A Population Analysis for Hierarchical Data Structures

Randal C Nelson
Hanan Samet

Computer Science Department
Center for Automation Research
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742

Abstract

A new method termed population analysis is presented for approximating the distribution of node occupancies in hierarchical data structures which store a variable number of geometric data items per node The basic idea is to describe a dynamic data structure as a set of populations which are permitted to transform into one another according to certain rules The transformation rules are used to obtain a set of equations describing a population distribution which is stable under insertion of additional information into the structure These equations can then be solved, either analytically or numerically, to obtain the population distribution Hierarchical data structures are modeled by letting each population represent the nodes of a given occupancy A detailed analysis of quadtree data structures for storing point data is presented, and the results are compared to experimental data Two phenomena referred to as *aging* and *phasing* are defined and shown to account for the differences between the experimental results and those predicted by the model The population technique is compared with statistical methods of analyzing similar data structures

CR Categories and Subject Descriptors E 1 [**Data**] Data Structures – trees, F 2 2 [**Theory of Computation**] Analysis of nonnumerical algorithms and problems – Geometrical problems and computations, H 3 3 [**Information Storage and Retrieval**] Content Analysis and Indexing – indexing methods

Key words and phrases file structures, bucketing methods, multidimensional attributes, hierarchical data structures, quadtrees

I Introduction

Hierarchical data structures are a class of data structures employing a representational scheme which can be applied at different spatial resolutions to allow the structure to be adapted to the data Examples include quadtree and octree varieties [Same84a], bintrees [Same84c], grid files [Niev84], and also techniques which are less explicitly based on spatial decomposition such as extendible hashing [Fagi79] All, however, are variable resolution representations which have locally similar structure at different resolutions

Because the local resolution of hierarchical data structures depends on the data being represented, performance analysis tends to be difficult Traditional worst-case analysis is often inappropriate because the worst case tends to be both very bad, and highly improbable More useful would be a "typical case" description of properties of interest such as required storage per data item or access time Most approaches to the analysis of hierarchical structures have thus been statistical in nature, most notably, Fagin *et al* in their analysis of extendible hashing [Fagi79] which turns out also to apply to certain types of quadtrees Regnier [Regn85] and Tamminen [Tamm83] have also published statistical analyses of grid files and a structure called EXCELL respectively

A major drawback of statistical analysis for hierarchical systems is that it can be quite complicated to perform, even for relatively simple cases (e g , for uniformly distributed point data [Fagi 79]) The prospect of attempting such an analysis for more complicated data primitives (e g , line segments or polygons) interacting in higher dimensional spaces is somewhat daunting Furthermore, since such analyses depend on some model of data distribution, they will yield at best, approximations to the expected values when real data is used For many applications, a good approximate model may be of as much practical value as an exact statistical computation

This study arose out of our attempt to analyze the storage behavior of certain quadtree data structures which we used in the implementation of a geographic information system [Same85c] A straightforward statistical analysis of these structures promised to be an exceedingly laborious task Since we were interested in the distribution of the nodes as a function of their occupancy, we decided to model a quadtree as a collection of populations where each population represented the nodes in the quadtree having a particular occupancy Thus the set of all empty nodes constitutes one such population, the set of all nodes containing a single data element another, and so forth A certain approximation is involved since the set of all nodes of a given occupancy contains nodes at many different levels in the quadtree Since the

nodes at different levels represent physical blocks of different areas, the population is not, strictly speaking, homogeneous However, because the structure of hierarchical data structures is similar at different resolutions, it was expected that the effect of this approximation would be relatively minor

As information is added to a quadtree with variable capacity nodes, each population grows in a manner which depends on the other populations For instance, consider a structure where nodes can hold up to m points and are split when the capacity is exceeded In this case, the probability of an insertion producing a new node with occupancy i depends both on the fraction of the nodes with occupancy $i-1$ and on the population of full nodes (occupancy m) since nodes of any occupancy less than or equal to m can be produced when a full node splits The basic idea is to determine a steady state, where the proportions of the various populations are constant under addition of new information according to some data model If such a steady state exists, then it can be taken as a representative distribution of populations from which expected values for structure parameters such as average node occupancy can be calculated

We used this approach to analyze quadtree structures for storing both point and line information We present here, an application of the technique to a quadtree structure for storing point data Our analysis of structures for storing line data is somewhat analogous, and is described in detail in [Nels86b]

The remainder of the paper is organized as follows Section II gives a brief overview of quadtrees Section III describes the use of the population model in detail, and illustrates its use by analyzing the PR quadtree for storing points Section IV accounts for the discrepancy in the model in terms of two phenomena termed *aging* and *phasing* which are characteristic of hierarchical data structures in general Section V contains conclusions and a summary of other work

