Finding Competitive Price

Yu Peng, Raymond Chi-Wing Wong
The Hong Kong University of Science and Technology
{gracepy,raywong}@cse.ust.hk

ABSTRACT

Dominance analysis is important in many multi-criteria decision making applications. Most previous works assume that the price of a service is given and study how to select "best" services according to multiple given attributes including attribute Price. In this paper, we propose an interesting data mining problem, *finding competitive price*, which has not been studied before. Given a set of existing services, for a new service, we want to find a price of the new service such that the new service is not worse than any existing services. The price found refers to a competitive price. We propose a spatial approach which makes use of some spatial properties and thus runs efficiently. Finally, we conducted experiments to show the efficiency of our proposed method.

Categories and Subject Descriptors

H.2.4 [**Database Manager**]

General Terms

Algorithms, Experimentation

Keywords

skyline, spatial database.

1. INTRODUCTION

Dominance analysis is important in many multi-criteria decision making applications. Recently, dominance analysis [21, 13, 24, 12, 25, 17, 15] has received a lot of interest from both research and applications.

Example 1 (Skyline). Consider four hotels, namely $h_1, h_2, ..., h_4$, which are near to a beach a_1 in Sydney as shown in Figure 1(a) where the price of each hotel is shown in Figure 1(b).

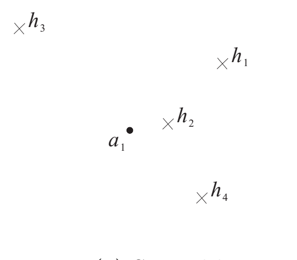
Consider that a customer wants to look for a hotel in Sydney using two factors/criteria: distance-to-beach and price. We transform the spatial layout in Figure 1(a) and the price of each hotel in Figure 1(b) into a new table T with two attributes, namely distance-to-beach and price, as shown in Table 1. This table is called a *decision-making table*. For example, consider hotel h_1. In Figure 1(a), we find the distance between h_1 and a_1, denoted by $d(h_1, a_1)$, equal to 3.0 km. Besides, in Figure 1(b), the price of h_1 is \$100. Then, we construct a tuple for h_1 in table T with (distance-to-beach, price) equal to (3.0, 100).

According to T, we want to determine the best possible choices for the customer. For two hotels h and l, if h is better than l in one factor, and is not worse than l in the other factor, then h is said to *dominate* l. We know that shorter distance to beach and lower price are more preferable. Thus, h_1 dominates h_3 because, compared with hotel h_3, hotel h_1 is closer to beach and has a lower price. Hotel h_2 does not dominate h_3 because h_3 has a lower price than h_2. Similarly, hotel h_3 does not dominate h_2 because h_2 has shorter distance to beach than hotel h_3. A hotel that is not dominated by any other hotel is said to be in the *skyline*. The hotels in the skyline are the best possible tradeoffs among the two factors in question. From this table, h_1, h_2 and h_4 are in the skyline. □

Example 2 (Application). Consider that a travel agency wants to open a new hotel h_f at location indicated in Figure 2. The travel agency has to find a suitable price for h_f called a *competitive price* of h_f so that h_f is competitive in the existing market (including hotels $h_1, h_2, ..., h_4$). From Figure 2, we find that $d(h_f, a_1) = 2.0$. If we set the price of h_f to \$300, according to T, h_f will be dominated by h_2. We say that \$300 is not a competitive price of h_f. However, if we set the price of h_f to \$230, h_f will not be dominated by any hotels in the existing market. We say that \$230 is a competitive price of h_f. □

From the above example, we observe that h_f may or may not be in the skyline with different prices. In this paper, we are studying to find a competitive price of h_f such that h_f is competitive in the existing market. This problem is called *finding simple competitive price*.

Finding a competitive price of h_f means that, after we set the price of h_f, h_f is *one* of the best choices for the customer to choose (because there may be more than one hotel in the skyline). In order to make sure that h_f will be chosen by a customer in the market with a higher probability, we would like to set the price of h_f such that not only h_f is in the skyline but also h_f dominates at least K existing hotels where K is an input parameter. This problem is called



Hotel	Price($)
h_1	100
h_2	250
h_3	200
h_4	220

(a) Spatial layout (b) Price

Figure 1: A running example

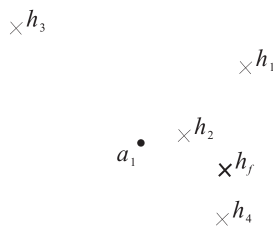


Figure 2: Hotels in the map with a new hotel h_f

finding K-dominating competitive price. The above problem makes sense since it is assumed that each hotel in the existing market must be currently chosen by some customers and thus still exists in the market. If this assumption does not hold, it is very likely that the hotels do not exist in the market because no customers choose these hotels. Thus, if h_f dominates these existing hotels, the customers who originally choose these hotels will choose h_f finally.

Example 3 (K-dominating Competitive Price). If we set the price of h_f to \$230, according to T, h_f does not dominate any hotels. However, if we set it to \$210, h_f dominates one hotel, namely h_4. \$210 is a price for problem finding 1-dominating competitive price but \$230 is not. □

Note that our two problems are not limited to one attraction. Instead, we consider multiple attractions. In addition to beach, Opera House and Sydney Aquarium are two other possible attractions in Sydney. In this paper, we will describe later how we consider multiple attractions.

