On Dominating Your Neighborhood Profitably

Cuiping Li[1] * Anthony K. H. Tung[2] Wen Jin[3] Martin Ester[3]
[1] Renmin University of China cuiping_li@263.net
[2] National U. of Singapore atung@comp.nus.edu.sg
[3]Simon Fraser University {wjin,ester}@cs.sfu.ca

ABSTRACT

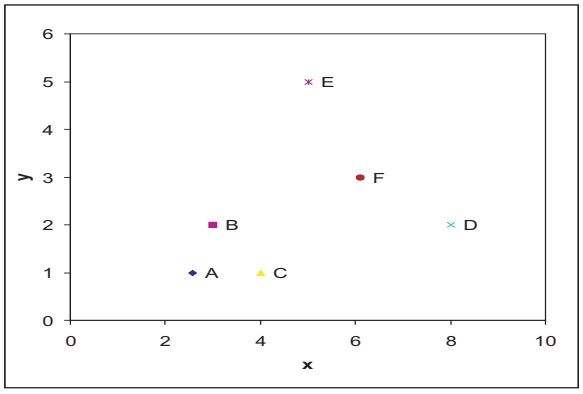
Recent research on skyline queries has attracted much interest in the database and data mining community. Given a database, an object belongs to the skyline if it cannot be dominated with respect to the given attributes by any other database object. Current methods have only considered so-called min/max attributes like price and quality which a user wants to minimize or maximize. However, objects can also have spatial attributes like x, y coordinates which can be used to represent relevant constraints on the query results. In this paper, we introduce novel skyline query types taking into account not only min/max attributes but also spatial attributes and the relationships between these different attribute types. Such queries support a micro-economic approach to decision making, considering not only the quality but also the cost of solutions. We investigate two alternative approaches for efficient query processing, a symmetrical one based on off-the-shelf index structures, and an asymmetrical one based on index structures with special purpose extensions. Our experimental evaluation using a real dataset and various synthetic datasets demonstrates that the new query types are indeed meaningful and the proposed algorithms are efficient and scalable.

1. INTRODUCTION

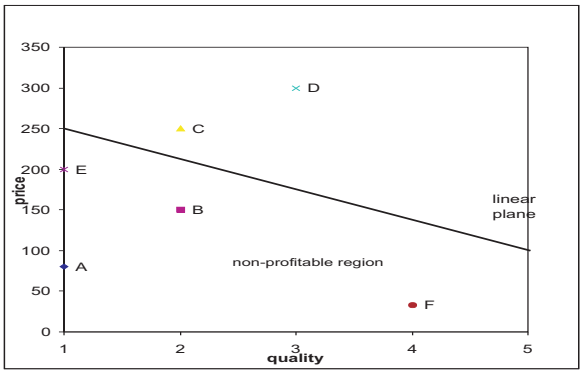
Recent research on skyline computation has attracted much interest in the database community [1, 2, 4, 14, 20]. Given a set of attributes, and a database D, an object q is said to be in the skyline of D if there is no object p in D such that p is as good or better in all dimensions and better in at least one dimension. If there exists such a p, then we say that q is dominated by p or p dominates q.

EXAMPLE 1. *Consider the six hotels listed in Table 1. If we compare the quality of these hotels based on the price and quality attributes as shown in Figure 1(b), then we can see that hotels A and F are the only two skyline points among the six hotels. This is because hotels B, C, D and E are all dominated by hotel A in terms of quality and price.*

From a customer's perspective, the skyline of a set of hotels is useful when selecting a hotel to stay since it is obvious that hotel A will always be a better choice when compared to B, C, D and E with F being an alternative if he/she wants to tradeoff quality for price. Answering such preference queries [14] is one reason why skyline computation has emerged as a hot research topic.



(a) Location of the 6 hotels in spatial dimensions x and y



(b) Location of the 6 hotels in min/max dimensions quality and price

Figure 1: Spatial and Min/Max Attributes of Hotels

Hotel	x	y	quality	price
A	2.58	1	1	80.2
B	3	2	2	150
C	4	1	2	250
D	8	2	3	300
E	5	5	1	200
F	6.11	3	4	33

Table 1: Six hotels with their spatial locations, quality and price

Besides quality and price, however, spatial location is also an im-

*Supported by NSFC(60673138, 60603046, 60473069, 60496325)

VLDB '07, September 23-28, 2007, Vienna, Austria.

portant aspect that affects customer decision. Going back to our running example, if the customer is attending a conference at location (8,2) on Figure 1(a), then he/she might consider staying in hotel D although it is not in the skyline in term of quality and price. Note that unlike the quality and price attributes in which "good" and "better" are defined right from the start, the preferred values for attributes x and y can be determined only with respect to some given reference point. To distinguish these two types of attributes, we will call attributes such as $quality$ and $price$ **min/max** attributes and attributes such as x and y **spatial attributes**.

While previous research on skyline queries has investigated the perspective of a customer who wants to find a good trade-off between hotel price and quality, minimizing the price for a given quality, our research is motivated by the hotel management point of view. The objective of a hotel manager is to maximize the price (and consequently, the profit) for a given quality within certain constraints given by the price and quality of competing hotels and their proximity. Thus, a hotel manager may want to answer the following types of queries:

1. For my hotel q at location (x,y), what is the nearest hotel p that dominates q in the min/max dimensions? We call p the *nearest dominator of q* and the distance between them the *nearest dominator distance* denoted as $ndd(q)$.

2. Let us assume that, to run a hotel profitably, the management must charge prices of at least \$250, \$210, \$180, \$140 and \$100 for 1^{st}, 2^{nd},3^{rd}, 4^{th} and 5^{th} class quality, respectively. Which hotel q is profitable according to that constraint while having the largest $ndd(q)$, i.e. has the largest spatial distance from all hotels who dominate it in the min/max dimensions?

3. Given the above profitability constraint and a distance threshold δ, find a hotel q such that $ndd(q) \geq \delta$ and the difference between the price charged and the minimal profitable price is the smallest.

EXAMPLE 2. *In Figure 1(a), the nearest dominator of hotel B is hotel A while the nearest dominator of hotel D is hotel C. We can see that $ndd(D) > ndd(B)$ although B dominates D in the min/max attributes. The above profitability constraint is represented as a plane in Figure 1(b). Hotels above this plane are profitable, others are not, i.e. only hotels C and D are profitable. Since $ndd(D) > ndd(C)$, hotel D is the answer for the second example query. Assuming $\delta = 4.5$, hotel E will be returned for the third query, since its nearest dominator is A with a distance of 4.6 and E's distance to the profitability plane is the smallest among all hotels whose nearest dominator distance is not smaller than δ.*

To illustrate that the above query types are important in a broad class of applications, let us discuss a second motivating scenario. A survey of a group of consumers is done with their age, salary plus the weight and price of the notebook they owned being recorded. In this case, the age and salary are the spatial attributes while the weight and price of the notebook are the min/max attributes. A consumer q with a large $ndd(q)$ own a notebook which is comparable to many other consumers with similar age and salary. Obviously, a notebook manufacturer will be interested to find notebook in the market which can cater to the biggest age-salary group so that he/she can build a similar or a slightly better one to target the same group as well. However, this have to be done within the profit constraint which correspond to the second query in the hotel example. Alternatively, he/she might choose to incur some minimized loss so that the notebook manufactured can cater to the need of a sufficiently large salary-age group to extend his/her customer base. This correspond to the third query in the earlier example.

Inspired by the above motivating applications, we call this new family of query types considering the relationship between min/max and spatial attributes, *neighborhood dominant queries* (NHDQs). In order to process these NHDQs efficiently, this paper explores two alternative approaches. The symmetrical approach treats both min/max and spatial attributes as equal and indexes them together in one R-tree. This approach is essentially the same as the approach taken by [11] which shows that their method is flexible enough to deal with many variants of the skyline problems. However, NHDQ queries have a non-symmetrical nature. While the spatial dimensions are used for capturing neighborhood information, the min/max dimensions are used to compute the dominant relationship. Therefore, we also propose an asymmetrical approach in which an R-tree index is built only on the min/max dimensions, and the spatial information is captured as bitmaps which are associated with the R-tree nodes.

The contributions of this paper are as follows:

- We introduce three novel types of skyline queries, which we call *neighborhood dominant queries*, that exploit not only min/max attributes but also spatial attributes. These queries support a micro-economic approach to decision making, considering not only the quality but also the cost of solutions.

- To efficiently process the proposed neighborhood dominant queries, we present symmetrical as well as asymmetrical methods, based on standard or extended index structures.

- Our experimental evaluation on a real dataset and on a variety of synthetic datasets demonstrates that the new query types produce meaningful results and the proposed algorithms are efficient and scalable.

The rest of the paper is organized as follows. Section 2 surveys the related work and Section 3 gives the problem statements. In Section 4 and Section 5, symmetry, asymmetry approaches on neighborhood dominant queries are presented respectively. We give our experimental results in section 6. We conclude the paper with a in section 7.

2. RELATED WORK

2.1 Skyline

The skyline computation originates from the maximal vector problem in computational geometry, proposed by Kung et al. [3]. The algorithms developed [9, 15] usually suits for a small dataset with computation done in main memory. One variant of maximal vector problem, which is related to but different from the notion of thick skyline, is the *maximal layers* problem[10, 16] which aims at identifying different layers of maximal objects.

