Optimal Region for Binary Search Tree, Rotation and Polytope

Kensuke Onishi[1] Mamoru Hoshi[2]

[1] Department of Mathematical Sciences, School of Science
Tokai University, 1117 Kitakaname, Hiratsuka, Kanagawa, 259-1292, Japan
Email: onishi@ss.u-tokai.ac.jp
[2] Graduate School of Information Systems, University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo, 182-8585, Japan
Email: hoshi@is.uec.ac.jp

Abstract *Given a set of keys and its weight, a binary search tree(BST) with the smallest path length among all trees with the keys and the weight is called optimal tree. Knuth showed that the optimal tree is computed in the time of square of the number of keys.*

In this paper, we propose algorithms that divide the weight space into regions corresponding to optimal trees by a construction algorithm of convex hull. It is proved that each BST has a non-empty region, called optimal region, for which only the tree is optimal and that two optimal regions are adjacent if the corresponding trees are transformed to each other by a single rotation. We describe a relation between associahedron and the convex hull of the whole BSTs.

1 Introduction

In this paper, we deal with weighted binary search trees (BST). Suppose a set of search keys $a_1 < a_2 < \cdots < a_n$ and a weight $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$, where w_i is a given weight of key a_i. The *weighted path length* of a binary search tree T, denoted by $E(T, \boldsymbol{w})$, is defined by $\sum_{i=1}^{n} w_i \cdot l_i$, where l_i is the path length from the root of T to node a_i and l_i is also called the level of the node. The vector $\boldsymbol{l}(T) := (l_1, \ldots, l_n)$ is called *level vector* of the given BST T. A BST with a $\boldsymbol{w}$ is optimal if $E(T, \boldsymbol{w})$ is smaller than or equal to $E(T', \boldsymbol{w})$ for any other BST T'. In [4] Knuth fixed the set of keys and its weight. In this paper, only the set of keys is fixed and the weight is regarded as a variable.

We investigate a region of the weight space $\boldsymbol{W}_n := \{\boldsymbol{w} = (w_1, \ldots, w_n) \,|\, w_i \in \mathbb{R}_+\}$ in which a given BST is optimal for *any* weight in the region. The region is called *optimal region* for the given BST. The weight space $\boldsymbol{W}_n$ can be divided into optimal regions defined below.

Let $\mathcal{L}_n$ be a set of all level vectors of BSTs with n nodes, which has same cardinal number with the set of all BSTs. Since $E(T, \boldsymbol{w})$ is rewrited as $\boldsymbol{l}(T) \cdot \boldsymbol{w}$ by level vector $\boldsymbol{l}(T)$, the optimal region $R(T)$ of T is defined by:

$$R(T) = \{\boldsymbol{w} \mid \boldsymbol{l}(T) \cdot \boldsymbol{w} \leq \boldsymbol{l}(T') \cdot \boldsymbol{w}, \forall \boldsymbol{l}(T') \in \mathcal{L}_n\}.$$

We consider two problems about optimal regions. One question is how to compute such regions. The other question is what properties optimal region has. Since there is an optimal tree for any weight, the weight space is divided into finite optimal regions. Some regions share their boundary in the subdivision. Two regions are *adjacent* if two regions share their facet. In other words, two BSTs T, T' are adjacent if there exists a weight $\boldsymbol{w}$ such that $\boldsymbol{l}(T) \cdot \boldsymbol{w} = \boldsymbol{l}(T') \cdot \boldsymbol{w} < \boldsymbol{l}(T'') \cdot \boldsymbol{w}, \forall \boldsymbol{l}(T'') \in \mathcal{L}_n$ $(T'' \neq T, T')$.

