

## Efficient Enumeration of All Ladder Lotteries with $k$ Bars

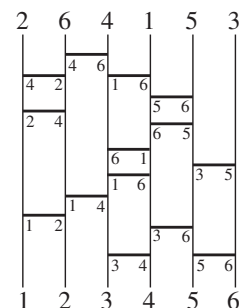
KATSUHISA YAMANAKA<sup>†1</sup> and SHIN-ICHI NAKANO<sup>†2</sup>

A *ladder lottery*, known as the “Amidakuji” in Japan, is a network with  $n$  vertical lines and many horizontal lines each of which connects two consecutive vertical lines. Each ladder lottery corresponds to a permutation. Ladder lotteries are frequently used as natural models in many areas. Given a permutation  $\pi$ , an algorithm to enumerate all ladder lotteries of  $\pi$  with the minimum number of horizontal lines are known. In this paper, given a permutation  $\pi$  and an integer  $k$ , we design an algorithm to enumerate all ladder lotteries of  $\pi$  with exactly  $k$  horizontal lines.

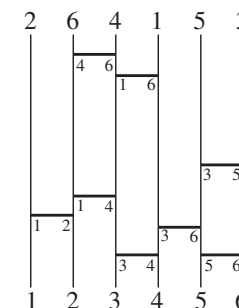
### 1. Introduction

A *ladder lottery*, known as the “Amidakuji” in Japan, is a common way to choose an assignment randomly. Formally, a ladder lottery of a permutation  $\pi = (p_1, p_2, \dots, p_n)$  is a network with  $n$  vertical lines (*lines* for short) and many horizontal lines (*bars* for short) as follows (see Fig. 1). The  $i$ -th line from the left is called *line  $i$* . The top ends of the  $n$  lines correspond to  $\pi$ . The bottom ends of the  $n$  lines correspond to the identical permutation  $(1, 2, \dots, n)$ . Each bar connects two consecutive lines. Each number  $p_i$  in  $\pi$  starts at the top end of line  $i$ , and goes down along the line, then whenever  $p_i$  comes to an end of a bar,  $p_i$  goes horizontally along the bar to the other end, then goes down again. Finally  $p_i$  reaches the bottom end of line  $p_i$ . We can regard a bar as a modification of the “current” permutation. In a ladder lottery a sequence of such modifications always results in the identical permutation  $(1, 2, \dots, n)$ .

The ladder lotteries are strongly related to primitive sorting networks, which are deeply investigated by Knuth [2]. A comparator in a primitive sorting network



**Fig. 1** A ladder lottery of the permutation  $(2,6,4,1,5,3)$  with 14 bars.



**Fig. 2** An optimal ladder lottery of the permutation  $(2,6,4,1,5,3)$ .

replaces  $p_i$  and  $p_j$  by  $\min(p_i, p_j)$  and  $\max(p_i, p_j)$ , while a bar in a ladder lottery always exchanges them.

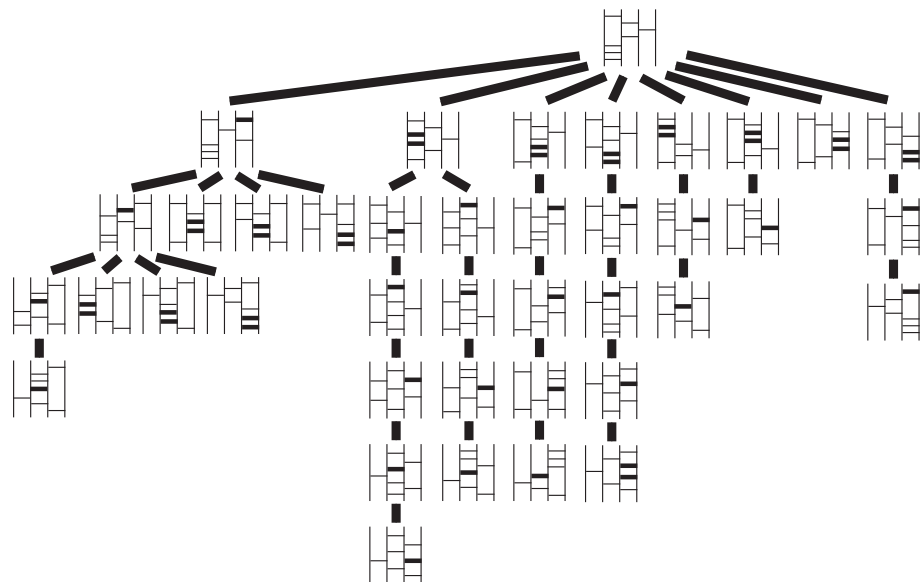
Given a permutation  $\pi = (p_1, p_2, \dots, p_n)$  the minimum number of bars to construct ladder lotteries of  $\pi$  is equal to the number of “reverse pairs” in  $\pi$ , which are pairs  $(p_i, p_j)$  in  $\pi$  with  $p_i > p_j$  and  $i < j$ . A ladder lottery of  $\pi$  with the minimum number of bars is *optimal*. The ladder lottery in Fig. 2 has eight bars, and its corresponding permutation  $(2,6,4,1,5,3)$  has eight reverse pairs:  $(2,1)$ ,  $(6,4)$ ,  $(6,1)$ ,  $(6,5)$ ,  $(6,3)$ ,  $(4,1)$ ,  $(4,3)$  and  $(5,3)$ , so the ladder lottery is optimal.

