ELSEVIER

Data & Knowledge Engineering 22 (1997) 309–318

DATA & KNOWLEDGE ENGINEERING

On sampling regional data[1,2]

Michael Vassilakopoulos[a], Yannis Manolopoulos[b,*]

[a]*Dept. of Applied Informatics, Univ. of Macedonia, GR-540 06 Thessaloniki, Greece*
[b]*Dept. of Informatics, Aristotelian Univ. of Thessaloniki, GR-540 06 Thessaloniki, Greece*

Received 6 February 1996; revised 30 August 1996; accepted 7 November 1996

Abstract

The region quadtree is a very popular hierarchical data structure for the representation of binary images (regional data) and it is heavily used at the physical level of many spatial databases. Random sampling algorithms obtain approximate answers of aggregate queries on these databases efficiently. In the present report, we examine how four different sampling methods are applied to specific quadtree implementations (to the most widely used linear implementations). In addition, we examine how two probabilistic models (a parametric model of random images and a model of random trees) can be used for analysing the cost of these methods.

Keywords: Regional data; Spatial databases; Linear quadtrees; Sampling algorithms; Performance analysis

1. Introduction

Statistical analyses are frequently based on large databases for scientific and decision-support purposes. These analyses are connected to aggregate queries: queries which require accessing a large number of records stored in the database (e.g. the calculation of the average, maximum or minimum value of a certain field present in all the records of the database). Often, approximate answers to such queries are sufficient and exact answers are practically impossible (due to time and storage-space limitations). Random sampling offers an efficient method for obtaining such approximate answers [3]; that is, we choose at random a limited number of records of the database and using these records we try to answer the aggregate query with the highest possible precision. Random sampling is performed through the structure that supports the database. In terms of conventional databases, sampling methods have been developed for hash files, grid files, B^+ trees and ranked B^+ trees (a survey of these

*Corresponding author; e-mail: manolopo@eng.auth.gr

[1]Part of this paper appears in preliminary form in the *Proceedings of 2nd Hellenic–European Research on Mathematics and Informatics (HERMIS) International Conference*, Vol. 2, pp. 651–658, Athens, September 1994.

[2]Work partially funded by the Greek General Secretariat of Research through the national program PENED95.

PII S0169-023X(96)00047-X

methods appears in [4]). Random sampling reduces the cost of accessing the database and the cost of using the retrieved data for performing computations or physical inspections. Sampling may even help in the estimation of various database parameters that are used by a query optimizer while choosing an evaluation plan. In spatial databases [2], sampling becomes even more important since these databases are typically very large and they are heavily used for analysis purposes.

In this report, we focus on a certain kind of spatial data called 2-D Regional Data, or, in other words, binary images (black regions on a white background). In this context, a random sampling query in a spatial database [4] is of the form: given a target region, generate a random sample of points uniformly distributed over this region. These points are sampled from the data structure that supports the spatial database. In [4], various sampling methods are presented for region quadtrees and R trees.

The Region quadtree [8] (or simply quadtree) is a very popular hierarchical data structure for the representation of binary images and it is heavily used at the physical level of spatial databases [11]. We can view such an image as a $2^n \times 2^n$ binary array, for some natural number n, where an entry equal to 0 stands for a white pixel and an entry equal to 1 stands for a black pixel. If every pixel of this image is white (black), its quadtree is made up of a single white (black) node. If, however, this image is not unicolour, its quadtree is made up of a gray root, which has four children sub-quadtrees, one for every quadrant of the image. We assume here that these children correspond, from left to right, to the North-West, North-East South-West and South-East quadrant of this image, respectively. An example of an 8×8 image and its quadtree are shown in Fig. 1a and Fig. 1c, respectively. Note that black (white) squares represent black (white) leaves, while circles represent gray nodes. The unicolour blocks, to which this image is partitioned by the quadtree external nodes, are depicted in Fig. 1b. Sampling in a quadtree assumes that the target region consists of all the quadrangular black blocks (black leaves) to which the quadtree partitions the image. Note that these blocks are disjoint.

Four different methods of sampling in quadtrees are presented in [4]. These are: the Sample First Algorithm, the Query First Algorithm, the Partial Area Algorithm and the Spatial Reservoir Algorithm. In [4], all these methods treat the quadtree as a logical structure (without considering any specific implementation, or assuming a straight pointer implementation). In addition, [4] presents some abstract formulae which express the cost of each method. These formulae are only qualitative, since they do not depend on an explicitly defined probabilistic model (they depend on quantities such as the average distance to a quadtree leaf), and cannot be used for evaluation and derivation of concrete conclusions.

