Mining Favorable Facets [*]

Raymond Chi-Wing Wong[1] Jian Pei[2] Ada Wai-Chee Fu[1] Ke Wang[2]
[1] The Chinese University of Hong Kong, {cwwong,adafu}@cse.cuhk.edu.hk
[2] Simon Fraser University, Canada, {jpei,wangk}@cs.sfu.ca

ABSTRACT

The importance of dominance and skyline analysis has been well recognized in multi-criteria decision making applications. Most previous studies assume a fixed order on the attributes. In practice, different customers may have different preferences on nominal attributes. In this paper, we identify an interesting data mining problem, *finding favorable facets*, which has not been studied before. Given a set of points in a multidimensional space, for a specific target point p we want to discover with respect to which combinations of orders (e.g., customer preferences) on the nominal attributes p is not dominated by any other points. Such combinations are called the favorable facets of p.

We consider both the effectiveness and the efficiency of the mining. A given point may have many favorable facets. We propose the notion of minimal disqualifying condition (MDC) which is effective in summarizing favorable facets. We develop efficient algorithms for favorable facet mining for different application scenarios. The first method computes favorable facets on the fly. The second method pre-computes all minimal disqualifying conditions so that the favorable facets can be looked up in constant time. An extensive performance study using both synthetic and real data sets is reported to verify their effectiveness and efficiency.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining & Spatial databases and GIS

General Terms

Algorithms, Theory, Performance, Experimentation

Keywords

skyline, preference, spatial data, materialization, warehouse

*R. C-W Wong and A. W-C Fu are supported in part by the RGC Earmarked Research Grant of HKSAR CUHK 4120/05E. J. Pei is supported in part by NSERC Discovery Grant 312194-05. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

1. INTRODUCTION

Dominance analysis is important in many multi-criteria decision making applications. As an example, consider a customer looking for a vacation package to San Jose using three criteria: price, hotel-class and number of stops. For two packages p and q, if p is better than q in one factor, and is not worse than q in any other factors, then p is said to *dominate* q. For example, we have three packages (price, hotel-class, number-of-stops): p_1 $(1600, 4, 1)$, p_2 $(3000, 5, 2)$ and p_3 $(2000, 3, 2)$. We know that lower price, higher hotel class and less stops are more preferable. Thus, p_1 dominates p_3 because p_1 has lower price, higher class and less stops. Package p_2 does not dominate p_3 because p_3 has lower price than p_2. Similarly p_3 does not dominate p_2 because p_2 has higher hotel class than p_3. A point that is not dominated by any other point is said to be a *skyline point* or it is in the *skyline.*

Recently, skyline analysis [16, 14] has received a lot of interest from both research and applications. Continuing with our example of vacation packages. Package p is in the skyline if it is not dominated by any other packages. The packages in the skyline are the best possible tradeoffs among the three factors in question. For example, p_1 is in the skyline because it is not dominated by p_2 and p_3. Similarly, p_2 is also in the skyline. However, p_3 is not in the skyline because p_3 is dominated by p_1.

In order to conduct a skyline analysis, an (either total or partial) order is assumed on each attribute to reflect the users' preference. In our example, lower price, higher hotel class and less stops are more preferable. Most previous studies [16, 14, 15, 2, 3, 4] assume that orders on attributes are predefined for all users. Recently, some studies [6, 5, 11, 10] consider that the orders on attributes are different for different users. However, in some situations, we may be interested in the orders that can make a given point a skyline point.

Consider a skyline analysis on selecting vacation packages. Table 1 shows a synthetic data set as our running example. Unlike the attributes price and hotel-class on which a total order exists for all customers, for the attribute hotel-group, different users may have different preferences. We refer to such an attribute which does not come with a fixed order for all users a *nominal attribute.*

Nominal attributes are common in data analysis. For choosing vacation packages, some commonly encountered nominal attributes include hotel-group and airline. It is easy to name a few other important business applications, such as realities (where type of realty, style and regions are examples of nominal attributes) and selecting air flights (where airline and transition airport are examples of nominal attributes).

What challenges do nominal attributes bring to skyline

Package ID	Price	Hotel-class	Hotel-group
a	1600	4	T (Tulips)
b	2400	1	T (Tulips)
c	3000	5	H (Horizon)
d	3600	4	H (Horizon)
e	2400	2	M (Mozilla)
f	3000	3	M (Mozilla)

Table 1: Table which contains a set of packages

Customer	Preference on Hotel-group	Skyline
Alice	$T \prec M$	{ a, c }
Bob	No special preference	{ a, c, e, f }
Chris	$H \prec M$	{ a, c, e }
David	$H \prec M \prec T$	{ a, c, e }
Emily	$H \prec T \prec M$	{ a, c }
Fred	$M \prec T$	{ a, c, e, f }

Table 2: Table which contains customer preferences

analysis? The change of preference orders on nominal attributes may lead to changes in skylines. Due to the changes of the orders on nominal attributes, some skyline points may become dominated by other points and some points that are not in the original skyline may become so if they are preferred by the updated order.

Example 1 (Skylines in Different Preferences). Consider the packages in Table 1 and the customer preferences in Table 2. In Table 1, the numeric attributes, price and hotel-class, are totally ordered. The lower the price and the higher the hotel-class, the more preferable a vacation package is. Hotel-group is a nominal attribute. Different customers have different preferences on that attribute. In Table 2, "$T \prec M$" denotes that the customer prefers Tulips to Mozilla.

When a customer such as Fred prefers Mozilla to Tulips, package f is in the skyline. However, for another customer such as Alice who prefers Tulips to Mozilla, f is not in the skyline anymore since it is dominated by a. Similarly, for a customer such as Chris preferring Horizon to Mozilla, f is not in the skyline, either, since it is dominated by c. Another interesting observation is that packages a and c are always in the skyline no matter what preference order is chosen (because a has the lowest price and c has the highest hotel class). Also, b and d are always dominated. ■

From the above example, we observe that, with different preferences, a particular package may or may not be in the skyline. How is such information useful in applications?

Example 2 (Applications). Let us consider the packages in Table 1 and the customer preferences in Table 2 again. Suppose that hotel-group Mozilla wants to promote its own packages to potential customers. The best strategy is to promote to those customers who may choose its packages rather than other packages. Which customers should Mozilla target for marketing?

