Improvements of Approximate Algorithms for Distance-Based Queries

Antonio Corral[1], Joaquin Cañadas[1] & Michael Vassilakopoulos[2]

[1]Department of Languages and Computation, University of Almeria,
04120 Almeria, Spain. E-mail: acorral@ual.es, jjcanada@.ual.es. Fax: +34950015129

[2]Data Engineering Lab, Department of Informatics, Aristotle University,
GR-54006 Thessaloniki, Greece. E-mail: mvass@computer.org. Fax: +30310 996360

Abstract. In modern database applications the similarity or dissimilarity of complex objects is examined by performing distance-based queries (DBQs) on data of high dimensionality. The R-tree and its variations are commonly cited multidimensional access methods that can be used for answering such queries. Although, the related algorithms work well for low-dimensional data spaces, their performance degrades as the number of dimensions increases *(dimensionality curse)*. In order to obtain acceptable response time in high-dimensional data spaces, algorithms that obtain approximate solutions can be used. Approximation techniques, like N-consider (based on the tree structure), or α-allowance and ε-approximate (based on distance), can be applied in branch-and-bound algorithms for DBQs in order to control the trade-off between cost and accuracy of the result [CCV02]. In this paper, we improve the algorithms presented in [CCV02] by extending the use of approximation techniques at the leaf level of R-trees and by applying a combination of two such techniques in the same algorithm (hybrid approximate scheme). We investigate the performance of these improvements for the most representative DBQs (the K-nearest neighbors query and the K-closest pairs query) in high-dimensional data spaces, as well as the influence of the algorithmic parameters on the control of the trade-off between the response time and the accuracy of the result.

Keywords: Approximate Queries, Advanced Query Processing and Optimization, Spatial and Temporal Databases, High-dimensional data spaces, R-trees.

Βελτιώσεις Προσεγγιστικών Αλγορίθμων για Επερωτήσεις Βασισμένες στην Απόσταση

Antonio Corral[1], Joaquin Cañadas[1] & Μιχαήλ Βασιλακόπουλος[2]

[1]Department of Languages and Computation, University of Almeria,
04120 Almeria, Spain. E-mail: acorral@ual.es, jjcanada@.ual.es. Fax: +34950015129

[2]Εργαστήριο Τεχνολογίας & Επεξεργασίας Δεδομένων,
Τμήμα Πληροφορικής, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, 54006 Θεσσαλονίκη.
E-mail: mvass@computer.org. Fax: 0310 996360

Περίληψη. Στις σύγχρονες εφαρμογές Βάσεων Δεδομένων η ομοιότητα, ή μη σύνθετων αντικειμένων εξετάζεται εκτελώντας επερωτήσεις βασισμένες στην απόσταση (EBA) επί δεδομένων πολλών διαστάσεων. Το R-δέντρο και οι παραλλαγές του, συχνά αναφερόμενες πολυδιάστατες μέθοδοι προσπέλασης, μπορούν να χρησιμοποιηθούν για την απάντηση τέτοιων επερωτήσεων. Αν και, οι σχετικοί αλγόριθμοι αποδίδουν καλά για χώρους λίγων διαστάσεων, η απόδοσή τους μειώνεται σημαντικά καθώς αυξάνεται ο αριθμός των διαστάσεων (κατάρα των διαστάσεων). Για να επιτευχθεί αποδεκτός χρόνος απόκρισης σε χώρους πολλών διαστάσεων, μπορούν να χρησιμοποιηθούν προσεγγιστικοί αλγόριθμοι. Προσεγγιστικές τεχνικές, όπως η N-consider (βασιζόμενη στη δεντρική δομή), ή η α-allowance και η ε-approximate (βασιζόμενες στην απόσταση), μπορούν να εφαρμοστούν σε αλγόριθμους διακλάδωσης-και-φραγής για EBA, έτσι ώστε να ελεγθεί το αντιστάθμισμα μεταξύ κόστους και ακρίβειας του αποτελέσματος [CCV02]. Στο άρθρο αυτό, βελτιώνουμε τους αλγόριθμους που παρουσιάζονται στο [CCV02], επεκτείνοντας τη χρήση των προσεγγιστικών μεθόδων στο επίπεδο των φύλλων των R-δέντρων και εφαρμόζοντας συνδυασμό δύο τέτοιων τεχνικών στον ίδιο αλγόριθμο (υβριδικό σχήμα προσέγγισης). Εξερευνούμε τις επιδόσεις αυτών των βελτιώσεων για τις πιο χαρακτηριστικές EBA (την επερώτηση K-κοντινότερων γειτόνων και την επερώτηση K-εγγύτερων ζευγών) σε χώρους πολλών διαστάσεων, όπως επίσης και την επιρροή των αλγοριθμικών παραμέτρων στον έλεγχο του αντισταθμίσματος μεταξύ χρόνου απόκρισης και ακρίβειας του αποτελέσματος.

Λέξεις Κλειδιά: Προσεγγιστικές Επερωτήσεις, Βελτιστοποίηση και Επεξεργασία Επερωτήσεων, Βάσεις Χωρικών και Χρονικών Δεδομένων, Χώροι Δεδομένων Πολλών Διαστάσεων, R-δέντρα.

Improvements of Approximate Algorithms for Distance-Based Queries

Antonio Corral[1], Joaquin Cañadas[1] & Michael Vassilakopoulos[2]

[1]Department of Languages and Computation, University of Almeria,
04120 Almeria, Spain. E-mail: acorral@ual.es, jjcanada@.ual.es

[2]Data Engineering Lab, Department of Informatics, Aristotle University,
GR-54006 Thessaloniki, Greece. E-mail: mvass@computer.org

Abstract. In modern database applications the similarity or dissimilarity of complex objects is examined by performing distance-based queries (DBQs) on data of high dimensionality. The R-tree and its variations are commonly cited multidimensional access methods that can be used for answering such queries. Although, the related algorithms work well for low-dimensional data spaces, their performance degrades as the number of dimensions increases *(dimensionality curse)*. In order to obtain acceptable response time in high-dimensional data spaces, algorithms that obtain approximate solutions can be used. Approximation techniques, like N-consider (based on the tree structure), or α-allowance and ε-approximate (based on distance), can be applied in branch-and-bound algorithms for DBQs in order to control the trade-off between cost and accuracy of the result [CCV02]. In this paper, we improve the algorithms presented in [CCV02] by extending the use of approximation techniques at the leaf level of R-trees and by applying a combination of two such techniques in the same algorithm (hybrid approximate scheme). We investigate the performance of these improvements for the most representative DBQs (the K-nearest neighbors query and the K-closest pairs query) in high-dimensional data spaces, as well as the influence of the algorithmic parameters on the control of the trade-off between the response time and the accuracy of the result.

Keywords: Approximate Queries, Advanced Query Processing and Optimization, Spatial and Temporal Databases, High-dimensional data spaces, R-trees.

1 Introduction

Large sets of complex objects are used in modern applications (e.g. multimedia databases [FBF94], medical images databases [KSF96], etc). To examine the similarity or dissimilarity of these objects, high-dimensional feature vectors are extracted from them and organized in multidimensional indexes. The most important property of this feature transformation is that the feature vectors correspond to points in the multidimensional Euclidean space. Then, distance-based queries (DBQs) are applied on the multidimensional points. The most representative DBQs are the K-nearest neighbors query (K-NNQ) and the K-closest pairs query (K-CPQ). The K-NNQ discovers K distinct points from a point data set that have the K smallest distances from a query point. The K-CPQ discovers K distinct pairs of points formed from two point data sets that have the K smallest distances between them.

The multidimensional access methods belonging to the R-tree family (the R*-tree [BKS90] and particularly the X-tree [BKK96]) are considered good choices for indexing high-dimensional point data sets in order to perform DBQs. This is accomplished by branch-and-bound algorithms that employ distance functions and pruning heuristics based on MBRs (Minimum Bounding Rectangles), in order to reduce the search space. The performance of these algorithms degrades as the number of dimensions increases *(dimensionality curse)*. However, it can be improved if the search space is restricted somehow. In many situations, for practical purposes, approximate results, that can be obtained significantly faster than the precise ones, are usually as valuable as them. In order to obtain sufficiently good results quickly in high-dimensional data spaces, we can adopt one of the two following directions: (1) modifications of the index structures by including approximate information (the VA-file [WSB98]

keeps approximations of points and the A-tree [SYU00] is a tree structured index where the representation of the MBRs and data objects is based on approximations relative to their parents MBRs), or (2) modifications of the search algorithms [AMN98, CiP00, CCV02] by restricting the search space by some means (the most representative approximation techniques are the ε-approximate [AMN98, CiP00], α-allowance and N-consider [CCV02] ones). Here, we will focus on the second category, i.e. the development of approximate branch-and-bound algorithms for DBQs using tree-like structures belonging to the R-tree family.