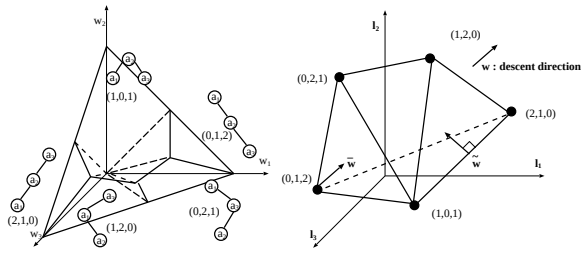

Figure 1: Optimal regions for BSTs with 3 nodes (left) and convex hull for $\mathcal{L}_3$ (right)

Figure 1 (left) is the subdivision of $\boldsymbol{W}_3$ by optimal regions with 3 nodes. Consider the space $\boldsymbol{L}_n := \{(x_1, \ldots, x_n) \mid x_i \in \mathbb{R}\}$ note that $\mathcal{L}_n \subsetneq \boldsymbol{L}_n$, which is a dual space of the weight space. In $\boldsymbol{L}_n$ each BST is regarded as a point and the weight corresponds to a descent direction. So, the computation of optimal BST for a given weight $\boldsymbol{w}(> \boldsymbol{0})$ is regarded as the following optimization problem in $\boldsymbol{L}_n$:

$$\min_{\boldsymbol{l}(T) \in \mathcal{L}_n} \boldsymbol{l} \cdot \boldsymbol{w} \text{ for a given } \boldsymbol{w}.$$

Consider a condition that the weighted path length of BST T is smaller or equal to that of BST T': $\{l(T) - l(T')\} \cdot \boldsymbol{w} \leq 0$. The inequality shows a halfspace in $\mathbf{W_n}$ and also in $\boldsymbol{L}_n$. When the inequality above is regarded as an inner product between $\{l(T) - l(T')\}$ and $\boldsymbol{w}$, the angle among the vectors is greater or equal to π. The condition $\boldsymbol{w} > \boldsymbol{0}$ for the weight is similarly dealt with (see Figure 2).

Consider a weight $\bar{\boldsymbol{w}}$ such that $\boldsymbol{l}(T) \cdot \bar{\boldsymbol{w}} < \boldsymbol{l}(T'') \cdot \bar{\boldsymbol{w}}$ for any $\boldsymbol{l}(T'') \in \mathcal{L}_n$ if exists (see Figure 1 (right)). In $\boldsymbol{W}_n$ the $\bar{\boldsymbol{w}}$ is a point of $R(\boldsymbol{l}(T))$. Consider an inequality $\boldsymbol{l}(T) \cdot \bar{\boldsymbol{w}} < \boldsymbol{x} \cdot \bar{\boldsymbol{w}}$, where $\boldsymbol{x}$ is variable in $\boldsymbol{L}_n$. Since $\bar{\boldsymbol{w}}$ is a constant vector, this inequality shows a halfspace which contains all point of $\mathcal{L}_n \setminus \{\boldsymbol{l}(T)\}$ in $\boldsymbol{L}_n$ and $\boldsymbol{l}(T)$ is on the boundary of the halfspace. So, $\boldsymbol{l}(T)$ becomes an vertex of convex hull of $\mathcal{L}_n$, denoted by $\mathrm{conv}(\mathcal{L}_n)$, if such a $\bar{\boldsymbol{w}}$ exists. Figure 1 (right) shows $\mathrm{conv}(\mathcal{L}_3)$. This figure suggest that any level vector of BST is a vertex of the convex hull.

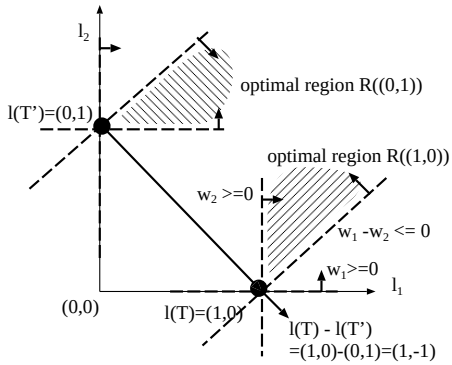

Figure 2: Convex hull for $\mathcal{L}_2$ and optimal regions for BSTs with 2 nodes