Consider package f of hotel-group Mozilla. As described in Example 1, Fred prefers Mozilla to Tulips and f is in the skyline in this case. Thus, Fred should be one of the targets for promotion. On the other hand, Alice and Chris should not be in the target list for promotion of f since f is not in the skyline with respect to their preference orders. The targets for promotion are those customers who include f in their skyline sets. Similarly, we can also consider package e for promotion. Table 3 shows whether we should promote packages e and f to each customer, respectively. Table 4 shows the conditions under which e or f will be disqualified as skyline points. How to find these conditions will be the core of our study. These conditions give the hotel-group better understanding of the reason why a package is favorable or not favorable. Therefore, they are considered interesting and useful information which is hidden in the data. ■

Customer	Package e	Package f
Alice	N	N
Bob	Y	Y
Chris	Y	N
David	Y	N
Emily	N	N
Fred	Y	Y

Table 3: Recommendation of packages to customers in Example 2

Package ID	Disqualifying Conditions
e	$T \prec M$
f	$T \prec M \vee H \prec M$

Table 4: Disqualifying Conditions in Example 2

From the above examples, we conclude that whether a customer is in the target list of a particular package depends on his/her preference. Given a particular package, it is interesting to find a set of preference conditions that favor the package. We model this task as mining favorable facets. This information is valuable to target the right customers whose preferences are consistent with the orders.

Generally, given a set of points in a multidimensional space where some nominal attributes are present, for a point p, **favorable facet mining** is to find the orders with respect to which p is in the skyline. With favorable facets, we can understand not only whether p is in the skyline of a preference, but also the conditions on the dimensions that can affect the skyline membership of p.

While enumerating all favorable facets is helpful, there can be a large number of such orders. In the example, in Table 1, as long as Tulip does not dominate Mozilla, e will be a skyline point. The favorable facets can be $M \prec T \prec H$, $T \prec H$, $H \prec T$, $H \prec M$, $M \prec H \prec T$, $M \prec T$, $M \prec H$, or no preference on hotel. Therefore, instead of returning all favorable facets, returning a succinct summarization of the facets such as the disqualifying conditions is more useful.

A naive method to compute the favorable facets is to enumerate the skylines for all possible combinations of the orderings of the nominal attributes. However, it is not scalable since there will be an exponential number of possible combinations. In this paper, we propose much more efficient ways of mining a summarization of the favorable facets.

To the best of our knowledge, the problem of mining favorable facets has not been identified before. Favorable facets can disclose novel information about skylines that cannot be addressed by the existing methods. We believe that this is the first work to study the problem.

In addition to identifying a new data mining problem, we also give effective and efficient solutions. Returning all favorable facets is expensive because there are typically many such orders. Also, it is difficult for the user to understand many orders in the result set. To tackle this problem, we propose to use a succinct representation called the minimal disqualifying condition (MDC) which can summarize favorable facets and is meaningful to the user.

In order to conduct efficient mining, we develop two algorithms. The first method finds on-the-fly the minimal conditions on the preference orders that disqualify a query point from the skyline. Using those minimal disqualifying conditions, we can answer querying of favorable facets effectively.

The second method pre-computes all minimal disqualifying conditions of every data point, and organize them in a compact way. Then, favorable facets can be analyzed online.

We present a systematic performance study using both real data sets and synthetic data sets to verify the effectiveness and the efficiency of our methods. The experimental results show that mining favorable facets is interesting, and our proposed methods are efficacious.

The rest of the paper is organized as follows. We discuss related work in Section 2. In Section 3, we investigate the changes of skylines due to changes of orders on attributes, and propose the notion of minimum disqualifying conditions (MDC). The MDC On-the-fly method and the MDC materialization method are developed in Section 4. A systematic performance study is reported in Section 5. The paper is concluded in Section 6.

2. RELATED WORK

Skyline queries have been studied since 1960s in the theory field where skyline points are known as *Pareto sets* and *admissible points* [9] or *maximal vectors* [8]. However, earlier algorithms such as [8, 7] are inefficient when there are many data points in a high dimensional space. The problem of skyline queries was introduced in the database context in [1].

We can categorize the existing work into two major groups – *full-space skyline queries* and *subspace skyline queries*.

Many efficient methods have been proposed for full-space skyline queries which return the set of skyline points in a specific space. Some representative methods include a bitmap method [16], a nearest neighbor (NN) algorithm [14], and branch and bound skylines (BBS) method [15].

Most of the existing studies assume that only numeric attributes present. Some recent studies [2, 3, 4, 6, 5, 11, 10] consider partially-ordered attributes. For example, [2, 3] transform each partially-ordered attribute into two integer attributes. Then, the conventional skyline algorithms can be applied. [4] studies estimating the cost of skyline operator involving the partial ordered attribute.

Nevertheless, most existing work assumes that *each attribute has only one order: either a total or a partial order.* In this paper, we consider the scenarios where different users may have different preferences on nominal attributes. That is, more than one order need to be considered in nominal attributes.

Some latest works consider skylines on nominal attributes. [6, 5] study how to specify a query based on preferences on nominal attributes. When preferences change, the query results can be incrementally refined. In [11], a user can specify some values in nominal attributes as an equivalence class to denote the same "importance" for those values. Whenever a value v has a higher preference than a value v' in the equivalence class, v also has a higher preference than all the other values in the equivalence class. [10] is an extension of [11]. In [10], whenever a user performs a query and obtains the results, if s/he finds that there are a lot of irrelevant results, s/he can modify the query by adding more conditions in the query so that the result set is smaller to suit her/his need.

Clearly, all those studies still focus on skyline computation. In this paper, we study the problem of favorable facet mining. The output is a representation for the facets. No previous studies has addressed the problem studied here.

3. SKYLINES AND ORDERS

A skyline analysis involves multiple attributes. A user's preference on the values in an attribute can be modeled by a partial order on the attribute. A *partial order* $\preceq$ is a reflexive, asymmetric and transitive relation. A partial order is also a total order if, for any two values u and v in the domain, either $u \preceq v$ or $v \preceq u$. We write $u \prec v$ if $u \preceq v$ and $u \neq v$. A partial order also can be written as $R = \{(u, v) | u \preceq v\}$. $u \preceq v$ also can be written as $(u, v) \in R$.

By default, we consider points in an m-dimensional[1] space $\mathbb{S} = D_1 \times \cdots \times D_m$. For each dimension D_i, we assume that there is a partial or total order R_i. For a point p, $p.D_i$ is the projection on dimension D_i. If $(p.D_i, q.D_i) \in R_i$, we also write $p.D_i \preceq q.D_i$ or simply $p \preceq_{D_i} q$. We can omit D_i if it is clear from the context.

For points p and q, p *dominates* q, denoted by $p \prec q$, if, for any dimension $D_i \in \mathbb{S}$, $p \preceq_{D_i} q$, and there exists a dimension $D_{i_0} \in \mathbb{S}$ such that $p \prec_{D_{i_0}} q$. If p dominates q, then p is more preferable than q according to the preference orders. The *dominance relation* R can be viewed as the integration of the preference partial orders on all dimensions. Thus, we can write $R = (R_1, \ldots, R_m)$. It is easy to see that the dominance relation is a strict partial order.

Given a data set $\mathcal{D}$ containing n points in space $\mathbb{S}$, a point $p \in \mathcal{D}$ is in the *skyline* of $\mathcal{D}$ (i.e., a *skyline point* in $\mathcal{D}$) if p is not dominated by any points in $\mathcal{D}$. The skyline of $\mathcal{D}$, denoted by $SKY(\mathcal{D})$, is the set of skyline points in $\mathcal{D}$.