The main objective of this paper is to study and improve the approximate branch-and-bound algorithms presented in [CCV02], the α-allowance and N-consider algorithms for K-NNQs and K-CPQs in high-dimensional data spaces, where point data sets are indexed in tree-like structures belonging to the R-tree family. These improvements consist in extending the application of approximation techniques at the leaf level of R-trees and in using a combination of two approximation techniques in the same branch-and-bound algorithm (hybrid approximate scheme = N-consider + α-allowance). We compare experimentally the new algorithms to the ones of [CCV02], in terms of the response time and the distance relative error with respect to the precise solutions, since these are the most important parameters taken into account by the users. Based on these results, we draw conclusions about the performance of the improved algorithms and examine the influence of the algorithmic parameters on the trade-off between accuracy and efficiency of each algorithm.

The paper is organized as follows. In Section 2, we review the literature and motivate the research reported in this paper. In Section 3, a brief description of the R-tree family, definitions of the most representative DBQs, an MBR-based distance function, a pruning heuristic and approximate branch-and-bound algorithms for DBQs are reviewed. In Section 4, improvements of the approximate algorithms are proposed. Moreover, in Section 5, a comparative performance study of these improvements is presented. Finally, the conclusions on the contribution of this paper and future work plans are summarized.

2 Background and Motivation

Numerous algorithms exist for answering DBQs. Most of these algorithms are focused on the K-NNQ over multidimensional access methods, from the incremental [HjS99] or non-incremental [RKV95] point of view. For the K-CPQ in spatial databases using R-trees, to the authors' knowledge, [HjS98, SML00, CMT00] are the only references in the literature. In [HjS98, SML00], incremental and iterative algorithms based on best-first search and priority queues were presented; and in [CMT00] non-incremental recursive and iterative branch-and-bound algorithms were presented for solving the K-CPQ in spatial databases. On the other hand, in [SSA97, KoS98, DiS01] similarity joins on high-dimensional point data sets, where two point sets of a high-dimensional vector space are combined such that the result contains all the point pairs which distance does not exceed a distance ε, have been developed. In [SSA97] an index structure (ε-kdB tree) and an algorithm for similarity self-join on high-dimensional points was presented. In [KoS98] the problem of computing high-dimensional similarity joins between two high-dimensional point data sets, where neither input is indexed (Multidimensional Spatial Join, MSJ), was investigated. In [DiS01] a new algorithm (Generic External Space Sweep, GESS), which introduces a rate of data replication to reduce the number of distance computations as an enhancement of MSJ, was proposed. Recently, in [BoK01] an analytical cost model and a performance study for the similarity join operation based on

indexes was proposed. In addition, a complex index architecture (Multipage Index, MuX) and join algorithm (MuX-join), which allows a separate optimization CPU time and I/O time, was presented.

In the approximate K-NNQ the user specifies a query point and a number K of answers to be reported. In contrast to the exact K-NNQ, the user is not strictly interested in the K nearest points, but only seeks for points which are not much far away from the query point than the K nearest ones. In [AMN98], an optimal algorithm for approximate nearest neighbor searching in a fixed dimension, using a multidimensional index structure with non-overlapped data regions that is stored in main memory (BBD-tree), was proposed. On the other hand, in [Cla94] the approximate closest-pair query is studied from the viewpoint of computational geometry and memory-based data structures. In addition, in metric tree indexes (M-tree [CPZ97]) the approximate nearest neighbor (ε-NNQ) has been examined [CiP00, ZSA98]. On one hand, in [CiP00] an algorithm, in which the error bound ε can be exceeded with a certain probability δ using information on the distance distribution of the query point, was proposed. On the other hand, in [ZSA98] three different approximation techniques over M-trees were proposed. These approaches provide a good cost performance, although the relative error is not bounded by any function of the input parameters.

In [CCV02], the application of three new approximation techniques (α-allowance, N-consider and M-consider) to recursive branch-and-bound algorithms for DBQs (K-NNQs and K-CPQs) using R-trees has been proposed. The N-consider (based on the structure of the access method) is a good approximate method for tuning the trade-off between cost and accuracy of the result and it is recommended when the users are not interested in a high accuracy, but in a fast answer. On the other hand, α-allowance is a distance-based approximate method, which gives a high accuracy of the query result, while the tuning of the trade-off cannot be easily controlled by the users. M-consider is an approximate method based on the number of subproblems generated by decomposition. It depends on the items examined during the whole execution of the respective exact algorithm and it was examined as a reference for comparing the other two approximate methods (α-allowance and N-consider), since it requires significant preprocessing to be executed.

Most of the previous efforts have been focused on the approximate K-NNQ using memory or disk-based metric data structures. For similarity joins involving two high-dimensional point data sets, reporting of the result requires excessive time, when the dimensionality increases [BoK01]. Here, our main objective is to investigate the behavior of DBQs, primarily K-CPQ, over high-dimensional point data sets that are indexed in tree-like structures belonging to the R-tree family (that are widely used in spatial databases) and to improve the approximate branch-and-bound algorithms proposed in [CCV02] by extending the approximation treatment at the leaf level and by using a hybrid approximate scheme that combines the N-consider and α-allowance approximation techniques.

3 Approximate Distance-Based Queries using R-trees

Recently, the high query execution cost of DBQs led to the quest for an approximate distance-based method that can be performed much faster than the precise one, at the expense of introducing a distance relative error of the result. Moreover, in practical situations and high-dimensional data spaces, the processing of exact DBQs is not always required by the users, and approximate answers suffice, especially when they are obtained at much lower

costs. A conventional approach to support DBQs in high-dimensional data spaces is to use data-partitioning index trees, like R-trees. Approximate branch-and-bound algorithms over R-tree indexes are useful methods for the processing of DBQs in acceptable response time and accuracy of the result.

3.1 The R-tree Family

R-trees [Gut84, GaG98] are hierarchical, height balanced multidimensional data structures. They are used for the dynamic organization of k-dimensional objects represented by their Minimum Bounding k-dimensional hyper-Rectangle (MBRs). An MBRs is determined by two k-dimensional points that belong to its faces, one that has the k minimum and one that has the k maximum coordinates (these are the endpoints of one of the diagonals of the MBR). Each R-tree node corresponds to the MBR that contains its children. The tree leaves contain pointers to the objects of the database, instead of pointers to children nodes. The nodes are implemented as disk pages. The rules obeyed by an R-tree are as follows.

- Leaves reside on the same level.
- Each leaf node contains entries of the form (MBR, Oid), such that MBR is the minimum bounding rectangle that encloses the spatial object determined by the identifier Oid.
- Every other node (internal) contains entries of the form (MBR, Addr), where Addr is the address of the child node and MBR is the minimum bounding rectangle that encloses MBRs of all entries in that child node.
- An R-tree of class (m, M) has the characteristic that every node, except possibly for the root, contains between m and M entries, where $m \leq \lceil M/2 \rceil$ (M and m are also called maximum and minimum branching factor or fan-out). The root contains at least two entries, if it is not a leaf.

Figure 1 depicts some rectangles on the left and the corresponding R-tree on the right. Dotted lines denote the bounding rectangles of the subtrees that are rooted in inner nodes.

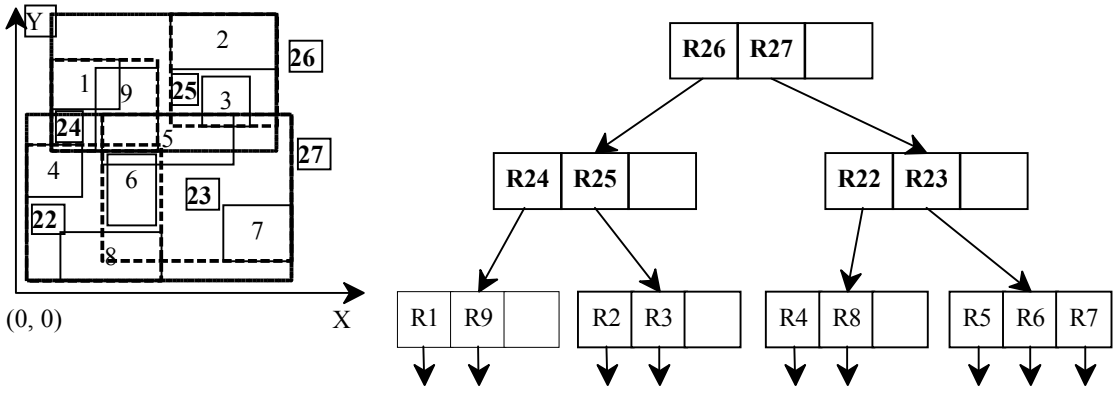

Figure 1. An example of an R-tree.

Many variations of R-trees have appeared in the literature (exhaustive surveys can be found in [GaG98, MTT99]). One of the most popular and efficient variations is the R*-tree [BKS90]. The R*-tree [BKS90] added two major enhancements to the original R-tree, when a node overflow is caused. First, rather than just considering the area, the node-splitting algorithm in R*-trees also minimizes the perimeter and overlap enlargement of

the MBRs. Second, an overflown node is not split immediately, but a portion of entries of the node is reinserted from the top of the R*-tree (forced reinsertion).

