Report 120

Analysis and Comparison of the Performance of the Fieldtree

Renato Barrera
Andrew U. Frank

Analysis and Comparison of the Performance of the Fieldtree [*]

Renato Barrera

Andrew U. Frank

National Center for Geographic Information and Analysis

and

Department of Surveying Engineering

University of Maine

Orono, ME 04469, USA

RENATO@MECAN1.bitnet

FRANK@MECAN1.bitnet

Abstract

The Fieldtree is compared to the Rtree and R^+tree using published analytical examples posed in an unidimensional environment and dealing with one or two populations of objects. All objects within a population are uniformily distributed and have identical sizes. Two varieties of the Fieldtree (namely the cover and the partition Fieldtree) are considered.

Comparisons are made for two kinds of questions: point location and range queries. The behavior of R^+tree is definitely better for point location queries and low overlap factors; however, that superiority

[*]This research was partially funded by grants from NSF under No. IST 86-09123 and Digital Equipment Corporation (Principal Investigator: Andrew U. Frank). The support from NSF for the NCGIA under grant number SES 88-10917 is gratefully acknowledged.

is lost when higher overlaps degrade its performance indefinitely, while all other trees show a robust behavior.

The cover Fieldtree and the Rtree behave identically for the case of one population. The cover Fieldtree may display a better performance when two populations of widely different objects are considered,

The cover Fieldtree performance was consistently better than the one of the partition variety under one dimensional point and range query. Further analysis shows that this superiority is preserved in the two dimensional case.

As expected, no known file structure is optimal for all applications. The analysis reported in this work indicates the convenience of the Fieldtrees for environments where range queries are common and where object nesting and/or high overlap factors occur.

1 Introduction

This article, a companion of [Frank 1989], compares several implementation of the Fieldtree to the Rtree and the R^+ tree. The comparisons are based solely upon the number of page accesses needed, are made for the one dimensional case only, and involve both point location and range queries. The bidimensional behavior for point queries of different variants of the Fieldtree is also examined.

This paper is organized as follows: Section 2 briefly gives cost estimates for point location and range queries for R^+trees and the Rtrees that contain a single population of objects; the following section examines the behavior of the Fieldtree under those type of queries. An analysis of the two dimensional Fieldtree is performed in section 4. The last section contains a comparison between the behavior of the cover Fieldtree and the Rtree for the case of several populations of objects. Finally, some conclusions are drawn.

2 results on the Rtree and R^+tree

Faloutsos et al. [Faloutsos 1987] have computed analytical results describing the behavior of Rtrees and R^+trees for point queries. Their examples, given in a one dimensional environment, consider one or two populations of objects. All objects in a population have identical size and their centroids are uniformily distributed on the [0.0, 1.0] interval; the data leaves of the trees are supposed to be full.

The cited reference uses the following terminology:

C Maximum number of data items in a data leaf.

n Total number of data items.

σ Size of a data item.

f Fan-out of the tree

O Number of data items that contain a point $(O = \sigma(n+1)/(1+\sigma))$

For consistency with the nomenclature of this paper, TR and TR^+ will stand for the cost of point query that uses respectively an Rtree or an R^+tree as its spatial access method. This cost will be measured in page accesses.

2.1 Point Queries for the R^+tree and the Rtree

Reference [Faloutsos 1987] obtains formulas for point queries that utilize Rtree and R^+tree. These formulas are:

$$TR^+ \approx 1 + log_f(n/C) + log_f(1/(1-\sigma n/C)) \qquad (1)$$

$$TR \approx 1 + log_f(n/C) + \frac{\sigma n}{C}\frac{(f - C/n)}{f-1} \qquad (2)$$

The following subsections will give formulas for range queries of length l, similar to (2), (1).

2.2 Range Queries for the R^+tree

Reference [Faloutsos 1987] provides a formula for h, the height of the R^+tree:

$$h = log_f((n - C)/O) \quad (3)$$

In addition to the effort involved in a point query, the system will have to perform two extra functions to answer a range query:

i) Scan an additional nl/C data pages

ii) Search through the h levels of the tree for the addresses of such pages

The lowest level of indices in an R^+tree contains references to approximately $(C - O)/f$ objects, the next level to $(C - O)/f^2$, and thus, the expected numer of additional index visits is given by $\frac{nl}{C-O}(1/f + \cdots + 1/f^h)$. Substituting h by its value in (3) renders a total of $\frac{nl}{C-O}(1 - \frac{(C-O)}{n})/(f-1)$ extra index pages.