In an application, there often exist some orders on the dimensions that hold for all users. In our running example as shown in Table 1, a lower price and a higher hotel-class are always more preferred by customers. Even, for nominal attributes, there may exist some universal partial orders. For example, in a realty market, detached houses are often more preferred than semi-detached houses. Hence, we assume that we are given a *template*, which contains a partial order for every dimension. The partial orders in the template are applicable to all users. Each user can then express his/her specific preference by refining the template. The containment relation of orders captures the refinement.

For partial orders R and R', R' is a *refinement* of R, denoted by $R \subseteq R'$, if for any $(u, v) \in R$, $(u, v) \in R'$. Moreover, if $R \subseteq R'$ and $R \neq R'$, R is said to be *weaker* than R' and R' be *stronger* than R.

EXAMPLE 3 (PREFERENCE ORDERS). Let $R = \{(T, M)\}$ and $R' = \{(T, M), (H, M)\}$. Then, $R \subseteq R'$. That is, R' is a refinement of R by adding a preference $H \prec M$. As $R \neq R'$, R is weaker than R' and R' is stronger than R. ■

[1]In this paper, we use the terms "*attribute*" and "*dimension*" interchangeably.

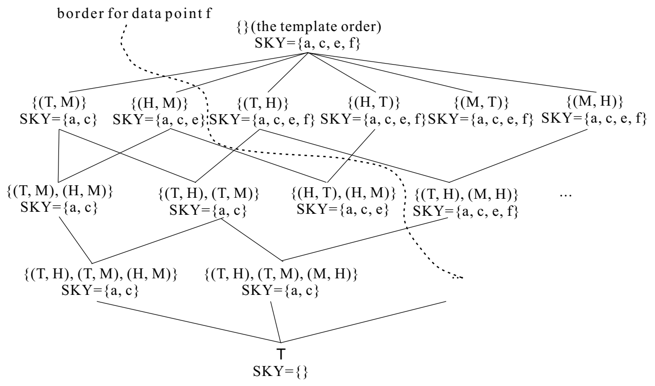

Figure 1: The lattice of refinement orders

PROPERTY 1. *For orders $R = (R_1, \ldots, R_m)$ and $R' = (R'_1, \ldots, R'_m)$, $R \subseteq R'$ if and only if $R_i \subseteq R'_i$ for $1 \le i \le m$.* ■

Now, we are ready to define the problem of favorable facets formally.

DEFINITION 1 (FAVORABLE FACETS). If p is a skyline point with respect to an order R_p, R_p is called a *favorable facet* of p. Given a data point p and a template R, the problem of *mining favorable facets* is to find all R_p that are favorable facets of p and are refinements of R (i.e. $R \subseteq R_p$). ■

3.1 A Naïve Method: Lattice Search

How can we compute the favorable facets of a point p? Let us consider a naïve approach which computes all skyline points with all possible refinements of a given template R. After the computation, all refinements with respect to which p is a skyline point are selected as the favorable facets of p.

All the possible refinements of a given template order form a partially ordered set (poset) L where the ordering is governed by the refinement relationship, and the template order serves as the unit(greatest) element in L. For example, all possible orders on attribute hotel-group in our running example form a poset as shown in Figure 1. In our running example, the template order R on dimension hotel-group is $\emptyset$. That is, the template does not prefer any hotel-group to another. Each node in the figure is associated with a refinement of the template. The skylines with respect to the orders are also shown. We write the orders in the form of a fully expanded transitive closure. For example, we do not have a node labeled $\{(T, H), (H, M)\}$ because its transitive closure is $\{(T, H), (H, M), (T, M)\}$. The node at the top corresponds to the template order. The node at the bottom is a special node which corresponds to a zero(least) element $\top$, and with this node, the poset L forms a lattice. The skyline with respect to $\top$ is defined to be $\emptyset$.

In the lattice, b is not a skyline point because it is dominated by a on attributes price and hotel-class and it has the same value as a on hotel-group. Interestingly, although e and f are less preferable than a on attributes price and hotel-class, a does not dominate e nor f since the template does not prefer T to M. In fact, both e and f are skyline points with respect to the template.

From Figure 1, we observe an important monotonicity property.

THEOREM 1 (MONOTONICITY). *Given a data set $\mathcal{D}$ and a template R, if p is not in the skyline with respect to R, then p is not in the skyline with respect to any refinement R' of R.*

Proof. Consider order R and points p and q such that $q \preceq p$ with respect to R. Then, for any attribute D, we have $q \preceq_D p$ with respect to R. Let R' be a refinement of R where $R \subseteq R'$. Then, $q \preceq_D p$ still holds in R'. That is, $q \preceq p$ still holds in R'. ■

Theorem 1 indicates that, when the orders on dimensions are strengthened, some skyline points may be disqualified. However, a non-skyline point never gains the skyline membership due to a stronger order. This monotonic property greatly helps in analyzing skylines with respect to various orders.

When there are multiple nominal attributes, each attribute has a corresponding lattice based on the template and refinements. The lattice for multiple attributes is the product of the lattices for individual attributes. The properties for the single nominal attribute carry forward to the product lattice.

Note that this lattice is different from SKYCUBE [17] because this lattice has the monotonic property which does not generally exist in SKYCUBE.

3.2 Minimal Disqualifying Conditions (MDC)

Although the lattice helps us to mine favorable facets, there can be many favorable facets for one query point. It is difficult for users to comprehend a large number of favorable facets. How can we represent the mining result in a concise and meaningful way?

EXAMPLE 4. In Table 1, with respect to what refinements of the template R that f is in the skylines? The refinement order lattice in Figure 1 can be used to find all favorable facets.

f is in the skyline provided that none of the following two conditions hold: $T \prec M$ (otherwise, $a \prec f$) or $H \prec M$ (otherwise $c \prec f$). Thus, (T, M) and (H, M) are called the *minimal disqualifying conditions*.

In the refinement order lattice, if an order R_x contains neither $T \prec M$ nor $H \prec M$, any order weaker than R_x does not contain these conditions. On the other hand, if an order R_x contains one of these two conditions, so does every order stronger than R_x. Based on the monotonicity property, there exists a *border* between the orders that qualify f as a skyline point and those that disqualify f which partitions the lattice into two parts: f is a skyline point in the upper part and is not a skyline point in the lower part, as shown in Figure 1. ■

Based on the observations in Example 4, to mine favorable facets, we only need to compute the border. Particularly, we can use the set of minimal disqualifying conditions.

Let R and R' be two partial orders. R and R' are *conflict-free* if there exist no values u and v such that $u \neq v$, $(u, v) \in R$, and $(v, u) \in R'$. $R \cup R'$ is still a partial order if R and R' are conflict-free. For example, if $R = \{(T, H)\}$ and $R' = \{(T, M)\}$, R and R' are conflict-free. If $R = \{(T, H)\}$ and $R' = \{(H, T)\}$, R and R' are not conflict-free.

