B-trees with Inserts and Deletes: Why Free-at-empty is Better Than Merge-at-half

Theodore Johnson, johnson@csd4.cs.nyu.edu
Dennis Shasha, shasha@cs.nyu.edu
Courant Institute of Mathematical Sciences, New York University*

September 2, 1992

Abstract

The space utilization of B-tree nodes determines the number of levels in the B-tree and hence its performance. Until now, the only analytical aid to the determination of a B-tree's utilization has been the analysis by Yao and related work. Yao showed that the utilization of B-tree nodes under pure inserts is 69%. We derive analytically and verify by simulation the utilization of B-tree nodes constructed from a mixture of insert and delete operations. Assuming that nodes only merge (i.e. are freed) when they are empty we show that the utilization is 39% when the number of inserts is the same as the number of deletes. However, if there are just 5% more inserts than deletes, then the utilization is over 62%. We also calculate the probability of splitting and merging. We derive a simple rule-of-thumb that accurately calculates the probability of splitting. We also model B-trees that merge half-empty nodes. The utilization of merge-at-half B-trees is slightly larger than the utilization of free-at-empty B-trees, but the restructuring rate is much higher. For most purposes, this implies that free-at-empty B-trees are a better implementation choice than merge-at-half B-trees. We present two models for computing B-tree utilization, the more accurate of which remembers items inserted and then deleted in a node.

1 Introduction

B-trees are commonly used in very large database systems to provide indices into the database. The space that the B-tree index consumes can be considerable. In order to allocate an appropriate amount of space for the B-tree, an estimate of the B-tree space utilization (expected percentage of a node used to store data) is needed. In addition to the utilization, we calculate the probability of splitting or merging a leaf-level node.

In this paper, we derive and solve equations that describe the equilibrium structure of a B-tree under a parameterized mix of insert and delete operations. A modify operation is modeled as an insert-delete pair. We assume that every item in the B-tree is equally likely to be deleted on a delete operation, and that every permutation of the operands of the insert operations is equally likely. The method is based on Yao's [19] except that we consider the solution to be the equilibrium point of a set of difference equations. We show how the probability that a node receives an insert can be calculated by keeping track of the number of *ghosts*, or items that have been deleted, that are in the B-tree.

We model two types of B-trees in this paper. The first type allows B-tree nodes to become arbitrarily small (i.e., contain as few as one entry). When a node becomes empty, the space is freed and the delete is propagated upwards. We call this type of B-tree a *free-at-empty* B-tree.

*This work was partially supported by the National Science Foundation under grants number DCR8501611 and IRI-8901699, and by the Office of Naval Research under grant number N00014-85-K-0046.

Free-at-empty B-trees are often used in database management systems [14, 13]. The second type of B-tree that we model merges a half-empty node with its neighbor on a delete (a *merge-at-half* B-tree). Bayer and McCreight first proposed this merging strategy [3]. We compare the two types of B-trees on the basis of their space utilization and restructuring rates and conclude that database management systems have made the right decision.

1.1 Previous Work

Yas derives an estimate of 69% utilization for pure insert operations in [19]. Mizoguchi [15] independently derives the same result by nearly the same methods. Yao's method of analysis is generalized to *fringe* analysis by Eisenbarth et. al. [6]. Fringe analysis is the analysis of the leaves of a tree data structure. Eisenbarth et.al. [6] show how to solve the matrix recurrence equations that result from a pure-insert model. B-tree variants are analyzed using fringe analysis in [2] and [1]. Another B-tree variant that has high space utilization is analyzed in [9]. An alternative approach to estimating the utilization of a B-tree appears in [11] and is improved on in [7]. Yao's method is elaborated upon to obtain the probability of splitting in [18]. The number of children of the root of a B-tree is approximated in [5].

None of the above papers addresses the problem of deletes in the instruction mix. Mizoguchi [12] proposes an approximate model for free-at-empty B-trees in order to analyze utilization. The range of his analysis is from pure inserts to 33% deletes/67% inserts. He also predicts the utilization at 50% deletes/50% inserts (pure modifies), but his solution is pessimistic. Our linear model is similar to Mizoguchi's. (We show that the linear model is not a good approximation as the percentage of deletes approaches 50%.)

Quitzow and Klopprogge [16] propose a differential equation model to predict the utilization for both free-at-empty and merge-at-half B trees. They analyze the case of pure inserts (where they are consistent with Yao) and the case of pure modifies. Our linear model is similar to theirs.

Wright and Langenhop [10] also have a model for merge-at-half B trees with pure modify operations. Their predictions are different from those in [16] (and ours). It turns out that simulation results fall roughly in between.

Hsu and Zhang [20] analyze restructuring rates (i.e., rates of splits and merges) as a function of the merging point using a simple mathematical model. Our qualitative conclusions are similar to theirs, but our quantitative predictions are significantly different.

So, the main contributions of the present work are an analysis of the utilization as the number of deletes approaches the number of inserts (indicating the sharp knee in the utilization curve), a set of predictions of restructuring rates as characterized by a simple rule of thumb, a description of the limits of the linear model (as compared with the ghost model), and a comparison of the analytical models with a simulation of an actual B tree.

Simulation Model We wrote a B-tree simulator to compare against our analytical results. The simulation builds a B-tree out of a sequence of inserts and deletes, then applies a long sequence of parameterized inserts and deletes. The simulation ran until the space utilization of the leaves was observed to reach an steady state value. Every item in the B-tree is equally likely to be chosen as the operand of a delete operation. The operands of the insert operations are chosen uniformly randomly from a large key-space.

2 Merge-at-Empty B-trees

2.1 Pure Inserts

In this section, we review Yao's analysis for pure inserts, because our analysis is a generalization of Yao's. We begin with the terminology and methodology.

A *B-tree* is a balanced tree in which the distance between the root and any leaf is the same [3]. A B-tree is of *parameter* p if each non-leaf node has between 1 and $2p-1$ children. The children of an interior node are accounted for by *entries* in a node; there is one entry per child. The entries contain key and pointer information. The leaf nodes contain the *items* in the B-tree, where an item consists of a key and a pointer to the associated record (This type of B-tree was first proposed by Wedekind [17] and is often called a B^+-tree). A B-tree node is of *order* k if it has k entries or items.

This analysis will count the entries in the interior nodes because analyses of interior nodes will be primarily concerned with their fanout. The analysis will count the items in the leaf nodes because we assume that the keys are stored in the leaves. This approach is in contrast to previous analyses that counted the number of keys, which was assumed to be one less than the number of pointers. We feel that our approach is more in keeping with current B-tree implementations, and simplifies the presentation of the analysis and results.

Define:

N: number of inserts (assumed successful).

$X_i(t[N])$: number of order i leaf-level nodes in a particular B-tree, $t[N]$, of size N (i.e., resulting from N inserts).

$A_i(N)$: expected number of order i leaf-level nodes in a random B-tree of size N.

$a_i(N)$: $= A_i(N)/N$.

U: expected space utilization of a random B-tree

P_s: probability of splitting on an insert

Let $t[N]$ be a particular random B-tree of parameter p with N items. Then $X_i(t[N])$ is the number of leaf-level nodes in $t[N]$ of order i. Let $\Pr[t[N]]$ be the probability that a random B-tree of parameter p with N items is $t[N]$. Since $A_i(N)$ is the expected number of leaf-level nodes of order i in a tree with N items,

$$A_i(N) = \sum_{t[N]} X_i(t[N]) \Pr[t[N]]$$

A B-tree of parameter p with N items is of *class* $(X_1, X_2, \ldots, X_{2p-1}|N)$ if it has X_1 nodes of order one on the leaf level, X_2 nodes of order two on the leaf level, etc. Let $\Pr[(X_1, \ldots, X_{2p-1}|N)]$ be the probability that a random B-tree of parameter p with N items is of class $(X_1, \ldots, X_{2p-1}|N)$. Another way to calculate A_i is:

$$A_i(N) = \sum_{(X_1,\ldots,X_i,\ldots,X_{2p-1}|N)} X_i \Pr[(X_1, \ldots, X_i, \ldots, X_{2p-1}|N)]$$

Suppose that we have a B-tree of class $(X_p, \ldots, X_{2p-1}|N)$[1] and an insertion occurs. If the insertion goes to an order i node, $p \leq i \leq 2p-1$, then the result is a B-tree of class $(X_p, \ldots, X_i - 1, X_{i+1}+1, \ldots, X_{2p-1}|N+1)$. If the insertion goes to an order $2p-1$ leaf node, then the result is a B-tree of class $(X_p+2, \ldots, X_{2p-1}-1|N+1)$, because splitting an order $2p-1$ node results in two order p nodes. Next, in order to specify completely the way that a B-tree evolves through inserts, we must specify the probability that an insert is directed to an order i node.

To do that, specify an input probability distribution. We will use a uniform distribution in the following sense: Any of the $N!$ permutations of an N item insert sequence are equally likely. If N items have already been inserted, then there are $N+1$ positions to which the next insert can be directed ($N-1$ intervals and 2 end positions), each of which is equally likely. The probability that a given node of order i receives an insert is thus $i/(N+1)$. In addition if the node is, say, the rightmost

[1] $X_1 = X_2 = \cdots = X_{p-1} = 0$

one, then the node has an additional probability of $1/(N+1)$ of receiving the insert. Therefore, the probability that an insert is directed to an order i node in a class $(X_p, \ldots, X_i, \ldots, X_{2p-1}|N)$ B-tree is

$$\begin{aligned}\frac{iX_i}{N+1} + \frac{1}{N+1}\,&\text{Pr[the rightmost item is in an order i node]}\\ &= \frac{iX_i}{N+1} + \frac{1}{N+1}\frac{iX_i}{N}\\ &= \frac{iX_i}{N}\end{aligned}$$

Using the above probabilities, we can develop a system of recurrence equations for the $A_i(N)$

$$\begin{aligned} A_p(N+1) = \quad & \textstyle\sum_{(X_p,\ldots,X_{2p-1}|N)} \Pr[(X_p,\ldots,X_{2p-1}|N)] \left\{ \frac{pX_p}{N}(X_p-1) + \right. \\ & \left. \tfrac{(2p-1)X_{2p-1}}{N}(X_p+2) + \tfrac{N-pX_p-(2p-1)X_{2p-1}}{N} X_p \right\} \\ A_i(N+1) = \quad & \textstyle\sum_{(X_p,\ldots,X_{2p-1}|N)} \Pr[(X_p,\ldots,X_{2p-1}|N)] \left\{ \frac{iX_i}{N}(X_i-1) + \right. \\ & \left. \tfrac{(i-1)X_{i-1}}{N}(X_i+1) + \tfrac{N-iX_i-(i-1)X_{i-1}}{N} X_i \right\} \qquad i \neq p \end{aligned}$$

These reduce to

$$\begin{aligned} A_p(N+1) = \quad & A_p(N)(1-p/N) + A_{2p-1}(N)\tfrac{2(2p-1)}{N} \\ A_i(N+1) = \quad & A_i(N)(1-i/N) + A_{i-1}(N)\tfrac{i-1}{N} \qquad i \neq p \end{aligned}$$

Making the substitution $a_i(N) = A_i(N)/N$

$$\begin{aligned} (N+1)a_p(N+1) = \quad & a_p(N)(N-p) + 2(2p-1)a_{2p-1}(N) \\ (N+1)a_i(N+1) = \quad & a_i(N)(N-i) + (i-1)a_{i-1}(N) \qquad i \neq p \end{aligned}$$