The X-tree [BKK96] is another variation that avoids splits that could result in high degree of overlap of MBRs in the internal R*-tree nodes. Experiments showed that the overlap of MBRs increases significantly for high dimensional data, resulting in performance deterioration in the R*-tree. Instead of allowing splits that produce high degree of overlaps, the nodes in X-tree are extended to a larger than the usual node size, resulting in the so-called *supernodes*. In order to find a suitable split, the X-tree also maintains the *history* of the previous split. The main objective of the overlap-minimizing split and *supernodes* is to keep the internal nodes as hierarchical as possible, and at the same time to avoid splits in the internal nodes that would result in a high overlap. Thus, a balance between overlap (that deteriorates performance) and large *supernodes* (thet result to a linear scan) is considered using a constant value (MAX_OVERLAP). Experiments, [BKK96], showed that X-tree outperforms R*-tree for point query and nearest neighbor queries in high-dimensional data spaces

3.2 Distance-Based Queries

Let us consider points in the k-dimensional data space ($D^{(k)} = \Re^k$) and a distance function of a pair of these points. A general distance function is the L_t-distance *(d_t),* or Minkowski distance between two points p and q in $D^{(k)}$ ($p = (p_1, p_2, \ldots, p_k)$ and $q = (q_1, q_2, \ldots, q_k)$) that is defined by:

$$d_t(p,q) = \left(\sum_{i=1}^{k} |p_i - q_i|^t \right)^{1/t}, \text{ if } 1 \le t < \infty \text{ and } \quad d_\infty(p,q) = \max_{1 \le i \le k} |p_i - q_i|$$

For t = 2 we have the Euclidean distance and for t = 1 the Manhattan distance. They are the most known L_t-distances. Often, the Euclidean distance is used as a distance function, but, depending on the application, other distance functions may be more appropriate. The k-dimensional Euclidean space, $E^{(k)}$, is the pair ($D^{(k)}$, d_2). In the following, we will use d instead of d_2. The most representative distance-based queries in $E^{(k)}$ are the following:

Definition. K-nearest neighbors query

Let P be a point data set ($P \neq \varnothing$) in $E^{(k)}$. Then, the result of the K-nearest neighbors query with respect to a query point q is the set K-NNQ(P, q, K) which contains all the ordered sequences of K ($1 \le K \le |P|$) different points of P, with the K smallest distances from q:

$$\text{K-NNQ}(P, q, K) = \{(p_1, p_2, \ldots, p_K) \in P^K : p_i \neq p_j \; i \neq j \; 1 \le i,j \le K \; \textit{and}$$
$$\forall p_i \in P - \{p_1, p_2, \ldots, p_K\}, \; d(p_1, q) \le d(p_2, q) \le \ldots \le d(p_K, q) \le d(p_i, q)\}$$

Definition. K-closest pairs query

Let P and Q be two point data sets ($P \neq \varnothing$ and $Q \neq \varnothing$) in $E^{(k)}$. Then, the result of the K-closest pairs query is the set K-CPQ(P, Q, K) which contains all the ordered sequences of K ($1 \le K \le |P| \cdot |Q|$) different pairs of points of $P \times Q$, with the K smallest distances between all possible pairs of points that can be formed by choosing one point of P and one point of Q:

$$\text{K-CPQ}(P, Q, K) = \{((p_1, q_1), (p_2, q_2), ..., (p_K, q_K)) \in (P \times Q)^K :$$
$$(p_i, q_i) \neq (p_j, q_j) \ \ i \neq j \ \ 1 \leq i,j \leq K \ \ \textit{and} \ \ \forall (p_i, q_j) \in P \times Q - \{(p_1, q_1), (p_2, q_2), ..., (p_K, q_K)\},$$
$$d(p_1, q_1) \leq d(p_2, q_2) \leq \ldots \leq d(p_K, q_K) \leq d(p_i, q_j)\}$$

Note that, due to ties of distances, the result of K-NNQ and K-CPQ may not be a unique ordered sequence. The aim of the exact algorithms for these queries is to find one of the possible instances, although it would be straightforward to obtain all of them.

3.3 MBR-based Distance Function and Pruning Heuristic

Usually, DBQs are executed using some kind of multidimensional index structure such as R-trees. If we assume that the point datasets are indexed on any tree structure belonging to the R-tree family, then the main objective while answering these types of queries is to reduce the search space. In [CMT00], a generalization of the function that calculates the minimum distance between points and MBRs (MINMINDIST) was presented. We can apply this distance function to pairs of any kind of elements (MBRs or points) stored in R-trees during the computation of branch-and-bound algorithms based on a pruning heuristic for DBQs. MINMINDIST(M_1, M_2) calculates the minimum distance between two MBRs M_1 and M_2. If any of the two (both) MBRs degenerates (degenerate) to a point (two points), then we obtain the minimum distance between a point and an MBR [RKV95] (between two points).

Definition. MINMINDIST(M_1, M_2)
Given two MBRs $M_1 = (a, b)$ and $M_2 = (c, d)$, in $E^{(k)}$,

$M_1 = (a, b)$, where $a = (a_1, a_2, ..., a_k)$ and $b = (b_1, b_2, ..., b_k)$ such that $a_i \leq b_i \ \forall 1 \leq i \leq k$

$M_2 = (c, d)$, where $c = (c_1, c_2, ..., c_k)$ and $d = (d_1, d_2, ..., d_k)$ such that $c_i \leq d_i \ \forall 1 \leq i \leq k$

we define *MINMINDIST(M_1, M_2)* as follows:

$$MINMINDIST(M_1, M_2) = \sqrt{\sum_{i=1}^{k} y_i^2}, \quad \text{such that} \quad y_i = \begin{cases} c_i - b_i, & if \ \ c_i > b_i \\ a_i - d_i, & if \ \ a_i > d_i \\ 0, & otherwise \end{cases}$$

The general pruning heuristic for DBQs over R-trees is the following: *"if MINMINDIST(M_1, M_2) > z, then the pair of MBRs (M_1, M_2) will be discarded"*, where z the distance value of the K-th nearest neighbor (K-NNQ) or the K-th closest pair (K-CPQ) that have been found so far.

3.4 Approximate Algorithms for Distance-Based Queries using R-trees

The most representative approximation techniques are the following:

- **ε-approximate method** [AMN98, CiP00]. Given any positive real ε (ε > 0) as maximum distance relative error to be tolerated, the result of a DBQ (K-NNQ or K-CPQ) is (1 + ε)-approximate if the distance of its i-th item is within relative error ε (or a factor (1 + ε)) of the distance of the i-th item of the exact result of the same DBQ, 1≤i≤K. For example, an (1+ ε)- approximate answer to the K-CPQ is an ordered sequence of K distinct pairs of points $((p_1', q_1'), (p_2', q_2'), ..., (p_K', q_K')) \in (P \times Q)^K$, such that (p_i', q_i') is the (1 + ε)-approximate closest pair of the i-th closest pair (p_i, q_i) of the exact result $((p_1, q_1), (p_2, q_2), ..., (p_K, q_K)) \in (P \times Q)^K$, that is $(d(p_i', q_i') - d(p_i, q_i)) / d(p_i, q_i) \leq \varepsilon$, 1≤i≤K. When the pruning heuristic is applied on a branch-and-bound algorithm, an item X is discarded if $MINMINDIST(X) > z/(1+\varepsilon) \Leftrightarrow MINMINDIST(X) + (MINMINDIST(X)*\varepsilon) > z \Leftrightarrow MINMINDIST(X) > z - (MINMINDIST(X)*\varepsilon)$. The decrease of z depends on the value of *MINMINDIST(X)* multiplied by ε (an unbounded positive real). Note that for ε = 0, the approximate algorithm behaves as the exact version and outputs the precise solution.
- **α-allowance method** [CCV02]. When the pruning heuristic is applied, an item X is discarded if $MINMINDIST(X) + \alpha(z) > z$, where z (z ≥ 0) is the current pruning distance value (e.g. the distance of the K-th closest pair found so far, for K-CPQ) and $\alpha(z) \geq 0$ is an allowance function. Typical forms of $\alpha(z)$ are: $\alpha(z) = \beta$ (β is a non-negative constant), and $\alpha(z) = z*\gamma$ (γ is a constant with 0≤γ≤1). For the second case, $MINMINDIST(X) + (z*\gamma) > z \Leftrightarrow MINMINDIST(X) > z - (z*\gamma) \Leftrightarrow MINMINDIST(X) > z*(1 - \gamma)$. In this case, the decrease of z depends on γ (positive real in the interval [0, 1]). We obtain the exact algorithm when γ = 0.
- **N-consider method** [CCV02]. In this case, the approximate branch-and-bound algorithm only considers a specified portion, or percentage N (0<N≤1) of the total number of items examined by the respective exact algorithm, when visiting an internal node (K-NNQ), or a pair of internal nodes (K-CPQ). The exact version of the algorithm is obtained when N = 1.

