IPSJ SIG Technical Report Vol.2012-AL-142 No.10
2012/11/2

Coding Ladder Lotteries

Tomoki Aiuchi[1,2] Katsuhisa Yamanaka[1,a)] Takashi Hirayama[1,b)] Yasuaki Nishitani[1,c)]

Abstract: A *ladder lottery*, known as the "Amidakuji" in Japan, is a common way to choose a permutation randomly. In this paper, we consider a problem of coding an optimal ladder lotteries. We first propose two codes for an optimal ladder lottery. The more compact codes among them needs at most $n + 2b - 1$ bits, where n is the number of vertical lines and b is the number of horizontal lines in an optimal ladder lottery. As a by-product of this result, we obtain an upper bound on the number of optimal ladder lotteries with n vertical lines and b horizontal lines. Furthermore we improve the code and experimentally show that the average length of the improved code is more compact.

Keywords: algorithm, encoding, decoding, ladder lottery, optimal ladder lottery

1. Introduction

Coding discrete objects is appealing theme to give a compact representation of them in a computer. Recently graph coding are widely investigated. For instance, compact codings for trees [2], [8], [10], [13], triangulations [1], [14], [15] and planar graphs [3], [4], [11], [13] have been presented.

In this paper we consider a problem of coding ladder lotteries. A *ladder lottery*, known as the "Amidakuji" in Japan, is a common way to choose a permutation randomly. Formally, a ladder lottery L of a permutation $\pi = (p_1, \ldots, p_n)$ is a network with n vertical lines (*lines* for short) and many horizontal lines (*bars* for short) each of which connects two consecutive vertical lines. See Fig. 1 for an example. The ith line from the left is called *line i*. The top endpoints of lines correspond to π. The bottom endpoints of lines correspond to the identical permutation $(1, \ldots, n)$. Each number p_i in π starts the top endpoint of line i, and goes down along the line, then whenever p_i comes to an endpoint of a bar, p_i goes horizontally along the bar to the other endpoint, then goes down again. Finally p_i reaches the bottom endpoint of line p_i. This path is called the *route* of number p_i. We can regard a bar as a modification of the current permutation, and a sequence of such modifications in a ladder lottery always results in the identical permutation $(1, \ldots, n)$.

A ladder lottery of a permutation $\pi = (p_1, \ldots, p_n)$ is *optimal* if it consists of the minimum number of bars among ladder lotteries of π. Let L be an optimal ladder lottery of π and b the number of bars in L. Then we can observe that b is equal to the number of "inversions" of π, which is a pair (p_i, p_j) in π with $p_i > p_j$ and $i < j$. The ladder lottery in Fig. 1 has thirteen bars, and permutation (6,4,3,5,2,1) has thirteen inversions: (6,4), (6,3), (6,5), (6,2), (6,1), (4,3), (4,2), (4,1), (3,2), (3,1), (5,2), (5,1), and (2,1), so the ladder lottery is optimal.

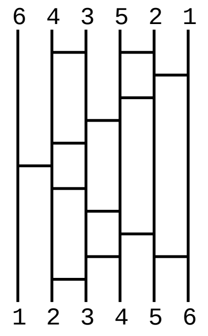

Fig. 1 An optimal ladder lottery of the permutation (6,4,3,5,2,1).

In this paper we present several codes for optimal ladder lotteries with n lines and b bars. The length of the one of our codes is $n + 2b - 1$ bits. We improve the code and experimentally show that the average length of the improved code is compact.

The ladder lotteries are strongly related to primitive sorting networks, which are deeply investigated by Knuth [12]. A comparator in a primitive sorting network replaces p_i and p_{i+1} by $\min(p_i, p_{i+1})$ and $\max(p_i, p_{i+1})$, while a bar in a ladder lottery always exchanges them.

Optimal ladder lotteries are also related to pseudoline arrangements. A pseudoline is an x-monotone curve in the plane, and an arrangement is a set of pseudolines in which every pair of pseudolines intersects exactly once. Arrangements of pseudolines are one of important and appealing objects in the area of geometry and combinatorics. Arrangements of pseudolines are the topic of a chapter in a representative handbook on computational geometry [9]. There is one-to-one correspondence between optimal ladder lotteries of $\pi = (n, \ldots, 1)$, namely a reverse permutation, and arrangements of n pseudolines [16].