II Quadtrees

The quadtree [Same84a] is a hierarchical, variable resolution data structure based on the recursive partitioning of the plane into quadrants This scheme is useful for representing data having geometric distribution at variable resolution Variations exist for representing planar regions [Klin71], collections of points [Fink74, Same84b], and collections of line segments[Same85b, Nels86a] as well as more complicated objects (e g rectangles) The basic principle generalizes to 3 and higher dimensions (e g , octrees [Hunt78, Jack80, Meag82] and bintrees [Know80, Tamm84, Same85a])

Quadtrees can be divided into two types those based on regular decomposition of space using pre-defined boundaries, and those where the partition is determined explicitly by the data as it is entered into the structure Figure 1 shows a quadtree representation for a set of points based on regular decomposition The region of interest has been recursively partitioned into quadrants until no quadrant contains more than a single point This structure is known as the PR quadtree [Oren82, Same84b] A second decomposition method has been investigated in applications where it is desired to adapt the structure closely to the data (e g , the classical point quadtree [Fink74]) The partitions are typically irregular and of odd sizes, and the shape of the final structure depends critically on the order in which the information was inserted into the tree

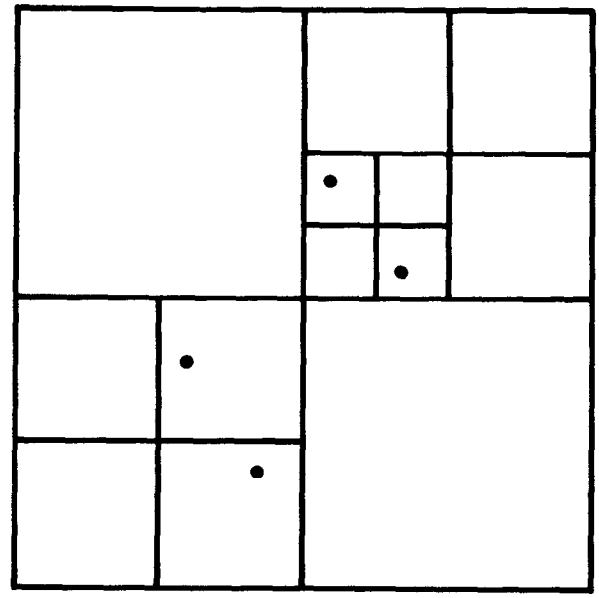

Figure 1 PR quadtree for four points Blocks are recursively quartered until no block contains more than one point

The condition used to determine when a quadtree block should be partitioned is called the *splitting rule* The form of the rule depends on the type of data being stored For instance, if a quadtree is being used to store a collection of of points, one possible rule is "split until no block contains more than one distinct point" This is the basis of the simple PR quadtree The generalized PR quadtree is obtained by permitting the nodes to contain more than one point The rule for the generalized PR quadtree then becomes "split until no block contains more than m points" This principle is similar to that used by Tamminen in his EXCELL system [Tamm81] and by Nievergelt in the grid file [Niev84]

III Computation of the expected distribution

Consider a quadtree data structure whose leaf nodes each contain between 0 and m data items which are members of some set A (e g , the PR quadtree for points) The number of data items stored in a node is the occupancy of the node We can describe the distribution of node occupancies in a particular quadtree Q by a state vector $\vec{d} = (p_0, p_1, \ldots, p_m)$ where p_i is the proportion of the nodes having occupancy i As a consequence of this definition $p_0 + p_1 + \ldots + p_m = 1$ We would like to define an average or typical state vector, say $\vec{e}$, for the data structure which could be used to predict storage properties with some degree of accuracy We will call this vector the expected distribution The following discussion makes specific reference to quadtrees, but the same principles apply in the case of octrees and higher dimensional data structures

A typical statistical approach to defining $\vec{e}$ would be the following Let $\vec{d}_i$ be the average value of the state vector $\vec{d}$ over all quadtrees which represent a subset of A containing i elements Generally, $\vec{d}_i$ is defined relative to some model of

data distribution Define the expected distribution $\vec{e}$ as the limit of the sequence $\vec{d}_1$, $\vec{d}_2$, If $\vec{e}$ exists and can be calculated, then it could serve as a representative distribution for quadtrees of a given type Fagin *et al* have used a statistical approach to calculate average node occupancies as a function of the number of data points in the context of extendible hashing [Fagi79] The application of their results, with slight modifications, to the PR quadtree, indicates that the limit $\vec{e}$ does not exist Specifically, the vector sequence $\vec{d}_1$, $\vec{d}_2$, undergoes oscillatory behavior of increasing period and non-decreasing amplitude We will show in Section IV that this sort of oscillatory behavior, which we refer to as *phasing*, is typical of hierarchical data structures in general when a uniform data distribution is present