In the present report, in Section 2 we describe the most popular quadtree implementations. In Section 3 we examine how the above four sampling methods are applied to specific quadtree

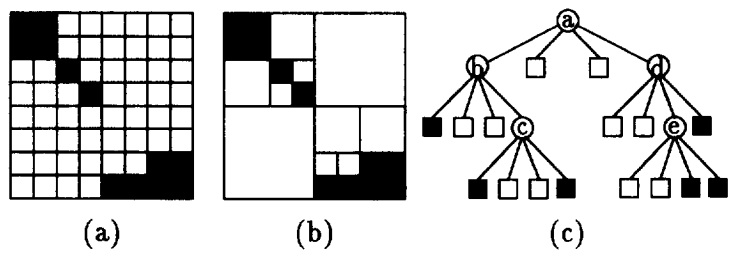


Fig. 1. (a)8×8 binary image, (b) the partitioning to unicolor quadrangular blocks, and (c) its quadtree.

implementations (to widely used linear implementations). In Section 4 we demonstrate how these methods can be analyzed: in Section 4.1 we introduce a parametric model of random images and (extending methods of [12]) we present analytic results on the cost of each method, whereas in Section 4.2 (extending methods of [14]) we produce similar analytic results based on a model of random trees. In the last section we summarize the contribution of this report and suggest plans for future research.

2. Quadtree implementations

The main quadtree implementations include pointer-based tree structures, linear Quadtrees and Depth First (DF) expressions [11]. We will briefly describe the most popular linear structures.

The linear quadtree representation consists of a set of values. For each black node of the pointer-based quadtree, there is one such value. The value of a node is an address (code-word) describing the position and size of the corresponding block in the image. These addresses can be stored as a list or in an efficient structure for secondary memory (such as a B-tree or one of its variations [1,10]). A database that uses this physical organization is described in [6]. If a B-tree is used for the whole database, in order to be able to discriminate between different images, the code-words should be augmented so as to include in their most significant bits the image identifier. There are variations of the linear representation where white nodes are stored too, or which are suitable for multicolour images. The most popular linear implementations are the FL (Fixed Length) and the FD (Fixed length-Depth) quadtrees [11].

In the former implementation, the address of a black quadtree node is a code-word that consists of n base 5 digits. Codes 0, 1, 2 and 3 denote directions NW, NE, SW and SE, respectively, while code 4 denotes a do-not-care direction. If the black node resides on level i, where $n \geq i \geq 0$, then the first $n - i$ digits express the directions that constitute the path from the root to this node and the last i digits are all equal to 4 (e.g., the code-word for the black child of node b is 004). In the latter implementation, the address of a black quadtree node has two parts: the first part is a code-word that consists of n base 4 digits. Codes 0, 1, 2 and 3 denote directions NW, NE, SW and SE, respectively. This code-word is formed in a similar way to the code-word of the FL-linear implementation with the difference that the last i digits are all equal to 0. The second part of the address has $\lceil log_2(n+1) \rceil$ bits and denotes the depth of the black node, or, in other words, the number of digits of the first part that express the path to this node (e.g. the code-word for the black child of node d is 330,01). Thus, using FL-linear representation, the size of an address in bits equals $\lceil n \cdot log_2(5) \rceil$, while using FD-linear representation the size of an address in bits equals $2n + \lceil log_2(n+1) \rceil$ [11].

3. Sampling methods and linear quadtrees

According to [3,5], there are various parameters that characterize different types of sampling: whether the sample size is fixed, whether samples are drawn with replacement, whether the access pattern is random or sequential, whether the size of the population (the number of database records) is known and whether each record has uniform inclusion probability.

In each of the sampling methods of this paper we want to obtain a uniform collection of s samples

(pixels) from the black regions of a $2^n \times 2^n$ image that is stored as a quadtree. The term uniform means that any pixel of our image is equally likely to be sampled (the probability of choosing a specific pixel is $1/4^n$). In addition, we assume that each of the s samples is independent of any other sample. This means that a certain pixel may appear more than once in the collection of the s samples. According to the classification of [3,5], a method that satisfies the uniformity and independence assumptions is characterized as Simple Random Sampling With Replacement (SRSWR).