Coding pseudoline arrangements has been investigated to obtain upper bounds of the number of pseudoline arrangements. Knuth [12] showed that an arrangement of n pseudolines can be encoded into $0.792n^2$ bits in 1992. This result immediately im-

[1] Department of Electrical Engineering and Computer Science, Iwate University, Ueda 4-3-5, Morioka, Iwate 020-8551, Japan.
[2] Currently he is a member of Leadkonan, Kitaoniyanagi 16-chiwari 164, Kitakami, Iwate 024-0072, Japan.
a) yamanaka@cis.iwate-u.ac.jp
b) hirayama@kono.cis.iwate-u.ac.jp
c) nisitani@iwate-u.ac.jp

plies that the number of arrangements of n pseudolines is bounded by $2^{0.792n^2}$. Felsner [5], [6] improves the length to $0.697n^2$ bits. Recently, Felsner and Valtr [7] improves to $0.657n^2$ bits.

In this paper we design codes for optimal ladder lotteries with n lines and b bars. A class of optimal ladder lotteries with n lines is a generalization of a class of arrangements of n pseudolines. For an optimal ladder lottery, $b \leq \binom{n}{2}$ holds, and $b = \binom{n}{2}$ if and only if the optimal ladder lottery corresponds to a reverse permutation, that is the optimal ladder lottery correspond to a pseudoline arrangement.

We first propose two codes. The more compact code among them needs at most $n + 2b - 1$ bits. By estimating an upper bound on the number of every possible code, we obtain an upper bound on the number of ladder lotteries with n lines and b bars. Our bound is not tighter than the bound by [7] for arrangement of n pseudolines. However, our bound can be applied for a class of optimal ladder lotteries with n lines and $b \leq \binom{n}{2}$ bars, which includes a class of arrangements of n pseudolines.

Furthermore, we improve the code and experimentally show that the average length of the improved code is more compact, and our codes contain no end-of-file.

Throughout of this paper, we assume that the end-of-file is prepared as a special character.

The paper organized as follows. Sections 2 and 3 define a route-based code and a line-based code for optimal ladder lotteries, respectively. In Section 4, we improve the line-based code proposed in Section 3. In Section 5, we compare proposed codes experimentally. Section 6 is a conclusion.

2. A Route-based Code

The coding idea for pseudoline arrangements by Felsner [5], [6] can be applied for optimal ladder lotteries easily. In this section we propose a code which is immediately derived from his coding [5], [6].

The coding idea is to represent each route as a binary code. The route turns left or right zero or more times. We encode the directions (left or right) into a binary code.

Let L be an optimal ladder lotteries of a permutation $\pi = (p_1, \ldots, p_n)$. We encode the route of p_i with a binary code as follows. Suppose the route of p_i turns left or right $t(i)$ times. We note that $t(i) \leq n - 1$ holds. Define d^i_j = ‘0’ if jth direction of the route of p_i is left, and d^i_j = ‘1’ if jth direction of the route of p_i is right. We construct a binary code $d^i_1 \ldots d^i_{t(i)}$ by arranging d^i_js, and then append ‘0’s so that the length of the code becomes $n - 1$ bits in total. We call the code a *route-code* of p_i and denote by $R(i) = d^i_1 \ldots d^i_{n-1}$. Route-codes for the optimal ladder lottery in Fig. 1 are $R(1)$ = “11111”, $R(2)$ = “1101$\underline{0}$”, $R(3)$ = “0101$\underline{0}$”, $R(4)$ = “110$\underline{00}$”, $R(5)$ = “01000”, and $R(6)$ = “00000”. The underlined ‘0’s are appended zeros.

We define a code for L. A *route-based code*, denoted by RC, for L is the code obtained by arranging route-codes of $p_1, \ldots, p_n$ in this order. The length of RC is $n(n - 1)$ bits. For instance, a route-based code for the optimal ladder lottery in Fig. 1 is “111111101$\underline{0}$0101$\underline{0}$110$\underline{00}$0100000000”. The underlined ‘0’s are appended zeros.

Now we explain how to reconstruct the original ladder lottery from a route-based code.

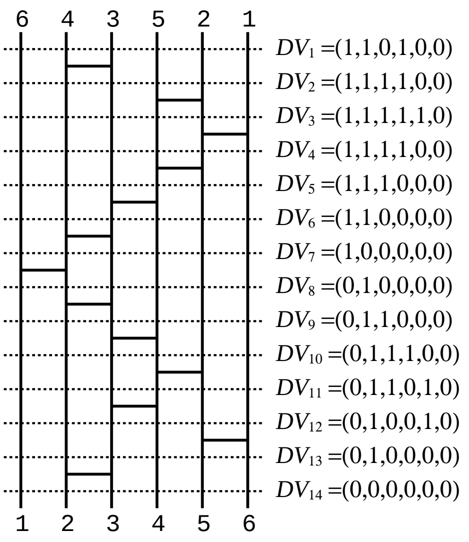

Fig. 2 Direction vectors in reconstruction.

