# Project Report
# Choosing Optimal *p*,*q* values and Development of Index Structures for storing pq-gram profiles of trees

Varunesh Mishra(Y7492) Nitish Srivastava(Y7268)
Department of Computer Science and Engineering, IIT Kanpur

**Abstract**

The *pq* gram distance is a recently proposed approach for approximate matching of hierarchical data. It works by breaking down a tree into small subtrees and using them as a measure of similarity between two trees. The distance is efficiently computable and being parametrized by $p$ and $q$, has the ability to be tuned for giving different amount of importance to different factors for comparing two trees, such as similar structure vs similar data or similar parents vs similar children. However, the exact nature of this tradeoff remains unknown. There is also a need to build an efficient index structure for storing *pq* gram profiles of trees so that the measure can be used effectively in real database management systems. We address these issues and try to provide a rigorous analysis of the *pq* gram distance.

## 1 Introduction

The *pq*-gram distance is a recent approach for approximate matching of hierarchical data like XML proposed by Augsten et al [1]. Intuitively, the *pq*-grams of a tree are all its subtrees of a specific shape. Two trees are similar if they have many *pq*-grams in common. By adjusting the two parameters $p$ and $q$, which specify the shape of the *pq*-grams, relevance of the tree structure can be controlled. The *pq*-gram distance behaves differently from the tree edit distance for structural and local changes. It gives more weight to edit operations that cause big changes in the tree structure. At the same time the distance respects the triangle inequality, thus retaining the pruning property which is useful in building index structures. In this project we present a rigorous analysis of *pq* distance by analytically exploring its behavior on random trees. We propose a reference-based index structure for storing *pq*-gram profiles of trees. We then present experimental results on the performance of our indexing scheme. Section 2 introduces the terminology associated with *pq*-grams. Section 3 describes previous work done in this area. Scetion 4 elaborates our work on analysis pf *pq*=gram distance for dependence on $p$ and $q$. Section 5 describes proposed the reference-based indexing.

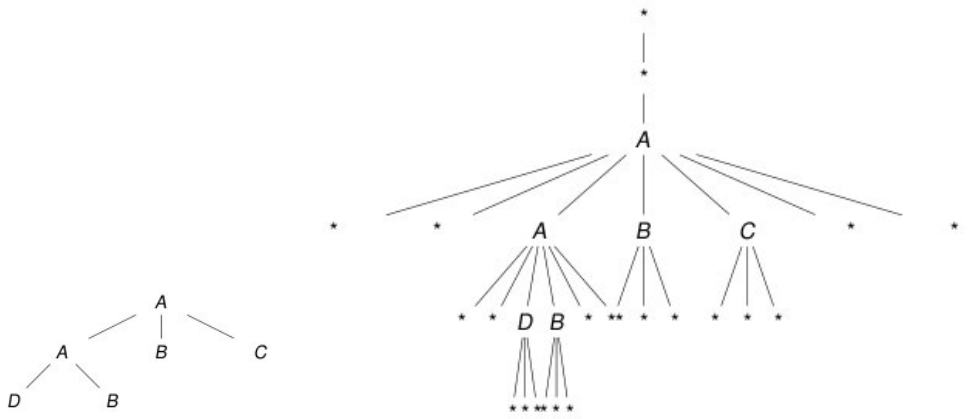
## 2 Preliminaries

In this section we recall the definitions of *pq* gram and *pq*-gram distance.

**Definition 1.** *pq-Extended-Tree. Let $T$ be a tree, and $p > 0$ and $q > 0$ be two integers. The pq-extended tree, $T^{p,q}$ , is constructed from $T$ by adding $p - 1$ ancestors to the root node, inserting $q - 1$ children before the first and after the last child of each non-leaf node, and adding q children to each leaf of $T$. All newly inserted nodes are dummy nodes (denoted by $*$) that do not occur in $T$.*

**Definition 2.** *pq-Gram. Let $T$ be a tree, $T^{p,q}$ the respective extended tree, $p > 0$, $q > 0$. A subtree of $T^{p,q}$ is a pq-gram $G$ of $T$ iff*

1. *$G$ has q leaf nodes and p non-leaf nodes,*
2. *all leaf nodes of $G$ are children of a single node $a \in N(G)$ with fanout q, called the anchor node,*
3. *the leaf nodes of $G$ are consecutive siblings in $T^{p,q}$ .*




(a) Tree $T$ (b) Extended Tree $T^{3,3}$

Figure 1: A sample tree $T$ along with its corresponding extended tree

**Definition 3.** *Label Tuple. Let G be a pq-gram with the nodes* $N(G) = \{v_1, \ldots, v_p, v_{p+1}, \ldots, v_{p+q}\}$*, where* $v_i$ *is the i-th node in preorder. The tuple* $\lambda^*(G) = (\lambda(v_1), \ldots, \lambda(v_p), \lambda(v_{p+1}), \ldots, \lambda(v_{p+q}))$ *is called the label tuple of G.*

where $\lambda(v)$ refers to the label of node $v$. Subsequently, if the distinction is clear from the context, we use the term *pq*-gram for both, the *pq*-gram itself and its representation as a label tuple.