In [7] we gave an algorithm to enumerate all optimal ladder lotteries of a given permutation  $\pi$ . The algorithm generates all optimal ladder lotteries of  $\pi$  in  $O(1)$  time for each. The idea of our algorithm in [7] is as follows. We first define a tree structure  $T_\pi$ , called *the family tree*, among optimal ladder lotteries, in which each vertex of  $T_\pi$  corresponds to each optimal ladder lottery and each edge of  $T_\pi$  corresponds to a relation between two optimal ladder lotteries. Then we design an efficient algorithm to generate all child vertices of a given vertex in  $T_\pi$ . Applying the algorithm recursively from the root of  $T_\pi$ , we can generate all vertices in  $T_\pi$ , and also corresponding optimal ladder lotteries. Based on such tree structure, but with some other ideas, a lot of efficient enumeration algorithms are designed [1, 4, 5].

However the algorithm in [7] works only if the number of bars is minimum. In this paper we generalize the algorithm to enumerate all ladder lotteries in  $S_{\pi,k}$ ,

<sup>†1</sup> Graduate School of Information Systems, The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan. [yamanaka@is.uec.ac.jp](mailto:yamanaka@is.uec.ac.jp)

<sup>†2</sup> Department of Computer Science, Gunma University, 1-5-1 Tenjin-cho, Kiryu, Gunma 376-8515, Japan. [nakano@cs.gunma-u.ac.jp](mailto:nakano@cs.gunma-u.ac.jp)



**Fig. 3** The family tree  $T_{\pi,k}$ , where  $\pi = (4, 2, 3, 1)$  and  $k = 7$ .

which is the set of all ladder lotteries of a given permutation  $\pi$  with exactly  $k$  bars. The algorithm enumerates all ladder lotteries in  $S_{\pi,k}$  in  $O(1)$  time for each. Note that if  $k$  is smaller than the number of reverse pairs in  $\pi$  then  $S_{\pi,k} = \phi$ . Also if the parity of  $k$  does not match the parity of the number of reverse pairs in  $\pi$  then  $S_{\pi,k} = \phi$ . In this paper we design a new family tree (see Fig. 3) to enumerate all ladder lotteries in  $S_{\pi,k}$  for any  $k$ .

The rest of the paper is organized as follows. Section 2 gives some definitions. Section 3 defines the tree structure among ladder lotteries in  $S_{\pi,k}$ . Section 4 gives an efficient algorithm to enumerate all ladder lotteries in  $S_{\pi,k}$ . Finally Section 5 is a conclusion.

## 2. Preliminary

A ladder lottery  $L$  of a permutation  $\pi = (p_1, p_2, \dots, p_n)$  is a network with  $n$  vertical lines (*lines* for short) and many horizontal lines (*bars* for short) as follows. The  $i$ -th line from the left is called *line  $i$* . The top ends of the  $n$  lines correspond

to  $\pi$ . The bottom ends of the  $n$  lines correspond to the identical permutation  $(1, 2, \dots, n)$ . Each bar connects two consecutive vertical lines. See Fig. ???. Each number  $p_i$  in  $\pi$  starts the top end of line  $i$ , and goes down along the line, then whenever  $p_i$  comes to an end of a horizontal bar,  $p_i$  goes to the other end, then goes down again. Finally  $p_i$  reaches the bottom end of line  $p_i$ . This path is called *the  $p_i$ -route*. We can regard a bar as a modification of the “current” permutation. In a ladder lottery a sequence of such modifications always results in the identical permutation  $(1, 2, \dots, n)$ .

Let  $\pi = (p_1, p_2, \dots, p_n)$  be a permutation. A *reverse pair* in  $\pi$  is a pair  $(p_i, p_j)$  with  $p_i > p_j$  and  $i < j$ . A reverse pair is called “inversion” in the area of algebra. Let  $r$  be the number of reverse pairs in  $\pi$ . We can observe that any ladder lottery of  $\pi$  contains at least  $r$  bars, since each bar “cancels” exactly one reverse pair in the “current” permutation (see, e.g., [3, 5.3.4 Fig. 45]). If a ladder lottery  $L$  contains exactly  $r$  bars, then we say that  $L$  is *optimal*.

A *swap operation*, which corresponds the notion of “braid relation” in the area of algebra, is a local modification of a ladder lottery as shown in Fig. 4. Note that each dashed circle contains exactly three bars. Applying this modification to a ladder lottery of  $\pi$  results in other ladder lottery of  $\pi$ , since the local permutation consisting of the modified three bars remains as it was. A swap operation (a) to (b) in Fig. 4 is called a *left swap operation to bar  $b_u$* . Note that in Fig. 4 the left swap operation moves bar  $b_u$  from the (upper) right of the 5-route to the (lower) left, and to apply the left swap operation we need some route, say the 3-route, to be left-turned there. Similarly, a local swap operation (b) to (a) in Fig. 4 is called a *right swap operation to bar  $b_d$* . Note that the operation moves bar  $b_d$  from the left of the 5-route to the right.