Let RC be a route-based code for an optimal ladder lottery L with n lines. Since we know the length of RC is $n(n - 1)$ bits, we can calculate the value of n by computing the length of RC and hence recognize the boundary between any two consecutive route-codes. Let focus on the first bit in each route-code. It represents the first direction of each route. Arranging the first bits in route-codes, we define a *direction vector* $DV_1 = (d^1_1, \ldots, d^n_1)$. For example, DV_1 for the ladder lottery in Fig. 1 is shown in Fig. 2. Intuitively, DV_1 represents the next direction of each route from the top in a ladder lottery.

Now, we show that a bar can be reconstructed from DV_1. In DV_1, j is *flip* if d^j_1 = ‘1’ and d^{j+1}_1 = ‘0’ holds. This represents an intersection, namely a bar, of the two routes p_j and p_{j+1}. DV_1 may contain multiple flips. We call the minimum flip the *first flip*. Intuitively, the first flip in DV_1 corresponds to the upper-left bar in L.

Let j be the first flip. We can reconstruct the bar in which two routes of p_j and p_{j+1} intersects. Then we construct DV_2 from DV_1 by passing through the bar. That is, we replace d^j_1 with d^j_2 and d^{j+1}_1 with d^{j+1}_2, and swap d^j_2 with d^{j+1}_2. DV_2 represents the next directions of routes after passing the bar corresponding to the first flip in DV_1. See Fig. 2. By repeatedly applying this process for $DV_2, \ldots, DV_b$, we can reconstruct all bars. Fig. 2 shows direction vectors constructed in a reconstruction. A direction vector may contain d^j_n which is not defined in route-code of p_j. For convenience, we assume that d^j_n = ‘0’. The last direction vector contains only ‘0’s.

Now we show the correctness of the above reconstruction. A problem is existences of appended ‘0’s. A ‘0’ in a direction vector means either a direction or the bottom endpoint of a route. Can we recognize a correct meaning of any ‘0’?

Let $DV_i = (v_1, \ldots, v_n)$ be a ith direction vector, $i \leq b$. First we show that a flip corresponds to a bar.

Lemma 2.1 Let v_j be any flip in DV_i. Then v_j and v_{j+1} corresponds to a bar.

Proof. Since v_j is a flip, v_j = ‘1’ and v_{j+1} = ‘0’. Hence a route corresponding to v_j goes from a line j to a line $j + 1$. Then

v_{j+1} = ‘0’ correspond to a direction. If v_{j+1} = ‘0’ corresponds to a bottom endpoint of a route, then the two routes corresponding to v_j and v_{j+1} have no corresponding intersection, which is a contradiction. ■

Lemma 2.1 implies that we correctly recognize a meaning of any ‘0’ in the reconstruction. A ‘0’ for a first flip is always a direction of a route.

Next we show an existence of a flip in DV_i.

Lemma 2.2 DV_i, $i \le b$, contains at least one flip.

Proof. We assume that there is no flip in DV_i. If DV_i contains only ‘0’s, then all bars are reconstructed, thus $i = b + 1$ holds. Otherwise there always exists some k such that v_j = ‘0’ if $j < k$ and v_j = ‘1’ if $k \le j$. We note that $j = 0$ holds if DV_i contains only ‘1’s. Then, the line n has a left endpoint of a bar, which is a contradiction. ■

We obtain the following lemma for reconstruction.

Lemma 2.3 One can reconstruct the original ladder lottery from a route-based code.

Proof. Immediate from Lemmas 2.1 and 2.2. ■

We have the following theorem.

Theorem 2.4 We can encode an optimal ladder lottery into a code of length $n(n-1)$ bits. Coding and decoding can be done in $O(n^2)$ time.

Proof. Coding can be done in $O(n^2)$ time in straightforward way.

In the reconstruction, we need to find the first flip in each direction vector. This step takes $O(n)$ time by a linear search on a direction vector. Therefore, this straightforward way takes $O(nb)$ time for reconstruction. We improve the running time. The idea is to maintain all flips by a list. First we construct a list consisting of all flips in DV_1 in $O(n)$ time. Then we maintain the list for DV_i for $i = 1, \ldots, b$. To construct DV_{i+1} from DV_i, at most two elements, which are the first flip and its next index in DV_i, are updated. Hence, an update for the list can be done in $O(1)$ time. Therefore, the running time to reconstruct all bars is $O(b)$ time. As an initialization, we calculate the value of n by reading the whole of the code. This takes $O(n^2)$ time. Therefore, the reconstruction can be done in $O(n^2)$ time. ■

3. A Line-based Code

We propose a new code for an optimal ladder lottery in this section. This code focuses on lines in an optimal ladder lottery. We represent each line as a sequence of endpoints of bars.