The Sample First (SF) strategy consists of a loop in which we generate a random pixel location (x,y) and then perform a point location query on the quadtree (a search for the pixel that corresponds to this location). If the pixel is black, it is accepted into the sample. Otherwise, we loop until a black pixel is accepted. This process is repeated until we collect s samples. The Query First (QF) strategy searches the quadtree beginning at the root. At each gray node we choose any of its four children with equal probability (1/4). If the child is a white leaf, we return to the root and repeat the sampling procedure. If the child is a black leaf, we choose any of the pixels it contains with equal probability ($1/4^i$ for a level-i leaf) and repeat the whole process until we collect s samples. If the child is a gray node, we apply this procedure recursively. Due to the equal probabilities of each possible choice at any node, this strategy is equivalent to the sample first strategy. It is thus sufficient to examine only one of these methods. For ease of exposition we talk about the QF method.

The application of the QF or SF method is straight in a simple pointer implementation of quadtrees [4]. We will present the QF method in an FD linear implementation. Let us call A the candidate black node. Then:

1. Let A represent the root of the corresponding logical tree structure.
2. We create the code-word that represents A; let D be the first part of this code-word (the part denoting the path to A).
3. We search the linear quadtree for a code-word B that has a first part C such that $C \geq D$ and $C[j] = D[j]$ for $n \geq j \geq i + 1$ (i is the level of A).
4. If B exists and has the same depth as A, then A is black; we choose any of the pixels it contains with probability $1/4^i$ and repeat from step 1 until we collect s samples.
5. If B exists and has a larger depth than A, then A is a gray node; we choose as new A any of its four children with probability 1/4 and repeat from step 2 for the new A.
6. If B does not exist, then A is white; if A is the root of the tree, then sampling is impossible and the procedure ends with an error code; otherwise, we repeat from step 1.

We will also present the QF method in a FL linear implementation. Let us call A the candidate black node. Then:

1. Let A represent the root of the corresponding logical tree structure.
2. We create the code-word that represents A; let D represent this code-word.
3. We search the linear quadtree for a code-word B that is smaller than or equal to D and is identical to D in its first $n - i$ digits, that is, in the digits of D that are not equal to the do-not-care direction.
4. If B exists and is equal to D, then A is a black node; we choose any of the pixels of A with probability $1/4^i$ and repeat from step 1 until we collect s samples.
5. If B exists and is smaller then D, then A is a gray node; we choose as new A any of its four children with probability 1/4 and repeat from step 2 for the new A.

6. If B does not exist, then A is white; if A is the root of the tree, then sampling is impossible and the procedure ends with an error code; otherwise, we repeat from step 1.

The Partial Area Quadtree (PAQT) strategy is based on a quadtree variation where at each internal node x we store four numbers $a_{x,1}$, $a_{x,2}$, $a_{x,3}$ and $a_{x,4}$ ($a_{x,j}$ represents the total black area in the leaves of the j-th subtree of node x). The only difference with the QF strategy is that we choose a child (a subtree) of x with probability $b_{x,j} = a_{i,j}/\Sigma_{k=1}^{4} a_{x,k}$. For example, the quadruples of numbers that correspond to the gray nodes a, b, c, d and e of Fig. 1c are (3/32,0,0,3/32), (1/16,0,0,1/32), (1/64,0,0,1/64), (0,0,1/32,1/16) and (0,0,1/64,1/64), respectively. It is obvious that subtrees which consist of a single white leaf will not be selected (the probability of choosing such a leaf is 0). This method guaranties that each call to the sampling procedure returns a black point. It is obvious that this strategy requires an implementation with very large storage requirements: gray nodes should be stored too (apart from the code-word for each gray node, four numbers describing the black area of its children are needed).

