]$ be the wiring diagram derived from W by applying a lower swap operation to an intersection c. Every child of W is $W[c]$ for some c, but not all $W[c]$ are children of W. $W[c]$ is a child of W only if c is the active intersection of $W[c]$. Now we classify each $W[c]$ as a child of W or not as follows. Remember the clean level of W is k. Let $U(i)$ be the region above line i, and $L(i)$ be the region below line i.

Type 1: c is a turn intersection.

Assume the local structure of W is as shown in **Fig. 7**(a) and c is the turn intersection of line i. If $n-1 \geq i \geq k$ then c can be lower swapped by Lemma 4.1, and the clean level of $W[c]$ is $i+2$ since $W[c]$ is not $(i+1)$-clear. Thus c is the only intersection in the active region of $W[c]$, so c is the active intersection of $W[c]$. Thus $W[c]$ is a child of W. If $i = k-1$, $W[c]$ is a child of W only when c can be lower swapped.

Type 2: c is not a turn intersection.

We need to consider only intersections which can be lower swapped. Such

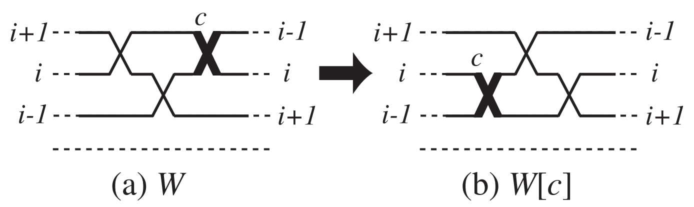

Fig. 7 Illustration for Type 1.

intersections exist only in $U(k)$. If c is "rightward" visible from neither line k nor $k-1$, then c is not the active intersection in $W[c]$, thus $W[c]$ is not a child of W. So assume otherwise.

If the lower swap operation moves c to $L(k)$, crossing the line k, then the clean level of $W[c]$ is $k+1$ and c is the only intersection in the new active region, so c is the active intersection of $W[c]$. Thus $W[c]$ is a child of W.

If the lower swap operation moves c to $L(k-1)$, crossing the line $k-1$, then the clean level of $W[c]$ remains k, and c is appended to the active region. $W[c]$ is a child of W if c is the active intersection of $W[c]$. Otherwise, $W[c]$ is not a child of W.

By maintaining (i) the clean level k (the clean level of $W[c]$ is always larger than or equal to the clean level of W) and (ii) the list of rightward visible intersections from line k or $k-1$, (those are candidate to be lower swapped, crossing the line k or $k-1$), and (iii) the list of the turn intersections, we can enumerate all children of W in $O(1)$ time for each on average.

Lemma 4.2 *All children of W can be enumerated in $O(1)$ time for each on average.*

The root R in S_n can be generated in $O(n^2)$ time because of its complete brick structure.

Theorem 4.3 *After generating and outputting the root R in S_n in $O(n^2)$ time, our algorithm enumerates all combinatorial structures of arrangements of n pseudolines in $O(1)$ time for each on average. The algorithm uses $O(n^2)$ working space.*

Proof. We show that the root R in S_n can be generated in $O(n^2)$ time. We start with n horizontal lines. Then we append intersections for each line i, $i = n, n-1, \ldots, 2$. Each line i goes up $n - i$ times, turns at the top, then goes down $i - 1$ times. When we append intersections of line i, intersections to go up are already completed, since those intersections corresponds to the crossing with lines of larger numbers. So we only need to append the $i - 1$ intersections to go down. Thus we can compute R in $O(n^2)$ time and space. $\square$

By the theorem above, our algorithm generates each wiring diagram in S_n in $O(1)$ time "on average." However it may have to return from the deep recursive calls without outputting any wiring diagram in S_n, after generating an wiring diagram corresponding to the rightmost leaf of a large subtree in the family tree. Therefore the next wiring cannot be generated in $O(1)$ time in worst case.

By modifying the algorithm so taht each wiring diagram at "even" depth in T_n is output before its children, and each wiring diagram at "odd" depth in T_n is output after its children[7], we can output the next wiring diagram in $O(1)$ time in worst case.

This technique is called "prepostordering." See the literature[7] for further details: in the literature[7] the method was not explicitly named, and the name "prepostorder" is given by Knuth[8].

We have the following theorem.

Theorem 4.4 *After $O(n^2)$ time preprocessing, we can enumerate all combinatorial structures of arrangements of n pseudolines in $O(1)$ time for each.*

5. Conclusion

In this paper we first defined the family tree among S_n. We also gave an algorithm to enumerate all wiring diagrams in S_n corresponding to combinatorial structures of arrangements of n pseudolines.

By implementing the algorithm we have computed the number of n pseudoline arrangements for each $n \leq 10$, as shown in Table 1. The numbers for $n \leq 10$ match to the reports by Knuth[6] and Widom et al.[9].

n	The number of combinatorial structures of arrangements of n pseudoline
1	1
2	1
3	2
4	8
5	62
6	908
7	24698
8	1232944
9	112018190
10	18410581880

Table 1 The number of arrangements of $n \leq 10$ pseudolines

References

1) D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Appl. Math.*, 65(1-3):21–46, 1996.

2) S.Felsner. On the number of arrangements of pseudolines. *Proc. The 12th annual Symposium on Computational geometry, (SCG 1996)*, 33–37, 1996.

3) J.Ferté, V.Pilaud, and M.Pocchiola. On the number of simple arrangements of five double pseudolines. In *Proc. Fall Workshop on Computational Geometry, (FWCG 2008)*, 45–46, 2008.

4) L.Finschi and K.Fukuda. Generation of oriented matroids – a graph theoretical approach. *Discrete Comput. Geom.*, 27:117–136, 2002.

5) J.E. Goodman. Proof of a conjecture of burr, grünbaum and sloane. *Discrete Math.*, 32:27–35, 1980.

6) D.E. Knuth. *Axioms and hulls.* LNCS 606, Springer-Verlag, 1992.

7) S.Nakano and T.Uno. Constant time generation of trees with specified diameter. *Proc. the 30th Workshop on Graph-Theoretic Concepts in Computer Science, (WG 2004)*, LNCS 3353:33–45, 2004.

8) S.Nakano and T.Uno. Personal communication. 2004.

9) M. Widom, N. Destainville, R. Mosseri, and F. Bailly. Two-dimensional random tilings of large codimension. *Proc. the 6th International Conference on Quasicrystals*, 1998.

10) K.Yamanaka, S.Nakano, Y.Matsui, R.Uehara, and K.Nakada. Efficient enumeration of all ladder lotteries. In *Proc. The 20th Workshop on Topological Graph Theory, (TGT20)*, 150–151, 2008.

11) K. Yamanaka, S. Nakano, Y. Matsui, R. Uehara, and K. Nakada. Efficient enumeration of all ladder lotteries. *Technical Report UEC-IS-09-01*, 2009. http://home.hol.is.uec.ac.jp/yamanaka/Tech/UEC-IS-09-01.pdf, submitted to a journal.