A generalization of the model of permutations for binary search trees

Xavier Meseguer

Report LSI-91-34
A Generalization of the Model of Permutations for Binary Search Trees

Xavier Meseguer†
Departament L.S.I (U.P.C)
Pau Gargallo 5, 08028 Barcelona, Spain
meseguer@lsi.upc.es

Abstract: The classical model of permutations, that allows the study of consecutive insertions in BST, fails when it considers intermixed deletions and insertions (Knott paradox). Our model solves this paradox by considering the inserted and deleted keys as Jonassen & Knuth proposed in [Jn,Kn 78] , and allows the development of the invariant theory of functions applied to algorithms, following the Knuth approach as in [Kn 77]. The model also explains the pattern defined by the first random insertions followed by random deletion, by applying the invariant properties of algorithms without computing combinatorial expressions. The model suggests a new line of research to face the general problem of intermixed random insertions and deletions.

1. Introduction

One of the suggested data types to solve the dictionary problem was the Binary Search Tree. It is in [Ah, Ho, Ul 76] chap.5 with two main operations: ins and del a key k on/from a tree t denoted

\[ \text{ins}(k, t) \quad \text{and} \quad \text{del}(k, t) \]

being \( \text{del}(k, t) \) the Hibbard deletion algorithm [Hi 62] . If the key k is randomly chose from the set of keys, then the operations randominsertion and randomdeletion algorithms must be used, and the dictionary problem become:

The study of the evolution of binary search trees generated by a process with intermixed random insertions and deletions.

Following [Kn 77] notation, this kind of process can be denoted as

\[ (\text{randomdeletion}, \text{randominsertion})^* \] \hspace{1cm} (1.1)

A process like (1.1) can be divided into two consecutive parts: the first which only has random insertions, and the second which begins at the first deletion and is composed by intermixed random insertions and deletions. Considering the first n random insertions, the average cost of some complexity measures can be experimentally computed, yielding:

The average time to insert or delete a key is \( O(\log n) \).

The average internal path length \( TPL(B_n) \) is \( O(n \log n) \).

The second part of process (1.1) begins with the first deletion and continues by making a great number of intermixed deletions and insertions. The average time to insert or delete keys grows faster than \( \log n \), and the \( TPL \) value also grows faster than the \( n \log n \) function, being \( n \) the size of trees. This fact was corroborated by the experiments of Eppinger [Ep 83] .

From a theoretical point of view, there are few papers on this matter, due to its inherent complexity and the absence of an appropriate mathematical model. This absence was not evidenced until the work by Knott

† Partially supported by UPC/PR9014 and BRA project number 3075. ALCOM.
Ph.Dr. Thesis [Kno 75]. Previously in 1962, a model, which we call Permutation Model and we denote $\mathcal{PM}$, was suggested by Hibbart in [Hi 62] to explain (1.1)

$\mathcal{PM}$: if $n$ keys are inserted, we assign the same probability to every permutation of keys; then the probability of a tree will be the addition of all the permutations that build up the tree. This assignment defines a probability distribution $P'$ over the set of trees $B_n$ that allows to compute complexity measures.

The Permutation Model was considered a good one for the first $n$ random insertions because its theoretical results were confirmed by the experimental ones. The model allowed the proof of the Hibbart theorem, that studies the effect of one random deletion:

Considering the set of $B_n$ with the probability $P'$, if we delete one key randomly selected, the shape distribution of trees is also random.

Like the generalization of the theorem over intermixed random deletion and random insertions was accepted without further analysis, the $\mathcal{PM}$ became the true mathematical model which explained processes like (1.1).

It seems that the problem was a solved one, but Knott [Kno 75] showed the fallacy of the theorem generalization with the following paradox:

If the process of Hibbart Theorem has been done, and then a new key is randomly inserted, the shape distribution of trees is not random.

Thus the permutation model was a model only for the first random insertions, therefore the dictionary problem became an open one. Later Jonassen-Knuth [Jo,Knu 78] studied the evolution of 3-BST with a new approach suggested by Knuth:

A model that desires explain a process like (1.1), must take into account the inserted and further deleted keys.

By applying the idea, they generated and computed combinatorial expressions, but they did not develop the new model. This new approach needed a new theoretical basis because it cannot be applied to the Permutation Model. The model developed in this paper is a generalization of the $\mathcal{PM}$ that includes the Knuth approach.

At the same time Knuth published [Knu 77] in which studies processes like (1.1), but with a different kind of operations and abstract data types. He applied the idea of "insensitivity" of a probability distribution. This idea will be applied in our paper as invariances of the algorithms. The last theoretical contribution was made by Culberson and Munro [Cu, Mu 90]. They proved that a sufficient number of pseudo-random updates(a random deletion followed by the insertion of the deleted key) gives $\Theta(n \sqrt{n})$ as the value of the TPL.

In 1990 we wrote two papers: the first one [Me 90] is an approximation to dynamical behaviour of BST in the same pattern as[Jo,Knu 78]. We defined the probabilistic deletion algorithms and some kinds of invariances. In the second [Ma,Me 90] paper we made a new approach to analysis of a single deletion in BST with a probabilistic deletion algorithm. Both papers can be considered as the basis of the present one.

This paper explains a mathematical model, that we will call Variations Model and will be denoted $\mathcal{VM}$. Is a natural generalization of the $\mathcal{PM}$ that develops the following Knuth ideas:

- To remember the deleted keys,
- To analyze the "insensitivity" of algorithms.

Our model allows the study of the pattern

$$randominsertion^{\kappa}randomdeletion^{\kappa}$$ (1.2)

viewed as a generalization of the Hibbart theorem, but it does not explain the general problem 1.1. We think that a model that explains (1.1) will be a generalization of $\mathcal{VM}$, like $\mathcal{VM}$ is a generalization of the permutation model. In this sense the model opens lines for further research.

In section 2 the classical model of permutations is analyzed by interpreting the insertion and deletion algorithm as mathematical transformations, and by showing the probability induced by this model. In section
3 our model of variations is defined by generalizing the results of the permutation model. The next section recalls the tools needed for the last sections. In the sections 5 and 6 the random insertion algorithm and the random deletion algorithm are analyzed, and the main theorems of this paper, the invariance theorems, are proved. In section 7 this invariance theorems are applied to prove some result over the pattern (1 2). Finally, in section 8 we set out the conclusions and further lines of research.

2. Permutations Model

The classical model will be recalled with a quite different point of view. The sets of trees, its construction and its defined probability will be analyzed. The algorithms will be viewed as dynamical transformation between the sets of trees.

2.1. Search Trees

We deal with the interval $[0, 1]$ as the set of keys to be inserted. Consider $k_1, k_2, \ldots, k_n$ the keys just now inserted. The construction of a tree can be interpreted by the function $\text{cons}$:

$$
\text{cons} : [0, 1] \times [0, 1] \times \cdots \times [0, 1] \times B_0 \rightarrow B_n
$$

$$
k_1, \ k_2, \ \cdots, \ k_n, \ \Box \rightarrow \text{cons}(k_1, k_2, \ldots, k_n, \Box)
$$

defining it as a composition of the algorithm $\text{ins}$:

$$
\text{cons}(k_1, k_2, \ldots, k_n, \Box) = \text{ins}(k_1, \text{ins}(k_2, \ldots (\text{ins}(k_n, \Box) \ldots))
$$

The whole set of trees $B_n$ can be builded up with the permutations of keys. The equivalent principle [Fla 88b] allows to identify the keys $\{k_1, k_2, \ldots, k_n\}$ with the labels $\{1, 2, \ldots, n\}$. Thus the function $\text{cons}$ over keys can be interpreted as the function $\text{CONS}$ over permutations of integers as:

$$
\text{CONS} : \mathcal{P}_n \times B_0 \rightarrow B_n
$$

$$(z, y, \ldots, a, \Box) \rightarrow \text{CONS}(z, y, \ldots, a, \Box) = \text{INS}(z, \text{INS}(y, \ldots (\text{INS}(a, \Box) \ldots))
$$

where $\text{INS}$ is the insertion algorithm acting over labels.