**Definition 4.** *pq-Gram Index. Let P be the set of all pq-grams of a tree T,* $p > 0$*,* $q > 0$*. The pq-gram index,* $\mathcal{I}^{p,q}(T)$*, of T is defined as the bag of label tuples of all pq-grams of T, i.e.,* $\mathcal{I}^{p,q}(T) = \uplus_{G \in P} \lambda^*(G)$*.*

For the tree $T$ shown in Figure 1, $\mathcal{I}^{3,3}(T)$ is ,

```
(*,*,A,*,*,A)
(*,*,A,*,A,B)
(*,*,A,A,B,C)
(*,*,A,B,C,*)
(*,*,A,C,*,*)
(*,A,A,*,*,D)
(*,A,A,*,D,B)
(*,A,A,D,B,*)
(*,A,A,B,*,*)
(*,A,B,*,*,*)
(*,A,C,*,*,*)
(A,A,D,*,*,*)
(A,A,B,*,*,*)
```

In other words, it is the multiset of label tuples. *pq*-gram Index is called *pq*-gram profile in this report, to avoid confusion with the 'index' structure that we propose for storing them.

**Definition 5.** *pq gram distance. Let* $T_1$ *and* $T_2$ *be trees,* $\mathcal{I}_1 = \mathcal{I}^{p,q}(T_1)$*,* $\mathcal{I}_2 = \mathcal{I}^{p,q}(T_2)$*,* $p > 0, q > 0$*. The pq-gram distance,* $d^{p,q}(T_1, T_2)$*, between the trees* $T_1$ *and* $T_2$ *is defined as the symmetric difference between the respective profiles:*

$$d^{p,q}(T_1, T_2) = |\mathcal{I}_1 \uplus \mathcal{I}_2| - 2.|\mathcal{I}_1 \sqcap \mathcal{I}_2|$$

It is the number of *pq*-grams that are different between $\mathcal{I}_1$ and $\mathcal{I}_2$. The *pq*-gram distance is a pseudo-metric, i.e.

- Always non-negative; and zero for identical trees

- Symmetric,
- Triangle inequality holds.

Zero *pq*-gram distance does not mean that the two trees are the same. Hence the distance is not a metric. The pseudo-metric properties are useful as they can be used for developing pruning strategies for designing fast search algorithms.

**Definition 6.** *Normalized pq gram distance. Let $T_1$ and $T_2$ be trees, $\mathcal{I}_1 = \mathcal{I}^{p,q}(T_1)$, $\mathcal{I}_2 = \mathcal{I}^{p,q}(T_2)$, $p > 0, q > 0$. The normalized pq-gram distance, $dist^{p,q}_{norm}(T_1, T_2)$, between the trees $T_1$ and $T_2$ is defined as :*

$$d^{p,q}_{norm}(T_1, T_2) = \frac{d^{p,q}}{|\mathcal{I}_1 \uplus \mathcal{I}_2| - |\mathcal{I}_1 \boxplus \mathcal{I}_2|}$$

This normalization preserves the pseudo-metric properties of *pq*-distance. For a proof of the fact that $d^{p,q}_{norm}$ is pseudo-metric refer to [1].
The size of the *pq*-gram profile of a tree is linear in the size of the tree. Let $l$ and $i$ be the number of leaf and non-leaf nodes of $T$, respectively. Then

$$|\mathcal{I}^{p,q}(T)| = 2l + qi - 1 \tag{2.1}$$

i.e.,$|\mathcal{I}^{p,q}(T)| = O(n)$. See [1] for a proof.

# 3 Previous work

The most well known distance function for trees is the tree edit distance, which is de- fined as the minimum cost of edit operations (node insertion, node dele- tion, and rename) that change one tree into another [Tai 1979]. Zhang and Shasha [Zhang and Shasha 1989] present an algorithm to compute the tree edit distance in $O(n^2 min^2(l, h))$ time and $O(n^2)$ space for trees with $n$ nodes, $l$ leaves, and height $h$. Since in a tree both $l$ and h may be of size $O(n)$, the worst case complexity of this algorithm is $O(n^4)$. Later works have improved the worst case complexity of the tree edit distance algorithm [Klein 1998; Chen 2001; Demaine et al. 2007], but all of them use at least $O(n^3)$ time and $O(n^2)$ space and do not scale to large trees.

Embedding based methods have been proposed by Garofalakis and Kumar [3]. They however achieve a worst case distortion of $O(\log^2 n \log^* n)$, which is quite large for sizeable trees.

The *pq*-gram distance was proposed as a distance measure for ordered trees by Augsten et. al, [1]. The intuitive idea behind the distance was that similar trees should have more substructures common than dissimilar ones. These substructures are called *pq* grams. The *pq* distance algorithm computes all *pq* grams present in a tree along with the frequency of occurence of the corresponding name-tuples and stores them in a sequential index. Then in order to find the distance the proposed algorithm looks up the tables for the two trees and computes the distance based on the number of common name tuple pairs. The authors in [1] found that *pq*-gram distance is competitive both in terms of quality and efficiency with other tree edit distance approximations, and the *pq*-gram distance is scalable to large data sets. It runs in $O(n \log n)$ time and $O(n)$ space.
In [2] Augsten et. al. show that *pq* gram profiles can be maintained incrementally with changes in the data trees. It is not efficient to recompute the *pq* gram profile of a tree from scratch when some changes have been made. The proposed algorithm is based on the log of tree edit operations.
These works highlight the importance of *pq*-gram distance which is a fast and scalable distance measure as far as distance computation is concerned. However efficient storage and retrieval of *pq*-gram profiles is a major issue. Also the exact dependence of the distance on $p$ and $q$ needs further exploration. Our work tries to address these problems.