There are many other approximation methods to apply in branch-and-bound algorithms for reporting the result quickly. For example, (1) to replace the complete distance computations in high-dimensional spaces by partial distance computations in a space with much lower dimensionality, choosing the more representative dimensions (relaxation method); (2) to extend the MBRs on the R-tree leaf nodes by a given distance in each dimension before computing the distance functions; (3) during the processing of the algorithm, to select randomly a number of candidates for searching (internal nodes), or for the result (leaf nodes) and report the best solutions obtained from such choices (random search method); etc. However, in this paper, we only focus on the search space reduction methods over tree-like structures as approximation techniques embedded in non-incremental branch-and-bound algorithms for DBQs.

The previous MBR-based distance function, pruning heuristic and approximation techniques can be embedded into approximate branch-and-bound algorithms for DBQs that are applied over indexes belonging to the R-tree family and obtain sufficiently good approximate results quickly. In order to design approximate branch-and-bound algorithms for processing K-NNQs or K-CPQs in a non-incremental way (K must be known in advance), an extra data structure that holds the K nearest neighbors or closest pairs, respectively, is necessary. This data structure, so called *K-heap*, is organized as a maximum binary heap [CMT00]. It is possible to apply the three approximation techniques to incremental and non-incremental branch-and-bound algorithms (iterative, or recur-

sive) for DBQs (K-NNQ, or K-CPQ) using tree-like structures belonging to the R-tree family. In the following, due to space limitations, only three such applications, one for each approximation technique, are demonstrated.

For the **ε-approximate** method ($\varepsilon > 0$), we are going to design an iterative and non-incremental approximate branch-and-bound algorithm (following a best-first search) for K-NNQ (KNNI) between a set of points P stored in an R-tree (R_P) and a query point q (z is the distance value of the K-th nearest neighbor found so far and at the beginning $z = \infty$). For the iterative approach, we need a minimum binary heap, H, of references to nodes of the R-tree <MINMINDIST, $Addr_p$>, which are kept ordered by increasing values of MINMINDIST; the item with the minimum MINMINDIST is visited first. The algorithm can be described by the following steps:

KNNI1 Start from the root of the R-tree and initialize the minimum binary heap H.

KNNI2 If you access an internal node, then calculate MINMINDIST for each possible M_i. Insert into the minimum binary heap H, those references to nodes of MBRs having $MINMINDIST(M_i, q) \leq z/(1+\varepsilon)$.

KNNI3 If you access a leaf node, then calculate $MINMINDIST(p_i, q)$ between q and each possible point stored in the node. If this distance is smaller than or equal to z, then remove the root of the K-heap and insert the new point p_i, updating this structure and z.

KNNI4 If the minimum binary heap, H, is empty, then stop.

KNNI5 Get the item on top of the minimum binary heap <MINMINDIST, $Addr_p$>. If this item has $MINMINDIST > z/(1 + \varepsilon)$, then stop. Else, repeat the algorithm from **KNNI2** for this item.

For the **α-allowance** method ($0 \leq \gamma \leq 1$), the recursive and non-incremental approximate branch-and-bound algorithm (following a depth-first traversal) for processing the K-NNQ between a set of points P stored in an R-tree (R_P) and a query point q can be described by the following steps:

KNNR1 Start from the root of the R-tree.

KNNR2 If you access an internal node, then calculate $MINMINDIST(M_i, q)$ between q and each possible MBR M_i, and sort them in ascending order of MINMINDIST. Following this order, propagate downwards recursively only for those MBRs having $MINMINDIST(M_i, q) \leq z^*(1 - \gamma)$.

KNNR3 If you access a leaf node, then calculate $MINMINDIST(p_i, q)$ between q and each possible point stored in the node. If this distance is smaller than or equal to z, then remove the root of the K-heap and insert the new point p_i, updating this structure and z.

For the **N-consider** method ($0 < N \leq 1$), the recursive and non-incremental approximate branch-and-bound algorithm (following a depth-first traversal[1]) for processing the K-CPQ between two sets of points (P and Q) indexed in two R-trees (R_P and R_Q) with the same height can be described by the following steps [CMT00] (z is the distance value of the K-th closest pair found so far and at the beginning $z = \infty$):

KCPR1 Start from the roots of the two R-trees.

[1] This traversal gives higher priority to those R-tree nodes in larger depth. This has the effect that approximate solutions (although non-precise) are usually available even if the processing of the algorithm is stopped before its normal termination and that the pruning distance (z) is updated very quickly.

KCPR2 If you access two internal nodes, choose only a portion *Total'* = N*Total of all possible pairs of MBRs (Total) stored in the nodes, then calculate MINMINDIST(M_i, M_j) between such pairs of MBRs (Total'). Propagate downwards recursively only for those pairs of MBRs *(Total')* having MINMINDIST(M_i, M_j) ≤ z.

KCPR3 If you access two leaf nodes, then calculate MINMINDIST(p_i, q_j) of each possible pair of points. If this distance is smaller than or equal to z, then remove the root of the K-heap and insert the new pair of points (p_i, q_j), updating this structure and z.

The main advantage of the recursive branch-and-bound algorithms is that they transform the global problem in smaller local ones at each tree level and we can apply pruning heuristics over every subproblem for reducing the search space. Moreover, for improving the I/O and CPU cost of the recursive branch-and-bound algorithm for K-CPQ, two techniques are used. The first improvement aims at reducing the number of I/O operations, and it consists in a *Global LRU buffer* [CVM01]. The second enhancement aims at reducing the CPU cost by using the *distance-based plane-sweep technique* [PrS85] to avoid processing all possible combinations of pairs of R-tree items from two internal or leaf nodes.

The application of best-first search to the K-CPQ is also similar to the application to the K-NNQ. The minimum binary heap, H, would contain references to pairs of nodes of the two R-trees <MINMINDIST, $Addr_p$, $Addr_q$>. The best-first search is suitable for incremental K-CPQs [HjS98], where K is unknown in advance and the users want to browse the pairs of points in increasing order of distance. The incremental processing tends to consume an amount of resources that depends strongly on the number of required items K in the result (if we want to obtain a large or the entire distance join result, the performance of the incremental algorithm is penalized [HjS98]), because the number of subproblems generated by decomposition increases with the depth of the R-trees. Furthermore, part of the minimum binary heap must migrate to disk, incurring frequent disk accesses. Also, best-first search is not favored by page replacement policies (e.g. LRU, FIFO, etc.) [CVM01]. The incremental processing for 2-dimensional points was studied in [HjS98], and its disadvantages for high-dimensional data spaces could get worse, primarily, with respect to the size of the minimum binary heap. For this reason, the iterative and non-incremental approximate branch-and-bound algorithm for K-CPQ in high-dimensional data spaces is not considered in this paper.

The algorithmic parameters γ and N (varying in the range [0, 1]) can act as adjusters of the trade-off between efficiency of the respective algorithm and accuracy of the result. The case of the parameter ε is different, since it is an unbounded positive real (ε > 0) and the users cannot directly control the accuracy of the result. The ε-approximate method could be considered a method that allows controlling of this trade-off, only if suitable upper bounds are determined for the parameter ε. Such bounds depend on the data distribution and the dimensionality (and the *dimensionality curse*). We are currently in the process of performing extensive experimentation in this context. Since our target is to be able to control the above trade-off effectively and easily, in this paper, we use the α-allowance technique for the development of our hybrid method (that combines structure and distance based methods).

4 Improving Approximate Algorithms for DBQs using R-trees

The main problem of the approximate algorithms for DBQs using R-trees in high-dimensional data spaces is *not to waste time computing distances at the leaf level that will not significantly improve the result*. In this context we propose the following two improvements: (1) to use an approximation technique at the leaf level; and (2) to combine both approximation techniques (N-consider and α-allowance) in order to find a good trade-off between the cost and the accuracy of the result.

4.1 Approximate Treatment at the Leaf Level

In [CCV02], the above two approximation techniques (N-consider and α-allowance) are applied on branch-and-bound algorithms for DBQs only at the internal levels (see section 3.4), using two parameters (N_I and γ_I) varying in the range [0, 1] that can act as adjusters of the balance between cost (response time) and the accuracy of the result (distance relative error). If we want to extend the application of the approximation techniques at the leaf level, independently to the treatment at the internal level, we have to incorporate two new parameters ($0 < N_L \leq 1$ and $0 \leq \gamma_L \leq 1$). Since, the main target is to avoid distance computations at the leaf level, during which significant improvements could not be obtained, we are going to focus on N-consider technique. α-allowance is an approximation technique based on distances and it is affected by the *dimensionality curse*; hence, its application at the leaf level will not reduce the number of distance computations significantly and the response time will not be affected.

For the N-consider method we must take into account the following modifications of the previous branch-and-bound algorithms for K-NNQ and K-CPQ:

KNNQ If you access a leaf node, choose only a portion *Total'* = N_L*Total of all possible points (Total) stored in the node, then calculate MINMINDIST(p_i, q) between q and such points *(Total')*. If this distance is smaller than or equal to z, then remove the root of the K-heap and insert the new point p_i, updating this structure and z.