Setting a competitive price is common in daily life applications. One example is setting a selling price (or a rental rate) of an apartment where attractions can be railway stations and shopping malls. Another example is setting a parking fee of a car park where attractions can be shopping malls and museums.

The following shows our contributions. Firstly, to the best of our knowledge, we are the first to study problems finding simple competitive price and finding K-dominating competitive price. In the problems, one of the objectives is to find a competitive price of a hotel h_f such that h_f is in the skyline. Most existing works in the literature study how to find all hotels in the skyline if the price of each hotel is *given*. Secondly, we propose an effective spatial approach by using some spatial properties. Thirdly, we conducted experiments to show the efficiency of our proposed approach and illustrate with a real case study.

The rest of the paper is organized as follows. Section 2 formulates our proposed problems, namely finding simple competitive price and finding K-dominating competitive price. Section 3 proposes a spatial approach. Section 4 discusses some extensions of our problem. Section 5 evaluates the proposed technique through extensive experiments with both real and synthetic datasets and illustrates the process with a real case study. Section 6 gives the related work. Section 7 concludes the paper with directions for future work.

Hotel	Distance-to-beach(km)	Price($)
h_1	3.0	100
h_2	1.0	250
h_3	4.0	200
h_4	2.5	220

Table 1: A decision-making table T

2. PROBLEM DEFINITION

We have a set H of m objects, namely $h_1, h_2, ..., h_m$, in the Euclidean space, each of which represents a *service-site* (e.g., a hotel in Figure 2). We also have another set A of n objects, namely $a_1, a_2, ..., a_n$, in the same space, each corresponding to an attraction-site (e.g., a beach). For each service-site $h \in H$, we use $h.p$ to denote its *price*. For each $h \in H$ and $a \in A$, the distance between h and a is denoted by $d(h, a)$.

We consider a general situation where each pair of objects in each Cartesian product $H \times \{a_j\}$ has a distinct distance for each $j \in [1, n]$. That is, for each attraction-site a_j, any 2 service-sites h and h' have distinct distances to a_j (i.e., $d(h, a_j) \neq d(h', a_j)$). This assumption allows us to avoid several complicated and uninteresting "boundary cases". When the assumption is not satisfied, an infinitesimal perturbation to the positions of some service-sites or attraction-sites can always be applied, to break the tie of the distances of two object pairs. Due to the tininess of perturbation, results obtained from the perturbed datasets should be as useful as those from the original datasets.

In order to analyze which service-site h in H is better than other service-sites in H, we define a table called a *decision-making table* T with $n+1$ attributes, namely $X_1, X_2, ..., X_{n+1}$, as follows. For each object $h \in H$, we construct a tuple in form of $(x_1, x_2, ..., x_{n+1})$ where x_j is equal to $d(h, a_j)$ for each $j \in [1, n]$ and x_{n+1} is equal to $h.p$. We denote each value x_j by $h.X_j$ for $j \in [1, n+1]$. Table 1 shows an example of the decision-making table T. In our running example, since there are 4 hotels and one attraction, $m = 4$ and $n = 1$. Thus, there are two attributes in T where X_1 ="Distance-to-beach" and X_2 ="Price". In T, there are 4 correspondence tuples. Let $\mathcal{X} = \{X_1, X_2, ..., X_{n+1}\}$.

Consider two service-sites h and h' according to table T. Service-site h is said to *dominate* h' if for any attribute $X \in \mathcal{X}$, $h.X \leq h'.X$ and there exists an attribute $X' \in \mathcal{X}$ such that $h.X' < h'.X'$. Given a hotel h, we define $D(h)$ to be a set of all service-sites which are dominated by h.

Definition 1 (Skyline). *A service-site $h \in H$ is in the* skyline *if h is not dominated by any service-sites in H.*

We denote all service-sites in the skyline by $SKY(H)$. In Table 1, it is easy to verify that h_1, h_2 and h_4 are in the skyline. Besides, $D(h_1) = \{h_3\}, D(h_2) = \emptyset, D(h_3) = \emptyset$ and $D(h_4) = \emptyset$.

Consider that a company wants to start a new service-site h_f and wants to find a competitive price of h_f, denoted by $h_f.p$, such that this new service-site will not be worse than any existing service-site. Suppose that $h_f.p$ is set to a non-negative value. We say that h_f meets the *skyline requirement* if h_f is in the skyline $SKY(H \cup \{h_f\})$.

Consider a scenario that the price of each existing service-site is non-zero. A trivial solution is to set the price of h_f equal to \$0. However, the company wants to earn as much profit as possible. In Example 2, we learn that if we set the price of h_f too high (e.g., \$300), then h_f is dominated by other existing service-sites. Apparently, we should choose a suitable value for the price of h_f which is not too low and too high. The following *monotonicity* property helps to determine a suitable value: *Consider two possible non-negative real numbers p_1 and p_2 where $p_1 \geq p_2$. If h_f meets the skyline requirement when $h_f.p$ is set to p_1, then h_f meets the skyline requirement when $h_f.p$ is set to p_2.*

Let p_s be a non-negative real number such that h_f meets the skyline requirement when $h_f.p$ is set to p_s. This monotonicity property suggests that h_f meets the skyline requirement when $h_f.p$ is set to *any* value at most p_s. This means that $h_f.p$ can be set to *many* possible values such that it satisfies the skyline requirement. In this paper, we are studying to return a *price range* of h_f (instead of a particular price value) such that h_f satisfies the skyline requirement. Let $p_{max,s}$ be the *maximum* possible price for the skyline requirement. Formally, we define a *price range* $\mathbf{R}_s$ of h_f in form of "$0 \leq h_f.p < p_{max,s}$" such that after we set $h_f.p$ to be any possible value in this range $\mathbf{R}_s$, h_f satisfies the skyline requirement. Note that, in order to avoid discussing the complicated and uninteresting boundary case, we assume that $h_f.p \neq p_{max,s}$ in our problem setting.