The above statistical calculation represents a considerable mathematical effort Furthermore, it cannot be easily generalized to hierarchical representations for other data primitives (e g , line segments) Calculating the vectors $\vec{d}$ directly seems to be difficult in general, especially if the data primitives are non-trivial For these reasons, we decided to pursue an alternative means of modeling the performance of a quatree

We model a quadtree as a set of populations of nodes where each population consists of all nodes having a given occupancy Thus empty nodes form one population, nodes containing one point a second, and so forth Insertion of a point into a node of occupancy i either transforms it into a node with occupancy $i+1$ or else causes the node to split, increasing several populations The expected distribution $\vec{e}$ is defined by the condition that the proportion of each population making up the structure remains unchanged if data is added to the tree in accordance with statistical expectations, i e , $\vec{e}$ is a fixed point under the operation of insertion This condition can be used to determine $\vec{e}$ if the data distribution and the statistical results of adding a datum to a node can be calculated The key difference between our method and the statistical approach of Fagin *et al* is that our method considers only the local probabilities for the distribution of data items in a single node into its quadrants rather than a distribution for the whole population It is more tractable than the direct statistical approach, and yields results which are in reasonable agreement with experimental data for several quadtree data structures We illustrate the use of this technique by means of an example, and then summarize the result of its application to the generalized PR quadtree

Consider the simple PR quadtree described above Every node contains either zero or one data points There are thus two distinct populations Let us refer to these as types n_0 and n_1 respectively If a point is added to a node of type n_0, that node is transformed into a single node of type n_1 On the other hand, if a point is added to a node of type n_1, then the block must be split, perhaps several times, until the two points lie in separate blocks In this case, several nodes, of both types, are generated

For any node type, the average result of adding a point to the node can be described by a transform vector $\vec{t}=(t_0,t_1)$ where t_0 is the average number of nodes of type n_0 produced by the insertion of a point, and t_1 is the average number of n_1 nodes Let $\vec{t}_i$ be the transform vector describing the results of adding a point to a node of type n_i The vectors $\vec{t}_i$ form the row of a matrix $\mathbf{T}$ called the transform matrix From the preceding discussion $\vec{t}_0 = (0,1)$ To determine $\vec{t}_1$, we use the geometry of the situation to write a recurrence relation If the distribution of data points is uniform, then in 3/4 of the cases, a single split will suffice, dividing the block into four quadrants, two of which are empty, and two of which contain a single point In one quarter of the cases, both points will end up in the same quadrant which must be split again under the same conditions as the original split This allows us to write the recurrence relation

$$\vec{t}_1 = \frac{3}{4}(2,2) + \frac{1}{4}(3,0) + \frac{1}{4}\vec{t}_1$$

Solving this vector equation for $\vec{t}_1$ gives $\vec{t}_1 = (3,2)$

If we now assume, that the probability of a data point being inserted into a node of a given occupancy is proportional to the numerical fraction of nodes of that type in the tree, then we can use the above results to write equations which $\vec{e}$ must satisfy Note that this assumption is equivalent to the assumption that the distribution of node occupancies is independent of the geometric size of the corresponding block i e , that nodes at depth n in the tree do not have different occupancy distributions than those at depth $n+1$ It turns out that this is not strictly true in real quadtrees – larger nodes tend to have slightly higher average occupancies We refer to this phenomenon as *aging*, and will examine it in more detail later However, for PR quadtrees, the approximation is close enough to be useful

Under the above assumption, insertion of new data points into a PR quadtree transforms nodes of types n_0 and n_1 with relative frequency e_0 and e_1 respectively (recall that $\vec{e}$ is the expected distribution) Because nodes are transformed in direct proportion to their abundance, the distribution of untransformed nodes is always $\vec{e}$ Thus in order for $\vec{e}$ to be a fixed point, the distribution of node types produced from the transformed nodes must also be $\vec{e}$ This allows us to formulate equations which can be solved for $\vec{e}$ as follows Suppose that the number of data points is increased by Δn In this case, the expected number of new nodes of each type can be calculated from the distribution of node types in the tree (assumed to be $\vec{e}$) and the transform vectors (represented by the matrix $\mathbf{T}$) Specifically, the expected number of new n_0 nodes is $\mathbf{T}_{00}e_0\Delta n + \mathbf{T}_{10}e_1\Delta n$, and the expected number of new n_1 nodes is $\mathbf{T}_{01}e_0\Delta n + \mathbf{T}_{11}e_1\Delta n$ The expected total number of new nodes is the sum of these two quantities Requiring the proportion of new n_0 nodes to be e_0 gives

$$\frac{\mathbf{T}_{00}e_0+\mathbf{T}_{10}e_1}{\mathbf{T}_{00}e_0+\mathbf{T}_{10}e_1+\mathbf{T}_{01}e_0+\mathbf{T}_{11}e_1} = e_0$$