DEFINITION 2 (MINIMAL DISQUALIFYING CONDITION). For a skyline point p and a template order R, a partial order

R' is called a *minimal disqualifying condition* (or **MDC** for short) if (1) $R' \cap R = \emptyset$, (2) R' and R are conflict-free, (3) p is not a skyline point with respect to $R \cup R'$, and (4) there exists no R'' such that $R'' \subset R'$ and p is not a skyline point with respect to $R \cup R''$. The set of minimal disqualifying conditions for p is denoted by $MDC(p)$. ■

Example 5 (Minimal disqualifying condition). In our running example, $R' = \{(T, M)\}$ and $R'' = \{(H, M)\}$ are the minimal conditions that disqualify f as a skyline point. They are the minimal disqualifying conditions of f. That is, $MDC(f) = \{\{(T, M)\}, \{(H, M)\}\}$. ■

Based on the monotonicity in Theorem 1, we can show that minimum disqualifying conditions can be used to summarize favorable facets effectively.

Theorem 2 (MDC). *For a point p and a template order R, p is in the skyline with respect to a refinement R' of R if and only if (1) p is in the skyline with respect to R, and (2) $\forall R_x \in MDC(p)$, $R_x \not\subseteq R'$.* ■

4. ALGORITHMS

In this section, we develop efficient algorithms to compute MDCs for favorable facet mining. We describe the MDC On-the-fly method in Section 4.1 and the MDC Materialization method in Section 4.2.

4.1 MDC-O: Computing MDC On-the-fly

For two skyline points p and q with respect to the template order $R = (R_1, \ldots, R_m)$, two cases may arise.

Case 1: There is an attribute D_{i_0} such that $p \prec_{D_{i_0}} q$. Then, q never dominates p in any refinement of R, and thus cannot lead to a minimal disqualifying condition for p. For example, in Table 1, c never dominates a because the Price value of a is smaller than that of c.

Case 2: There does not exist any attribute D_{i_0} such that $p \prec_{D_{i_0}} q$. That is, p never dominates q on any dimension. Then, q may dominate p with respect to some refinements of R, and thus may lead to a minimal disqualifying condition of p.

Formally, q *quasi-dominates* p if $(p.D_i, q.D_i) \notin R_i$ for $1 \leq i \leq m$. For example, in Table 1, a quasi-dominates f because a has a lower price and a higher hotel-class than f. The only reason that a does not dominate f in the template is that T does not dominate M. As shown in Example 5, in a refinement $R' = \{(T, M)\}$ of R, a dominates f. Thus, R' is a minimal disqualifying condition for f.

Given a template order $R = (R_1, \ldots, R_m)$, if q quasi-dominates p, the *minimum condition* that q dominates p is $R^{q \to p} = \{(q.D_i, p.D_i) | q.D_i \neq p.D_i \text{ and } (q.D_i, p.D_i) \notin R_i \text{ for } 1 \leq i \leq m\}$. It is easy to see that q dominates p in $(R \cup R^{q \to p})$. $R^{q \to p}$ strengthens the template in a minimal way such that q dominates p.

Based on the above idea, we have the MDC-O (for MDC on-the-fly) algorithm as shown in Algorithm 1 to compute the minimal disqualifying conditions of a point p as follows. Let R be the template order. We assume that in a pre-processing step the skyline $SKY(R)$ has been computed. Let $SKY(R) = \{p_1, \ldots, p_k\}$. We check whether p is in $SKY(R)$. If no, we report to the user that p is not in the skyline with respect to any refinement of R. Otherwise, we initialize $MDC(p)$ to $\emptyset$. For each p_j where $1 \leq j \leq k$ and $p_j \neq p$ such that p_j quasi-dominates p, if $MDC(p)$ does not have a condition weaker than $R^{p_j \to p}$, then we insert $R^{p_j \to p}$ into $MDC(p)$ and remove any stronger conditions in $MDC(p)$.

Algorithm 1 MDC On-the-fly Method **MDC-O**(p, $\mathcal{D}$, R)

```
Input: a data point $p$, data set $\mathcal{D}$, order template $R$, and
    the skyline with respect to $R$ $SKY(R)$
Output: the minimal disqualifying conditions of $p$
 1: suppose that the skyline $SKY(R)$ is $\{p_1, \ldots, p_k\}$
 2: if $p \notin SKY(R)$ then
 3:    return that $p$ cannot be a skyline point with respect
       to any refinement of $R$
 4: else
 5:    set $MDC(p) = \emptyset$
 6:    for $j := 1$ to $k$ do
 7:       if $p \neq p_j$ and $p_j$ quasi-dominates $p$ then
 8:          if $MDC(p)$ does not have a condition weaker
             than $R^{p_j \to p}$ then
 9:             add $R^{p_j \to p}$ to $MDC(p)$ and remove any
                stronger conditions in $MDC(p)$
10:    return $MDC(p)$
```

Here, if there are two disqualifying conditions and one is stronger than the other, we store the weaker one. Since both conditions disqualify a data point as a skyline point, the stronger one is redundant and can be removed.

The algorithm assumes a pre-processing step for computing the skyline points with respect to R. One can adopt any existing algorithm (e.g., [2, 3]) for computing the skyline for partially ordered domains.

Let us analyze the complexity of MDC-O. We have to check whether $p \in SKY(R)$, which takes $O(k)$ time, where k is the number of skyline points with respect to R. For each skyline point p_j quasi-dominating p, the minimum condition is computed with cost $O(m')$, where m' is the number of nominal dimensions. The minimum conditions are inserted into the set of minimal disqualifying conditions if the current set does not have a weaker condition. The insertion takes cost $O(l)$, where l is the maximum number of minimal disqualifying conditions of a skyline point. Thus, the overall complexity is $O((m' + l)k)$.

4.2 MDC-M: A Materialization Method

When there are many skyline points in $SKY(R)$, the MDC-O algorithm can be costly and may not return the MDCs in real time. In some applications, online response may be required. For example, a salesperson may want to check online whether a product is in the skyline with respect to a customer's preference.

In order to meet the online response requirement, one feasible solution is to materialize all minimal disqualifying conditions of all data points in the data set. We propose an algorithm as shown in Algorithm 2 for this purpose.

The algorithm consists of two parts. The first part computes the skyline points with respect to R. The second part finds all MDCs.