# 4 Choosing optimal $p$, $q$ values

The *pq*-gram distance is parametrized by two constants $p$ and $q$. These can be set by designer of the database system and must usually be determined by a domain expert who understands the underlying semantics of the

data and can assess how important various factors are in determining the distance between two trees. It can be seen intuitively that increasing $p$ and $q$ values makes the profile more 'rigid', i.e. more importance is being given to the structure of the tree as compared to the data. Decreasing them makes the profile insensitive to structure. As an extreme case, the values $p = 1$ and $q = 0$ would result in a $1, 0$-gram profile which would be just an unordered list of all node labels in the tree. Here the structure of the tree is completely lost while all the labels are retained. Besides, increasing $p$ relative to $q$ implies that more importance is being given to the ancestors than to the children,i.e., two nodes are being considered same only when they share a large number of ancestors. These intuitive ideas give an insight into the nature of the problem. We formulate rigourously the exact trade-off that exists in the system and how changes in $p$ and $q$ affect the final $pq$-gram distance. We will take a random tree and probablistically analyze it with repsect to changes in $p$ and $q$. Then we can optimize the value of $p$ and $q$ *based on requirement*. We think that there cannot be an absolute optimal value since the notion of similarity of trees itself is not absolute.(it depends on how important strcutural changes are relative to labels).

## 4.1 Construction of a random tree

Given a label set $\Sigma$, a random tree $T^k$ is constructed as follows

$T^k \leftarrow \{\text{root}\}, i \leftarrow 0$
**while** $i < k$ **do**
  Choose a leaf node $v$ from $T^k$ uniformly randomly from the set of leaf nodes
  Choose $\xi$ by sampling from $\mathcal{N}(\mu, \sigma)$ and rounding off to the nearest integer till the integer obtained is positive.
  Add $\xi$ leaves to $T^k$ as children of $v$.
  $i \leftarrow i + 1$
**end while**
Assign a label to each node of $T^k$ by sampling uniformly randomly from the set of labels $\Sigma$.

## 4.2 Analysis of random tree

Let $l_{h,k}$ be the number of leaves at height $h$ in $T^k$. Let $n_{h,k}$ be the number of internal nodes at height $h$ in $T^k$. Let $v_i$ be a positive random integer obtained by sampling the gaussian $\mathcal{N}(\mu, \sigma)$ in the $i$-th iteration while building the tree $T^k$ and rounding the result to the nearest integer(If the integer so obtained is not positive, the sampling is repeated). Then,

$$\begin{aligned}
l_{h,k+1} &= l_{h,k} - \text{leaves at height } h \text{ which become internal nodes in the next iteration} \\
&\quad + \text{leaves which are created at height } h \text{ by addition of leaves to some node at height } h-1 \\
&= l_{h,k} - \frac{l_{h,k}}{\sum_{h=0}^{\infty} l_{h,k}} + v_{k+1} \frac{l_{h-1,k}}{\sum_{h=0}^{\infty} l_{h,k}} \\
n_{h,k+1} &= n_{h,k} + \text{leaves at height } h \text{ which become internal nodes in the next iteration} \\
&= n_{h,k} + \frac{l_{h,k}}{\sum_{h=0}^{\infty} l_{h,k}}
\end{aligned}$$

After $k$ iterations, total number of internal nodes is $k$ since each iterations introduces exactly one internal node.

$$\sum_{h=0}^{\infty} n_{h,k} = k \tag{4.1}$$

Total number of nodes in the tree increases by $v_i$ in the $i$-th iteration. We initially start with one root node. Therefore, total number of nodes in the tree is

$$\sum_{h=0}^{\infty} (l_{h,k} + n_{h,k}) = 1 + \sum_{i=0}^{k} v_i \tag{4.2}$$

Total number of leaf nodes in $T^k$ is therefore,

$$\sum_{h=0}^{\infty} l_{h,k} = \sum_{i=0}^{k} v_i - k + 1 \tag{4.3}$$

Let $V_k = \sum_{i=0}^{k} v_i$. We get the following recursive equations for describing a random tree.

$$n_{h,k+1} = n_{h,k} + \frac{l_{h,k}}{V_k - k + 1} \tag{4.4}$$

$$l_{h,k+1} = l_{h,k} + \frac{v l_{h-1,k} - l_{h,k}}{V_k - k + 1} \tag{4.5}$$

$$l_{0,0} = 1, n_{0,0} = 0 \tag{4.6}$$

We use Z-transforms for solving these equations. In order to solve 4.5,

$$l_{h,k+1} - l_{h,k} = \frac{v_k l_{h-1,k} - l_{h,k}}{V_k - k + 1} \tag{4.7}$$

$$\Rightarrow (V_k - k + 1)(l_{h,k+1} - l_{h,k}) = v l_{h-1,k} - l_{h,k} \tag{4.8}$$