Similarly, requiring the proportion of new n_1 nodes to be e_1 gives

$$\frac{\mathbf{T}_{01}e_0+\mathbf{T}_{11}e_1}{\mathbf{T}_{00}e_0+\mathbf{T}_{10}e_1+\mathbf{T}_{01}e_0+\mathbf{T}_{11}e_1} = e_1$$

These equations can be concisely expressed in matrix notation by the formula

$$\vec{e}\mathbf{T} = a\vec{e} \qquad (1)$$

where a is the scalar $\mathbf{T}_{00}e_0+\mathbf{T}_{10}e_1+\mathbf{T}_{01}e_0+\mathbf{T}_{11}e_1$ Note that since a is a function of $\vec{e}$, (1) does not represent a set of linear equations, but rather a set of quadratic equations involving the components of $\vec{e}$ This particular example can be solved analytically to yield $\vec{e}=(1/2,1/2)$, the only positive solution Thus a PR quadtree with a maximum of one point

per node, should have approximately equal numbers of full and empty nodes This agrees fairly well with experiments in storing random points in PR quadtrees where we found approximately 53% empty and 47% full nodes The causes of the slight discrepancy are examined later

The above technique can be applied to the generalized PR quadtree where a node may contain up to m points The integer m is known as the node capacity The expected distribution $\vec{e}$ has $m+1$ components, there are $m+1$ node types n_0 through n_m, and there are $m+1$ transform vectors $\vec{t}_0$ through $\vec{t}_m$ which form the rows of the $m+1 \times m+1$ transform matrix $\mathbf{T}^m$ The transform vectors $\vec{t}_0$ through $\vec{t}_{m-1}$ are simple because the new data point is just added to the node without causing a split, so that a node of type n_i becomes a node of type n_{i+1} The vectors $\vec{t}_0$ through $\vec{t}_{m-1}$ thus have the form

$$\vec{t}_i = (0, \ ,0,1,0, \ ,0)$$

where there are $m+1$ components and 1 is in the $i+1^{st}$ position

The vector $\vec{t}_m$ which represents the splitting of a node into four quarters when its capacity is exceeded is found by determining the expected distribution of the points into the quadrants, and using this information to write a recurrence relation for $\vec{t}_m$ as illustrated above in the case of the PR quadtree for $m=1$

In general, the expected distribution of $m+1$ points into quadrants is just the binomial distribution for $m+1$ objects placed independently into four buckets The expected number of buckets containing i items, P_i, is thus given by

$$P_i = \binom{m+1}{i} \frac{3^{m+1-i}}{4^m}$$

The term P_{m+1} is equal to 4^{-m}, and represents the case where all $m+1$ points end up in the same quadrant so that recursive splitting occurs A recursive relation for $\vec{t}_m$ can thus be written

$$\vec{t}_m = (P_0, P_1, \ , P_m) + P_{m+1}\vec{t}_m$$

This can be solved to obtain the following expression for the components $\mathbf{T}_{mi}$ of $\vec{t}_m$

$$\mathbf{T}_{mi} = \binom{m+1}{i} \frac{3^{m+1-i}}{4^m - 1}$$

Note that for m larger than three or four, the probability of splitting more than once is negligible, and $\mathbf{T}_{mi}$ is closely approximated by P_i

As in the $m=1$ case, a set of equations for the components of $\vec{e}$ can be expressed in terms of the transform matrix $\mathbf{T}$

$$\vec{e}\mathbf{T} = a\vec{e}$$

The scalar a is the normalizing factor in computing the proportions and is given by

$$a = \sum_{i=0}^{m} \sum_{j=0}^{m} \mathbf{T}_{ij} e_i$$

that is, the sum of the coefficients of $\vec{e}$ multiplied by the sum of the components of the transform matrix in the corresponding row The row sums represent the number of nodes produced by a node of type n_i upon absorbing an additional point, and thus all equal unity with the exception of row m whose sum is $(4^{m+1}-1)/(4^m-1)$ which is slightly greater than four Thus a can be written

$$a = e_0 + e_1 + \ + \frac{4^{m+1}-1}{4^m-1} e_m$$

The matrix equation represents a set of $m+1$ quadratic equations for the components of $\vec{e}$ In general, such a set of equations can have up to 2^{m+1} solution vectors, however, since the components of $\vec{e}$ represent proportions, we are interested only in solutions for which all the components are positive It can be shown, that for sets of equations of the above form, at most one positive solution is possible (see [Nels86b]) We are thus free to solve the equations numerically, with the assurance that any positive solution we find will be appropriate