The keys $\{1, 2, \ldots, n\}$ can be inserted in $n!$ different orders corresponding to the number of permutations of the set. This set of entries induces a probability distribution $P'$ over the trees by starting with the hypothesis of the equally likely probability of permutations. Different permutations can construct the same tree, then we define the $P'$ as the ratio of the number of permutations that gives the tree $t$ to the total number $n!$ of permutations:

$$
P'(t) = \sum_{x \in \mathcal{P}(\{1, 2, \ldots, n\}) \text{ generates } t} \frac{1}{n!}
$$

This probability has the recursive expression [Ba, Ca, Di, Ma]

$$
P'(t) = \frac{1}{|t|} P'(t_1) P'(t_r) \text{ being } t = \bigvee_{t_l \neq t_r} \bigwedge_{t_l \neq t_r} \left( \begin{array}{c}
k \\
t_l \\
t_r
\end{array} \right)
$$

(2.1.1)

where $k$ is the root, and $t_l$ and $t_r$ are the left and right sons.
2.2. Insertion algorithm

The insertion of a key over a tree of $\mathcal{B}_n$ can be interpreted by the function

$$[0, 1] \times \mathcal{B}_n \rightarrow \mathcal{B}_{n+1}$$

$$k, t \mapsto \text{ins}(k, t)$$

where the set $\mathcal{B}_n$ has been constructed with the keys $k_1, k_2, \ldots, k_n$, which verify

$$[0, 1] = [0, k_1] \cup [k_1, k_2] \cup \ldots \cup [k_n, 1],$$

If a new key $k'$ is inserted then

$$k' \in [0, k_1] \text{ or } k' \in [k_1, k_2] \text{ or } \ldots \text{ or } k' \in [k_n, 1].$$

The question that now arises is how the $\text{INS}$ function, which should imply the commutativity of the following diagram, can be defined

$$k_1, k_2, \ldots, k_n \xrightarrow{\text{ins}(k', t)} k_1, \ldots, k_{i-1}, k'_i, k_{i+1}, \ldots, k_n$$

$$\downarrow \quad \quad \quad \quad \downarrow$$

$$1, 2, \ldots, n \xrightarrow{\text{INS}(i', t)} 1, 2, \ldots, i-1, i, i+1, \ldots, n+1$$

being $i-1 < i' < i$. The answer is that we must leave unchanged the keys smaller than $i'$ and we must add the unity to the keys equals or greater than $i'$.

**EXAMPLE 2.1.3.** To insert the label $1' < 1$ in the tree

```
    1
   / \  \\
  2 1 3
```

$\text{INS} \left( \begin{array}{c} 1' \\ 2 \end{array} \right) = \left( \begin{array}{c} 2 \\ 1 \\ 3 \end{array} \right) \in \mathcal{B}_3$

This rule permits to interpret the insertion as

$$\text{INS} : \{1, 2, \ldots, n+1\} \times \mathcal{B}_n \rightarrow \mathcal{B}_{n+1}$$

$$i, t \mapsto \text{INS}(i, t)$$

From now on the superscript *prime* in the labels to be inserted shall be omitted.

The last question we must face is to set up an hypothesis which induces the same probability distribution as the construction of the set $\mathcal{B}_{n+1}$ does:

**Hypothesis** of Random Elected: the probability of the key $k'$ belongs to an interval is the same for all intervals:

$$\text{prob}(k' \in [0, k_1]) = \text{prob}(k' \in [k_1, k_2]) = \ldots = \text{prob}(k' \in [k_n, 1]).$$

The next example shows the construction of a tree following a consecutive insertions:

**EXAMPLE 2.1.4.** The tree

```
    4
   /|
  3 5
 /|
2 5
 /|
1 4
```
can be constructed by \( INS(1, INS(4, INS(1, INS(1, INS(1, O)))))) \)

### 2.3. Deletion algorithm

Consider \( \{k_a, \ldots, k_s\} \) the keys just now inserted. The deletion algorithm can be interpreted as:

\[
\begin{align*}
\text{del} : \{k_a, \ldots, k_s\} \times B_n & \rightarrow B_{n-1} \\
\quad k_i, t & \mapsto \text{del}(k_i, t)
\end{align*}
\]

The transformation of the set of keys gives the diagram:

\[
\begin{array}{cc}
\{k_a, \ldots, k_h, k_i, k_j, \ldots, k_s\} & \xrightarrow{\text{del}(k_i, t)} & \{k_a, \ldots, k_h, k_j, \ldots, k_s\} \\
\downarrow & & \downarrow \\
\{1, 2, \ldots, n\} & \xrightarrow{\text{DEL}(i, t)} & 1, 2, \ldots, n - 1
\end{array}
\]

The commutativity of diagram suggest that the \( \text{DEL} \) function must be defined as a composition of two processes: the deleted process that deletes the key \( i \), followed by the renumbered process that renumbers the keys in the following way: it leaves unchanged the smaller keys and subtracts the unit to the greater keys.

**EXAMPLE 2.2.1.** By the Hibbard deletion algorithm:

\[
\begin{array}{c}
\text{DEL} \\
\begin{array}{c}
2 \\
\begin{array}{c}
1 \\
3
\end{array}
\end{array}
\end{array}
= \begin{array}{c}
1 \\
\begin{array}{c}
3 \\
2
\end{array}
\end{array}
\]

Then (Is \( D_0 \) in [Knu 77])

\[
\text{DEL} : \{1, 2, \ldots, n\} \times B_n \rightarrow B_{n-1} \\
\quad x, t & \mapsto \text{DEL}(x, t)
\]

This definition, which is the only one that can be accepted by the permutation model, causes the paradox of Knott. It is worth of nothing that the renumbering process implies that the deleted keys will be lost, whereas the Knuth idea suggest that the the keys previously inserted and deleted will be remembered.

### 3. Variations model

The new model named \textit{Variations model} and denoted \( VM \) shall be exposed. As it includes the Knuth idea this model does not contains the Knott paradox. Thus every tree given by a process as \( \text{insertion}, \text{deletion} \) could be represented and in this sense is why we say that it is a model. The model will be explained by showing, first, the new definition of a tree, which will be followed by an interpretation of the algorithms, and finally, by defining a new probabilistic distribution. We must point out that like the new tree sets are mathematical abstractions, then the reader does not interpret them as a real search trees.

#### 3.1. Search Trees

In the classical model \( PM \) it has been understood that if \( t \in B_n \) then \( t \) was a tree of size \( n \) with labels from the set \( \{1, 2, \ldots, n\} \). Thus the label of a node could be identified with its position in inorder search.