Adding to (1) the accesses due to the extra data and index pages, substituting O by its value, and supposing that $n \gg C$, TrR^+, the effort required to satisfy a range query in an R^+tree is:

$$TrR^+ = 1 + log_f(\frac{n}{(C - \sigma n)}) + \frac{f}{f-1}(1 + \frac{\sigma n}{(C - \sigma n)})\frac{nl}{C} \quad (4)$$

2.2.1 Range Queries for the Rtree

As in the case of an R^+ tree, solving a range query in an Rtree environment will involve scanning new data pages and searching through h levels of the index hierarchy. In this case, a range query of length l will involve scanning through $(l + \sigma)n/C$ extra data pages. Supposing that $n \gg 1$ and that $(fC \gg \sigma n)$, $h \approx log_f(n/C)$ [Faloutsos 1987], those extra $(l + \sigma)n/C$ data page accesses will involve fetching $(l + \sigma)n/(fC)$ pages at the leaf level of the tree, $(l + \sigma)n/(f^2C)$ at the following one, etc., so a total of

$$(l + \sigma)n/C \times (1/f + \cdots + 1/f^h)$$

$$= \quad (l+\sigma)n)/C) \times (1 - f^h)/(f-1)$$

$$= \quad (l+\sigma n)/C) \times (1 - C/n)/(f-1)$$

extra index page accesses will be needed. Adding the previous figures to those of (2) a formula for the number of pages needed for solving a range query in an Rtree environment, called TrR, can be obtained.

$$TrR = 1 + log_f(n/C) + \frac{(f - C/n)}{(f-1)}\frac{(l+\sigma)n}{C} \qquad (5)$$

3 Fieldtree Analysis: One Dimension, One Population

This section will deal on the obtention, for the one dimensional Fieldtree, of formulas similar to (2, 5). Two Fieldtree variants will be considered: the cover and the partition Fieldtree [Frank 1989]. The later variant will be further analyzed under two conditions, depending on how overflow is handled (by fields or by ad-hoc records).

For each tree, two implementations of the tree's hierarchy will be considered: indexing by pointers interior to the fields, and utilizing a B^+tree-like structure and positional codes as the indexing mechanism.

All our results are concerned only with disk accesses, for that factor is usually more critical than others, i.e., processing time, storage needed, etc.

Subsection 3.1 will introduce some new nomenclature; the following two subsections will be dedicated to the obtention of formulas for the two components of the access cost of a query that each tree variant needs for satisfying point and range queries, namely: i) Data page accesses and ii) Index page accesses. Finally, subsection 3.4 will compare the results of the different variants of the Fieldtree to the R^+tree and to the Rtree.

3.1 Nomenclature

Using the same nomenclature as [Faloutsos 1987], C, n, f, σ will respectively stand for the leaf capacity, the total number of objects, the fan-out factor of the index and the size of an object. Three new variables will be introduced:

l Size of a range-query window.

h_f Height of the binary Fieldtree.

h Height of an B^+-tree, or an R^+-tree or an R-tree, depending on the case.

To distinguish among the different implementations, variable names will be composed of two parts: one referring to the type of access, and a posfix related to the type of tree. The part related to the type of access will be one of the following:

I, Ir	# of index pages retrieved for point and range queries respectively.
D, Dr	# of data pages retrieved for point and range queries.
T, Tr	Total # of pages retrieved for point and range queries.

The type of tree used in the solution of the query will determine a posfix in accordance with the following list:

p^{no}	Partition Fieldtree with no added overflow pages.
p^{ov}	Partition Fieldtree, ad-hoc overflow pages.
C	Cover tree.
R^+, R	R^+tree, Rtree respectively.

A subindex '$2d$' appended to a prefix will denote a two-dimensional tree; its absence will denote a one-dimensional case. A subindex 'p' appended to a posfix will denote that the tree hierarchy is implemented by pointers among fields; its absence will signify an implementation by multiway trees.

As an example, DrP_p^{po} will refer to the number of datapage accesses for a range query involving a partition Fieldtree implemented by pointers, and TR to the total number of accesses for a point query to an Rtree.

In the analysis of all Fieldtree variants, two conditions should be satisfied:

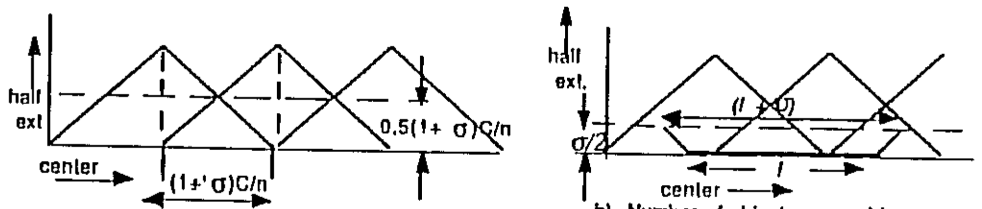
i) **Fit:** Objects should be placed in fields in size according to its importance. When all objects are of the same size, their preferred position will be in the leaves of the tree

ii) **Capacity:** There should be enough space in the Fieldtree to store all the objects. The leaf fields will be filled to their capacity

Te extent of the initial field will be $1 + \sigma$ in order to accommodate all objects of extent σ with centers in the $(0, 1)$ interval.

3.2 Cover Fieldtree

<u>Data access</u> Our analysis will consider only cases where $(\sigma n/C < 1)$, since for that range of values the behavior of the R^+tree and the behavior of some versions of the partition field tree will be properly defined.



a) Condition of fit

b) Number of objects covered in a range query

Fig. 1) Space conditions in the 1-d cover tree

Figure (1.a) will be used to show that whenever $\sigma n/C < 1$ all data can be placed in the leaf fields and still satisfy the condition on fit. That figure displays the loci of the leaf fields of the tree in transformed *(center, half-extent)* coordinates: it shows that all objects whose half-extent is less than $0.5/(2^{h_f})$ will fit into a leaf field, and thus, since $\sigma < C/n$, all objects can fit into those leaves.