Let L be an optimal ladder lottery with n lines and b bars. We denote by $V(i) = (v_1, \ldots, v_{|V(i)|})$ a sequence of endpoints of bars on a line i from the top to the bottom. We encode $V(i)$ with $C(i) = (c_1 \ldots c_{|V(i)|})$ where c_j = ‘0’ if v_j is a right endpoint on i and c_j = ‘1’ if v_j is a left endpoint on i. We call $C(i)$ a *line-code* for a line i. Finally we concatenate line-codes $C(i)$ for $i = 1, \ldots, n$ in this order, and insert ‘0’ between any consecutive two line-codes to represent a boundary. We call the obtained code a *line-based code*, and denote by LC. For example, line-codes in the ladder lottery in Fig. 1 are $C(1)$ = “1”, $C(2)$ = “11011”, $C(3)$ = “0100110”, $C(4)$ = “110010”, $C(5)$ = “01001”, and $C(6)$ = “00”. Its line-based code is “1$\underline{0}$11011$\underline{0}$0100110$\underline{0}$110010$\underline{0}$01001$\underline{0}$00”. The underlined ‘0’s represent boundaries between two consecutive line-codes.

Now we explain how to reconstruct the original optimal ladder lottery from a line-based code. Let LC be a line-based code for an optimal ladder lottery L.

In LC, a ‘0’ represents either a right endpoint of a bar or a boundary of consecutive two line-codes. A key of reconstruction is to recognize boundaries in line-codes. If the boundaries in LC are recognized, then it is easy to reconstruct each line i for $i = 1, \ldots, n$ from the corresponding line-code. Fig. 3 shows an example of reconstruction of the optimal ladder lottery in Fig. 1 from its line-codes. We explain how to recognize boundaries.

Since the line 1 contains only left endpoints, the first consecutive ‘1’s followed by one ‘0’correspond to $C(1)$ and the boundary between $C(1)$ and $C(2)$. Now we assume that the boundary between line-codes $C(i-1)$ and $C(i)$ is recognized and the line $i-1$ is reconstructed. Then, we know the number of left endpoints of bars on a line $i-1$. Since it equals to the number right endpoints of bars on a line i, we obtain the number of ‘0’s in $C(i)$. Hence, the boundary between line-codes $C(i)$ and $C(i+1)$ can be recognized in LC, and a line i is reconstructed from $C(i)$.

We estimate the length of LC for an optimal ladder lottery with n lines and b bars. Since LC contains 2 bits for each bar and 1 bit for each boundary, its length is $n + 2b - 1$ bits.

Theorem 3.1 Let L be an optimal ladder lottery with n lines and b bars. There exists a code for L of length $n + 2b - 1$ bits. Coding and decoding can be done in $O(n + b)$ time.

As a byproduct, we obtain the following corollary.

Corollary 3.2 The number of optimal ladder lotteries with n lines and b bars is at most $\binom{n+2b-1}{b}$.

Proof. A line-based code includes b ‘1’s. Hence the number of every possible line-code for an optimal ladder lottery with n lines and b bars is at most $\binom{n+2b-1}{b}$. ■

4. Improvements

In this section we design a new code by improving the line-based code in the previous section. The lengths of the proposed codes are at most $n + 2b - 1$ bits. On the other hand, we show that the new codes are more compact than the original line-based code by an experiment in the next section.

First, we introduce three improvement ideas. Based on these ideas, we propose a new code.

Let L be an optimal ladder lottery with n lines and b bars, and LC its line-based code.

Improvement 1

The first improvement idea is to save the line-code for the line n. There are only right endpoints of bars on n, and the number of right endpoints of bars on n is equal to the number of left endpoints of bars on $n-1$. Hence, even if the line-code for n and its preceding ‘0’ corresponding to the boundary in LC are omitted, we can reconstruct the original optimal ladder lottery by computing the number of right endpoints on n from a line-code for $n-1$.

Improvement 2

The following property of an optimal ladder lottery is useful for saving bits.