The $a_i(N)$ describe the proportion of nodes on the leaf level that are of order i. Up to this point, we have been following Yao's treatment of the pure-insert problem. Next, however, we will deviate from Yao's methods and transform the equations into a set involving terms of the form $\Delta a_i(N+1) = a_i(N+1) - a_i(N)$:

$$\begin{aligned} (N+1)\Delta a_p(N+1) = \quad & -a_p(N)(p+1) + 2(2p-1)a_{2p-1}(N) \\ (N+1)\Delta a_i(N+1) = \quad & -a_i(N)(i+1) + (i-1)a_{i-1}(N) \qquad i \neq p \end{aligned}$$

We want to solve for the equilibrium point of this set of simultaneous difference equations – that is, the point where $\Delta a_i(N+1) = 0$. At the equilibrium point, the $a_i(N)$ will not change with N, so we can remove the dependence on N to obtain the set of equations:

$$\begin{aligned} 0 = \quad & -(p+1)a_p + 2(2p-1)a_{2p-1} \\ 0 = \quad & -(i+1)a_i + (i-1)a_{i-1} \qquad i \neq p \end{aligned}$$

In order to prove the existence and uniqueness of the equilibrium point, make the substitution $P_i = ia_i$. Then we have the equations

$$\begin{aligned} 0 = \quad & -(p+1)P_p + 2pP_{2p-1} \\ 0 = \quad & -(i+1)P_i + iP_{i-1} \qquad i \neq p \end{aligned}$$

The above equations sum to 0, so it is easy to see that the above p equations are of rank $p-1$. Therefore, the matrix of coefficients is singular, so that the set of equations has a non-trivial solution. That the matrix is of rank $p-1$ also means that it has only one eigenvalue of 0, so the equilibrium point is unique. Since the equilibrium point of the P_i exists and is unique, the equilibrium point of the a_i exists and is unique also. The solution that we are looking for is subject to

$$\sum_{i=p}^{2p-1} i a_i = 1.$$

The stability of the solution follows from the fact that non-equilibrium states will tend towards equilibrium states, as can be seen by considering the system to be a system of flows. Think of each a_i as being a cell. The flow out of a cell is directly proportional to the value of the cell, the flow in is directly proportional to the value its neighbors. Suppose that a cell has less than its equilibrium value. Because $\sum_{i=p}^{2p-1} i a_i = 1$, some other cells must have more than their equilibrium value. Therefore the flow out of the cell will be less than at equilibrium, and the flow in will be larger than at equilibrium. The result is a net flow in, and the value of the cell will grow. The converse holds if a cell has greater than its equilibrium value. This argument generalizes to a sequence of cells with less or more than their equilibrium values.

We have a system of simultaneous linear equations that we can solve numerically. However, this system is simple enough that we can give an algebraic solution.

Theorem 1 (Yao)

$$a_i = \frac{1}{i(i+1)}(H(2p) - H(p))^{-1} \quad p \leq i \leq 2p-1$$

where $H(p)$ is the harmonic function $H(p) = \sum_{i=1}^{p} 1/i \approx \ln p$.

The following lemma has appeared in [18]:

Lemma 1 *The probability of inserting at an order i node on the leaf level when all of the operations on the B-tree are inserts is ia_i.*

Corollary 1 *The probability of inserting at a full node on the leaf level when all of the operations on the B-tree are inserts is*

$$P_s = (2p-1)a_{2p-1}$$

The value $(2p-1)a_{2p-1}$ is approximately $1/(2p \ln 2)$ when p is large (where the maximum node size is $2p-1$). This contradicts the false intuition that the probability of splitting at the leaf level would be $1/p$, as p insertions into an half-empty node causes a split. The reason is that the B-tree is growing; the factor of $2 \ln 2$ in the denominator is the result.

The *space utilization*, U, of a B-tree is the portion of space taken up by the B-tree that stores information. The following lemma, which has appeared in [12] and [16], will also be useful:

Lemma 2 *If U is the utilization of the B-tree, then*

$$U = \frac{1}{(2p-1)\sum a_j}$$

Corollary 2 *The utilization of a B-tree with pure-insert operations is*

$$U = \frac{2p[H(2p) - H(p)]}{2p-1} \approx 69\%$$

Where $H(p)$ is the harmonic function defined previously.

Proof: Sum the a_i and apply the lemma. •

2.2 Pure-Modify Operations

The next instruction mix that we will examine is *pure-modify*. Every operation is assumed to be a modify operation, which is modeled as a delete followed by an insert. That is, a modify deletes one key, then inserts one key. We assume that the B-tree is initially built from some sequence of operations (which may contain some deletes), then has a long sequence of modify operations applied to it. Both the insert and delete of the modify operation are assumed to be successful.

2.2.1 Ghost Model

Both [12] and [16] have analyzed the case of pure-modify operations on free-at-empty B-trees, but they assume that the probability that a node receives an insert is proportional to its size. The problem with this model is that after a large number of modify operations, the number of items in a node bears little relation to the number of insert operations that were performed on the node, yet the probability that a node will receive an insert is related to the number of inserts that have been performed on the node. We will define a *ghost* to be a data item that was inserted, then later deleted. Though the ghost no longer exists in the B-tree, it still affects the distribution of insert operations. For example, a node that contains only one key must contain at least $p-1$ ghosts, and therefore at least p inter-key spaces, because the node was at least half-full at one point. We must therefore keep track of the number of ghosts at each node.

Define:

N: number of items in the B-tree.

M: number of modify operations.

$A_i(M)$: expected number of order i nodes after modify operation M.

$a_i(M)$: $= A_i(M)/N$.

$B_i(M)$: expected number of order i nodes after the delete portion of modify operation M.

$C_i(M)$: average number of ghosts contained in all of the order i nodes after M modify operations.

$C'_i(M)$: average number of ghosts contained in an order i node after M modify operations. $C'_i(M) = C_i(M)/A_i$.

$c_i(M)$: proportion of ghosts that are in order i nodes after M modifies. $c_i(M) = C_i(M)/M$.

$D_i(M)$: average number of ghosts contained in all of the order i nodes after the delete portion of modify operation M.

$D'_i(M)$: average number of ghosts contained in an order i node after the delete portion of modify operation M. $D'_i(M) = D_i(M)/A_i$.

$d_i(M)$: proportion of ghosts that are in order i nodes after the delete portion of modify operation M. $d_i(M) = D_i(M)/M$.

P_m: probability of removing (merging) a node because of a delete.

Let us first develop the equations that describe the a_i. After the delete portion of the modify:

$$B_i(M+1) = (1 - i/N)A_i(M) + \frac{i+1}{N}A_{i+1}(M) \quad i \neq 2p-1$$
$$B_{2p-1}(M+1) = (1 - \frac{2p-1}{N})A_{2p-1}(M)$$

Next we turn to the insert. The number of nodes of order i after an insert is the number of nodes before the insert, minus the expected loss, plus the expected gain:

$$A_i(M+1) = (previous) - (\Pr[\text{order i node hit}])(loss) - (\Pr[\text{order i} - 1 \text{ node hit}])(gain)$$

Here, the previous value is $B_i(M+1)$ and the gain or loss is 1. After a large number of modifies, there will be many more ghosts than data items in the tree. Therefore, the probability that a node of order i receives the insert approaches $c_i(M)$.

$$\begin{aligned} A_i(M+1) = & \quad B_i(M+1) - c_i(M)\cdot 1 + c_{i-1}(M)\cdot 1 \\ = & \quad (1-i/N)A_i(M) + \tfrac{i+1}{N}A_{i+1}(M) - c_i(M) - c_{i-1}(M) \\ a_i(M+1) = & \quad (1-i/N)a_i(M) + \tfrac{i+1}{N}a_{i+1}(M) - c_i(M)/N + c_{i-1}(M)/N \end{aligned}$$

Solving for $\Delta a_i(M+1)$,

$$N\Delta a_i(M+1) = \quad -ia_i(M)+(i+1)a_{i+1}(M)-c_i(M)+c_{i-1}(M)$$

The equations for the $a_i(M)$ depend on values for the $c_i(M)$. We next develop equations that describe the $c_i(M)$. Again model a modify as a delete followed by an insert, and calculate the new number of ghosts as the previous number minus the expected loss plus the expected gain. Start with the delete:

$$D_i(M+1) = (previous) - (\Pr[\text{order i node hit}])(loss) + (\Pr[\text{other node hit}])(gain)$$

The previous value is $C_i(M)$. The probability that a delete is directed to an order i node is $iA_i(M)/N$, the proportion of data items contained in order i nodes. The loss is $C_i'(M)$, the expected number of ghosts in an order i node. D_i will gain if a delete was directed to an order $i+1$ node or to an order 1 node. The gain from an order $i+1$ node is $C_{i+1}'(M)+1$, because a new ghost is created. If a delete is directed to an order 1 node, then the node is merged. The node immediately to the right will receive the merged node's key range - thus will receive the merged node's ghosts. The probability that an order i node receives the deleted node's ghosts is $a_i(M)/\sum_{j=1}^{2p-1} a_j(M)$. Therefore:

$$\begin{aligned}
D_i(M+1) = & \quad C_i(M) - (\tfrac{iA_i(M)}{N})(C_i'(M)) + (\tfrac{(i+1)A_{i+1}(M)}{N})(C_{i+1}'(M)+1) \\
& \quad +(\tfrac{A_1(M)}{N})(\tfrac{a_i(M)}{\sum_{j=1}^{2p-1} a_j(M)})(C_1'(M+1)+1) \\
= & \quad C_i(M) - (\tfrac{iA_i(M)}{N})(\tfrac{C_i(M)}{A_i(M)}) + (\tfrac{(i+1)A_{i+1}(M)}{N})(\tfrac{C_{i+1}(M)}{A_{i+1}}+1) \\
& \quad +(\tfrac{A_1(M)}{N})(\tfrac{a_i(M)}{\sum_{j=1}^{2p-1} a_j(M)})(\tfrac{C_1'(M+1)}{A_1(M)}+1) \\
= & \quad C_i(M) - \tfrac{iC_i(M)}{N} + \tfrac{(i+1)C_{i+1}(M)}{N} + \tfrac{a_i(M)C_1(M)}{N\sum_{j=1}^{2p-1} a_j(M)} + (i+1)a_{i+1}(M) + \tfrac{a_i(M)a_1(M)}{\sum_{j=1}^{2p-1} a_j(M)} \\
\frac{D_i(M+1)}{M} = & \quad c_i(M) - \tfrac{ic_i(M)}{N} + \tfrac{(i+1)c_{i+1}(M)}{N} + \tfrac{a_i(M)c_1(M)}{N\sum_{j=1}^{2p-1} a_j(M)} + \tfrac{(i+1)a_{i+1}(M)}{M} + \tfrac{a_i(M)a_1(M)}{M\sum_{j=1}^{2p-1} a_j(M)}
\end{aligned}$$

The terms $((i+1)a_{i+1}(M))/M$ and $(a_i(M)a_1(M))/(M\sum_{j=1}^{2p-1} a_j(M))$ become insignificant as M becomes large, so we may drop them. Since $D_i(M+1)/M = D_i(M+1)/(M+1) + D_i(M+1)/[M(M+1)]$, we may assume that $D_i(M+1)/M \approx d_i(M+1)$, so that:

$$\begin{aligned}
d_i(M+1) = & \quad c_i(M) - \tfrac{ic_i(M)}{N} + \tfrac{(i+1)c_{i+1}(M)}{N} + \tfrac{a_i(M)c_1(M)}{N\sum_{j=1}^{2p-1} a_j(M)} \qquad i = 1,\ldots,2p-2 \\
d_i(M+1) = & \quad c_i(M) - \tfrac{ic_i(M)}{N} + \tfrac{a_i(M)c_1(M)}{N\sum_{j=1}^{2p-1} a_j(M)} \qquad i = 2p-1
\end{aligned}$$