The trees which are going to be explained break this last identification by accepting a set of labels greater than the size of the tree. For example:

\[
\begin{array}{c}
5 \\
\begin{array}{c}
3 \\
8
\end{array}
\end{array}
\quad \quad \quad
\begin{array}{c}
7 \\
\begin{array}{c}
2
\end{array}
\end{array}
\]
Since the labels can be selected from different sets, for example the labels 3, 5, 8 are elements from any set \( \{k, k+1, \ldots, m\} \) for \( 1 \leq k \leq 3 \) and \( m \geq 8 \), then the notation of a tree should include the set from which the labels was been selected. For example:

\[
\begin{array}{c}
5 \\
3 \quad 8 \\
2 \quad \ldots \quad 9
\end{array} \quad \begin{array}{c}
7 \\
2 \\
1 \quad \ldots \quad 7
\end{array}
\]

For our purpose we will identify trees as

\[
\begin{array}{c}
5 \\
3 \quad 8 \\
2 \quad \ldots \quad 9
\end{array} \equiv \begin{array}{c}
4 \\
2 \quad 7 \\
1 \quad \ldots \quad 8
\end{array}
\]

because one of them is like the other adding or subtracting some value over all items. If the tree \( t^{-i} \) denotes the tree \( t \) after subtracted \( i \) over all items, then following equivalent relation, which generalizes the previous identification, can be defined:

\[
\begin{array}{c}
t \\
k \quad \ldots \quad m
\end{array} \equiv \begin{array}{c}
i \\
k \quad \ldots \quad m
\end{array} \quad \text{iff} \quad t^{-k} = i^{-k} \quad \text{and} \quad m - k = i - k
\]

The tree with keys in \( \{1, \ldots, m\} \) will be defined as the class representative, and from now the quocient set will be refered as the set of trees. This new description of trees suggest a new notation to the set of trees by indicating two parameters \( n \) and \( m \), then \( B^m_n \) means the set of trees with size \( n \) labeled by keys from the set \( \{1, 2, \ldots, m\} \) being \( m \geq n \). For example

\[
\begin{array}{c}
5 \\
1 \quad 8 \\
1 \quad \ldots \quad 9
\end{array} \in B^3_3 \quad \begin{array}{c}
7 \\
2 \\
1 \quad \ldots \quad 7
\end{array} \in B^7_2
\]

Definition 3.1. We define the set \( B^m_n \) as:

\[
B^m_n = \left\{ \begin{array}{c} \emptyset \\
1 \quad \ldots \quad m
\end{array} \right\} \quad \forall m
\]

\[
B^m_n = \left\{ \begin{array}{c} k \\
k \quad \ldots \quad m
\end{array} \right\} \quad \left\{ \begin{array}{c} t_l \\
k \quad \ldots \quad k - 1
\end{array} \right\} \quad \left\{ \begin{array}{c} t_r \\
k + 1 \quad \ldots \quad m
\end{array} \right\} \quad \text{being} \quad \left\{ \begin{array}{c} 1 \leq k \leq m \\
1 \quad \ldots \quad k - 1
\end{array} \right\} \in B^{k-1}_{|t_l|} \quad \left\{ \begin{array}{c} t_r^k \\
1 \quad \ldots \quad m - k
\end{array} \right\} \in B^{m-k}_{|t_r|}
\]

EXAMPLE 3.1.1.: the set \( B^3_2 \) will be

\[
\begin{array}{c}
1 \\
2 \\
1 \quad \ldots \quad 3
\end{array} \quad \begin{array}{c}
1 \\
3 \\
1 \quad \ldots \quad 3
\end{array} \quad \begin{array}{c}
2 \\
1 \\
1 \quad \ldots \quad 3
\end{array} \quad \begin{array}{c}
2 \\
3 \\
1 \quad \ldots \quad 3
\end{array} \quad \begin{array}{c}
3 \\
1 \\
1 \quad \ldots \quad 3
\end{array} \quad \begin{array}{c}
3 \\
2 \\
1 \quad \ldots \quad 3
\end{array}
\]

To simplify the notation the trees of \( B^m_n \) will be denoted as \( t^m_n \), or \( t \) if \( n = m \). Then set \( B_n \) can be understood as a reduction to the classical model of \( B^m_n \). (There exist a bijective application between both sets).
Our model contains the triangular family of tree-sets showed in the next diagram, being the first row the linear family allowed by the classical model.

\[
\begin{array}{cccccc}
B_0^0 & B_1^1 & B_2^2 & B_3^3 & \cdots & B_m^m \\
B_0^1 & B_1^2 & B_2^3 & \cdots & B_{m-1}^m \\
B_0^2 & B_1^3 & \cdots & B_{m-2}^m \\
& & \cdots & \cdots & \cdots \\
& & & \cdots & B_{m-3}^m \\
& & & & \cdots \\
& & & & \cdots \\
B_0^n & & & & & \\
\end{array}
\]

Diagram 1

3.2. Insertion and deletion algorithm

If a key is going to be inserted in \( B_n^m \), the rule defined for the insertion in the model of permutations will be applied:

**EXAMPLE 3.2.1:**

\[
INS \left( 1, \left( \begin{array}{c}
2 \\
1 \\
1 \ldots 3
\end{array} \right) \right) = \left( \begin{array}{c}
3 \\
2 \\
1 \\
1 \ldots 4
\end{array} \right)
\]

\[
INS \left( 4, \left( \begin{array}{c}
3 \\
2 \\
1 \ldots 3
\end{array} \right) \right) = \left( \begin{array}{c}
3 \\
2 \\
2 \\
1 \ldots 4
\end{array} \right)
\]

Then the insertion of a label increases the size and the set of labels, it acts as a horizontal transformation in diagram 1 and it can be interpreted by the function:

\[
INS : \{1, 2, \ldots, m+1\} \times B_n^m \longrightarrow B_{n+1}^{m+1}
\]

\[
y \quad , \quad t_n^m \longrightarrow INS(y, t_n^m)
\]

The deletion algorithm is understood as the classical model without renumber the labels, and can be interpreted as the function

\[
DEL : \{1, 2, \ldots, m\} \times B_n^m \longrightarrow B_{n-1}^m
\]

\[
x \quad , \quad t_n^m \longrightarrow DEL(x, t_n^m)
\]

which shows that even a key has been deleted the set of labels has not been reduced. The deletion algorithm acts as a vertical transformation.

**EXAMPLE 3.2.2:** By Using the Hibbert deletion algorithm:

\[
DEL \left( 2, \left( \begin{array}{c}
2 \\
1 \\
1 \ldots 7
\end{array} \right) \right) = \left( \begin{array}{c}
1 \\
3 \\
1 \ldots 7
\end{array} \right)
\]

Finally we show that every process with intermixed insertions and deletions can be represented in the triangular diagram:
EXAMPLE 3.2.3.: the tree \[
\begin{array}{c}
3 \\
/ \\
/ \\
1 \\
\downarrow \\
4 \\
\downarrow \\
1 \ldots 5
\end{array}
\in B_3^5
\]
can be generated by

a) \(DEL(5, INS(1, INS(4, DEL(1, INS(1, INS(2, INS(1, \Box)))))))\)
b) \(DEL(2, INS(4, INS(1, DEL(3, INS(3, INS(1, \Box)))))))\)
c) \(INS(1, INS(1, DEL(1, INS(1, DEL(3, INS(3, INS(1, \Box)))))))\)

that suggest the following paths:

\[
\begin{array}{cccccc}
B_0^a & B_1^a & B_2^a & B_3^a & B_4^a & B_5^a \\
\downarrow^a & \downarrow^c & \downarrow^a & \downarrow^c & \downarrow^a & \downarrow^c \\
B_0^c & B_1^c & B_2^c & B_3^c & B_4^c & B_5^c \\
\end{array}
\]

3.3. Probability distribution

As it has been done with the permutations in the classical model, it can be established the:

**Hypothesis** of equally likely probability of variations: all the variations of the set \(\{1, 2, \ldots, m\}\) took \(n\) by \(n\) has the same probability to be considered.