Let $X(z_1, z_2) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} l[m,n] z_1^{-m} z_2^{-n}$ be the $Z$-tranform of $l[m,n]$. Using

$$Z(nx[n]) = -z\frac{dX(z)}{dz} \tag{4.9}$$

$$Z(x[n-k]) = z^{-k}X(z) \tag{4.10}$$

$$\text{We get, } Z(nx[n+1]) = -z\frac{dZ(x[n+1])}{dz} = -z\frac{dzX(z)}{dz} \tag{4.11}$$

In order to simplify the above equation 4.8 we assume that the number of nodes created in each iteration is constant (= $v$) and not sampled from a gaussian. Then $V_k = kv$. Using equation 4.11, we get

$$(v-1)kl_{h,k+1} - (v-1)kl_{h,k} + l_{h,k+1} - l_{h,k} = v l_{h-1,k} - l_{h,k}$$
$$-(v-1)z_2(z_2\frac{\partial X}{\partial z_2} + X) + (v-1)z_2\frac{\partial X}{\partial z_2} + z_2 X = \frac{k}{z_1}X$$

Let $u(x,y) = X(z_1, z_2)$

$$y^2u_y + uy - ky^2u_y - kyu + kyu_y - yu_y + uy = \tfrac{ku}{x}$$

$$\Rightarrow \quad u_y(y^2 - y)(k-1) = u(\frac{k}{x} - 2y + ky)$$

$$\Rightarrow \quad u_y = \frac{u}{(k-1)y(y-1)}\left(\frac{k}{x} + (k-2)y\right)$$

$$\Rightarrow \quad \frac{du}{u} = \frac{1}{k-1}\left(k\frac{dy}{xy(y-1)} + (k-2)\frac{dy}{y-1}\right)$$

$$\Rightarrow \quad \frac{du}{u} = \frac{1}{k-1}\left(\frac{k}{x}\left(\frac{dy}{y-1} - \frac{dy}{y}\right) + (k-2)\frac{dy}{y-1}\right)$$

$$\Rightarrow \quad \ln u = \frac{1}{k-1}\left(\frac{k}{x}\ln\frac{y-1}{y} + (k-2)\ln(y-1)\right) + \frac{c_1}{x} + \ln c_2$$

$$\Rightarrow \quad u = \left(\frac{y-1}{y}\right)^{k/x(k-1)} .(y-1)^{(k-2)/(k-1)} e^{c_1/x} c_2$$

Using the initial conditions the constants can be eliminated. The coefficients of $x^{-h}y^{-k}$ in $u$ give $l_{h,k}$. To solve equation 4.4, we note that it is in the form of a difference equation.

$$\begin{aligned} n_{h,k+1} - n_{h,k} &= \frac{l_{h,k}}{V_k - k + 1} \\ \sum_{k=0}^{T-1}(n_{h,k+1} - n_{h,k}) &= \sum_{k=0}^{T-1} \frac{l_{h,k}}{V_k - k + 1} \\ n_{h,T} - n_{0,0} &= \sum_{k=0}^{T-1} \frac{l_{h,k}}{V_k - k + 1} \end{aligned}$$

We can again make simplifying assumptions for $V_k$ and use the Z-transform of $l_{h,k}$ to find the Z-transform of $n_{h,k}$. These statistics of the random tree as used in the next section. We have not been able to get closed form expressions for $l_{h,k}$ or $n_{h,k}$.

## 4.3 Probabilistic analysis

The most general form of a label-tuple of a $pq$ gram is

$$\begin{aligned} \lambda &= (*^a v_1, v_2, \ldots, v_b, *^c, w_1, w_2, \ldots, w_d, *^e), \text{ where } a+b=p,\, c+d+e=q, a, b, c, d, e \geq 0 \\ \lambda^p &= (*^a v_1, v_2, \ldots, v_b) \\ \lambda^q &= (*^c, w_1, w_2, \ldots, w_d, *^e) \end{aligned}$$

Where $v_i$'s and $w_i$'s are labels assigned to nodes to of the tree and * represents the null node. $v_b$ is the anchor node. By the construction of the tree, the probablity of the $p$-part occuring, for given values of $a$, $b$, $c$, $d$, $e$, $v_i$'s and $w_i$'s is independent of the probability of the $q$-part.
Therefore,

$$P(\lambda) = P(\lambda^p)P(\lambda^q)$$

Let $\delta_k$ be a random variable equal to the $pq$ gram distance between two random trees. The aim is to find the expected value and variance in $\delta_k$ as a function of $k$, $p$, $q$ and the parameters needed for modeling the random tree($\mu$, $\sigma$). The approach is as follows

- We try to find how many times each possible $pq$ gram $\lambda$ of the most general form occurs in $T^k$. That is for given values of $a$, $b$, $c$, $d$, $e$, $v_i$'s and $w_i$'s how many times does it occur in $T^k$ and with what variance. Let the associated random variable for the lexicographically ordered $pq$ grams $\lambda_i$ be $\eta_i$.
- To find this we find how many times each possible $pq$ gram $\lambda$ of the most general form occurs in $T^k$ with anchor node *at height* $h$. That is find $\eta_{i,h}$.