Lemma 4.1 Let x, y be two consecutive bars with left endpoints

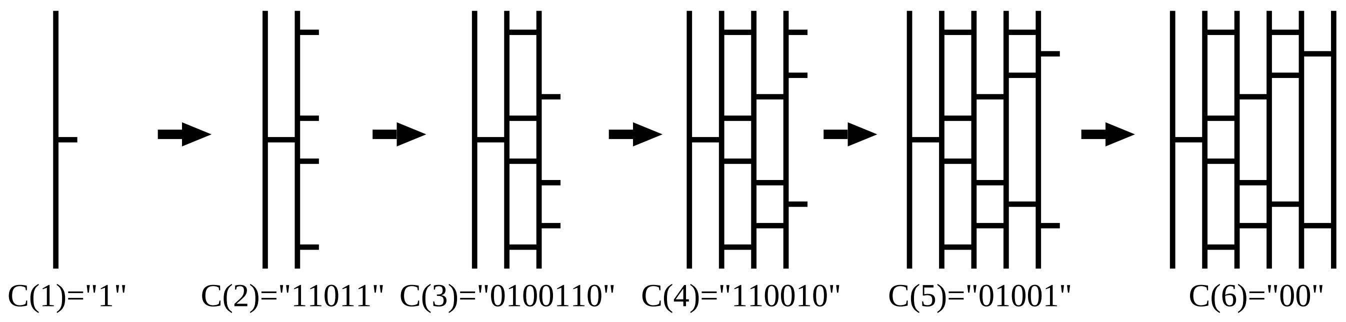

Fig. 3 An example of reconstruction from line-codes

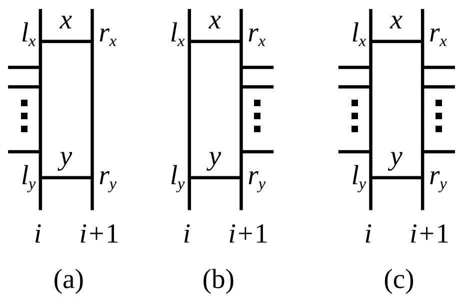

Fig. 4 Possible situations for consecutive two bars.

on a line i and right endpoints on a line $i+1$ in an optimal ladder lottery (See Fig. 4). We denote by l_x, r_x the left and the right endpoints of x. Similarly, we denote by l_y, r_y the left and the right endpoints of y. A configuration between the two bars is either of three cases: (1) there is at least one right endpoint between l_x and l_y and no endpoint between r_x and r_y (Fig. 4(a)), (2) there is no endpoint between l_x and l_y and at least one left endpoint between r_x and r_y (Fig. 4(b)), or (3) there is at least one right endpoint between l_x and l_y and at least one left endpoint between r_x and r_y (Fig. 4(c)).

Proof. We assume that there is no endpoint between l_x and l_y, and no endpoint between r_x and r_y. By removing x and y, we obtain a ladder lottery with the less number of bars, which is a contradiction. ∎

Lemma 4.1 gives an idea for saving bits. If there is no endpoint between l_x and l_y, then there always exists at least one endpoint between r_x and r_y (Fig. 4(b)). Hence we can save 1 bit among bits representing endpoints between r_x and r_y.

For example, we obtain “$\texttt{1\underline{0}11011\underline{0}00010\underline{0}11000\underline{0}0001\underline{0}00}$” by applying this idea to *LC* for the ladder lottery in Fig. 1. The underlined ‘0’s represent boundaries between two consecutive line-codes.

Reconstruction of *LC* with this idea can be done in straightforward way. The line 1 is clearly reconstructed from the corresponding line-code. Suppose a line i is reconstructed. Then we can reconstruct the saved ‘1’s in a line-code for a line $i+1$, by checking the reconstructed line i.

Improvement 3

Along with improvement 2, we can save some bits for right endpoints of bars on the line $n-1$. Since the line n has no left endpoint of any bar, any pair of consecutive two bars between lines $n-1$ and n always forms the configuration illustrated in Fig. 4(a). Let x, y be any consecutive two bars with left endpoints on the line $n-1$ and right endpoints on the line n. We denote by l_x, r_x the left and the right endpoints of x. Similarly, we denote by l_y, r_y the left and the right endpoints of y. Since there is no endpoint between r_x and r_y, there is at least one right endpoint of a bar between l_x and l_y. Therefore we can save 1 bit among ‘0’s representing the number of right endpoints between l_x and l_y. By this improvement zero or more ‘0’ in the line-code for $n-1$ is saved.

For example, *LC* with improvement 3 for the ladder lottery in Fig. 1 is “$\texttt{1\underline{0}11011\underline{0}01001100\underline{0}1100100\underline{0}0101\underline{0}00}$”. In this example, one ‘0’ is saved in the line-code for the line 5. The underlined ‘0’s represent boundaries between two consecutive line-codes.

Reconstructing the original ladder lottery L from a line-based code with improvement 3 is slightly complicated. If we apply the decoding of a line-based code directly, we may fall into the situation that some ‘0’s are insufficient. Basically, a decoding of a line-based code with improvement 3 is similar for a line-based code. However, we have to carefully decode the line-code for $n-1$ which may contain saved ‘0’s.