The $C_i(M+1)$ evolve from the $D_i(M+1)$ in the following way:

$$C_i(M+1) = (previous) - (\Pr[\text{order i node hit}])(loss) + (\Pr[\text{order i} - 1 \text{ node hit}])(gain)$$

The previous is $D_i(M+1)$. The probability that an insert is directed to an order i node is $d_i(M+1)$, and the gain or loss is $D_i'(M+1)$. Therefore:

$$\begin{aligned}
C_i(M+1) = & \quad D_i(M+1) - (d_i(M+1))(D_i(M+1)/A_i(M)) + \\
& \quad (d_{i-1}(M+1))(D_{i-1}(M+1)/A_{i-1}(M)) \\
c_i(M+1) = & \quad d_i(M+1) - \tfrac{d_i^2(M+1)}{A_i(M)} + \tfrac{d_{i-1}^2(M+1)}{A_{i-1}(M)} \\
c_i(M+1) = & \quad c_i(M) - \tfrac{ic_i}{N} + \tfrac{(i+1)c_{i+1}(M)}{N} + \tfrac{a_i(M)c_1(M)}{N\sum_{j=1}^{2p-1} a_j(M)} - \tfrac{d_i^2(M+1)}{A_i(M)} + \tfrac{d_{i-1}^2(M+1)}{A_{i-1}(M)}
\end{aligned}$$

At the equilibrium point, the $a_i(M)$, $c_i(M)$ and the $d_i(M)$ are constant, so remove their dependence on M. Including the exceptional cases at $i = 1, p, 2p-1$, the equations that determine the equilibrium point of the a_i are:

$$
\begin{array}{lcll}
0 = & -a_i + (i+1)a_{i+1} - c_i & i = 1 & (2.1) \\
0 = & -ia_i + (i+1)a_{i+1} - c_i + c_{i-1} + 2c_{2p-1} & i = p & (2.2) \\
0 = & -ia_i - c_i + c_{i-1} & i = 2p-1 & (2.3) \\
0 = & -ia_i + (i+1)a_{i+1} - c_i + c_{i-1} & \text{otherwise} & (2.4)
\end{array}
$$

In addition, $\sum_{i=1}^{2p-1} ia_i = 1$. If we solve for the point at which $N\Delta c_i(M+1) = 0$, the following equations describe the equilibrium point of the c_i:

$$
\begin{array}{lcl}
0 = & -ic_i + (i+1)c_{i+1} + \frac{a_i c_1}{\sum_{j=1}^{2p-1} a_j} - \frac{d_i^2}{a_i} & i = 1 \\
0 = & -ic_i + (i+1)c_{i+1} + \frac{a_i c_1}{\sum_{j=1}^{2p-1} a_j} - \frac{d_i^2}{a_i} + \frac{d_{i-1}^2}{a_{i-1}} + \frac{d_{2p-1}^2}{a_{2p-1}} & i = p \\
0 = & -ic_i + \frac{a_i c_1}{\sum_{j=1}^{2p-1} a_j} - \frac{d_i^2}{a_i} + \frac{d_{i-1}^2}{a_{i-1}} & i = 2p-1 \\
0 = & -ic_i + (i+1)c_{i+1} + \frac{a_i c_1}{\sum_{j=1}^{2p-1} a_j} - \frac{d_i^2}{a_i} + \frac{d_{i-1}^2}{a_{i-1}} & \text{otherwise}
\end{array}
$$

Additionally, $\sum_{i=1}^{2p-1} c_i = 1$.

We can simplify the equations beyond this: Consider equations (2.1 - 2.4). Each is parameterized for a certain set of values of i. Let the equation for a particular i be denoted by e_i. Add e_1 and e_2 to get:

$$0 = 3a_3 - c_2 - a_1$$

Adding e_3 to this, we get

$$0 = 4a_4 - c_2 - a_1$$

This form continues up to e_{p-1}

$$0 = pa_p - c_{p-1} - a_1$$

Adding e_p, we get

$$0 = (p+1)a_{p+1} - c_p + 2c_{2p-1} - a_1$$

This form continues until e_{2p-1}

$$0 = (2p-1)a_{2p-1} - c_{2p-2} + 2c_{2p-1} - a_1$$

Adding e_{2p-1}, we get:

$$a_1 = c_{2p-1}$$

Substituting c_{2p-1} for a_1 in the previous equations, we get:

$$
\begin{array}{rll}
a_1 = & c_{2p-1} & \\
ia_i = & c_{i-1} + c_{2p-1} & 2 \le i \le p \\
ia_i = & c_{i-1} - c_{2p-1} & p+1 \le i \le 2p-1
\end{array}
$$

Examining the equations for the d_i, we see that $d_i \to c_i$ as $N \to \infty$. Let:

$$v = \sum_{i=1}^{2p-1} a_i = \sum_{i=1}^{2p-1} c_i/(i+1) + [2H(p) - H(2p-1)]c_{2p-1}$$

Using this and the equations for the a_i, we get:

Theorem 2 *The values of c_i, $1 \leq i \leq 2p-1$ satisfy the following system of non-linear equations:*

$$\begin{array}{rcl}
0 = & -c_1 + 2c_2 + \frac{c_{2p-1}c_1}{v} - \frac{c_1^2}{c_{2p-1}} & i=1 \\
0 = & -2c_2 + 3c_3 + \frac{(c_1+c_{2p-1})c_1}{2v} - \frac{2c_2^2}{(c_1+c_{2p-1})} + \frac{c_1^2}{c_{2p-1}} & i=2 \\
0 = & -ic_i + (i+1)c_{i+1} + \frac{(c_{i-1}-c_{2p-1})c_1}{iv} - \frac{ic_i^2}{c_{i-1}+c_{2p-1}} + \frac{(i-1)c_{i-1}^2}{c_{i-1}+c_{2p-1}} & 2 < i < p \\
0 = & -pc_p + (p+1)c_{p+1} + \frac{(c_{p-1}+c_{2p-1})c_1}{pv} & \\
 & -\frac{pc_p^2}{c_{p-1}+c_{2p-1}} + \frac{(p-1)c_{p-1}^2}{c_{p-1}+c_{2p-1}} + \frac{(2p-1)c_{2p-1}^2}{c_{2p-2}-c_{2p-1}} & i=p \\
0 = & -ic_i + (i+1)c_{i+1} + \frac{(c_{i-1}-c_{2p-1})c_1}{iv} - \frac{ic_i^2}{c_{i-1}-c_{2p-1}} + \frac{(i-1)c_{i-1}^2}{c_{i-2}+c_{2p-1}} & i=p+1 \\
0 = & -ic_i + (i+1)c_{i+1} + \frac{(c_{i-1}-c_{2p-1})c_1}{iv} - \frac{ic_i^2}{c_{i-1}-c_{2p-1}} + \frac{(i-1)c_{i-1}^2}{c_{i-1}-c_{2p-1}} & p+1 < i < 2p-1 \\
0 = & -(2p-1)c_{2p-1} + \frac{(c_{2p-2}-c_{2p-1})c_1}{(2p-1)v} - \frac{(2p-1)c_{2p-1}^2}{c_{2p-2}-c_{2p-1}} + \frac{(2p-2)c_{2p-2}^2}{c_{2p-3}-c_{2p-1}} & i=2p-1 \\
0 = & 1 - \sum_{j=1}^{2p-1} c_j &
\end{array}$$

The equations for the Δc_i are linearly dependent: they sum to 0. Remove one of the equations to get $2p-1$ equations in $2p-1$ variables, and the system is ready to be solved by a non-linear equation solving package. The package we used is NAG [8].

Three simulation experiments were run. A B-tree was built using insert operations, and then a long sequence of modify operations was performed. The experiments were run for $p = 5, 10, 15, 20$. Table 1 compares the utilization, U, and probability of splitting, P_s, from the simulation and the ghost model. The calculated values for the utilization are within 10% of the utilization observed in the simulation. The calculated and observed probabilities of splitting diverge, but they both decrease exponentially with p. The values of the probability of splitting or merging are difficult to calculate because they become very small as the maximum node size increases. Figure 1 compares the distribution of the a_i. Figure 2 shows the calculated utilization for p between 5 and 25. In the model as well as the simulation, the utilization approaches 39%, and the probability of splitting or deleting a node decreases exponentially with p.

2.3 Parameterized Inserts and Deletes

An intermediate operation mix between pure-insert and pure-modify is the case where there are deletes in the operation mix, but there are more inserts than deletes. We will call this operation mix *parameterized by q*. Let us define:

L: number of operations performed.

q: probability that a given operation is a delete.

Therefore, an insert will occur with probability $1-q$ and the expected number of items in the tree after L operations will be $L(1-2q)$.

2.3.1 Ghost Model

Because there are deletes in the instruction mix, but there are more inserts than deletes, both the number of items and the number of ghosts in the leaves will grow with L. The expected number of items in the tree will be $L(1-2q)$ after L operations, and the expected number of ghosts will be Lq, so that the total expected number of ghosts and items will be $L(1-q)$. Therefore $C_i = Lqc_i$ and $A_i = L(1-2q)a_i$. The actual number of items and ghosts in the B-tree will have a binomial distribution, but by the law of large numbers, the deviation will become negligible compared to the mean.

The equations that describe the A_i, $1 < i < 2p-1$ and $i \neq p$ are:

$$\begin{aligned}
A_i(L+1) = & \quad q\left(\frac{-i}{L(1-2q)}A_i(L) + \frac{i+1}{L(1-2q)}A_{i+1}(L)\right) + \\
& (1-q)\left(\frac{C_i(L)+iA_i(L)}{L(1-q)}(-1) + \frac{C_{i-1}(L)+(i1-)A_{i-1}(L)}{L(1-q)}\right) + A_i(L) \\
(L+1)(1-2q)a_i(L+1) = & \quad -iqa_i(L) + (i+1)qa_{i+1}(L) - qc_i(L) - i(1-2q)a_i(L) \\
& +qc_{i-1}(L) + (i-1)(1-2q)a_{i-1}(L) + L(1-2q)a_i(L) \\
(L+1)(1-2q)\Delta a_i(L+1) = & \quad -(i(1-q)+(1-2q))a_i - qc_i + (i+1)qa_{i+1} + \\
& qc_{i-1} + (i-1)(1-2q)a_{i-1}
\end{aligned}$$