A *redundant pair* is a pair of parallel bars appearing consecutively between the same pair of lines. In Fig. 5 examples are shown in the dashed circles. A *left warp operation* is a modification of a ladder lottery as shown in Fig. 5, in which (1) remove some redundant pair, then (2) append it at the lower left corner. The reverse operation is called a *right warp operation*.

## 3. The Family Tree

Let  $S_{\pi,k}$  be the set of all ladder lotteries of a given permutation  $\pi =$

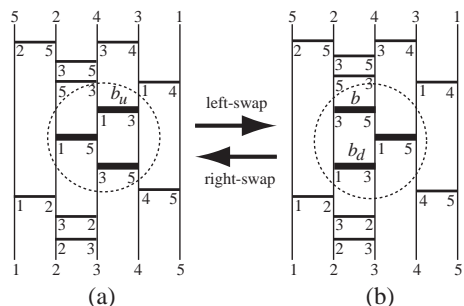


Fig. 4 A local swap operation.

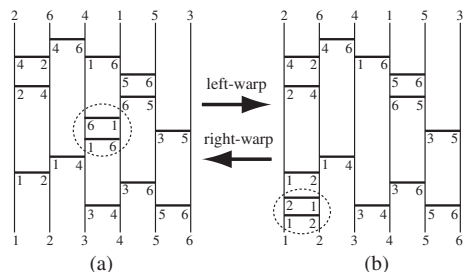


Fig. 5 A warp operation.

$(p_1, p_2, \dots, p_n)$  with exactly  $k$  bars. Assume that  $k$  is not smaller than the number of reverse pairs in  $\pi$ , and the parity of  $k$  and the number of reverse pairs match.

In this section we design a tree structure  $T_{\pi,k}$  among  $S_{\pi,k}$ , in which each vertex of  $T_{\pi,k}$  corresponds to a ladder lottery in  $S_{\pi,k}$ , and each edge corresponds to a relation between two ladder lotteries.

Assume  $S_{\pi,k} \neq \phi$ . Pick up any  $L_n \in S_{\pi,k}$ . Observe the  $n$ -route in  $L_n$ . The  $n$ -route partitions  $L_n$  into the upper part  $L_n^U$  and the lower part  $L_n^L$ . We say  $L_n$  is  $n$ -clean if (i)  $L_n^U$  has no bar and (ii) the  $n$ -route is  $x$ -monotone. If  $L_n$  is  $n$ -clean then removing the  $n$ -route from  $L_n$  then patching  $L_n^U$  and  $L_n^L$ , as shown in Fig. 6, results in a ladder lottery, say  $L_{n-1}$ , in  $S_{\pi',k-1}$ , where  $\pi'$  is the permutation derived from  $\pi$  by removing  $n$ . Similarly observe the  $(n-1)$ -

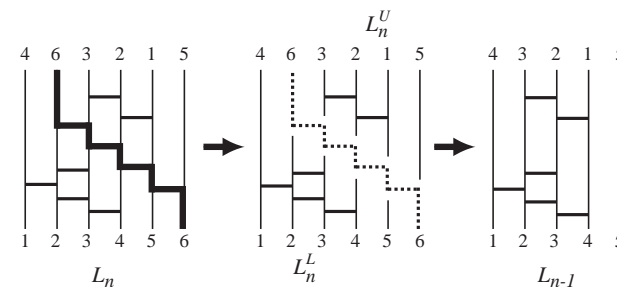


Fig. 6 The removal of the  $n$ -path.

route in  $L_{n-1}$ . We say  $L_{n-1}$  is  $(n-1)$ -clean if (i)  $L_{n-1}^U$  has no bar and (ii) the  $(n-1)$ -route is  $x$ -monotone. Repeat this process until some non-clean ladder lottery appears or  $L_2$  is derived. If  $L_i$  is  $i$ -clean for each  $i = 3, 4, \dots, n$ , then  $L_n$  is called the root lottery of  $S_{\pi,k}$ , denoted by  $R$  (See Figs 8 and 12). Otherwise we define the clean level of  $L_n$  as follows. The clean level of  $L_n$  is  $c$  if  $L_i$  is  $i$ -clean for  $i = n, n-1, \dots, c$  but not  $(c-1)$ -clean. Especially if  $L_n$  is not  $n$ -clean then the clean level of  $L_n$  is  $n+1$ , and the clean level of  $R$  is 3. Note that if  $L_n \in S_{\pi,k}$  has the clean level  $c$ , then  $n$ -route,  $(n-1)$ -route,  $\dots$ ,  $c$ -route form so called “a brick structure,” as follows.

For each  $p_a \geq c$  in  $\pi$ , let  $(q_1, q_2, \dots, q_b)$  be the decreasing list of numbers each of which is larger than  $p_a$  and locating to the left of  $p_a$  in  $\pi$ . In  $L_n$ , the  $p_a$ -route first go left  $b$  times, along the bars sharing with  $q_1$ -route,  $q_2$ -route,  $\dots$ ,  $q_b$ -route, then turn, then go right  $p_a - a + b$  times. Note that on the right side of the  $p_a$ -route with  $p_a \geq c$ , every  $x$ -route with  $x < c$  is  $x$ -monotone (left-down), every bar is on the  $y$ -route for some  $y > p_a$ , and there is no bar which can be left swap. (Otherwise there exists some route with left-turn, a contradiction.) Also either (1)  $L_n$  has at least one bar in the region below the  $c$ -route and above the  $(c-1)$ -route, or (2) the  $(c-1)$ -route is not  $x$ -monotone in  $L_{c-1}$ . See some examples in Fig. 7. The region is called the active region of  $L_n$ . Especially, we define the active region of  $R$  is  $\phi$  for convenience (in the proof of Lemma 3.2).