KCPQ If you access two leaf nodes, choose only a portion *Total'* = N_L*Total of all possible pairs of points (Total) stored in the nodes, then calculate MINMINDIST(p_i, q_j) between such pairs points *(Total')*. If this distance is smaller than or equal to z, then remove the root of the K-heap and insert the new pair of points (p_i, q_j), updating this structure and z.

4.2 Hybrid Use of the Approximation Techniques

It is possible to combine both approximation techniques (α-allowance and N-consider) in one approximate branch and-bound algorithm in order to improve the balance between the cost and the accuracy of the result. Such a combination would be interesting, because the N-consider technique (a structure-based approximate method) is appropriate for reducing the response time, whereas the α-allowance technique (distance-based approximate method) is recommended for obtaining a good accuracy of the result [CCV02]. Moreover, the approximation techniques can be applied to both internal and leaf levels.

This combination consists of two consecutive filters at internal levels of the R-tree. In the first filter (based on the structure), we adopt N-consider approximation technique, producing a set of candidates. Each candidate is then examined by the second filter (based on distance), using the α-allowance approximation technique. At the leaf level, since we try to avoid distance computations, which will not improve significantly the result, N-consider is adopted as approximation technique. As an example, the recursive and non-incremental hybrid approximate branch-and-bound algorithm for processing K-CPQ between two sets of points (P and Q) indexed in two R-trees (R_P and R_Q) with the same height can be described by the following steps (z is the distance value of the K-th closest pair found so far and at the beginning $z = \infty$):

KCPH1 Start from the roots of the two R-trees.

KCPH2 If you access two internal nodes, choose only a portion *Total'* = N_I*Total of all possible pairs of MBRs (Total) stored in the nodes, then calculate MINMINDIST(M_i, M_j) between such pairs of MBRs *(Total')*. Propagate downwards recursively only for those pairs of MBRs *(Total')* having MINMINDIST(M_i, M_j) $\leq z*(1 - \gamma_I)$.

KCPH3 If you access two leaf nodes, choose only a portion *Total'* = N_L*Total of all possible pairs of points (Total) stored in the nodes, then calculate MINMINDIST(p_i, q_j) between such pairs points (Total'). If this distance is smaller than or equal to z, then remove the root of the K-heap and insert the new pair of points (p_i, q_j), updating this structure and z.

We have included the α-allowance approximation technique as a second filter, since it can reduce the response time with a negligible increase of the *average distance relative error (ADRE).* In order to obtain ADRE, we calculate the exact result for the DBQ off-line, then apply the related approximate algorithm and calculate the average distance relative error of all the K items of the result. Note, that no important improvement would be achieved if we only considered only the N-consider method as a filter. Furthermore, we have to note that the merit of α-allowance method that the relative error is bounded by γ [CCV02], disappears when it is combined with N-consider.

5 Experimental Results

This section provides the experimental results that aim at measuring and evaluating the behavior of the exact and approximate recursive branch-and-bound algorithms for K-NNQ and K-CPQ using R*-trees and X-trees (MAX_OVERLAP=0.2) for synthetic point data sets following Uniform distribution in the range [0, 1] in each dimension. The data sets cardinality was equal to 100000, K=100, there was a global LRU buffer (256 pages) and several dimensions (d=2, 5, 10, 15, 20, 25 and 30) were tested. The query points for K-NNQ were generated randomly in the space of the data points stored in the indexes.

All experiments run on a Linux workstation with a Pentium III 450 MHz processor, 128 Mb of main memory, and several Gb of secondary storage, using the *gcc* compiler. The index page size was 4Kb (m=0.4*M), and the fan-out decreased when the dimensionality increased. We are going to compare the performance of the approximate branch-and-bound algorithms (recursive or iterative) based mainly on the total response time (cost of the query in seconds) and ADRE, since they are the most representative performance metrics taken into account by the users. The indexes construction was not taken into account for the total response time.

	K-NNQ (I/O)		K-CPQ (I/O)		K-NNQ (sec)		K-CPQ (sec)	
Dim	*R*-tree*	*X-tree*	*R*-tree*	*X-tree*	*R*-tree*	*X-tree*	*R*-tree*	*X-tree*
2	7	6	1481	1855	0.01	0.01	2.02	2.64
5	133	76	9266	24340	0.20	0.14	38.82	40.67
10	2977	2823	169770	1161576	1.53	1.22	2073.33	1354.68
15	4607	4528	836047	16627295	2.32	2.14	12407.73	12632.61
20	6064	5582	1418215	28594577	3.55	3.25	25518.45	26579.54
25	6992	6745	1579301	41174579	3.29	3.02	33405.19	35228.23
30	8579	7801	1519414	54045175	3.72	3.52	39737.53	42539.46

Table 1. Performance of the exact DBQs on R*-trees and X-trees, in terms of the number of disk accesses (I/O) and response time (sec).

From the results of the Table 1[2], it is apparent that the X-tree is the most appropriate index structure for the K-NNQ (this is in accordance to the results of [BKK96]). On the other hand, the results with respect to the K-CPQ show that the R*-tree is a better index structure than the X-tree (primarily, due to the absence of *super-nodes* that increase the number of backtracks at internal nodes, decreasing the search performance). The main objective of the R*-tree is to minimize the overlap between nodes at the same tree level (the less the overlap, the smaller the probability to follow multiple search paths). Therefore, the overlap plays an important role on K-CPQs. If we compare the results for K-CPQs with respect to the I/O activity and the response time, we can conclude that this query becomes extremely expensive as the dimensionality increases, in particular for values larger than 5, and we are going to focus on such dimensions. Therefore, the use of approximate branch-and-bound algorithms is justified for the K-CPQ, since they aim at obtaining acceptable results quickly.

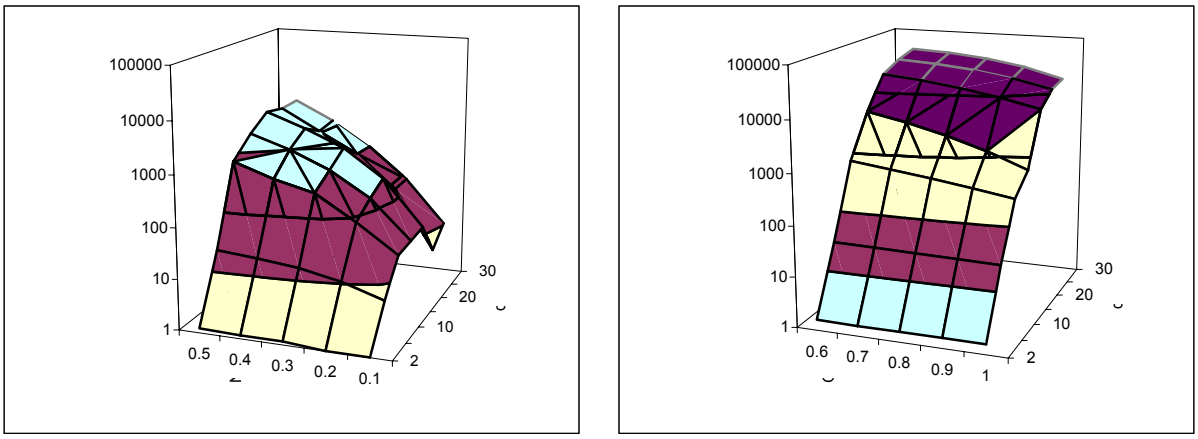

Figure 2. Response time (logarithmic scale) of the K-CPQ approximate algorithms using N-consider (left) and α-allowance (right) methods, as a function of the algorithmic parameters and the dimensionality.

In Figure 2[2], we compare the response time (logarithmic scale) of the approximation methods (N-consider and α-allowance) as a function of the values of approximation parameters, N and Gamma (γ), and the dimensionality. N-consider is considerably faster than α-allowance, mainly for dimensions larger than 5. For example, in the most expensive case (d=30, N=0.5 and Gamma=1), N-consider outperforms α-allowance by a factor of 13. The behavior of N-consider is very interesting, since with the increase of dimensionality, the response time can decrease (d=25). This is due to the fact that the number of considered items in the combination of two internal

[2]Table 1 and Figure 2 also appear in [CCV02]; they are included here for the completeness of the presentensation.

nodes is independent of the increase of dimensionality, and it depends strongly on the R-tree structures (maximum and minimum branching factors, heights, etc.).

In Figure 3 we illustrate one of the several diagrams that we created for studying the effect of the N_L and N_I values (the two approximation parameters of the approximate treatment at the leaf level) on the response time and ADRE. By studying these diagrams, we conclude that the ranges [0.2, 0.6] for N_I and [0.6, 1.0] for N_L are the most promising in order to obtain a good balance between cost and accuracy of the result. For example, we have observed that if we consider $N_I = 0.7$ and $N_L = 0.6$, then the reduction of N_I produces a better cost-accuracy balance than the reduction of N_L; hence low values of N_L are not good candidates to be considered.