Let us analyze the complexity of the second part of our algorithm. Since we have to check every pair of skyline points p and q, and derive the corresponding minimal disqualifying conditions for p over q and for q over p, the overall complexity is $O((m' + l)k^2)$. However, the lookup of the MDC for a given point p will be very fast. If an array is used to store

Algorithm 2 MDC Materialization Method **MDC-M**$(\mathcal{D}, R)$

```
Input: data set $\mathcal{D}$ and order template $R$
Output: the set of skyline points with respect to $R$ and
    their minimal disqualifying conditions
1:  compute the skyline with respect to $R$, suppose the skyline $SKY(R)$ is $\{p_1, \ldots, p_k\}$
2:  set $MDC(p_i) = \emptyset$ for $(1 \le i \le k)$
3:  for $i := 1$ to $(k-1)$ do
4:    for $j := (i+1)$ to $k$ do
5:      if $p_i$ quasi-dominates $p_j$ then
6:        if $MDC(p_j)$ does not have a condition weaker than $R^{p_i \to p_j}$ then
7:          add $R^{p_i \to p_j}$ to $MDC(p_j)$ and remove any stronger conditions in $MDC(p_j)$
8:      if $p_j$ quasi-dominates $p_i$ then
9:        if $MDC(p_i)$ does not have a condition weaker than $R^{p_j \to p_i}$ then
10:         add $R^{p_j \to p_i}$ to $MDC(p_i)$ and remove any stronger conditions in $MDC(p_i)$
11: return $MDC$
```

all MDCs, it takes constant time or $O(1)$ time to reach the required MDC.

Example 6 (The MDC Materialization Method). Let us illustrate Algorithm 2 using Table 5, which is an extension of Table 1 by adding one more nominal attribute, airline. Suppose that the template R on hotel-group and airline are set to $\emptyset$.

For the first step, we compute the skyline with respect to R, which is $\{a, c, d, e, f\}$. We can remove data point b because b is not in the skyline with respect to R.

Then, we check whether any skyline point quasi-dominates another. We compare a and c first, which do not quasi-dominate each other. Then, we compare a and d. a quasi-dominates d. The minimum condition that a dominates d $R^{a \to d} = \{(T, H), (G, R)\}$. Similarly, we compare a and e and obtain $R^{a \to e} = \{(T, M), (G, R)\}$. We compare a and f and obtain $R^{a \to f} = \{(T, M), (G, W)\}$. The MDC of each data point after a is compared with all other points is shown in Table 6.

In the second iteration, we compare c with other points. First, we compare c and d. c quasi-dominates d. The minimum condition that c dominates d $R^{c \to d} = \{(G, R)\}$. From Table 6, the current MDC of d contains only $\{(T, H), (G, R)\}$, which is stronger than $\{(G, R)\}$. Thus, we insert $\{(G, R)\}$ and remove $\{(T, H), (G, R)\}$. Similarly, we compare c with e and f. We obtain the MDC as shown in Table 7 after processing c.

Similarly, we continue with the remaining iterations. It can be verified that the final MDCs are as shown in Table 7. ■

4.3 Indexing for Speed-up

When the given template R consists of a set $\mathcal{T}$ of totally ordered attributes and a set $\mathcal{N}$ of nominal attributes with no total order, we can make use of an indexing structure for speeding up the process for either the on-the-fly or the materialization approach. The idea is based on the following lemma.

Lemma 1. *Given a template with a set $\mathcal{T}$ of totally ordered attributes. Given a point p, if q quasi-dominates p, then for each attribute D_i in $\mathcal{T}$, $q \prec_{D_i} p$.*

Package ID	Price	Hotel-class	Hotel group	Airline
a	1600	4	T (Tulips)	G (Gonna)
b	2400	1	T (Tulips)	G (Gonna)
c	3000	5	H (Horizon)	G (Gonna)
d	3600	4	H (Horizon)	R (Redish)
e	2400	2	M (Mozilla)	R (Redish)
f	3000	3	M (Mozilla)	W (Wings)

Table 5: A table which contains a set of packages with two nominal attributes.

Package	MDC
a	-
c	-
d	{(T, H), (G, R)}
e	{(T, M), (G, R)}
f	{(T, M), (G, W)}

Table 6: MDCs of points after comparing a to other points.

Package	MDC
a	-
c	-
d	{(G, R)}
e	{(T, M), (G, R)}
f	{(T, M), (G, W)}, {(H, M), (G, W)}

Table 7: MDCs of points after comparing c to other points.

In either MDC-O or MDC-M, the first step is to compute the skyline $SKY(R)$. An R-tree [12] can be built with dimensions only for the totally ordered attributes $\mathcal{T}$. Points in the skyline set $SKY(R)$ are inserted into this R-tree based on the attributes in $\mathcal{T}$. Then, to find points that quasi-dominates p, a range search is conducted on the R-tree with a range of $<= p.D_i$ for each dimension D_i in the R-tree. The set of points Q returned by the range search is a superset of the set of points that quasi-dominate p. We then examine the nominal attributes in $\mathcal{N}$ of each point q in Q. First we check whether $(q.D_i, p.D_i)$ for each $D_i \in \mathcal{N}$ is conflict-free with R. If there is any conflict with R, point q is discarded. Otherwise, the minimum condition that q dominates p is given by $R^{q \to p} = \{(q.D_i, p.D_i) | D_i \in \mathcal{N} \ \wedge \ q.D_i \neq p.D_i\}$. We then try to insert $R^{q \to p}$ in the same manner as in MDC-O and MDC-M.

With the R-tree search method above, the time complexity for the MDC computation in MDC-O will become $O((m + l)r)$, where $O(r)$ is cost of a range search in the R-tree. For the materialization algorithm MDC-M, the time complexity for computing the MDCs for all points becomes $O((m + 1)kr)$. The complexity for looking up the MDC of a particular point with MDC-M remains $O(1)$.

5. EMPIRICAL STUDY

We have conducted extensive experiments on a Pentium IV 3.2GHz PC with 2GB memory, on a Linux platform. The algorithms were implemented in C/C++. In our exper-

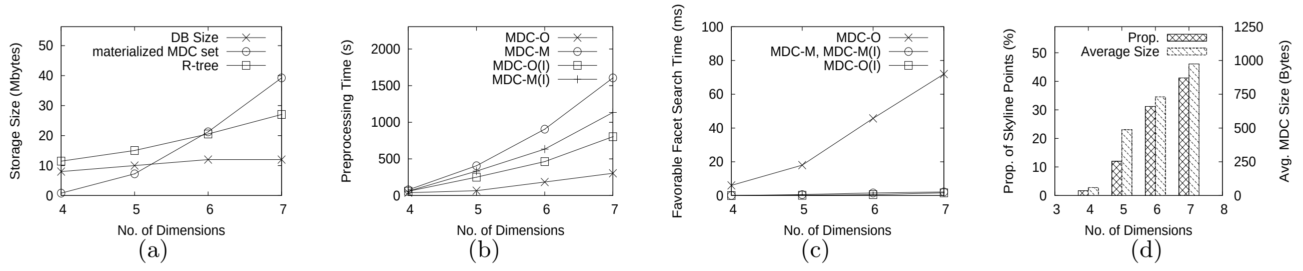

Figure 2: Scalability with respect to dimensionality where the number of numeric attributes is fixed to be 3