Now we assign a parent ladder lottery in  $S_{\pi,k}$  for each ladder lottery  $L_n$  in  $S_{\pi,k} - \{R\}$  as follows. We assume that  $L_n$  has the clean level  $c$ . Let  $AP$  (Active Path) be the maximal  $x$ -monotone subpath of the  $(c-1)$ -route ending at the

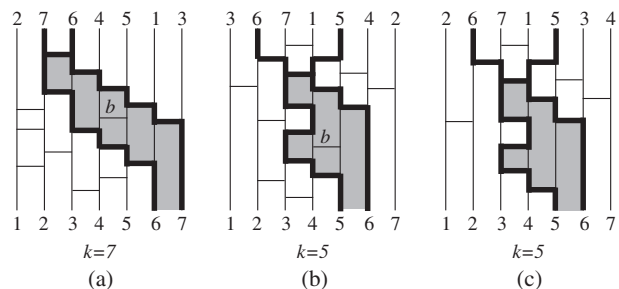


Fig. 7 The active regions.

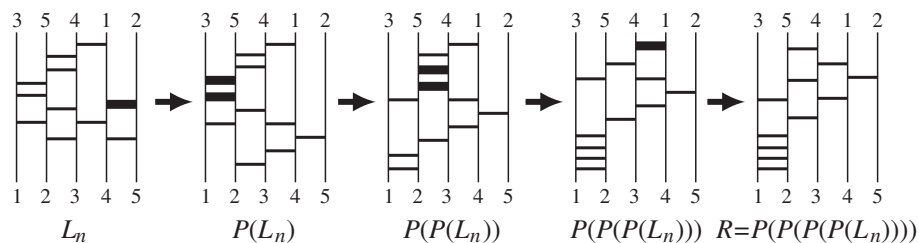


Fig. 8 The sequence of a ladder lottery  $L_n$  of (3,5,4,1,2) with exactly 11 bars.

bottom of line  $c - 1$ . Note that  $AP$  is never left-down. We say a bar  $b$  connecting line  $l$  and  $l + 1$  is *upward visible from  $AP$*  if (1) the lowest end of a bar on  $l$  above  $AP$  is the end of  $b$ , and (2) the lowest end of a bar on  $l + 1$  above  $AP$  is the end of  $b$ . Note that if  $b$  is upward visible from  $AP$  then  $b$  can be left-swapped and other upward visible bar from  $AP$  never has an end on line  $l$  nor  $l + 1$ . Thus the number of the upward visible bars from  $AP$  is at most  $\frac{n}{2}$ . Now we define the parent ladder lottery of  $L_n \in S_{\pi,k} - \{R\}$ , as follows. We have the following two cases.

**Case 1:** The active region has at least one visible bar from  $AP$ .

Among the upward visible bars from  $AP$ , the rightmost bar is called *the active bar* of  $L_n$ . In Figs. 7(a) and (b),  $b$  is the active bar. Apply the left swap operation to the active bar and let  $P(L_n)$  be the derived ladder lottery.

**Case 2:** The active region has no visible bar from  $AP$ .

Then at the left end of  $AP$  there exists a redundant pair, which is called *the active pair*. See Fig. 7(c). Apply the left warp operation to the active pair and let  $P(L_n)$  be the derived ladder lottery.

We say  $P(L_n)$  is *the parent ladder lottery* of  $L_n$ , and  $L_n$  is a *child ladder lottery* of  $P(L_n)$ . Note that the parent ladder lottery of  $L_n$  is unique, while  $P(L_n)$  may have many children. Also note that the clean level of  $P(L_n)$  is smaller than or equal to the clean level of  $L_n$ , and if they have the same clean level then  $P(L_n)$  has less or equal number of bars in the active region, and if they have the same number of bars in the active region then  $P(L_n)$  has shorter  $AP$ .

We have the following lemma.

**Lemma 3.1** For any  $L_n \in S_{\pi,k} - \{R\}$ ,  $P(L_n) \in S_{\pi,k}$  holds.

**Proof.** Since the each of the two operations to derive the parent reserves the permutation. □

Given a ladder lottery  $L_n$  in  $S_{\pi,k} - \{R\}$ , by repeatedly finding the parent ladder lottery of the derived ladder lottery, we can have the unique sequence  $L_n, P(L_n), P(P(L_n)), \dots$  of ladder lotteries in  $S_{\pi,k}$ , which eventually ends up with the root  $R$ . See Fig. 8. The active bars and the active pairs are depicted by thick lines.

We have the following lemma.

**Lemma 3.2** The sequence  $L_n, P(L_n), P(P(L_n)), \dots$  of  $L_n \in S_{\pi,k} - \{R\}$  ends with  $R \in S_{\pi,k}$ .