Problem 1. *(Finding Simple Competitive Price) Given a new service-site h_f, we want to find a price range $\mathbf{R}_s$ of h_f in form of "$0 \leq h_f.p < p_{max,s}$" where $p_{max,s}$ is a non-negative real number such that (1) $p_{max,s}$ is maximized and (2) h_f is in $SKY(H \cup \{h_f\})$ if we set $h_f.p$ to be any possible value in $\mathbf{R}_s$.*

After we set the price of h_f to a value within $\mathbf{R}_s$, h_f is *one* of the best choices for the customer to choose (because there may be more than one service-site in the skyline). In order that h_f becomes more competitive among all the service-sites in $SKY(H \cup \{h_f\})$, we want that h_f can attract more customers who originally chose other service-sites. This motivates us to propose another problem in which not only h_f is not worse than any existing service-site but also h_f dominates at least K existing service-sites. Intuitively, if K is larger, then h_f dominates more service-sites. Consequently, more customers will choose h_f.

We say that h_f meets the *K-dominating requirement* if h_f dominates at least K existing service-sites (i.e., $|D(h_f)| \geq K$). Similarly, let $p_{max,d}$ be the *maximum* possible price for the K-dominating requirement. We define the *price range* $\mathbf{R}_d$ of h_f in form of "$0 \leq h_f.p < p_{max,d}$" such that after we set $h_f.p$ to be any possible value in this range $\mathbf{R}_d$, h_f satisfies the K-dominating requirement.

A service-site h_f is said to meet requirement $\mathcal{R}$ if h_f satisfies *both* the skyline requirement and the K-dominating requirement. Note that $p_{max,s}$ is the maximum possible price for the skyline requirement and $p_{max,d}$ is the maximum possible price for the K-dominating requirement. Thus, the maximum possible price for requirement $\mathcal{R}$, denoted by p_{max}, is equal to $\min\{p_{max,s}, p_{max,d}\}$.

Similarly, requirement $\mathcal{R}$ satisfies the monotonicity property: *Consider two possible non-negative real numbers p_1 and p_2 where $p_1 \geq p_2$. If h_f meets $\mathcal{R}$ when $h_f.p$ is set to p_1, then h_f meets $\mathcal{R}$ when $h_f.p$ is set to p_2.*

Problem 2. *(Finding K-dominating Competitive Price) Given a non-negative integer K and a new service-site h_f, we want to find a price range $\mathbf{R}$ of h_f in form of "$0 \leq h_f.p < p_{max}$" where p_{max} is a real number such that (1) p_{max} is maximized, (2) h_f is in $SKY(H \cup \{h_f\})$ and (3) $|D(h_f)| \geq K$ if we set $h_f.p$ to be any possible value in $\mathbf{R}$.*

In the above problem formulation, Condition (2) and Condition (3) correspond to the skyline requirement and the K-dominating requirement, respectively. In the above problem, we want to maximize p_{max}. In the following, when we say the *optimal* solution, we refer to this maximized value.

Note that problem Finding K-dominating Competitive Price is more general than problem Finding Simple Competitive Price. This is because when K is equal to 0, problem Finding K-dominating Competitive Price becomes problem Finding Simple Competitive Price. In the following, we focus on solving problem Finding K-dominating Competitive Price.

A naive approach to this problem is described as follows. For each possible non-negative real number v, it tries to set the price of h_f to be v and test whether h_f satisfies the skyline requirement and the K-dominating requirement. If yes, v is a possible solution for p_{max}. Finally, it selects the greatest value such that h_f satisfies the requirements. But, since there are a large number of possible values, this approach is infeasible. In the following, we propose an approach which avoids testing the requirements with a large number of possible values.

3. SPATIAL APPROACH

In this section, we propose a spatial approach which makes use of some spatial properties and runs efficiently in large datasets.

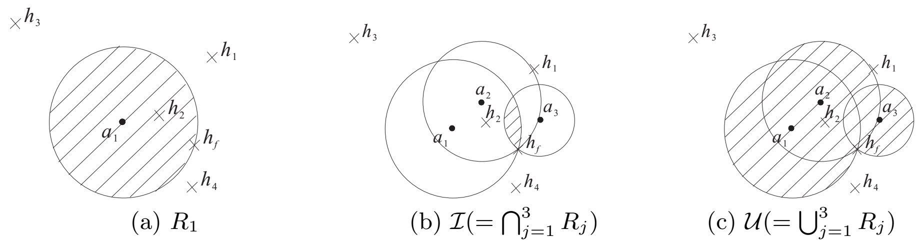
Problem Finding K-Dominating Competitive Price has two requirements, namely the skyline requirement and the K-dominating requirement. We propose a spatial approach which meets the above two requirements. Specifically, it involves the following three major phases.

- **Phase 1 (for Skyline Requirement):** We find the maximum possible price for the skyline requirement, denoted by $p_{max,s}$.
- **Phase 2 (for K-Dominating Requirement):** We find the maximum possible price for the K-dominating requirement, denoted by $p_{max,d}$.
- **Phase 3 (for Requirement $\mathcal{R}$):** We compute the maximum possible price for requirement $\mathcal{R}$ (which combines the above two requirements), denoted by p_{max}, to be $\min\{p_{max,s}, p_{max,d}\}$.

