Self-Organization and Evolution on Large Computer Data Structures*

Joaquim Gabarró Llorenç Roselló
Universitat Politècnica de Catalunya
Departament de Llenguatges i Sistemes Informàtics
Campus Nord–Mòdul C6
C/ Jordi Girona Salgado, 1–3
08034 Barcelona, Spain
{gabarro, lrosello}@lsi.upc.es

Abstract

We study the long time evolution of a large data structure while inserting new items. It is implemented using a well known computer science approach based on 2-3 trees. We have seen self-organized critical behavior on this data structure. To tackle this problem we have introduced and studied experimentally three statistical magnitudes: the stress of a tree, the sequence of jump points and the distribution of subtrees inside a tree.

The stress measures the amount of free space inside the 2-3 tree. When the stress increases some part of the tree is restructured in a way close to an avalanche. Experimentally we obtain a potential law for stress distribution. When the tree does not have more free space in any internal node, needs to grow up. When this happens, the height of the whole tree increases by one and we have a jump point. Experimentally these points have good expected behavior. A 2-3 tree is composed from a great number of other 2-3 trees called their subtrees. We have studied experimentally the distribution of the different subtrees inside the tree.

Finally we analyze these results using simple theoretical models based on fringe analysis, Markov and branching processes. These models give us a quite good description of the long term process.

Keywords. *Self-Organization, sand-pile paradigm, 2-3 trees, fringe analysis, Markov process.*

*This work has been partially supported by DGICYT under grant PB95-0787 (project KOALA), also by ACI with Universidad de Chile DOGC 2320–30.1.1997, also by ESPRIT LTR Project no. 20244 — ALCOM-IT, also by CICIT TIC97-1475-CE, also by PR98-11 of the Universitat Politècnica de Catalunya.

1 Introduction

One of the most important task of computers consists on store large sets of data information. Therefore, a corner-stone in computer science consists on the organization and efficient manipulation of large quantities of data items.

To tackle this problem different data structures have been investigated. Between them dictionaries have an important place. In computer science a dictionary can find, insert or delete an information item. Dictionaries having good performances require special constraints. They are implemented with *balanced trees.* A *tree* is rooted a directed acyclic graph (look at the figure 1). Moreover a tree is *balanced* if approximately all the leaves are at the same distance from the root. There are different kinds of balanced trees (AVL, 2-3 trees, Red-Black,. . .).

Here we study efficient dictionaries as a self-organized critical structure [3]. Good tree structures require sophisticated update algorithms in order to maintain the performance. More precisely, while evolving the tree needs to maintain the "general shape" ever if the concrete details change continuously. Therefore along its evolution a balanced tree behaves as self-organized structure, having a critical behavior. This happens because update algorithms are based on the application of a sequence of local rules. A *local rule* modifies just few closely related nodes. Under these rules the tree behaves like a big cellular automata. Moreover, when some part of the tree becomes too stressed (contains too much information in a little space) it is splitted and give rise to something quite close to an "avalanche". These two facts (the local rules and the avalanches) seem to make balanced trees close to the "sand pile" paradigm [4].

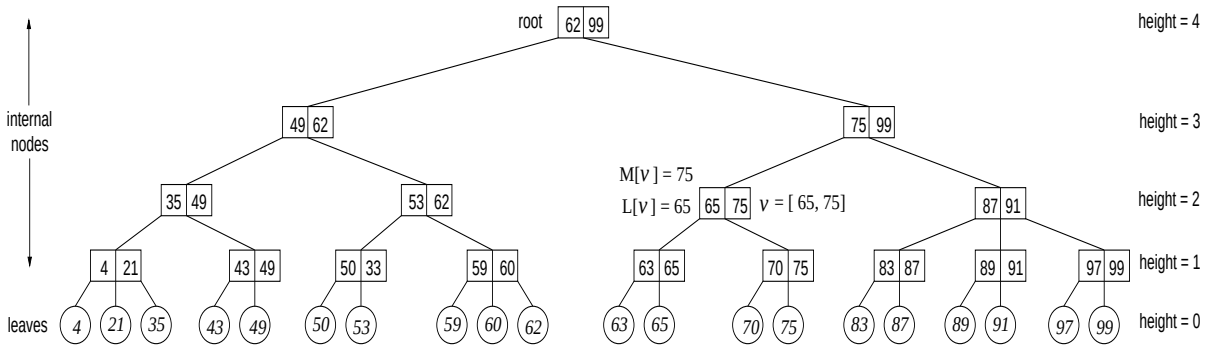


Figure 1. An example of a 2-3 Tree**.**

Here we give some precise limits to this idea.

The application of self-organized criticality ideas in computer science is not new. Basically it has been applied to other domains having a collectivity of items. Between them we can cite the work of W. Bauer and S. Pratt [5] word processing or the work of M. Takayasu, H. Takayasu and T. Sato [14] on information traffic. However the possibility to study large data sets with the methods of self-organized criticality seems to put these methods into the core of computer science.

We are going to study one of the most popular data structure: the 2-3 trees. These trees were introduced by J. Hopcroft in the seventies [1]. We will study experimentally the long time evolution of the 2-3 trees. We will see that 2-3 trees under this approach behaves as a self-organized critical structure. To attack this problem we have introduced and studied three magnitudes: the *stress of a tree*, the sequence of *jump points* and the *distribution of subtrees* inside a tree. Let us describe the content of the following sections.

The section 2 is devoted to the the 2-3 Trees. More precisely, we consider how this data structure evolves under the addition of new items of information.

In section 3 we introduce the *stress* of the 2-3 Trees. The *stress* gives us information about the free space inside the data structure. When the stress increases some internal part of the tree is restructured. This restructuring process is close to an avalanche. Experimentally we get a potential law.