**Proof.** For each  $L_n \in S_{\pi,k}$  we define its *clean potential*  $C(L_n) = (s, t, u)$ , where  $s$  is the clean level of  $L_n$ ,  $t$  is the number of bars in the active region of  $L_n$  and  $u$  is the length of  $AP$ . For  $L_1, L_2 \in S_{\pi,k}$  with  $C(L_1) = (s_1, t_1, u_1)$  and  $C(L_2) = (s_2, t_2, u_2)$ , we say  $L_1$  is *cleaner than  $L_2$*  if (1)  $s_1 < s_2$ , (2)  $s_1 = s_2$  and  $t_1 < t_2$ , or (3)  $s_1 = s_2$ ,  $t_1 = t_2$  and  $u_1 < u_2$ . For any  $L_n \in S_{\pi,k}$  we can observe that  $P(L_n)$  is cleaner than  $L_n$  and  $R$  is the cleanest among  $S_{\pi,k}$ . Thus for any  $L_n \in S_{\pi,k}$  the sequence of clean potentials  $C(L_n), C(P(L_n)), C(P(P(L_n))), \dots$  always ends at  $C(R)$ . □

By merging all these sequences we can have *the family tree* of  $S_{\pi,k}$ , denoted by  $T_{\pi,k}$ , in which the root of  $T_{\pi,k}$  corresponds to  $R$ , the vertices of  $T_{\pi,k}$  correspond

to the ladder lotteries in  $S_{\pi,k}$  and each edge corresponds to a relation between a ladder lottery in  $S_{\pi,k}$  and its parent. See Fig. 3. The active bars and the active pairs are depicted by thick lines.

#### 4. Enumerating All Ladder Lotteries

In this section we give an efficient algorithm to enumerate all ladder lotteries in  $S_{\pi,k}$ .

If we have an algorithm to enumerate all children of a given ladder lottery in  $S_{\pi,k}$ , then by recursively applying the algorithm starting at the root  $R$  of  $S_{\pi,k}$ , we can enumerate all ladder lotteries in  $S_{\pi,k}$ . Now we design such an algorithm.

We need some definitions. Let  $L_n \in S_{\pi,k}$  with  $\pi = (p_1, p_2, \dots, p_n)$ . Assume  $L_n$  has the clean level  $c$ . So each bar locating on the right of the  $c$ -route is contained in some  $x$ -route with  $x > c$ , but either (1) in the active region (See Fig. 7) there is at least one bar which is not contained in any  $x$ -route with  $x \geq c - 1$  or (2) the  $(c - 1)$ -route is not  $x$ -monotone. Each  $x$ -route with  $x \geq c$  goes left along bars (sharing with larger routes), “turns,” then goes right along bars (sharing with smaller routes). For each  $x$ -route with  $x \geq c$ , if  $b$  is the first bar to go right after bars to go left, then  $b$  is called *the turn bar* of  $x$ . Note that only if the  $x$ -route contains both at least one bar to left and one bar to right, the  $x$ -route has the turn bar. Otherwise if the  $x$ -route contains only bars to left (or right) then the  $x$ -route is  $x$ -monotone in  $L_n$  and has no turn bar. Also note that the turn bar is defined only for the  $x$ -route with  $x \geq c$ . In the next lemma we show that on the  $x$ -route with  $x \geq c$ , only turn bars has a chance to be right swapped.

**Lemma 4.1** *Let  $L_n$  be a ladder lottery having the clean level  $c$ . On the  $x$ -route with  $x \geq c$  only the turn bar has a chance to be right swapped.*

**Proof.** Since  $L_n$  has the clean level  $c$ , the  $x$ -route of each  $x \geq c$  first goes left along bars, turns, then goes right along bars. See Fig. 4(b). A bar  $b_d$  can be right swapped only if the vertical segment between the left end of  $b_d$  and the left end of  $b$  has no right end of other bars. Here the route passes  $b$ , then left-turn, then passes  $b_d$ . Thus on the  $x$ -route with  $x \geq c$ , only the turn bar has a chance to satisfy the condition. See an example in Fig. 9. The turn bars of the 9-route and the 7-route can be right swapped, but the turn bar of the 8-route can not be.  $\square$

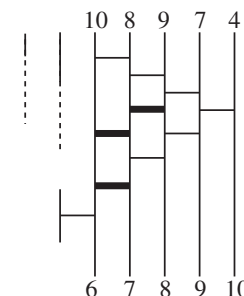


Fig. 9 Illustration for Lemma 4.1.