Borzsonyi et al. first introduce the skyline operator over large databases [14] and also propose a divide-and-conquer method. The method based on [3, 12] partitions the database into memory-fit partitions. The partial skyline objects in each partition is computed using a main-memory-based algorithm [15, 9], and the final skyline is obtained by merging the partial results. In [4], the authors proposed two progressive skyline computing methods. The first employs a bitmap to map each object and then identifies skyline through bitmap operations. Though the bit-wise operation is fast, the huge length of the bitmap is a major performance concern. The second method introduces a specialized B-tree which is built for each combination list of dimensions that a user might be interested in. Data in each list is divided into batches. The algorithm processes each batch with the ascending index value to find skylines.

Kossmann et al. present an online algorithm, NN, based on the nearest neighbor search. It gives a big picture of the skyline very

quickly in all situations. However, it has raw performance when large amount of skyline needs to be computed. The current most efficient method is BBS (branch and bound skyline), proposed by Papadias et al., which is a progressive algorithm to find skyline with optimal times of node accesses [2, 11]. Balke et al. [19] in their paper show how to efficiently perform distributed skyline queries and thus essentially extend the expressiveness of querying current Web information systems. They also propose a sampling scheme that allows to get an early impression of the skyline for subsequent query refinement.

2.2 Microeconomic Data Mining

The microeconomic approach to data mining has been introduced by Kleinberg et al [7] formalizing the optimization problem of enterprises based on data allowing the enterprise to predict the utility of a customer w.r.t. a chosen decision. [7] focuses on a special class of such optimization problems, so-called segmentation problems, and shows that all discussed segmentation problems are NP-complete. [6] also shows how sensitivity analysis of the microeconomic optimization problem can distinguish interesting from uninteresting changes of the decision of the enterprise. In [7], the same authors investigate segmentation problems in more details. As an approximate algorithm for the catalog segmentation problem, they outline a sampling-based algorithm (enumerating and measuring all possible partitions of the customers in the sample) and prove probabilistic bounds for its result quality and runtime.

The existing methods on microeconomic data mining mainly focus on the efficiency issues of extracting interesting patterns from raw data. As far as we know, the only attempt to link microeconomic data mining with dominant relationship is in [8]. However, only min/max attributes are being considered. As shown in this paper, spatial dimensions add a new level of complexity and power to the analysis of dominant relationship.

3. PROBLEM STATEMENTS

Let S be a set of spatial dimensions and D be a set of min/max dimensions of the dataset H.

DEFINITION 1. **Dominating Relationship**
Let D be a set of min/max dimensions [1] of the dataset H, $p = (p_1, \ldots, p_d) \in H$ dominates *another object $q = (q_1, \ldots, q_d) \in H$, denoted as $p \succ q$, if $p_i \leq q_i (1 \leq i \leq d)$ and at least for one attribute say the jth attribute$(1 \leq j \leq d)$, $p_j < q_j$. On the other hand, q is a dominated object, p is a dominator of q.* □

DEFINITION 2. $ND(q)$, $ndd(q)$
Let p, q be two points in H such that 1)p dominate q, 2)among all points that dominate q, p is nearest to q in the space of S. We call p the nearest dominator of q, $ND(q)$. We will use $ndd(q)$ to refer to the distance between q and $ND(q)$. □

We can now formally define the queries that we will look at in this paper.

Problem 1: Nearest Dominators Query (NDQ)
Given any arbitrary object q in H, find its nearest dominator $ND(q)$. □

DEFINITION 3. **Hyperplane Dominating Relationship**
Let S be a set of spatial dimensions and D be a set of min/max dimensions of the dataset H. Given an object $h \in H$ and a hyperplane P in the space of D, we say P dominates h, $P \succ h$ if there exists a point p in the plane P, such that $p \succ h$ in min/max attributes. □

In the above definition, hyperplane P is called *profitability constraint*, which works as the input of a microeconomic query for evaluating the degree of profits.

Problem 2: Least Dominated, Profitable Points Query (LDPQ)
Let S be a set of spatial dimensions and D be a set of min/max dimensions of the dataset H. Given a hyperplane, P [2] in the space of D, find the points $t \in H$,

1. $P \succ t$.
2. There does not exist any other points $p \in H - \{t\}$ such that p satisfies (i), and $|ndd(p)|^3 > |ndd(t)|$

□

Problem 2 aims to find those objects which are profitable based on a *profitability constraint*, and meanwhile locate in the area with most competitive advantage i.e. cover the biggest area. While most companies will like to ensure profitability, there are also companies which like to take some minimized loss in order to build a larger customer base. This corresponds to the following query.

Problem 3: Minimal Loss and Least Dominated Points Query (ML2DQ)
Let S be a set of spatial dimensions and D be a set of min/max dimensions of the dataset H. Given a threshold δ and a profitability constraint in the form of a hyperplane, P, find the point t, $t \in H$ such that

1. $ndd(t) \geq \delta$
2. There does NOT exist any other points $p \in H - \{t\}$ such that p satisfy (i) and distance of p to the plane P in the min/max dimensional space is less than the distance of t to P.

□

Note that LDPQ and ML2DQ are in fact constrained optimization problem and are the dual problem of each other. LDPQ hope to maximize $ndd(q)$ while satisfying the constraint that the solution must come from the profitable region. ML2DQ on the other hand aims to minimize loss (i.e. the distance going into the non-profit region) while satisfying the constraint on $ndd(q)$. In addition, it should also be easy to define a top-k version of LDPQ and ML2DQ which can easily handle by our algorithms with some trivial changes.

As a side note, it is also possible to define LDPQ and ML2DQ using an analogous concept to $ndd(q)$ call *further dominating distance of q*, $fdd(q)$. The distance $fdd(q)$ represents the largest distance such that any point within a distance of $fdd(q)$ is dominated by q. This concept is a more aggressive measure compared to $ndd(q)$ which only guarantee that all points within $ndd(q)$ cannot

[1] The original definition of dominating relationship [14] is based on the minimum or maximum condition, that is, for corresponding dimension, the smaller/larger the value, the better the object in this dimension. Here without loss of generality, we adapt to the minimum condition.

[2] Although we use a linear hyperplane as the constraint here, other non-linear constraints can easily be adopted as long as we have a way to estimate the nearest/furthest distance between the constraint and a point or a minimum bounding box. Non-linear constraint can also be approximated by piecewise linear splines.

[3] While we use only the nearest dominator in our definition here, the proposed techniques in this paper is still valid even if we allow the user the flexibility of using the distance to the nearest n^{th} dominators. We have however avoid doing so in order not to add unnecessary complexity to the description.

dominate q. The two concepts of $ndd(q)$ and $fdd(q)$ are however very similar and all algorithms that applied for $ndd(q)$ will be applicable to $fdd(q)$. As such, our studies will only focus on $ndd(q)$ for this paper.

4. SYMMETRICAL METHODS

In this section, we will first look at a symmetrical approach to answering the three types of queries that we have defined. In this approach, we treat both types of dimensions to be the equal and build an R-tree that index the points based on all the dimensions. As mentioned in [11], this approach is applicable for many variations of the skyline problems. We will briefly touch on R-tree here to set the context for discussion. Interested readers can refer to [5] for more details on this popular indexing structure.

An R-tree is a height balanced tree in which each node is a minimum bounding box(MBR) that most tightly bounds the MBRs of its children nodes. This property applies recursively for all nodes in the tree until the leaf nodes where MBRs most tightly bound a set of spatial objects. In a N dimensional space, an MBR, R, can thus be represented by two points $R_l = \{R_{l1}, .., R_{lN}\}$ and $R_u = \{R_{u1}, .., R_{uN}\}$ where R_{li} and R_{ui} is the lower bound and upper bound value for the MBR along dimension i respectively. We use **symmetrical R-tree** to refer to an R-tree which is built on both spatial and min/max attributes. The MBRs in a R-tree satisfy the following property as proven in [13] which will be useful for our algorithm derivation later on.

PROPERTY 4.1. **MBR Face Property**
Every face of an MBR in an R-tree contain at least one point. □

Since we need to distinguish the spatial attributes and min/max attributes of the symmetrical R-tree in the context of our study, we will assume for ease of discussion that the first $|D|$ attributes are the min/max attributes and the remaining $|S|$ attributes are the spatial attributes. To further ensure clarity, we will use R^S_{li}, R^S_{ui} to represent the lower bound and upper bound of dimension i if it is a spatial attribute and R^D_{li}, R^D_{ui} to do so if it is a min/max attribute. As and when needed, we will use R^S_l, R^S_u to represent the whole set of lower bound and upper bound values for the spatial attributes of R and R^D_l, R^D_u for the min/max attributes.

4.1 NDQ with Symmetrical R-tree

Given a point p in the database H, our algorithm for finding its nearest dominator is based on a **best first traversal** of the nodes in the symmetrical R-tree. In this approach, a heap is maintained for storing every MBR, R that could potentially or definitely contain a point that dominates p in the min/max dimensions.