The capacity condition requires the existence of at least n/C fields. Since all of them can be leaves, the height h_f of the tree is fixed by the equation $2^{h_f} = n/C$ as:

$$h_f = log_2(n/C)$$

Fig. 1.b) shows, again in transformed space, the loci of the of the fields needed for a range query of length l. From that figure results that $n(l+\sigma)/C$ fields must be visited for such query.

With that information, formulas can be deduced for the data page accesses in point and range queries (DC, DrC):

$$\begin{aligned} DC &= 1 + \frac{\sigma}{(1+\sigma)}\frac{n}{C} \\ DrC &= 1 + \frac{(\sigma+l)}{(1+\sigma)}\frac{n}{C} \end{aligned}$$

The following paragraphs will obtain the number of index page accesses needed, first when the index is implemented by a multiway tree, and afterwards, when implemented by pointers iside the fields.

Index access - Multiway trees Access operations are divided into two phases: i) getting the address of the first field, and ii) obtaining the addresses of the remaining ones.

The first phase will descend to the leaves of the tree in $h = log_f(n/C)$ accesses; each additional field to be accessed will take $1/f + \cdots + 1/f^h = (1 - 1/f^h)/(f-1) = (1 - C/n)/(f-1)$ extra index accesses, and so the formulas for the index accesses for point and range queries (IC, IrC) are:

$$\begin{aligned} IrC &= log_f(n/C) + \frac{(\sigma+l)}{(1+\sigma)}\frac{(n/C-1)}{(f-1)} \\ IC &= log_f(n/C) + \frac{\sigma}{(1+\sigma)}\frac{(n/C-1)}{(f-1)} \end{aligned}$$

Index access- Interior pointers This implementation, discussed in [Frank 1989], has each non-leaf field pointing to two successors. Getting the address of the data record requires $(h_f - 1)$ accesses, where h_f is given by $log_2(n/C)$.

Two facts will be used in the obtention of the index acces cost for a range query involving 'x' consecutive leaf fields: i) a binary tree with x leaves has $x - 1$ intermediate nodes and ii) a balanced tree of x leaves has a height of $log_2(x)$.

The first fact implies that the traversal of x leaves will involve $x - 1$ extra field accesses; the second that such traversal will climb $log_2(x)$ levels in the Fieldtree. It should be accounted that an intermediate field first traversed during the first leaf field access will be revisited for each level ascended. Taking all that into consideration, the cost for the index accesses for point and range queries (IC_p, IrC_p) is given as:

$$\begin{aligned} IC_p &= log_2(n/C) - 1 + (DC - 1) - log_2 DC \\ IrC_p &= log_2(n/C) - 1 + (DrC - 1) - log_2 DrC \end{aligned}$$

Cost for point and range queries Adding the index and the data part of the cost, the total costs for point and range queries for the cover tree with a single population are:

$$\begin{aligned} TC &= 1 + log_f(n/C) + \frac{\sigma n}{C}\frac{(f - C/n)}{f - 1} \\ TC_p &= log_2(\frac{(n/C)}{1 + \sigma n/C}) + 2\sigma n/C \\ TrC &= 1 + log_f(n/C) + \frac{(f - C/n)}{(f - 1)}\frac{(l + \sigma)n}{C} \\ TrC_p &= log_2(\frac{(n/C)}{1 + (\sigma + l)n/C}) + 2(\sigma + l)n/C \end{aligned}$$

3.3 Partition Fieldtree

Three more variables, in addition to those defined in the previous paragraphs, will be needed for the analysis of this variety of the Fieldtree:

θ Fraction of the objects that will be stored at the leaves of the Fieldtree.

k Number of intermediate levels occupied by data in a Fieldtree.

ω Fraction of the point queries that can be solved without accessing the Fieldtree intermediate nodes.

The conditions on fit and capacity cited in Sec. 3.1 will now be used to derive *theta*, k and ω as a function of σ, n and C.

Condition on fit. The Fieldtree has 2^{h_f} leaves spanning the interval $1.0 + \sigma$. Fig (2) shows how fields are displayed in (*center, half-extent*) transformed space [Frank 1989]. By hypothesis, a portion $1 - \theta$ of all objects will migrate to ancestor fields. It can be seen from that figure that the portion of migrating nodes must be at least a/b. Substituting a, b by the values given in Fig.2 gives the following result:

$$\sigma \times 2^{h_f}/(\sigma + 1) \leq 1 - \theta \tag{6}$$

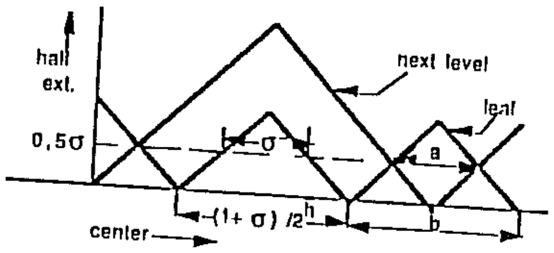


Fig. 2 Conditions of fit for the partition tree

An equation for ω, the fraction of the point queries solvable with a single field access will now be obtained, and for that, we will resort again to transformed (*center, half-extent*) space. In that space, the loci of all objects that contain point x_1 is a cone with its apex in $(x_1, 0)$ and with edges at $45°$. In figure 3 it can be seen that all point queries in interval z will consult one field only. Since the length of z is $b - a$, a fraction $(b - a)/b$ of the queries will fulfill that condition and thus the following equation proceeds:

$$\omega = (2 \times \theta - 1) \tag{7}$$

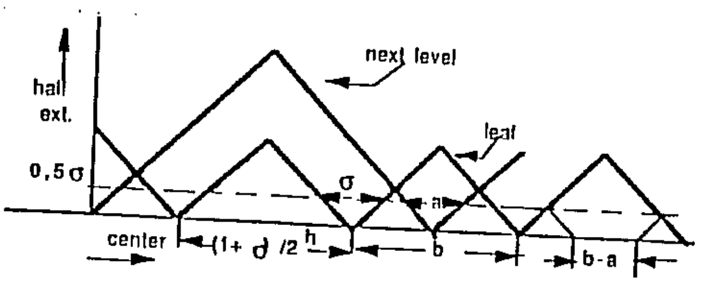


Fig. 3 Conditions for a single access in a point query

Condition on Capacity. This condition will be split into two parts: i) for the space inside the leaves, and ii) for the space in the intermediate nodes.

The first part of the condition renders formula (8). This expression is obtained by considering that a leaf can contain up to C objects and that a fraction θ of the n objects must be stored into leaves. Since a full binary tree of height h has 2^h leaves, it follows that:

$$(\theta n)/C \leq 2^{h_l} \tag{8}$$

θ can be obtained as a corollary to the previous expression. If all leaves are to be packed, formulas (6, 8) should be equalities and combining them it follows that:

$$\theta = \frac{C}{C + \frac{\sigma n}{1+\sigma}} \tag{9}$$

A mathematical expression for the second part of the capacity condition can be obtained through the following considerations:

The number of objects to be placed at intermediate nodes is $n(1-\theta)$. If $n\theta$ objects are to be at the 2^h leaves, there is room for $.5^m n\theta$ of them at m^{th} next level. Thus, the k intermediate levels of of the Fieldtree have a spare capacity for $(.5 + \ldots + .5^k)n\theta = (1.0 - .5^k)n\theta$ objects. Since that spare capacity must be sufficient to hold the objects that migrate from the leaves ($n(1-\theta)$), an expression follows:

$$(1.0-\theta)/\theta \leq 1.0 - 0.5^{k-1}$$

As a corollary k is given by

(10)

$$
\begin{aligned}
k &= log_2(\theta/(2\theta - 1)) \\
&\approx \lceil log_2(\frac{C}{C - \sigma n}) \rceil
\end{aligned} \qquad (11)
$$

Eqn (11) was obtained by supposing that $\sigma \ll 1$, and renders a set of values as:

$$
k = \begin{cases} 1 & \text{if} \quad 0 < \sigma n/C \leq 0.5 \\ 2 & \text{if} \quad 0.5 < \sigma n/C \leq 0.6666 \\ \ldots & \end{cases}
$$

3.3.1 Partition Fieldtree- no overflow pages

$\theta n/C$ leaf fields will be available in this tree version. All objects that can not be positioned in that level, either because of a bad fit or because of saturation of the leaves will migrate to ancestor fields.

<u>Data Acesses</u> For point queries, one field access will suffice for a fraction ω of the queries; the rest will require $k + 1$ accesses; thus a point query will require $1 + k(1 - \omega) = 1 + 2k(1 - \theta)$ and so, substituting values, DP^{no} is:

$$DP^{no} \approx 1 + \frac{2k\sigma n}{C + \sigma n}$$

A range query of length l needs a visit to the initial point and then a traversal of $x = \theta n l/C$ consecutive leaves. To do that $x(0.5 + ... + 0.5^k) = x(1 - 0.5^k) =$ intermediate fields with data will be visited. A number of them, given by min(k, $nl\theta/C$), would be revisited and thus, the formula for the data page accesses is:

$$DrP^{no} = DP^{no} + \theta(1 - 0.5^k)nl/C - \min(k, log_2(\theta nl/C))$$

As a corollary of the analysis of the previous paragraph, the total number of data pages $num_{f\text{-}part}$ in a partition tree can be obtained by making $x = \theta n(1 + \sigma)/C$; if $\sigma \ll 1$, that number is approximately

$$num_{f\text{-}part} = \frac{n(2 - 0.5^k)}{C + \sigma n} \qquad (12)$$

Index Accesses- Multiway tree Again, access operations will have two phases: i) getting the address of the initial and ii) acquiring the remaining addresses. The first phase will cause $log_f(num_{f\_part})$ accesses. For a point query, the second phase will take $(DP^{no}-1)(1-1/num_{f\_part})/(f-1)$ accesses; for a range query it will require $(DrP^{no}-1)(1-1/num_{f\_part})/(f-1)$ accesses, yielding:

$$IP^{no} = log_f(num_{f\_part}) + (DP^{no}-1)\frac{(1-1/num_{f\_part})}{(f-1)}$$
$$IrP^{no} = log_f(num_{f\_part}) + (DrP^{no}-1)\frac{(1-1/num_{f\_part})}{(f-1)}$$