The equations were set up and solved for PR quadtrees with node capacities m ranging between one and eight points For each node capacity $1 \leq m \leq 8$, the transform matrix $\mathbf{T}$ was used to obtain a system of equations describing the expected distribution $\vec{e}$ The systems were solved numerically using an iterative technique which converged on the positive solution Experimental data was collected by constructing ten quadtrees of 1000 random points for each case and averaging the results Corresponding data points from different trees were typically within about 10% of each other The theoretical and experimental values obtained are compared in Table 1

Experiment and theory agree fairly well as to the general form of the expected distribution $\vec{e}$ Both show, for all node capacities m, a distribution which has a small value for low occupancies, rises to a peak, and decreases again for high occupancies

A quantity, which conveniently summarizes the information contained in $\vec{e}$ for many practical applications, is the average node occupancy This value is calculated from $\vec{e}$ by adding e_1 to twice e_2 and so forth, i e , the dot product of $\vec{e}$ with the vector $(0,1,2, \ ,m)$ A good general idea of the accuracy of the theoretical model is obtained by comparing, for each m, the average node occupancy predicted by the model with that observed in the experiments These values are tabulated in Table 2 The agreement is clearly not exact, but it is close enough to be useful, and to establish the utility of the underlying model

Two basic trends in Table 2 are noteworthy First, the theoretical occupancy predictions are slightly, but uniformly higher than the experimental values Second, the size of the discrepancy seems to have a cyclical structure This behavior is primarily due to two phenomena, exhibited by hierarchical data structures under certain circumstances, which are not taken into account in the rather simple model derived above The explanation of these phenomena is the subject of the next section

IV. Sources of discrepancy aging and phasing

We will first examine a phenomenon referred to as *aging* which accounts for the consistent over-estimation of the average node occupancy by the theoretical model Recall that in the derivation of the model, an assumption was made that the probability of a point, randomly selected from a uniform distribution, falling into a node of type n_i (occupancy i) was proportional to the fraction of total nodes which were of type n_i Since the probability of a point falling into a node of type n_i is actually proportional to the fraction of the total area occupied by nodes of type n_i, this was equivalent to the assumption that the distribution of types in the population of nodes having area $(1/2^{2i})$ was independent of i The fact

Table 1
Expected distribution in PR quadtrees
theoretical (thy) and experimental (exp)

Bucket size	Expected distribution vector
1	thy (500 , 500) exp (536 , 464)
2	thy (278 , 418 , 304) exp (326 , 427 , 247)
3	thy (165 , 320 , 305 , 210) exp (213 , 364 , 273 , 149)
4	thy (102 , 239 , 276 , 225 , 158) exp (139 , 293 , 264 , 184 , 120)
5	thy (065 , 179 , 238 , 220 , 172 , 126) exp (084 , 217 , 241 , 204 , 151 , 104)
6	thy (043 , 132 , 200 , 207 , 176 , 137 , 105) exp (050 , 150 , 201 , 215 , 176 , 127 , 081)
7	thy (028 , 098 , 165 , 189 , 173 , 143 , 114 , 090) exp (034 , 110 , 177 , 214 , 187 , 143 , 091 , 044)
8	thy (019 , 073 , 135 , 168 , 166 , 145 , 119 , 097 , 078) exp (024 , 086 , 151 , 206 , 194 , 156 , 100 , 049 , 034)

Table 2
Average Node Occupancy

node capacity	experimental occupancy	theoretical occupancy	percent difference
1	0 46	0 50	7 2
2	0 92	1 03	10 8
3	1 36	1 56	12 9
4	1 85	2 10	11 6
5	2 44	2 63	7 4
6	3 03	3 17	4 4
7	3 44	3 72	7 5
8	3 79	4 25	10 8

Table 3
Occupancy by node size

Depth	n_0 nodes	n_1 nodes	occupancy
4	6 6	20 1	75
5	300 2	354 3	54
6	533 7	411 6	44
7	225 4	144 9	39
8	71 5	49 6	41
9	16 1	19 5	55

that this assumption does not hold exactly is the reason that the theoretical values tend to be uniformly higher than the experimental ones In particular, nodes having greater area will, on the average, tend to have a higher occupancy

The higher occupancy of large nodes can be understood in two ways If the quadtree is viewed as a static structure where a set of points is given and splitting of quadrants takes place until no block contains more than m points, then for a random, uniform distribution, the bigger nodes will tend to be better filled simply because their area is larger and the point density is (more or less) uniform This occurs in spite of the fact that the splitting is adaptive