LEMMA 4.1. **Case 1**
Given a point $p = \{p_1, ..., p_N\}$ and an MBR, R, **some** *points in R could dominate p if $R^D_{li} \leq p_i \leq R^D_{ui}$ for all min/max attribute i.*

PROOF. We take the extreme case where there is a point at $\{R^D_{l1}, ..., R^D_{lN}\}$ and since $R^D_{li} < p_i$ for all min/max attribute, the point at that position will dominate p. □

LEMMA 4.2. **Case 2**
Given a point $p = \{p_1, ..., p_N\}$ and an MBR, R, **some** *points in R would* **definitely** *dominate p if $R^D_{li} \leq p_i$ for all min/max attribute i and there exist exactly $|D| - 1$ min/max dimensions j such that $R^D_{uj} < p_j$.*

PROOF. Based on Property 4.1 stated earlier, there must be a point on the face containing the diagonal joining $\{R^D_{l1}$,...,R^D_{ij} ,...,$R^D_{lN}\}$ to $\{R^D_{l1}$,...,R^D_{uj},...,$R^D_{lN}\}$. Since the point with the worst dominant power is at $\{R^D_{l1}$,...,R^D_{uj},...,$R^D_{lN}\}$ which still dominates p based on our condition, there must be at least one point in R that dominate p. □

LEMMA 4.3. **Case 3**
Given a point $p = \{p_1, ..., p_N\}$ and an MBR, R, all points in R will dominate p if $R^D_{ui} < p_i$ for all min/max dimensions i.

PROOF. The point with worst dominant power in this case is at $\{R^D_{u1}$,...,R^D_{uj},...,$R^D_{uN}\}$ which still dominate p. Furthermore, all other points in the MBR will dominate $\{R^D_{u1}$,...,R^D_{uj},...,$R^D_{uN}\}$. By transitivity, all points in R will definitely dominate p. □

Figure 2 lists the three cases corresponding to Lemma 4.1, 4.2 and 4.3 respectively. Note that Case 1, Case 2 and Case 3 are increasing restricted version of the previous case and as such in our algorithm, they will be handled in reverse order.

DEFINITION 4. *MinDist(R,p)*
The minimum distance between an MBR R and a point p in the spatial dimensions is defined as:

$$\textit{MinDist(R,p)} = \sum_{i=|D|+1}^{N} |p_i - r_i|^2$$

where
$r_i = R^S_{li}$ *if* $p_i < R^S_{li}$,
$r_i = R^S_{ui}$ *if* $p_i > R^S_{ui}$ *and*
$r_i = p_i$ *otherwise.* □

The algorithm first starts from the root MBR of the R-tree and places its children MBRs into the heap by removing those that do not correspond to any of the three cases. Within the heap, the MBRs are ordered based on two criteria:

1. All MBRs corresponding to Case 3 are ordered before all MBRs corresponding to Case 2 while all MBRs from Case 1 are ranked last.

2. With each group, MBRs with smaller MinDist to p computed using spatial attributes are ordered before MBRs with larger MinDist.

This is then repeated recursively beginning from the MBR at the top of the heap again by taking its children MBRs and inserting those that potentially or definitely dominate p into the heap. The algorithm maintains a variable $Best$ which is initially set to ∞ and are updated based on the following two rules:

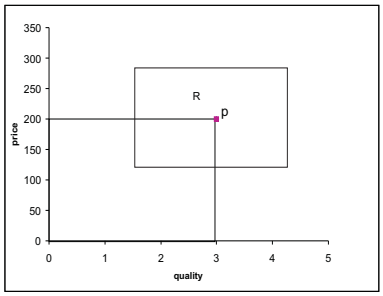
DEFINITION 5. *MaxDist(R,p)*
The maximum distance between an MBR R and a point p in the spatial dimensions is defined as:

$$\textit{MaxDist(R,p)} = \sum_{i=|D|+1}^{N} |p_i - r_i|^2$$

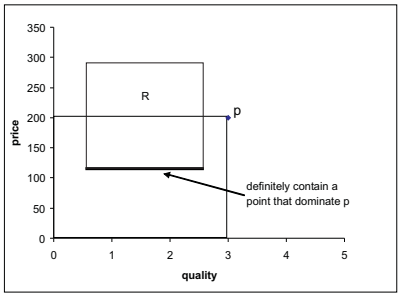
where
$r_i = R^S_{li}$ *if* $p_i \geqslant (R^S_{li} + R^S_{ui})/2$,
$r_i = R^S_{ui}$ *otherwise.* □

Rule 1: If current MBR, R, being processed corresponds to Case 2 with respect to p then we assign $Best$ to be the minimum of $Best$ and MaxDist(R,p).

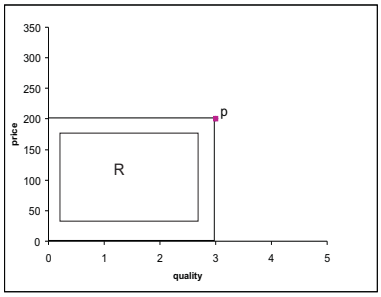
MaxDist(R,p) here corresponds to the furthest distance between p and any point in R. The rationale for this is that although there exists one point in R that dominates p, we do not know its distance to p. Thus we can only assume that the point is furthest away from p in the MBR and set $Best$ to such a value only if it is still smaller than $Best$.



(a) Case 1: **Some** points from R **could** dominate p



(b) Case 2: **Some** points from R **definitely** dominate p



(c) Case 3: **All** points from $R1$ **definitely** dominate p

Figure 2: Three different cases for the dominant relationship between R and p

DEFINITION 6. *MinMaxDist(R,p)*
The MinMax distance between an MBR R and a point p in the spatial dimensions is defined as:

MinMaxDist(R,p)=

$$\min_{|D|+1\leq k\leq N}(|p_k - rm_k|^2 + \sum_{\substack{|D|+1\leq i\leq N \\ i\neq k}} |p_i - rM_i|^2)$$

where

$rm_k = R^S_{lk}$, *if* $p_k \leq (R^S_{lk} + R^S_{uk})/2$,
$rm_k = R^S_{uk}$, *otherwise.*

and

$rM_i = R^S_{li}$, *if* $p_i \geq (R^S_{li} + R^S_{ui})/2$,
$rM_i = R^S_{ui}$ *otherwise.* □

Rule 2: If current MBR, R being processed corresponds to Case 3, with respect to p, then we assign $Best$ to be the minimum of $Best$ and MinMaxDist(R,p).

$MinMaxDist(R,p)$ here corresponds to a minimized upper bound on the distance between a point in R and p. If $MinMaxDist$ is smaller than $Best$, then it is guarantee that R contains at least one dominator of p which have a distance shorter than $Best$. Note that $MinMaxDist$ can only be applied for MBR corresponding to Lemma 4.3 because it makes use of the property that each face of the R contains at least a point and this is only true if all points in R dominate p.

Intuitively, the variable $Best$ thus stores the minimum upper bound on the distance between a point that dominates p and p itself. Thus any MBR, R that has $MinDist(p,R)$ greater than $Best$ will never contain the nearest dominator of p and can be removed without further processing. Also, if R is not within any of the three cases we shown in Figure 2, then it can be removed as well since no points in R will ever dominate p.

The algorithm terminates when there is no more MBRs in the heap. Note that if $Best$ remains at ∞ at the end of the algorithm, this means there is no nearest dominator for p i.e. p is a skyline point in the min/max dimensional space.

The pseudo-code of NDQ algorithm is listed as Algorithm 1.

4.2 LDPQ with Symmetrical R-tree

Unlike NDQ in which a specified point p is given, LDPQ does not focus its search on any particular portion of the space formed by the spatial attributes. Instead a profitability constraint is given in the form of a plane P in the space formed by the min/max attributes. Since we assume all attributes are min attributes, it is assumed that P is anti-correlated with respect to all the min attributes else we will be able to always choose the minimum value otherwise. Given P, the min/max dimensional space is divided into two regions, profitable and non-profitable.

ALGORITHM 1. *A Symmetrical NDQ Method.*
Input: *An R-tree R of H, query point p*
Output: *NDQ answer in H*
Method:

```
1:  Best := ∞; E={};
2:  Insert the root MBR of R-tree in the heap;
3:  WHILE exist MBRs in heap DO
4:      Extract an MBR R in heap;
5:      IF MinDist(R,p) > Best OR R not in all 3 cases
6:          Exit; % Handle next R on heap
7:      ELSE IF R is Case 3 MBR and MinMaxDist(R,p) < Best
8:              Best = MinMaxDist(R,p);
9:      ELSE IF R is Case 2 MBR and MaxDist(R,p) < Best
10:             Best = MaxDist(R,p);
11:     IF R is not a leaf MBR
12:         FOR each child c of R DO
13:             IF c in one of 3 cases, add c to heap;
14:     ELSE FOR each point x of R DO
15:         IF x is dominated by p, add x to E
16: Compute nearest dominator q among points in E;
17: Output q;
```

To simplify discussion, we will assume that skyline points with respect to the min/max attributes had been detected and a scan through such a list had been done to find those points that are in the valid region. If any of these point are in the skyline, then the solution is found and no search need to be done.