Without loss of generality, we suppose that a saved ‘0’ appears immediately after each ‘1’ except the last ‘1’ in the line-code for $n-1$. See Fig. 5(a) for an example. In Fig. 5(a), each saved ‘0’ is represented by ‘0’ with a diagonal line, and the line-code for $n-1$ contains two saved ‘0’s.

Suppose we apply the same decoding of a line-based code to a line-based code with improvement 3. If we know that the currently decoding line-code is for $n-1$, then the line-code can be decoded by finding saved ‘0’s, because any saved ‘0’ appears immediately after each ‘1’ except the last ‘1’. Unfortunately, during a reconstruction, we do not know the number of lines in L. Hence, it seems to be difficult to know whether or not the currently decoding line-code is for $n-1$.

Our idea of reconstruction is as follows. First, we reconstruct L by the same decoding of a line-based code. If the reconstruction is a failure, it implies that there are saved ‘0’s in the line-code for $n-1$. Hence, we retry the decoding for two line-codes for $n-1$ and n more carefully. Otherwise, L can be reconstructed correctly.

The details are as follows. Let *LC* be a line-based code for L and LC' be a line-based code for L with improvement 3. We denote by $C'(n-1)$ a line-code for $n-1$ in LC'. Let p be the position of the last ‘1’ in LC', and e be the number of ‘0’s succeeding ‘1’ in the position p.

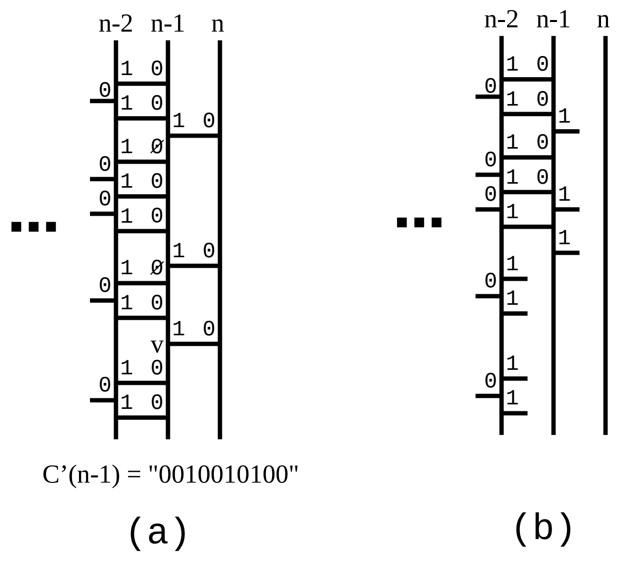

Fig. 5 Illustration for improvement 3.

Let ℓ be the line whose line-code contains p. We can observe that ℓ is the line $n-1$ or less. If the line $n-1$ contains no left endpoint, then ℓ is les than $n-1$. Let v be the bottom left endpoint on ℓ. Then v corresponds to the '1' in p. See 5(a) for an example.

We assume that the same decoding for LC is performed for LC' ignoring saved '0's. After decoding the '1' in p, we can compute (1) the number of right endpoints on ℓ under v in the reconstructed ladder lottery and (2) the number of right endpoints on $\ell+1$ in the reconstructed ladder lottery. Let r be the sum of the two numbers (1) and (2). See Fig. 5(b) for an example. A ladder lottery in Fig. 5(b) is obtained by decoding $C'(n-1)$ in Fig. 5(a) ignoring saved '0's (This reconstruction is a failure). Then we have $r = 7$. We note that the value of (1) is greater than the number of left endpoints under v in the original ladder lottery L if there is at least one saved '0'.

By comparing e with r, we can determine whether or not a saved '0' is contained in $C'(n-1)$.

Lemma 4.2 We have the following three cases.
(a) If $e > r+1$, then $\ell < n-1$ and there is no saved bit.
(b) If $e = r+1$, then $\ell = n-1$ and there is no saved bit.
(c) If $e < r+1$, then $\ell = n-1$ and there is one or more saved bits.

Proof. We can observe that e is the sum of (i) the number of right endpoints under v on ℓ in L and (ii) the number of right endpoints on $\ell+1$ and (iii) the number of '0's corresponding to boundaries between lines i and $i+1$ for $i = \ell, \dots, n-1$. Since $\ell \leq n-1$ holds, e always counts the '0' corresponding to the boundary between $n-1$ and n. r is greater than or equal to the sum of (i) and (ii).

Case (a): Since $e > r+1$, then e counts two or more '0's corresponding to boundaries. Therefore $\ell < n-1$ holds, and hence there is no saved '0'.

Case (b): If $\ell < n-1$, then there is no saved bit. Then r is equal to the sum of (i) and (ii), and (iii) is greater than 1. Hence $e > r+1$ holds, which is a contradiction, so $\ell = n-1$ holds. By $\ell = n-1$, e counts only the '0' representing the boundary between $n-1$ and n for (iii). Hence r is equal to the sum of (i) and (ii). Therefore there is no saved bit.