If \(t_n^m \in B_n^m\) then \(\binom{n}{k}\) different subsets of size \(n\) of \(\{1, 2, \ldots, m\}\) can be permuted, and

\[
P(t_n^m) = \frac{1}{\binom{n}{k}} P'(t) \tag{3.3.1}
\]

can be defined being \(P'\) the probability distribution of classical model which can be computed starting from the shape of tree \(t\). The idea is to divide this probability \(P'\), which is a function of the shape, by the number of possible set of labels. Consider

\[
t_n^m = \begin{pmatrix}
t_l \\
1 \ldots k - 1 \\
\end{pmatrix} \begin{pmatrix}
t_r \\
k + 1 \ldots m \\
\end{pmatrix}
\]

being \(|t| = n, \quad |t_l| = l, \quad |t_r| = r\). The equations (3.3.1) and the recursive definition of \(P'\) (2.1.1) allows the:

**Definition 3.3.1** the probability of a tree \(t_n^m \in B_n^m\) is

\[
P \left( \begin{array}{c}
\Box \\
1 \ldots m
\end{array} \right) = 1 \quad \forall m \quad n = 0
\]

\[
P \left( \begin{array}{c}
\begin{pmatrix}
t_l \\
1 \ldots k - 1 \\
\end{pmatrix} \\
\begin{pmatrix}
t_r \\
k + 1 \ldots m \\
\end{pmatrix}
\end{array} \right) = \frac{(k-1)(m-k)}{n\binom{m}{k}} P \left( \begin{array}{c}
\begin{pmatrix}
t_l \\
1 \ldots k - 1 \\
\end{pmatrix}
\end{array} \right) P \left( \begin{array}{c}
\begin{pmatrix}
t_r \\
k + 1 \ldots m \\
\end{pmatrix}
\end{array} \right)
\]

From (3.3.1), if \(t_n^m \in B_n^m\) then \(P(t_n^m) = P'(t_n^m)\), coinciding both distributions.
4. Needed tools

4.1. Induced, probabilistic and random algorithms

Consider the algorithm $A : B_n^m \rightarrow B_p^j$ and the equivalent relation $\equiv$ defined over all tree-sets, as for example the shape relation. If the algorithm is compatible with the equivalent relation

$$t_n^m \equiv \tilde{t}_n^m \implies A(t_n^m) \equiv A(\tilde{t}_n^m)$$

then an induced algorithm $A^\equiv$ can be defined

$$A^\equiv : \frac{B_n^m}{\equiv} \rightarrow \frac{B_p^j}{\equiv}$$

$$\begin{array}{c}
[t_n^m] \\
\mapsto \quad A^\equiv ([t_n^m])
\end{array}$$

If $A$ is a probabilistic algorithm, then

$$A(t_n^m) = \alpha_1(T_i)^j_p + \ldots + \alpha_s(T_i)^j_p \quad \frac{\alpha_1 + \ldots + \alpha_s = 1}{\alpha_1 + \ldots + \alpha_s = 1}$$

that means that its images are linear combination of trees from $B_p^j$ with real coefficients. This set of images will be denoted with a hat as $\hat{B}_p^j$, then $A : B_n^m \rightarrow \hat{B}_p^j$. The algorithm can be naturally extended over $A : B_n^m \rightarrow \hat{B}_p^j$. The coefficients $\alpha_i$ can be understood as the probability to obtain the tree $(T_i)^j_p$ starting from the application of $A$ over the tree $t_n^m$, then $\alpha_i$ can be denoted as $\text{prob} (A(t_n^m) = (T_i)^j_p)$. Then the algorithm can be expressed as

$$A(t_n^m) = \sum_{T_i^j_p \in \hat{B}_p^j} \text{prob} (A(t_n^m) = T_i^j_p) \quad T_i^j_p \quad \text{being} \quad \sum_{T_i^j_p \in \hat{B}_p^j} \text{prob} (A(t_n^m) = T_i^j_p) = 1$$

by omitting the subscript $i$.

If this probabilistic algorithm is compatible with an equivalent relation

$$t_n^m \equiv \tilde{t}_n^m \implies \text{prob} (A(t_n^m) \in \frac{T_i^j_p}{\equiv}) = \text{prob} (A(\tilde{t}_n^m) \in \frac{T_i^j_p}{\equiv}) \quad \forall \frac{T_i^j_p}{\equiv} \in \frac{B_p^j}{\equiv}$$

then it induces an algorithm over the class $\frac{B_n^m}{\equiv}$, and the new coefficients will be denoted as

$$\text{prob} (A^\equiv ([t_n^m]) = [T_i^j_p])$$

Let $A$ an algorithm defined

$$A : \{a_1, a_2, \ldots, a_s\} \times B_n^m \rightarrow B_p^j$$

$$a_i \quad , \quad t_n^m \quad \mapsto \quad A(a_i, t_n^m)$$

over two sets as the insertion and deletion algorithms are. If there exist a probability function $\text{prob}$ defined over the set $\{a_1, a_2, \ldots, a_s\}$ the algorithm can be probabilized yielding a random one:

**Definition 4.1.1** we will say that we randomize the algorithm $A$ obtaining the random algorithm $AR$ by making

$$AR : B_n^m \rightarrow B_p^j$$

$$t_n^m \rightarrow AR(t_n^m) = \sum_{i=1}^{s} \text{prob}(a_i) A(a_i, t_n^m)$$

where $\text{prob}(a_i)$ can be understood as $\text{prob}(AR(t_n^m) = (T_i)^j_p)$.

We want to underline that, although the algorithm $A$ has two parameters, the label and the tree, the randomized algorithm $AR$ only has one parameter: the tree. Then this kind of algorithms could be considered as a function from/to the sets represented in diagram 1, and can be extended over the set of linear combination of trees $AR : \hat{B}_n^m \rightarrow \hat{B}_p^j$.
4.2. Induced probabilities

Consider the probabilistic algorithm \( A : B_n^m \rightarrow B_p^q \), then the probability defined in \( B_n^m \) induces a probability in \( B_p^q \) defined as

\[ P_A : B_p^q \rightarrow [0,1] \]

\[ T_p^q \rightarrow P_A(T_p^q) = \sum_{t_n^m \in B_n^m} \text{prob} (A(t_n^m) = T_p^q) P(t_n^m). \]

An equivalent relation defined over the tree-set induces over the quotient set the probability

\[ P^\equiv : B_n^m / \equiv \rightarrow [0,1] \]

\[ [t_n^m] \rightarrow P^\equiv([t_n^m]) = \sum_{t_n^m \in [t_n^m]} P(t_n^m) . \]

that is the addition of the probability of trees of the class.

**EXAMPLE 3.3.2:** We can add the probability of trees, in the example 3.1.1, with the same shape to obtain the probability induced by the equivalent relation shape, denoted \( P^{SH} \):

\[
P^{SH} \left( \begin{array}{c}
\ast \\
1 \ldots 3
\end{array} \right) = P \left( \begin{array}{c}
2 \\
1 \ldots 3
\end{array} \right) + P \left( \begin{array}{c}
3 \\
1 \ldots 3
\end{array} \right) + P \left( \begin{array}{c}
3 \\
2 \ldots 3
\end{array} \right)
\]

Then the probability induced by a probabilistic algorithm and an equivalent relation \( \equiv \), denoted \( P^\equiv_A \) will be

\[ P^\equiv_A : B_p^q / \equiv \rightarrow [0,1] \]

\[ [T_p^q] \rightarrow P^\equiv_A([T_p^q]) \]

and should be computed by

\[ P^\equiv_A([T_p^q]) = \sum_{T_p^q \in [T_p^q]} \sum_{t_n^m \in B_n^m} \text{prob} (A(t_n^m) = T_p^q) P(t_n^m) \quad (4.2.1) \]

**Lemma 4.2.1** If a probabilistic algorithm \( A : B_n^m \rightarrow B_p^q \) is compatible with the equivalent relation \( \equiv \) then \( P^\equiv_A = P_A^\equiv \). (Note: we should denote \( P_A^\equiv \) as \( P_A^\equiv \), but we think that taking out the superscript increases the readability of the paper).