$$\eta_i = \sum_{h=0}^{k} \eta_{i,h}$$

- To find $\eta_{i,h}$ we find $\eta_{i,h,j}$ the number of times $\lambda_i$ occurs *at a given node position* $j$ at height $h$.

$$\eta_{i,h} = \sum_{j=0}^{\infty} \eta_{i,h,j}$$

## 4.4 Probability of occurence of $\lambda^p$($p$ part of $\lambda$)

If $a > 0$ then $\lambda^p$ can occur only at height $h = b - 1$. Therefore,

$$P(\lambda^p | a > 0) = \begin{cases} 0 & h \neq b-1 \\ \dfrac{1}{|\Sigma|^b} & h = b-1 \end{cases}$$

Else it can occur anywhere.

$$P(\lambda^p|a=0) = \frac{1}{|\Sigma|^p}$$

Combining the above,

$$P(\lambda^p) = \frac{1}{|\Sigma|^b}(\delta_{a,0} + \overline{\delta_{a,0}}\delta_{h,b-1})$$

where, $\delta_{i,j} = 1$ if $i = j$ , 0 otherwise. $\lambda^p$ can occur only once at a given anchor node position. Therefore expected number of times $\lambda^p$ occurs at a given node position and height is equal to the above probability.

## 4.5 Probability of occurence of $\lambda^q$($q$ part of $\lambda$)

If the anchor node is a leaf node then $d$ must be 0, ie

$$P(\lambda^q|v_b\text{is a leaf node}) = \begin{cases} 0 & d \neq 0 \\ 1 & d = 0 \end{cases}$$

Let $\xi$ be a random variable sampled from $\mathcal{N}(\mu, \sigma)$, rounded to the nearest integer
If the anchor node is not a leaf node, $d$ must be less than the number of children of the node ($\xi$).

$$P(\lambda^q|v_b\text{is not a leaf node}) = \begin{cases} 0 & d > \xi \\ \dfrac{1}{|\Sigma|^d} & l \leq \xi \end{cases}$$

The above can be combined as,

$$P(\lambda^q) = \frac{1}{|\Sigma|^d}\left(\delta_{d,0}P(leaf|h) + (1 - P(leaf|h))\,P\left(d \leq \xi\right)\right)$$

Note that a given $\lambda^q$ can occur several times at the same anchor node. Therefore the above is the probability of occurence of $\lambda^q$ *atleast once*. However, given that labels are assigned randomly and the size of the label set $\Sigma$ is large enough, the chance of same $\lambda^q$ repeating is quite small. We assume that the expected number of times a given $\lambda^q$ occurs at a given node position $j$ at height $h$ to be this probability.That is,

$$\begin{aligned}
E[\eta_{i,h,j}] &= P(\lambda^p)P(\lambda^q) \\
&= \frac{1}{|\Sigma|^{b+d}}(\delta_{a,0} + \overline{\delta_{a,0}}\delta_{h,b-1})\left(\delta_{d,0}P(leaf|h) + (1 - P(leaf|h))\,P\left(d \leq \xi\right)\right) \\
Var[\eta_{i,h,j}] &= P(\lambda)\left(1 - P(\lambda)\right)^2 + (1 - P(\lambda))\left(0 - P(\lambda)\right)^2 \\
&= P(\lambda)\left(1 - P(\lambda)\right) \\
&= E[\eta_{i,h,j}](1 - E[\eta_{i,h,j}])
\end{aligned}$$

## 4.6 Building up

We can use the results of the tree analysis from the previous section to find $P(leaf|h)$. Then using the statistics for $l_{h,k}$ and $n_{h,t}$ computed earlier, we build up from

$$\eta_{i,h,j} \rightarrow \eta_{i,h} \rightarrow \eta_i$$

Finally to compute the statistics of $pq$-gram distance, we recall

$$d^{pq}(T_1, T_2) = |\mathcal{I}_1 \uplus \mathcal{I}_2| - 2.|\mathcal{I}_1 \boxplus \mathcal{I}_2|$$

$\eta_i^k$ denote the number of times $pq$-gram $i$ occurs in $T^k$. Then,

$$E[d^{pq}(T_1, T_2)] = \sum_i E[\eta_i^1] + \sum_i E[\eta_i^2] - 2\sum_i E[min(\eta_i^1, \eta_i^2)]$$