The equations that describe the C_i, $1 < i < 2p-1$ and $i \neq p$ are:

$$\begin{aligned}
& C_i(L+1) = \\
& \quad q\left[\frac{iA_i(L)}{L(1-2q)}\left(-\frac{C_i(L)}{A_i(L)}\right) + \frac{(i+1)A_{i+1}}{l(1-2q)}\left(\frac{C_{i+1}(L)}{A_{i+1}(L)}+1\right) + \left(\frac{A_1(L)}{L(1-2q)}\right)\left(\frac{A_i(L)}{\sum A_j(L)}\right)\left(\frac{A_1(L)}{C_1(L)}+1\right)\right] \\
& \qquad +(1-q)\left(\frac{C_i(L)+iA_i(L)}{L(1-q)}\left(\frac{C_i(L)}{A_i(L)}\right) + \frac{C_{i-1}(L)+A_{i-1}(L)}{L(1-q)}\left(\frac{C_{i-1}(L)}{A_{i-1}(L)}\right)\right) + C_i(L) \\
& (L+1)qc_i(L+1) = \\
& \qquad iqa_i(L)\left(-\frac{qc_i(L)}{(1-2q)a_i(L)}\right) + (i+1)qa_{i+1}(L)\left(\frac{qc_{i+1}(L)}{(i-2q)a_{i+1}}+1\right) + \\
& \quad qa_1(L)\left(\frac{a_i(L)}{\sum a_j(L)}\right)\left(\frac{qc_1(L)}{(1-2q)a_1(L)}+1\right) + (qc_i(L)+i(1-2q)a_i(L))\left(-\frac{qc_i(L)}{(1-2q)a_i(L)}\right) + \\
& \qquad (qc_{i-1}(L)+(i-1)(1-2q)a_{i-1}(L))\left(\frac{qc_{i-1}(L)}{(1-2q)a_{i-1}(L)} + Lqc_i(L)\right. \\
& (L+1)q\Delta c_i = \\
& \quad -c_i\left(\frac{iq^2}{1-2q} + \frac{q^2c_i}{(1-2q)a_i} + iq + q\right) + \frac{(i+1)q^2c_{i+1}}{1-2q} + c_{i-1}\left(\frac{q^2c_{i-1}}{(1-2q)a_{i-1}} + (i-1)q\right) + \\
& \qquad qa_i\left(\frac{qc_1+(1-2q)a_1}{(1-2q)\sum a_j}\right) + (i+1)qa_{i+1}
\end{aligned}$$

Let $v = \sum a_i$. After taking into account the special cases at $i=1$, $i=p$ and $i=2p-1$, and performing algebraic manipulation, we get:

Theorem 3 *For a free-at-empty B-tree that is parameterized by q, the steady state values of a_i and c_i, $1 \leq i \leq 2p-1$ is the solution to the following set of non-linear equations:*

$$\begin{aligned}
0 = & \quad -(i(1-q)+(1-2q))a_i - qc_i + (i+1)qa_{i+1} & i=1 \\
0 = & \quad -(i(1-q)+(1-2q))a_i - qc_i + (i+1)qa_{i+1} + qc_{i-1} + (i-1)(1-2q)a_{i-1} & \\
& \quad +2qc_{2p-1} + 2(2p-1)(1-2q)a_{2p-1} & i=p \\
0 = & \quad -(i(1-q)+(1-2q))a_i - qc_i + qc_{i-1} + (i-1)(1-2q)a_{i-1} & i=2p-1 \\
0 = & \quad -(i(1-q)+(1-2q))a_i - qc_i + (i+1)qa_{i+1} + qc_{i-1} + (i-1)(1-2q)a_{i-1} & \text{otherwise}
\end{aligned}$$

$$\begin{aligned}
0 = & \quad -c_i(iq + qc_i/a_i + (1-2q)(i+1)) + (i+1)qc_{i+1} & \\
& \quad +\frac{a_i}{v}(qc_1 + (1-2q)a_1) + (1-2q)(i+1)a_{i+1} & i=1 \\
0 = & \quad -c_i(iq + qc_i/a_i + (1-2q)(i+1)) + c_{i-1}(qc_{i-1}/a_{i-1} + (1-2q)(i-1)) + & \\
& \quad \frac{a_i}{v}(qc_1 + (1-2q)a_1) + c_{2p-1}\left(\frac{qc_{2p-1}}{a_{2p-1}} + (1-2q)(2p-1)\right) & \\
& \quad +(i+1)qc_{i+1} + (1-2q)(i+1)a_{i+1} & i=p \\
0 = & \quad -c_i(iq + qc_i/a_i + (1-2q)(i+1)) + c_{i-1}(qc_{i-1}/a_{i-1} + (1-2q)(2p-2)) & \\
& \quad +\frac{a_i}{v}(qc_1 + (1-2q)a_1) & i=2p-1
\end{aligned}$$

$$0 = \quad -c_i(iq + qc_i/a_i + (1-2q)(i+1)) + c_{i-1}(\frac{qc_{i-1}}{a_{i-1}} + (1-2q)(i-1)) + \frac{a_i}{v}(qc_1 + (1-2q)a_1) + (i+1)qc_{i+1} + (1-2q)(i+1)a_{i+1} \qquad \text{otherwise}$$

$$0 = \quad 1 - \sum_{j=1}^{2p-1} c_i$$

The equations for the equilibrium point of the Δc_i are linearly dependent, as again they sum to zero. In order to solve the system, remove one of the Δc_i equations, which gives $4p-2$ equations in $4p-2$ variables.

Note that if $q = .5$, then we have the pure modify model, and if $q = 0$, then we have the pure insert model. Thus this model generalizes both the pure-modify ghost model and Yao's pure-insert model.

If we add together the equations for the equilibrium points of the Δa_i, we get the relation:

$$0 = -(1-2q)\sum_{i=1}^{2p-1} a_i - a_1 q + (1-2q)(2p-1)a_{2p-1} + qc_{2p-1}$$

The formula can be interpreted as follows: On a delete operation, the probability of merging a node is a_1. Since q is the probability of an operation being a delete, $qa_1 = R_d =$ the rate at which nodes are merged, in nodes per operation. Similarly, $(1-2q)(2p-1)a_{2p-1} + qc_{2p-1} = R_s =$ the rate at which nodes split. If we let $R_s - R_d = R_g =$ the rate at which nodes are added to the B-tree, and we recall the relationship between U and $\sum a_i$, we have the relationship:

$$U = \frac{1-2q}{(2p-1)R_g} \tag{1}$$

As we shall see in the section 2.3.2, U remains stable over a wide range of q if p is large. Since R_d is negligible compared to R_s if $q < .5$, the relationship can be used as a rule of thumb to calculate the probability of splitting (see section 4.1).

The equations were solved for $q = .45$ and $q = .47$, and p varying between 5 and 20. Tables 2 and 3 show a comparison between analytical results and simulation results.

Unfortunately, these equations are difficult to solve, as non-linear equation solvers require a good estimate of the starting point. We calculated the solutions for only a few cases. This points out the need for an approximation that can be easily solved.

2.3.2 Linear Model

The difficulty in solving the ghost model of a B-tree is that the ghosts satisfy a non-linear recurrence. Ghosts are necessary when the number of ghosts in a node outnumber the number of items. If q is small, however, then the number of items in a node is greater than the number of ghosts in a node. In addition, the distribution of the ghosts among the nodes of different order has a similar distribution to the number of items. Therefore, we can try making the approximation that the probability that a node of order i receives an insert is ia_i. Using this approximation gives us linear equations, so we will call this model the *linear model*. Mizoguchi described a similar model for B-trees with even maximum node size [12]. In this section, we will describe the linear model using the methods that we have developed, examine its range of accuracy, then examine the results on B-tree utilization.

The linear model is described by the following recurrence:

$$\begin{aligned} A_i(L+1) = \quad & q\left[(1 - \tfrac{i}{(1-2q)L}A_i(L) + \tfrac{i+1}{(1-2q)L}A_{i+1}(L)\right] \\ & +(1-q)\left[(1 - \tfrac{i}{(1-2q)L})A_i(L) + \tfrac{i-1}{(1-2q)L}A_{i-1}(L)\right] \end{aligned}$$

If we then solve for ΔP_i, where $P_i = ia_i$, we get:

Theorem 4 *Under the approximation that a node of order i receives an insert is ia_i, the P_i are the solution to the following set of linear equations:*

$$\begin{array}{rcl} 0 = & -(1+(1-2q))P_1 + qP_2 & i = 1 \\ 0 = & -(p+(1-2q))P_p + (1-q)pP_{p-1} + qpP_{i+1} + 2p(1-q)P_{2p-1} & i = p \\ 0 = & -(2p-1+(1-2q))P_{2p-1} + (1-q)(2p-1)P_{2p-2} & i = 2p-1 \\ 0 = & -(i+(1-2q))P_i + (1-q)iP_{i-1} + qiP_{i+1} & \text{otherwise} \\ 0 = & 1 - \sum_{i=1}^{2p-1} P_i & \end{array}$$

The existence, uniqueness and stability of the equilibrium point holds by the previous argument.

Tables 4,5, 6 and 7 show a comparison between the simulation results and the analytic results. Examining the tables, we see that the linear approximation is very accurate for $q < .4$. At $q = .45$, the ghost model gives results that more closely match those of the simulation. The case of $q = .5$ gives the free-at-empty pure-modify models of [12] and [16]; Table 4 shows a comparison. The a_i calculated from the model for $q = .5$ is plotted along with the simulation and ghost model results in Figure 1. As can be seen, the linear model does not calculate the distribution of nodes for $q = .5$.

If $q < .5$ however, the linear model becomes more accurate as p increases. In other words, when inserts significantly outnumber deletes and the maximum node size increases, the linear model becomes more accurate. This happens for two reasons: as p increases, the distribution of the c_i becomes more similar to the distribution of the ia_i; also, there are fewer very small nodes, the nodes for which ia_i is the poorest approximation to c_i.

Figure 3 shows a comparison between the ghost model, the linear model and the simulation. Notice that the linear model always gives low estimates of the utilization, even for the pure-modify case. The low estimates of the utilization occur because the linear model underestimates the probability that a small node receives an insert and thus becomes larger (a small node must have received many deletes). This means that we can use the linear model to provide a lower bound on the utilization of a random B-tree.

Figure 4 shows how the utilization varies with p for different q. Notice that as p becomes large, the utilization curve becomes flatter with a sharper knee near $q = .5$. If we examine the entries in Table 5 for $p = 100$, we see that the linear model predicts a utilization of 65.48% even when $q = .47$, which is a lower bound by the previous paragraph.

While the linear model gives reasonable estimates of the space utilization, it gives very poor estimates of the probability of splitting or merging when q becomes close to .5. The linear model predicts a quadratic decrease in P_s and P_m as p becomes larger for pure-modify operations [12], but the ghost model and the simulation show an exponential decrease. Since the linear model underestimates the probability that a small node receives an insert, the linear model overestimates the probability of merging on a delete (the simulation and the ghost model predict that P_m is almost zero if $q < .5$). Because the predicted utilization is close to the actual utilization, the rule-of-thumb requires that the overestimate of merges must be balanced by an overestimate of splits. However, if q is small enough ($q < .4$), then the linear model gives good estimates of P_m and P_s.

3 Merge-at-Half B-trees

We next analyze merge-at-half B-trees. The analysis of merge-at-half B-trees is much more difficult than the analysis of free-at-empty B-trees because the result of restructuring due to a merge is much more complicated in a merge-at-half B-tree than in a free-at-empty B-tree. Therefore, we present only an approximate analysis for the purpose of comparing to free-at-empty B-trees. The following analysis is similar to the analysis of [16], who model the B-tree nodes using differential equations. However, [16] examines only the cases of pure-insert and pure-modify. For a comparison, we need to know the utilization for the entire range of q, because merges might (and in fact do) cause the utilization to increase when there are deletes in the instruction mix. Also, [16] does not calculate the restructuring probabilities.