In the following, we describe how we make use of some spatial properties to perform the above three phases efficiently.

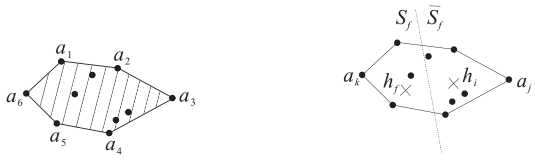
3.1 Notations

Given an attraction-site a_j and a new service-site h_f, we define the *critical region* (named as "dominance region" in [19, 20]) for attraction-site a_j, denoted by R_j, to be the



(a) R_1 (b) $\mathcal{I}(=\bigcap_{j=1}^{3} R_j)$ (c) $\mathcal{U}(=\bigcup_{j=1}^{3} R_j)$

Figure 3: Region



(a) An example (b) Illustration of proof of Lemma 1

Figure 4: Convex Hull

region occupied by the circle centered at a_j with radius equal to $d(h_f, a_j)$. For example, in Figure 3(a) which has the same objects as Figure 2, the shaded region is R_1.

The critical region for attraction-site a_j is used to efficiently determine whether a service-site $h_i \in H$ is nearer to attraction-site a_j compared with h_f. Specifically, if a service-site h_i is inside R_j, then we know that h_i is nearer to attraction-site a_j compared with h_f. Otherwise, we know that h_i is farther from a_j. For example, in Figure 3(a), since h_2 is in R_1, h_2 is nearer to a_1 compared with h_f. On the other hand, h_3 is farther from a_1 since h_3 is outside R_1.

We also define $\bigcap_{j=1}^{n} R_j$ ($\bigcup_{j=1}^{n} R_j$) to be the intersection (union) among all regions represented by $R_1, R_2, ..., R_n$. Let $\mathcal{I} = \bigcap_{j=1}^{n} R_j$ and $\mathcal{U} = \bigcup_{j=1}^{n} R_j$. For example, Figure 3(b) and Figure 3(c) show the same objects as Figure 3(a) with two additional attraction-sites, namely a_2 and a_3. The shaded region in Figure 3(b) denotes $\mathcal{I}$ and the shaded region in Figure 3(c) denotes $\mathcal{U}$.

Given a set A of n objects in a Euclidean space, the *convex hull* of A [10] is a minimal set of objects in A such that these objects form a convex polygon and all objects in A are inside the region occupied by this convex polygon. Let the *region* occupied by the convex polygon for the convex hull of A be $CH(A)$. For example, Figure 4(a) shows some objects represented by black dots where A is equal to a set of all black dots. The convex hull of A is $\{a_1, a_2, a_3, a_4, a_5, a_6\}$. The convex polygon in the figure corresponds to the polygon for the convex hull of those objects. The shaded region in the figure corresponds to $CH(A)$ (i.e., the region occupied by the polygon).

3.2 Properties

In this subsection, we give some *spatial properties* for problem Finding K-Dominating Competitive Price which can be used to speed up the computation.

This problem has two requirements. Consider the first requirement, the skyline requirement. In order to determine $p_{max,s}$ for the skyline requirement, we have the following two lemmas.

LEMMA 1. *Suppose h_f is inside $CH(A)$. h_f is in the skyline $SKY(H \cup \{h_f\})$ no matter what value $p_{max,s}$ is.*

Proof: We prove by contradiction. Suppose $h_f.p$ is set to a non-negative value such that h_f is not in the skyline $SKY(H \cup \{h_f\})$. That is, there exists a service-site h_i dominating h_f.

Since h_i dominates h_f, we deduce that, for all attribute $X \in \mathcal{X}$, $h_i.X \leq h_f.X$ and there exists an attribute $X' \in \mathcal{X}$ such that $h_i.X' < h_f.X'$. Consider that we draw a perpendicular bisector of a line segment joining h_i and h_f. Figure 4(b) shows an example that h_f is inside $CH(A)$ and there exists a service-site h_i and an attraction-site a_j where $d(h_i, a_j) < d(h_f, a_j)$. The dashed line denotes the perpendicular bisector of a line segment joining h_i and h_f. The bisector cuts the Euclidean space into two sides: one side S_f containing h_f and the opposite side $\overline{S}_f$ not containing h_f. It is easy to verify that, for each attraction-site a which is inside S_f, $d(h_f, a) < d(h_i, a)$. Since h_f is inside $CH(A)$, we deduce that S_f contains a service-site a_k other than a_j in the convex hull of A. Since a_k is inside S_f, we deduce that $d(h_f, a_k) < d(h_i, a_k)$. That is, $h_f.X_k < h_i.X_k$, which leads to a contradiction that, for all attribute $X \in \mathcal{X}$, $h_i.X \leq h_f.X$. □

LEMMA 2. *Suppose h_f is not inside $CH(A)$. If we set $p_{max,s}$ to be $\min_{h \in \mathcal{I}} h.p$, then h_f is in the skyline $SKY(H \cup \{h_f\})$.*

Proof: Note that h_i is nearer to all attraction-sites compared with h_f if and only if h_i is inside $\mathcal{I}$. With notation $\mathcal{I}$, Lemma 2 can be re-written as follows: "h_i dominates h_f if and only if (1) h_i is inside $\mathcal{I}$ and (2) $h_f.p \geq h_i.p$". If we set $p_{max,s}$ to be $\min_{h \in \mathcal{I}} h.p$, since $h_f.p < p_{max,s}$, h_f is in the skyline $SKY(H \cup \{h_f\})$. □

From Lemma 1 and Lemma 2, we learn that, if h_f is inside region $CH(A)$, $p_{max,s}$ can be set to any value and thus h_f is in the skyline $SKY(H \cup \{h_f\})$. Note that we do not need to scan any service-sites in this case. If h_f is not inside region $CH(A)$, then $p_{max,s}$ is set to be $\min_{h \in \mathcal{I}} h.p$. In this case, we only need to scan the service-sites in region $\mathcal{I}$.