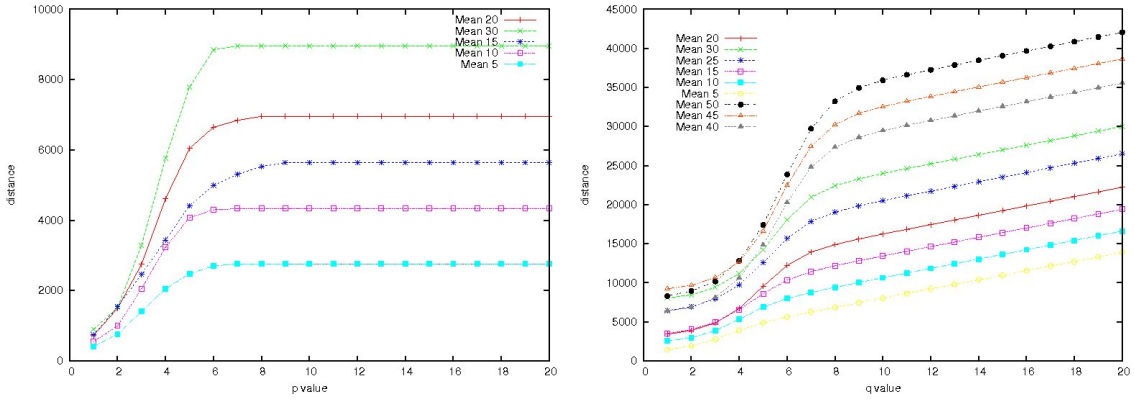
$$E[d^{pq}(T_1, T_2)] = \sum_i E[\eta_i^1] + \sum_i E[\eta_i^2] - 2\sum_i \sum_{\alpha=0}^{\infty} \alpha\left(P(\eta_i^1 = \alpha, \eta_i^2 \geq \alpha) + P(\eta_i^2 = \alpha, \eta_i^1 > \alpha)\right)$$

$$E[d^{pq}(T_1, T_2)] = \sum_i E[\eta_i^1] + \sum_i E[\eta_i^2] - 2\sum_i \sum_{\alpha=0}^{\infty} \alpha\left(P(\eta_i^1 = \alpha)\left(\sum_{\beta=\alpha}^{\infty} P(\eta_i^2 = \beta)\right) + P(\eta_i^2 = \alpha)\left(\sum_{\beta=\alpha+1}^{\infty} P(\eta_i^1 = \beta)\right)\right)$$

We need to compute this to find the expected *pq* distance.

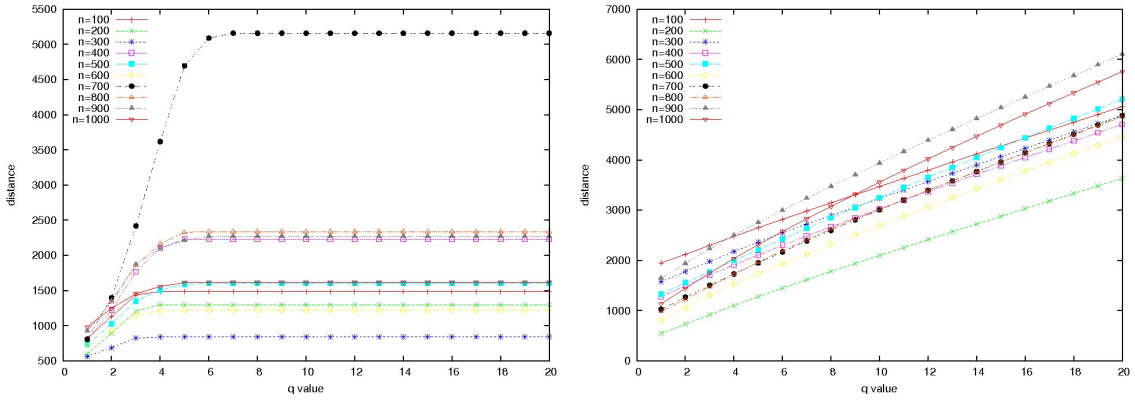
## 4.7 Experimental results

We studied the variation of *pq* distance with change in the mean number of children nodes generated in each step of the random tree growing process. Results show that the behaviour of *pq* distance converges with increase in both $p$ and $q$. For $p$ it becomes constant after a certain value, while it increases linearly for $q$ after a certain value. This shows that after a certain value pf $p$ and $q$, the effect of increasing $p$ and $q$ does not help in increasing the ability of the distancs to separate trees. As the *pq* grams are made more and more rigid, the number of non-trivial common *pq* grams decreases and the number of *pq* grams which largely consist of null($*$) nodes incraeses. By increasing $q$ beyond that point, the size of the database is inflated as *pq*-grams are longer. The number of *pq* grams increases linearly. This explains the linear asymptotic behaviour with change in $q$. Increasing $p$ does not change the number of *pq*-gram present in the tree. Hence the distance converges to a constant where the *pq* grams have been made so structurally rigid that the only *pq* grams common are the trivial ones.