Another way of looking at the same phenomenon is to consider the quadtree as a dynamic structure which has reached its present state through a history of point insertions A particular leaf node can be considered to have a lifetime extending from the time it is created by the splitting of its parent, until it absorbs enough points to fill it to capacity and is itself split This "filling up" process will be referred to as *aging* Clearly, as a population of nodes of a given size ages, the average occupancy increases Consider a time scale defined by the rate of insertion of points, where each point insertion represents a tick of the clock On such a scale, if the points are drawn from a uniform distribution, large nodes will, on the average, age faster than small ones since their area is greater and more points will be inserted into them in a given interval of time Now, consider a quadtree which has populations of nodes of two or more sizes Since nodes at any level are created by the splitting of full nodes in the previous generation, the average occupancy of a hypothetical age-zero population of nodes is the same for every generation Since large nodes are formed before small ones, the average number of clock ticks which have passed since the creation of the node is greater for large nodes than for small ones Combined with the fact that large nodes fill up faster than small ones, this implies that the effects of aging (i e , increased occupancy) are always more pronounced in the large node population Thus the average occupancy of large nodes would be expected to be higher

Table 3 demonstrates that the relative, average occupancy of nodes does indeed decrease with block size The data represent averages over 10 PR quadtrees of 1000 points with $m=1$ Block area is proportional to 4^{-depth}, hence the large nodes appear first in the table The general tendency is for occupancy to decrease with node size towards the expected value for a population created by splitting a set of full nodes This value is given by the dot product $\vec{t}_m \cdot (0,1, \ldots, m-1, m)$ which is 40 for $m=1$ The experimental data shows the predicted decrease towards this value which is reached at depths 7 and 8 The anomalously high value for the average occupancy at the deepest level (depth = 9) is an artifact of the implementation which truncates the tree at that depth However, since there are only a few nodes at the maximum depth, the net effect of this perturbation on the experimental data is negligible

We can now describe, at least qualitatively, the correction which must be applied to the population model to account for the effects of aging If larger nodes have a higher average occupancy, then conversely, nodes with higher occupancies tend to have larger average sizes Since nodes of higher occupancy are larger, they are individually more likely to be encountered when a point is inserted Thus to maintain a steady state, the fraction of high-occupancy nodes must be less than predicted by the original model, which assumed the average sizes of the nodes to be independent of the occupancy Conversely, the fraction of low occupancy nodes must be higher Examination of the occupancy data indicates that this correction is consistent with the observed discrepancy between the theoretical and experimental values The effect of the correction on the on the modeled average occupancy would be to decrease it, which is also consistent with the observed discrepancy

A second effect referred to as *phasing* is responsible for the periodic behavior of the discrepancy between the observed and theoretical average occupancies considered as a function of node capacity m This effect is due to the fact that for a uniform distribution of points, the nodes in the quadtree tend to stay about the same size Moreover, all nodes of the same size tend to fill up and split at about the same time As a quadtree is built, there will be cycles of relative activity as the uniform point density approaches a value at which nodes of a certain size split, followed by periods of inactivity as the new, smaller nodes fill up Thus, when the points in the quadtree are drawn from a uniform distribution, the nodes will tend to split and fill in phase This activity will follow a cycle with a logarithmically increasing period which repeats every time the number of points increases by a factor of four The average occupancy will follow a similar cycle, attaining its highest value just before a group of uniformly sized nodes begins to split up, and its lowest value just after most of the nodes have been split This effect becomes more pronounced as the node capacity increases since the probability of having a local density fluctuation which would require splitting at more than one level decreases with increasing m Because the distribution of node sizes depends only on the statistical fluctuations in point density which is scale invariant for a uniform distribution, the oscillations will not damp out Thus the limit of the sequence $\vec{d}_1, \vec{d}_2,$ mentioned in section II does not exist

Table 4 illustrates the cyclical variation of the average occupancy with the number of points for a node capacity of 8 The data were generated by averaging the results from 10 quadtrees built from the specified number of points drawn from a uniform distribution The sample sizes were chosen along a logarithmic scale so that the number of points in the samples quadruples over four steps Note that relative maxima and minima are separated by factors of four (four steps) as hypothesized The data is shown plotted on a semi-log scale in Figure 2 which illustrates the cyclical behavior more clearly

For different node capacities, the relative maxima and minima of the average occupancy occur at different numbers of data points Thus when the size of the data sample is fixed and the node capacity is allowed to vary, the average occupancy will be observed at different points along the cyclical curve Since the prediction of the population model is independent of the number of data points, the discrepancy between the observed and predicted values would be expected to vary cyclicly with node capacity if the data is gathered for a fixed number of points The smooth oscillation in the percent difference between theoretical and experimental results in Table 2 represents approximately such cycle

The observed oscillatory behavior is due to semi-synchronous splitting of nodes of a given generation when the density of a uniform distribution of points reaches a certain value If a non-uniform distribution were used, the effect would be expected to disappear Table 5, and Figure 3 show the results of the same experiment using a Gaussian distribution of points two standard deviations wide centered in the square region covered by the quadtree Oscillatory behavior is observed while the number of points is relatively small, but in this case it damps out as node populations in regions of different densities get out of phase For the case of the Gaussian, a small residual oscillation might be expected because of the large central area of near constant density