Cost for Point and Range Queries Using the results of the previous paragraph, the effort to answer queries for the multiway implementations can be quantified as:

$$\begin{aligned} TP^{mo} &= 1 + log_f(\frac{n(2-0.5^k)}{C+\sigma n}) + \frac{2k\sigma n}{C+\sigma n}\frac{(f-\frac{C/n+\sigma}{(2-0.5^k)})}{(f-1)} \\ T_rP^{mo} &= 1 + log_f(\frac{n(2-0.5^k)}{C+\sigma n}) + [\frac{n(2k\sigma+l(2-0.5^k))}{C+\sigma n} - \min(k, log_2(\frac{nl}{C+\sigma n}))]\frac{(f-\frac{C/n+\sigma}{(2-0.5^k)})}{(f-1)} \end{aligned}$$

The solution of point query in a partition Fieldtree with fields indexed by internal pointers only requires a descent down the Fieldtree, and so, TP_p^{no} is given by:

$$TP_p^{no} = log_2(\frac{n}{C+\sigma n})$$

The solution of a range query in a partition Fieldtree indexed by internal pointer requires two phases: i) a descent down the tree, and ii) a visit to a total of $num_{f\_part} \times (nl+\sigma)$ fields ($n(l+\sigma)\theta/C$ of which are leaves). The second phase will imply reclimbing some levels of the tree; while doing that, a $log_2(n(l+\sigma)\theta/C))$ fields would be revisited. Taking all that into consideration, the cost of that range query is:

$$T_rP_p^{no} = 1 + log_2(\frac{n}{C+n(l+\sigma)}) + n(l+\sigma)\frac{(2-0.5^k)}{C+\sigma n}$$

3.3.2 Partition Fieldtree- overflow pages

There will be $\theta n/C$ leaf fields available in this version of the partition tree. If an object cannot be placed in a leaf field (for reasons of bad fit or because of saturation of the field) it will migrate to fields in the next level; fields at that level will be provided with an overflow page in case of saturation.

<u>Data access</u> This tree will behave identically to the non-overflow version until no more available space remains in the first intermediate level. At that moment, two thirds of the data are stored at the leaves ($\theta = 0.666$). For bigger loads, i.e., for values of θ in the range $0.5 < \theta < 0.666$, an overflow page will provide enough storage and data need not migrate to higher levels.

Thus, for $.666 < \theta < 1$ a fraction $(1 - \omega)$ of the queries will require two page access, the remaining fraction needing only one. For $0.5 < \theta < 0.666$ a fraction $1 - \omega$ will require three page access. The formula for the datapage cost is:

$$DP^{pov} \approx \begin{cases} 1 + 2\sigma n/(C + \sigma n) & \text{if} \quad 0 < \sigma n/C \le 0.5 \\ 1 + 4\sigma n/(C + \sigma n) & \text{if} \quad 0.5 < \sigma n/C \le 1 \end{cases}$$

For a range query of length l, $x = \theta nl/C$ leaf fields must be traversed in addition to the first one. This traversal will need to access $0.5x$ intermediate leaves; one of this intermediate leaves will be revisited and hence must be discounted. When $0.666 < \theta < 1$ each of those intermediate leaves will have one disk access associated; otherwise they will have two. The resulting formulas is:

$$DrP^{pov} \approx \begin{cases} DP^{pov} + 1.5nl/(C + \sigma n) & \text{if} \quad 0 < \sigma n/C \le 0.5 \\ DP^{pov} + 2l/(C + \sigma n) & \text{if} \quad 0.5 < \sigma n/C \le 1.0 \end{cases}$$

<u>Access- Multiway tree</u> The formulas, similar to those used for the case with overflow, are:

$$IP^{pov} = log_f(1.5n/(C + \sigma n)) + (DP^{pov} - 1)\frac{(1 - (C/n + \sigma)/1.5)}{(f - 1)}$$

$$IrP^{pov} = log_f(1.5n/(C + \sigma n)) + (DrP^{pov} - 1)\frac{(1 - (C/n + \sigma)/1.5)}{(f - 1)}$$

with the understanding that DP^{or} (DrP^{or}) will not consider overflow buckets.

Cost for Point and Range Queries Based on the previous paragraphs, the formulas for the partition Fieldtree with overflow pages and indexed by a multiway tree can be rendered as:

$$TP^{por} = 1 + log_f(\frac{1.5n}{C+\sigma n}) + \begin{cases} \frac{2\sigma n}{C+\sigma n} \frac{(f-(C/n+\sigma)/1.5)}{f-1} & \text{if} \quad 0 < \sigma n/C \leq 0.5 \\ \frac{4\sigma n}{C+\sigma n} \frac{(f-(C/n+\sigma)/1.5)}{f-1} & \text{if} \quad 0.5 < \sigma n/C \leq 1.0 \end{cases}$$

$$TrP^{por} = 1 + log_f(\frac{1.5n}{C+\sigma n}) + \begin{cases} \frac{n(2sigman+1.5l)}{C+\sigma n} \frac{(f-(C/n+\sigma)/1.5)}{f-1} & \text{if} \quad 0 < \sigma n/C \leq 0.5 \\ \frac{n(4\sigma+2l)}{C+\sigma n} \frac{(f-(C/n+\sigma)/1.5)}{f-1} & \text{if} \quad 0.5 < \sigma n/C \leq 1.0 \end{cases}$$