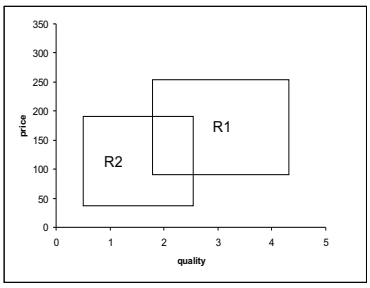
Handling LDPQ is more complex than handling NDQ as two types of MBRs must be monitored during the search.

1. **Potentially Dominated MBRs (PdMBR)**
   These are the MBRs that are potentially dominated by some points and are candidates for the output answers.

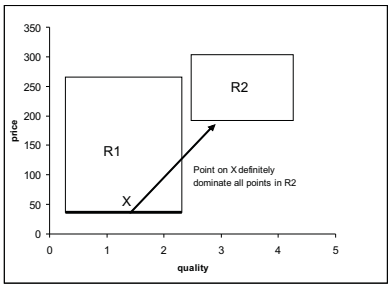
2. **Potentially Nearest Dominator (PnrMBR)**
   These are the MBRs that potentially contain the nearest dominators for those points in PdMBR.

Note that the set of PdMBRs and the set of PnrMBRs might not be mutually exclusive since the points in PnrMBR are not restricted to only points in the valid region. For an MBR $R2$, we denote all its potential dominating MBRs as $PnrMBR(R2)$ and for each MBR $R1$ we denote all the MBRs that it can potentially dominate as PdMBR(R1). We determine that the dominant relationship between MBRs from PdMBR and PnrMBR can be separate into 3 cases based on the following lemmas.

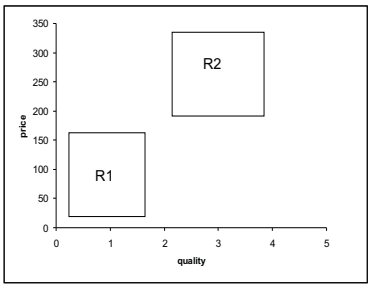
LEMMA 4.4. **Case 1**



(a) Case 1: **Some** points from $R1$ **could** dominate **some** points from R2



(b) Case 2: **Some** points from $R1$ **definitely** dominate **all** points from R2



(c) Case 3: **All** points from $R1$ **definitely** dominate **all** points from R2

Figure 3: Three different cases for the dominant relationship between two MBRs

Given two MBRs, $R1$ and R_2, **some** *points in $R1$* **could** *dominate* **some** *points in R_2 if $R1^D_{li} \leq R2^D_{ui}$ for all min/max attribute i.*

PROOF. Assuming this is not the case, then $R1^D_{li} > R2^D_{ui}$ for some i. All points in R_2 will always be better (i.e. lower) than $R1$ in the i^{th} dimension making it impossible for any point in $R1$ to dominate them. □

LEMMA 4.5. **Case 2**
Given two MBRs, $R1$ and R_2, **some** *points in $R1$* **definitely** *dominate* **all** *points in R_2 if $R1^D_{ui} \leq R2^D_{li}$ for $|D|-1$ min/max attribute i and $R1^D_{lj} \leq R2^D_{lj}$ for the remaining min/max attribute j.*

PROOF. From Property 4.1 [4], we know that each face of R1 will consist of at least 1 point. We will prove our lemma by identifying a face from $R1$ which definitely consists of a point that dominates all points in R_2. Let dimension 1,2,...,i,...,j-1,j+1,...$|D|$ be the set of dimensions in which $R1^D_{ui} \leq R2^D_{li}$ for $|D|-1$ min/max. Let dimensions j be the dimension in which $R1^D_{lj} \leq R2^D_{lj}$. Consider the face of $R1$ which contain the point ($R1^D_{l1}$, ..., $R1^D_{l|D|}$) and ($R1^D_{u1}$, ...,$R1^D_{lj}$,..., $R1^D_{u|D|}$). Let us call this face of $R1$, X. We claim that any point on X will definitely dominates all point in $R2$. This is because the weakest dominant point on X is ($R1^D_{u1}$, ..., $R1^D_{lj}$, ..., $R1^D_{u|D|}$) while the strongest dominant point in R_2 is ($R2^D_{u1}$, ..., $R2^D_{uj}$, ..., $R2^D_{u|D|}$) and based on the conditions we set, we can see that the weakest dominant point on X will still dominate the strongest dominant point in R_2. □

LEMMA 4.6. **Case 3**
Given two MBRs, $R1$ and R_2, **all** *points in $R1$ will* **definitely** *dominate* **all** *points in R_2 if $R1^D_{ui} < R2^D_{li}$ for all min/max attribute i.*

PROOF. Since the upper bounds of $R1$ are all smaller than the corresponding lower bounds of R_2, this means that all points in R_2 will be dominated by any point in $R1$ even if it is located at ($R2^D_{u1}$,...,$R2^D_{u|D|}$). □

Figure 3(a), 3(b) and 3(c) depicts three examples to illustrate the three cases that are stated in Lemma 4.4, 4.5 and 4.6 respectively. Again, it is easy to see that the conditions for Case 1, 2 and 3 are increasingly restrictive. Any pair of MBR, $R1$ and $R2$ that belong to any of the 3 cases will be monitored.

The above three cases provide different pruning types using different spatial measurements computed on the spatial attributes. For each $R2$ in PdMBR, we keep track of two variables as threshold values for pruning and comparing with other MBRs from PdMBR.

1. **Minimum Lower Bound,** $ndd_{mlb}(R2)$
   Let $ND_{mlb}(R2)$ denote $R1 \in PnrMBR(R2)$ which have the **minimum** value among the **lower bound distance** of the MBRs in $PnrMBR(R2)$ to R2. The variable $ndd_{mlb}(R2)$ represent the lower bound distance of $R1$ to R_2

2. **Minimum Upper Bound,** $ndd_{mub}(R2)$
   Let $ND_{mub}(R2)$ denote $R1 \in PnrMBR(R2)$ which have the **minimum** value among the **upper bound distance** of the MBRs in $PnrMBR(R2)$ to R2. The variable $ndd_{mub}(R2)$ represent the upper bound distance of $R1$ to R_2

We will now explain how these two variables are update according to the three cases and how pruning will be done accordingly.

DEFINITION 7. *MinMinDist(R1,R2)*
Let CORNER(R)=$\{ p : p = \{p_{|D|+1}, ..., p_N\}$, $p_i = R^S_{li}$ or $p_i = R^S_{ui}$ $\}$ denote the set of corners for an MBR, R. The MinMinDist between two MBRs, $R1$ and $R2$ based on their spatial attributes is defined as:

$$\textit{MinMinDist(R1,R2)}= \min_{p \in CORNER(R2)} MinDist(R1, p) \qquad \square$$

DEFINITION 8. *MaxMaxDist(R1,R2)*
Let CORNER(R)=$\{ p : p = \{p_{|D|+1}, ..., p_N\}$, $p_i = R^S_{li}$ or $p_i = R^S_{ui}$ $\}$ denote the set of corners for an MBR, R. The MaxMaxDist between two MBRs, $R1$ and $R2$ based on their spatial attributes is defined as:

$$\textit{MaxMaxDist(R1,R2)}= \max_{p \in CORNER(R2)} MaxDist(R1, p) \qquad \square$$

Here, MinMinDist(R1,R2) and MaxMaxDist(R1,R2) are easily understandable corresponding to the minimum and maximum distance between any pair of points from the two different MBRs respectively.

Rule 1: Update for Case 3
If R1 and R2 follow Case 3, then $ndd_{mlb}(R2)$ will be updated with MinMinDist(R1,R2) if the current value of $ndd_{mlb}(R2)$ is higher. This is because $ndd_{mlb}(R2)$ is suppose to contain the lower bound value of $ndd(p)$ for any p in R2. For the upper bound value $ndd_{mub}(R2)$, we will use a different function, MaxMinMaxDist(R1, R2) and update $ndd_{mub}(R2)$ with the value of MaxMinMaxDist(R1, R2) if MaxMinMaxDist(R1,R2) is smaller than the current value of $ndd_{mub}(R2)$

[4]Note that although we are processing MBRs consisting of both spatial and min/max dimensions while dominant relationship is only based on min/max dimensions, this property is still true as we are projecting the points into a lower dimensional MBR consisting of only the min/max dimensions when we do the dominant reasoning.

DEFINITION 9. *MaxMinMaxDist(R1,R2)*
Let CORNER(R)=$\{ p : p = \{p_{|D|+1}, ..., p_N\}$, $p_i = R^S_{li}$ *or* $p_i = R^S_{ui}$ $\}$ *denote the set of corners for an MBR,* R. *The MaxMinMaxDist between two MBRs,* $R1$ *and* $R2$ *based on their spatial attributes is defined as:*

MaxMinMaxDist(R1,R2)=
$$\max_{p \in CORNER(R2)} MinMaxDist(R1, p) \qquad \square$$

Intuitively, MaxMinMaxDist(R1, R2) computes an upper bound on the distance between any point p in $R2$ to a nearest point in $R1$ that **definitely** dominates p. Note that we can use such a measure because we know that **all** points in R1 definitely dominate R2 and thus each face of R1 definitely contains a point that dominates a point in each face of R2.