**Proof:** If the second summation of (4.2.1) is broke over the equivalent class \([t_n^m]\) and the order of additions is changed

\[ P^\equiv_A([T_p^q]) = \sum_{[t_n^m] \in B_n^m / \equiv} \sum_{T_p^q \in [T_p^q]} \sum_{t_n^m \in [t_n^m]} \text{prob} (A(t_n^m) = T_p^q) P(t_n^m) \]

by doing the last summation

\[ \sum_{[t_n^m] \in B_n^m / \equiv} \sum_{T_p^q \in [T_p^q]} \text{prob} (A(t_n^m) \in [T_p^q]) P(t_n^m) . \]

If \( A \) induces \( A^\equiv \), then

\[ \sum_{[t_n^m] \in B_n^m / \equiv} \text{prob} (A^\equiv([t_n^m]) = [T_p^q]) \sum_{t_n^m \in [t_n^m]} P(t_n^m) . \]

As the last summation is the probability of quotient class \( P^\equiv ([t_n^m]) \), then the result expression is \( P_A^\equiv \).
4.3 Probabilistic functions

A Probabilistic Function (for short PF) \( f \) over the set \( B_n^m \) is defined as \( f : B_n^m \rightarrow [0, 1] \). The linear combination and the tree combination of PFs over the same sets will be defined by extending the definitions suggested in [Ma,Me 90].

**Definition 4.3.1** The linear combination of two PFs \( f \) and \( g \), both defined over \( B_n^m \), denoted by \( af + bg \), is a PF defined over \( B_n^m \) given by

\[
(af + bg)(T) = a \cdot f(T) + b \cdot g(T) \quad \text{with} \quad 0 \leq a, b \leq 1 \quad \text{and} \quad a + b = 1
\]

The definition of tree combination is needed because we will apply PFs over sets recursively defined.

**Definition 4.3.2** Let \( f \) and \( g \) PFs over the sets \( B_i^j \) and \( B_i^j \). The following tree combination can be constructed

\[
\begin{array}{c}
\text{k} \\
\downarrow \downarrow \\
\text{f} \\
\downarrow \\
\text{g} \\
\text{m} \end{array} : B_n^m \rightarrow [0, 1]
\]

being \( k = i + 1 \) and \( m = i + j + 1 \), and denoted with a pair of lines as

\[
\begin{array}{c}
k \\
\downarrow \\
f \\
\downarrow \\
g \\
\text{m} \end{array}
\left[
\begin{array}{c}
\begin{array}{c}
T_i \\
1 \ldots K - 1
\end{array} \\
T_r - K
\end{array}
\right] =
\delta(k, K) \delta(l, L) \delta(r, R) \cdot f \left( \left[
\begin{array}{c}
1 \ldots K - 1
\end{array}
\right] \right) \cdot g \left( \left[
\begin{array}{c}
1 \ldots m - K
\end{array}
\right] \right)
\]

being \( |T_i| = L \) and \( |T_r| = R \) and \( \delta \) the generalized Delta of Kronecker function:

\[
\delta(z, y) = \begin{cases} 
0 & \text{if } z \neq y \\
1 & \text{if } z = y 
\end{cases}
\]

4.4 Invariances

The invariances of a probability are going to be recalled from [Ma,Me 90]. They could be considered as a generalization of insensibility idea applied in [Knu 77].

Let \( A : B_n^m \rightarrow B_n^m \).

**Definition 4.4.1** We shall say that the probability \( P \) is strong invariant under the algorithm \( A \), or \( A \) permits the strong invariance of the probability \( P \) iff the induced probability is the same as the defined probability

\[
\forall T_p^q \in B_p^q \quad P(T_p^q) = P_A(T_p^q).
\]

By considering the equivalent relation \( \equiv \)

**Definition 4.4.2** We shall say that the probability \( P \) is \( \equiv \)-weak invariant under the algorithm \( A \), or \( A \) permits the \( \equiv \)-weak invariance of \( P \) iff the induced probability and the defined one are equals over every class from quotient set \( B_p^q / \equiv \)

\[
\forall [T_p^q] \in B_p^q / \equiv \quad P(\left[ T_p^q \right]) = P_A(\left[ T_p^q \right]).
\]

This invariances cannot be widely applied in the \( PA \) model due to the lineal network of tree sets instead, the model \( VM \) allow the widely application of invariances due its triangular network of trees. The following lemmas can be easily proved.
Lemma 4.4.1 Let $\equiv$ an equivalent relation over $\mathcal{B}_p^m$, and $\mathcal{A}^\equiv$ the probabilistic algorithm induced by $\mathcal{A} : \mathcal{B}_p^m \rightarrow \mathcal{B}_p^m$, then $\mathcal{P}^\equiv$ is strong invariant under $\mathcal{A}^\equiv$ iff $\mathcal{P}$ is $\equiv$-weak invariant under $\mathcal{A}$.

Lemma 4.4.2 Let the composition $\mathcal{B}_p^m \xrightarrow{\mathcal{A}} \mathcal{B}_p^m \xrightarrow{\hat{\mathcal{A}}} \mathcal{B}_p^m$. If the algorithms $\mathcal{A}$ and $\hat{\mathcal{A}}$ are strong invariants then $\hat{\mathcal{A}} \circ \mathcal{A}$ is strong invariant.

Corollary 4.4.1 If the algorithm $\mathcal{A}$ permits the strong invariance of $\mathcal{P}$, then $\mathcal{A}^k$ also permits its strong invariance for $k \geq 1$.

Corollary 4.4.2 If the algorithm $\mathcal{A}$ permits the $\equiv$-weak invariance and is compatible with the equivalent relation then $\mathcal{A}^k$ also permits the $\equiv$-weak invariance.

Lemma 4.4.3 If the algorithm $\mathcal{A}$ is strong invariant, and $\hat{\mathcal{A}}$ is $\equiv$-weak invariant, the $\hat{\mathcal{A}} \circ \mathcal{A}$ is weak invariant. But if $\mathcal{A}$ is $\equiv$-weak invariant and $\hat{\mathcal{A}}$ is strong invariant, the composition $\hat{\mathcal{A}} \circ \mathcal{A}$ can not be $\equiv$-weak invariant. This suggests that the insertions after deletions destroy some kinds of invariance properties, instead the inverse compositions lets the weak invariance.

4.5 Equivalent relation shape

Let $\mathcal{V}_{m,n}$ the set of variations of $m$ labels $\{1,2,\ldots,m\}$ took $n$ by $n$, and $\mathcal{P}_n$ the set of permutations of $\{1,2,\ldots,n\}$. A function which leaves the relative order of labels unchanged will be defined:

Definition 4.5.1 The projection $\rho$:

$$
\rho : \mathcal{V}_{m,n} \rightarrow \mathcal{P}_n
$$

so that, if $v = (v_1, v_2, \ldots, v_n)$ and $p = (p_1, p_2, \ldots, p_n)$ then $p_i < p_i$ iff $v_i < v_i$.

This definition allows to define

Definition 4.5.2 The application $\pi$:

$$
\pi : \mathcal{B}_p^m \rightarrow \mathcal{B}_p^m
$$

so that, if $t^m_n$ has been constructed by the variation $v$ then $\left(\begin{array}{c} T \\ 1 \ldots n \end{array}\right)$ must has been constructed by the permutation $\rho(v)$.

Finally we define

Definition 4.5.3 Two trees will be shape equivalents:

$$
t^m_n \equiv_{SH} \hat{t}^m_n \iff \pi(t^m_n) = \pi(\hat{t}^m_n)
$$

Lemma 4.5.1 Let $t^m_n \in \mathcal{B}_p^m$, then $P^{SH}(\{t^m_n\}) = P(t^m_n)$

Lemma 4.5.2 The probability defined for the model $\mathcal{V}_M$ in 3.3.1 is a good one.

4.6 Properties of the internal path length

The internal path length of a tree $\mathcal{IPL}(t)$ is the complexity measure that yields the addition of the length of all paths of the tree, being the length of path the number of nodes we find into. $\mathcal{IPL}(\mathcal{B}_p^m)$ means the average value of the $\mathcal{IPL}$ with the probability $P$ defined in our model of variations. The following lemmas can be easily proved.