Consider a weight $\tilde{\boldsymbol{w}}$ such that $\boldsymbol{l}(T) \cdot \tilde{\boldsymbol{w}} = \boldsymbol{l}(T') \cdot \tilde{\boldsymbol{w}} < \boldsymbol{l}(T'') \cdot \tilde{\boldsymbol{w}}, \forall \boldsymbol{l}(T'') \in \mathcal{L}_n$ (see Figure 1 (right)). In $\boldsymbol{W}_n$ the $\tilde{\boldsymbol{w}}$ is a point which is included in $R(\boldsymbol{l}(T))$ and $R(\boldsymbol{l}(T'))$. Consider $\boldsymbol{l}(T) \cdot \tilde{\boldsymbol{w}} < \boldsymbol{x} \cdot \tilde{\boldsymbol{w}}$, where $\boldsymbol{x}$ is variable in $\boldsymbol{L}_n$. This inequality shows a halfspace which contains all point of $\mathcal{L}_n \setminus \{\boldsymbol{l}(T), \boldsymbol{l}(T')\}$ in $\boldsymbol{L}_n$ and $\boldsymbol{l}(T)$, $\boldsymbol{l}(T')$ are on the boundary of the halfspace. So, there exists a edge connecting between $\boldsymbol{l}(T)$ and $\boldsymbol{l}(T')$ in $\mathrm{conv}(\mathcal{L}_n)$. Optimal regions $R((1,0,1))$,$R((1,2,0))$ share a facet and level vectors $(1,0,1)$ and $(1,2,0)$ are connected by an edge of the convex hull in Figure 1.

In section 2, we give an algorithm for computing all the optimal regions by using a construction algorithm of convex hull. In this algorithm we need to generate all BSTs. We introduce two algorithms for the generation.

In section 3, some properties of optimal regions are shown. The connectivity, convexity of optimal regions are easily derived from the definition. We show that each region is non-empty. We also show that two regions are adjacent if two BSTs are transformed to each other by a single rotation.

In section 4, we describe relation between the convex hull of all level vector and associahedron. The connection between BST and triangulation of convex polygon is shown.

2 Construction Algorithm of Optimal Regions

In this section we propose an algorithm for computing all optimal regions.

2.1 Generation of all level vectors

2.1.1 direct generation

In this subsection we state direct generation of all BSTs. Any BST with n nodes is expressed by n-dimensional level vector. The vector always includes one

0, which corresponds to the root node. We must select one index, say i, for the root node. The level vector is divided into two parts: smaller and larger parts rather than the index i. Each part also becomes level vector of BST with $i-1$ nodes and with $n-i$ nodes. So, we can use recursive generation. This algorithm is shown in Figure 1.

Algorithm 1 Generation of all level vectors

```
 Input:   number of nodes n;
 Output:  vector[1..n]
          all level vectors with n nodes;
function Generate(i, j, level) {
  if (i == j) then set level to vector[i];
    return;
  for (k=i+1; k<j; k++) {
    set level to vector[k];
    Generate(i, k-1, level+1);
    Generate(k+1, j, level+1);
  }
}
```

This algorithm is very simple, but needs $O(nC_n)$ memory for keeping all level vectors where C_n is the nth *Catalan number*. The time complexity of the algorithm is also $O(nC_n)$.

2.1.2 computation by generation tree

In this section we state computation by using *generation tree*, which is a kind of binary decision diagram. The sequence of labels on a path of the generation tree corresponds to a binary tree.

Firstly, we explain *gambler's ruin sequence* ([5, p.268]). The sequence is a binary sequence of length $2n$ satisfying two condition (1) the number of 0s and that of 1s are n; (2) for any prefix of the sequence, the number of 0s is greater than or equal to the number of 1s. It is well-known that the number of gambler's ruin sequence of length $2n$ is equal to the nth Catalan number C_n and that there is one to one correspondence between such sequences and BSTs with n nodes([5, pp.268-277], [7, pp.219-229]). Let B_n be the set of all gambler's ruin sequence of length $2n$. We propose a generation method of B_n by generation tree.