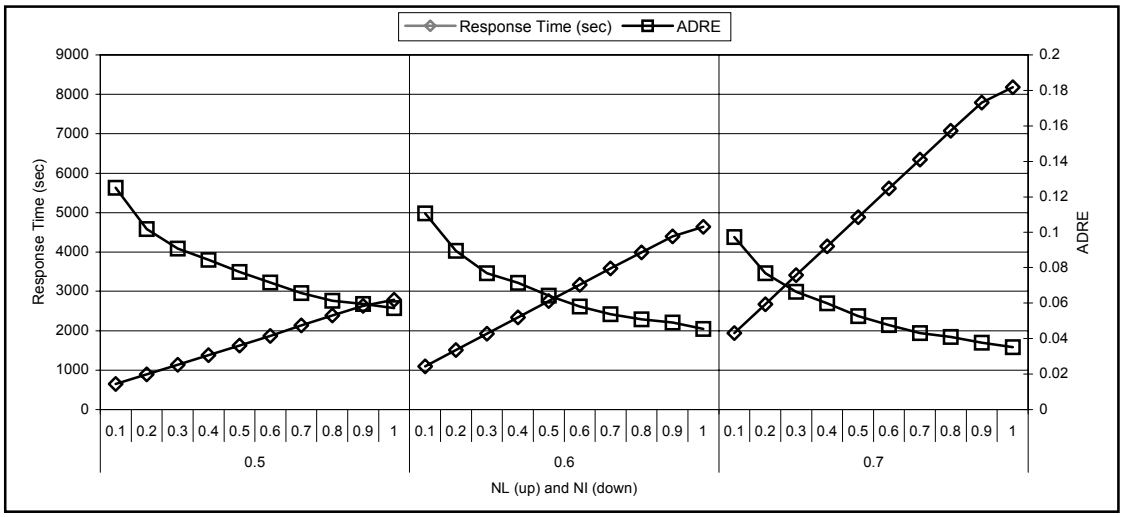

Figure 3. The effect of the two approximation parameters (N_L and N_I) of the approximate treatment at leaf level on the response time and ADRE. d = 25.

In order to illustrate the behavior of the approximate treatment at the leaf level of R-trees, in Figure 4, the response time (left) and ADRE (right) of the approximate K-CPQ algorithm using N-consider technique as a function of N_I (internal nodes) and N_L (leaf nodes) are depicted. In the left part of the Figure 4, we can observe the effect of increasing N_L for a given N_I with respect to the response time (the lower N_L, the faster the approximate algorithm is). In the right part of the same figure, the behavior is just the opposite with respect to ADRE (the higher N_L, the more accurate the approximate algorithm is). Finally, if we compare both charts, we can observe that the reduction of the response time with the decrease of N_L (for a given N_I value) is greater than the increment of ADRE.

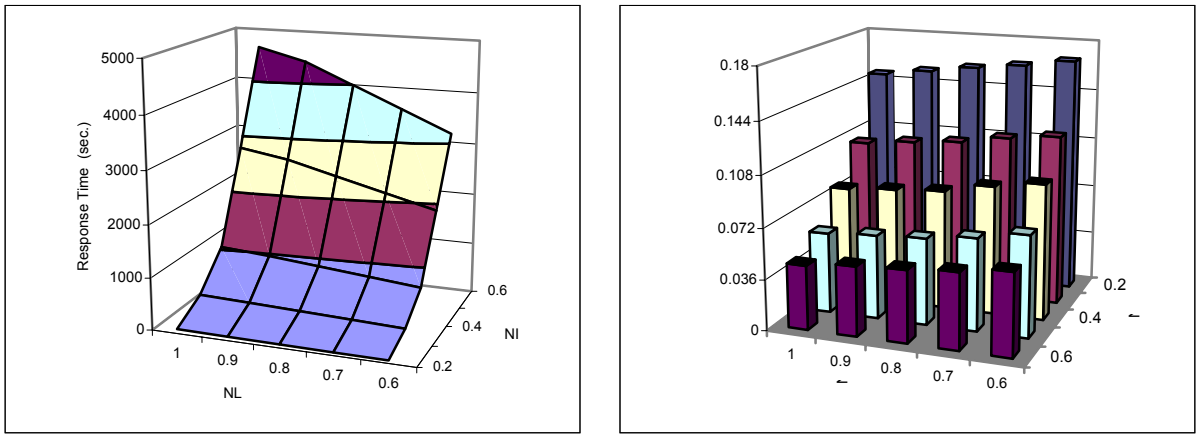

Figure 4. Response time (left) and ADRE (right) of the K-CPQ approximate algorithm using the N-consider technique, as a function of the approximation parameters N_I (internal nodes) and N_L (leaf nodes). d = 25.

In Figure 5, we compare the evolution of the response time and ADRE for the K-CPQ hybrid approximate branch-and-bound algorithm (KCPH), when N_L and $Gamma_I$ (γ_I) vary in the range [0.6, 1.0], for N_I = 0.4. We must highlight that the trends for N_I values in the range [0.2, 0.6] were similar. From such a figure, we can observe that the behavior of KCPH is very similar for a given pair (N_I, N_L) with the variation of $Gamma_I$ in [0.6, 1.0]: *the higher $Gamma_I$ value, the faster the approximate algorithm is, although their ADRE values increase by a negligible quantity*. For example, when N_I = 0.4 and N_L = 1.0, the algorithm becomes faster by a factor of 2.72, if we consider its response time performance for $Gamma_I$ = 0.6 (993.12 sec) and $Gamma_I$ = 1.0 (365.57 sec), although the increase of ADRE is only 0.4307% (0.079791 for $Gamma_I$ = 0.6 and 0.084098 for $Gamma_I$ = 1.0). Therefore, for the K-CPQ hybrid approximate algorithm (KCPH), large values of $Gamma_I$ are good choices to be considered.

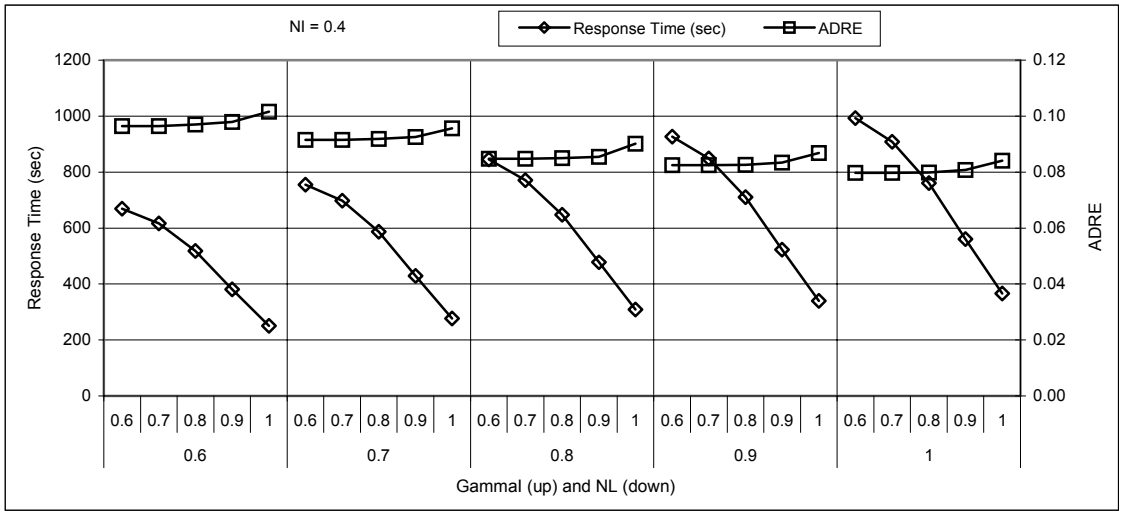

Figure 5. Response time versus ADRE for the K-CPQ hybrid approximate algorithm (KCPH), as a function of N_L and $Gamma_I$ (γ_I) in the range [0.6, 1.0], for N_I = 0.4. d = 25.

Figure 6 depicts the experimental results of the hybrid approximate scheme (the first filter is N-consider and the second filter is α-allowance at internal levels and only one filter, N-consider, is applied at the leaf level). In these two diagrams, we have fixed the $Gamma_I(\gamma_I)$ value (1.0 in the left and 0.9 in the right diagram) according to the conclusions obtained from Figure 5. The other parameters (N_I and N_L) vary in their most representative ranges ($0.2 \leq N_I \leq 0.6$ and $0.6 \leq N_L \leq 1.0$). By observing both diagrams, we conclude that $Gamma_I = 1.0$ provides the fastest response time. The explanation of this behavior is due to the fact that after choosing a portion of the items involved in the combination of two internal nodes (N_I) in the first filter, the hybrid approximate algorithm in the second filter chooses only the pairs of MBRs with MINMINDIST = 0 ($Gamma_I = 1.0$); besides it only considers a portion of the items involved in the combination of two leaf nodes (N_L).