Consider the second requirement, the K-dominating requirement. In order to determine $p_{max,d}$ for this requirement, we have the following lemma.

LEMMA 3. *Suppose that there are at least K service-sites not in $\mathcal{U}$. If we set $p_{max,d}$ to be the K-th greatest price (i.e., $h.p$) among all service-sites h not in $\mathcal{U}$, then h_f dominates at least K service-sites.*

Proof: Note that h_i is farther from all attraction-sites compared with h_f if and only if h_i is not inside $\mathcal{U}$. With notation $\mathcal{U}$, we re-write Lemma 3 as follows "h_f dominates h_i if and only if (1) h_i is not inside $\mathcal{U}$ and (2) $h_f.p \leq h_i.p$." If we set $p_{max,d}$ to be the K-th greatest price (i.e., $h.p$) among all service-sites h not in $\mathcal{U}$, since $h_f.p < p_{max,d}$, h_f dominates at least K service-sites. □

With the above lemmas, in order to meet the K-dominating requirement, we have to set $p_{max,d}$ to be the

Algorithm 1 Algorithm for Finding K-Dominating Competitive Price

```
1: find $\mathcal{I}$ and $\mathcal{U}$
2: // Phase 1: To meet the skyline requirement, we find a price $p_{max,s}$
3: find $CH(A)$
4: if $h_f$ is inside $CH(A)$ then
5:     $p_{max,s} \leftarrow \infty$
6: else
7:     $p_{max,s} \leftarrow \min_{h \text{ is inside } \mathcal{I}} h.p$
8: end if
9: // Phase 2: To meet the $K$-dominating requirement, we find a price $p_{max,d}$
10: $p_{max,d} \leftarrow$ the $K$-th greatest price among all service-sites not in $\mathcal{U}$
11: // Phase 3: Finding $K$-Dominating Competitive Price $p_{max}$
12: $p_{max} \leftarrow \min\{p_{max,s}, p_{max,d}\}$
13: return $p_{max}$
```

K-th greatest price among all service-sites not in $\mathcal{U}$. There are two issues related to the above lemma. The first issue is how we set $p_{max,d}$ when there are less than K service-sites not in $\mathcal{U}$. In this case, we report to the user that h_f cannot dominate at least K service-sites with the current value of K and suggest the user should provide a smaller value of K. The second issue is how we set $p_{max,d}$ when K is set to 0. In this case, we set $p_{max,d}$ to ∞.

In order to satisfy requirement $\mathcal{R}$ (which combines the above two requirements), the final price is set to $p_{max} = \min\{p_{max,s}, p_{max,d}\}$.

3.3 Algorithm

Algorithm 1 shows the algorithm for finding K-dominating competitive price. With the above lemmas, it is easy to verify the following theorem.

Theorem 1 (Correctness). *Algorithm 1 returns the optimal solution of problem Finding K-Dominating Competitive Price.*

Some readers may notice that $p_{max,s}$ can be equal to ∞ (which can be found in line 5 of Algorithm 1) when h_f is inside $CH(A)$. However, if K is greater than 0, then it is easy to verify that $p_{max,d}$ is equal to a value at most the price of one of the service-sites in H and thus is not equal to ∞. Finally, the competitive price p_{max} (which is equal to $\min\{p_{max,s}, p_{max,d}\}$) is not equal to ∞. If K is equal to 0, then we set $p_{max,d}$ to ∞. In this case, it is possible that p_{max} is ∞ (because $p_{max,s}$ can be equal to ∞ and $p_{max} = \min\{p_{max,s}, p_{max,d}\}$). Since it is not reasonable to return ∞ as the answer for p_{max} in real-life applications, in our implementation, we set p_f to be the price of the nearest service-site of h_f.

We know that the lemmas (Lemmas 1, 2 and 3) can help us to find the competitive price efficiently since we do not need to scan all service-sites in H. Besides, we make use of an indexing structure, an R*-tree, to further speed up the computation of finding some service-sites dominated by a given service-site. Moreover, we will give a theoretical time complexity analysis.

3.4 Detailed Steps and Theoretical Analysis

In Algorithm 1, we need to determine two regions $\mathcal{I}$ and $\mathcal{U}$. Besides, we also need to find the service-sites in $\mathcal{I}$ and find the service-sites not in $\mathcal{U}$. In the following, we describe how we achieve the above steps *efficiently* by using some *spatial index techniques.*

Now, we give the detailed steps of Algorithm 1 and analyze its complexity. There are five major steps in the algorithm.