In section 4 we study the *jump points.* When the tree does not have more free place needs to grow up. The sequence of jump points measure the moments where data structure is globally full and needs to allocate new space in order to continue the process. Experimentally jump points have precise expectations.

Inside a 2-3 Tree we have other 2-3 Trees called their subtrees. In section 5 we have studied experimentally the *distribution of the subtrees.*

In section 6 we recall one of the few theoretical instruments introduced in order to study balanced trees. This is the *fringe analysis* based on the evolution of the bottom of the tree. In this section we also model the 2-3 Trees as a branching process.

Finally, in section 7 we justify the preceding experimental results using fringe analysis, Markov and branching processes.

2 Sequences of random 2-3 Trees

Let us introduce the basic tree structure; the 2-3 Tree balanced data structure [1, 9]. The figure 1 gives us an example of such a tree. More accurately, a 2-3 Tree is rooted directed acyclic graph such as:

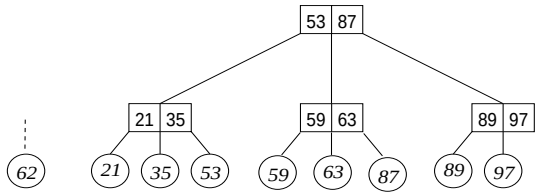
(1) There are two kind of nodes. The *internal nodes* having successors and the *leaves* with no successors. Every internal node n stores two values, the left key $L[n]$ and the middle key $M[n]$ such as $L[n] < M[n]$. We identify n with its stored values and we write $n = [L[n], M[[n]]$ (for instance root $= [62, 99]$). Every leaf contains just one key, we identify a leaf with its key.

(2) Every internal node has 2 or 3 sons. Node $[87, 91]$ has three sons: a left son $[83, 87]$, a middle son $[89, 91]$ and a right son $[97, 99]$. Node $[49, 62]$ has only two sons: a left son $[35, 49]$ and a right son $[53, 62]$.

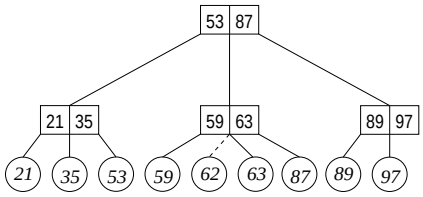
(3) The value of $L[n]$ is the largest leaf in the left subtree of n (the left subtree is the tree having as a root the left son of n). For instance $L[49, 62] = \max\{4, 21, 35, 43, 49\}$. The value of $M[n]$ is the largest leaf in the subtree having as a root the second son of n. For instance $L[49, 62] = \max\{50, 53, 59, 60, 62\}$ and $L[87, 91] = \max\{89, 91\}$.

(4) All the paths going from the root of the tree to a leaf have the same length.

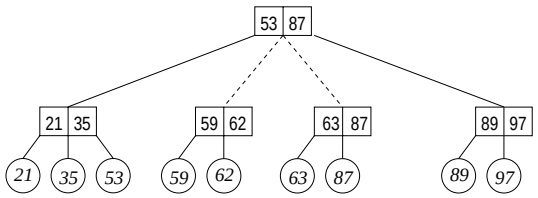
In a 2-3 Tree all the nodes have a well defined notion of height. All the leaves have height 0, for instance height$[70] = 0$. The parent of the leaf has



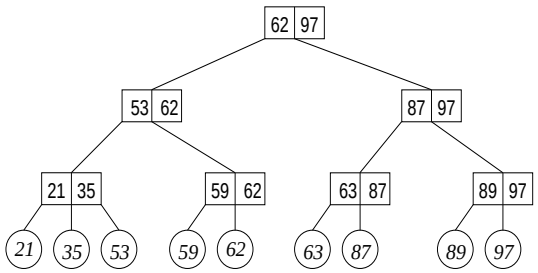
(a) The 2-3 Tree T_8 corresponding to the sequence 97,89, 87, 53, 35, 63, 59, 21. The item $\mathsf{K}_9 = 62$ has to be incorporated.



(b) The key 62 has been attached. The node [59,63] becomes unstable.



(c) The node [59,63] has been splitted into [59,62], [63,87]. The node [53,87] becomes unstable.



(d) The node [53,87] has been splitted. A new node has been generated. This is the 2-3 Tree T_9 corresponding to the sequence 97, 89, 87, 53, 35, 63, 59, 21, 62.

Figure 2. The figures (a),(b),(c) and (d) display the step corresponding to insertion of the key 62. In this step there are 2 splits and the height is increased by one.

height one, $\mathsf{height}[70, 75] = 1$ and successively. The *height of the three* is the height of the root, in this case $\mathsf{height}[\mathsf{root}] = 4$.

In a 2-3 Tree the set of leaves are sorted in increasing order. For instance, in the preceding example the ordered set of keys starts as $4, 21, 35, \ldots$.

We study the long term evolution of the 2-3 Trees. To attack this problem, we grow up a 2-3 Tree from a random sequence of keys written K_t, with t integer $t \geq 1$. In order to figure out the process we will consider the sequence K_t starting with the keys 97, 89, 87, 53, 35, 63, 59, 21, 62, 65, 75, 43, 49, 4, 99, 83, 60, 50, 70, 91. The process evolves constructing a sequence T_t of 2-3 Trees. The process behaves in discrete *steps*. In the step t, the tree T_{t-1} is updated adding the leaf K_t giving T_t. The figure 2 mimics the step 9 starting with T_8 (constructed from the subsequence 97, 89, 87, 53, 35, 63, 59, 21) when the new key $\mathsf{K}_9 = 62$ arrives (see figure 2(a)). In this step T_8 is rearranged in order to incorporate 62, at the end of the step we have T_9 (see figure 2(d)).