The Spatial Reservoir (SR) strategy sequentially examines all the black leaves of a quadtree and at each leaf performs a binomial sampling procedure between two files. The first file consists of all the black leaves encountered so far and the second file consists of the next leaf (say the x-th leaf). The s samples are allocated to the two files according to a binomial distribution: we choose α pixels from the first file and $s - \alpha$ pixels from the second file, where $P\{\alpha = z\}$ equals $\binom{s}{z} p^z (1-p)^{s-z}$ with $p = 1 - (w_x/\Sigma_{j=1}^{x} w_j)$ and w_j equal to the area of the j-th leaf. The α pixels allocated to the first file can be chosen as a random subset of the sample constructed from the first $x - 1$ leaves. The related algorithm for a pointer quadtree implementation has been given in [4], where it is also proved that it creates a uniform collection of s samples (pixels) from the black regions stored in the quadtree. This algorithm adapted to FD or FL linear quadtrees is as follows (x represents the leaf under examination, w_x represents the area of x and W represents the total area of the leaves encountered so far):

1. We search the linear structure for the smallest code-word stored. This code-word represents x.
2. We create a collection of s samples from the sub-image corresponding to x using SRSWR and we set $W = w_x$.
3. We search the linear structure for the next larger code-word stored. This code-word represents x.
4. We set $W = W + w_x$ and we calculate α according to a binomial distribution based on w_x and W.
5. We choose at random (with equal probabilities) $s - \alpha$ samples from the collection created so far and we replace these samples with a collection of $s - \alpha$ samples that we create from the sub-image corresponding to x using SRSWR.
6. If we have not reached the last code-word stored in the linear structure we repeat from step 3.

4. Analysis of sampling methods

A quadtree for an image of size $2^n \times 2^n$ has maximum height n. We will call such a tree a class-n quadtree. A node that corresponds to a single pixel is at level 0, while the root is at level n. A node at level i, where $0 \leq i \leq n$, represents a subarray of $2^i \times 2^i (= 4^i)$ pixels, while there are at most 4^{n-i} nodes at this level. In the next two subsections we analyze the performance of the four sampling methods (applied to linear quadtrees) using two probabilistic models.

4.1. The model of random images

In the model of random images each pixel is statistically independent of any other pixel, as far as its colour is concerned. It is also assumed that a pixel is black with probability p and white with probability $1-p$. In the rest of the paper we will refer to an image obeying this model as a random image. These mean that in a quadtree the block corresponding to a level-i node is black with probability $p^{(4^i)}$, white with probability $(1-p)^{(4^i)}$, and gray with probability $1-p^{(4^i)}-(1-p)^{(4^i)}$. Generally, we will use the symbols B_i and W_i to denote the probability that the block corresponding to a level-i node is black or white, respectively.

The random image model is realistic for a particular area of values of p. More specifically, when p is not close to 0.5 (e.g. $p=0.9$) our model expresses spatial coherence to a significant extent and can represent images found in practice (for example medical or meteorological images). In our analysis we consider class-n quadtrees that represent random images. We can now present the following propositions.

Proposition 4.1. *In the SF or in the QF method the average number of calls to the sampling procedure in order to obtain s samples or abort the procedure due to a white image is*:

$$(1-p)^{4^n}+s\left(\sum_{c=1}^{\infty}\sum_{l=0}^{4^n-1}\binom{4^n}{l}c\left(\frac{l}{4^n}\right)^{c-1}\frac{4^n-l}{4^n}(1-p)^l p^{4^n-l}\right)$$

Proof. If our image is all white (prob. W_n), the sampling procedure will return an error at the first call. Consider an image that has $l<4^n$ white pixels and 4^n-l black pixels. Its probability of existence is $P_1=(1-p)^l p^{4^n-l}$. Since an area is sampled with probability proportional to its size, the probability of choosing a white pixel in $c-1$ calls and a black pixel in the c-th call is $P_2=(l/4^n)^{c-1}\cdot(4^n-l)/4^n$. Thus, the contribution of this particular situation to the calculation of the average value of calls (needed for obtaining this particular sample) will be $c\cdot P_1\cdot P_2$. Summing up for every possible number of calls c, every possible number of white pixels l in an image, all possible images that have l white pixels and multiplying the result by s, we get the above proposition. □

Proposition 4.2. *In the SF or in the QF method (applied to a linear quadtree) the average number of code-words searched in one call to the sampling procedure is*:

$$(n+1)-\sum_{i=1}^{n}(1-p)^{4^i}+p^{4^i}$$

Proof. Consider a class-n quadtree. The probability that a level-i node ($i<n$) exists in this tree and is a leaf equals $W_i(1-W_i^3)+B_i(1-B_i^3)$. In other words, this is the probability that the block represented by this node is white or black and that the blocks represented by its siblings are not all completely white or black, respectively. Such a node is sampled with probability $1/4^{n-i}$ and requires searching for $n-i+1$ code-words. There are 4^{n-i} such potential nodes at level i. In addition, the tree will consist of a single white or black node with probability W_n+B_n. Such a node is sampled with probability 1 and requires searching only for its own code-word. Adding the expected values of the number of code-words searched for all level-i nodes and for all levels we have

$$1 \cdot 1 \cdot (W_n + B_n) + \sum_{i=0}^{n-1} (n-i+1)4^{n-i}(1/4^{n-i})(W_i - W_{i+1} + B_i - B_{i+1})$$