Step 1 (Finding $\mathcal{I}$ and $\mathcal{U}$): Firstly, for each $a_j \in A$, we construct R_j which is a circle centered at a_j with radius $d(h_f, a_j)$. Since the circle construction takes $O(1)$ time and there are n attraction-sites in A, this step takes $O(n)$ time. Secondly, we construct $\mathcal{I}$ ($\mathcal{U}$) by performing an intersection (union) operation over all R_j's. In our implementation, we conceptually represent $\mathcal{I}$ ($\mathcal{U}$) by storing a list L of R_j's. Note that L contains n elements for R_j. This step takes $O(n)$ time. Thus, Step 1 takes $O(n)$ time.

Step 2 (Finding Convex Hull): Step 2 finds the convex hull over set A. Let $\alpha(N)$ be the running time to find the convex hull over a set of size N. Step 2 takes $O(\alpha(n))$ time. We adopt the algorithm from [10] to find the convex hull where the running time of this algorithm is $O(N \log N)$ time. Thus, Step 2 requires $O(n \log n)$ time.

Step 3 (Checking whether h_f is inside $CH(A)$): It is easy to verify that checking whether h_f is inside $CH(A)$ requires $O(n)$ time. If h_f is inside $CH(A)$, then we assign $p_{max,s}$ with ∞, which takes $O(1)$ time. Otherwise, we do the following step. We find all service-sites in region $\mathcal{I}$. This step can be done by performing range queries over set H. Specifically, for each R_j in L (representing $\mathcal{I}$), we perform a range query over set H with a circle centered at a_j with radius $d(h_f, a_j)$. Let $\beta(N)$ be the running time of a range query over the dataset of size N. Since L contains n elements and each range query takes $O(\beta(m))$ time, the running time of this sub-step is $O(n \cdot \beta(m))$. Then, we perform intersection operations among all results obtained from the above range queries. With the bitwise implementation, an intersection operation with two sets can be done in $O(1)$ time. Since there are $O(n)$ intersection operations, this step takes $O(n)$ time. Among all service-sites in region $\mathcal{I}$, we find the smallest price. The overall running time for Step 3 is $O(n + n \cdot \beta(m) + n) = O(n \cdot \beta(m))$. If $\mathcal{I}$ is empty or there is no service-site in $\mathcal{I}$, then we assign $p_{max,s}$ with ∞ immediately.

In the literature, $\beta(m)$ is theoretically bounded [9]. Let J be the greatest result size of a range query (i.e., the greatest number of service-sites in the range). In our problem setting, since a range query can be executed in $O(J + \log m)$ time [9], Step 3 takes $O(n(J + \log m))$ time.

In our implementation, we adopt an R*-tree [7] to support range queries. It has been shown that the R*-tree performs efficiently in real cases and is commonly adopted for range queries although it does not have good worst-case asymptotic performance. Specifically, we build an R*-tree over all service-sites in H and then perform a range query over this R*-tree.

Step 4 (Finding the K-th Greatest Price Among all Service-sites not in $\mathcal{U}$): Firstly, we find a set R of all service-sites in $\mathcal{U}$. Similar to Step 3, this step can be done in $O(n \cdot \beta(m))$ time. Secondly, we find a set S of all service-sites not in $\mathcal{U}$ by $H - R$, which takes $O(1)$ with the bitwise implementation. Thirdly, we sort all service-sites in S in descending order of their prices, which takes $O(m \log m)$ time. Fourthly, we find the service-site h with the K-th greatest price, which takes $O(1)$ time. This value corresponds to $p_{max,d}$. The running time of Step 4 is $O(n \cdot \beta(m) + 1 + m \log m + 1) = O(n \cdot \beta(m) + m \log m)$ time.

With the method used in [9], Step 4 can be done in $O(n(J + \log m) + m \log m)$ time.

Step 5 (Finding the minimum value from $\{p_{max,s}, p_{max,d}\}$): We find p_{max} with $\min\{p_{max,s}, p_{max,d}\}$, which takes $O(1)$ time.

Thus, the running time of Algorithm 1 is $O(n + \alpha(n) + n \cdot \beta(m) + n \cdot \beta(m) + m \log m + 1) = O(n + \alpha(n) + n \cdot \beta(m) + m \log m)$ time.

THEOREM 2. *The complexity of Algorithm 1 is $O(n + \alpha(n) + n \cdot \beta(m) + m \log m)$.*

By using the methods used in [10] and [9], Algorithm 1 takes $O(n + n \log n + n(J + \log m) + m \log m) = O(n \log n + n(J + \log m) + m \log m)$ time.

4. DISCUSSION

In this section, we focus on three issues related to our problem. The first one is how to apply our method when multiple non-spatial attributes are considered (Section 4.1). The second one is how to find a reasonable value of K for the K-dominating requirement (Section 4.2).

4.1 Handling Multiple Non-spatial Attributes

In this section, we will extend our problems and our proposed techniques to a general scenario when there are multiple non-spatial attributes. In this paper, we study one single non-spatial attributes, namely attribute Price, in order to simplify our discussion. In our running example, each service-site can have multiple non-spatial attributes. For example, in addition to attribute Price, it can have non-spatial attributes such as star rate. Under this general scenario, the two problems studied in this paper are still the same except the definition of dominance relationship among service-sites for the decision making table.