In the step t the tree T_t is built. To do this, T_{t-1} is updated with K_t. Some internal work is needed in T_{t-1} to do this. This work can be organized in two main phases (a precise description is given in [1]).

Attach phase. To insert K_t into T_{t-1}, first K_t is linked at the bottom of the tree to a node n having height 1. For instance, in the figure 2(a), $\mathsf{K}_9 = 62$ is attached at the node $n = [59, 63]$, see figure 2(b). The process K_t is linked together with the smaller leaf having a value greater than K_t. Moreover all the leaves have to remain sorted, therefore 62 is located at the left side of 63. The new keys will be always located at "the left". Now a problem appears because n has 4 sons violating the condition 2 of 2-3 Trees. To solve this situation a second phase starts.

Reconstruction phase. This process reconstructs the tree in a bottom up way. Any node with 4 sons becomes "unstable" and *splits* into two other nodes (each one with 2 sons). The split process raises up while nodes with 4 sons exists. The split process always *stops* when it arrives at a node with 2 sons (it will have 3 sons now). If the root becomes unstable, a new node is generated and the height of the tree increases by one. In the preceding case, $[59, 63]$ splits into nodes $[59, 62]$ and $[63, 87]$, see the figure 2(c). Now the problem appears with the

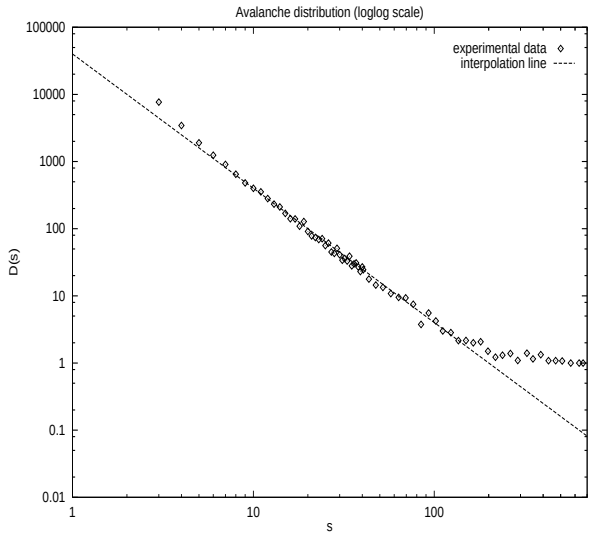


Figure 3. The distribution of stress follows a power law of exponent -2 during a long interval of data.

root node $[53, 87]$. This node also splits into $[53, 62]$ and $[87, 97]$ but to do this a new node $[62, 97]$ is created.

Following the preceding algorithm we get T_{20} after the insertion of $\mathsf{K}_{20} = 91$ (figure 1). Of course the final tree depends heavily on the *order* of the sequence. The same set of values arriving in another order will give a different final tree. However the basic parameters of the tree, like the height, seem to have a well expected behavior.

3 Stress in 2-3 Trees

Given a 2-3 Tree, we can see the insertion/accumulation of keys as something like a slowly increasing external stress field supported by the tree in all their nodes. When the stress in a given node exceeds the critical value (recall that a node accepts at most three sons) the stress is shared up to its ancestors (the node splits and the stress of the father increases). We define the *stress* S of a node as the number of keys to be inserted through this node in order to split it. Given a 2-3 Tree every internal node n fixes a subtree (it is also a 2-3 Tree) having n as a root. Therefore the stress of n is the number of keys accepted by this subtree before its root n splits. We can see the stress of a (sub)tree measures the "free space" or the "capacity" to accept new keys before the (sub)tree becomes crowded. Let us point several facts about the stree S.

(1) The stress of a node is a statistical property. It depends on the structure of its current subtree. It depends also on the sequence of keys being inserted. Perhaps some long sequences can be accepted but other short sequences (having other values) cannot be allocated inside the subtree.

(2) The stress of a node n depends on the position of n inside the tree. If n lies at the bottom of the tree, it has a really small stress because its subtree is small. On the other way, nodes near the root of the main tree should accept more keys before splitting.

(3) It is necessary to decide when to start counting the keys being inserted through a node n. We choose as the start time the moment when the node n is created. Recall that node n is created in one split. Just when the node is created we start to count the keys being inserted through it. We choose as the stop time the moment when node n is splitted. In this way the stress of a node n is the number of keys inserted thought it along its "life".

(4) Now we are going to precise what is the "life of a node n". During its life the number of sons changes from two to three, therefore the structure of n changes along its life. Let us explain this fact: when created n has just two sons (this is determined by the algorithm). During some time n will continue to have two sons. At some stage the current subtree determined by n becomes full and a new son is added. In this moment node n has three sons. During some time this node will continue having three sons. Finally, no more keys can be allocated inside the current tree and n splits. This is the end of n's life.

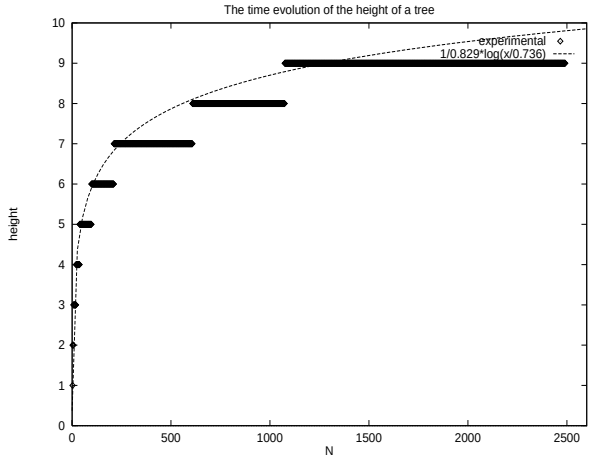
Computer simulations of the evolution of a 2-3 Trees allow us to obtain a distribution of S, it follows a power law with exponent -2, see figure 3.