The formulas the partition Fieldtree with overflow pages and indexed by internal pointers will now be obtained:

A point query will be solved by a total of $log_2(n\theta/C$ acceses; if $0.5 \leq \sigma n/C < 1$, a fraction $(1-\omega)$ of the queries will require accessing an extra page, so:

$$TP_p^{por} = \begin{cases} log_2(n/(C+\sigma n)) & \text{if} \quad 0 < \sigma n/C \leq 0.5 \\ log_2(n/(C+\sigma n)) + 2\sigma n/(C+\sigma n) & \text{if} \quad 0.5 < \sigma n/C \leq 1.0 \end{cases}$$

A range query, besides the effort needed for a point query, will need to cover $1 + \theta nl/C$ leaves and hence, to traverse $\theta nl/2C$ intermediate fields; one of them should have been previously visited during the first record access; thus

$$TrP_p^{por} = \begin{cases} log_2(n/(C+\sigma n)) + (1.5nl)/(C+\sigma n) & \text{if} \quad 0 < \sigma n/C \leq 0.5 \\ log_2(n/(C+\sigma n)) + (2nl+2\sigma)/(C+\sigma n) & \text{if} \quad 0.5 < \sigma n/C \leq 1.0 \end{cases}$$

3.4 Comparison of the results

We will consider a case taken from [Faloutsos 1987], where the population is of 100,000 objects, the fan-out factor f is 50, and the field capacity C is of 50 objects.

For the different trees considered Fig. 4 compares cost of a point query as a function of the parameter $\sigma n/C$. The Fieldtree versions that utilize internal pointers have been omitted since the comparison is very disfavorable to them.

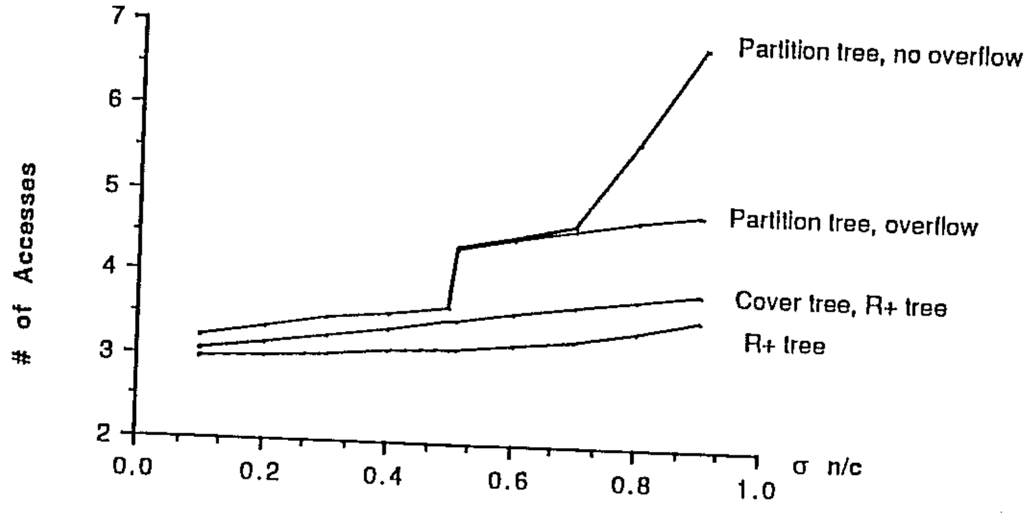


Fig 4 Comparison for 1-d Point Queries

From the previous figure can be noted that:

- As expected, behavior of the Fieldtree with a multiway index is always superior to the one that contains pointers inside the fields.
- As reported in [Faloutsos 1987] the R^+ tree outperforms the Rtree.
- The Cover Fieldtree and the Rtree perform identically.
- The Cover Fieldtree outperforms both variants of the partition Fieldtree. The variant of the partition Fieldtree that provides overflow record outperforms its alternate variant.

Fig. 5. shows the average number of pages needed to access C objects in a range query as a function of the size of the query window. Three sets of graphs are provided, corresponding to values of $\sigma n/C$ of 0.1, 0.3 and 0.5.

That figure shows that :

- For the one population Fieldtree case, implementation by multiway trees outperforms that of pointers interior to the fields.
- Rtrees and Cover Fieldtrees that use multiway indexes again behave identically. They outperform all other varieties.

- R^+trees are outperformed by partition Fieldtrees for high values of $\sigma n/C$