A *generation tree* is a binary decision tree: (1) Any internal node has two branches with labels $0, 1$; (2) There are two leaf nodes(`False`, `True`); Consider an internal node and the subsequence from the root to the node. Let N_0, N_1 be numbers of 0s and 1s in the subsequence, respectively. By the following rule 0-branch and 1-branch of the node are connected with.

0-branch is connected with

$$\begin{cases} \texttt{False} \text{ leaf} & \text{if } N_0 = n \\ \text{child internal node} & \text{otherwise} \end{cases}$$

1-branch is connected with

$$\begin{cases} \texttt{False} \text{ leaf} & \text{if } N_0 = N_1 \\ \texttt{True} \text{ leaf} & \text{if } N_0 = n \\ \text{child internal node} & \text{otherwise} \end{cases}$$

(3) Any path from the root node to `True` leaf corresponds to a binary tree. An example of the generation tree for 3 nodes is shown in Figure 3 (left).

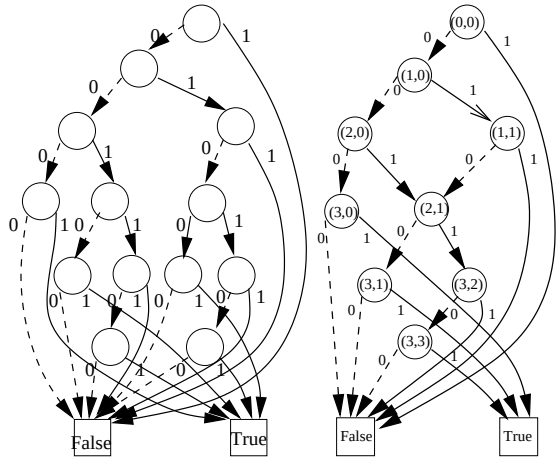

Figure 3: Generation tree for BST with 3 nodes (left) and shared diagram (right)

Generation tree includes all gambler's ruin sequences. This tree has redundant parts. A subtree with prefix $b_1 b_2 \dots b_k$ is defined as the subtree where the root node can be reached along the prefix in the tree. In Figure 3 (left), the subtree with prefix 001 is the very same with the subtree with prefix 010. These parts of the tree can be shared (Figure 3 (right)). In the generation tree, all tuple of such parts can be identified. So, we gain new diagram (Figure 3 (right)), called *shared diagram*. Such kind of share is done when two prefixes have the same number N_0 and N_1. Because the structure of subtree depends only on the numbers of 0s and 1s which can be used in the subtree, that is, $n - N_0$ and $n - N_1$. Thus these numbers are determined by the numbers in prefix. In other words, each internal node corresponds to a pair of the numbers of 0s and that of 1s (N_0, N_1) where $N_0 = 0, \dots, n, N_1 = N_0, \dots, n$. The number of such pairs is easily computed:

$$(n+1) + n + (n-1) + \cdots + 2 = \frac{n}{2}(n+3) = O(n^2).$$

Thus the shared diagram for BST with n nodes has $\frac{n}{2}(n+3)+2 = \frac{n}{2}(n+3+4)$ nodes and $n^2 + 3n$ edges. Since this diagram has $O(n^2)$ nodes, $O(n^2)$ memory needed for keeping this diagram and the time complexity of computation of this diagram are also $O(n^2)$. When all sequences are generated from the diagram, $O(nC_n)$ time needs.

2.2 Computation by using convex hull

In the previous subsection we showed two methods of generation of all level vectors. In this subsection we show a computing method for optimal regions by using convex hull.

Consider the space of level vector, which is the dual space of the weight space. The optimality of BST is equivalent to that its level vector is a vertex on convex hull of $\mathcal{L}_n$. The structure of optimal regions is also equal to the structure of $\mathrm{conv}(\mathcal{L}_n)$ because of duality. Thus we propose a computation method by using convex hull (see Algorithm 2).