$$D(S) \propto S^{-2}.$$

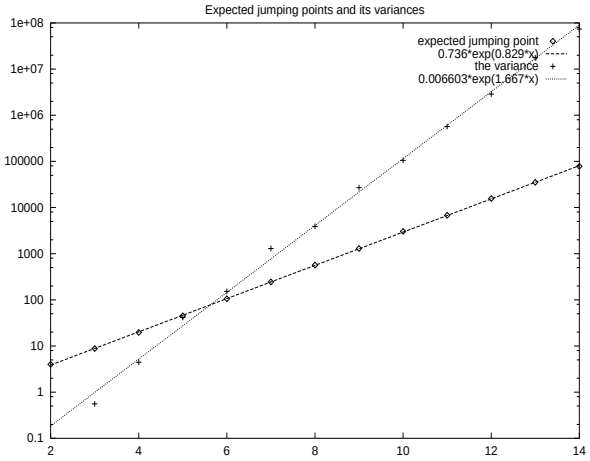
Later on we will derive a theoretical explanation of this fact based on very simple suppositions about when a split does happen.

Special care has to be taken with experimental results concerning large stress. Large stress corresponds to very big 2-3 Trees. For this kind of trees it becomes quite difficult to have faithful statistical estimations. In order to get good results for this kind of stress we will need to put these trees inside much greater trees, but this is quite cumbersome and we will still have the problem with bigger trees. Therefore, the experimental points at the end of plot are not on the line because the experiment gives us poor statistics of big trees. However we point out that we have a power law over ten decades and this is and important range of behavior.

From the preceding approach it has been seen that the split operation appears as an important one. In the



(a) Evolution of the heights corresponding to a long sequence.



(b) Expected values and variances

Figure 4. Long term evolution and statistics of the J_h.

following section we study splits along the time.

4 Jump points

As we have seen, the "avalanche" phenomena (and therefore the SOC behavior) in a 2-3 Tree is heavily related to the discontinuous process of the split of a node with two sons. The best way in order to study this splitting is considering the evolution of the heights in a 2-3 Tree, T_t, let us we write $\mathsf{H}_t = \mathsf{height}(\mathsf{T}_t)$. The evolution of H_t is far from being smooth. The sequence H_t qualitatively verifies:

(1) The height H_t remains constant for very long intervals of t. Therefore, the size of the intervals is a random variable.

(2) The sequence H_t increases one unit between long constant intervals. We call *jump points* J_h the points where the height increases in one unit.

(3) The length of the constant height intervals increases (from one to another) approximatively as an exponential.

We study experimentally H_t from two complementary points of view. First we analyze the sequence of jump points J_h estimating $E(\mathsf{J}_h)$ and $Var(\mathsf{J}_h)$.

The evolution of H_t does not have a smooth behavior. To study this fact we consider the sequence of points J_t where the height increases. Given a sequence K_t, the value J_h is the number of inserted keys when the tree jumps into height h. Formally $\mathsf{J}_1 = 1$ and for $h > 1$, we have $\mathsf{J}_h = t$ if $\mathsf{height}(\mathsf{T}_{t-1}) = h - 1$ but $\mathsf{height}(\mathsf{T}_t) = h$. The sequence J_h depends on the sample K_t, for instance when $\mathsf{K}_t = 97, 99, \ldots$ we have $\mathsf{J}_h = 1, 4, 9, 20, \ldots$. The figure 4(a) plots J_h for a larger sample, the continuous plot is the expected behavior (to be precise we have plotted the inverse function).

The process J_h has a good probabilistic behavior. To estimate it, we had done 50 different samples of K_t with 100 000 keys each one. The keys are taken from a uniform random variable in $[0, 1]$. The figure 4(b) give us the behavior of J_h. If we fit $\ln E(\mathsf{J}_h)$ versus h with a linear regression we obtain $\ln E(\mathsf{J}_h) = 0.829h - 0.306$. From this

$$E(\mathsf{J}_h) = 0.736e^{0.829h}, \tag{1}$$

By similar methods from figure 4(b) we get:

$$Var(\mathsf{J}_h) = 6.603 \times 10^{-3} e^{1.667h} \tag{2}$$

5 Subtree distribution and lognormal laws

The preceding section gave us information about the long term evolution of the sequence T_t. However we would like to have some more information about the internal structure of the trees appearing in T_t.

Consider a very large 2-3 Tree, this tree contains many different subtrees (any internal node determines a subtree having this node as a root). Some of these subtrees seems to appear more frequently than others. The distribution of the different "subtrees" seems to be a difficult topic. Therefore we will change "subtree" by "size of the subtree". By size we mean the number of leaves and we will study such a distribution.

To do this we need to design a procedure allowing us to locate for any new inserted key a subtree inside the

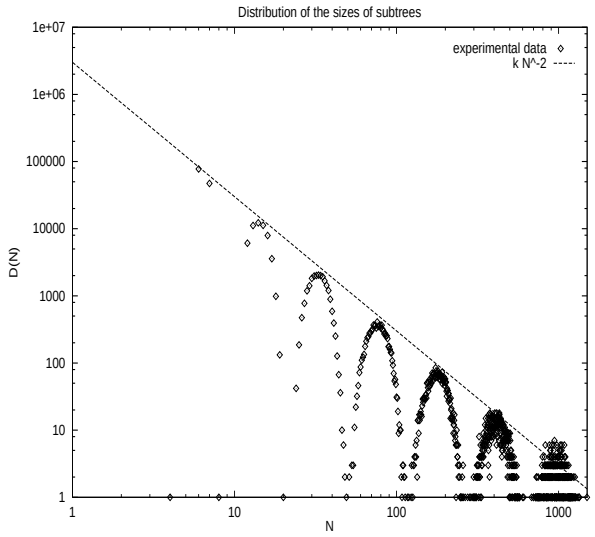


Figure 5. The maximums of this distribution are on a straight line with slope -2 in a loglog scale.