The cyclicity observed in our results is the same effect predicted by Fagin *et al* [Fagi79] in their analysis of extendible hashing, where it appears as higher terms in a Fourier

Table 4
Variation of occupancy with tree size
averages for 10 trees

points	nodes	occupancy
64	16 9	3 79
90	21 7	4 15
128	35 2	3 64
181	54 4	3 33
256	67 3	3 80
362	90 7	3 99
512	145 0	3 53
724	216 4	3 35
1024	266 5	3 84
1448	350 8	4 13
2048	560 5	3 65
2896	876 6	3 30
4096	1075 6	3 81

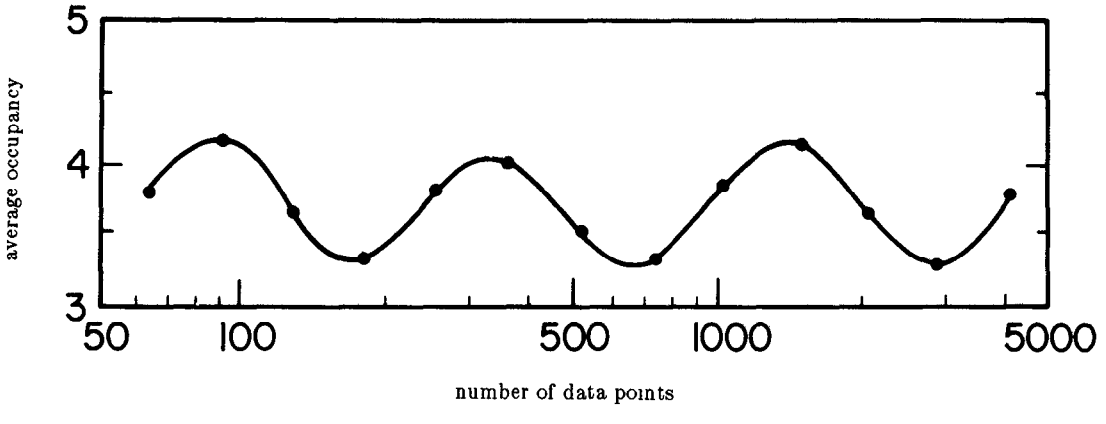

Figure 2 Experimental results and interpolated curve showing variation of average node occupancy with number of data points for uniform distribution of data

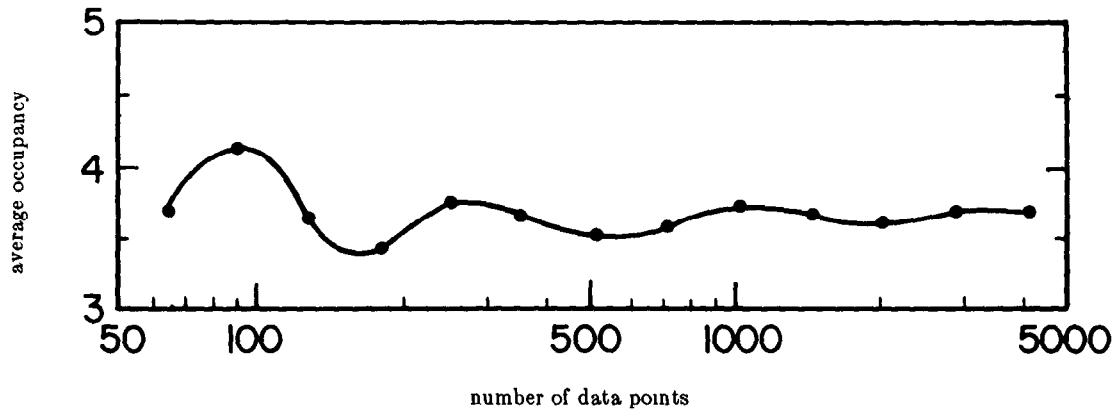

Figure 3 Experimental results and interpolated curve showing variation of average node occupancy with number of data points for Gaussian distribution of data

Table 5
Variation of occupancy with tree size
Gaussian distribution

points	nodes	occupancy
64	17 2	3 72
90	21 7	4 15
128	35 2	3 63
181	52 3	3 46
256	68 2	3 75
362	99 1	3 65
512	144 1	3 55
724	203 5	3 56
1024	275 5	3 72
1448	393 4	3 68
2048	565 3	3 62
2896	784 9	3 69
4096	1104 7	3 71

series expansion Our discussion indicates that such cyclical behavior will tend to appear in data structures based on regular spatial decomposition whenever a data model assuming uniform distribution is incorporated