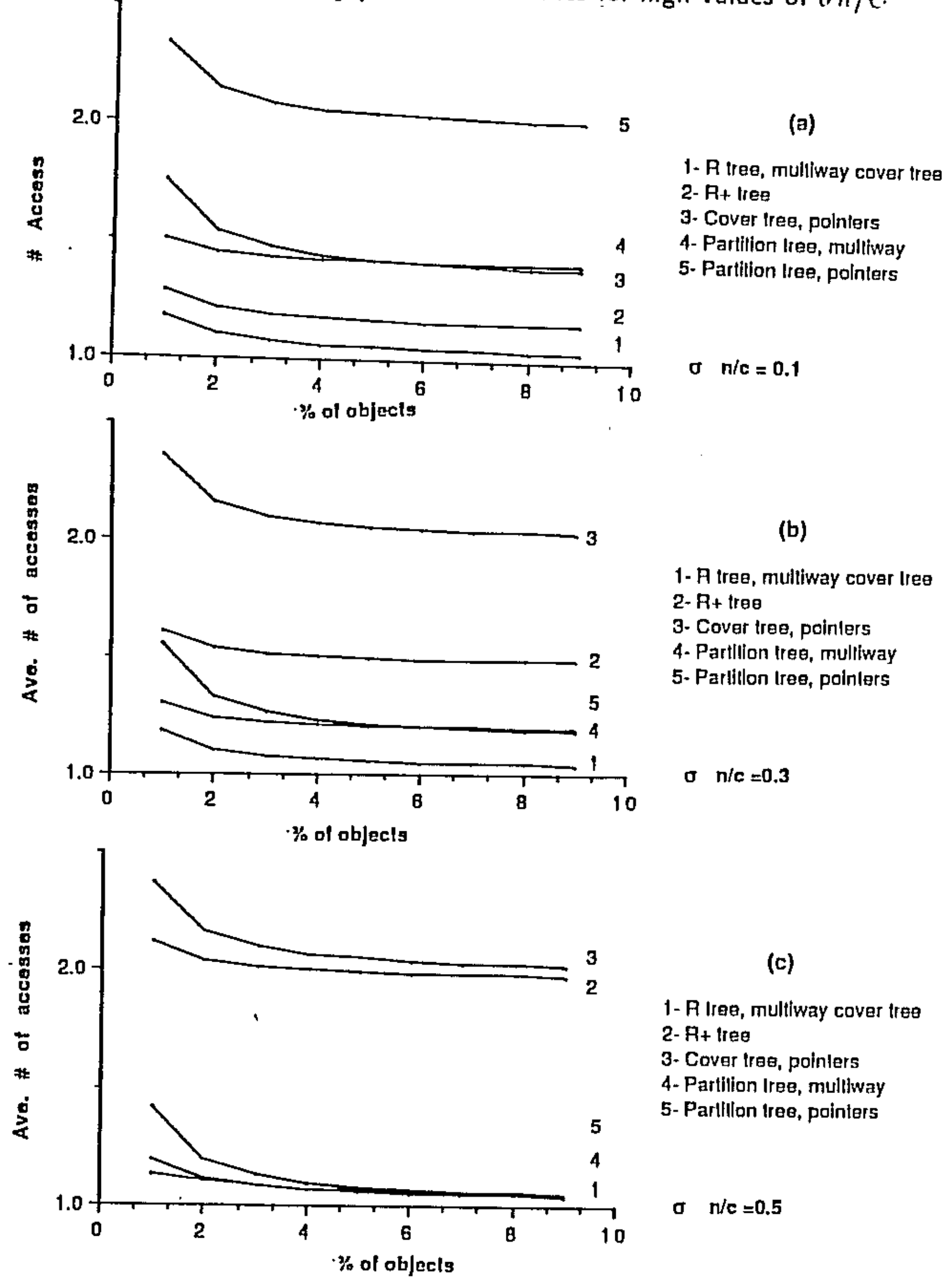


Figure 5: Comparison of Range Query performance

4 Two Dimensional Behavior of the Fieldtree

In the one dimensional environment, the Cover Fieldtree showed a behavior superior to that of the Partition Fieldtree both for point and for range queries. In a two dimensional case, a point query in and $4(\sigma\phi)^2$. The sum of the product of the factors times the number of field visited gives the number of data records in a point query:

$$\phi = \frac{\sqrt{n/C}}{1+\sigma(1+\sqrt{n/C})} \approx \frac{\sqrt{n/C}}{1+\sigma\sqrt{n/C}}$$

$$D_{2d}P = 1 + \frac{4\sigma\sqrt{n/C}}{1+\sigma\sqrt{n/C}} \qquad (13)$$

4..2 Cover tree- 2 dimensions

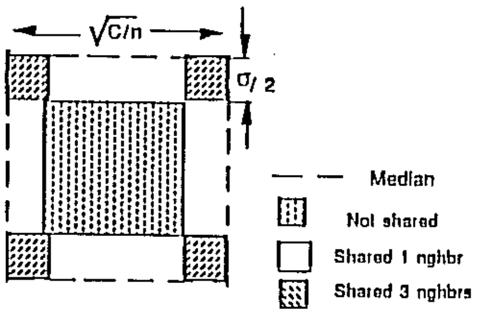


Fig 8) Loci of centers of objects shared with other fields

The extension of a field to have full occupancy is simply

$$2^{-h} = \sqrt{C/n}$$

Fig. 8 shows the regions the regions in space where a point query for a cover Fieldtree needs will meet one, two or four fields. A portion of the queries will be satisfied with accessing one field, while fractions will require respectively two and four fields. Adding up, the following formula results:

$$D_{2d}C = 1 + 2\sigma\sqrt{n/C} \qquad (14)$$

Formula (14) always renders less acceses than (13) for $0 \leq \sigma\sqrt{n/C} \leq 1.0$. Thus we can conclude that the cover tree is still superior to the partition tree for two dimensions.

5 Two Populations Results: Rtree vs. Cover Fieldtree

Faloutsos et al. [Faloutsos 1987] consider the case of a 1 dimensional Rtree with two populations of objects. The number and the extension of the objects in each population are called (σ_1, n_1), (σ_2, n_2).