3.1 Analysis

The first step in the analysis is to specify how the tree is modified on an insert or a delete. The action on an insert is the same that for a free-at-empty B-tree.

$$\begin{aligned}(X_p,\ldots,X_i,X_{i+1},\ldots,X_{2p-1}) &\rightarrow (X_p,\ldots,X_i-1,X_{i+1}+1,\ldots,X_{2p-1})\\ (X_p,\ldots,X_{2p-1}) &\rightarrow (X_p+2,\ldots,X_{2p-1}-1)\end{aligned}$$

The action on a delete is also the same as that for a free-at-empty B-tree if no node gets merged (no delete from an order p node).

$$(X_p,\ldots,X_i,X_{i+1},\ldots,X_{2p-1}) \rightarrow (X_p,\ldots,X_i+1,X_{i+1}-1,\ldots,X_{2p-1})$$

If a delete is directed to an order p node, then the node gets merged with its neighbor. Thus the action of a delete that causes a merge depends on the sibling of the merged node. Suppose that the sibling is an order j node. Then

$$\begin{array}{ll}(X_p,\ldots,X_{2p-1}) \rightarrow & \\ \qquad (X_p-2,\ldots,X_{2p-1}+1) & j=p\\ (X_p,\ldots,X_{(p+j-1)/2},\ldots,X_j,\ldots,X_{2p-1}) \rightarrow & \\ \qquad (X_p-1,\ldots,X_{(p+j-1)/2}+2,\ldots,X_j-1,\ldots,X_{2p-1}) & p+j-1 \text{ is even}\\ (X_p,\ldots,X_{(p+j)/2-1},X_{(p+j)/2},\ldots,X_j,\ldots,X_{2p-1}) \rightarrow & \\ \qquad (X_p-1,\ldots,X_{(p+j)/2-1}+1,X_{(p+j)/2}+1,\ldots,X_j-1,\ldots,X_{2p-1}) & p+j-1 \text{ is odd}\end{array}$$

Note that not every node order has the chance of gaining on a merge operation. In particular, if p is even, then $(3p-2)/2$ is the largest order of nodes that will gain; if p is odd, then $(3p-1)/2$ is the largest order of nodes that will gain.

Since the structure of the evolution of the B-tree is different if p is odd or even, let us examine first the case when p is even. An order i node will be merged with its neighbor if its neighbor is of order p and receives a delete (assume a node merges with its right neighbor, except if the node is the rightmost node, in which case it merges with its left neighbor). The probability that a node of order i is the neighbor of the merging node is $A_i(L)/V(L)$, where $V(L)=\sum_{i=p}^{2p-1}A_i(L)$. The result of the merge operation is two nodes of order $(i+p-1)/2$, or a node of order $(i+p)/2$ and a node of order $(i+p-2)/2$. Therefore, one node of order j will be created if a node of order $2j-p$ or a node or order $2i-p+2$ is merged, and two nodes of order i will be created if a node of order $2j-p+1$ is merged. If $j=(3p-2)/2$, then $2j-p+2=2p$, so nodes of order $(3p-2)/2$ can't be created from merges involving nodes of order $2j-p+2$. If the neighboring node involved in the merge is of order p, then a node of order $2p-1$ is created. So if $j=p$, then nodes of order j aren't created from merges with nodes of order $2j-p=p$.

In order to simplify the analysis, approximate the probability that an insert is directed to an order i node by

$$\frac{iA_i(L)}{L(1-2q)}$$

Using this approximation, we have:

Theorem 5 *If p is even and we use the linear approximation, then in a merge-at-half B-tree the a_i, $1\le i\le p$ satisfy the following set of nonlinear equations:*

$$\begin{array}{lll}0= & -(p+(1-2q))a_p+q(p+1)a_{p+1}+2(1-q)(2p-1)a_{2p-1} & \\ & \quad +\frac{qpa_p}{v}(-a_p+2a_{p+1}+a_{p+2}) & i=p\\ 0= & -(i+(1-2q))a_i+q(i+1)a_{i+1}+(1-q)(i-1)a_{i-1} & \\ & \quad +\frac{qpa_p}{v}(-a_i+a_{2i-p}+2a_{2i-p+1}+a_{2i-p+2}) & p<i<\dfrac{3p-2}{2}\end{array}$$

$$
\begin{array}{lll}
0 = & -(i+(1-2q))a_i + q(i+1)a_{i+1} + (1-q)(i-1)a_{i-1} & \\
& +\frac{qpa_p}{v}(-a_i + a_{2i-p} + 2a_{2i-p+1}) & i = \dfrac{3p-2}{2} \\
0 = & -(i+(1-2q))a_i + q(i+1)a_{i+1} + (1-q)(i-1)a_{i-1} & \\
& -\frac{qpa_pa_i}{v} & 3p/2 \leq i < 2p-1 \\
0 = & -(i+(1-2q))a_i + (1-q)(i-1)a_{i-1} & \\
& +\frac{qpa_p}{v}(-a_i + a_p) & i = 2p-1 \\
0 = & 1 - \sum_{j=p}^{2p-1} ia_i &
\end{array}
$$

When p is odd, the equations that describe the a_i are the same as when p is even, except that $a_{(3p-1)/2}$ is the highest order node that can gain on a merge, and only from an order $2i-p$ node:

Theorem 6 *If p is odd and we use the linear approximation, then in a merge-at-half B-tree the a_i, $1 \leq i \leq p$ satisfy the following set of nonlinear equations:*

$$
\begin{array}{lll}
0 = & -(p+(1-2q))a_p + q(p+1)a_{p+1} + 2(1-q)(2p-1)a_{2p-1} & \\
& +\frac{qpa_p}{v}(-a_p + 2a_{p+1} + a_{p+2}) & i = p \\
0 = & -(i+(1-2q))a_i + q(i+1)a_{i+1} + (1-q)(i-1)a_{i-1} & \\
& +\frac{qpa_p}{v}(-a_i + a_{2i-p} + 2a_{2i-p+1} + a_{2i-p+2}) & p < i < \dfrac{3p-1}{2} \\
0 = & -(i+(1-2q))a_i + q(i+1)a_{i+1} + (1-q)(i-1)a_{i-1} & \\
& +\frac{qpa_p}{v}(-a_i + a_{2i-p}) & i = \dfrac{3p-1}{2} \\
0 = & -(i+(1-2q))a_i + q(i+1)a_{i+1} + (1-q)(i-1)a_{i-1} & \\
& -\frac{qpa_pa_i}{v} & \dfrac{3p-1}{2} < i < 2p-1 \\
0 = & -(i+(1-2q))a_i + (1-q)(i-1)a_{i-1} & \\
& +\frac{qpa_p}{v}(-a_i + a_p) & i = 2p-1 \\
0 = & 1 - \sum_{j=p}^{2p-1} ia_i &
\end{array}
$$

We solved the above set of non-linear equations with the numerical analysis package NAG [8].

3.2 Comparison

We modified the free-at-empty simulator to make a merge-at-half B-tree and performed experiments to compare against the results of our analysis. In Tables 8, 9 and 10, we compare the results from the analysis and the simulations.

The utilization predicted by the analysis and by the simulation agree well, although the difference increases as q approaches .5. Surprisingly, the space utilization can increase as deletes become more common in the instruction mix. The utilization increases because a merging node will increase in size. However, the increase in utilization is small, as the utilization never goes above 71%.

The space utilization stays level for most values of the parameter q, but decreases sharply as q approaches .5. Furthermore, for $q = .5$, the space utilization decreases as p increases. Both the analysis and the simulation indicate that the utilization will approach about 60%.

The analysis and the simulation do not agree as well on the probability of splitting or merging. The difference is small when $q = .1$, but becomes larger as q increases. The increasing error is due to ignoring the effect of ghosts, whose effect becomes more significant as q increases. The analysis consistently underestimates the probability of splitting or merging.

Next, we use the results on free-at-empty B-trees from section 2 to compare merge-at-half B-trees against free-at-empty B-trees. For the comparison, we used a B-tree with $p = 40$. In Figure 5, we compare the utilization of a merge-at-half and a free-at-empty B-tree. The figure shows that the utilization of both B-trees remains close for most values of the parameter q. Up to $q = .45$ the utilization of the merge-at-half B-tree is less than 10% greater than the utilization of the free-at-empty tree, but when $q = .5$, the difference becomes 60%

In Figure 6, we compare the probability of restructuring on an operation ($q\,\Pr[\text{merge on delete}] + (1-q)\Pr[\text{split on insert}]$). The probability of restructuring a merge-at-half B-tree is 20% greater when $q = .1$ and 300% greater when $q = .45$. When $q = .5$, the rates cannot be compared because the probability of restructuring a free-at-empty B-tree becomes infinitesimal, but the probability of restructuring a merge-at-half tree becomes large. Table 11 also compares restructuring rates for several values of the parameter p.

4 Conclusion

This chapter has presented methods for calculating the equilibrium utilization and distribution of nodes in a B-tree when the probability of an operation being a delete ranges between 0 and .5. The random B-tree is modeled by a set of simultaneous recurrence equations. By transforming the recurrence equations into difference equations and searching for the equilibrium point, the problem is transformed into finding the solution of a set of equations.

In order to model the situation when deletes are allowed into the operation mix, the notion of a ghost (an item that was deleted from the B-tree) is used. The distribution of ghosts in the B-tree affects the input probability distribution; thus it must be calculated along with the distribution of the nodes in the B-tree. The ghost model achieved accurate calculations: less than 5% error when $q \leq .45$ and $p \geq 20$ in calculating both the utilization and the probability of splitting on an insert.

As q approaches .5, the distribution of nodes and ghosts in the B-tree becomes harder to calculate accurately. When $q = .47$, the error in calculating the utilization and the probability of splitting becomes 10%. When $q = .5$, the error in calculating the utilization is within 10%, but the analytical probability of splitting is inaccurate. However, the simulation agrees with the model's prediction that the probability of splitting or merging is very small and decreases exponentially with p.

The major problem with the ghost model is that it is hard to solve. An approximation to the ghost model is presented that reduces the problem of calculating the utilization and probability of splitting and merging in a random B-tree to solving a set of simultaneous linear equations. This model is very accurate for a wide range of the parameter q. However, when q becomes larger than .4, the linear model becomes inaccurate and the ghost model must be used.

4.1 Pragmatic Conclusions

The calculations of the utilization of random free-at-empty B-trees parameterized by p and q shows that U remains between 60% and 70% for most values of q ($q \leq .45$) if p is large ($p \geq 15$), but drops to 39% when $q = .5$, or if all operations are modifies. Trees in which deletes outnumber inserts are rarely interesting in practice and are difficult to model in the limit. The knee of the utilization curve gets closer to $q = .5$ and becomes sharper as p becomes larger.

The tendency of the utilization to remain near 69% can be explained by the following arguments: If there are even just a few more inserts than deletes, the B-tree will grow at the net insert rate (the rate of inserts minus the rate of deletes). Furthermore, nodes with more items are more likely to get hit by a delete than smaller nodes, so that 1) it is hard for small nodes to become smaller and 2) larger nodes tend to bunch up near 69%. These trends can be seen in Figure 7. The curve has the same shape as a curve for a pure-insert B-tree. The number of nodes with i items decreases exponentially as i decreases when $i < p$. Since the largest nodes are the most likely to get hit with a delete, the largest nodes are pushed downwards. Fewer nodes get split, so fewer half-full nodes appear, and half-full nodes tend to get pushed upwards. These tendencies become stronger as p increases, causing the utilization to stay near the pure insert utilization even as q approaches .5.