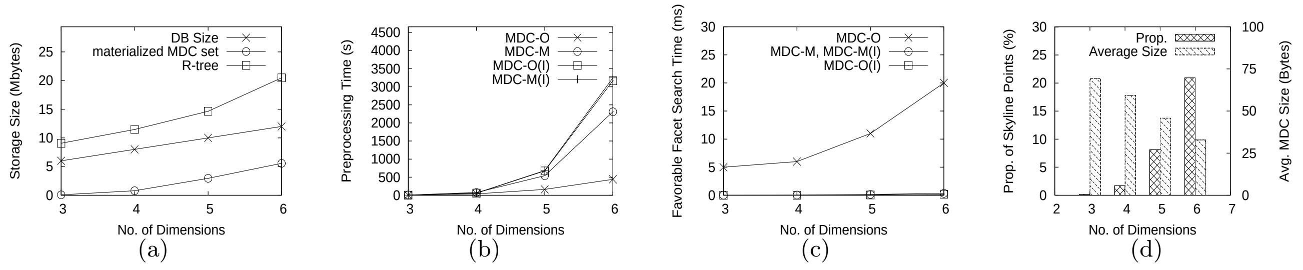

Figure 3: Scalability with respect to dimensionality where the number of nominal attributes is fixed to be 1

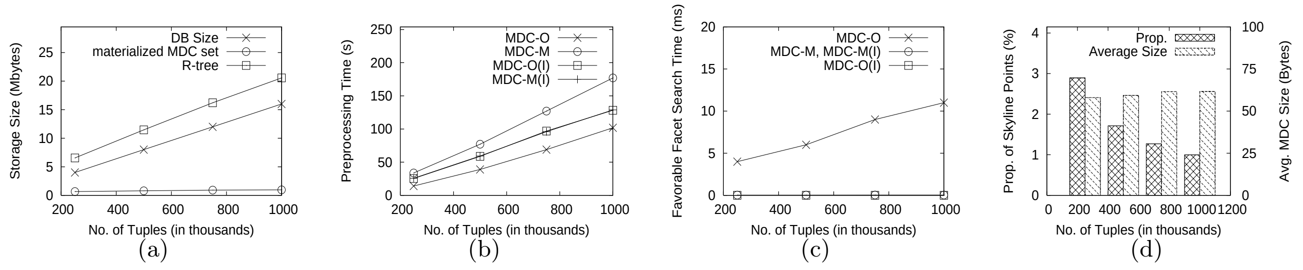

Figure 4: Scalability with respect to database size

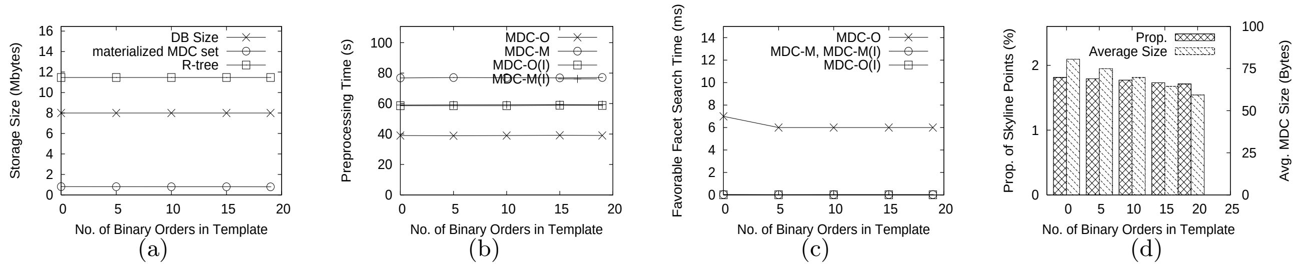

Figure 5: Scalability with respect to template size

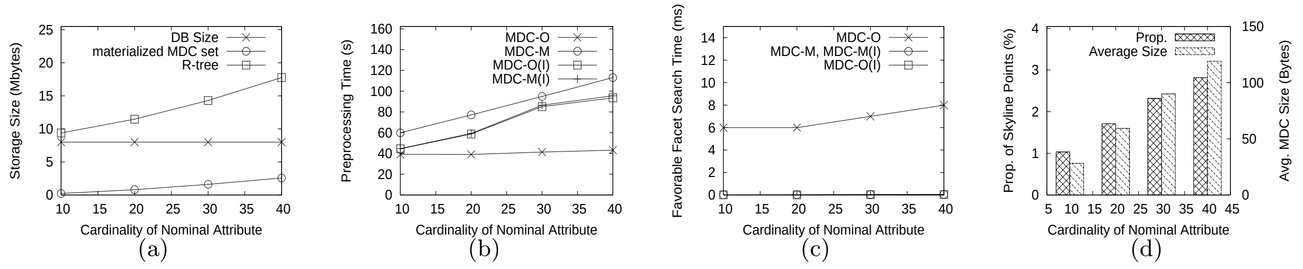

Figure 6: Scalability with respect to cardinality of nominal attribute

Parameter	Default value
No. of tuples	500K
No. of numeric dimensions	3
No. of nominal dimensions	1
No. of values in a nominal dimension	20
Zipfian parameter θ	1

Table 8: Default values

iments, we adopted the data set generator released by the authors of [1]. As this data set generates only numeric attributes, we modified the program to generate both numeric attributes and nominal attributes. The numeric attributes are totally ordered and are generated as in [1]. The nominal attributes are generated according to a Zipfian distribution [13]. By default, we set the Zipfian parameter $\theta = 1$. The default values of other experimental parameters are shown in Table 8. Note that the total number of dimensions is equal to the number of numeric dimensions plus the number of nominal dimensions. By default, we adopted a template where the most frequent value in a nominal dimension has a higher preference than all other values. For R-tree indexing, each node occupies one page size of 4Kbytes. In the default settings, with this page size, the maximum branching factor is 130, and the minimum number of children is set to 65.

We denote the on-the-fly method and the materialization method without any indexing by MDC-O and MDC-M, respectively. We denote the two methods with R-tree indexing [12] by MDC-$O(I)$ and MDC-$M(I)$, respectively.

We evaluate the performance of the algorithms in terms of (1) the memory requirements, (2) pre-processing time, and (3) the execution time of a favorable facet search, which finds the MDC for a given data point. We also evaluate the algorithms in term of the proportion of the skyline points with respect to the template and the average size of MDCs of each data point stored. For pre-processing, MDC-M and MDC-$M(I)$ construct a materialized MDC set, and MDC-$O(I)$ and MDC-$M(I)$ construct an R-tree, and all methods require the computation of the skyline set for template R, $SKY(R)$. MDC-M and MDC-$M(I)$ stores a materialized MDC set and MDC-$O(I)$ stores an R-tree for the search. The favorable facet search time of MDC-M and MDC-$M(I)$ is the lookup time in the materialized MDCs while that of MDC-O is the execution time of Algorithm 1 and that of MDC-$O(I)$ is the time of an on-the-fly search using the R-tree.