V. Conclusions

We have presented a method for analyzing hierarchical data structures based on a model of such structures as populations of nodes of different occupancies The model allows the analysis of such data structures without laborious statistical derivations Only the probabilities of the local interaction of the data primitive with the quadrants of a node need be evaluated The model represents a formalization of the intuitive notion of a "typical case" By investigating the sources of discrepancy with experimental data, two phenomena which are characteristic of hierarchical data structures were identified aging and phasing Aging is responsible for larger nodes having higher than average occupancy even when splitting is adaptive Phasing causes a cyclical variation in the average occupancy of nodes which is periodic in the logarithm of total number of data items stored in the structure, particularly when the distribution of data is uniform

We have applied a similar population analysis to a quadtree line representation called the PMR quadtree [Nels86a] The adaptation is relatively simple, and yields results which agree with experimental data even better than in the case of the PR quadtree This is encouraging, particularly since straightforward statistical analysis of quadtree line representations appears to be much more difficult than the corresponding analysis of point structures, and has not, to our knowledge, been performed Details of this analysis are contained in [Nels86b]

Acknowledgements This work was supported in part by the National Science Foundation under Grant DCR-86-05557

References

[Fagi79] - R Fagin, J Nievergelt, N Pippenger, H R Strong, Extendible hashing – a fast access method for dynamic files, *ACM Transactions on Database Systems 4*, 3(September 1979), 315-344

[Fink74] - R A Finkel and J L Bentley, Quad trees a data structure for retrieval on composite keys, *Acta Informatica 4*, 1(1974), 1-9

[Hunt78] - G M Hunter, Efficient computation and data structures for graphics, Ph D dissertation, Department of Electrical Engineering and Computer Science, Princeton University, Princeton, NJ, 1978

[Jack80] - C L Jackins and S L Tanimoto, Oct-trees and their use in representing three-dimensional objects, *Computer Graphics and Image Processing 14*, 3(November 1980), 249-270

[Klin71] - A Klinger, Patterns and search statistics, in *Optimizing Methods in Statistics*, J S Rustagi, Ed , Academic Press, New York, 1971, 303-337

[Know80] - K Knowlton, Progressive transmission of grey-scale and binary pictures by simple, efficient, and lossless encoding schemes, *Proceedings of the IEEE 68*, 7(July 1980), 885-896

[Meag82] - D Meagher, Geometric modeling using octree encoding, *Computer Graphics and Image Processing 19*, 2(June 1982), 129-147

[Nels86a] - R C Nelson and H Samet, A consistent hierarchical representation for vector data, *ACM SIGGRAPH '86*, Dallas, August, 197-206

[Nels86b] - R C Nelson and H Samet, Population analysis of quadtrees with variable node size, Computer Science TR-1740, University of Maryland, College Park, MD, December 1986

[Niev84] - J Nievergelt, H Hinterburger, and K C Sevcik, The grid file an adaptable symmetric multi-key file structure, *ACM Transactions On Database Systems 9*, 1(March 1984), 38-71

[Oren82] - J A Orenstein, Multidimensional tries used for associative searching, *Information Processing Letters 14*, 4(June 1982), 150-157

[Regn85] - M Regnier Analysis of Grid File algorithms, *BIT 25*, 2(1985), 335-357

[Same84a] - H Samet, The quadtree and related hierarchical data structures, *ACM Computing Surveys 16*, 2(June 1984), 187-260

[Same84b] - H Samet, A Rosenfeld, C A Shaffer, R C Nelson, and Y-G Huang, Application of hierarchical data structures to geographic information systems phase III, Computer Science TR-1457, University of Maryland, College Park, MD, November 1984

[Same84c] - H Samet and M Tamminen, Efficient image component labeling, Computer Science TR-1420, University of Maryland, College Park, MD, July 1984

[Same85a] - H Samet and M Tamminen, Efficient component labeling of images of arbitrary dimension, Computer Science TR-1480, University of Maryland, College Park, MD, February 1985

[Same85b] - H Samet and R E Webber, Storing a collection of polygons using quadtrees, *ACM Transactions on Graphics 4*, 3(July 1985), 182-222

[Same85c] - H Samet, A Rosenfeld, C A Shaffer, R C Nelson, Y-G Huang, and K Fujimura, Application of hierarchical data structures to geographic information systems phase IV, Computer Science TR-1578, University of Maryland, College Park, MD, December 1985

[Tamm81] - M Tamminen, The EXCELL method for efficient geometric access to data, *Acta Polytechnica Scandinavica*, Mathematics and Computer Science Series No 34, Helsinki, 1981

[Tamm83] - M Tamminen, Performance analysis of cell based geometric file organizations, *Computer Vision, Graphics, and Image Processing 24*, 2(November 1983), 168-181

[Tamm84] - M Tamminen, Comment on quad- and octtrees, *Communications of the ACM 27*, 3(March 1984), 248-249