whole tree, consider the following situation: We start with the tree T_{t-1} built with the random sequence of keys until the step $t-1$ (assume t large). At step t a new key K_t is introduced. This key generates a sequence of splits until the perturbation stops updating a node so called n_t. This node is the root of a subtree T_{n_t}. For this tree T_{n_t} count the number N_{n_t} of leaves. If we perform this experiment for a large number of keys we can see that some values N_{n_t} appears more frequently than others. In this way we get experimentally the probability distribution $D(N)$ of the possible values of N. The figure 5 gives us such an experimental distribution.

This plot shows us that some subtrees are much more probable than others. For instance the first maximum appears when $N = 6$ and the first minimum is when $N = 8$. It means that subtrees with 6 leaves have high probability and subtrees with 8 leaves have a really small probability. The loglog plot given in the figure 5 seems to be a sequence of "mountains" having decreasing height. Intuitively each mountain seems to describe the distribution of 2-3 Tree for a fixed height. Evolving from one mountain to the next one mimics the process of increasing the height by one. Much more intriguing is the concrete form of the mountains.

Sequence of log-normal laws. Each mountain seems to be a parabola. The distribution of sizes given in the figure 5 seems to be a sequence of parabolas. For every parabola we have

$$\log D(N) = a \log^2 N + b \log N + c$$

Using the experimental data we get the coefficients a, b, c for the first 4 parabolas. The data describing the last two parabolas are too confusing and we avoid them.

Parabola	a	b	c
1	-333.33	9333.3	-53333
2	-9.5125	656.36	-9322.2
3	-0.3033	50.651	-1753.7
4	-0.0156	5.7314	-458.85

We have to guess a distribution function giving a parabola in a log-log plot. We are looking for a distribution appearing in processes described by many items interferring between them. Under these two hypothesis the log-normal law is a clear candidate. For instance, B. A Huberman and R. M. Lukose [10] use the log-normal in order to describe internet traffic. This law also appears to model the stock prices [11]. Let us develop this point of view. For a fixed parabola the log-normal law is

$$D(N) = \frac{1}{\sigma\sqrt{2\pi}N} e^{\frac{-(\ln N - \mu)^2}{2\sigma^2}}$$

for $N > 0$ and zero otherwise. Taking logarithms, $\ln D(N)$ is:

$$\frac{-1}{2\sigma^2}\ln^2 N + \left(\frac{\mu}{\sigma^2} - 1\right)\ln N - \left(\ln \sigma\sqrt{2\pi} + \frac{\mu^2}{2\sigma^2}\right).$$

We can express the expectation and variance in terms of a and b getting:

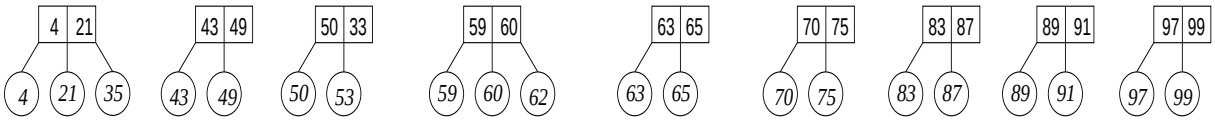
$$\sigma^2 = -\frac{1}{2a} \quad \text{and} \quad \mu = -\frac{1}{2a}(b+1)$$

Using the experimental data we get

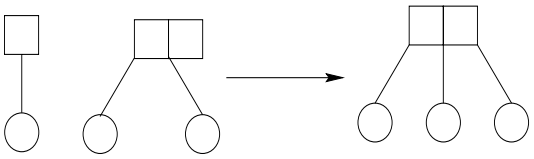
Parabola	σ^2	μ
1	0.0015	14.001
2	0.0526	34.552
3	1.6485	85.149
4	32.017	215.52

Let us consider briefly the sequence of the maxima and the minima described by the sequence of parabolas.

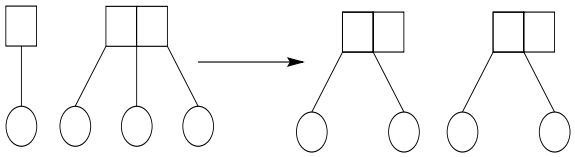
Sequence of minima. The subtrees having a number of leaves close to $8, 20, 49, 118, 250, \ldots$ have small probability. These numbers coincide quite well with the expected jumping points (see equation 1) at heights $h = 2, 3, 4, 5, 6, \ldots$ Therefore the minima reflects the moments when the trees do not have more room to store keys.



(a) The fringe corresponding to the preceding tree T_{20} build from the sequence $\mathsf{K}_t = \{97, 89, \ldots, 91\}$ containing 20 elements. An example of x node is $[43, 49]$, the leaves 43 and 49 are 1-type. An example of node y is $[4, 21]$, the leaves 4, 21 and 35 have 2-type. In this case $X_{20} = 14$ and $Y_{20} = 6$ note that $X_{20} + Y_{20} = 20$.



(b) Update case: The new key hits an x node. The node x is updated into a node y. The number of 1-type leaves decrease by 2 and the number of 2-type leaves increases by 3. When an update occurs at step $t + 1$ in a fringe X_t,Y_t we have $X_{t+1} = X_t - 2$ and $Y_{t+1} = Y_t + 3$.



(c) Split case: The new key hits and y node. The node y is splitted into 2 nodes x. The number of 1-type leaves increases by 4 and the number of 2-type leaves decreases by 3. When an split occurs at step $t + 1$ in a fringe X_t,Y_t we have $X_{t+1} = X_t + 4$ and $Y_{t+1} = Y_t - 3$.