Formally, suppose each service-site h has q non-spatial attributes, namely $\gamma_1, \gamma_2, ..., \gamma_q$. Without loss of generality, let the last attribute γ_q be attribute Price. For $k \in [1, q]$, the value of each attribute γ_k for h is represented by $h.\gamma_j$. In this problem, for each existing service-site h and each non-spatial attribute γ_j, $h.\gamma_j$ is given. For the new service-site h_f, each non-spatial attribute γ_j other than γ_q is given. We want to find $h_f.\gamma_q$ to satisfy the skyline requirement and the K-dominating requirement. That is, we study Problem 2 but there are multiple non-spatial attributes (instead of a single non-spatial attribute).

We adapt the construction of the decision-making table with $n + q$ attributes as follows. For each object $h \in H$, we construct a tuple in form of $(x_1, x_2, ..., x_n, x_{n+1}, x_{n+2}, ..., x_{n+q})$ where x_j is equal to $d(h, a_j)$ for each $j \in [1, n]$ and x_j is equal to $h.\gamma_{j-n}$ for each $j \in [n + 1, n + q]$. We denote each value x_j by $h.X_j$ by $j \in [1, n + q]$. Let $\mathcal{X}' = \{X_1, X_2, ..., X_{n+q}\}$. The dominance relationship is defined on $\mathcal{X}'$ instead of $\mathcal{X}$. With this adapted dominance relationship, we can define the two problems accordingly under this general scenario.

Our proposed approach (Algorithm 1) can be adapted to this general scenario. All changes in the algorithm for this scenario are related to the adapted dominance relationship which involves the *additional* non-spatial attributes. Specifically, there are two changes in the algorithm. The first

Service-site	Star rate (γ_1)	Price (γ_2)
h_1	3	100
h_2	4	250
h_3	3	200
h_4	4	220

Table 2: The running example with 2 non-spatial attributes

Hotel	Distance-to-beach(km)	Star rate	Price($)
h_1	3.0	3	100
h_2	1.0	4	250
h_3	4.0	3	200
h_4	2.5	4	220

Table 3: A decision-making table T

change is related to the skyline requirement. Let H' be a set of service-sites h inside $\mathcal{I}$ such that, for each non-spatial attribute γ other than attribute Price (i.e., γ_q), $h.\gamma \le h_f.\gamma$. In Line 7 of Algorithm 1, we modify it to "$p_{max,s} \leftarrow \min_{h \in H'} h.p$". The second change is related to the K-dominating requirement. Let H'' be a set of service-sites h not in $\mathcal{U}$ such that, for each non-spatial attribute γ other than attribute Price (i.e., γ_q), $h_f.\gamma \le h.\gamma$. In Line 9 of Algorithm 1, we modify it to "$p_{max,d} \leftarrow$ the K-th greatest price among all service-sites h in H''". Let us call the modified algorithm *Improved Algorithm*

It is easy to verify the following theorem.

THEOREM 3. Improved Algorithm *returns the optimal solution of problem Finding K-Dominating Competitive Price when there are multiple non-spatial attributes.*

EXAMPLE 1 (MULTIPLE NON-SPATIAL ATTRIBUTES). *Let us illustrate the algorithm with our running example. We use the example as shown in Figure 1 but the price table is changed from the table in Figure 1 (which contains one non-spatial attribute, namely "price") to the table in Table 2 (which contains two non-spatial attributes, namely "star rate" and "price"). Here, attribute "star rate" corresponds to γ_1 while attribute "price" corresponds to γ_2. Note that different from attribute "price" in which a smaller value is more preferable, in attribute "star rate", a larger value is more preferable.*

Similarly, we have the corresponding decision-making table as shown in Table 3.

In this example, suppose that we set the star rate of the new hotel h_f to 3 and the location of h_f to the location as shown in Figure 2. Let K be 2.

In Phase 1, we find all the hotels in $\mathcal{I}$. Only one service-site, h_2, locates inside $\mathcal{I}$. Since $h_2.\gamma_1 = 4 > 3 = h_f.\gamma_1$, $H' = \{h_2\}$. Therefore, $p_{max,s}$ is set to the price of h_2, \$250.

In Phase 2, we find all the hotels not in $\mathcal{U}$. There are three service-sites, h_1, h_3 and h_4, not in $\mathcal{U}$. And all of them have a value on γ_1 no less than h_f. Thus, $H'' = \{h_1, h_3, h_4\}$. The 2-nd greatest price among all the hotels in H'' is the price of h_3, \$200. Therefore, $p_{max,d}$ is set to be \$200.

In Phase 3, we can compute $p_{max} = \min\{\$250, \$200\} = \$200$. Thus, the 2-dominating price of h_f is \$200. □

4.2 How to set K

How to set a proper K is essential for the K-dominating requirement. In some cases, users have their mind to set the value of K because they want that h_f must dominate at least K service-sites in the existing market. However, in some other cases, users do not have any idea about how to set the value of K. In this section, we propose a method to help users to determine the value of K in this case.

As we know, different values of K result in different prices, but which price among those is more reasonable and thus benefits the new service-site is still not known. In this subsection, we propose a model to suggest a way to find an appropriate value of K based on the well-studied field of economy and business [8].

There are some existing models in economical and business research studying *customer retention/attrition* [8]. In this subsection, we borrow the concept of the traditional demand-and-supply model [8] in the literature to find an appropriate value of K.

In this model, each service-site h is associated with a *demand*, denoted by $h.d$, representing the total number of customers that would like to choose this service-site. In our running example, if there are 50 customers who want to accommodate in hotel h, the demand of this hotel $h.d$ is 50. In our problem, for each $h \in H$, $h.d$ is given. However, $h_f.d$ is not given. In the following, we propose a model to estimate $h_f.d$.