The simulation and the ghost model show that the probability of merging a node on a delete in a free-at-empty B-tree is almost zero. From the parameterized ghost model, we can relate the utilization of the B-tree and the probability of splitting on an insert by the formula:

$$P_s = \frac{1-2q}{(1-q)(2p-1)U} \tag{2}$$

This follows from the relation 1 between the rate of growth and the utilization derived in section 2.3.1 by assuming that the number of merges is negligible and adjusting the formula to 'per insert' instead of 'per operation'.

As it stands, the relationship 2 is not useful unless we have either the utilization or the probability of splitting available. In Figures 3 and 4, we see that the utilization tends to remain near the pure-insert level until q is approximately .47, so we can estimate the utilization to be 68% regardless of p and q (approximating the space utilization by 68% instead of 69% increases the range of the rule of thumb). This gives:

Rule of Thumb 1 (Probability of splitting in a free-at-empty B-tree)

$$P_s = \frac{1-2q}{(1-q)} \cdot \frac{1}{.68(2p-1)}$$

Figure 8 compares the probability of splitting derived from the rule of thumb against the simulation for varying q with $p = 30$ and $p = 40$. As can be seen, the rule of thumb gives very good estimates of the probability of splitting up to at least $q = .45$.

The tendency of free-at-empty B-tree utilization to remain near the pure-insert level suggests that merging B-tree nodes when they are empty is not a wasteful strategy in terms of storage and is significantly better in terms of restructuring.

The upper levels of a free-at-empty bottom-up restructuring B-tree have the same distribution of node sizes that the leaf-level of a pure-insert has, as intuition would suggest. The simple structure of the upper levels of a free-at-empty B-tree allows a simple but accurate calculation the root level and order.

In this paper, we also solved an approximate model of a merge-at-half B-tree. The simplified model seriously underestimates the probability of restructuring. However, for the purpose of comparison to a free-at-empty B-tree, the underestimation is not a problem.

A merge-at-half B-tree will always have a space utilization of at least 50%. When all operations are modify operations, or when the number of insert operations is the same as the number of delete operations, then the utilization will be about 60%. In contrast, a free-at-empty B-tree has a 0% lower bound on its space utilization, and will have about 39% utilization on a pure-modify instruction mix. However, the space utilization of a free-at-empty B-tree remains high if there are just a few more insert operations than delete operations. Thus, merge-at-half usually buys little in terms of space utilization.

In Figure 6, we showed that the restructuring rate of a merge-at-half B-tree is significantly larger than the restructuring rate of a free-at-empty B-tree for all values of $q \geq .1$. For many concurrent B-tree algorithms used in practice [4, 13], restructuring causes a serialization bottleneck. Thus, one simple but important way to increase concurrency in B-tree operations is to reduce the probability of restructuring. Since merge-at-half buys little space utilization but is expensive in terms of restructuring, we recommend that B-trees (especially concurrently accessed ones) use free-at-empty.

5 Acknowledgements

We would like to thank Jeanette Schmidt for helpful suggestions concerning ghosts, the referees for their careful reading, and Johna Johnson for her editorial assistance.

References

[1] R. Baeza-Yates. Expected behavior of B^+-trees under random inserts. *Acta Informatica*, 27, 1989.

[2] R. Baeza-Yates and P. Larson. Performance of B^+-trees with partial expansions. *IEEE Trans. on Knowledge and Data Engineering*, 1, 1989.

[3] R. Bayer and E.M. McCreight. Organization and maintainance of large ordered indices. *Acta Informatica*, 1(3):173–189, 1972.

[4] R. Bayer and M. Schkolnick. Concurrency of operations on B-trees. *Acta Informatica*, 9:1–21, 1977.

[5] J. Driscoll, S. Lang, and L. Franklin. Modeling B-tree insertion activity. *Information Processing Letters*, 26:5–18, 1987.

[6] B. Eisenbarth and et. al. The theory of fringe analysis and its applications application to 2-3 trees and B-trees. *Information and Control*, 55(1):125–174, 1982.

[7] G. Gupta and B. Srinivasan. Approximate storage utilization of B-trees. *Information Processing Letters*, 22:243–246, 1986.

[8] NAG (USA) inc. NAG. 1131 Warren Ave.; Downers Grove, IL 60515.

[9] K. Kuspert. Storage utilization in B^*-trees with a generalized overflow technique. *Acta Informatica*, 19(4):35–55, 1983.

[10] C. Langenhop and W. Wright. An efficient model for representing and analyzing B-trees. In *ACM-NCC*, pages 35–40, 1985.

[11] C. Leung. Approximate storage utilization of B-trees: A simple derivation and generalizations. *Information Processing Letters*, 19:199–210, 1984.

[12] T. Mizoguchi. On the required space for random split trees. In *Allerton Conference*, pages 265–273, Monticello, IL, 1979.

[13] C. Mohan. ARIES/KVL: A key-value locking method for concurrency control of multiaction transactions operating on b-tree indexes. Research Report RJ 6864, IBM, 1989.

[14] C. Mohan and F. Levine. ARIES/IM: An efficient and high concurrency index management method using write-ahead logging. Research Report RJ 6864, IBM, 1989.

[15] T. Nakamura and T. Mizoguchi. An analysis of storage utilization in block split, data structuring scheme. In *4th International Conference on Very Large Databases*, pages 489–495, Berlin, 1978.

[16] K. Quitzow and M. Klopprogge. Space utilization and access path length in B-trees. *Information Systems*, 5:7–16, 1980.

[17] H. Wedekind. On the selection of access paths in a data base system. In J.W. Klimbie and K.L. Koffeman, editors, *Database Management*, pages 385–397. North Holland Publishing Company, 1974.

[18] W. Wright. Some average performance measures for the B-tree. *Acta Informatica*, 16, 1985.

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

[20] B. Zhang and M. Hsu. Unsafe operations in B-trees. to appear in Acta Informatica.

6 Tables and Figures

p	utilization		probability of splitting (merging)	
	analytical	simulation	analytical	simulation
5	47.51%	45.31%	$1.02 \cdot 10^{-2}$	$1.96 \cdot 10^{-2}$
10	42.67	39.48	$2.54 \cdot 10^{-4}$	$1.19 \cdot 10^{-3}$
15	40.87	38.40	$6.40 \cdot 10^{-6}$	$8.33 \cdot 10^{-5}$
20	39.93	39.77	$1.60 \cdot 10^{-7}$	$2.05 \cdot 10^{-5}$

Table 1: Free-at-empty ghost model and simulation for pure modifies: utilization and probabilities of splitting and merging

p	utilization		probability of splitting		probability of deleting	
	analytical	simulation	analytical	simulation	analytical	simulation
5	57.81	54.29	$3.69 \cdot 10^{-2}$	$4.11 \cdot 10^{-2}$	$2.36 \cdot 10^{-3}$	$5.31 \cdot 10^{-3}$
10	62.38	58.34	$1.53 \cdot 10^{-2}$	$1.51 \cdot 10^{-2}$	$1.86 \cdot 10^{-6}$	$6.6 \cdot 10^{-5}$
15	64.36	60.86	$9.74 \cdot 10^{-3}$	$9.34 \cdot 10^{-3}$	$1.95 \cdot 10^{-9}$	0
20	65.36	62.39	$7.13 \cdot 10^{-3}$	$6.82 \cdot 10^{-3}$	$2.44 \cdot 10^{-12}$	0

Table 2: Free-at-empty ghost model and simulation, $q = .45$ (10% more inserts than deletes)

p	utilization		probability of splitting		probability of deleting	
	analytical	simulation	analytical	simulation	analytical	simulation
5	54.88	50.31	$2.63 \cdot 10^{-2}$	$3.05 \cdot 10^{-2}$	$3.86 \cdot 10^{-3}$	$1.02 \cdot 10^{-2}$
10	60.40	53.54	$9.86 \cdot 10^{-3}$	$9.56 \cdot 10^{-3}$	$5.28 \cdot 10^{-6}$	$1.05 \cdot 10^{-4}$
15	63.36	57.17	$6.16 \cdot 10^{-3}$	$5.78 \cdot 10^{-3}$	$5.40 \cdot 10^{-9}$	0
20	64.82	58.94	$4.47 \cdot 10^{-3}$	$4.13 \cdot 10^{-3}$	$9.80 \cdot 10^{-12}$	0

Table 3: Free-at-empty ghost model and simulation, $q = .47$ (6% more inserts than deletes)

p	utilization		probability of splitting (merging)	
	analytical	simulation	analytical	simulation
5	43.02%	45.31%	$4.00 \cdot 10^{-2}$	$1.96 \cdot 10^{-2}$
10	39.34	39.48	$1.00 \cdot 10^{-2}$	$1.19 \cdot 10^{-3}$
15	38.21	38.40	$4.44 \cdot 10^{-3}$	$8.33 \cdot 10^{-5}$
20	37.66	39.77	$2.50 \cdot 10^{-3}$	$2.05 \cdot 10^{-5}$

Table 4: Free-at-empty linear model and simulation for pure modifies: utilization and probabilities of splitting and merging

p		q as percentage						
		5	10	20	30	40	45	47
5	analytical	71.12	70.35	68.06	63.91	56.06	50.14	47.41
5	simulation	71.14	70.34	69.01	66.45	59.75	54.29	50.31
10	analytical	70.10	69.74	68.64	66.54	60.38	52.70	48.02
10	simulation	70.51	70.13	69.20	67.77	63.04	58.34	53.52
20	analytical	69.68	69.51	68.97	67.90	64.70	58.77	53.12
20	simulation	69.85	69.98	69.57	68.03	64.86	62.39	58.94
30	analytical	69.55	69.43	69.08	68.37	66.26	62.07	57.13
30	simulation	69.69	70.05	69.14	68.60	67.03	64.30	-
40	analytical	69.49	69.40	69.14	68.61	67.03	63.87	59.83
40	simulation	69.60	71.49	69.11	68.43	67.53	65.06	-
70	analytical	69.41	69.36	69.21	68.92	68.01	66.21	63.82
100	analytical	69.38	69.34	69.24	69.04	68.41	67.15	65.48

Table 5: Free-at-empty linear model and simulation for varying node sizes and percentages of deletes: utilization

p		q as percentage						
		5	10	20	30	40	45	47
5	ana.	.148	.140	.123	.100	.0721	.0562	.0497
5	sim.	.148	.139	.120	.0926	.0618	.0411	.0305
10	ana.	.0711	.0671	.0576	.0453	.0293	.0197	.0157
10	sim.	.0706	.0668	.0571	.0446	.0271	.0151	.00956
20	ana.	.0349	.0328	.0279	.0216	.0132	.00799	.00572
20	sim.	.0347	.0324	.0277	.0213	.0129	.00682	.00413
30	ana.	.0230	.00217	.0184	.0142	.00853	.00497	.00340
30	sim.	.0230	.0212	.0183	.0139	.00808	.00433	-
40	ana.	.0173	.0162	.0137	.0105	.00629	.00360	.00239
40	sim.	.0178	.0164	.0137	.0105	.00592	.00325	-
70	ana.	.00982	.00922	.00780	.00597	.00353	.00198	.00128
100	ana.	.00686	.00644	.00544	.00416	.00245	.00136	.000869