For measurements (1) and (2), each experiment was conducted 100 times and the average of the results was reported. For measurement (3), in each experiment, we randomly selected 100 data points from the data set for the favorable facet search, and the average search time is reported.

5.1 Synthetic Data Sets

Three types of data sets are generated as described in [1]: (1) *independent data sets*, (2) *correlated data sets* and (3) *anti-correlated data sets*. The detailed description of these data sets can be found in [1]. For the interest of space, we only show the experimental results for the anti-correlated data sets. The results for the independent data sets and the correlated data sets are similar in the trend but their execution times are much shorter.

5.1.1 Comparison with the Naïve Method

We compare MDC-O and MDC-M methods with the naïve method ($Naive$) described in Section 3.1. We use the default values except that the cardinality is equal to 5 because the naïve method ran out of storage when the cardinality is above 5. In the default data set, the storage size and the pre-processing time of $Naive$ are 9MB and 954,801s, respectively. In comparison MDC-M requires 22.6KB for storage and 49s for pre-processing. Consider the favorable facet search time. For the materialized methods, on average, MDC-M requires 0.001s and $Naive$ requires 22s for the search. MDC-M is much faster than the naïve method because it returns the concise MDC results. Thus, the MDC methods outperform the naïve method. In the following, we focus on the MDC methods.

5.1.2 Effect of the Number of Dimensions

We fixed the number of numeric attributes to 3 and varied the number of nominal attributes from 1 to 4. Figure 2(a) shows that the R-tree sizes increase with the number of nominal attributes.The increase in materialization size is due to the increase in the number of conditions in the MDC for each data point. When the number of nominal attributes increases, the storage size of the materialized MDC set surpasses that of the R-tree indexing. This is because the MDC sizes increase very rapidly with the number of nominal attribute while the R-tree sizes are less sensitive to this factor.

As shown in Figure 2(b), the pre-processing times of MDC-M, MDC-$M(I)$ and MDC-$O(I)$ increase with the number of nominal attributes. This is because more nominal attributes need to be compared in order to generate a minimal disqualifying condition for each data point when the number of nominal attributes increases. The pre-processing time of MDC-$O(I)$ is smaller than MDC-$M(I)$ because MDC-$M(I)$ involves the additional step of MDC materialization. MDC-M is the slowest because it does not benefit from any speedup by the indexing technique. MDC-O is the fastest because it only computes the skyline set $SKY(R)$.

From Figure 2(c), the favorable facet search times for MDC-M and MDC-$M(I)$ are close to zero since it is a simple lookup of the materialized MDC set. With increase in the number of nomimal attributes, the search time increases for MDC-$O(I)$ since there are more skyline points and the index size is larger. MDC-O needs to find the MDCs of a point from the skyline set $SKY(R)$. When there are more nominal dimensions, it takes more time. MDC-M and MDC-$M(I)$ are much faster than MDC-O because they pre-computed the result. It is interesting to see that MDC-$O(I)$ is very fast because MDC-$O(I)$ builds an index which speeds up the search. In Figure 2(d), as we expected, the number of skyline points increases when the number of nominal attributes increases. This is because there is higher chance that a data point is not dominated by other data points. Besides, it is trivial that the average size of MDCs among all points stored increases when there are more nominal attributes.

The effects of variations in the number of numeric attributes have been studied and the results are shown in Figure 3 where we fixed the number of nominal attributes to 1 and varied the number of numeric attributes from 2 to 5. As expected, both the execution time and the storage size increase with the number of numeric dimensions. Besides, we observe that the average size of MDCs decreases when the number of numeric attributes increase. When there are more numeric attributes, there is higher chance that a data point the data set is not quasi-dominated by other points. It is more likely that the data point is a skyline point with respect to any refinement of the template and thus has no MDCs stored. Thus, the average size of MDCs decreases.

5.1.3 *Effect of the Number of Tuples*

In this experiment, the number of tuples are varied from 250K to 1,000K. Figure 4(a) shows that the storage sizes of all algorithms increase with the number of tuples. When there are more tuples, $SKY(R)$ increases in size which leads to bigger R-tree and increase in the materialization size. The pre-processing time of all algorithms except MDC-O increases as shown in Figure 4(b), because more tuples are processed. The search time of MDC-O increases with the data size. In Figure 4(d), the proportion of skyline points decreases with the number of tuples because there is a higher chance that a data point is dominated by another data point when there are more data points. The average size of MDCs is not affected because it is mainly dependent on the nominal values.

5.1.4 *Effect of the Template Size*

We next studied the effect of different template sizes. Let v_c be the most frequent value in the nominal attribute. We started the experiment with a template that was an empty set. Then, one binary order was added to the template at a time. The binary order to be added was $v_c \prec v_i$, where v_i was a value in the nominal attribute, $v_i \neq v_c$, and v_i was not in the current template. We observe that there is no significant effect of the template size in Figures 5(a), (b), (c) and (d). This is because the storage size, the pre-processing time and the search time are mainly dependent on the size of $SKY(R)$, and when the template contains more orders, the number of skyline points decreases slightly. This is because the more order pairs in the template creates more chance of domination among the tuples. The average MDC size decreases also since MDC stores the conditions which do not appear in the template.

Let us call the above set of templates (A). We have also conducted the experiments with two other kinds of templates: (B) the templates where the least frequent value has a higher preference than the other values and (C) the templates in which the “median” frequent value has a higher preference than the other values, where the “median” frequent value is the value ranked in the middle of the values. The storage size for the MDC materialization is the smallest when we apply the template containing the least frequent value with a higher preference. This is because there is less chance for the data points with the lowest preference (on nominal attributes) to be quasi-dominated by the data points with the higher preference (on nominal attributes). As there are a lot of data points with the lower preference and we now do not need to store MDCs related to these data points, the storage size is thus smaller.

5.1.5 *Effect of the Cardinality of Nominal Attributes*

Figure 6 shows the results with the variation of the cardinality of a nominal attribute. In Figure 6(a), the storage sizes of the MDC materializations increase with the cardinality. As there are more values in the nominal attribute, the number of the minimal disqualifying conditions for a data point is greater, yielding the increase in the storage size of the MDCs. Since there are more values in a nominal attribute, the R-tree sizes increase. For similar reasons, in Figure 6(b), the pre-processing times of MDC-M, MDC-$M(I)$ and MDC-$O(I)$ increase with cardinality. The pre-processing time of MDC-O remains unchanged. The favorable facet search times of all algorithms (Figure 6(c)) remain nearly the same because the cardinality of nominal attribute does not affect the search significantly. In Figure 6(d), the proportion of skyline points increases with the cardinality of nominal attribute. This is because there is higher chance that a data point is not dominated by others when the cardinality is larger. It is trivial that the average MDC size increases with the cardinality of nominal attribute.