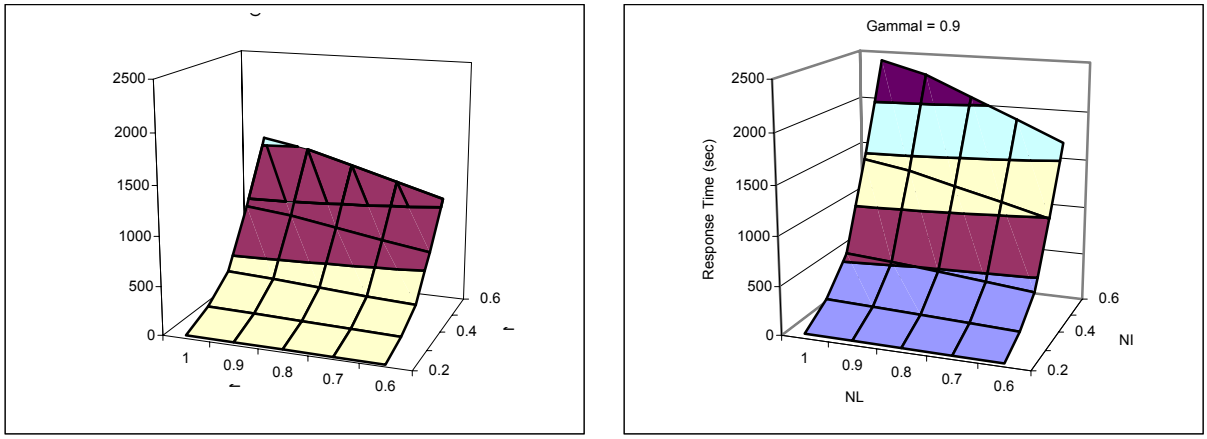

Figure 6. Response time for the K-CPQ hybrid approximate algorithm (KCPH), as a function of N_L in the range [0.6, 1.0] and N_I in the range [0.2, 0.6], for $Gamma_I(\gamma_I) = 1.0$ (left) and $Gamma_I(\gamma_I) = 0.9$ (right). d = 25.

In Figure 7, the ADRE for the same values of the approximation parameters (N_I, N_L and $Gamma_I$) using the hybrid approximate algorithm (KCPH) is depicted. We can observe that the differences of the ADRE values are almost negligible, and both charts follow the same tendency with very similar performance values. This means that $Gamma_I = 1.0$ is the best value for our hybrid approximate scheme with respect to the balance between cost (response time) and accuracy of the result (ADRE). For example, if we select the most "expensive" case depicted in Figures 6 and 7 ($N_I = 0.6$, $N_L = 1.0$) and consider $Gamma_I = 0.9$ or $Gamma_I = 1.0$, we observe that the algorithm becomes faster by a factor of 1.54 (2410.29 sec for $Gamma_I = 0.9$ and 1569.73 sec $Gamma_I = 1.0$), although the increase of ADRE is 0.3561% (0.045669 for $Gamma_I = 0.9$ and 0.049230 for $Gamma_I = 1.0$).

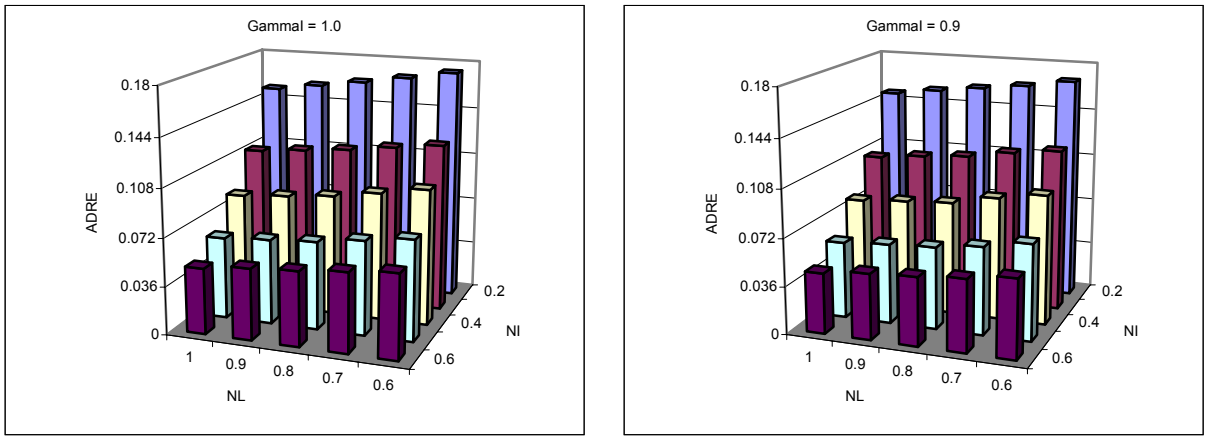

Figure 7. ADRE for the K-CPQ hybrid approximate algorithm (KCPH), as a function of N_L in the range [0.6, 1.0] and N_I in the range [0.2, 0.6], for $Gamma_I$ (γ_I) = 1.0 (left) and $Gamma_I$ (γ_I) = 0.9 (right). d = 25.

In Figure 8, a comparison of the response time (left) and ADRE (right) of four approximate algorithms and the exact algorithm is depicted (note that the ADRE for the exact algorithm is always 0). The approximate algorithms follow the α-allowance (Alpha), the N-consider (N-In), the treatment at the leaf level (N-Le) and the Hybrid techniques. In order to compare the related approximate algorithms, using the conclusions drawn from the previous diagrams and the conclusions of [CCV02], we chose the following parameter values for each method. *Alpha*: Gamma = value on x-axis; *N-In*: N_I = 0.5 + value on x-axis (i.e. N_I = 0.1, 0.2, 0.3, 0.4, and 0.5); *N-Le*: N_I =0.4, N_L = value on x-axis; *Hybrid*: $Gamma_I$ =1.0, N_I =0.4, N_L =value on x-axis. It can be seen that the Hybrid technique has the most stable balance between cost and accuracy of the answer for the configuration: N_I = 0.4, $Gamma_I$ = 1.0 and N_L varying in the range [0.6, 1.0]. For instance, the hybrid approximate scheme (N_I = 0.4, $Gamma_I$ = 1.0, N_L = 0.9) outperforms N-Le (N_I = 0.4, N_L = 0.9) by a factor of 2.85 and N-In (N_I = 0.4) by a factor of 3.07. On the other hand, the increase of ADRE for Hybrid is a 0.44% with respect to N-Le and 0.71% with respect N-In**.** Moreover, if we compare the behavior of the hybrid scheme with respect to the "pure" α-allowance technique (Alpha), the former is in average 68 times faster than the latter (response time in logarithmic scale), but this reduction of the response time results in an increase of ADRE (reduction of the accuracy of the query result) by 9% on average. In other words, we observe a huge reduction of the response time for just a small of the ADRE value.

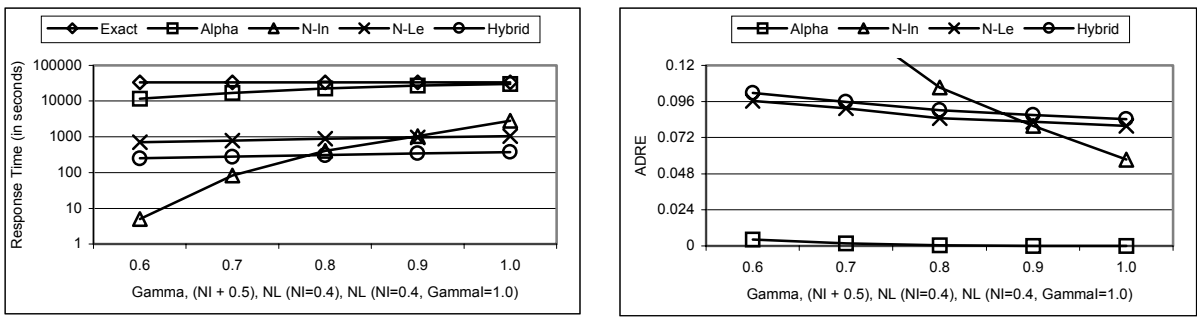

Figure 8. Comparison of the response time in logarithmic scale (left) and ADRE (right) of the exact algorithm and four approximate algorithms. d = 25.

The most important conclusions from the experimentation are the following:

- The best index structure for K-NNQ is the X-tree. However, for K-CPQ, the R*-tree outperforms the X-tree with respect to the I/O activity and the response time. Moreover, according to [CCV02], N-consider (a structure-based approximation technique) is the best method for tuning the search and finding a good balance between cost (response time) and accuracy of the result (ADRE).
- The N-consider technique is the most appropriate to extend the approximate treatment at the leaf level, since it can easily avoid distance computations at the leaf level that will not significantly improve the result of DBQs.
- The hybrid approximate scheme, a combination of N-consider and α-allowance approximation techniques in the same algorithm (KCPH), is an important improvement, since the candidates selected by the N-consider technique are filtered by the α-allowance method. It produces important reductions in response time with a small increase of ADRE. Moreover, KCPH provides a stable balance between cost and accuracy of the result and it outperforms the simpler versions of approximate algorithms [CCV02].
- For our hybrid approximate algorithm (KCPH), we have found a set of values for the approximate parameters (N_I, $Gamma_I$ and N_L) useful for the users in order to find a good balance between cost and accuracy of the result: $0.2 \leq N_I \leq 0.6$, $Gamma_I = 1.0$ and $0.6 \leq N_L \leq 1.0$. The other values have not been included for the following reasons: (1) $N_I > 0.6$, the approximate algorithms are very slow although their results are very close to the exact ones; (2) $Gamma_I < 1.0$, the results are more accurate but the response time is higher; and (3) $N_L < 0.6$, the approximate algorithms are faster but the inexactness of the result is higher.