Algorithm 2 Computation of all optimal regions

Input: n number of nodes in BST
Output: all optimal regions for BSTs with n nodes
1. Generate all level vectors $\mathcal{L}_n$;
2. Compute $\mathrm{conv}(\mathcal{L}_n)$;

Time complexity of this algorithm depends on computation time of convex hull, which is well-known $O(N^{\lfloor d/2 \rfloor})$, where N is the number of points and d is dimension (see [1]). In this case $N = C_n$ and $d = n$, the time complexity of this algorithm is

$$O({C_n}^{\lfloor n/2 \rfloor}).$$

3 Properties of Optimal Regions

In this section we show some properties of optimal regions.

3.1 Convexity and existence

The optimal region is defined by the intersection of some halfspaces. So, the region becomes convex cone if exists. When the optimal region for a given tree exists, the region is convex cone (see Figure 1 (left)).

We show the existence of optimal region for any BST T. Let a_i be a root node of T. Let $T_L(T_R)$ be the left (right) subtree in T, respectively. It is well-known that when T with $\boldsymbol{w}$ is optimal, then T_L with $w_1, \ldots, w_{i-1}$ and T_R with $w_{i+1}, \ldots, w_n$ are also optimal.

Consider a weight $\boldsymbol{w}$ such that $w_i \geq W\left(1-\frac{1}{n^2}\right)$ where $W=\sum_i w_i$. Let T be any BST with n nodes. If a_i is not the root node of T, then

$$E(T,\boldsymbol{w}) \geq 1 \cdot w_i \geq W\left(1-\frac{1}{n^2}\right).$$

When a_i is the root node of T, then

$$\begin{aligned} E(T,\boldsymbol{w}) &\leq (n-1)(W-w_i) \leq W\frac{n-1}{n^2} \\ &< \frac{W}{n} \leq W\left(1-\frac{1}{n^2}\right). \end{aligned}$$

The first inequality is shown from the fact that the level of any node is less than $n-1$ for any BST with n nodes. Thus, if $w_i \geq (1-\frac{1}{n^2})W$, any optimal tree with $\boldsymbol{w}$ has a_i as the root node.

We propose an algorithm for computing weight $\boldsymbol{w}$ such that a given BST is optimal under $\boldsymbol{w}$, denoted by `Weight(i,j,`T, W`)` (see Algorithm 3).

Algorithm 3 Compute an optimal weight for given BST

Input: T: BST for $a_i, \ldots, a_j (i \leq j)$;
W: sum of weights of nodes $a_i, a_{i+1}, \cdots, a_j$;
Output: weight of each nodes $w_i, \ldots, w_j$;

`function Weight(i,j,`T, W`){`
  if `(i == j)` **then** set $w_i = W$;
  if `(i != j)` **then**
    a_k: root node of T;
    $T_L(T_R)$: left (right) subtree of T;
    set $w_i = W\left(1-\frac{1}{n^2}\right)$;
  if `(k == i)` **then** call `Weight(i+1,j,`$T_R, \frac{W}{n^2}$`);`
  if `(k == j)` **then** call `Weight(i,j-1,`$T_L, \frac{W}{n^2}$`);`
  if `((k > i) && (k < j))` **then**
    call `Weight(i,k-1,`$T_L, \frac{W}{2n^2}$`);`
    call `Weight(k+1,j,`$T_R, \frac{W}{2n^2}$`);`
`}`

Algorithm 3 returns an optimal weight $\boldsymbol{w}$ for T when we call `Weight(1,n,`$T, 1$`)`. For any BST an optimal weight can be computed. Consequently, we can show the existence of optimal region for any BST.

Theorem 1 *The optimal region for any BST is non-empty and convex.*

3.2 Adjacency

In this subsection, we show that if two BSTs are transformed to each other by a single rotation, the two corresponding optimal regions are adjacent.