Case (c): If $\ell < n-1$, then there is no saved bits. Then we have $e \geq r+1$, which is a contradiction. Therefore $\ell = n-1$ holds. Since e counts, for (iii), only the '0' corresponding to the boundary between $n-1$ and n, r is greater than the sum of (i) and (ii). It implies that there is at least one saved bit. ∎

For example, $e < r+1$ holds for a ladder lottery in Fig 5.

By Lemma 4.2, we have the following decoding algorithm. First, we decode LC' by the same decoding of LC until p, then we perform a suitable reconstruction by comparing e and r. For the cases (a) and (b) in Lemma 4.2, we can reconstruct by the completely same decoding for LC. For the case (c), we reconstruct $C'(n-1)$ again so that all saved bits are found.

Combining improvements 1-3

Now we show that the three improvements can be applied to a line-based code simultaneously.

The three improvements have no duplicate of saved '0's or '1's. Hence, we can apply improvements 1-3 to a line-based code. We denote by IC a code obtained by applying improvements 1-3 to a line-based code. We call IC an *improved code* for an optimal ladder lottery. For example IC for the optimal ladder lottery in Fig. 1 is "1$\underline{0}$11011$\underline{0}$00010$\underline{0}$11000$\underline{0}$001". The underlined '0's represent boundaries between two consecutive line-codes. We denote by $C_I(i)$ a line-code for a line i in IC. For example, we have $C_I(1)$ = "1", $C_I(2)$ = "11011", $C_I(3)$ = "00010", $C_I(4)$ = "11000", $C_I(5)$ = "001 and $C_I(6) = \varepsilon$. for the ladder lottery in Fig. 1.

Now we explain how to decode IC. A basic process is a similar way for a line-based code with improvement 3. First, we apply the same decoding of a line-based code with improvement 2 to IC. If the decoding is a failure, we retry to decode $C_I(n-1)$ carefully. Otherwise, L can be reconstruct correctly.

Let q be the position of the last '1' in IC, and f the number of '0's succeeding '1' in q. Let k be the line whose line-code contains q. k is a line $n-1$ or less. Let u be the bottom left endpoint on k. Since $C_M(n) = \varepsilon$ is holds in IC, f is equal to the number of right endpoints under u. We assume that the same decoding for a line-based code with improvement 2 is performed for IC. After decoding the '1' in q, we can compute the number of right endpoints on k under u in the reconstructed ladder lottery. Let s be the number. We note that saved left endpoints on k by improvement 2 are recognized and reconstructed by the decoding for a line-based code with improvement 2.

We have the following lemma.

Lemma 4.3 We have the following three cases.
(a) If $f > s$, then $k < n-1$ and there is no saved bit.
(b) If $f = s$, then $k = n-1$ and there is no saved bit.
(c) If $f < s$, then $k = n-1$ and there is one or more saved bits.

Proof. The proof is similar to Lemma 4.2. ∎

By Lemma 4.3, we have the decoding algorithm along with a line-based code with improvement 3.

Theorem 4.4 Let L be an optimal ladder lottery with n lines and b bars. We can compute a code IC in $O(n+b)$ time, and reconstruct L from IC in $O(n+b)$ time.

5. Experimental Result

We compare the average length of the codes proposed in previous sections.

An environment for an experiment is as follows. (1) OS: FreeBSD 8.2-RELEASE, (2) CPU: AMD Phenom(tm) II X6 1065T Processor (2909.62-MHz K8-class CPU), (3) Main memory: 4GB and (4) Programming language: Common lisp (5) Compiler: SBCL 1.0.57.

Now we show processes in our experiment. First we enumerate all permutations of n elements. Second we enumerate all optimal ladder lotteries of each permutation. Third we compute three codes RC, LC and IC for every optimal ladder lottery. Finally we calculate the average length of each code.

We have designed an efficient algorithm that enumerates all optimal ladder lotteries of a given permutation [16]. Using the algorithm, we enumerated all optimal ladder lotteries of every permutation with n elements. For $n = 1, 2, \ldots, 9$, Table 1 shows the number of permutations with n elements, the number of optimal ladder lotteries with n lines, and the average number of bars in optimal ladder lotteries with n lines. Table 2 shows average lengths of RC, LC and IC.