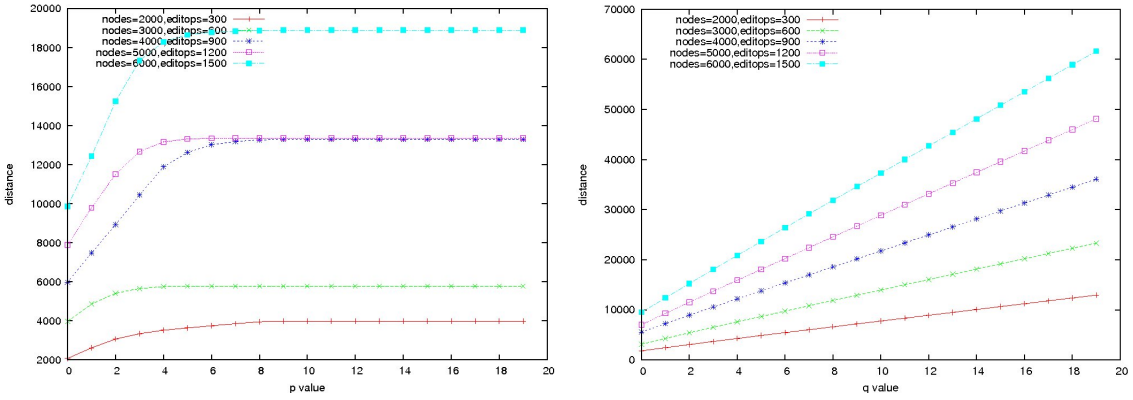

(a) $p$ varying with $q = 3$ (b) $q$ varying with $p = 3$

Figure 2: Plot of *pq*-gram distance between two random trees $T^{100}$ with change in $p$ and $q$




(a) $p$ varying with $q = 3$ (b) $q$ varying with $p = 3$

Figure 3: Plot of *pq*-gram distance between a random tree $T^{100}$ and and a tree obtained by performing $n$ random insert, relabel and delete operations with change in $p$ and $q$




(a) $p$ varying with $q = 3$ (b) $q$ varying with $p = 3$

Figure 4: Plot of $pq$-gram distance between a tree (generated using the XMLgen XML dataset generator) and and a tree obtained by performing $n$ random insert, relabel and delete operations on it with change in $p$ and $q$

# 5 Index structure of $pq$-gram profiles

We propose a reference-based index structure which uses the pseudo-metric property of $pq$-gram distance to prune out certain parts of the database. We use the Maximum Variance heuristic to select a set of trees from the database to act like references. The distance of all trees to each of these references is then computed. This is a one-time, preprocessing cost. To use the references, a search algorithm computes the distance between the query $q$ and all the references. Then for each tree $t_i$ in the database, a lower bound ($LB$) and an upper bound ($UB$) for $d^{p,q}(t_i, q)$ is calculated as

$$LB = max_{v_j \in V}(|d^{p,q}(q, v_j) - d^{p,q}(v_j, t_i)|)$$
$$UB = min_{v_j \in V}(d^{p,q}(q, v_j) + d^{p,q}(v_j, t_i))$$

For a range query $(q, r)$, $LB$ and $UB$ are used as follows

- if $r < LB$ $t_i$ is pruned
- if $r > UB$ $t_i$ is added to the result set
- if $LB \leq r \leq UB$ $t_i$ is added to the candidate set

The elements in the candidate set are then compared with $q$ using $pq$-gram distance. Those with distance less than $r$ are included in the result set.

## 5.1 Maximum Variance Heuristic

In this section we describe the maximum variance heuristic used to select trees for taking as references. Maximum Variance heuristic assumes that queries follow the same distribution as the database. It selects a reference set that represents the distribution in the database. Each new reference prunes some part of the database not pruned by the current trees in the reference set. The intuition is to select references that are *far away* from each other. The algorithm is as follows:

{Input:Database $S$, with $|S| = N$, number of references $m$, cutoff percentage $perc$ and a length $L$}
{Output: Set of references $V = \{v_1, v_2, \ldots, v_m\}$}
$V \leftarrow \{\}$
**for all** $s_i \in S$ **do**
  Select sample set of trees, $S' \subset S$.
  Compute $D_i = \{d^{p,q}(s_i, s_j) | \forall s_j \in S'\}$
  Compute mean $\mu_i$ and variance $\sigma_i$ of the distances in $D_i$.

**end for**
$w = L.perc$
Sort thr $N$ trees in descending order of theri variances
**while** $|V| < m$ **do**
$V \leftarrow V \cup s_1$
$S \leftarrow S - \{s_j\}, \forall s_j \in S$ with $d^{p,q}(s_1, s_j) < (\mu_1 - w)$ or $d^{p,q}(s_1, s_j) > (\mu_1 + w)$
**end while**
return $V$

## 5.2 Assignement of References

Each tree $s_i$ in the database is assigned a subset of the reference set $V$ as the pruning set to be used for pruning $s_i$ when servicing range queries. We have a large number of references but use only a subset of them to index each database tree. Formally, given a set of $m$ references ($m > k$), our goal is to assign a set of $k$ references to each database tree such that at least one of these $k$ references will remove $s_i$ from the candidate set for as many queries as possibles.