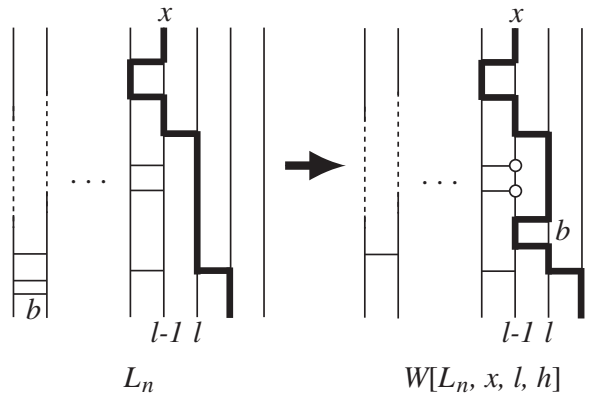
Let  $S[L_n, b]$  be the ladder lottery derived from  $L_n$  by applying the right swap operation to a bar  $b$ . If  $L_n \in S_{\pi,k}$  has the clean level  $c$  and has a redundant pair  $b$  at the lower left corner of  $L_n$  then let  $W[L_n, x, l, h]$  be the ladder lottery derived from  $L_n$  by a right warp operation to  $b$  so that (1)  $b$  has ends at line  $l - 1$  and  $l$ , (2)  $b$  is on the  $x$ -route of  $W[L_n, x, l, h]$ , (3)  $b$  appears at the left end of  $AP$  of  $W[L_n, x, l, h]$ , and (4) the vertical segment between the left end of  $b$  and the left end of the bar on the  $x$ -route above  $b$  has exactly  $h$  right ends of bars in  $W[L_n, x, l, h]$ . See Fig. 10, where  $h = 2$  corresponds the two white circles. Every child of  $L_n$  is either  $S[L_n, b]$  for some  $b$  or  $W[L_n, x, l, h]$  for some  $x, l$  and  $h$ , but not all  $S[L_n, b]$  or  $W[L_n, x, l, h]$  are children of  $L_n$ .  $S[L_n, b]$  is a child of  $L_n$  only if  $b$  is the active bar of  $S[L_n, b]$ . Also  $W[L_n, x, l, h]$  is a child of  $L_n$  only if  $b$  is the active pair of  $W[L_n, x, l, h]$ .

Now we first classify each  $S[L_n, b]$  into children of  $L_n$  and non-children, as follows. Remember the clean level of  $L_n$  is  $c$ . Let  $R(i)$  be the region on the right side of the  $i$ -route, and  $L(i)$  be the region on the left side of the  $i$ -route. We have two types.

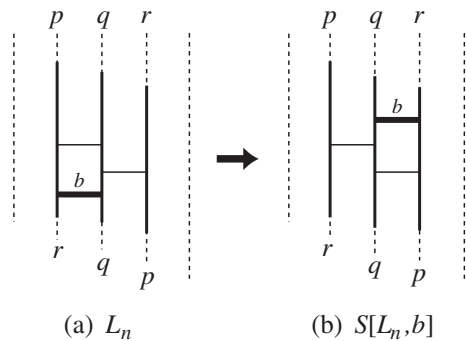
**Type 1:**  $b$  is a turn bar.

If  $b$  can not be right swapped then  $S[L_n, b]$  is not defined. Assume otherwise.

Note that such a bar exists only on the  $x$ -routes with  $x \geq c$ . Assume that  $b$  is on the  $q$ -route, and in  $L_n$  the routes of  $p, q, r$  pass through as shown in Fig. 11(a). Since  $b$  is a turn bar of the  $q$ -route,  $q \geq c$  and  $p > q > r$  hold.



**Fig. 10** Illustration for  $W[L_n, x, l, h]$ .



**Fig. 11** Illustration for Type 1.

Now  $S[L_n, b]$  is not  $p$ -clean since  $b$  in  $S[L_n, b]$  is not on the  $x$ -route with  $x > p$ . Thus the clean level of  $S[L_n, b]$  is increased to  $p + 1$ , and  $b$  is the only bar in the active region of  $S[L_n, b]$ . Since  $b$  is upward visible from  $AP$ , which is the maximal  $x$ -monotone subpath of the  $p$ -route ending at the bottom of line  $x$ ,  $b$  is the active bar of  $S[L_n, b]$ . Thus  $L[L_n, b]$  is a child of  $L_n$ . Otherwise  $S[L_n, b]$  is not a child of  $L_n$ .

**Type 2:**  $b$  can be right swapped but  $b$  is not a turn bar.

Such a bar, say  $b$ , exists only in  $L(c) \cap L(c+1) \cap \dots \cap L(n)$ , and  $b$  is “downward” visible from some  $x$ -route, and a right swap operation to  $b$  moves  $b$  to  $R(x)$  crossing the  $x$ -route.

Note that the left boundary of  $L(c) \cap L(c+1) \cap \dots \cap L(n)$  is  $x$ -monotone. If the right swap operation moves  $b$  to  $R(x)$  crossing the  $x$ -route with  $x \geq c$ , then the clean level of  $S[L_n, b]$  is  $x + 1$  and  $b$  is the only bar in the new active region and  $b$  is upward visible from  $AP$ , so  $b$  is the active bar of  $S[L_n, b]$ . Thus  $S[L_n, b]$  is a child of  $L_n$ .

If the right swap operation moves  $b$  to  $R(c - 1)$ , crossing  $AP$ , which is the maximal  $x$ -monotone subpath of the  $(c - 1)$ -route ending at the bottom of line  $c - 1$ , then the clean level of  $S[L_n, b]$  remains  $c$ , and  $b$  is appended to the active region. Assume the active bar of  $L_n$  has the left end on line  $s$  and  $b$  in  $L_n$  has the right end on line  $t$ . So  $b$  in  $S[L_n, b]$  has the right end on line  $t + 1$ . If  $t + 1 \geq s$ , then  $b$  is the active bar of  $S[L_n, b]$ , otherwise  $b$  is not. Thus  $S[L_n, b]$  is a child of  $L_n$  if and only if  $t + 1 \geq s$ .