Lemma 4.6.1 If $t^m_n \equiv_{SH} \hat{t}^m_n$ then $\mathcal{IPL}(t^m_n) = \mathcal{IPL}(\hat{t}^m_n)$.

Lemma 4.6.2 We can compute the $\mathcal{IPL}(\mathcal{B}_p^m)$ as a class function module shape:

$$
\mathcal{IPL}(\mathcal{B}_p^m) = \sum_{\{t^m_n\} \in \mathcal{B}_p^m} \mathcal{IPL}(\{t^m_n\}) \cdot \mathcal{P}^{SH}(\{t^m_n\})
$$
Lemma 4.6.3 \( TPL(B_n^m) = TPL(B_n^m) \quad \forall m \geq n \)

This last lemma proves that the \( TPL \) will be the same for all set of every diagonal of Diagram 1:

\[
\begin{array}{c}
B_n^m \\
B_n^{n+1} \\
B_n^{n+2} \\
\vdots \\
B_n^{m}
\end{array}
\]

5. Insertion

If a label is going to be inserted, it will be inserted at the left or at the right son depending on the relative value between the root and the label. Thus, the algorithm can be expressed:

\[
INS\left( y, \left( \begin{array}{c} k \\
\frac{t_l}{1 \ldots k - 1} \\
\frac{t_r}{k + 1 \ldots m} \end{array} \right) \right) =
\]

\[
\delta(1 \leq y \leq k) \\
INS\left( y, \left( \begin{array}{c} t_l \\
\frac{1 \ldots k - 1}{1 \ldots k - 1} \end{array} \right) \right) \\
INS\left( y, \left( \begin{array}{c} t_r \\
\frac{k + 1 \ldots m + 1}{k + 1 \ldots m} \end{array} \right) \right)
\]

\[
+ \delta(k + 1 \leq y \leq m + 1) \\
INS\left( y, \left( \begin{array}{c} t_l \\
\frac{1 \ldots k - 1}{1 \ldots k - 1} \end{array} \right) \right) \\
INS\left( y, \left( \begin{array}{c} t_r \\
\frac{k + 1 \ldots m}{k + 1 \ldots m} \end{array} \right) \right)
\]

where a generalized version of Delta of Kronecker has been applied

\[
\delta(exp) = \begin{cases} 
0 & \text{if exp = false} \\
1 & \text{if exp = true}
\end{cases}
\]

The algorithm can be randomized yielding

\[
INSR : B_n^m \longrightarrow B_n^{m+1} \\
t_n^m \longrightarrow INSR(t_n^m) = \sum_{1 \leq y \leq m+1} prob(select(y)) INS(y, t_n^m)
\]

which with the equally likely probability of labels yields

\[
INSR(t_n^m) = \frac{1}{m+1} \sum_{1 \leq y \leq m+1} INS(y, t_n^m)
\]

The probability induced by this algorithm can be expressed

\[
P_{INSR}(T_n^{m+1}) = \sum_{t_n^m \in T_n^m} prob(INSR(t_n^m) = T_n^{m+1}) P(t_n^m)
\]

where \( prob(INSR(t_n^m) = T_n^{m+1}) \) means the probability that the tree \( T_n^{m+1} \) can be satisfied by inserting some label on the tree \( t_n^m \). This probability can be interpreted by a probabilistic function \( INSR(t_n^m) \) which is applied over the tree \( T_n^{m+1} \).
Lemma 5.1 The probability function $\text{INSR}(t^n_m)$ is

\[
\text{INSR}\left(\begin{pmatrix}
t_l \\
1 \ldots k - 1
\end{pmatrix}
\begin{pmatrix}
k + 1 \\
k + 1 \ldots m
\end{pmatrix}
\right) = \frac{k}{m + 1}
\]

\[
\text{INSR}\left(\begin{pmatrix}
t_l \\
1 \ldots k - 1
\end{pmatrix}
\begin{pmatrix}
t^+1_r \\
k + 1 \ldots m + 1
\end{pmatrix}
\right)
\]

\[+ \frac{m - k + 1}{m + 1} \text{INSR}\left(\begin{pmatrix}
t_l \\
1 \ldots k - 1
\end{pmatrix}
\begin{pmatrix}
t_r \\
k + 1 \ldots m
\end{pmatrix}
\right)
\]

Lemma 5.2 The probability induced by an insertion is

\[
P_{\text{INSR}}\left(\begin{pmatrix}
T \\
1 \ldots m + 1
\end{pmatrix}
\right) = L^{(k-1)}_{\frac{(m-k+1)}{n}} P_{\text{INSR}}\left(\begin{pmatrix}
T_l \\
1 \ldots K - 1
\end{pmatrix}
\right) P\left(\begin{pmatrix}
T^+1_r \\
1 \ldots m - K + 1
\end{pmatrix}
\right)
\]

\[+ R^{(k-1)}_{\frac{(m-k+1)}{n}} P\left(\begin{pmatrix}
T_l \\
1 \ldots K - 1
\end{pmatrix}
\right) P_{\text{INSR}}\left(\begin{pmatrix}
T^+1_r \\
1 \ldots m - K + 1
\end{pmatrix}
\right)
\]

Proof: Applying the function of lemma 5.1 over the tree

\[
T = \begin{pmatrix}
T_l \\
1 \ldots K - 1
\end{pmatrix}
\begin{pmatrix}
K \\
K + 1 \ldots m + 1
\end{pmatrix}
\]

as definition 4.3.2 does

\[
\frac{k}{m + 1} \sum_{t \in \mathbb{G}^m_n} \delta(r, R) \delta(l + 1, L) \delta(k + 1, K) \text{INSR}(t_l, T_l) \delta(t^+1_r, T^+1_r) P(t)
\]

\[+ \frac{m - k + 1}{m + 1} \sum_{t \in \mathbb{G}^m_n} \delta(r + 1, R) \delta(l, L) \delta(k, K) \text{INSR}(t^+1_r, T^+1_r) \delta(t_l, T_l) P(t)
\]

Like the recursive expression of the probability distribution is known

\[
= \frac{k}{m + 1} \sum_{t \in \mathbb{G}^m_n} \delta(k + 1, K) \text{INSR}(t_l, T_l) \delta(t^+1_r, T^+1_r) \frac{(k-1)(m-k)}{n(m-n)} P\left(\begin{pmatrix}
t_l \\
1 \ldots k - 1
\end{pmatrix}
\right) P\left(\begin{pmatrix}
t^+1_r \\
1 \ldots m - k
\end{pmatrix}
\right)
\]

\[+ \frac{m - k + 1}{m + 1} \sum_{t \in \mathbb{G}^m_{n-1}} \delta(k, K) \text{INSR}(t_r, T_r) \delta(t_l, T_l) \frac{(k-1)(m-k)}{n(m-n)} P\left(\begin{pmatrix}
t_l \\
1 \ldots k - 1
\end{pmatrix}
\right) P\left(\begin{pmatrix}
t^+1_r \\
1 \ldots m - k
\end{pmatrix}
\right)
\]