Figure 6. The fringe analysis. In this model, the probability of an split is $p = \frac{3}{7}$. The probability to have an update is $q = \frac{4}{7}$.

Sequence of maxima. The subtrees with approximately $6, 14, 33, 76, 180 \ldots$ leaves have high probability. The distribution of maxima follows with a high accuracy the exponential law

$$N_{max}(h) = \frac{13}{5}\left(\frac{7}{3}\right)^h.$$

Moreover the distribution of the maxima is given by the potential law:

$$D(N_{max}) = N_{max}^{-2}$$

In the following two sections we introduce some theoretical developments to study the preceding experimental results.

6 Fringe analysis, Skip Tree and branching processes approximations

The evolution of the T_n is difficult to study. In order to get some approximate knowledgement A. Yao [15] introduced the fringe analysis. This approach has been fully developed by B. Eisenbarth et alters [2]. Let us recall the basics of the fringe analysis as is given in [2]. The fringe of a tree is composed by the subtrees on the bottom part of a tree. Our fringe is composed by trees of height one. A bottom node with two sons is called and x node, and a bottom node with three sons is called an y node. These nodes separate leaves into 1-type leaves if their parents are x nodes and 2-type leaves if their parents are y nodes. The figure 6(a) show us examples of x, y nodes and X_{20} and Y_{20} for T_{20}. Let us consider the evolution of the fringe along the time. Let X_t and Y_t be the random variables associated to the number of 1-type leaves and 2-type leaves respectively at the step t. We assume that $X_t + Y_t = t$ being t the number of keys of the tree. When a new key falls into a bottom node we have two possibilities.

Update case. The key can hit a bottom node x. In this case this node is *updated* into a node y. This case is schemitized in the figure 6(b). When this case happens at step $t + 1$ in a fringe X_t and Y_t we have:

$$\begin{aligned} X_{t+1} &= X_t - 2 \\ Y_{t+1} &= Y_t + 3 \end{aligned}$$

Split case. The new inserted key hits an y node. In case the node y is *splitted* into 2 nodes x. The case is given in the figure 6(c). When this case occurs at the step $t+1$ in a fringe X_t and Y_t we have:

$$\begin{aligned} X_{t+1} &= X_t + 4 \\ Y_{t+1} &= Y_t - 3 \end{aligned}$$

The probability that a key hits a bottom node x is $\frac{1}{t}X_t$ and for a node y is $\frac{1}{t}Y_t$. A split occurs when the key hits a node y, therefore let us study the evolution of Y_t. In the step $t+1$, the conditional expectations of Y_{t+1} verifies:

$$\begin{aligned} E(Y_{t+1} \mid X_t, Y_t) &= \frac{Y_t}{t}(Y_t - 3) + \frac{X_t}{t}(Y_t + 3) \\ &= Y_t + \frac{3}{t}X_t - \frac{3}{t}Y_t \\ &= Y_t - \frac{6}{t}Y_t + 3 \end{aligned}$$

As $E(Y_{t+1}) = E(E(Y_{t+1} \mid Y_t))$ we rewrite as

$$E(Y_{t+1}) = E(Y_t) - \frac{6}{t}E(Y_t) + 3$$

In the work of B. Eisenbarth et alters [2] this kind of recurrent equations are solved exactly. Here we apply an easy approximated method. We can approach $\frac{d}{dt}E(Y_t) = E(Y_{t+1}) - E(Y_t)$ and we get the following equation

$$\frac{d\,E(Y_t)}{d\,t} + \frac{6}{t}E(X_t) = 3$$

having as solution

$$E(Y_t) = \frac{3}{7}t + O(t^{-6})$$

When a new key is inserted at step t, we call $p(t)$ the probability to have and split. This probability coincides with the probability to hit a 2-type leave, that it:

$$p(t) = \frac{1}{t}E(Y_t) = \frac{3}{7} + O(t^{-7})$$

If the key hits a node x this node, it will give an update with probability

$$\begin{aligned} q(t) &= \frac{1}{t}E(X_t) = 1 - \frac{1}{t}E(Y_t) \\ &= \frac{4}{7} + O(t^{-7}) \end{aligned}$$

When t becomes large we have:

$$\begin{aligned} p &= \lim_{t\to\infty} p(t) = \frac{3}{7} \\ q &= \lim_{t\to\infty} q(t) = \frac{4}{7} \end{aligned}$$

Using the fringe analysis we have computed the *probability to have a split* node as $p = \frac{3}{7}$. The *probability to have an update* is $q = 1 - p = \frac{4}{7}$. We would like to emphasize that the p and q have been obtained only in the fringe approximation. It is far from being obvious that this result can be extended to all the internal nodes of a 2-3 Tree. Even worse we think that such a probabilities does not exist of a random 2-3 Tree. The fringe analysis appears to us as something like a "first approximation". Intuitively speaking, the fringe analysis has common points with the mean field approximations as it is developed by H.J. Jensen in [12].

Skip tree approximation. Tree data structure where a node has a constant probability to split or to update has been recently studied by X. Messeguer [13]. It is called the Skip Tree. In the following section we will take the probability $3/7$ in order to explain qualitatively the preceding experimental results. Therefore we will approach 2-3 Trees as Skip Trees having a constant $3/7$ split probability.

Branching processes approximation. Let us approximate a 2-3 Tree with a branching process [6]. To do this we adopt a top down view of the tree. The construction of the tree starts from the root and branches down down to the leaves. We assume that any internal node has a probability p_{three} to branch into 3 sons and a probability $p_{two} = 1 - p_{three}$ to branch into 2 sons.