Rule 2: Update for Case 2
If R1 and R2 follow Case 2 (but not Case 3), the update of of $ndd_{mlb}(R2)$ is similar as in Rule 1 above. For $ndd_{mub}(R2)$ however, since we are only sure that there are some points in $R1$ that dominate all points in R_2, we can only use MaxMaxDist(R1,R2) as the minimum upper bound and update $ndd_{mub}(R2)$ with MaxMaxDist(R1,R2) if MaxMaxDist(R1,R2) is smaller than the current value of $ndd_{mub}(R2)$.

Rule 3: Update for Case 1
If R1 and R2 follow Case 1 (but not Case 2 and 3), we will not be sure whether the point in R1 dominates any point in R2. As such both $ndd_{mlb}(R2)$ and $ndd_{mub}(R2)$ cannot be updated. However being in Case 1 mean that R1 cannot be removed as a potential dominator of R2 and could be further expanded unless Rule 4 applies.

Rule 4: Local Pruning
Given R1 and R2, R1 can be removed from PnrMBR(R2) if MinMinDist(R1,R2) $> ndd_{mub}(R2)$. The reason is that $R1$ could never contain the nearest dominator for any points in R_2 since its nearest point to R_2 is already at a greater distance than $ndd_{mub}(R2)$.

Rule 4 is a local pruning in the sense that we are just pruning off the potential nearest dominators of R2. We next describe how a global pruning can be done for PdMBR. We maintain a variables $Best$ which is always updated as the highest $ndd_{mlb}(R2)$ for all $R2 \in$ PdMBR that had been processed. Our pruning go as follows:

Rule 5: Global Pruning
An MBR $R2$ will be removed from $PdMBR$ as a potential answer if $ndd_{mub}(R2) < Best$. This is because all points in $R2$ can never have a nearest dominator that is further away than $Best$. $\square$

The pseudo-code of LDPQ algorithm is listed as Algorithm 2. To perform best first search, the algorithm maintain a heap which store all R in PdMBR sorted in **decreasing** value of $ndd_{mlb}(R)$. Initially, MBRs at the first level of the R-tree are retrieved. For each MBR, R, the function Initialize() is called to compute the following six variables: PdMBR(R), PnrMBR(R), ND$_{mlb}(R)$, $ndd_{mlb}(R)$, ND$_{mub}(R)$ and $ndd_{mub}(R)$. This is done by comparing R against the rest of the MBRs. The algorithm maintains a temp list to keep all MBRs with their six variables. If R intersect the profitable region, it is then inserted into $heap$.

Once all the first level MBRs are processed. The algorithm then access the top MBR, R, from the heap and update $Best$ if this give a better result i.e. the nearest dominator is guaranteed to be further away than $Best$ for some point in R. Next, R and ND$_{mlb}(R)$ [5] are expanded by retrieving their children nodes. This is done in two phases.

ALGORITHM 2. *A Symmetrical LDPQ Method.*
Input: *An R-tree of* H, *a hyperplane* P
Output:*LDPQ answer in* H
Method:

```
1:  heap = ∅; Best = 0;
    %heap ordered by decreasing ndd_mlb(R)
2:  Retrieve the first level of MBRs in R-tree;
3:  FOR each MBR R in the first level
4:     Initialize(R), put R into the templist;
5:     IF R intersect profitable region
6:        Insert R into heap;
7:  WHILE heap not empty DO
8:     Retrieve R from top of heap;
9:     Best = max(Best,ndd_mlb(R));
10:    FOR each MBR R' in the templist, heap DO
11:       Remove R, ND_mlb(R) from all PdMBR(R'), PnrMBR(R');
12:       IF R equals to ND_mlb(R')
13:          Select a new ND_mlb for R' from the children of R;
14:       IF ND_mlb(R) equals to ND_mlb(R')
15:          Select a new ND_mlb for R' from the children of ND_mlb(R);
16:    FOR each child r of R, NDmlb(R) DO
17:       Initialize(r), put r into the templist;
18:    PdMBR= Children of R ∪ PdMBR(ND_mlb(R));
19:    PnrMBR = Children of ND_mlb(R) ∪ PnrMBR(R);
20:    FOR each R2 ∈ PdMBR DO
21:       FOR each R1 ∈ PnrMBR DO
22:          IF R1 and R2 DOES NOT follow all 3 cases OR
              MinMinDist(R1, R2)> ndd_mub(R2)
23:             Exit; % Handle next R1
24:          ELSE IF R1 and R2 follow Case 3
25:             ndd_mlb(R2)=min(ndd_mlb(R2),MinMinDist(R1,R2));
26:             ndd_mub(R2)=min(ndd_mub(R2),MaxMinMaxDist(R1,R2));
27:          ELSE IF R1 and R2 follow Case 2
28:             ndd_mlb(R2)=min(ndd_mlb(R2),MinMinDist(R1,R2));
29:             ndd_mub(R2)=min(ndd_mub(R2),MaxMaxDist(R1,R2));
30:          Add R1 into PnrMBR(R2);
31:          Add R2 into PdMBR(R1);
32:          Maintain ND_mlb(R2),ND_mub(R2);
33:       IF R2 intersect profitable region
34:          Add R2 into heap;
35:    Perform global pruning on heap;
36: Output Best and best point p;
```

First, we need update the variables of some MBRs in the temp list and the heap (line 10-15). Since R and ND$_{mlb}(R)$ are expanded to their children, they can not be potential nearest dominator or dominated MBRs any more. For each MBR R' in the temp list, we need remove R and ND$_{mlb}(R)$ from all $PdMBR(R')$ and $PnrMBR(R')$. If R happened to be the ND$_{mlb}(R')$, we need select a new ND$_{mlb}$ for R' from the children of R. If ND$_{mlb}(R)$ happened to be the ND$_{mlb}(R')$, we need select a new ND$_{mlb}$ for R' from the children of ND$_{mlb}(R)$. Once the ND$_{mlb}$ is changed, the value of the variable $ndd_{mlb}(R)$ need to be changed at the same time. Obviously, all MBRs in the heap need to be updated also. To simplify the implementation and keep one copy for the six variables of each MBR, we can associate a unique ID with each MBR, and just keep the IDs in the heap. In this case, only the temp list need to be updated.

Next, we must compute the potential dominating and dominated MBRs for the children nodes of R and ND$_{mlb}(R)$ respectively.Line 16-17 is used to initialize the children of R and ND$_{mlb}(R)$. After this, the dominating relationship inside the R or ND$_{mlb}(R)$ was captured. For the children of ND$_{mlb}(R)$, the MBRs that are potentially dominated by them are the children of R and all those MBRs

[5]This is the node that potentially dominate all points in R with at least distance $ndd_{mlb}(R)$) as define earlier

which are potentially dominated by $\text{ND}_{mlb}(R)$. These MBRs are added into PdMBR at Line 18. Line 19 update PnrMBR, the set of MBRs that potentially dominate the children of R using the same reasoning. Line 20-34 then take each pair of MBRs from PdMBR and PnrMBR and compute the values of the six variables by considering the three cases of dominating relationship that we discuss earlier. Line 35 then perform a global pruning removing all R in the heap with $ndd_{mub}(R2) < Best$.

The next item on the heap is then retrieved and the above procedure is repeated until there is no more items in the heap. The object that last update the variable $Best$ will then be output.

4.3 ML2DQ with Symmetrical R-tree

We next look at the handling of ML2DQ using the symmetrical R-tree. The query consists of a distance bound δ and a profitability constraint P. The aim of this type of query is to find a point q in the unprofitable region bounded by P and the min/max attribute axes such that the distance to P is minimized while satisfying the constraints all that $ndd(q) \leq \delta$. Note that the use of ML2DQ is necessary only if no points q in the profitable region satisfy $ndd(q) \leq \delta$. This can be trivially checked by issuing a LDPQ query right from the start. As such in this section, we will only handle the case in which the top answer come from the non-profitable region.

To answer ML2DQ, we adopt the same best first search approach as LDPQ. The pruning comes in three forms. First, only MBRs intersecting the non-profitable region are considered for reason mentioned earlier. Second, MBRs with nearest dominators that are less than a distance of δ are removed. Third, MBRs which are too far away from P to be the result are also removed.

Performing the first form of pruning involves trivial geometry computation which we will not describe here. For the second form of pruning, we again separate the MBRs into PdMBR and PnrMBR and compute various bounds on the spatial distance between the MBRs based on the three possible cases of dominant relationship we described for LDPQ.

Pruning by Constraint
An MBR $R2$, can be removed from PdMBR if $ndd_{mub}(R2) < \delta$. Obviously, all points in $R2$ will never satisfy the constraint that the nearest dominator must be a distance of δ away.

The third type of pruning is to remove MBRs that are too far away from P in the min/max dimensional space and thus can never be among the best result. To achieve this goal, we again need to define the MinDist and MinMaxDist for an MBR, R2 to the plane P, in time in the min/max dimensional space.