If the right swap operation moves  $b$  to  $R(c - 1)$ , crossing the  $(c - 1)$ -route but not  $AP$ , then  $b$  is not upward visible from  $AP$ . Thus  $S[L_n, b]$  is not a child of  $L_n$ .

Otherwise the right swap operation moves  $b$  to  $R(x)$  crossing the  $x$ -route with  $x < c - 1$ . The clean level of  $S[L_n, b]$  remains  $c$ , and  $b$  is not the active bar in  $S[L_n, b]$ . Thus  $S[L_n, b]$  is not a child of  $L_n$ .

Also we classify each  $W[L_n, x, l, h]$  into children of  $L_n$  and non-children, as follows.

**Type 3:**  $W[L_n, x, l, h]$ .

If  $L_n$  has no redundant pair at the lower left corner of  $L_n$  then  $W[L_n, x, l, h]$  is not defined. Assume otherwise. Let  $c$  be the clean level of  $L_n$ .

Each  $x$ -route with  $x \geq c$  goes left along bars, “turns,” then goes right along bars. For each  $x \geq c$ , let  $u_x$  be the leftmost line on which the  $x$ -route passes. Note that  $AP$  is never left-down.

Now for each  $x \geq c$  and each  $l$ , where  $x \geq l \geq \max(u_x, 2)$ , and each possible

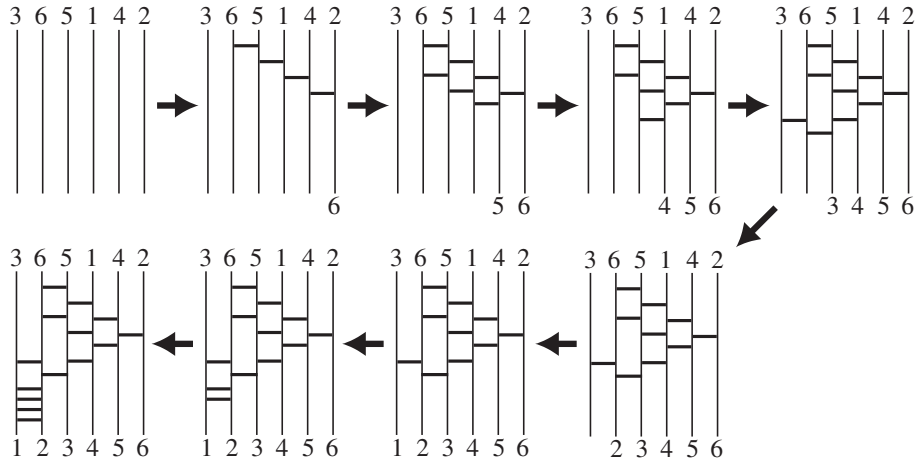


Fig. 12 Computation of the root  $R$  in  $S_{\pi,k}$ , where  $\pi = (3, 6, 5, 1, 4, 2)$  and  $k = 14$ .

$h$ ,  $W[L_n, x, l, h]$  is a child of  $L_n$ . For  $l$  with  $x \geq l > u_x$ ,  $W[L_n, x, l, h]$  is derived from  $L_n$  by removing a redundant pair at the lower left corner, then replace a suitable bar on the  $x$ -route by triple bars. For  $l = u_x$ ,  $W[L_n, x, l, h]$  is derived from  $L_n$  by removing a redundant pair at the lower left corner, then appened a redundant pair so that the redundant pair appears at the left end of  $AP$  of  $W[L_n, x, l, h]$

For  $x = c - 1$  we need to check more carefully. Assume  $AP$  of  $L_n$  has left end on line  $u$ . If  $L_n$  has no active bar, then for each  $l$  with  $c - 1 \geq l \geq \max(u, 2)$  and each possible  $h$ ,  $W[L_n, c - 1, l, h]$  is a child of  $L_n$ .

If  $L_n$  has the active bar  $b$ , then assume it has the left end on line  $s$ . Note that we have  $u < s$ . Then for each  $l$  with  $c - 1 \geq l \geq s$  and each possible  $h$ ,  $W[L_n, c - 1, l, h]$  is a child of  $L_n$ . Note that the left end of  $AP$  of  $W[L_n, c - 1, l, h]$  is on line  $l - 1$ . For each  $l$  with  $s - 1 \geq l \geq \max(u, 2)$ ,  $W[L_n, c - 1, l, h]$  still has the active bar  $b$ , so the parent of  $W[L_n, c - 1, l, h]$  is not  $L_n$ , and  $W[L_n, c - 1, l, h]$  is not a child of  $L_n$ .

For  $x < c - 1$ ,  $W[L_n, x, l, h]$  is not a child of  $L_n$ .

Now we have the following algorithm.