5.1.6 *Effect of the Zipfian Parameter*

We have also conducted experiments with variation of the Zipfian parameter θ. When θ increases, the storage sizes of all algorithms decrease. When θ is large, there are more tuples with the frequent value in the nominal attribute. Thus, there is a higher chance that these tuples quasi-dominate other tuples, yielding disqualifying conditions. However, the template contains a binary order where the most frequent value has a higher preference than other values. Thus, these disqualifying conditions are not minimal and are not stored. The pre-processing time and the favorable facet search time decreases when θ increases. The reasons are similar. The number of skyline points increases when θ increases. This is because there is higher chance that a tuple is not dominated by others when θ is larger. The average MDC size remains nearly the same since θ does not affect the average MDC size significantly.

5.1.7 *Conclusion*

Comparing the on-the-fly method and the materialization method, there is a trade-off between pre-processing time plus storage and the search time. The R-tree indexing helps to greatly reduce the search time for the on-the-fly method on the expenses of more pre-processing and more storage. The R-tree also reduces the pre-processing time of the materialization method in most cases.

5.2 Real Data Sets

To demonstrate the usefulness of our methods, we ran our algorithms on two real data sets.

5.2.1 *Nursery Data Set*

The first set is *Nursery*, which is publicly available from the UCIrvine Machine Learning Repository[2]. *Nursery* was derived from a hierarchical decision model originally developed to rank applications for nursery schools in Ljubljana and Slovenia where the rejected applications frequently needed an objective explanation. Each tuple is an application to the nursery schools. Semantically, if an application is in the skyline, it can be considered a good candidate. Different nursery schools may have different order preferences on the nominal attributes.

In the Nursery data set, there are 12,960 instances and 8 attributes. We transformed 6 attributes (parents, has_nurs, housing, finance, social and health) to numeric attributes because these values are totally ordered. For example, the “social” attribute has three possible values: non-problematic,

[2]`http://kdd.ics.uci.edu/`

slightly problematic and problematic, we matched them to the numbers 0, 1 and 2, respectively. The remaining two attributes are form of the family (e.g. incomplete family and foster family) and the number of children, since there is no trivial order on their values, they would be our nominal attributes. (Note that although the number of children is a numeric attribute, it is not clear whether a family with one child is "better" than a family with two children.) The cardinality of both nominal attributes are equal to 4. We have adopted the default settings for the remaining settings in this experiment. The results in the performance are similar to those for the synthetic data sets. The storage size of the materialized MDC set is 91KB. The pre-processing times of $MDC\text{-}O(I)$, $MDC\text{-}M$ and $MDC\text{-}M(I)$ are 92s, 94s and 60s, respectively. The storage size of the R-tree index is 902KBytes. The favorable facet search time of the materialization methods and $MDC\text{-}O(I)$ is negligibly small.

5.2.2 Automobile Data Set

Our second real data set is *Automobile*, also adopted from the UCIrvine Machine Learning Repository. Our goal is to study the utility of favorable facets. We have chosen four attributes in the experiments, namely "symboling", "normalized-losses", "price" and "make". Attribute "symboling" is the assigned numeric insurance risk rating. The smaller the value is, the safer the vehicle is. Attribute "normalized-losses" is the relative average loss payment per insured vehicle year. If the value is smaller, the loss will be smaller. Hence, the only nominal attribute is "make", representing the car brand names. There were only 205 tuples. The computation times were negligibly small. We are interested to see the meaning and utilization of the favorable facets for different data points. Three car brand names are chosen for our study, namely Honda, Mitsubishi and Toyota. We would find the disqualifying condition for 3 data points that belong to these three brand names, respectively.

1. Our first selected data was a Honda car which was a skyline point in some orders. The disqualifying condition is the following Make order $Toyota \prec Honda$.

2. The second data was a Mitsubishi car and the minimal disqualifying conditions were the following Make order $Honda \prec Mitsubishi$ or $Toyota \prec Mitsubishi$.

3. Finally, we chose a Toyota car and it gave an empty disqualifying condition. The reason was that this car model had the lowest price among all cars in the database. So, it was in the skyline set with respect to any order.

From the above results, a salesperson for the first car should not try to promote the car to a customer that prefers Toyota to Honda, but he may promote it to a customer who prefers Mitsubishi to Honda, since the car has some other advantage that can be attractive. The second car should be promoted to someone who prefers Mitsubishi. The third car can be promoted to any customer.

6. CONCLUSION

In this paper, we identify and tackle an interesting data mining problem, finding favorable facets, which has not been studied before. Given a set of points in a multidimensional space such as a collection of products with attributes like brand, type, color and price, for a specific target point we want to discover with respect to which combinations of orders (e.g., customer preferences) on the nominal attributes the given point is not dominated by any other points. We consider both the effectiveness and the efficiency of the mining. We propose the notion of minimal disqualifying condition (MDC) which is effective in summarizing the favorable facets. We develop efficient algorithms for favorable facet mining for different application scenarios. An extensive performance study using both synthetic and real data sets is reported to verify their effectiveness and efficiency.

As future work, dominance analysis with respect to dynamic data and ordering is an interesting topic.

7. REFERENCES

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

[2] C.-Y. Chan, P.-K. Eng, and K.-L. Tan. Stratified computation of skylines with partially-ordered domains. In *SIGMOD*, 2005.

[3] C.Y. Chan, P.-K. Eng, and K.-L. Tan. Efficient processing of skyline queries with partially-ordered domains. In *ICDE*, 2005.

[4] S. Chaudhuri, N. Dalvi, and R. Kaushik. Robust cardinality and cost estimation for skyline operator. In *ICDE*, 2006.

[5] J. Chomicki. Database querying under changing preferences. In *Annals of Mathematics and Artificial Intelligence*, 2006.

[6] J. Chomicki. Iterative modification and incremental evaluation of preference queries. In *FoIKS*, 2006.

[7] J. L. Bentley et al. Fast linear expected-time algorithms for computing maxima and convex hulls. In *SODA'90*.

[8] J. L. Bentley et al. On the average number of maxima in a set of vectors and applications. In *Journal of ACM, 25(4)*, 1978.

[9] O. Barndorff-Nielsen et al. On the distribution of the number of admissable points in a vector random sample. In *Theory of Probability and its Application, 11(2)*, 1966.

[10] W.-T. Balke et al. Eliciting matters - controlling skyline sizes by incremental integration of user preferences. In *DASFAA'07*.

[11] W.-T. Balke et al. Exploiting indifference for customization of partial order skylines. In *the 10th International Database Engineering and Applications Symposium*, 2006.

[12] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, 1984.

[13] N.L. Johnson, S. Kotz, and A.W. Kemp. Univariate discrete distributions. In *Wiley-Interscience Publication*, 1992.

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

[15] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. In *ACM Transactions on Database Systems, Vol. 30, No. 1*, 2005.

[16] K.-L. Tan, P.K. Eng, and B.C. Ooi. Efficient progressive skyline computation. In *VLDB*, 2001.

[17] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang. Efficient computation of the skyline cube. In *VLDB*, 2005.