DEFINITION 10. *MinDist(R2,P)*
The minimum distance between R2 and the plane P in the min/max dimensional space is the distance of the point ($R2^D_{u1}$,...,$R2^D_{u|D|}$) to the plane P if P does not intersect P else MinDist(R2,P)=0. □

DEFINITION 11. *MaxDist(R2,P)*
The minimum distance between R2 and the plane P in the min/max dimensional space is the distance of the point ($R2^D_{l1}$,...,$R2^D_{l|D|}$) to the plane P. □

As mentioned earlier, since we assume that all attributes in D are min attributes, the plane P will be anti-correlated compared against all the axes of the min/max attributes. Given this fact, the minimum distance between the P and an MBR, R2 will be the distance from $\{R2^D_{u1},...,R2^D_{u|D|}\}$ to P unless P intersect R2 in which case the MinDist is obviously 0. For the same reason, we compute MaxDist(R2,P) to be the distance between $\{R2^D_{l1},...,R2^D_{l|D|}\}$ which is the maximum distance for a point in $R2$ to move into the unprofitable region away from P.

ALGORITHM 3. *A Symmetrical ML2DQ Method.*
Input: *An R-tree R of H, a hyperplane P*
Output: *ML2DQ answer in H*
Method:

```
1:  heap = ∅; Best = ∞;
    % heap ordered by increasing MinDist(R,P)
2:  Retrieve the first level of MBRs in R-tree;
3:  FOR each MBR R in the first level DO
4:     Initialize(R), put R into the templist;
5:     IF R intersect non-profitable region
6:        Insert R into heap;
7:  WHILE heap not empty DO
8:     Retrieve R from top of heap;
9:     Best = max(Best,MaxDist(R, P));
10:    FOR each MBR R' in the templist, heap DO
11:       Remove R, ND_mlb(R) from all PdMBR(R'), PnrMBR(R');
12:       IF R equals to ND_mlb(R')
13:          Select a new ND_mlb for R' from the children of R;
14:       IF ND_mlb(R) equals to ND_mlb(R')
15:          Select a new ND_mlb for R' from the children of ND_mlb(R);
16:    FOR each child r of R, NDmlb(R) DO
17:       Initialize(r), put r into the templist;
18:    PdMBR= Children of R ∪ PdMBR(ND_mlb(R));
19:    PnrMBR = Children of ND_mlb(R) ∪ PnrMBR(R);
20:    FOR each R2 ∈ PdMBR DO
21:       FOR each R1 ∈ PnrMBR DO
22:          IF R1 and R2 DOES NOT follow all 3 cases OR
              MinMinDist(R1, R2)> ndd_mub(R2)
23:             Exit; % Process next R1
24:          ELSE IF R1 and R2 follow Case 3
25:             ndd_mub(R2)=min(ndd_mub(R2),MaxMinMaxDist(R1,R2));
26:          ELSE IF R1 and R2 follow Case 2
27:             ndd_mub(R2)=min(ndd_mub(R2),MaxMaxDist(R1,R2));
28:          Add R1 into PnrMBR(R2);
29:          Add R2 into PdMBR(R1);
30:          Maintain ND_mub(R2);
31:       IF ndd_mub(R2) ≥ δ % Pruning by Constraint
32:          Add R2 into heap;
33:    Perform global pruning on heap;
34: Output Best and best point p;
```

Global Pruning
To perform pruning, we monitor MaxDist(R,P) for all MBRs in PdMBR and maintain the smallest MaxDist among them as $Best$. We then remove an MBR R away from PdMBR if MinDist(R,P)> $Best$. Note that $Best$ are initially set to ∞.

The pseudo-code of ML2DQ algorithm is listed as Algorithm 3. The overall structure of the algorithm is generally similar to the LDPQ algorithm except for the ordering of the heap and the use of the two pruning methods i.e. the constraint and global pruning.

5. ASYMMETRICAL METHODS

While a symmetrical approach is attractive because it can reuse a generic R-tree for supporting other forms of spatial and skyline queries, it might not always be the best solution since the queries that we are trying to handle are asymmetrical by nature; the spatial attributes and min/max attributes play different roles and have different characteristics in all the three types of queries.

In NDQ, it is important to determine dominant relationship for the query point p before finding the nearest neighbors among the dominant points. In LDPQ and ML2DQ, the profitability constraint P is only defined in terms of min/max attributes and the dominant relationship must be determined before neighborhood relationship is evaluated.

On the other hand, there is also a difference in characteristic between dominant relationship and spatial closeness of two points. A

point p that dominates another point q might not be spatially close even in the multi-dimensional space formed by the min/max attributes. For the spatial attributes however, neighborhood closeness is rather important and much more pruning can in fact be enforced in the early stage of the query answering if higher resolution spatial information is provided.

In view of this, we will next propose our solution by making use of an **asymmetrical R-tree**. Before construction of the tree, a clustering of the points are first employed in the spatial dimensions by grouping the points into k microclusters [18], MC_1,...,MC_k. This step can be finished by a typical pre-processing algorithm BIRCH [17]. For each microcluster, MC_i, we assign a cluster id, i, and keep track of its mean value, $MC_i.m$ and radius, $MC_i.r$ which is the distance between $MC_i.m$ and the furthest point in the cluster. Given any two microclusters, MC_i and MC_j, we pre-compute their maximum and minimum distance and store them in a lookup table.

DEFINITION 12. $MinDist(MC_i, MC_j)$
The minimum distance, $MinDist(MC_i, MC_j)$, between two microclusters MC_i and MC_j is $dist(MC_i.m, MC_j.m) - MC_i.r - MC_j.r$ if this is greater than 0, else $MinDist(MC_i, MC_j) = 0$ □

DEFINITION 13. $MaxDist(MC_i, MC_j)$
The maximum distance, $MaxDist(MC_i, MC_j)$, between two microclusters MC_i and MC_j is $dist(MC_i.m, MC_j.m) + MC_i.r + MC_j.r$. □

During the construction of the asymmetrical R-tree, the MBRs are formed by the min/max attributes while spatial information are captured in a bitmap of size k with bit i representing the absence and presence of MC_i in the MBR.

DEFINITION 14. *MCin(R)*
Given an MBR, R, in an asymmetrical R-tree, we use MCin(R)=$\{MC_{R1}$,...,$MC_{R|MCin(R)|}\}$ to denote the set of microclusters that are mark as present in R. □

Now let us look at how to answer NDQs with an asymmetrical R-tree.

5.1 NDQ with Asymmetrical R-tree

Given the query point p, we first define its minimum and maximum distance with respect to any microcluster MC_i.

DEFINITION 15. $MinDist(p, MC_i)$
The maximum distance between p and MC_i is define as:

$$MinDist(p, MC_i) = dist(p, MC_i.m) - MC_i.r$$

if $dist(p, MC_i.m) > MC_i.r$

0 otherwise. □

DEFINITION 16. $MaxDist(p, MC_i)$
The maximum distance between p and MC_i is define as:

$$MaxDist(p, MC_i) = dist(p, MC_i.m) + MC_i.r. \quad \square$$

Based on this, we can redefine the MinDist, MinMaxDist and MaxDist of an MBR R in the asymmetrical R-tree with respect to a point p.

DEFINITION 17. *MinDist(R,p)*

MinDist(R,p)=
$$min\{MinDist(p, MC_{Ri}), MC_{Ri} \in MCin(R)\} \quad \square$$

DEFINITION 18. *MaxDist(R,p)*

MaxDist(R,p)=
$$max\{MaxDist(p, MC_{Ri}), MC_{Ri} \in MCin(R)\} \quad \square$$

DEFINITION 19. *MinMaxDist(R,p)*

MinMaxDist(R,p)=
$$min\{MaxDist(p, MC_{Ri}), MC_{Ri} \in MCin(R)\} \quad \square$$

By plugging these three new definitions into Algorithm 1, we will have an algorithm that answer NDQ based on asymmetrical approach.

5.2 LDPQ and ML2DQ with Asymmetrical R-tree

As can be seen from the previous section, it is relatively easy to convert a NDQ algorithm on a symmetrical R-tree to a NDQ algorithm on a asymmetrical R-tree. This is because for an asymmetrical R-tree, the MBR coordinates are maintained for the min/max attributes and as such all dominant inference on the MBRs in the previous section can be applied. The only difference is that spatial inferences on the spatial attributes need different processing strategy. In an asymmetrical R-tree, the microclusters capture higher resolution spatial information in a bid to prune off search space as early as possible. To adapt Algorithm 2 and 3 in the previous section for computing LDPQ and ML2DQ on a asymmetrical R-tree, we need to redefine the function MinMinDist, MaxMaxDist and MaxMinMaxDist for two MBRs R1 and R2.

DEFINITION 20. *MinMinDist(R1,R2)*

MinMinDist(R1,R2)= $min\{$MinDist(MC_{R1i},MC_{R2j})
$MC_{R1i} \in MCin(R1), MC_{R2i} \in MCin(R2)\}$ □

Essentially, given the two set of microclusters that are present in MBR $R1$ and R2, we pick a pair of microclusters from $R1$ and R2 with the smallest pairwise minimum distance and take such a distance to be MinMinDist.

DEFINITION 21. *MaxMaxDist(R1,R2)*

MaxMaxDist(R1,R2)= $max\{$MaxDist(MC_{R1i},MC_{R2j})
$MC_{R1i} \in MCin(R1), MC_{R2j} \in MCin(R2)\}$ □

MaxMaxDist on the other hand, computes the exact opposite of MinMinDist. Given the two set of microclusters, it picks a pair which maximize the maximum distance between them.