**Procedure**  $\text{find-all-children}(L_n, c, a)$

{  $L_n$  is the current ladder lottery,  $c$  is the clean level of  $L_n$ , and  $a$  is the active bar or the active pair, and  $s$  is the line on which  $a$  has the right end. }

**begin**

```

01 Output  $L_n$  {Output the difference from the previous one.}
02 for each  $x \geq c$ 
03 begin
04   if the turn bar  $b$  of the  $x$ -route is right swappable
05     then  $\text{find-all-children}(S[L_n, b], x + 1, b)$ 
06   for each downward visible (non-turn) bar  $b$  from the  $x$ -route
07      $\text{find-all-children}(S[L_n, b], x + 1, b)$ 
08 end
09 for each downward visible bar  $b$  from  $AP$ 
    { $b$  in  $L_n$  has the right end on line  $t$ .}
10   if  $t \geq s - 1$  then  $\text{find-all-children}(S[L_n, b], c, b)$ 
11 if  $L_n$  has no redundant pair at the lower left corner then return
12 for each  $x \geq c$ 
    { $u_x$  is the leftmost line on which the  $x$ -route passes.}
13   for each  $l = \max(u_x, 2)$  to  $x$ 
14     for each possible  $h$ 
15        $\text{find-all-children}(W[L_n, x, l, h], x + 1, \phi)$ 
16 if  $L_n$  has no active bar
    { $u$  is the line on which  $AP$  of  $L_n$  has left end.}
17   for each  $l = \max(u, 2)$  to  $c - 1$ 
18     for each possible  $h$ 
19        $\text{find-all-children}(W[L_n, x, l, h], c, \phi)$ 
20 else
21   for each  $l = s$  to  $c - 1$ 
22     for each possible  $h$ 
23        $\text{find-all-children}(W[L_n, x, l, h], c, \phi)$ 
end

```

```

Algorithm find-all-ladder-lotteries( $\pi = (p_1, p_2, \dots, p_n)$ )
begin
01  Compute  $R$ 
02  for each  $x \geq 2$ 
03    if the turn bar  $b$  of the  $x$ -route is right swappable
04      then find-all-children( $S[R, b], x + 1, b$ )
05  if  $L_n$  has no redundant pair at the lower left corner then return
06  for each  $x \geq 2$ 
07    for each  $l = \max(u_x, 2)$  to  $x$ 
08      for each possible  $h$ 
09    find-all-children( $W[R, x, l, h], c, \phi$ )
end

```

By maintaining (i) the clean level  $c$ , (ii) the list of downward visible bar from  $x$ , for each  $x \geq c - 1$  (those are candidate bars to be right swapped, crossing the  $x$ -route), (iii) the active bar, and (iv) the current ladder lottery, we can enumerate all children of  $L_n$  in  $O(1)$  time for each on average.

**Lemma 4.2** *One can enumerate all children of  $L_n$  in  $O(1)$  time for each on average.*

From the above lemma, we obtain the following theorem.

**Theorem 4.3** *After computing and outputting the root  $R$  in  $S_{\pi,k}$  in  $O(n + k)$  time, the algorithm runs in  $O(|S_{\pi,k}|)$  time. The algorithm uses  $O(n + k)$  working space.*

**Proof.** We show that the root  $R$  in  $S_{\pi,k}$  can be generated in  $O(n + k)$  time. See Fig. 12 for a sketch. We start with  $n$  vertical lines. Then we append the  $j$ -route for each  $j = n, n - 1, \dots, 3$ . Each  $j$ -route goes left with some bars, turns, then goes right with some bars. When we append the  $j$ -route the part of route to go left is already completed, since those bars correspond to the crossing with the routes of larger numbers. So we only need to append the part to go right, consisting of  $x$ -monotone path. Finally we append the redundant pairs at the lower left corner. Thus we can compute  $R$  in  $O(n + k)$  time and space.  $\square$

By the theorem above, our algorithm generates each ladder lottery in  $S_{\pi,k}$  in  $O(1)$  time “on average.” However it may have to return from the deep recursive

calls without outputting any ladder lottery in  $S_{\pi,k}$ , after generating a ladder lottery corresponding to the rightmost leaf of a large subtree in the family tree. Therefore the next ladder lottery in  $S_{\pi,k}$  cannot be generated in  $O(1)$  time in worst case.

By modifying the algorithm so that each ladder lottery at “even” depth in  $T_{\pi,k}$  is output “before” its children, and each ladder lottery at “odd” depth in  $T_{\pi,k}$  is output “after” its children [6], we can output the next ladder lottery in  $O(1)$  time in worst case.

**Theorem 4.4** *After computing and outputting the root  $R$  in  $S_{\pi,k}$  in  $O(n + k)$  time, the algorithm enumerates all ladder lotteries in  $S_{\pi,k}$  in  $O(1)$  time for each. The algorithm uses  $O(n + k)$  working space.*

## 5. Conclusion

In this paper, we gave an algorithm to enumerate all ladder lotteries of a given permutation  $\pi$  with exactly  $k$  bars. Our algorithm uses  $O(n + k)$  space and enumerates all ladder lotteries in  $S_{\pi,k}$  in  $O(1)$  time for each in worst case.

## References

- 1) D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Appl. Math.*, 65(1-3):21–46, 1996.
- 2) D.E. Knuth. *Axioms and hulls*. LNCS 606, Springer-Verlag, 1992.
- 3) D.E. Knuth. *The art of computer programming*. 3, Addison-Wesley, 2nd edition, 1998.
- 4) Z. Li and S. Nakano. Efficient generation of plane triangulations without repetitions. *Proc. The 28th International Colloquium on Automata, Languages and Programming, (ICALP 2001)*, LNCS 2076:433–443, 2001.
- 5) S. Nakano. Efficient generation of triconnected plane triangulations. *Comput. Geom. Theory and Appl.*, 27(2):109–122, 2004.
- 6) 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.
- 7) 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.