Let T, T' be BSTs which are transformed to each other by a single rotation. We modify the weight of `Weight(1,n,`$T, 1$`)` so that T and T' are only optimal. Without loss of generality, assume that a_i is the root of the BST. Figure 4 shows the BST T (left) and T' (right). T is transformed to T' by a single rotation. Consider the following weight for these BSTs:

$$w_i = w_j = \frac{1}{2}W\left(1 - \frac{1}{n^2}\right),$$

where W is the sum of weights of the BST. The weight of each subtree T_1, T_2 and T_3 are set to be $\frac{1}{3}\frac{W}{n^2}$. The weight for T_1, T_2, T_3 are obtained by the algorithm above. This weight called *modified weight*.

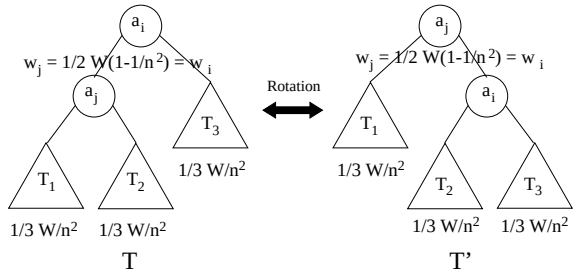

Figure 4: Rotation and modified weights

We show that for modified weight $\tilde{\boldsymbol{w}}$, T and T' are optimal and other BSTs are not optimal. Claim that $E(T, \tilde{\boldsymbol{w}}) = E(T', \tilde{\boldsymbol{w}})$. When the root node is neither a_i nor a_j in BST T'', the level of a_i and a_j is greater than or equal to 1. So, the weighted path length of T'' satisfies

$$\begin{aligned} E(T'', \tilde{\boldsymbol{w}}) &\geq 2 \cdot \frac{1}{2}W\left(1 - \frac{1}{n^2}\right) \\ &= \frac{W}{n^2}(n^2 - 1) \end{aligned} \tag{1}$$

When the root node is a_i and the level of a_j is greater than or equal to 2, the same inequality (1) is settled.

The last case is that the root node is a_i and its child is a_j or vise versa, corresponding BSTs are T and T', respectively. Any node have smaller weight than $W/3n^2$ and the number of such nodes is at most $n - 1$. In this case $E(T, \tilde{\boldsymbol{w}})$ is bounded as:

$$\begin{aligned} E(T, \tilde{\boldsymbol{w}}) &\leq \frac{W}{2}\left(1 - \frac{1}{n^2}\right) + \frac{W}{3n^2}(n-1) \\ &= \frac{W}{6n^2}(3n^2 + 2n - 5) < \frac{W}{n^2}(n^2 - 1) \end{aligned}$$

The last inequality is settled when $n \geq 1$. Thus the following inequality is shown:

$$E(T, \tilde{\boldsymbol{w}}) = E(T', \tilde{\boldsymbol{w}}) < E(T'', \tilde{\boldsymbol{w}})$$

Finally the theorem below is shown.

Theorem 2 *If two binary search trees are transformed by a single rotation, their optimal regions are adjacent.*

In [6] *rotation distance* between a pair of BSTs is defined as the minimum number of rotations needed to convert one BST into the other. When two binary tree is transformed to each other by a single rotation, the rotation distance is one.

Restating the theorem above in terms of rotation distance, we get the following corollary.

Corollary 1 *If rotation distance is equal to one for given two binary search trees, then two corresponding optimal regions are adjacent. In addition, the regions share a facet.*

4 Relation with Associahedron and Triangulation

4.1 Associahedron

In this subsection we describe relation between associahedron and the convex hull of level vector. We describe that any edge of associahedron K_{n-1} exists on the $\mathrm{conv}(\mathcal{L}_n)$.