Algebraic manipulations complete the proof. □

Proposition 4.3. *Let* $R_i = 4^n - 4^{i+1}$, $T_i = 3 \cdot 4^i - 1$ *and* $V_{i,l,k} = p^{4^i+l+k} \cdot (1-p)^{4^n-4^i-l-k}$. *In the PAQT method (applied to a linear quadtree) the average number of code-words searched in order to obtain s samples or abort the procedure due to a white image is*:

$$(1-p)^{4^n} + s\left(p^{4^n} + \sum_{i=0}^{n-1}\sum_{k=0}^{R_i}\sum_{l=0}^{T_i}\binom{R_i}{k}\binom{T_i}{l}(n-i+1)\frac{4^n}{4^i+l}V_{i,l,k}\right)$$

Proof. If our image is all white the sampling procedure will return an error at the first call. If our image is all black then only one code-word will be searched. Consider an image that has a level-i black node, $l < 3 \cdot 4^i$ extra black pixels in the three blocks that correspond to the sibling nodes of this black node and $k < 4^n - 4^{i+1}$ extra black pixels in any other node. The probability of the existence of this image is $P_1 = p^{4^i+l+k}(1-p)^{4^n-4^i-l-k}$. It is not difficult to see that such a level-i node is sampled with probability $P_2 = 4^i/(4^i+l+k)$ and requires searching for $n-i+1$ code-words. Thus, the contribution of this particular situation to the calculation of the average number of code-words searched (in order to obtain a particular sample) will be $(n-i+1) \cdot P_1 \cdot P_2$. There are 4^{n-i} such potential nodes at level i. Summing up for every possible level, every potential black node, every possible values of l and k, all possible images that have 4^i+l+k black pixels and multiplying the result by s, we get the above proposition. □

Proposition 4.4. *In the SR method (applied to a linear quadtree) the average number of code-words accessed in order to obtain s samples is*:

$$p^{4^n} + \sum_{i=0}^{n-1} 4^{n-i}(p^{4^i} - p^{4^{i+1}})$$

Proof. In [12] it is proved that the average number of black leaves at level i of a class-n quadtree representing a random image equals p^{4^i} when $i = n$ and $4^{n-i}(p^{4^i} - p^{4^{i+1}})$ when $n > i \geq 0$. Since in the SR method we examine all nodes stored in a linear quadtree, summing up the numbers of black leaves for all levels, we get the above proposition. □

4.2. The model of random trees

Shaffer and Samet have given a descriptive definition of a model of random quadtrees [7,9]. According to this model "each leaf node is assumed to be equally likely to appear at any position and level in the tree". This model is generally considered very realistic for images usually found in practice and it has been used for the analysis of neighbour-finding algorithms producing results close to the statistics of real tests.

A formal equivalent of the above definition was presented in [14]. First, a branching process by which we can construct any legal class-n region quadtree has been described. More specifically, following this process

- at the beginning, we perform the initial branching: we choose between the tree root being a black node, a white node or a gray node,
- at any level i, $n \geq i > 1$, for any gray node at this level, we perform a level-i branching: we choose between 79 different subtree configurations and set this node to be the root of the chosen configuration (since a child can be black, white or gray, the total number of subtree configurations is 4^3 $(=81)$ minus 2 configurations where all children are black or white),
- at level 1, for any gray node at this level, we perform a level-1 branching: we choose between 14 different subtree configurations and set this node to be the root of the chosen configuration (since a child can be black or white, the total number of subtree configurations is 4^2 $(=16)$ minus 2 configurations where all children are black or white).