Table 6: Free-at-empty linear model and simulation for varying node sizes and percentages of deletes: probability of splitting

p	q as percentage						
	5	10	20	30	40	45	47
5 ana.	$3.93 \cdot 10^{-7}$	$7.75 \cdot 10^{-6}$	$1.95 \cdot 10^{-4}$	$1.63 \cdot 10^{-3}$	$9.04 \cdot 10^{-3}$	$1.95 \cdot 10^{-2}$	$2.62 \cdot 10^{-2}$
5 sim.	0	0	0	0	$2.10 \cdot 10^{-3}$	$5.31 \cdot 10^{-3}$	$1.02 \cdot 10^{-2}$
10 ana.	0	0	$5.04 \cdot 10^{-8}$	$6.27 \cdot 10^{-6}$	$3.17 \cdot 10^{-4}$	$1.89 \cdot 10^{-3}$	$3.74 \cdot 10^{-3}$
10 sim.	0	0	0	0	0	$6.60 \cdot 10^{-10}$	$1.05 \cdot 10^{-4}$
20 ana.	0	0	0	$3.44 \cdot 10^{-10}$	$1.45 \cdot 10^{-6}$	$6.74 \cdot 10^{-5}$	$2.98 \cdot 10^{-4}$
20 sim.	0	0	0	0	0	0	0
30 ana.	0	0	0	0	$1.15 \cdot 10^{-8}$	$4.16 \cdot 10^{-6}$	$4.12 \cdot 10^{-5}$
30 sim.	0	0	0	0	0	0	0
40 ana.	0	0	0	0	$1.14 \cdot 10^{-10}$	$3.21 \cdot 10^{-7}$	$7.14 \cdot 10^{-6}$
40 sim.	0	0	0	0	0	0	0
70 ana.	0	0	0	0	0	$2.65 \cdot 10^{-10}$	$6.63 \cdot 10^{-8}$
100 ana.	0	0	0	0	0	0	$9.07 \cdot 10^{-10}$

Table 7: Free-at-empty linear model and simulation for varying node sizes and percentages of deletes: probability of merging

		Utilization						
p		$q = .1$	$q = .2$	$q = .3$	$q = .4$	$q = .45$	$q = .47$	$q = .5$
20	ana.	70.09	70.36	70.48	69.70	67.92	66.43	62.91
20	sim.	70.09	70.31	70.34	70.24	69.24	68.33	66.07
30	ana.	69.98	70.37	70.72	70.41	68.93	67.37	61.43
30	sim.	70.41	70.10	70.54	69.98	69.85	68.98	64.47
40	ana.	69.94	70.38	70.86	70.86	69.66	68.22	60.51
40	sim.	70.05	70.49	70.49	70.63	69.59	69.54	63.30

Table 8: Comparison of analytical and simulation space utilization for merge-at-half B-trees

		Pr[split on insert]						
p		$q = .1$	$q = .2$	$q = .3$	$q = .4$	$q = .45$	$q = .47$	$q = .5$
20	ana.	.0331	.0284	.0221	.0136	.00852	.00669	.00711
20	sim.	.0340	.0321	.0279	.0201	.0144	.0132	.0118
30	ana.	.0218	.0185	.0142	.00843	.00483	.00329	.00226
30	sim.	.0225	.0211	.0178	.0128	.00584	.00491	.00521
40	ana.	.0162	.0138	.0105	.00615	.00344	.00224	.00126
40	sim.	.0170	.0153	.0133	.00939	.00443	.00331	.00226

Table 9: Comparison of analytical and simulation probability of splitting on an insert for merge-at-half B-trees

		Pr[merge on delete]						
p		$q = .1$	$q = .2$	$q = .3$	$q = .4$	$q = .45$	$q = .47$	$q = .5$
20	ana.	.0644	.0571	.0478	.0384	.0391	.0450	.0762
20	sim.	.0691	.0645	.0580	.0494	.0446	.0483	.0572
30	ana.	.0428	.0372	.0299	.0210	.0181	.0193	.0465
30	sim.	.0452	.0432	.0379	.0305	.0193	.0163	.0330
40	ana.	.0320	.0276	.0217	.0143	.0109	.0107	.0324
40	sim.	.0318	.0320	.0283	.0215	.0122	.0110	.0193

Table 10: Comparison of analytical and simulation probability of merging on an delete for merge-at-half B-trees

p		Pr[restructure on operation]						
		$q = .1$	$q = .2$	$q = .3$	$q = .4$	$q = .45$	$q = .47$	$q = .5$
20	half	.0362	.0341	.0298	.0235	.0223	.0247	.0417
20	empty	.0291	.0221	.0149	.00774	.00378	.00223	.00005
30	half	.0261	.0224	.0189	.0135	.0108	.0108	.0244
30	empty	.0190	.0146	.00973	.00513	.00283	.00180	.00000
40	half	.0178	.0166	.0139	.00941	.00680	.00622	.0168
40	empty	.0147	.0109	.00735	.00355	.00178	.00125	.00000

Table 11: Comparison of merge-at-half and free-at-empty probability of restructuring in an operation $(qP_m + (1-q)P_s)$

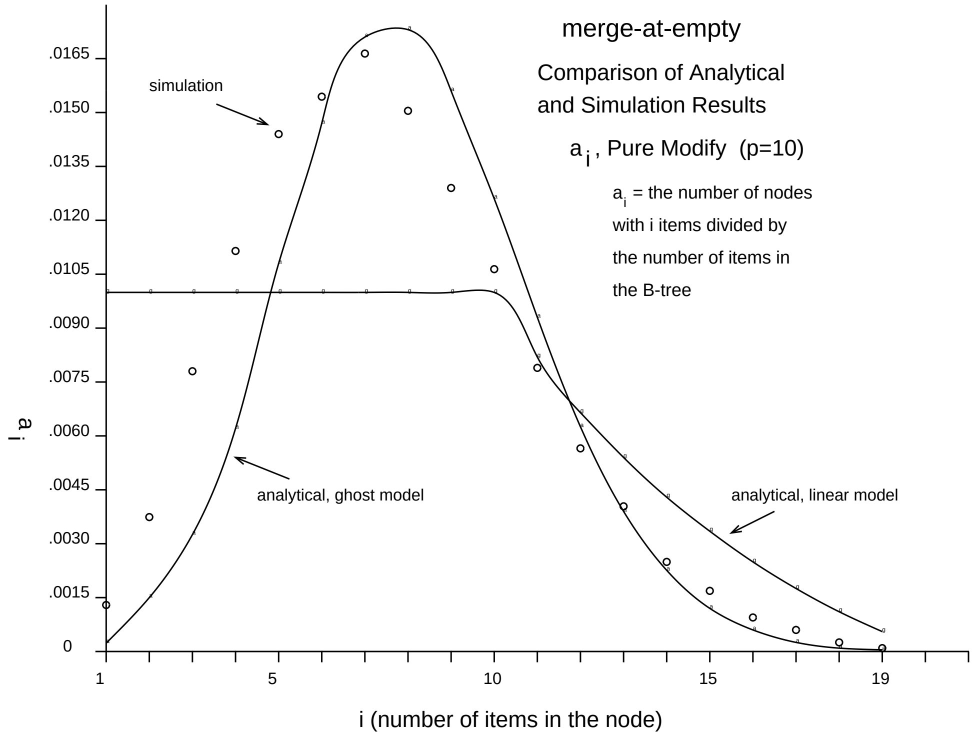

Figure 1:

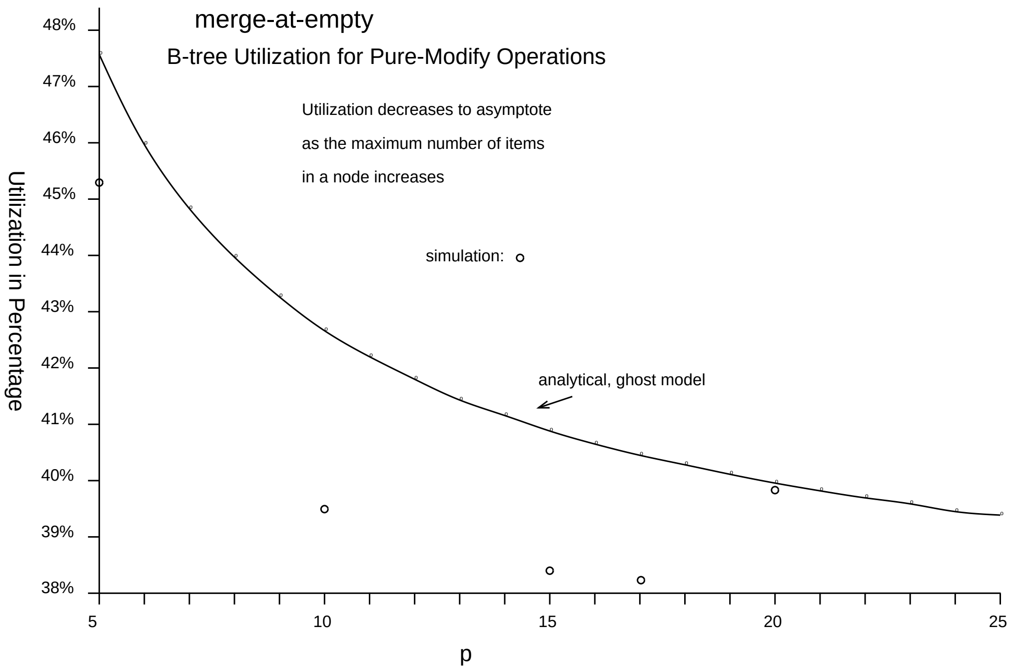

Figure 2:

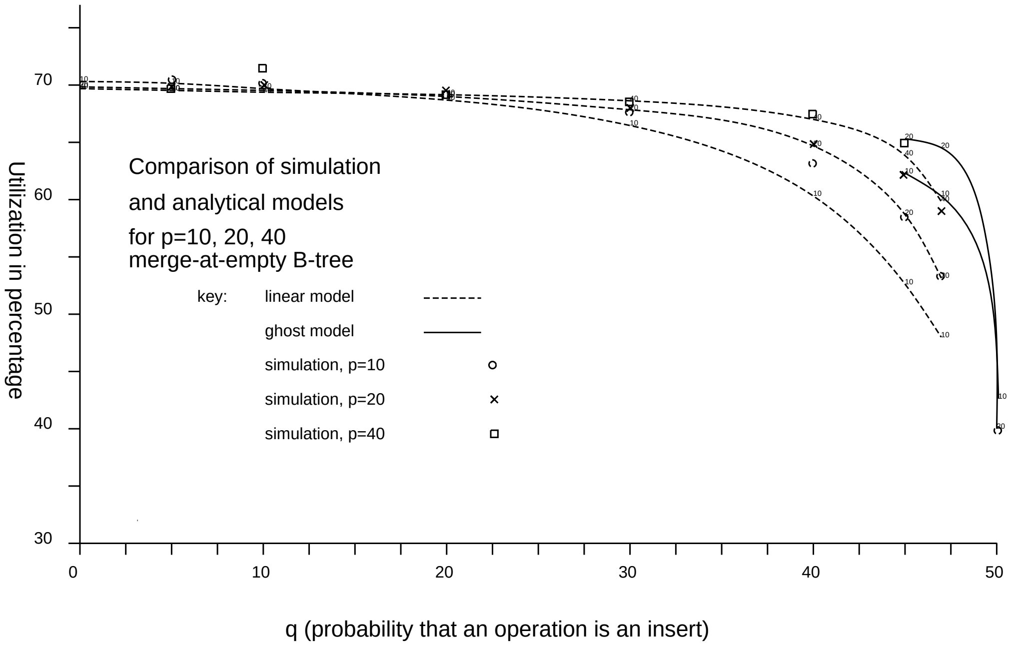

Figure 3:

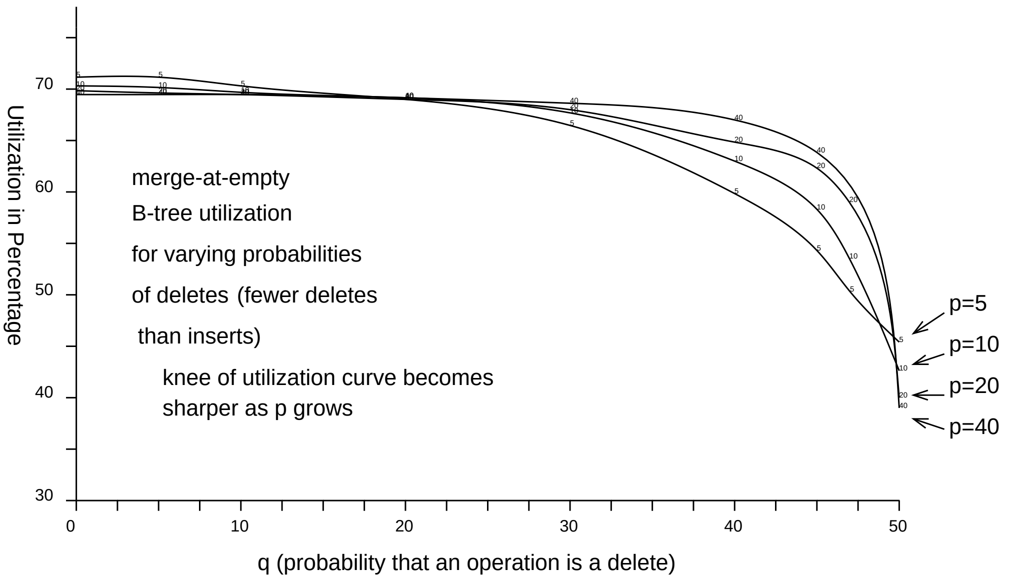

Figure 4: from simulation points

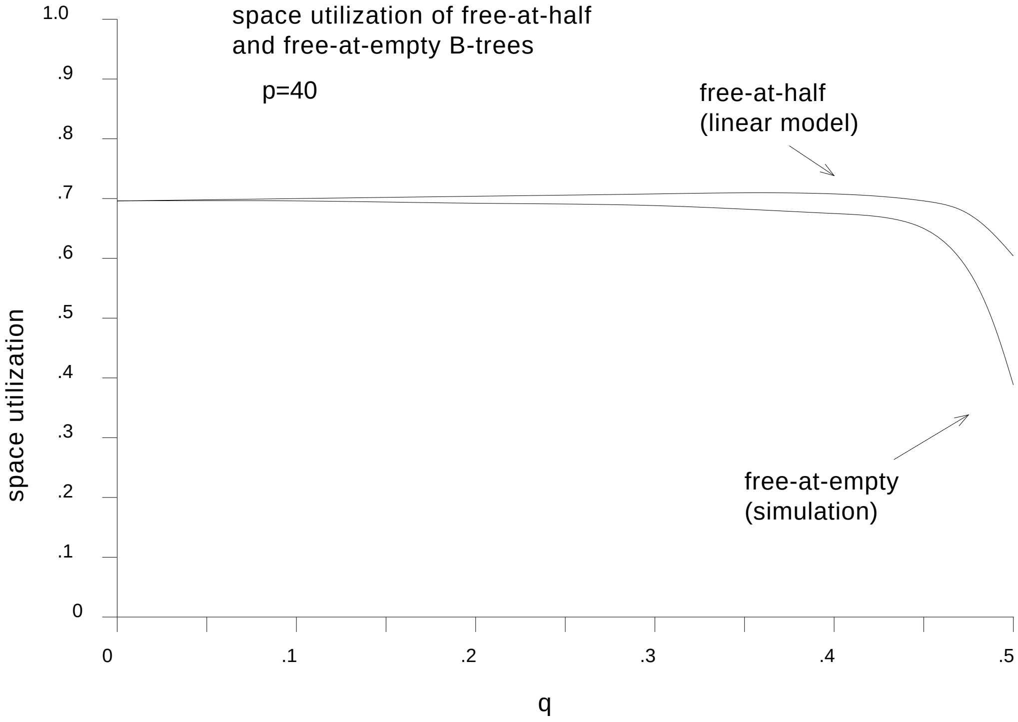

Figure 5:

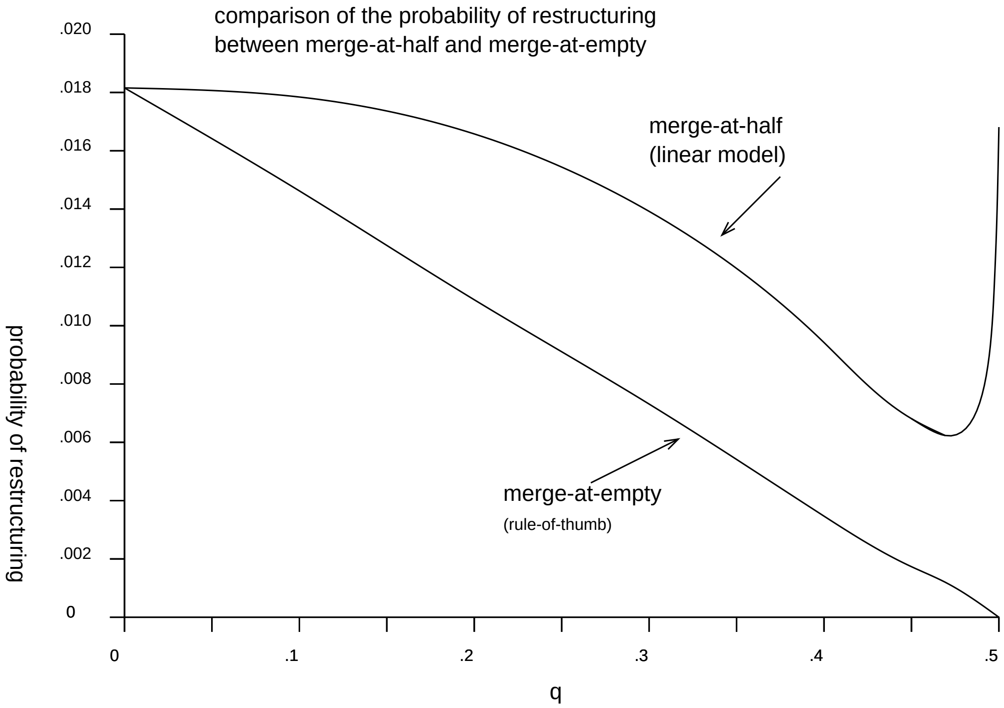

Figure 6:

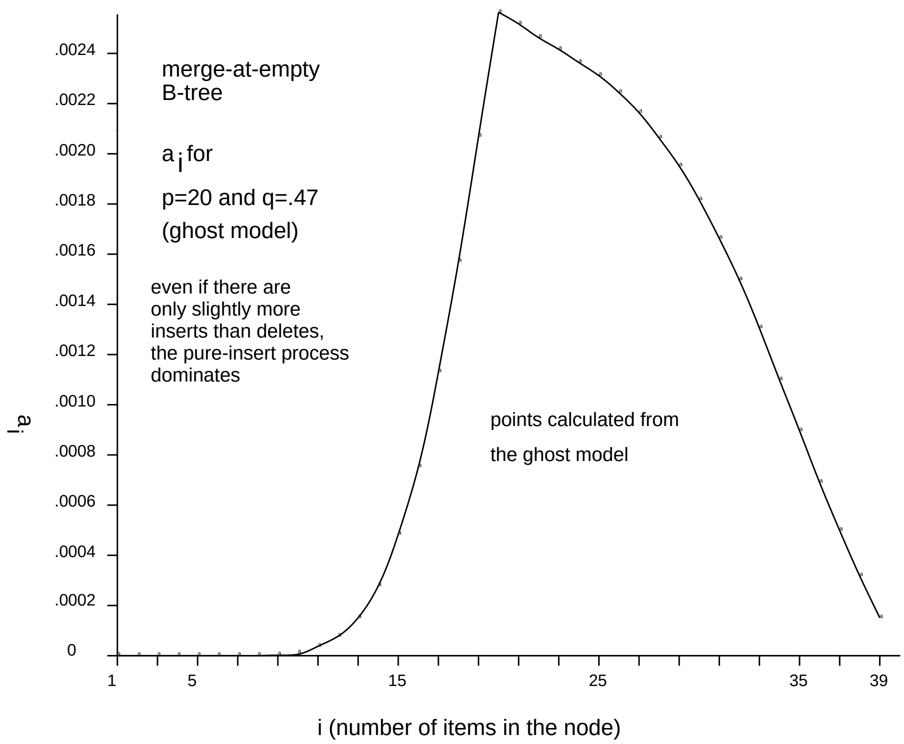

Figure 7:

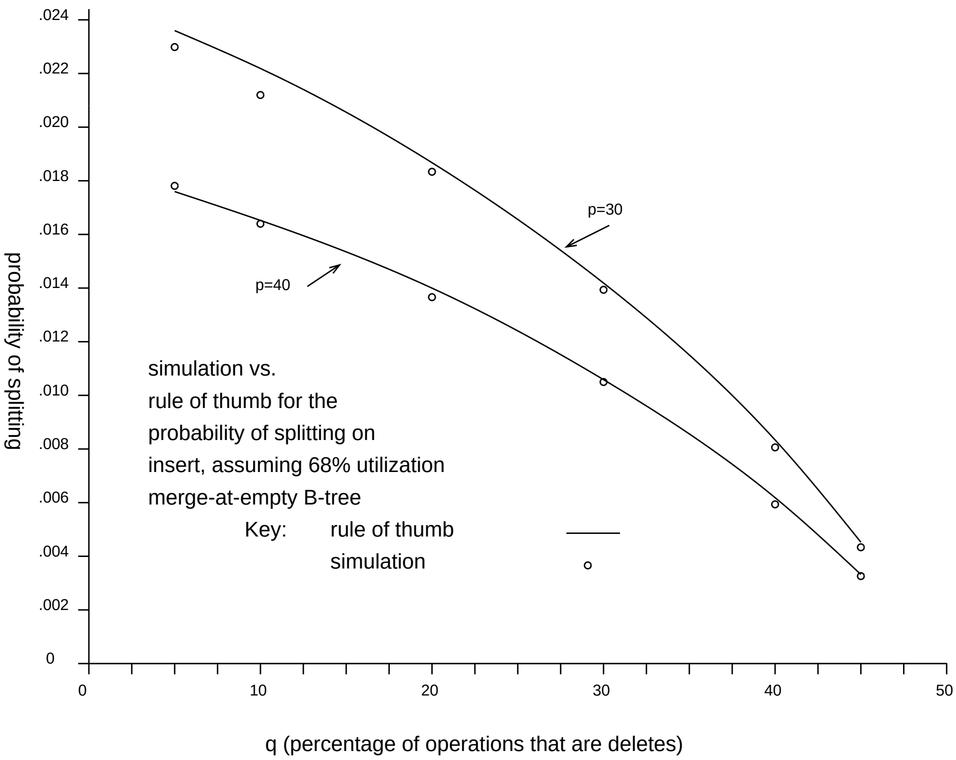

Figure 8: