On Generating Dominators of Customer Preferences

Jiang Bian
University of Central Florida
Orlando, USA

Weibo Wang
University of North Carolina at Chapel Hill
Chapel Hill, USA

Xiang Zhang
Pennsylvania State University
State College, USA

Wei Wang
University of California, Los Angeles
Los Angeles, USA

Arthur Huang
University of Central Florida
Orlando, USA

Zhishan Guo
University of Central Florida
Orlando, USA

***Abstract*—Manufacturing decisions on how to design new products have tremendous impact on the profitability of the manufacturer. This problem has recently attracted extensive research interests and motivated highly productive activities in developing the microeconomic framework for data mining and finding skyline objects in high-dimensional data. In this paper, we investigate a basic designing problem: designing products that satisfy the preferences of all customers. We formalize this problem as generating dominators (products) that dominate the preference dataset. The problem is naturally related to the microeconomic framework of data mining and the problem of finding skyline objects. The designing problem can be optimized from either the manufacturer's perspective or the customer's perspective. Our framework integrates these two perspectives and achieves optimization in a single effort. We show that this problem is NP-complete and study its computational properties. A deterministic greedy algorithm and a randomized greedy algorithm are developed. Extensive experimental evaluation on both real and simulated datasets demonstrates the effectiveness and efficiency of the proposed algorithms.**

I. Introduction

Manufacturers often face the problem of designing new products based on customer feedback. A simple example is as follows.

Example I.1. Cell phone product design. *Suppose that a phone manufacturer wants to design a set of new products based on a customer survey shown in Table I. Four customers each answers a questionnaire asking his/her preference on three features: camera, smart phone and quick messaging. Three options -1,0,1 are provided for each feature. 1 indicates that the customer will not buy this product* without *this feature; -1 indicates that the customer will not buy a product* with *this feature; 0 indicates that the customer* does not care *whether this feature is present. The manufacturer wants to design a set of new products that satisfy all customer preferences.*

A trivial solution to this problem is to design a product for each customer according to his/her preference. In reality, this may not be feasible due to the large number of customers. Now suppose that the manufacturer has the ability to produce a set of K products. The key problem is whether there exists a set of K products that satisfy all customers. If yes, what features should be included in each product? If no, what is the minimum number of products (K_{min}) needed in order to satisfy all customers?

Intuitively, a product satisfies a preference if (1) it contains all features labeled by 1 in the preference, and (2) it does not contain any feature labeled by -1 in the preference. The product may or may not contain a feature labeled as 0 in the preference. (Including such a feature may be less desirable since it may increase the overall cost of the product. More details will be discussed in Section III.)

TABLE I
AN EXAMPLE PREFERENCE DATASET

	camera	smart phone	quick messaging
p_1	1	0	-1
p_2	1	0	0
p_3	0	1	-1
p_4	0	-1	1

TABLE II
A SET OF PRODUCTS SATISFYING THE EXAMPLE DATASET

	camera	smart phone	quick messaging
d_1	1	0	0
d_2	0	1	0
d_3	0	0	1

Using the example preference data in Table I, when $K = 3$, a set of new products are shown in Table II, with 1 or 0 indicating whether a certain feature is included in a product or not. In the new products, d_1 satisfies p_1 and p_2, d_2 satisfies p_3, and d_3 satisfies p_4.

The product designing problem can be formally defined as follows. Given a preference dataset $P = \{p_1, p_2, \cdots, p_N\}$ ($p_i \in \{1, 0, -1\}^M$, where M is the total number of features), and an integer K ($K \leq N$), find a set of products $D = \{d_1, d_2, \cdots, d_K\}$ ($d_j \in \{1, 0\}^M$) such that every p_i is satisfied by at least one product in D.

Decision making based on customer preference has recently received extensive attention in the database and data mining communities [1]–[3]. Under the microeconomic framework for

*This work is supported by NSF grants CNS 1850851, C-Accel 1937833, and a startup grant from the University of Central Florida.

978-1-7281-0858-2/19/$31.00 ©2019 IEEE

data mining [1], [4]–[8], an enterprise makes a decision that maximizes the utility over all customers. Various classical can be viewed as using such kind of framework, among which the hypercube segmentation problem [1], [4] is a special case of generating new products based on preferences of customers. In the hypercube segmentation problem, the proposed model actually does not consider the hard constraint that a customer will not buy a product with certain features. In real life, however, this is an essential factor that must be taken into consideration when designing new products. For example, some customers may not buy a smart phone product due to the bundled data plan. Disregarding the "may not buy"constraint may prevent designing suitable products satisfying this group of customers.

Another well-studied area is finding skyline objects in a dataset [2], [9]–[12]. An object p is said to dominate another object q if it is at least as good as q in every dimension and may be better than q in some dimensions. The skyline of a dataset consists of all objects not dominated by any other object. Interestingly, skyline objects can be a special solution to our problem. If we define the satisfaction relationship (between a product and a user preference) as the dominant relationship in skyline queries, the set of skyline objects can also be treated as a set of products that satisfy the customers' preferences. The difference is that the skyline objects are always selected from the existing data, while the generated dominators are not (an example is that d_1 is not in P). The dominators are generated through feature selection, where it is different from searching for the skyline objects. When using skyline objects as the dominators, there are many variations of skyline techniques derived to reduce the vary large number of discovered skyline objects (see Section II). However, we have discussed in the experiment section that our proposed dominators method outperforms the skyline technique and its variation top-K skylines, to solve this problem.

In this paper, we target on the problem of generating new products that satisfy all customers' preferences. We give formal definition of the problem and study its computational properties. Our contributions are summarized as follows:

- We formulate the problem as generating *dominators* over preferences. Our formulation naturally incorporates the skyline of the preference dataset as a special case of final products.
- Dominator generation problem can be optimized from two different perspectives. From the customer's perspective, the generated set of products maximize the customer's satisfaction. From the manufacturer's perspective, the product cost is reduced as much as possible. We show that our formulation can integrate these two seemly contradictory perspectives so that the optimization can be achieved in a unified framework.
- We prove that the problem of generating dominators is NP-complete by reduction from the clique cover problem. We investigate its computational properties and leverage them to develop a deterministic greedy algorithm and a randomized greedy algorithm.
- We evaluate the performance of the proposed approaches on both real and simulated datasets.

Note that, in this paper, we study how to generate dominators to satisfy all customers. This is the most basic form of a variety of related problems. Various extensions can be derived from this basic form. For example, an extension is how to generate a set of dominators to satisfy a subset of customers. This is synergetic to the well studied set cover problem, whose most basic form is to cover all elements in a given universe [13]. One of its variation is the partial set cover problem in which only a subset of the universe is covered [14]. The basic form of generating dominators studied in this paper is of both theoretical and practical importance. Our results serve as a starting point and provide foundations for further investigation of its various extensions.

The rest of the paper is organized as follows. The related work is discussed in Section II. The problem definition and its computational properties are studied in Section III. In Section IV, we propose two greedy algorithms to solve the problem efficiently. The experimental results are reported in Section V. We conclude our work in Section VI.

II. RELATED WORK

Under the microeconomic data mining framework, the enterprise has a set of decisions to make in order to maximize the utility [1], [4]–[8]. Each customer makes contribution to the overall utility depending on the decisions made. Various problems have been studied within this framework. For example, the hypercube segmentation problem and the catalog segmentation problem are investigated in [1], [4]. The two problems try to segment the customers to maximize the overall utility from the enterprise's perspective. In [5], a specialized segmentation problem called customer-oriented catalog segmentation problem is studied. It maximizes the utility from the customer's perspective. In this paper, we show that these two perspectives can be integrated under our formulation. The hypercube segmentation problem can also be viewed as a special case of generating new products based on the customers' preferences. However, the hypercube segmentation problem does not consider the situation that a customer may not buy a product with certain feature.

Finding skyline objects in high-dimensional data has been extensively studied recently [2], [9]–[12], [15], [16]. One limitation is that there may be too many skyline objects. To reduce the number of skyline objects, the K-dominant skyline [9], top-K skylines [10] are proposed. The K-dominant skyline relaxes the dominant definition by only requiring the values of data point p be better than or equal to the values of q on K dimensions. The concept of top-K skylines is proposed to select only K skyline objects that maximize a given objective function, such as the total number of dominated objects [10] or the profit [12]. Both the K-dominant skyline and top-K skylines try to select from existing points in the dataset.

In [15], the authors study the problem of generating a set of new data points (not in the existing data) that dominate the existing data points. The goal is different from ours in the

TABLE III
DOMINANT OPERATOR

$d^{(m)}$	$p^{(m)}$		
	-1	0	1
0	$true$	$true$	$false$
1	$false$	$true$	$true$

TABLE IV
DISTANCE MATRIX FOR A PRODUCT AND A PREFERENCE

$d^{(m)}$	$p^{(m)}$		
	-1	0	1
0	0	0	∞
1	∞	1	0

sense that new data points generated in [15] are combinations of existing products (e.g., new combinations of hotels and airlines). Every feature in the existing must be included in the new generated data. In our problem, we consider which features need to be included in the final products.

Preference computing has also been studied in the problem of finding preference levels among different features [3], [17], [18], where a user specifies his/her preference of a feature by assigning a weight. These preferences are used to rank different objects. This is also different from our goal of generating new products.

III. PROBLEM DEFINITION

In this section, we formalize the product designing problem as the generating dominators problem. We show that this problem is NP-complete and study its computational properties.

A. Dominators and Distance Functions

Suppose that there are M features can be included in a product. The preference dataset $P = \{p_1, p_2, \cdots, p_N\}$ ($p_i \in \{1, 0, -1\}^M$) contains the preferences of N customers. Each preference is a $\{-1, 0, 1\}$ vector of M dimensions, with 1/-1 indicating that the customer may not buy this product without/with certain feature, and 0 indicating that the customer does not care if certain feature is present. A product d is a binary vector of M dimensions with 1/0 indicating whether certain feature is included in the product or not. For example, Table I shows a preference dataset containing 4 preferences. Table II shows a set of 3 products. The satisfying relationship between a product and a preference can be formalized using the dominant operator defined as follows.

Let $d =< d^{(1)}, d^{(2)}, \cdots, d^{(M)} >$ be an M dimensional product vector and $p =< p^{(1)}, p^{(2)}, \cdots, p^{(M)} >$ be an M dimensional preference vector. Intuitively, when $p^{(m)} = -1$, a satisfying product cannot include this feature; when $p^{(m)} = 1$, a satisfying product must include this feature; and when $p^{(m)} = 0$, a satisfying product may or may not include this feature. This intuition can be formalized by the *dominant operator* defined on $(d^{(m)}, p^{(m)})$ as shown in Table III. A product d *dominates* a preference p if the dominant operator returns $true$ for all M features. A product is a *dominator* if it dominates at least one preference.

In our running example shown in Tables I and II, d_1 dominates p_1 and p_2. Note that d_1 does not dominate p_3 as the dominant operator returns "false" on the feature smart phone. d_1, d_2 and d_3 are all dominators since each of them dominates at least one preference.

A set of dominators D is referred to as a *dominant set*. D is a *complete dominant set*, if every preference in P is dominated by at least one member in D. The size of D is defined as the number of dominators in D. In the running example, $D = \{d_1, d_2, d_3\}$ is a size 3 complete dominant set. We use Γ_P^K to denote a size K complete dominant set of the preference dataset P.

Based on the dominant operator in Table III, we can define the distance between a product d and a preference p. The distance matrix $dist(d^{(m)}, p^{(m)})$ is shown in Table IV. The distance is ∞ when the dominant operator returns "false", indicating that the preference is not satisfied by the product. The distance is 0 when a desired feature is included. When the product includes an unnecessary feature, i.e., $p^{(m)} = 0$ and $d^{(m)} = 1$, the customer who buys d has to pay for an extra feature that he/she does not care about. This may reduce the customer's satisfaction. We therefore penalize the dominator with an unnecessary feature m by assigning $dist(d^{(m)}, p^{(m)})$ a positive value. The positive value can be interpreted as the cost of having the unnecessary feature. Without loss of generality, we set this cost to be 1[1]. The distance between p and d is

$$Distance(d, p) = \sum_{m=1}^{M} dist(d^{(m)}, p^{(m)}).$$

If $Distance(d, p) = \infty$, then d does not dominate p.

Given a dominant set D, a preference p may be dominated by multiple dominators. In this case, the customer tends to choose the product that is closest to his/her preference. Thus we define the distance between p and D as the shortest distance between p and any dominator in D,

$$Distance(D, p) = min_{d \in D} Distance(p, d).$$

We can further define the distance between a preference dataset P and a dominant set D. It measures the overall satisfaction that the product set D can provide for the preference set P.

$$Distance(D, P) = \sum_{p \in P} Distance(D, p).$$

In our running example, the distance between the preference data $\{p_1, p_2, p_3, p_4\}$ and the dominant set $D = \{d_1, d_2, d_3\}$ is 0, since every preference is dominated by a dominator with distance 0.

B. The Problem of Generating Dominators

The manufacturer is interested in finding a set of products that satisfy the preference dataset P of N customers. In our formulation, this set of products is a complete dominant set. Suppose that the manufacturer wants to design in total $K (K \leq N)$ different products. The general problem is to find a size K complete dominant set Γ_P^K. More specifically, we are interested in the following related problems.

For a fixed K, multiple complete dominant sets may exist. We need to determine which set is the best. The optimal size K complete dominant set can be defined based on two different

[1]Note that our problem formulation and algorithms can also be applied to the case in which the costs are any positive values.

criteria. The first one is to maximize the user satisfaction. The second one is to minimize the overall cost of products.

The customer satisfaction that a complete dominant set Γ_P^K can provide is measured by $Distance(\Gamma_P^K, P)$. Smaller distance indicates better satisfaction. Based on this criterion, we can define the optimal complete dominant set as

$$optimal(\Gamma_P^K) = \underset{\Gamma_P^K}{\operatorname{argmin}}(Distance(\Gamma_P^K, P)). \quad (1)$$

We refer to the problem of finding optimal size K complete dominant set based on criterion (1) as *the most satisfying size K complete dominant set problem*.

The optimality of size K complete dominant sets can also be defined based on the overall cost of the products. We define the cost of a product $d =< d^{(1)}, d^{(2)}, ..., d^{(M)} >$ as

$$Cost(d) = \sum_{i=1}^{M} cost(d^{(i)}),$$

where $cost(d^{(i)})$ is the cost of feature $d^{(i)}$, which can be any positive value. For simplicity, we assume $cost(d^{(i)}) = 1$. Our formulation and algorithms can also be applied to the general case in which $cost(d^{(i)})$ is any positive value. Based on this criterion, we can define the optimal complete dominant set as

$$optimal^{'}(\Gamma_P^K) = \underset{\Gamma_P^K}{\operatorname{argmin}} \sum_{d \in \Gamma_P^K} w_d Cost(d), \quad (2)$$

where w_d is the number of customers who choose d as their favorite product (i.e., d is the product that is closest to their preferences). Criterion (2) minimizes the overall cost of products in Γ_P^K. We refer to the problem of finding optimal size K complete dominant set based on criterion (2) as *the most cost efficient size K complete dominant set problem*.

Criterion (1) is from the customers' perspective, which tries to maximize the customer satisfaction. Criterion (2) is from the manufacturer's perspective, which tries to minimize the cost of making these products. Interestingly, we can show that these seemly contradictory criteria are actually equivalent under our framework.

Theorem III.1. *The problem of finding most satisfying size K complete dominant set and the problem of finding most cost efficient size K complete dominant set are equivalent.*

Proof. For a given a preference $p \in \{1, 0, -1\}^M$, we denote the number of 1's in p as a, the number of -1's in p as b, and the number of 0's in p as c. We have $a + b + c = M$. Let $d \in \{1, 0\}^M$ be the closest dominator of p in a complete dominant set D. According to the satisfying relationship, d must have at least a features. If the number of features in d is larger than a, d has some unnecessary features for p. An unnecessary feature is a feature that p does not care about but d has. Suppose that there are x unnecessary features in d ($0 \leq x \leq c$). The distance between p and d is x, and the total number of features in d is $a + x$. Let $\parallel p \parallel$ represent the number of 1's in p. We have

$$Distance(d, p) + \parallel p \parallel = \parallel d \parallel = Cost(d).$$

Summing over all preferences in the preference dataset P, we have the following equation,

$$Distance(D, P) + \sum_{p \in P} \parallel p \parallel = \sum_{d \in D} w_d Cost(d), \quad (3)$$

where $\sum_{p \in P} \parallel p \parallel$ is the total number of 1's in P, which is a constant for a given P, w_d is the number of preferences to which d is the closest dominator, and $\sum_{d \in D} w_d Cost(d)$ measures the total cost of D.

Equation (3) reveals that there is a constant difference between the overall satisfaction that D can provide and the overall cost of D. As a result, minimizing $Distance(D, P)$ has the same effect as minimizing the total cost. Thus the two problems are equivalent. □

Note that Theorem III.1 also holds when the feature cost $cost(d^{(i)})$ is any positive value. In this case, at the left-hand side of Equation (3), $\|p\|$ represents the overall cost of features that are labeled with 1 in p, and $\sum_{p \in P} \parallel p \parallel$ represents the overall cost of features that are labeled with 1 in P, which is also a constant given P and feature costs.

Theorem III.1 indicates that reducing the cost of products and increasing the overall satisfaction of users can be achieved in a single effort.

C. Complexity of the Problem

In this section, we prove that finding the size K complete dominant set problem is NP-complete by reduction from the clique cover problem [19].

In graph theory, a clique is a subgraph $G^{'} \subset G$ where there is an edge between each pair of vertices in $G^{'}$. The clique cover problem determines whether the vertices of a graph can be partitioned into K cliques. It is an NP-complete problem [19].

To show the reduction, we first introduce the concept of *compatible preferences*. Two preferences p and q are compatible if there does not exist a pair of values $(p^{(m)}, q^{(m)})$ such that both 1 and -1 appear. In our running example, p_1 is compatible with p_2 and p_3. However, p_1 is not compatible with p_4 as the pair of values on the quick messaging feature is $(-1, 1)$. This means that p_1 and p_4 have conflicting opinions on the quick messaging feature.

Based on the definition of compatible preferences, we can build an undirected graph $G = (V, E)$ for a preference dataset P. Each vertex in V is a preference. If preferences p_i and p_j are compatible, there is an edge $< i, j >$ between the two corresponding vertices v_i, v_j. A clique in this graph represents a subset of preferences that are compatible with each other. In our running example, there is a clique containing the vertices corresponding to p_1, p_2, and p_3. Note that the compatible relationship between preferences is not transitive. In our running example, (p_1, p_2) and (p_2, p_4) are both compatible preferences. However, p_1 is not compatible with p_4. It takes $O(N^2)$ time to construct the graph as it requires testing every pair of preferences to compute the edges.

Lemma III.2. *Preferences in the same clique can be dominated by one dominator.*

Proof. We prove this lemma by giving a method that generates such a dominator. For each feature, if its corresponding value is 1 for any preference in the clique, we set its corresponding value in the generated dominator to be 1. Otherwise, we set it to be 0. Note that this can always be done since all preferences are compatible with each other and there are no conflicting opinions. The generated dominator dominates all preferences in the clique. □

In our running example, a dominator $< 1, 1, 0 >$ can be generated to dominate p_1, p_2, and p_3.

Theorem III.3. *Finding the size K complete dominant set problem is NP-complete.*

Proof. Finding the size K complete dominant set problem is in P because a solution to it can be verified in polynomial time. Next, we show that such a solution can be reduced from a solution to the clique cover problem.

Given a preference dataset, we can build an undirected graph $G = (V, E)$ as described above. If there are K cliques covering all vertices, we can generate K products, each dominating all vertices in a clique. All vertices in G are covered by the K cliques. Therefore, all preferences are dominated by the K products. Thus a solution that finds K cliques for the clique cover problem can be used as a subroutine to find the size K complete dominant set. □

Since maximizing the customer's satisfaction is equivalent to minimizing the cost of the products, in the following discussion, we focus on the most satisfying size K complete dominant set problem. We develop a deterministic greedy algorithm and a randomized greedy algorithm to find a solution efficiently.

IV. ALGORITHMS

From now on, we will simply refer to the most satisfying size K complete dominant set as the *optimal size K complete dominant set*. A naive algorithm to this problem is to enumerate all size K dominant sets to find the optimal one. The size of the search space is $\binom{2^M}{K}$. This exhaustive enumeration algorithm is not feasible in practice since the number of features M can be large.

In this section, we give two greedy algorithms to solve the problem efficiently. One is a deterministic algorithm, and the other is a randomized algorithm. The algorithms are based on merging semi-products. In the next, we first introduce the concept of profiles and the merge operation defined on them.

A. Profiles and Merge Operation

Let Dis^K represent the distance between the preference dataset P and an optimal size K complete dominant set Γ_P^K.

Property IV.1. *$Dis^K \leq Dis^{K-1}$ for any $K > 1$.*

Proof. Let Γ_P^{K-1} be an optimal size $K-1$ complete dominant set. We can create a size K complete dominant set Γ_P^K from Γ_P^{K-1} without increasing the overall distance: we keep every dominator in Γ_P^{K-1} and add an extra dominator d'. Note that this can always be done as long as the newly added d' dominates some preferences in P. For any preference $p \in P$, if its distance to d' is less than its distance to its closest dominator in Γ_P^{K-1}, its distance to Γ_P^K will be less than its distance to Γ_P^{K-1}. This makes $Dis^K < Dis^{K-1}$. Otherwise, the overall distance remains the same. This completes the proof. □

TABLE V
PROFILES GENERATED FROM THE PREFERENCE DATASET

	camera	smart phone	quick messaging
f_1	1	$*$	0
f_2	1	$*$	$*$
f_3	$*$	1	0
f_4	$*$	0	1

It is trivial to find the optimal size N complete dominant set, where N is the number of preferences in P. For each preference $p_n \in P$ ($1 \leq n \leq N$), we can design a dominator d_n that has distance 0 to p_n as follows. For every feature, if it is 1 in the preference, then the product has that feature. If it is -1 or 0 in the preference, then the product does not have that feature. The overall distance between P and $D = \{d_1, d_2, ..., d_N\}$ is 0. D is the optimal size N complete dominant set.

To find the optimal size K complete dominant set, intuitively, we can start from the size N complete dominant set and merge dominators to reduce the size of the dominant set. Next, we introduce the concept of profiles and define the merge operation on them.

A *profile* is a $\{1, 0, *\}^M$ vector. It is a semi-product with the symbol $*$ representing that the product may or may not have certain feature. In our running example, a profile $< *, 0, 1 >$ is a semi-product, which supports quick messaging, but does not include the smart phone feature. Whether it has a camera is yet to be decided.

For each preference, we can generate a profile as follows. For every feature, if it is 1 in the preference, then the profile has that feature. If it is -1 in the preference, then the product does not have that feature. If it is 0 in the preference, then the product may or may not have that feature and we use a $*$ to denote this case. Following our running example in Table I, the profiles of the preferences are shown in Table V. Changing $*$ in a profile to either 0 or 1 will generate a product that dominates the preference from which the profile is generated. For example, from f_1 in Table V, we can generate two products, $< 1, 0, 0 >$ and $< 1, 1, 0 >$. Both of them dominate p_1 in Table I.

The basic idea of our algorithm is to first generate a profile for each preference and then merge them until there are K profiles left. These profiles can be used to generate the final K products.

The *merge operation* on two profiles can be conducted as follows. Two profiles f_i and f_j can be merged if they only differ in the positions where at least one of the profiles has a

$*$. In Table V, f_1 and f_2 can be merged, but f_3 and f_4 cannot. When merging two profiles, we change $*$ in one profile to the corresponding value in the other profile. In the above example, f_1 and f_2 can be merged to generate profile $< 1, *, 0 >$, which is the same as f_1.

The profiles can be used to find the $K(K \leq N)$ complete dominant set. The overall process is as follows.

- First, we generate a profile for every preference in P. There are N profiles.
- Then, we reduce the number of profiles by merging them until only K profiles left.
- Finally, K products are generated by replacing all $*$'s in the profiles by 0's.

Compared to the brute force algorithm, the size of the search space is dramatically reduced, from $\binom{2^M}{K}$ to $\binom{N}{K}$.

B. A Greedy Approach

The profiles can be merged in different orders. A greedy approach is to merge the two profiles that would introduce the minimal distance increase. More formally, we can define the distance introduced by merging two profiles as follows.

We first define the distance matrix of a profile and a preference as in Table VI. The distance between a profile f and a preference p is

$$Distance(f, p) = \sum_{m=1}^{M} Distance(f^{(m)}, p^{(m)}).$$

If the distance between a profile and a preference is ∞, the profile cannot dominate the preference. Note that this distance matrix is a natural extension of the distance matrix of a product and a preference in Table IV. The only difference is that we need to define the distances between * and $\{0, 1, -1\}$ when we consider profiles.

For a set of profiles F, the distance between a preference p and F is the distance between p and its closest profile in F

$$Distance(F, p) = min_{f \in F} Distance(f, p).$$

The distance between F and the preference dataset P is

$$Distance(F, P) = \sum_{p \in P} Distance(F, p).$$

For any two profiles f_i and f_j in F, we use f_{merged} to denote the new profile generated by merging f_i and f_j, and F' to denote the new profile set,

$$F' = F \backslash \{f_i, f_j\} \cup \{f_{merged}\}.$$

The distance introduced by merging two profiles is thus

$$\Delta Dis_F(f_i, f_j) = Distance(F', P) - Distance(F, P).$$

Note that, similar to Property IV.1, merging two profiles does not decrease the distance between the profile set and the preference set. Thus we have $\Delta Dis_F(f_i, f_j) \geq 0$.

$Distance(F, P)$ is closely related to $Distance(D, P)$: if the final product set D is obtained by replacing every $*$ in F by 0, we have $Distance(D, P) = Distance(F, P)$.

TABLE VI
DISTANCE MATRIX FOR A PROFILE AND A PREFERENCE

$f^{(m)}$	$p^{(m)}$		
	-1	0	1
0	0	0	∞
1	∞	1	0
$*$	0	0	1

TABLE VII
DISTANCE MATRIX FOR TWO PROFILES

$f_i^{(m)}$	$f_j^{(m)}$		
	$*$	0	1
0	0	0	∞
1	1	∞	0
$*$	0	0	1

Thus, $Distance(D, P)$ is minimized when $Distance(F, P)$ is minimized. Thus a straightforward greedy approach is to minimize $\Delta Dis_F(f_i, f_j)$ at each step. That is, among all pairs of profiles, we always merge the pair that introduces the minimal increase in distance.

The complexity to compute $\Delta Dis_F(f_i, f_j)$ is $O(NM|F|)$, where N is the number of products, M is the number of features, and $|\cdot|$ is the size of the set. Note that we need to compute $\Delta Dis_F(f_i, f_j)$ for all profile pairs to find the best one. In the next, we introduce a more efficient greedy approach for determining which profile pair to merge.

C. A More Efficient Greedy Approach

Table VII shows the distance matrix for two profiles. If two profiles are not compatible at a feature, their distance is ∞. To determine which two profiles to merge more efficiently, we introduce the following criterion which directly examines the two profiles

$$Dis'_F(f_i, f_j) = \sum_{m=1}^{M} (n_i * I(f_i, m) + n_j * I(f_j, m)) Distance(f_i^{(m)}, f_j^{(m)}),$$

where n_i and n_j are the numbers of preferences that f_i and f_j dominate respectively, and I is an indicator function

$$I(f_i, m) = \begin{cases} 1 & f_i^{(m)} = * \\ 0 & \text{otherwise} \end{cases}$$

Property IV.2. *$Dis'_F(f_i, f_j)$ is an upper bound of $Dis_F(f_i, f_j)$.*

Proof. Let $Dom(f_i)$ be the preferences dominated by f_i, and $Dom(f_j)$ be the preferences dominated by f_j. We have $|Dom(f_i)| = n_i$ and $|Dom(f_j)| = n_j$. Let f_{merged} be the new profile generated by merging f_i and f_j. It is obvious that the merge may only increase the distance between the preferences and the profiles when $f_i^{(m)} = *$ and $f_j^{(m)} = 1$, or $f_i^{(m)} = 1$ and $f_j^{(m)} = *$.

Now assume that f_{merged} is the closest profile to all preferences in $Dom(f_i)$ and $Dom(f_j)$. Consider the case where $f_i^{(m)} = *$ and $f_j^{(m)} = 1$. $f_{merged}^{(m)}$ is 1 after merging. On this feature, the overall distance introduced by merging is thus $[n_i * I(f_i, m) * Distance(f_i^{(m)}, f_j^{(m)})]$. Similarly, for the case where $f_i^{(m)} = 1$ and $f_j^{(m)} = *$, the distance introduced by merging is $[n_j * I(f_j, m) * Distance(f_i^{(m)}, f_j^{(m)})]$. Summing over M features, $Dis'_F(f_i, f_j)$ represents the overall distance

Algorithm 1: A Deterministic Greedy Algorithm for Generating an Optimal Complete Dominant Set

Input: P: preference dataset, K: the size of the complete dominant set

Output: $successful$: a flag indicating whether a size K complete dominant set is found, D: a size K optimal complete dominant set

Generate profile set $F = \{f_1, ..., f_n\}$ for P ;
$successful$:=True ;
while $\parallel F \parallel > K$ *and* $successful$ =*True* **do**
  Find the profiles f_i, f_j with smallest $\Delta Dis'_F(f_i, f_j)$;
  if $\Delta Dis'_F(f_i, f_j) = \infty$ **then**
    $successful$:=False ;
  else
    Remove f_i, f_j from F;
    Merge f_i and f_j as f_{ij} ;
    Put f_{ij} into S ;
Create a product for each profile in F and put all created products into D ;
Return $successful$ and D.

Algorithm 2: A Randomized Greedy Algorithm for Generating an Optimal Complete Dominant Set

Input: P: preference dataset, K: the size of the complete dominant set

Output: $successful$ indicating if a size K complete dominant set is found, D the size K optimal complete dominant set

Generate profile set $F = \{f_1, ..., f_n\}$ for P ;
$Candidate := F^{'}$ ($F^{'}$ contains K randomly selected profiles from F) ;
$Remaining := F \backslash F^{'}$;
$successful$:=True ;
while $Remaining! = \emptyset$ *and* $successful$ =*True* **do**
  Find the profiles f_i, f_j with smallest $\Delta Dis'_F(f_i, f_j)$ (where $f_i \in Candidate$ and $f_j \in Remaining$);
  if $\Delta Dis'_F(f_i, f_j) = \infty$ **then**
    $successful$:=False ;
  else
    Remove f_i from $Candidate$;
    Remove f_j from $Remaining$;
    Merge f_i and f_j as f_{ij} ;
    Put f_{ij} into $Candidate$;
Create a product for each profile in $Candidate$ and put all created products into D ;
Return $successful$ and D.

introduced by merging f_i and f_j. In this case, we have $Dis'_F(f_i, f_j) = Dis_F(f_i, f_j)$.

However, f_{merged} may not always be the closest profile to every preference in $Dom(f_i)$ and $Dom(f_j)$. In this case, we have $Dis'_F(f_i, f_j) > Dis_F(f_i, f_j)$. □

The complexity to compute $\Delta Dis'_F(f_i, f_j)$ is $O(M)$, which is much less than the complexity to compute $\Delta Dis_F(f_i, f_j)$. Our greedy algorithms use $\Delta Dis'_F(f_i, f_j)$ to determine which profile pair to merge.

D. Deterministic and Randomized Greedy Algorithms

Algorithm gives the pseudo code of a deterministic greedy algorithm. The algorithm first builds a profile for each preference. It then repeatedly merges profile pairs based on $\Delta Dis'_F(f_i, f_j)$ until the total number of profiles is reduced to K. Final products are then created from the K profiles. If no profile pair can be merged before the size is reduced to K, the greedy algorithm fails to find a size K complete dominant set for P. The time complexity of Algorithm 1 is $O(MN^2(N-K))$.

One problem of this approach is that the greedy algorithm may converge at a local optimum. We thus apply a randomized method to alleviate the problem. The pseudo code is shown in Algorithm 2. This algorithm first creates a candidate set with K randomly selected profiles. It then selects from the remaining profiles to merge into the candidate set. The algorithm finishes when all profiles are merged into the candidate set. Final products are then generated for the profiles in the candidate set. If there are profiles cannot be merged into the candidate set, the randomized greedy algorithm fails to find a size K complete dominant set. This algorithm can be run multiple times with different initial candidate sets to increase the probability of finding the global optimum. The time complexity of Algorithm 2 is $O(MK(N-K)^2)$, which is less than that of the deterministic version since K is usually smaller than N.

Using the running example, suppose that we are interested in finding size 2 optimal complete dominant set. The deterministic greedy algorithm first generates the profiles as shown in Table V. Two profiles f_1 and f_2 will be merged first to generate profile $f_{12} =< 1, *, 0 >$. The next two profiles to be merged are f_{12} and f_3. Merging them will result $f_{123} =< 1, 1, 0 >$. Now only f_{123} and f_4 are remained and they cannot be further merged. The algorithm replaces the $*$ in f_4 and a size 2 complete dominant set is found as shown in Table VIII.

The randomized greedy algorithm will first generate a size 2 candidate profile set. Suppose that in one trial the candidate set contains f_1 and f_4. f_2 and f_3 are not in the candidate set. f_2 is the best profile to be merged with the candidate profile set (with f_1). Profile f_{12} will be generated and kept in the candidate set after merging. Next, f_3 will be merged into the candidate set with f_{12}. Now all profiles are merged into the candidate set. The candidate set has two profiles f_{123} and f_4. The algorithm will generate the optimal size 2 complete dominant set based on this candidate set. However, if in a trial, f_1 and f_3 are selected to be included in the candidate set, f_4 cannot be merged into the candidate set. The randomized greedy algorithm will fail to identify the size 2 complete dominant set.

TABLE VIII
AN OPTIMAL SIZE 2 SOLUTION OF THE EXAMPLE DATA SET

	camera	smart phone	quick messaging
d_1'	1	1	0
d_2'	0	0	1

Note that, for illustration purpose, we have assumed that the cost of an unnecessary feature is 1. Our algorithms can also be applied when the cost of the feature is any positive quantitative value. This can be easily done by replacing the 1's in the distance matrices by the corresponding feature cost. All results discussed in the paper still hold for the quantitative feature cost setting.

V. EXPERIMENTS

In this section, we present extensive experimental results on evaluating the effectiveness and efficiency of the developed algorithms. All algorithms are implemented in C++. The experiments are performed on a desktop computer with an Intel i7 2.67GHz CPU and 8GB RAM running Ubuntu 64-bits operating system.

Datasets: We use both real and simulated data to evaluate the performance of the proposed algorithms. The real data are obtained from the online survey of cell phone products collected through Amazon Market Turk[2]. The survey has two questions. Question 1: Which of the following features do you like to have in your cell phone? (A list of 30 Boolean attributes, such as WiFi, Blutooth, and Camera, are shown as check boxes. Users can select features they like.) Question 2: Which of these following features you do NOT like to have in your cell phone? (The same 30 Boolean attributes were shown as check boxes). A user cannot select the same feature for both questions. The feature selected for Question 1 has value 1, the feature selected for Question 2 has value -1, and the feature not selected has value 0 (don't care). There are 237 different answers (preferences) received. In total, customers selected 869 features as must have and 319 features as must not have. On average, customers selected 3-6 features as must have and 1-2 features as must not have.

Three algorithms are evaluated in our experiments: the brute force exhaustive search algorithm, the deterministic greedy algorithm, and the randomized greedy algorithm.

A. Effectiveness Evaluation on Simulated Data

We first evaluate whether the developed greedy algorithms can find the optimal complete dominant set. Using the original real dataset, the exhaustive search algorithm cannot finish after running a few days. It leaves the ground truth unknown. We thus simulate five smaller datasets as follows. For each simulated dataset, we randomly choose 5 features and 10 preferences from the original real dataset. Table IX gives the statistics of the simulated data. On average, each simulated data has 17 must have features and 5 must not have features.

Tables X-XI show the results of the three algorithms on the simulated data sets. For each algorithm, we run it on each simulated dataset 10 times, once for each possible value of K varying from 1 to 10, for a total of 50 runs. Each run produces a complete dominant set unless the algorithm fails to find one. Table X(a) gives the number of complete dominant sets each algorithm finds. The exhaustive search algorithm finds the globally optimal solutions which serves as the ground truth. Among the 50 runs, the exhaustive search algorithm finds 40 complete dominant sets. The deterministic greedy algorithm finds 38 complete dominant sets for these 40 runs and the randomized greedy algorithm finds complete dominant sets for all 40 runs. 32 out of the 38 solutions identified by the deterministic greedy algorithm are optimal, yielding a success ratio of 84%. 36 out of the 40 solutions identified by the randomized greedy algorithm are optimal, yielding a success ratio of 90%. The randomized strategy for generating candidate profile set has higher chance of finding solutions and often produces better solutions. We study the number of optimal solutions found at each setting of K (see Table X(b)). When K is 1 or 2, there does not exist any complete dominant set. When K ranges from 3 to 6, the exhaustive searching algorithms can always find complete dominant sets but the greedy algorithms may not always find them. The randomized greedy algorithm finds more optimal solutions than the deterministic greedy algorithm. When K is larger than 6, all three algorithms can find the optimal solutions . We further examine the generated dominators and calculate the average distances between the solutions and the preferences (see Table XI) for all settings of K. The average distance measures the customer satisfaction that the generated complete dominant set can provide. The smaller, the better. The distances for all algorithms decrease with increasing K. Larger K allows the algorithms to find solutions with better satisfaction. The exhaustive search algorithm always find solutions with smallest distances. The deterministic greedy algorithm always performs the worst among the three. The difference between the three algorithms shrinks when K increases. When $K > 6$, there is no difference.

TABLE IX
SIMULATED DATA DESCRIPTIONS

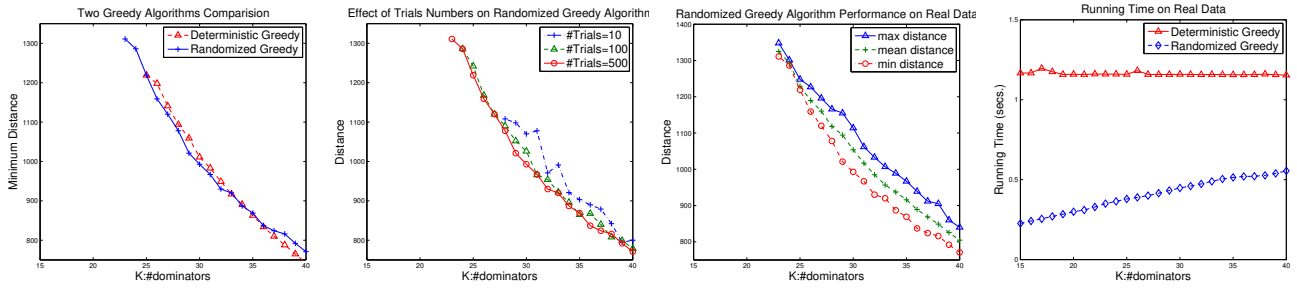
♯ simulated data set	5
♯ features per data set	5
♯ preferences per data set	10
♯ "must have" features per data set	17
♯ "must not" features per data set	5

B. Effectiveness Evaluation on Real Data

We further compare the complete dominant sets generated by the deterministic and randomized greedy algorithms using the original survey dataset. K ranges from 15 to 40. Results of the randomized greedy algorithm are based on 500 trials. Note that the exhaustive search algorithm is too time consuming to run on this dataset.

We compare the distances between the entire set of preferences and the size K complete dominant sets found by the randomized greedy algorithm and by the deterministic

[2]http://en.wikipedia.org/wiki/Amazon_Mechanical_Turk



(a) (b) (c) (d)

Fig. 1. Effectiveness (a)(b)(c) and Efficiency (d) evaluation on real data

TABLE X
NUMBER OF SOLUTIONS ON SIMULATED DATA

(a) Number of Solutions

Exhaustive Search	40
Randomized Greedy (total found,optimal)	(40,36)
Deterministic Greedy (total found,optimal)	(38,32)

(b) Number of Optimal Solutions

	K : the size of the complete dominant set									
	1	2	3	4	5	6	7	8	9	10
Exhaustive Search	0	0	5	5	5	5	5	5	5	5
Deterministic Greedy	0	0	2	2	4	4	5	5	5	5
Randomized Greedy	0	0	3	4	5	5	5	5	5	5

TABLE XI
AVERAGE DISTANCES OF SOLUTIONS ON SIMULATED DATA

	K : the size of the complete dominant set									
	1	2	3	4	5	6	7	8	9	10
Exhaustive Search	NA	NA	11.6	6.6	3.8	2.0	1.2	0.4	0	0
Deterministic Greedy	NA	NA	12.3	7.6	4.2	2.4	1.2	0.4	0	0
Randomized Greedy	NA	NA	12.2	7.0	3.8	2.0	1.2	0.4	0	0

greedy algorithm respectively for various settings of K (see Figure 1(a)). When K is relative small (≤ 35), the randomized algorithm performs better than the deterministic algorithm. The randomized algorithm either finds a solution when the deterministic algorithm does not ($K = 23$ or 24), or finds better solutions (smaller distance, $K \in [25, 32]$). However, the performance difference between these two approaches shrinks when K increases ($K \in [33, 36]$). For larger K ($K \in [37, 40]$), the deterministic algorithm outperforms the randomized algorithm.

We vary the number of trials (runs) from 10, 100, to 500 to study its effect on the randomized greedy algorithm (see Figure 1(b)). The minimal distance for each specific trial setting is reported. The 500 trial setting manages to find complete dominant sets when K is 23 or larger. The 10 trial setting starts to find solutions when K is 28 or larger. Even though these greedy algorithms do not always find the globally optimal solution, we can still observe that the distance exhibits a clear descending trend as K increases, as stated in Property IV.1. The local jerkiness in the performance curve of 10 trial setting is due to the randomness in selecting initial candidates. Such jerkiness can be smoothed out with more trials, as demonstrated by the smoother curves of the 100 trials and 500 trials.

To further evaluate the randomized greedy algorithm, we compare the minimum distance, the maximum distance, and the average distance of the complete dominant sets found in the 500 trial setting (see Figure 1(c)). The largest difference between the maximum distance and the minimum distance is 134 ($K = 29$). Note that there are 237 preferences with 30 attributes in the original dataset. On average, the difference is less than 1 feature per preference. This small variation demonstrates the robustness of the randomized greedy algorithm.

C. Efficiency Evaluation on Both Datasets

In this subsection, we evaluate the efficiency of the three algorithms. The reported running time of the randomized greedy algorithm is averaged on 500 trials.

Table XII gives the average running time of the exhaustive algorithm, the deterministic greedy algorithm, and the randomized greedy algorithm on the five simulated data. In this experiment, the time is measured in seconds and rounded to two digits after the decimal point. The running time of the exhaustive search algorithm increases exponentially with respect to increasing K. The two greedy algorithms are much more efficient. Their running time is always less than 0.01 second on the simulated data set.

Figure 1(d) shows the running time of both greedy algorithms on the real dataset. On average, it takes 1.2 second for the deterministic algorithm to finish, and 0.3-0.6 second for the randomized algorithm to finish for each trial. The randomized greedy algorithm is 1 to 3 times faster than the deterministic greedy algorithm. This is due to the lower time complexity of the randomized greedy algorithm when $K < N$.

D. Complete Dominant Set versus Skyline

Skyline query has been extensively studied recently. In this subsection, we examine the effectiveness of using skyline points as an alternative method to satisfy the customers' preferences. We follow the conventional dominant relationship definition used in the skyline query: 1 and -1 can be viewed as stronger opinions than 0, thus we define a binary relation $0 \preceq 1$ and $0 \preceq -1$ on a given feature. For preferences p and q, p is dominated by q if $q^{(m)} \preceq p^{(m)}$ is not true for any $m \in [1, M]$, but $p^{(m)} \preceq q^{(m)}$ may hold for some $m \in [1, M]$. Both the skyline model and the top-K skyline model are examined. The K-dominant skyline is not applicable to our problem because we require that the satisfaction relationship is held for every feature.

TABLE XII
RUNNING TIME EVALUATION ON SIMULATED DATA (IN SECONDS)

	K : the size of the complete dominant set									
	1	2	3	4	5	6	7	8	9	10
Exhaustive Search	0.01	0.01	0.01	0.02	0.26	1.33	5.26	17.77	64.53	153.72
Deterministic Greedy	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Randomized Greedy	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

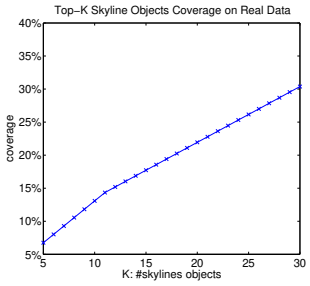


Fig. 2. Top-K skyline model evaluation on real data

The set of the skyline points can be treated as the size S complete dominant set, where $S \leq N$ is the number of skyline points. We run skyline query on the real data. Among the 237 preferences, 210 are skyline points. It is much larger than the complete dominant set, which contains only 23 products.

The top-K skyline model reports a smaller number of skyline points. We rank the skyline points by the number of preferences that they dominate. Figure 2 shows the number of preferences in the real data dominated by the top-K skyline points for various K. Note that our method generates 23 products to dominate all preferences for the real data set. From Figure 2, we can see that the top-23 skyline points dominate less than 25% of the preferences. This demonstrates the advantage of the proposed dominant set model in designing new products.

VI. CONCLUSION AND FUTURE WORK

Designing new products based on customer feedback is a crucial problem faced by the manufacturers. In this paper, we investigate the problem of designing new products to satisfy the user preferences. The problem is formalized as generating complete dominant set to maximize overall user satisfaction and reduce overall product costs. Our problem formulation incorporates these two criteria so that optimization can be achieved in a single effort. We prove that the problem is NP-complete by reduction from the clique cover problem. Two greedy algorithms are developed to obtain solutions efficiently. The effectiveness and efficiency of the proposed methods are evaluated by extensive experiments on both real and simulated datasets.

Various extensions of the basic model can be explored. One direction is to study the partial dominant version of the problem. That is to find a set of products that cover a major portion of the customers. This problem is inherently more complex because, in addition to the optimization criteria used in this paper, one also needs to determine which subset of the customers should be satisfied. Another direction is to design approximation algorithms whose performance has theoretical guarantees. We plan to explore these two directions in our future work.

REFERENCES

[1] J. Kleinberg, C. Papadimitriou, and P. Raghavan, "A microeconomic view of data mining," 1998.

[2] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *International Conference on Data Engineering*, 2001, pp. 421–430.

[3] W. Kiesling, "Foundations of preferences in database systems," in *Proceedings of the 28th international conference on Very Large Data Bases*, ser. VLDB '02. VLDB Endowment, 2002, pp. 311–322.

[4] J. M. Kleinberg, C. H. Papadimitriou, and P. Raghavan, "Segmentation problems," *Journal of The ACM*, vol. 51, pp. 263–280, 2004.

[5] M. Ester, R. Ge, W. Jin, and Z. Hu, "A microeconomic data mining problem: customer-oriented catalog segmentation," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '04. New York, NY, USA: ACM, 2004, pp. 557–562.

[6] Z. Zhang, L. V. S. Lakshmanan, and A. K. H. Tung, "On domination game analysis for microeconomic data mining," *ACM Transactions on Knowledge Discovery From Data*, vol. 2, pp. 1–27, 2009.

[7] L. Zhu, C. Li, A. Tung, and S. Wang, "Microeconomic analysis using dominant relationship analysis," *Knowledge and Information Systems*, pp. 1–33, 2011.

[8] M. Miah, "Most popular package design," in *4th Conference on Information Systems Applied Research*, 2011.

[9] C.-Y. Chan, H. V. Jagadish, K. lee Tan, A. K. H. Tung, and Z. Zhang, "Finding k-dominant skylines in high dimensional space," in *In SIGMOD*, 2006, pp. 503–514.

[10] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, "Selecting stars: The k most representative skyline operator," in *International Conference on Data Engineering*, 2007, pp. 86–95.

[11] N. Sarkas, G. Das, N. Koudas, and A. K. H. Tung, "Categorical skylines for streaming data," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 239–250.

[12] Q. Wan, R. C.-W. Wong, and Y. Peng, "Finding top-k profitable products," in *ICDE*, 2011, pp. 1055–1066.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2001.

[14] R. Gandhi, S. Khuller, and A. Srinivasan, "Approximation algorithms for partial covering problems," in *International Colloquium on Automata, Languages and Programming*, 2004.

[15] Q. Wan, R. C.-W. Wong, I. F. Ilyas, M. T. Özsu, and Y. Peng, "Creating competitive products," *Proc. VLDB Endow.*, vol. 2, pp. 898–909, August 2009.

[16] Z. Zhang, H. Lu, B. C. Ooi, and A. K. H. Tung, "Understanding the meaning of a shifted sky: a general framework on extending skyline query," *The Vldb Journal*, vol. 19, pp. 181–201, 2010.

[17] D. Mindolin and J. Chomicki, "Discovering relative importance of skyline attributes," *Proc. VLDB Endow.*, vol. 2, pp. 610–621, August 2009.

[18] F. Zhao, G. Das, K.-L. Tan, and A. K. H. Tung, "Call to order: a hierarchical browsing approach to eliciting users' preference," in *International Conference on Management of Data*, 2010, pp. 27–38.

[19] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.