Then, a simple probabilistic model for this branching process has been created by assigning parametric probabilities to the different choices for every branching. Note that in order for this process to be legal under the fundamental probability axioms, for every specific branching, the probabilities of all the different choices must sum to 1. In [14] it was proved that, for specific values of the parameters defining the branching probabilities, the formal random tree model is equivalent to Samet and Shaffer's descriptive model and the probability of a black or white node existing anywhere in the tree equals $1/(2n+2)$ (constant). This value is sufficient for performing our analysis of the sampling algorithms. In our analysis we consider class-n quadtrees that obey the random tree model. Note that in the model of random trees there are correlations between the probabilities of different nodes and this limits the analytic results that can be proved with this model.

We can now present the following propositions.

Proposition 4.5. *In the SF or in the QF method (applied to a linear quadtree) the average number of code-words searched in one call to the sampling procedure is*:

$$\frac{n+2}{2}$$

Proof. The probability that a level-i node $(i<n)$ of a class-n quadtree exists in this tree and is a white or black leaf equals $1/(n+1)$. Such a node is sampled with probability $1/4^{n-i}$ and requires searching for $n-i+1$ code-words. Note also that there are 4^{n-i} such potential nodes at level i. Adding the average numbers of the code-words searched for all the nodes at each level, we have

$$\sum_{i=0}^{n} (n-i+1)\frac{1}{n+1}\frac{1}{4^{n-i}}4^{n-i} = \frac{n+2}{2}.$$

Proposition 4.6. *In the SR method (applied to a linear quadtree) the average number of code-words accessed in order to obtain s samples is*:

$$\frac{4^{n+1}-1}{6n+6}$$

Proof. We associate one random variable with each potential node of level i of a class-n quadtree, $0 \leq i \leq n$. This variable equals 1 when the related node exists and is black; otherwise it equals 0. By

summing up the average values of all these random variable we have that the average number of black leaves at level i is $4^{n-i}/(2n+2)$, $0 \leq i \leq n$. Since in the SR method we examine all nodes stored in a linear quadtree, summing up the average numbers of black leaves for all levels, we have

$$\sum_{i=0}^{n} 4^{n-i} \frac{1}{2n+2} = \frac{1}{2n+2} \frac{4^{n+1}-1}{3} = \frac{4^{n+1}-1}{6n+6}. \quad \square$$

5. Conclusion

In the present report, we discussed the application of four different sampling methods (the Sample First, Query First, Partial Area and Spatial Reservoir algorithms) to the most popular quadtree implementations found in spatial databases, that is the FL and FD quadtree linear implementations. We have also presented analytic results (propositions) on the cost of these methods. These results were presented as a demonstration of how the sampling methods can be analyzed under two probabilistic models (random trees and random images). Thus, these propositions can be either used for a comparative study of the different algorithms when evaluated for specific parameter values, or they can used as the basis for further research on the analytic cost of sampling.

Note that the model or random trees is widely accepted as a realistic image model. Moreover, it leads to closed formulae. The model of random images is reasonable for certain categories of images only when p is not close to 0.5 and when n is not very large. However, it can lead to the proof of more analytic results than the model of random trees.

As an example of evaluating the formulae developed we present Table 1. For image sizes from 128×128 pixels to 1024×1024 pixels (from $n=7$ to $n=10$) the results of Propositions 4.2 and 4.4 when p equals 0.6, 0.9 and 0.95 and the results of Propositions 4.5 and 4.6 are depicted. It is evident that spatial coherence is higher for the model of random trees. However, the model of random images could describe certain kinds of images (like an image of a population of insects) when n is not larger than 9 and when p equals 0.9 or 0.95. Propositions 4.1 and 4.3 cannot be evaluated easily, since they contain factorials of large numbers, like 4^n (Proposition 4.1 even contains an infinite sum). However, they are presented in this article for the sake of completeness and as elements leading to further theoretical research.

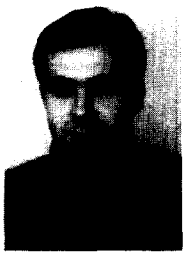
Table 1
For the two models, the number of code-words searched in one call to the sampling procedure in the SF/QF method and the number of code-words accessed in the SR method

Method	Random images			Random trees
	$p=0.6$	$p=0.9$	$p=0.95$	
SF or QF ($n=7$)	7.84	7.16	6.71	4.5
SR ($n=7$)	8237	6113	4175	1365
SF or QF ($n=8$)	8.84	8.16	7.71	5.0
SR ($n=8$)	32948	24453	16701	4855
SF or QF ($n=9$)	9.84	9.16	8.71	5.5
SR ($n=9$)	131792	97813	66804	17476
SF or QF ($n=10$)	10.84	10.16	9.71	6.0
SR ($n=10$)	527169	391251	267217	63550

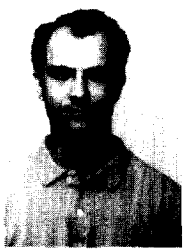
Future research could examine improved sampling methods or new analytic results based on other random models. For example, extensions of the model of random images that improve spatial coherence (and make the model more realistic) [13] could be used.