6 Conclusions and Future Work

In this paper we have improved the approximate branch-and-bound algorithms for DBQs over R-trees indexing high-dimensional data [CCV02] by extending the approximate treatment at the leaf level and by using a combination of two approximation methods in the same algorithm (hybrid approximate scheme = N-consider + α-allowance). This hybrid approximate algorithm (KCPH) is easily adjustable by the users who wish to find a good trade-off between the response time and the accuracy of the answer. The most important conclusions drawn from our study and the experimentation are the following:

- N-consider (structure-based approximation technique) is the best method for tuning the search and finding a good balance between cost (response time) and accuracy of the result (ADRE). This adjustment can be achieved by using the parameter N in the range (0, 1]. Moreover, this approximation technique is the most suitable for extending the approximate treatment at the leaf level, since it can easily avoid distance computations at the leaf level, during which significant improvements in the result of DBQs could not be obtained.
- The hybrid approximate scheme is a combination of N-consider and α-allowance approximation techniques in the same algorithm. In this new approximate algorithm (KCPH), candidates selected by the N-consider technique are filtered by the α-allowance method. We have included the α-allowance method as

a second filter, since it can reduce the response time with a negligible increase of the ADRE. Moreover, KCPH outperforms the simple versions of the other approximate algorithms [CCV02].

- For our hybrid approximate algorithm (KCPH), we have detected a set of values for the approximate parameters (N_I, $Gamma_I$ and N_L) useful for the users in order to find a good balance between cost and accuracy of the result: $0.2 \leq N_I \leq 0.6$, $Gamma_I = 1.0$ and $0.6 \leq N_L \leq 1.0$.

Future research may include:

- The use of our hybrid approximate scheme (N-consider and α-allowance) for *high-dimensional similarity joins* [KoS98], adapting our KCPH approximate algorithm.
- The use of our hybrid approximate scheme over hash-like structures (e.g. Locally-Sensitive Hash [GIM99]), in a sense similar to the ε-approximate technique.
- The extension of our recursive approximate branch-and-bound algorithms for K-CPQ to execute approximate K-Self-CPQ and approximate self-high-dimensional similarity join, where both data sets refer to the same entity.
- The use of synthetic data following other distributions (e.g. Gaussian), or the use of real high-dimensional data sets from real color images, CAD data, etc, for experimentation.

REFERENCES

[AMN98] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman and A.Y. Wu: "An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions", *Journal of the ACM*, Vol.45, No.6, pp.891-923, 1998.

[BBK97] S. Berchtold, C. Böhm, D.A. Keim and H.P. Kriegel: "A Cost Model for Nearest Neighbor Search in High-Dimensional Data Space", *Proceedings 1997 ACM PODS Symposium*, pp.78-86, Tucson, AZ, 1997.

[BKS90] N. Beckmann, H.P. Kriegel, R. Schneider and B. Seeger: "The R*-tree: and Efficient and Robust Access Method for Points and Rectangles", *Proceedings 1990 ACM SIGMOD Conference*, pp.322-331, Atlantic City, NJ, 1990.

[BKK96] S. Berchtold, D. Kiem and H.P. Kriegel: "The X-tree: An Index Structure for High-Dimensional Data", *Proceedings 22nd VLDB Conference*, pp.28-39, Bombay, India, 1996.

[BoK01] C. Bohm and H.P. Kriegel: "A Cost Model and Index Architecture for the Similarity Join". *Proceedings ICDE Conference*, pp.411-420, Heidelberg, Germany, 2001.

[CCV02] A. Corral, J. Cañadas and M. Vassilakopoulos: "Approximate Algorithms for Distance-Based Queries in High-Dimensional Data Spaces using R-trees", to appear in *Proceedings 6th Conf. on Advances in Databases and Information Systems (ADBIS*), Bratislava, Slovakia, September 8-11, 2002.

[CiP00] P. Ciaccia and M. Patella: "PAC Nearest Neighbor Queries: Approximate and Controlled Search in High-Dimensional and Metric Spaces". *Proceedings ICDE Conference*, pp.244-255, San Diego, CA, 2000.

[Cla94] K.L. Clarkson: "An Algorithm for Approximate Closest-Point Queries". *Proceedings 10th ACM Symposium on Computational Geometry*. New York, pp.160-164. 1994.

[CMT00] A. Corral, Y. Manolopoulos, Y. Theodoridis and M. Vassilakopoulos: "Closest Pair Queries in Spatial Databases", *Proceedings 2000 ACM SIGMOD Conference*, pp.189-200, Dallas, TX, 2000.

[CPZ97] P. Ciaccia, M. Patella and P. Zezula: "M-tree: An Efficient Access Method for Similarity Search in Metric Spaces", *Proceedings 23rd VLDB Conference*, pp.426-435, Athens, Greece, 1997.

[CVM01] A. Corral, M. Vassilakopoulos and Y. Manolopoulos: "The Impact of Buffering for the Closest Pairs Queries using R-trees", *Proceedings 5th Conf. on Advances in Databases and Information Systems (ADBIS)*, pp.41-54, Vilnius, Lithuania, 2001.

[DiS01] J.P. Dittrich and B. Seeger: "GESS: a scalable similarity-join algorithm for mining large data sets in high dimensional spaces". *Proceedings 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p.p.47-56, San Francisco, CA, 2001.

[FBF94] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz: "Efficient and Effective Querying by Image Content", *Journal of Intelligent Information System*, Vol.3, No.3-4, pp.231-262, 1994.

[GaG98] V. Gaede and O. Günther: "Multidimensional Access Methods", *ACM Computing Surveys*, Vol.30, No.2, pp.170-231, 1998.

[GIM99] A. Gionis, P. Indyk and R. Motwani: "Similarity Search in High Dimensions via Hashing", *Proceedings 25th VLDB Conference*, pp.518-529, Edinburgh, Scotland, UK, 1999.

[Gut84] A. Guttman: "R-trees: A Dynamic Index Structure for Spatial Searching", *Proceedings 1984 ACM SIGMOD Conference*, pp.47-57, Boston, MA, 1984.

[HjS98] G.R. Hjaltason and H. Samet: "Incremental Distance Join Algorithms for Spatial Databases", *Proceedings 1998 ACM SIGMOD Conference*, pp.237-248, Seattle, WA, 1998.

[HjS99] G.R. Hjaltason and H. Samet: "Distance Browsing in Spatial Databases". *ACM Transactions on Database Systems*, Vol.24 No.2, pp.265-318, 1999.

[KoS98] N. Koudas and K.C. Sevcik: "High Dimensional Similarity Joins: Algorithms and Performance Evaluation", *Proceedings ICDE Conference*, pp.466-475, Orlando, FL, 1998.

[KSF96] F. Korn, N. Sidiropoulos, C. Faloutsos, C. Siegel and Z. Protopapas: "Fast Nearest Neighbor Search in Medical Images Databases", *Proceedings 22nd VLDB Conference*, pp.215-226, Bombay, India, 1996.

[MTT99] Y. Manolopoulos, Y. Theodoridis and V. Tsotras: *Advanced Database Indexing*, Kluwer Academic Publishers, 1999.

[PrS85] F.P. Preparata and M.I. Shamos: *Computational Geometry: An Introduction*, Springer Verlag, 1985.

[RKV95] N. Roussopoulos, S. Kelley and F. Vincent: "Nearest Neighbor Queries", *Proceedings 1995 ACM SIGMOD Conference*, pp.71-79, San Jose, CA, 1995.

[SML00] H. Shin, B. Moon and S. Lee: "Adaptive Multi-Stage Distance Join Processing", *Proceedings 2000 ACM SIGMOD Conference*, pp.343-354, Dallas, TX, 2000.

[SSA97] K. Shim, R. Srikant and R. Agrawal: "High-Dimensional Similarity Joins", *Proceedings ICDE Conference*, pp.301-311, Birmingham UK, 1997.

[SYU00] Y. Sakurai, M. Yoshikawa, S. Uemura and H. Kojima: "The A-tree: An Index Structure for High-Dimensional Spaces Using Relative Approximation". *Proceedings 26th VLDB Conference*, pp.516-526, Cairo, Egypt, 2000.

[WSB98] R. Weber, H.J. Schek and S. Blott: “A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces”. *Proceedings 24th VLDB Conference*, pp.194-205, New York City, NY, 1998.

[ZSA98] P. Zezula, P. Savino, G. Amato and F. Rabitti: “Approximate Similarity Retrieval with M-Trees”. *VLDB Journal*, Vol.7, No.4, pp.275-293. 1998.