Associahedron K_{n-1} is a polytope with C_n vertices each of which corresponds to a binary parenthesization for a sequence of length $n+1$. When two vertices are transformed by associative law, the two vertices are connected by edge. This polytope is constructed as a secondary polytope from n-gon. The detail of this polytope is found in [8]. We illustrate K_2 in Figure 5. In K_2 there are five vertices: $((12)(34)), (1(2(34))), (1((23)4)), ((1(23))4), (((12)3)4)$. The second vertex means: (a) make a pair of last two characters (34); (b) make a pair of second character and the pair in (a) (2(34)); (c) make a pair of first character and the pair in (b) (1(2(34))). Associative law is defined by changing the order of pairing i.e. $A(BC)$ is transformed to $(AB)C$ by associative law. For example, when $A = 1$, $B = 2$, $C = (34)$, then $(1(2(34)))$ is transformed to $((12)(34))$ by associative law. So, these two vertices are connected by edge in Figure 5.

We explain a relation between K_{n-1} and $\mathrm{conv}(\mathcal{L}_n)$. It is well-known that one to one correspondence between binary parenthesization of $(n+1)$-sequence and BST with n nodes. A pair of brackets in the sequence is equivalent to one node in BST. The level of the node is equal to the number of pairs of brackets at outside of the pair of brackets. We claim that any edge of associahedron becomes the edge of $\mathrm{conv}(\mathcal{L}_n)$. Consider one binary parenthesization of $(n+1)$-sequence $(((1 \cdots j)(j+1 \cdots i))(i+1 \cdots n+1))$, where each part is already parenthesized. As the parenthesization above corresponds to a BST T with n nodes, every subparenthesization is also

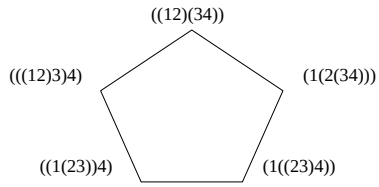

Figure 5: Associahedron K_2

regarded as subtree of the BST. Since the first subparenthesization contains j characters, the corresponding subtree contains $j-1$ nodes and is regarded as T_1 in Figure 4 (left). The second and third parts are also regarded as T_2 and T_3, respectively. The parenthesization above is transformed to $((1\cdots j)((j+1\cdots i)(i+1\cdots n+1)))$ by associative law. This parenthesization means: (a) make a pair of $((j+1\cdots i)(i+1\cdots n+1))$; (b) make a pair of $((1\cdots j)((j+1\cdots i)(i+1\cdots n+1)))$. So, the corresponding BST is equal to the T' in Figure 4 (right) which is computed from T by a single rotation.

In Theorem 2, we show the existence of edge in $\mathrm{conv}(\mathcal{L}_n)$ when two BST is transformed by a single rotation.

Corollary 2 *There is one to one correspondence between $\mathcal{L}_n$ and vertex of K_{n-1}. Moreover, conv($\mathcal{L}_n$) contains all edges of K_{n-1}.*

[**Remark**] Some edges of $\mathrm{conv}(\mathcal{L}_n)$ are characterized by associahedron. Since C_3 is equal to 5, K_2 and $\mathrm{conv}(\mathcal{L}_3)$ have five vertices. While K_2 is hexagon, $\mathrm{conv}(\mathcal{L}_3)$ has more two edges (see Figure 1 (right)). So, the edges of K_{n-1} is proper subset of the edges of $\mathrm{conv}(\mathcal{L}_n)$.

4.2 Triangulation of polygon

In this subsection, we describe a relation between triangulation of polygon and BST. It is well-known that there exists one to one correspondence between triangulation of $(n+2)$-gon and binary tree with n nodes. *Rotation* is a transformation among two BSTs. There is also *diagonal flip* in triangulation of $(n+2)$-gon. In [6], the one-to-one correspondence between rotation and diagonal flip was shown. In other words, the traversing among BSTs by rotation is equivalent to the traversing among triangulations by diagonal flip.

We show that an optimal weight of a BST is computed from the area of corresponding triangulation such that BST with the weight is optimal. A BST and corresponding triangulation are fixed. Each triangle is labeled by the index of node of the BST.