By computing the Delta de Kronecker function
\[ \begin{align*}
&= \frac{K - 1}{m + 1} \sum_{t_i \in E_{\mathcal{L}, j}^{K - 2}} I N S R(t_i, T_i) \frac{(K - 2)(m - K + 1)}{R_n} P \left( \left( 1 \ldots K - 2 \right) \right) P \left( \left( \frac{t_i}{1 \ldots m - K + 1} \right) \right) \\
&\quad + \frac{m - K + 1}{m + 1} \sum_{t_r \in E_{\mathcal{L}, j}^{K - 1}} I N S R(t_r^{-K}, T_r^{-K}) \frac{(K - 1)(m - K + 1)}{R_n} P \left( \left( 1 \ldots K - 1 \right) \right) P \left( \left( \frac{t_r^{-K}}{1 \ldots m - K} \right) \right)
\end{align*} \]

and given out of summation all expressions it can be done

\[ \begin{align*}
&= \frac{K - 1}{m + 1} \frac{(K - 2)(m - K + 1)}{R_n} P \left( \left( 1 \ldots m - K + 1 \right) \right) \sum_{t_i \in E_{\mathcal{L}, j}^{K - 2}} I N S R(t_i, T_i) P \left( \left( \frac{t_i}{1 \ldots K - 2} \right) \right) \\
&\quad + \frac{m - K + 1}{m + 1} \frac{(K - 1)(m - K + 1)}{R_n} P \left( \left( 1 \ldots K - 1 \right) \right) \sum_{t_r \in E_{\mathcal{L}, j}^{K - 1}} I N S R(t_r^{-K}, T_r^{-K}) P \left( \left( \frac{t_r^{-K}}{1 \ldots m - K} \right) \right)
\end{align*} \]

the following expression can be obtained by replacing the induced probability of sons

\[ \begin{align*}
&= \frac{K - 1}{m + 1} \frac{(K - 2)(m - K + 1)}{R_n} P \left( \left( 1 \ldots m - K + 1 \right) \right) P_{INSR} \left( \left( \frac{t_i}{1 \ldots K - 1} \right) \right) \\
&\quad + \frac{m - K + 1}{m + 1} \frac{(K - 1)(m - K + 1)}{R_n} P \left( \left( 1 \ldots K - 1 \right) \right) P_{INSR} \left( \left( \frac{t_r^{-K + 1}}{1 \ldots m - K + 1} \right) \right).
\end{align*} \]

And by applying the following identities the lemma is proved.

\[ i \binom{i - 1}{j - 1} = j \binom{i}{j} \Rightarrow \begin{cases} (K - 1)(K - 2) = L(K - 1) \\
(m - K + 1)(m - K) = R(m - K + 1) \\
(m + 1)(m + 1) = (m + 1)(m + 1) \end{cases} \]

\[ \square \]

**Theorem 5.1** The probability \( P \) is strong invariant under the random insertion algorithm \( I N S R \).

**Proof:** It will be proved by induction: suppose

\[ P_{INSR} \left( \left( \frac{T}{1 \ldots i} \right) \right) = P \left( \left( \frac{T}{1 \ldots i} \right) \right) \quad 0 < i \leq m \]

then the induced probability become

\[ P_{INSR} \left( \left( \frac{T}{1 \ldots m + 1} \right) \right) = \frac{(n + R)(m - K + 1)}{R_n} P \left( \left( \frac{t_i}{1 \ldots K - 1} \right) \right) P \left( \left( \frac{t_i^{-K + 1}}{1 \ldots m - K + 1} \right) \right) \]

Like \((L + R) = n\) then the last expression gives the \( P \left( \left( \frac{T}{1 \ldots m + 1} \right) \right) \). The basis of induction is proved by:

\[ P \left( \left( \frac{1 \ldots i}{1 \ldots i} \right) \right) = \frac{1}{i} \]

\[ P_{INSR} \left( \left( \frac{1 \ldots i}{1 \ldots i} \right) \right) = \sum_{t_i \in E_{\mathcal{L}, j}^{K - 1}} I N S R \left( \left( \frac{1 \ldots i}{1 \ldots i - 1} \right) \right) - \left( \frac{1 \ldots i}{1 \ldots i - 1} \right) \right) P \left( \left( \frac{1 \ldots i - 1}{1 \ldots i - 1} \right) \right) = \frac{1}{i} \]

\[ \square \]

**Corollary 5.1** The probability \( P \) is strong invariant below the random insertion algorithm \( I N S R^k \) for \( k \geq 0 \).
6 Deletions

The Hibbart deletion will be considered, and its randomized version DELR will be defined by

\[
\text{DELR} : B_n^m \rightarrow \tilde{B}_{n-1}^m
\]

\[
t_n^m \mapsto \text{DELR}(t_n^m) = \sum_{x \in I_n^m} \text{prob}(\text{select } x) \text{DEL}(x, t_n^m)
\]

From the hypothesis of equally likely probability of labels the algorithm become:

\[
\text{DELR}(t_n^m) = \frac{1}{n} \sum_{x \in I_n^m} \text{DEL}(x, t_n^m)
\]

which is equal to

\[
\text{DELR}(t_n^m) = \frac{1}{n} (\text{DEL}(x_1, t_n^m), \ldots, \text{DEL}(x_1, t_n^m)) \quad \text{for } x_1 < x_2 < \ldots < x_n \quad (6.1)
\]

The item \(\text{DELR}(t_n^m)\) can be interpreted as a probabilistic function and can be applied over the set \(B_n^m\) as it has been done with the insertion algorithm. But we do not make the same theoretical study as in insertions, because the \(P_H\) expression is very complex. An alternative path can be used by applying the compatibility between the algorithm and the equivalent relation shape:

**Lemma 6.1** The algorithm DELR induces an algorithm DELR\(^{SH}\)

\[
\text{DELR}^{SH} : B_n^m / SH \rightarrow \tilde{B}_{n-1}^m / SH
\]

**Proof:** The following expression must be proved

\[
t_n^m \equiv_{SH} \tilde{t}_n^m \implies \text{DELR}(t_n^m) \equiv_{SH} \text{DELR}(\tilde{t}_n^m)
\]

but, by equation (6.1) and the definition of the equivalence 4.5.3, is equivalent to prove

\[
\pi(t_n^m) = \pi(\tilde{t}_n^m) \implies \pi(\text{DEL}(x_1, t_n^m)) = \pi(\text{DEL}(x_1, \tilde{t}_n^m))
\]

We do

\[
\pi(\text{DEL}(x_1, t_n^m)) = \text{DEL}(\rho(x_1), \pi(t_n^m)) = \text{DEL}(\rho(x_1), \pi(\tilde{t}_n^m))
\]

\[
= \pi(\text{DEL}(\tilde{\rho}^{-1} \circ \rho(x_1), \tilde{t}_n^m)) = \pi(\text{DEL}(\tilde{x}_1, \tilde{t}_n^m))
\]

where the equality \(\tilde{\rho}^{-1} \circ \rho(x_1) = \tilde{x}_1\) has been applied.

\(\square\)

**Theorem 6.1** The probabilistic random algorithm DELR\(^{SH}\) permits the strong invariance of the induced probability \(P_{SH}\).

**Proof:** The expression \(P_{DELR}^{SH} = P_{SH}\) must be proved. It is equal at

\[
\forall [T_{n-1}^m] \in B_{n-1}^m / SH \quad P_{DELR}^{SH}([T_{n-1}^m]) = P_{SH}^{SH}([T_{n-1}^m])
\]

Applying the induced probability lemma 4.6.3 we obtain

\[
P_{DELR}^{SH}([T_{n-1}^m]) = \sum_{[t_n^m] \in B_n^m / SH} \text{prob}(\text{DELR}^{SH}[t_n^m] = [T_{n-1}^m]) \quad P_{SH}^{SH}(t_n^m)
\]

and like \(DELR^{SH}\) can be interpreted as a probabilistic function, the following identity must be proved

\[
\sum_{[t] \in B_n^m / SH} \text{DELR}^{SH}([t], [T]) \quad P_{SH}^{SH}([t]) = P_{SH}^{SH}([T])
\]

16
But this equality can be viewed as the probability induced by only one deletion in the Model of Permutations. This fact can be done by applying the following table

\[
\begin{array}{c|c}
V M & P M \\
\hline
P^{SH}(T) & P(T) \\
\sum_{i \in \mathcal{B}_n^{SH}} & \sum_{i \in \mathcal{B}_n} \\
DEL_{R}^{SH}([T],[T]) & DEL_{R}(t,T)
\end{array}
\]

which translates our equation to \( \sum_{i \in \mathcal{B}_n} DEL_{R}(t,T)P'(t) = P'(T) \) which is the same as \( P_{R}^{SH}(T) = P'(T) \). By considering the Hibbart Theorem, the Hibbart deletion algorithm verifies this equation.