Let us explain how to build a 2-3 Tree having height h using a branching process. The construction starts at the root (having height h) and has h steps. At the first step of the construction, root branches, giving 3 sons with probability p_{three} and 2 sons with probability p_{two}. At the next step, all the internal node (having height $h-1$) branch again. The construction stops after h steps. In this way we obtain a tree having the "shape" of a 2-3 Tree. Let N_t the number of leaves obtained in such a construction.

To develop this approximation, we need to estimate p_{three} and p_{two}. We can do it using fringe analysis. Along the time the probability $p_{three}(t)$ can be computed as:

$$\begin{aligned} p_{three}(t) &= \frac{\frac{1}{3}E(Y_t)}{\frac{1}{2}E(X_t) + \frac{1}{3}E(Y_t)} \\ &= \frac{\frac{1}{3}p(t)}{\frac{1}{2}q(t) + \frac{1}{3}p(t)} \end{aligned}$$

because $E(X_t)$ and $E(Y_t)$ are the expected number of leaves of 1-type and 2-type and we have to divide them by 2 and 3 if we want to know the expected number of nodes of type x and y respectively. When the time in-

creases:

$$p_{three} = \lim_{t\to\infty} p_{three}(t) = \frac{1}{3}$$
$$p_{two} = \lim_{t\to\infty} p_{two}(t) = \frac{2}{3}$$

Let us study the statistics of N_h. The value $E(\mathsf{N}_h)$ will approximate the expected number of leaves in a 2-3 Tree of height h. When $h = 1$

$$E(\mathsf{N}_1) = 2 \cdot \frac{2}{3} + 3 \cdot \frac{1}{3} = \frac{7}{3}$$

otherwise [6]:

$$E(\mathsf{N}_h) = E^h(\mathsf{N}_1) = \left(\frac{7}{3}\right)^h$$

In the preceding section we have found experimentally the sequence of maxims $N_{max}(h)$. Note that $E(\mathsf{N}_h) \approx N_{max}(h)$, therefore the branching process fits with some accuracy the experimental results.

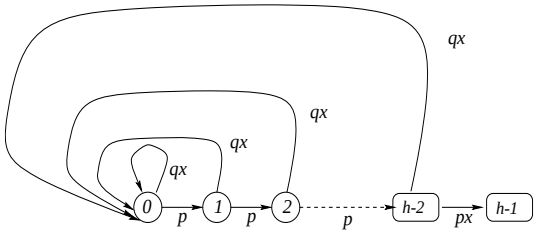


Figure 7. The Markov chain of the process. A split has probability p while an update has probability $q = 1 - p$. The variable x is used in order to compute the generating function.

7 A model for stress distribution, jump points and maxima distribution

Let us study the distribution of stress S in a 2-3 Tree. First of all we parameterize the stress in terms of the height of the tree. The stress at height h written S_h can be defined as the number of keys needed to jump from a tree having height h into a tree having height $h + 1$, that means:

$$\mathsf{S}_h = \mathsf{J}_{h+1} - \mathsf{J}_h$$

For a fixed h, if the value of $Var(\mathsf{S}_h)$ is not too important, we can assume:

$$\mathsf{S}_h \approx E(\mathsf{S}_h) = E(\mathsf{J}_{h+1}) - E(\mathsf{J}_h)$$

In this case the stress is just a function of the random variable h written $S(h)$. The probability P as a function of S is:

$$P(S) = \frac{1}{|d\,S/d\,h|} P(h).$$

Therefore we need to compute (or at least have some estimate) $P(h)$ and $S(h)$. We do it under the Skip Tree approximation induced by the fringe analysis.

First, let us study $P(h)$. A new inserted key will produce a tree having height h if this key generates a sequence of $h - 1$ splits followed by one update. Therefore, the probability to get a tree with height h when just one key is inserted is:

$$P(h) = p^{h-1}q = \frac{q}{p} e^{(\ln p)h} \tag{3}$$

Now let us deduce $E(\mathsf{J}_h)$ and $Var(\mathsf{J}_h)$. The number of keys needed to produce a tree with height h can be described as a Markov chain 7. Recall that, when a key is inserted a perturbation raises up through a sequence of splits until one update stops it. If a sequence of $h - 1$ splits occurs, the tree gets height h and the process stops, otherwise a new key is inserted and the process starts again:

- The chain has h states numbered from 0 to $h - 1$. State $0 \le i \le h - 1$ models "the number of consecutive splits is i".
- When the perturbation raises up to the node i it has two possible evolutions. It can follow up to the state $i+1$ with probability p doing one split. Otherwise one update stops it with probability $1-p = q$. In this second case the process needs to start again, therefore there is a transition from the state i to the state 0.
- Any loop describes the insertion of just one key performed by the tree without changing the height.
- The straight path ending in the state $h-1$ describes the insertion of the key augmenting the height in one unit.

The expected number of keys [8] needed to reach the height h can be computed using the generating function $g(x)$ of the chain. Let us briefly explain the structure $g(x)$. Any insertion of a key ending with a failure "counts" just one unit, therefore there is only one variable x for any loop. For instance the loop representing 3 splits and one update contributes with p^3qx to the generating function. The sequence of $h - 1$ splits contributes with $p^{h-1}x$. Unfolding the loops we get:

$$\begin{aligned} g(x) &= qx + pqx + \cdots h^{h-2}qx + p^{h-1}x \\ &= \frac{p^{h-1}x}{1 - (1 - p^{h-1})x} \end{aligned}$$