Consider the weight in Algorithm 3. When root node has $W(1-\frac{1}{n^2})$ and its child has at most $\frac{W}{n^2}$ for any subtree, the BST is optimal. Let a_i and a_j be parent and child in the BST and their weights be w_i, w_j, respectively. If the pair of w_i

and w_j satisfies

$$w_i/w_j \geq \frac{1-1/n^2}{1/n^2} = n^2 - 1,$$

then a_i becomes the root node in the subtree. When all pair of parent and child satisfy this condition, the BST with the weight is optimal.

Let s_i, s_j be normalized area of triangle in the triangulation such that $0 < s_i < 1$ and $\sum_i s_i = 1$ where s_i, s_j correspond to nodes a_i, a_j, respectively.

Constant integers K_{ij} and K are defined by:

$$K_{ij} = \left\lfloor \frac{\log s_i - \log(n^2-1)}{\log s_j} \right\rfloor,$$

$$K = \max_{\text{any pair}(a_i,a_j)} K_{ij}.$$

For any pair of parent and child, the following relation is shown:

$$s_i/(s_j)^K \geq n^2 - 1.$$

Thus the BST under $({s_1}^{Kl_1}, {s_2}^{Kl_2}, \ldots, {s_n}^{Kl_n})$ is optimal where l_i is level of node a_i.

5 Conclusion

In this paper optimal regions of binary search trees are dealt with. We state about generation of optimal regions, properties and relation with associahedron and triangulation. The generation of optimal regions needs two steps: (1) computation of all level vector with n nodes; (2) construction of convex hull of all level vectors. All level vector are expressed by shared generation tree with $O(n^2)$ memory and $O(n^2)$ time. And the program generate *all* optimal regions in $O(C_n^{\lfloor n/2 \rfloor})$ time. Time complexity per an optimal region is $O(C_n^{\lfloor n/2 \rfloor - 1})$.

We show that every optimal region is non-empty and convex. The result is equivalent to that all level vectors are vertices of the convex hull. Adjacency of two regions are also showed: if two trees are transformed to each other by a single rotation, their optimal regions share a facet. In the convex hull their level vectors are connected by an edge.

Two related topics are described in Section 4. The first is the relation with associahedron. The convex hull of level vectors has the same number of vertexes and strictly contains all edges of associahedron(see Figure 1 (right) and Figure 5). The adjacency is not completely characterized by only single rotation. It is interesting problem to consider sufficient and necessary condition of adjacency among two optimal regions.

Triangulation of polygon is also related with the optimal regions. In this paper we show that computability of an optimal weight from a given triangulation. The weight depends on the pair of nodes a_i and a_j. So, the constant $K_{i,j}$ is also depends

on corresponding weights s_i, s_j. If the number K in Section 4.2 is used, then optimal weight for a given BST can be computed from corresponding triangulation of a polygon and K.

References

[1] H. Edelsbrunner : *Algorithms in Combinatorial Geometry*, Springer-Verlag, Berlin, 1987.

[2] D. Eppsitein: Finding the k shortest paths. SIAM *Journal of Computing*, Vol. 28, 1998, pp.652-673.

[3] D.E. Knuth: *The Art of Computer Programming*, Vol. 1, Third Edition, Addison-Wesley, 1997.

[4] D.E. Knuth: *The Art of Computer Programming*, Vol. 3, Second Edition, Addison-Wesley, 1998.

[5] R. Sedgewick, P. Flajolet: *An Introduction to the Analysis of Algorithms*, Addison-Wesley, 1996.

[6] D.D. Sleator, R.E. Tarjan and W.P. Thurston: Rotation Distance, Triangulations, and Hyperbolic Geometry, *Proceedings of the Eighteenth Annual ACM Symposium on the Theory of Computing*, 1986, pp.122 - 135.

[7] R. P. Stanley: *Enumerative Combinatorics*, Vol. 2, Cambridge University Press, 1999.

[8] G. M. Ziegler: *Lectures on Polytope*, Springer-Verlag, 1995.