\[ \square \]

**Corollary 6.1** The probability \( P \) is SH-weak invariant below the algorithm \( DEL_{R} \).

**Corollary 6.2** The probability \( P \) is SH-weak invariant below \( (DEL_{R})^{k} \) \( \forall k > 0 \).

7 The composition of random insertions with random deletions

We have now a triangular net of tree-sets with two well defined transformations \( INS_{R} \) and \( DEL_{R} \):

\[
\begin{array}{c}
\mathcal{B}_{i} \xrightarrow{INS_{R}} \mathcal{B}_{i+1} \\
\downarrow DEL_{R} \quad \forall i, j : j \geq i \geq 1 \\
\mathcal{B}_{i-1}
\end{array}
\]

being \( INS_{R} \) the random insertion algorithm and \( DEL_{R} \) the random deletion algorithm naturally extended over this sets, both having only one parameter: a lineal combination of trees. The first algorithm increases the size and the number of keys meanwhile the deletion algorithm decreases the size leaving unchanged the number of keys.

All the process in

\[(random\_deletion, random\_insertion)^{n}\]

can be strictly represented by a path in the Diagram 1:

**EXAMPLE 7.1.** the pattern of [Jo,Knu 78] papers \( INS_{R} INS_{R} (INS_{R} DEL_{R})^{n} \) is the path

\[
\begin{array}{cccccccc}
\mathcal{B}_{0} & \xrightarrow{INS_{R}} & \mathcal{B}_{1} & \xrightarrow{INS_{R}} & \mathcal{B}_{2} & \xrightarrow{INS_{R}} & \mathcal{B}_{3} & \mathcal{B}_{4} & \mathcal{B}_{5} & \cdots \\
\downarrow DEL_{R} & & & & & & & & & \\
\mathcal{B}_{0} & & \mathcal{B}_{1} & & \mathcal{B}_{2} & \xrightarrow{INS_{R}} & \mathcal{B}_{3} & \mathcal{B}_{4} & \cdots \\
\downarrow DEL_{R} & & & & & & & & & \\
\mathcal{B}_{0} & & \mathcal{B}_{1} & & \mathcal{B}_{2} & \xrightarrow{INS_{R}} & \mathcal{B}_{3} & \cdots \\
\downarrow DEL_{R} & & & & & & & & & \\
\mathcal{B}_{0} & & \mathcal{B}_{1} & & \mathcal{B}_{2} & \xrightarrow{INS_{R}} & \cdots \\
\end{array}
\]

As the \( INS_{R} \) algorithm allows the strong invariance of the probability, and there exist a deletion algorithm ([Me 90]) strong invariant also, for this pattern, then the composition allows the strong
invariance. Thus after intermixed random deletions and random insertions the probability distribution of trees will be random.

The corollary 5.1, proving that the probability is strong invariant under the algorithm $(INSR)^k$, means that a process that inserts $k$ labels

$$\mathcal{B}_i \xrightarrow{(INSR)^k} \mathcal{B}_{i+1} \xrightarrow{(INSR)^k} \cdots \xrightarrow{(INSR)^k} \mathcal{B}_{i+k} \iff \mathcal{B}_i \xrightarrow{(INSR)^k} \mathcal{B}_{i+k}$$

induces a probability distribution equal to the previous one defined. The corollary 6.2 means that the process that random deletes $k$ labels

$$\mathcal{B}_i \xrightarrow{DELR} \mathcal{B}_{i-1} \xrightarrow{DELR} \cdots \xrightarrow{DELR} \mathcal{B}_{i-k}$$

induces a probability distribution also equal to the previous defined probability, but over class of the the quotient set $\mathcal{B}_{i-k}/SH$, because is $SH$-weak invariant. By joining this two corollaries, the next theorem can be proved:

**Theorem 7.1** The probability $P$ is $SH$-weak invariant under the composition of algorithms

$$(DELR)^n \circ INSR^m : \mathcal{B}_i \xrightarrow{(DELR)^n \circ INSR^m} \mathcal{B}_{i+m-n} \iff \mathcal{B}_{i+n} \xrightarrow{(DELR)^k} \mathcal{B}_{i+n-k}$$

**Proof:** $P$ is strong invariant over $INSR^k$ by corollary 5.1 and by corollary 5.1 $(DELR)^n$ is $SH$-weak invariant, then the composition is $SH$-weak invariant by lemma 4.4.3.

\[\square\]

A real process of $m$ random insertions followed by $k$ random deletions began in $\mathcal{B}_0$, and can be represented by the diagram

$$\mathcal{B}_0 \xrightarrow{INSR^m} \mathcal{B}_{m-k} \xrightarrow{INSR^k} \mathcal{B}_m \xrightarrow{DELR^k} \mathcal{B}_{m-k}$$

where the sets $\mathcal{B}_{m-k}$ and $\mathcal{B}_m$ have the same value of the $TPL$ because they are in the same diagonal. This fact suggest the theorem:

**Theorem 7.2** The algorithm $(DELR)^k \circ INSR^m : \mathcal{B}_0 \longrightarrow \mathcal{B}_{m-k}$ induces the same $TPL$ as the induced by the algorithm $INSR^{m-k} : \mathcal{B}_0 \longrightarrow \mathcal{B}_{m-k}$.

**Proof:** Let $A = (DELR)^k \circ INSR^m$, and $TPL_A$ the induced $TPL$. Because the $TPL$ is a class function

$$\text{TPL}_A(\mathcal{B}_{m-k}) = \sum_{[t_{m-k}] \in \mathcal{B}_{m-k}/SH} IPL([t_{m-k}]) P_A^H([t_{m-k}])$$

\[18\]
as $A$ is $SH$-weak invariant by Theorem 7.1

$$
= \sum_{[r^m_{m-k}] \in B^m_{m-k}/_R} I MPL([r^m_{m-k}]) P^{SH}([r^m_{m-k}]) = ITPL(B^m_{m-k})
$$

applying the diagonal property of the $ITPL$. Denoting $C = INS R^{m-k}$, by the strong invariance of the insertion

$$
ITPL_k(B^m_{m-k}) = ITPL(B^m_{m-k})
$$

This theorem proves that the processes

$$
B^0 \xrightarrow{INS R^{m-k}} B^m_{m-k} \xrightarrow{INS R^k} \xrightarrow{DEL R^k} B^m_{m-k}
$$

induces the same value of the $ITPL$.

8 Conclusion and further research

This paper presents a new model of variations, which allows the generalization of known results and suggest new lines of research.

We have generalized the Hibbart Theorem to explain the pattern (1.2) which has never been proved due to the complexity of combinatorial computations. We can do it because our model allows the application of the theory of invariances, induced algorithms and probabilities.

Our model lets a new lecture of well-known results and concepts. The following table shows the equivalent means:

- random insertion $\iff$ $INS R$
- random deletion $\iff$ $DEL R$
- random trees $\iff$ Sets $B^m_n$ with the probability distribution $P$.

Also the classical theorem of Hibbart can be rewritten by saying:

The algorithm $INS R^m \circ DEL R$ is $shape$-weak invariant

or the Hibbart paradox can be explained by a confusion between strong and weak invariance applied over the deletion algorithm.

Further lines of research could be focused:

- can be proved the existence of a strong invariant deletion algorithm?, or a probabilistic one?
- what conditions allow that the composition of weak and strong invariant algorithms yields a weak invariant one.
- Can the paths in the triangular net be classified?

In summary, our model solves the Knott paradox by applying the Knuth idea but it does not explain the general dictionary problem. We think that the model capable to explain the general problem will be a generalization of the model of variations.

References