The expected number of keys needed to get a tree of height h is given by

$$g'(1) = \frac{1}{p^{h-1}}$$

and its variance is

$$g'(1) + g''(1) - g'(1)^2 = \frac{1-p^{h-1}}{p^{2h-1}}.$$

As the number of keys needed to jump into height h is precisely J_h we have

$$E(\mathsf{J}_h) = g'(1) = \frac{1}{p^{h-1}} = pe^{(\ln 1/p)h}$$

and

$$\begin{aligned} Var(\mathsf{J}_h) &= g'(1) + g''(1) - g'(1)^2 \\ &\approx \frac{1}{p^{2(h-1)}} = p^2 e^{2(\ln 1/p)h} \end{aligned}$$

Finally we can study the stress. The stress corresponding to a given height is given by

$$S(h) = E(\mathsf{J}_{h+1}) - E(\mathsf{J}_h) = \frac{q}{p^h} = qe^{(\ln 1/p)h}.$$

and finally we get:

$$P(S) = \frac{1}{|d\,S/d\,h|} P(h) = -\frac{1}{\ln p}\, S^{-2}.$$

Therefore $D(S) \propto S^{-2}$. Note that the exponent -2 of the distribution does not depend on the concrete values of p and q.

As we have seen, it is possible to estimate the probability p of having a split with the fringe analysis [7] and the experiments performed by [2]. Using the value $p = 3/7$ we obtain

$$\begin{aligned} E(\mathsf{J}_h) &= \left(\frac{7}{3}\right)^{h-1} = \frac{3}{7} e^{(\ln 7/3)h} \\ &= 0.429 e^{0.847h} \end{aligned}$$

and

$$\begin{aligned} Var(\mathsf{J}_h) &= \left(\frac{7}{3}\right)^{2(h-1)} = \left(\frac{3}{7}\right)^2 e^{2(\ln 7/3)h} \\ &= 0.184 e^{1.694h} \end{aligned}$$

The coincidence with the experimental data is quite good in the exponents but it is poor in the coefficient.

Finally let us give an explanation of the power law $D(N_{max}) = N_{max}^{-2}$ describing the experimental results about the maxima of the sequences of lognormal distributions. Let us assume $N_{max}(h) = E^h(\mathsf{N}_1)$, recalling that $E(\mathsf{N}_1) = \frac{7}{3}$ and $p = \frac{3}{7}$ we get:

$$\begin{aligned} P(N_{max}) &= \frac{1}{|d\,N_{max}/d\,h|} P(h) \\ &= \frac{q}{p} \frac{1}{|\ln E(\mathsf{N}_1)|} N_{max}^{\frac{\ln p}{\ln E(\mathsf{N}_1)} - 1} \\ &= \frac{q}{p} \frac{1}{|\ln E(\mathsf{N}_1)|} N_{max}^{-2} \end{aligned}$$

therefore $D(N_{max}) \propto N_{max}^{-2}$.

8 Conclusions

We have found a *real system*, in the sense that it lives in *real* computers, used to manipulate *real* data having a self-organized critical behavior. The 2-3 Tree are efficient dictionaries an we suspect that this efficiency is related with the SOC behaviour. Moreover we guess that his kind of phenomena will happen in all other self-balanced data structures like AVL o Red-Black trees.

Moreover we have developed theoretical methods based on fringe analysis, Skip Trees and branching processes. These methods are close (at least in spirit) to the mean field approximation theory used in self-organized phenomena. In spite of their apparent simplicity these methods help us understand some basic facts. For instance, they give us quite an accurate explanation of the potential laws.

Today the process of data by computers seems to take enormous proportions. If this fact continues to grow up computer science certainly will need to adapt the methods of physical sciences, specially those developed to study the condensed matter.

Acknowledgments We acknowledge P. Bak, A. Corral, J. Delgado, A Escalada and X. Messeguer for many helpful discussions. Part of this work was done by L. Roselló in the Niels Bohr Institute.

References

[1] A. Aho, J. Hopcroft, and J. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley, 1974.

[2] R.A. Baeza-Yates and P.V. Poblete. Higher-order analysis of 2–3 trees. *International Journal of Foundations of Computer Science*, 6(1):1–10, 1995.

[3] P. Bak. *How nature works*. Copernicus, 1996.

[4] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality. *Phys. Rev. A*, (38):364, 373, 1988.

[5] W. Bauer and S. Pratt. Word processors with line wrap: Cascading, sef-organization criticality, random walks, difussion, and predictability. *Phys. Rev. A*, 54(54):R1009–R1012, 1996.

[6] A.T. Bharucha-Reid. *Elements of the theory of Markov processes and their applications*. McGraw-Hill, 1960. There is a Dover reedition.

[7] B. Eisenbarth, N. Ziviani, G. Gonnet, K. Mehlhorn, and D. Wood. The theory of fringe analysis and its application to 2–3 trees and B-trees. *Information and Control*, 55(1-3):125–174, 1982.

[8] W. Feller. *An introduction to probability theory and its applications*, volume I. John Wiley, third edition, 1969.

[9] G.H. Gonnet and R. Baeza-Yates. *Handbook of Algorithms and Data Structures*. Addison-Wesley, 1991.

[10] B.A. Huberman and R. Lukose. Social dilemmas and Internet congestion. *Science*, 277:535, 477, 1997.

[11] J. Hull. *Options, futures, and other derivatives*. Prentice Hall, 1997.

[12] H.J. Jensen. *Self-Orzanized Criticality*. Cambridge, Lecture Notes in Phisics, 10, 1991.

[13] X. Messeguer. Skip trees, an alternative data structure to skip lists in a concurrent approach. *Informàtique Thèorique et applications*, 31(3):251–269, 1997.

[14] M. Takayasu, H. Takayasu, and T. Sato. Critical behaviors and $\frac{1}{f}$ noise information traffic. *Physica A*, 233:824–834, 1996.

[15] A. Yao. On random 2-3 trees. *Acta Informatica*, (9):159, 170, 1978.