References

[1] D.J. Abel, A B^+-tree structure for large quadtrees, *Computer Vision, Graphics and Image Processing* 27(1) (1984) 19–31.
[2] R. Laurini and D. Thomson, *Fundamentals of Spatial Information Systems* (Academic Press, London, 1992).
[3] F. Olken and D. Rotem, Random sampling from database files: A survey, *Proc. 5th Inter. Conf. on Statistical and Scientific Database Management* (1992) 92–111.
[4] F. Olken and D. Rotem, Sampling from spatial databases, *Proc. 9th IEEE Data Engineering Conf.* (1993) 199–208.
[5] F. Olken, Random sampling from databases, *Ph.D. Dissertation* (Department of Computer Science, University of California at Berkeley, CA, 1993).
[6] C.A. Shaffer, H. Samet and R.C. Nelson, QUILT: A geographic information system based on quadtrees, *International Journal of Geographic Information Systems* 4(2) (1990) 103–131.
[7] H. Samet, Neighbour finding techniques for images represented by quadtrees, *Computer Graphics and Image Processing* 18(11) (1982) 35–57.
[8] H. Samet, The quadtree and related hierarchical data structures, *ACM Computing Surveys* 16(2) (1984) 187–260.
[9] H. Samet and C. A. Shaffer, A model for the analysis of neighbour finding in pointer-based quadtrees, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 7(6) (1985) 717–720.
[10] H. Samet, C.A. Shaffer, R.C. Nelson, Y.G. Huang, K. Fujimura and A. Rosenfeld, Recent developments in linear quadtree-based geographic information systems, *Image and Vision Computing* 5(3) (1987) 187–197.
[11] H. Samet, *Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS* (Addison-Wesley, Reading, MA, 1990).
[12] M. Vassilakopoulos and Y. Manolopoulos, Analytical results on the quadtree storage-requirements, *Proc. 5th Inter. Conf. on Computer Analysis of Images and Patterns*, Budapest (1993) 41–48.
[13] M. Vassilakopoulos and Y. Manolopoulos, Analytical comparison of two spatial data structures, *Information Systems* 19(7) (1994) 569–582.
[14] M. Vassilakopoulos and Y. Manolopoulos, On random models for analysing region quadtrees, *Pattern Recognition Letters* 16 (1995) 1137–1145.



Michael Gr. Vassilakopoulos was born in Thessaloniki, Greece in 1967. He received a B. Eng. (1990) from the Dept. of Computer Eng. and Informatics of the Univ. of Patras and a Ph.D. (1995) from the Dept. of Electrical and Computer Eng. of the Aristotle Univ. of Thessaloniki. His B. Eng. thesis deals with Algorithms of Computational Geometry and his doctoral thesis deals with Spatial Data Structures. Apart from the above, his research interests include Graphics, Databases and Multimedia. For 1996–1997, Dr. Vassilakopoulos has been appointed Instructor at the Dept. of Applied Informatics of the Univ. of Macedonia in Thessaloniki and the Dept. of Informatics of the Technology Educational Institute of Thessaloniki.



Yannis Manolopoulos was born in Thessaloniki, Greece in 1957. He received a B. Eng. (1981) in Electrical Eng. and a Ph.D. (1986) in Computer Eng., both from the Aristotle Univ. of Thessaloniki. He has been with the Department of Computer Science of the Univ. of Toronto, the Department of Computer Science of the Univ. of Maryland at College Park and the Department of Electrical and Computer Eng. of the Aristotle Univ. of Thessaloniki. Currently, he is Associate Professor at the Department of Informatics of the latter university. He has published over 70 papers in refereed scientific journals and conference proceedings. He is author of two textbooks on data/file structures, which are recommended in the vast majority of the computer science/engineering departments in Greece. His research interests include spatial and temporal databases, geographical information systems, text databases, data/file structures and algorithms and performance evaluation of disk systems.