Given a microcluster MC_{R2i} from $MCin(R2)$, let us denote the microcluster in $MCin(R1)$ which has the smallest MaxDist to MC_{R2i} as NNMAX(MC_{R2i},$MCin(R1)$) i.e. we are comparing one single microcluster from $MCin(R2)$ against the whole set of microcluster in $R1$ to find the nearest one from R1. We define MaxMinMaxDist of two MBRs from the asymmetrical R-tree as follow:

DEFINITION 22. *MaxMinMaxDist(R1,R2)*

MaxMinMaxDist(R1,R2)=
max$\{$MaxDist(MC_{R2i}, NNMAX(MC_{R2i},$MCin(R1)$))
where $MC_{R2i} \in MCin(R2)\}$ □

In other word, MaxMinMax estimates the distance between each microcluster $MC_{R2i} \in MCin(R2)$ to its nearest dominator in R1 based on pairwise MaxDist and then take the maximum one among all these pairs to estimate an upper bound on ndd(p) for all points p in the MBR R2.

Once MinMinDist, MaxMaxDist and MaxMinMaxDist are defined for MBRs in a asymmetrical R-tree, Algorithm 2 in the previous section can then be used for answer LDPQ query while Algorithm 3 can be used for computing answers to ML2DQ queries.

6. EXPERIMENTAL EVALUATION

To evaluate the efficiency and scalability of our query processing algorithms, we conducted extensive experiments. We implemented all algorithms using Microsoft Visual C++ V6.0, and conducted the experiments on a PC with Intel Pentium 4 2.4GHz CPU, 3G main memory and 80G hard disk, running Microsoft Windows XP Professional Edition. We conducted experiments on both synthetic and real life data sets.

6.1 Results on Synthetic Data Sets

We generate a set of synthetic data. For the min/max dimensions, we use the data generator that is used in [14] to generate data sets with three different distributions:Uniform(Uni), Correlated(Cor) and Anti-correlated(Ant). For spatial dimensions, we generate data sets with two different distributions: uniform(Uni) and clustered(Clu). The clustered distribution consist of 8 Gaussian distributed clusters which are randomly placed. By combining the three types of distribution for min/max dimensions and the two types of distribution for spatial dimensions, we obtain 6 different types of data sets: Uni-Uni, Uni-Clu, Cor-Uni, Cor-Clu, Ant-Uni, Ant-Clu. As an example, Uni-Clu refer to a dataset in which the data is uniformly distributed in the min/max dimensions and clustered in the spatial dimensions. We generate 5 different data sets for each of the 6 types and take the running time to be average over the 5 different data sets.

The default values of dimensionality is 8, data size is 100k and the default number of microclusters is 50.[6] For simplicity, we used the same number of min/max attributes and spatial attributes. As an example, for 12 dimensions, we chose 6 dimensions as min/max and the other 6 dimensions as spatial attributes.

6.1.1 NDQ

In this experiment, we evaluated the efficiency of our symmetrical algorithm (SYM-NDQ) and the asymmetrical algorithm (ASYM-NDQ) for answering a NDQ query. We compare our algorithms with a Naive method which returns the nearest dominator of a point by perform all pair comparison between the points.

Figure 4(a) shows the run time of the three algorithms for answering a NDQ query on six types of data sets. Obviously, from the results, we can see that SYM-NDQ outperforms Naive on all data sets. This is because pruning can reduce computation for those MBRs whose MinDist to the query point is greater than the current value of the variable $Best$. Among the algorithms, ASYM-NDQ performs best as expected due to the separate index for different types of attributes. In addition, R-Trees only achieve high performance for low dimensionality (usually ≤ 5), and the R-tree in the asymmetrical approach has smaller dimensionality.

Next, we look at the run time of the three algorithms as the number of dimension increases.Since the trends are the same for all six data sets, we only show the results on the data sets with Ant-Clu distribution as it is the most efficient. We increase the number of dimension from 4 to 12. Figure 4(b) shows the run time of all three algorithms. We observe that with increasing number of dimension, the runtime of Naive and symmetrical method increases more significantly than that of asymmetrical method. This is again due to the fact that the R-tree is more effective in the asymmetrical approach because of the smaller number of dimensions indexed and because spatial pruning is done at finer granularity in the earlier part of the R-tree search.

Figure 4(c) shows the run time of the three algorithms as the number of points increases from 20,000 to 100,000. From the results, we can see that both the symmetrical and the asymmetrical algorithms are scalable with respect to the size of data sets. However, the asymmetrical method is more efficient than the symmetrical method

6.1.2 LDPQ

In this experiment, we evaluated the performance of the three algorithms Naive, SYM-LDPQ and ASYM-LDPQ for answering a LDPQ query. The default data size is 10,000. In this case, the Naive algorithm perform a ASYN-NDQ search for all points in the profitable region and select the point with the highest nearest dominator distance among them. To test the efficiency and the scalability of the three algorithms, we choose a profitability hyperplane that roughly splitting the data set into two parts of similar size.

Figure 5(a) shows the run time of the three algorithms for answering a LDPQ query on six types of data sets. We can find that ASYM-LDPQ is always the fastest one among the three algorithms. Among the six distributions, we can see that the computation on the Ant-Clu data set gives the most improvement from symmetrical method to asymmetrical method while that of the Cor-Uni data set is the most modest. This is because: 1) the domination relationship computation on the anti-correlated min/max dimensions is the fastest since in this case the number of skyline objects is the largest, and less dominating/dominated relationship needs to be maintained between MBRs (points). 2) the distance computation on the clustered spatial dimensions is the fastest since in this case spatial proximity can be explored and more pruning can be done in the early stage of query processing.

Figure 5(b) shows the run time of each algorithm with increasing dimensionality. Here we only show the results on the data sets with Ant-Clu distribution for the same reason mentioned earlier. Clearly, we can find that with increasing number of dimension, the runtime of Naive and symmetrical method increases more significantly than that of asymmetrical method. The difference between the run time of SYM-LDPQ and ASYM-LDPQ increases with dimensionality. This is again due to the fact that the R-tree is more effective in the asymmetrical approach because of the smaller number of dimensions being indexed and because of the fine granularity of the microclusters which support effective spatial pruning even with higher dimensionality.

Figure 5(c) shows how the run time of each algorithm scales up as the number of points increase. We can find that although both the symmetrical and the asymmetrical algorithms scale linearly, the run time of the asymmetrical algorithm scales better than that of the symmetrical algorithm. As the number of points increases, the asymmetrical approach only slightly worsens since smaller number of dimensions means that the height of the R-tree increase slower than the symmetrical approach when more points are added.

6.1.3 ML2DQ

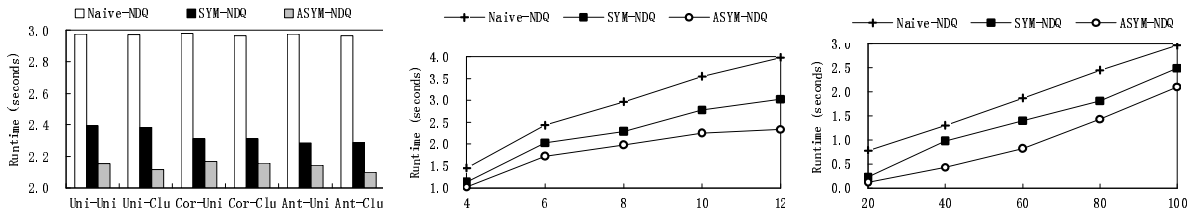
We performed similar experiments for ML2DQ queries. We set the input distance threshold as the average distance of points to their nearest dominator.

Figure 6(a) shows the run time of the three algorithms (Naive, SYM-ML2DQ, ASYM-ML2DQ) for answering a ML2DQ query on six types of data sets. Figure 6(b) and 6(c) shows the run time of the three algorithms as the dimensionality and the number of points increases respectively. As expected, the asymmetrical approach performs the best in both the figures for similar reasons as LDPQ.

6.2 Results on NBA Data Set

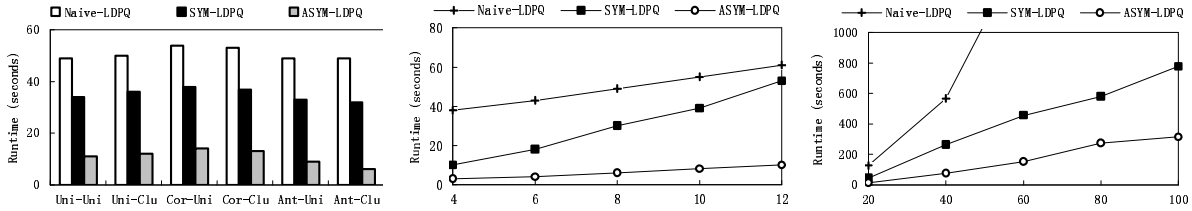
We downloaded from the NBA official website (www.nba.com)

[6]In our full paper, varying number of microcluster size from 50 to 100 does not bring about significant changes in performance for all datasets



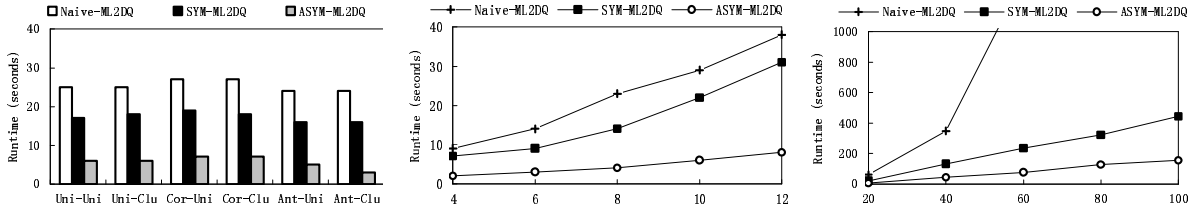
(a) varying data distribution (b) varying dimensionality (c) varying number of points(1k)

Figure 4: Query Performance for NDQ



(a) varying data distribution (b) varying dimensionality (c) varying number of points(1k)

Figure 5: Query Performance for LDPQ



(a) varying data distribution (b) varying dimensionality (c) varying number of points(1k)

Figure 6: Query Performance for ML2DQ

the Great NBA Players' technical statistics from 2004 to 2005, and downloaded NBA players' salary information from website: http://asp.usatoday.com/sports/basketball/nba/salaries/.