{Input:Database $S$, with $|S| = N$, Reference set $V$, $|V| = m$, Sample queries $Q, |Q| = q$ and References per tree $k$}
{Output: $E = \{E_1, E_2, \ldots, E_N\}$, $E_s$ is assigned to tree $s \in S$ }
$G[i] = 0, 1 \leq i \leq m$ {Total gain from each reference $v_i \in V$}
$E_i = \{\}, 1 \leq i \leq N$ {Initialize reference set of each sequence}
**for all** $s \in S$ **do**
**repeat**
$Vcount[i] = 0, 1 \leq i \leq m$. {Initialize gain for $[v_i, s]$ pair}
**for all** $[v, Q_j], \forall v \in V$ and $\forall Q_j \in Q$ **do**
**if** $PRUNE(s, Q_j, v)$ **then**
Vcount[v]++
**end if**
**end for**
Let $e = argmax_x(Vcount[x])$
$G[e] += Vcount[e]$
$V = V - \{e\}$
$E_s = E_s \cup \{e\}$
Remove from $Q$ queries for which $s$ is pruned with reference $e$.
**until** $|E_s| = k$
Re-insert all deleted entries from sets $V$ and $Q$.
**end for**
**for all** $v \in V$ **do**
**if** $G[v] \leq |Q|$ **then**
$V = V - \{v\}$
**end if**
**end for**
Update the reference sets $E_s, \forall s \in S$

# 6 Conclusions

The analysis shows that it is possible to find optimal values for $p$ and $q$ to be used for finding the $pq$-gram distance by noting the value after which the behaviour of $pq$-gram distance converges. The probabilistic analysis did not yield closed form solutions. This leaves scope for further improvement. However, the process of formulation of the problem rogorously and arriving at the skeletal form of the probabilistic analysis is quite insightful. The proposed index structure performs better than than the case where a list of $pq$-gram profiles is stored linearly.

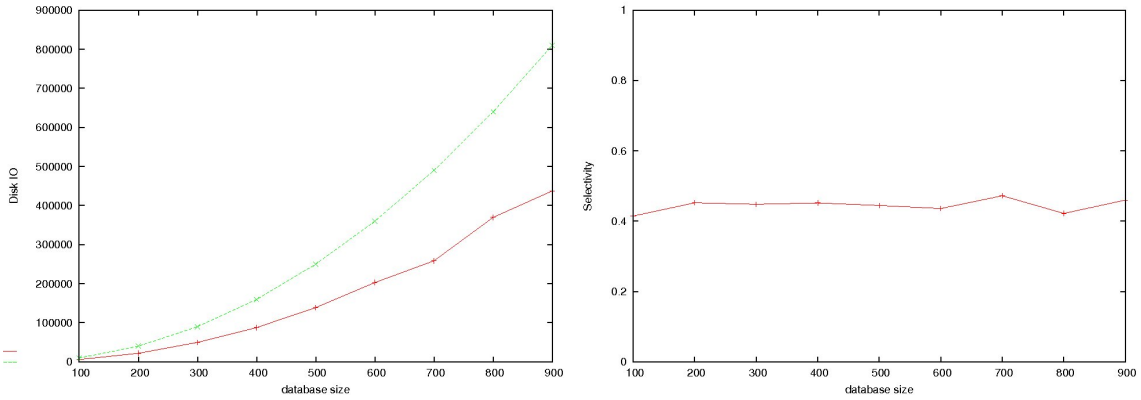



Figure 5: Comparison of reference-based indexing method with the naive linear search

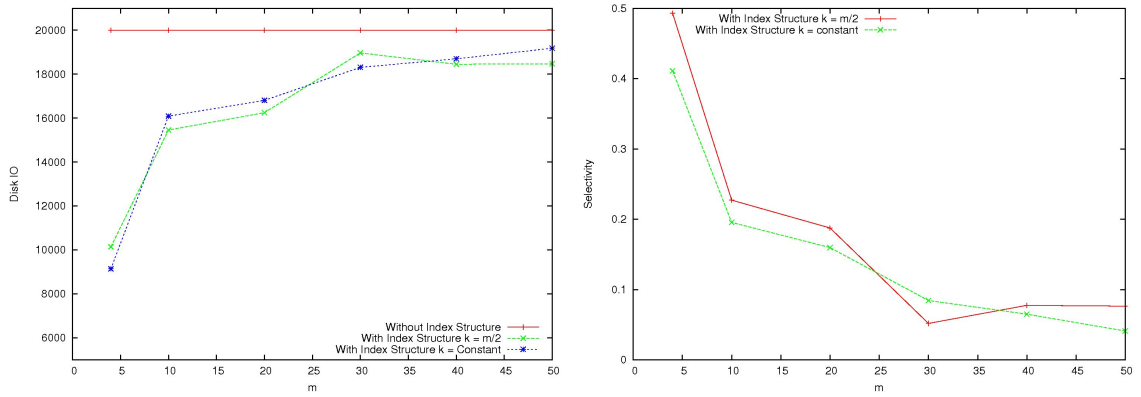



Figure 6: Change in selectivity and disk I/Os for processing a random range query with different size of reference set($m$) and per tree reference set($k$)

# 7 Acknowledgements

We would like to thank Dr Arnab Bhattacharya for guiding us at all junctures during the project.

# References

[1] Nikolaus Augsten, Michael Böhlen, and Johann Gamper. Approximate matching of hierarchical data using pq-grams. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 301–312. VLDB Endowment, 2005.

[2] Nikolaus Augsten, Michael Böhlen, and Johann Gamper. An incrementally maintainable index for approximate lookups in hierarchical data. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 247–258. VLDB Endowment, 2006.

[3] Minos Garofalakis and Amit Kumar. Xml stream processing using tree-edit distance embeddings. *ACM Trans. Database Syst.*, 30(1):279–332, 2005.