Table 1 The number of permutations, the number of optimal ladder lotteries with n lines, the average number of bars.

n	# of permutations with n elements	# of optimal ladder lotteries with n lines	average # of bars in a ladder lottery
2	2	2	0.5
3	6	7	1.7
4	24	43	4.0
5	120	476	7.1
6	720	9,661	11.4
7	5,040	361,071	16.8
8	40,320	24,787,065	23.2
9	362,880	3,111,103,213	30.6

Table 2 The average length of each code per an optimal ladder lottery.

n	RC (bits)	LC (bits)	IC (bits)
2	2.0	2.0	1.0
3	6.0	5.4	3.3
4	12.0	10.7	7.7
5	20.0	18.1	14.1
6	30.0	27.8	22.4
7	42.0	39.5	32.7
8	56.0	53.3	44.8
9	72.0	69.2	58.8

Table 2 shows that IC is the most compact among the three codes. Although the length of IC is $n + 2b - 1$ bits in the worst case, the experimental result shows that the average length of IC is more compact than the worst case. This implies that our improvements are valid on saving bits.

6. Conclusion

We have designed three codes for an optimal ladder lotteries with n lines and b bars. The route-based code can be computed and decoded in $O(n^2)$ time, and the line-based code can be computed and decoded in $O(n + b)$ time. Furthermore we improved the line-based code. For improved code, coding and decoding can be done in $O(n + b)$ time.

We computed the average length for four codes RC, LC and IC using enumeration algorithms. The experimental result shows that IC is the most compact code among them.

The length of the improved code is at most $n+2b-1$ bits. However, the experimental result shows that the average length of the improved code is more compact than the worst-case length.

Our future works are as follows. (1) Can we formally give upper and lower bounds on the average length of the improved code? (2) Is there a more compact code? (3) Can we give a information-theoretical lower bound for an optimal ladder lottery with n lines and b bars?

Acknowledgement

We would like to thank participants of The 16th Enumeration-algorithm Seminar for useful discussions. We also would like to thank Takehiro Ito and Kei Uchizawa for useful discussions. This work was supported by JSPS KAKENHI Grant Number 23700012.

References

[1] L.C. Aleardi, O. Devillers, and G. Schaeffer. Succinct representations of planar maps. *Theoretical Computer Science*, 408:174–187, 2008.

[2] D.A. Benoit, E.D. Demaine, J.I. Munro, and V. Raman. Representing trees of higher degree. *Algorithmica*, 43:275–292, 2005.

[3] Y.T. Chiang, C.C. Lin, and H.I. Lu. Orderly spanning trees with applications. *SIAM Journal on Computing*, 34:924–945, 2005.

[4] R.C.N. Chuang, A. Garg, X. He, M.Y. Kao, and H.I. Lu. Compact encodings of planar graphs via canonical orderings and multiple parentheses. *Proc. of 25th International Colloquium on Automata, Languages and Programming (ICALP 1998)*, LNCS 1443:118–129, 1998.

[5] S. Felsner. On the number of arrangements of pseudolines. In *Proc. The 12th annual Symposium on Computational Geometry, (SCG 1996)*, pages 30–37, 1996.

[6] S. Felsner. On the number of arrangements of pseudolines. *Discrete & Computational Geometry*, 18:257–267, 1997.

[7] S. Felsner and P. Valtr. Coding and counting arrangements of pseudolines. *Discrete & Computational Geometry*, 46:405–416, 2011.

[8] R.R. Geary, R. Raman, and V. Raman. Succinct ordinal trees with level-ancestor queries. *ACM Transactions on Algorithms*, 2(4):510–534, 2006.

[9] J.E. Goodman. Pseudline arrangements. *Handbook of Discrete and Computational Geometry* CRC Press, pages 97–128, 2004.

[10] G. Jacobson. Space-efficient static trees and graphs. *Proc. of the 30th IEEE Symposium on Foundations of Computer Science, (FOCS1989)*, pages 549–554, 1989.

[11] K. Keeler and J. Westbrook. Short encodings of planar graphs and maps. *Discrete Applied Mathematics*, 58:239–252, 1995.

[12] D.E. Knuth. *Axioms and hulls.* LNCS 606, Springer-Verlag, 1992.

[13] J.I. Munro and V. Raman. Succinct representation of balanced parentheses and static trees. *SIAM Journal on Computing*, 31(3):762–776, 2001.

[14] D. Poulalhon and G. Schaeffer. Optimal coding and sampling of triangulations. *Algorithmica*, 46:505–527, 2006.

[15] K. Yamanaka and S. Nakano. A compact encoding of plane triangulations with efficient query supports. *Information Processing Letters*, 110(18-19):803–809, 2010.

[16] K. Yamanaka, S. Nakano, Y. Matsui, R. Uehara, and K. Nakada. Efficient enumeration of all ladder lotteries and its applications. *Theoretical Computer Science*, 411:1714–1722, 2010.