The NBA data set has more than 20 attributes, from which we chose the following four *min/max attributes*: the number of games played(GP), points per game(PPG), rebounds per game(RPG) and assists per game(APG). For these attributes, the larger the values are, the better. This mean that player A dominates player B if A's attribute values are not less than B's, and A has at least one attribute better than B. We selected weight, height and position as *spatial attributes* since these attributes are not performance measures to be minimized or maximized but can be used to identify players that are comparable in terms of their attributes and are expected to have similar performance. Weight and height are numerical attributes with canonical distance definitions, while the distance of positions is defined to be 1 if two players are in different positions and 0, otherwise. Finally, we used the salary attribute to define a profitability constraint.

There are some very interesting results for the neighborhood dominant queries. For example, let us consider Yao Ming, a 7.6 feet, 310 pounds all-star center in Houston Rockets, who had 80(GP), 18.3(PPG), 8.4(RPG) and 0.8(APG) for the NBA 2004-2005 season. His nearest dominator is neither the best center in the NBA, Shaquille O'Neal, and or the NBA MVP, Kevin Garnett, a forward from Minnesota Timberwolves with 6.11 feet height and 220 pounds weight, and a statistics of 82(GP), 22.2(PPG), 13.5(RPG) and 5.7(APG). Instead, he is a young center in Phoenix Suns, Amare Stoudemire. Amare is 6.10 feet high and weighs 245 pounds with a record of 80(GP), 26.0(PPG), 8.9(RPG) and 1.6(APG).

Based on the Rosen-MacDonald's superstar theory[7], the players' salary can be estimated by a regression of different technical statistics. This theory can be represented by our profitability hyperplane, which can be thought of as a measure for deciding a NBA basketball player's payment. Any player under or on this profitability hyperplane receives a salary that is well-deserved according to his talent.

In our experiments, the profitability hyperplane was chosen according to a simplified version of this superstar theory as follows: 0.03*GP + 0.20*PPG + 0.32*RPG + 0.45*APG. We found the top-3 LDPQ players as shown in Table 2. While these players are not the absolute top players, they meet the profitability constraint and outperform other players with similar values for the *spatial attributes*: height, weight and position. These players can only be "dominated" by those players with significantly different *spatial attribute* values, i.e. by those players with much better physical conditions or playing different positions. For example, D. Wade, a guard in Miami Heat, is best among all the guards with similar height and weight, but he has a "dominator" L. James (6.4 feet, 212 pounds) in Cleveland Cavaliers, who has the advantage of a height of 6.8 feet and a weight of 240 pounds. So from the point of performance-salary trade-off, it is desirable for a team to hire these players.

Name	GP	PPG	RPG	APG	Salary	Height	Weight	position
D.Wade	77	24.1	5.2	6.8	2.8m	6.4f	212b	guard
S.Marion	81	19.4	11.3	1.9	12m	6.7f	228b	forward
E.Brand	81	20.0	9.5	2.6	12m	6.8f	254b	forward

Table 2: Top-3 LPDQ NBA players

If we set $\delta = 5$, and keep the same hyperplane as before, the top-3 ML2DQ players are shown in Table 3. We can observe that the results are quite interesting too. For example, R. Davis (6.7feet 195pounds) is a guard in Minnesota Timberwolves whose nearest

[7]http://www.westga.edu/ bquest/2005/nba/NBA1.htm

Name	GP	PPG	RPG	APG	Salary	Height	Weight	position
C.Maggette	66	22.2	6.0	3.4	7m	6.6f	225b	forward
R.Hamilton	76	18.7	3.9	4.9	7.8m	6.7f	193b	guard
R.Davis	82	16.0	3.0	3.0	5.4m	6.7f	195b	guard

Table 3: Top-3 ML2DPQ players

dominator is Kevin Garnett (6.11 feet,220 pounds). The distance of height and weight between these two players is large enough ($> \delta$), which means Davis is a strong player among players with similar physical condition and position. Only C.Maggette and R.Hamilton have similarly small difference between their actual salary and the profitability constraint.

To conclude, our experiments on synthetic datasets demonstrate the efficiency and scalability of our methods for processing NDP, LDPQ and ML2DQ queries. In addition, we show that the asymmetrical algorithms consistently and significantly outperform their symmetrical counterparts.

7. CONCLUSION

Skyline queries have recently emerged as a promising paradigm for decision support. These queries find objects that are outstanding, i.e. cannot be dominated, in terms of a set of attributes to be minimized or maximized. In this paper, we have introduced three novel types of skyline queries, so-called neighborhood dominant queries, that exploit not only min/max attributes but also spatial attributes. Such queries support a micro-economic approach to decision making, considering not only the quality but also the cost of solutions. To efficiently process the proposed neighborhood dominant queries, we presented symmetrical as well as asymmetrical index-based methods. While the symmetrical approach has the advantage of using off-the-shelf index structures, our experimental evaluation shows that the asymmetrical approach clearly performs better for a wide range of synthetic datasets. Our evaluation on the NBA Great Players dataset demonstrates that the proposed new query types produce meaningful and interesting results.

This paper suggests several promising directions for future research. From a practical point of view, the integration of the proposed query types into SQL and their treatment by the query optimizer of a DBMS deserve further investigation. A more theoretical question is what other query types may be defined in our framework taking into account min/max and spatial attributes. Corresponding efficient query processing algorithms will have to be developed. Finally, in the spirit of the micro-economic framework, methods for ranking the usefulness of query results would be desirable, in particular in the case of large databases with long result lists. Such an approach could bridge the gap between the two alternative paradigms of skyline queries and rank-aware query processing.

8. REFERENCES

[1] F. Ramsak D. Kossmann and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *VLDB*, 2002.

[2] G. Fu D. Papadias, Y. Tao and B. Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD*, 2003.

[3] H. T. Kung et. al. On finding the maxima of a set of vectors. In *JACM*, 22(4), 1975.

[4] K. Tan et al. Efficient progressive skyline computation. In *VLDB*, 2001.

[5] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD Conference*, pages 47–57, 1984.

[6] C.H. Papadimitriou J. Kleinberg and P. Raghavan. Segmentation problems. In *STOC*, 1998.

[7] C.H. Papadimitriou J. Kleinberg and P. Raghavan. A microeconomic view of data mining. In *Data Min. Knowl. Discov.*, 2(4): 311-322, 1998.

[8] Cuiping Li, Beng Chin Ooi, Anthony K. H. Tung, and Shan Wang. Dada: a data cube for dominant relationship analysis. In *SIGMOD Conference*, pages 659–670, 2006.

[9] J. Matousek. Computing dominances in e^n. In *Inf. Process. Lett.*, 1991.

[10] F. Nielsen. Output-sensitive peeling of convex and maximal layers. In *Thesis*, 1996.

[11] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. Progressive skyline computation in database systems. *ACM Trans. Database Syst.*, 30(1):41–82, 2005.

[12] F. P. Preparata and M. I. Shamos. Computational geometry: An introduction. In *Springer-Verlag*, 1985.

[13] Nick Roussopoulos, Stephen Kelley, and Frédéic Vincent. Nearest neighbor queries. In *SIGMOD Conference*, pages 71–79, 1995.

[14] D. Kossmann S. Borzsonyi and K. Stocker. The skyline operator. In *ICDE*, 2001.

[15] I. Stojmenovic and M. Miyakawa. An optimal parallel algorithm for solving the maximal elements problem in the plane. In *Parallel Computing*, 1988.

[16] R. L. Rivest T. Cormen, C. E. Leiserson and C. Stein. Introduction to algorithms, second edition. In *The MIT Press*, 2001.

[17] R. Ramakrishnan T. Zhang and M. Livny. Birch:an efficient data clustering method for very large databases. In *SIGMOD*, 1996.

[18] Anthony K. H. Tung W. Jin and J. Han. Mining top-n local outliers in very large databases. In *KDD*, 2001.

[19] J. X. Zheng W.-T. Balke, U. Guntzer. Efficient distributed skylining for web information systems. In *EBDT*, 2004.

[20] Wei Wang Xuemin Lin, Yidong Yuan and Hongjun Lu. Stabbing the sky:efficient skyline computation over sliding windows. In *ICDE*, 2005.