That reference also proves that in the case of two population of objects one of them dominates, i.e.: all formulas are the same as if they were from a single population with the dominant characteristics.

If n, C_1, C_2 are defined as $n = n_1 + N_2$, $C_1 = Cn_1/n$, $C_2 = Cn_2/n$, then the population dom that dominates is:

$$dom = \begin{cases} 1 \text{ if} & \sigma_{1,parent,1} > \sigma_{2,parent,1} \\ 2 & \text{otherwise} \end{cases}$$

where $sigma_{i,parent,1} = \frac{1+\sigma_i}{n_i+1}(C_i - 1) + \sigma_i$

In the cover Fieldtree two situations can happen:

1. The importances of both types of objects are the same and, thus, their items are placed in the same level of the Fieldtree. The formulas are calculated taking into account the greatest of σ_1, σ_2
2. The importance of both types of objects is different and they are placed in the different levels of the Fieldtree.

Situation (2) is particularly attractive for range queries since then both populations become independent; no coupling at all occurs for the case of pointers interior to the fields, and very litte in the case of multiway trees. In this later case, the coupling can be greatly diminished if the positional keys of the objects are ordered "by levels", i.e., keys of fields a given level of the Fieldtree preceding all those corresponding to descendant levels.

An example of i.e. the following: suppose that C and f are the same as above, and the existance of two populations such that

$$\begin{aligned} n_2 &= 16n_1 \\ n_1 + n_2 &= 10^5 \\ \sigma_1 &= 16 * \sigma_2 \\ \sigma_1 n_1 / C &= 0.2 \end{aligned}$$

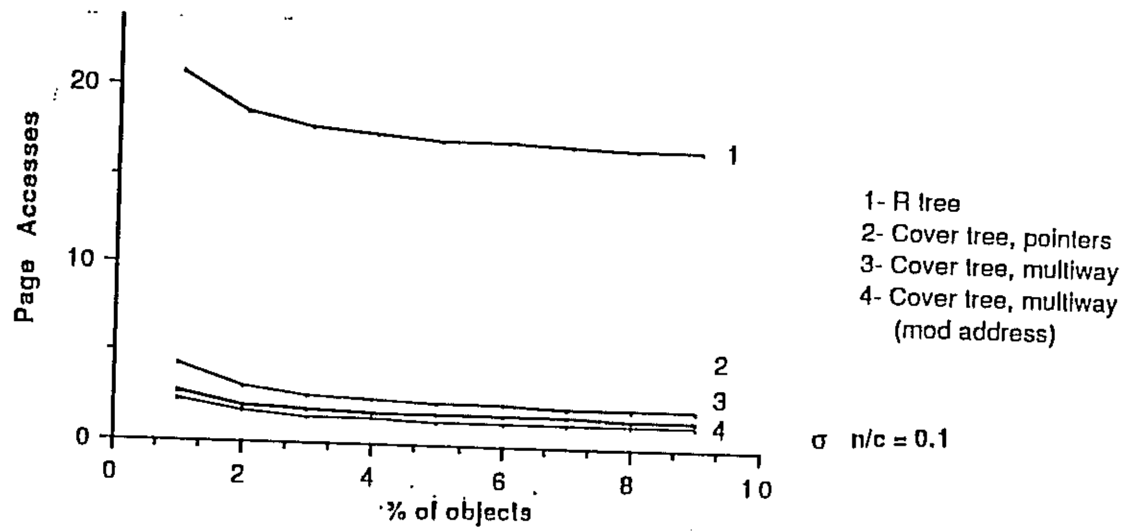


Fig 9) Ave. number of page accesses, "C" objects

Figure (9) shows the advantages of the Fieldtree for range query; it also shows that the superiority of the utilization multiway trees over that of interior pointer becomes less marked for mixed populations. Examples can be found, specially in the two dimensional case, where that superiority disappears.

6 Conclusions

The analysis reported in this paper demonstrates the following facts:

- The cover variety of the field tree behaves better than the partition variety for one and two dimensions.

- The cover Fieldtree behaves identically to the *R*tree for the single population case. It outperforms it for range queries in environments of mixed populations. The advantage is due to the capability of the Fieldtree of segregating objects according to their type, thus eluding the collection of objects of undesired type.

The behavior of an access method is very much dependent of its implementation; however. it seems that the Fieldtree is a good alternative to the *R*tree [Greene 1989] for cases where range queries are predominant and nesting and intersection of objects is common.

7 Acknowledgements

The authors wish to thank Max Egenhofer and Douglas Hudson for their discussions and their suggestions to improve this paper, and Jeff Jackson for his assistance in its preparation.

References

[Faloutsos 1987] Faloutsos et al. Analysis of Object-Oriented Spatial Access Methods. In: Proceedings of SIGMOD Conference, May 1987.

[Frank 1989] A. Frank and R. Barrera. The Fieldtree: A Data Structure for Geographic Information Systems. Technical Report, Department of Surveying Engineering, University of Maine, Orono, ME, March 1989. submitted for publication.

[Greene 1989] D. Greene. An Implementation and Performance Analysis of Spatial Data Access Methods. In: Proceedings IEEE Fifth International Conference on Data Engineering, Los Angeles, CA, February 1989.