The demand of each service-site can be obtained from some external sources of information about the proportion of the usage of the service-sites. In our motivating example, this information can be the proportion of using the hotels [3]. Similar kinds of information are also available in other applications like setting a selling price of an apartment [2] and setting a parking fee of a car park [4].

Definition 1 (Potential Loser). *Given a service-site h in H, h is said to be a* potential loser *if and only if there exists a non-negative real number p such that when $h_f.p$ is set to p, h_f dominates h.*

We define the set of all potential losers in H to be PL. Without loss of generality, we assume that PL contains l service-sites. Note that h_f dominates at most l service-sites if we set $h_f.p$ to a particular non-negative real number. Thus, the greatest possible value of K that we can set is l. Without loss of generality, we assume that PL contains $h_1, h_2, ..., h_l$ where $h_i.p \geq h_{i+1}.p$ for $i = 1, 2, ..., l-1$.

Definition 2 (Real Loser). *Given a potential loser h and a non-negative real number p, h is said to be a* real loser *with respect to p if and only if $h_f.p$ is set to p and h_f dominates h.*

Given a non-negative real number p, we define the set of all real losers with respect to p to be $RL(p)$. If h is a real loser with respect to p, then we know that h_f dominates h. Thus, we know that some of the current customers choosing h may change their preference and finally choose h_f instead of h. However, in the real-life applications, not all customers choosing h are eager to making this change. In order to capture this, we define an input parameter α which is a real number between 0 and 1 and denotes the *transfer rate* representing the fraction of customers who originally want to choose h (before h_f is set up) and finally choose h_f (after h_f is set up). Similarly, the transfer rate α can be obtained from some external sources of information about customers' behaviors. There are a lot of studies related to customers' behaviors in the literature of psychology, sociology, social anthropology and economics. This information can be found in the studies about customers' behaviors [18, 14].

Suppose that $h_f.p$ is set to p. According to this transfer rate, given a real loser h with respect to p, there are $\alpha \times h.d$ customers who originally want to choose h (before h_f is set up) and finally choose h_f (after h_f is set up). Thus, by considering all real losers in $RL(p)$, we deduce that the total number of customers who originally want to choose some real losers with respect to p and finally choose h_f is equal to

$$\sum_{h \in RL(p)} \alpha \times h.d$$

Let us denote the above equation by a function f as follows.

$$f(p) = \sum_{h \in RL(p)} \alpha \times h.d$$

Now, we are ready to define a formula for $h_f.d$ as follows.

$$h_f.d = f(p)$$

Next, we describe how to determine the value of K by using a new concept of *utility*. Given a non-negative real number p, the *utility* of h_f with respect to p, denoted by $U(p)$, is defined as follows.

$$U(p) = p \times f(p)$$

Note that $f(p)$ corresponds to $h_f.d$. We conclude that $p \times f(p)$ corresponds to the total income for h_f.

Let $u_i = U(h_i.p)$ for $i = 1, 2, ..., l$. We deduce that

$$u_i = h_i.p \times f(h_i.p)$$

Note that $h_i.p$ is monotonically decreasing when i increases. Besides, $f(h_i.p)$ is monotonically increasing when i increases. Thus, we do not know whether u_i is larger than u_{i+1} or not where $i = 1, 2, ..., l-1$.

According to $u_1, u_2, ..., u_l$, we want to determine the value of K as follows. Firstly, we calculate all values $u_1, u_2, ..., u_l$. Secondly, we find an integer i_o which gives the greatest value of u_{i_o} among $u_1, u_2, ..., u_l$ (i.e., $i_o = argmax_i u_i$). Thirdly, we set K to i_o.

Example 2 (Finding K). *In the running example of Figure 2, the set of potential losers is $\{h_1, h_3, h_4\}$. Thus, l is 3 and the greatest possible value of K we can set is 3. Suppose that $h.d$ is set to 100 for each $h \in H$ and α is set to 0.1. We obtain that $u_1 = 2200$, $u_2 = 4000$ and $u_3 = 3000$. Thus, the value K is 2 and the maximum utility is 4000. We will apply this model in the case study described in Section 5.* □

5. EMPIRICAL STUDIES

In this section, we verify the scalability of our proposed algorithms. The algorithms were implemented in C/C++. All the experiments were performed on a 2.4GHz PC with 4.0GB RAM, on a Linux platform. We ran experiments on both real and synthetic datasets.

The synthetic datasets were generated as follows. Firstly, we collect the locations of objects in North American (e.g., roads, populated places and cultural landmarks) from Digital Chart of the World [5]. Let W be the set of objects.

Then, from W, we randomly select m objects as service-sites and n objects as attraction-sites. These service-sites and these attraction-sites form set H and set A, respectively. The location of a new service-site h_f is randomly generated. Since there is no attribute related to price in set H, we generate the price of each service-site in H as follows. For each service-site h in H, we find its nearest attraction-site a and set $h.p$ to be a value which is randomly picked from a normal distribution with mean equal to $x/(1 + d(h, a)^2)$ and standard derivation equal to σ, where x and σ are two input parameters. Intuitively, x is the (expected) greatest possible price of a service-site when we consider the nearest attraction-site. By default, we adopt $x = 150$ and $\sigma = 27$ since the real dataset (to be described next) has such a distribution. The default values of the parameters are shown as follows: $m = 200000, n = 20, K = 20, x = 150$ and $\sigma = 27$. In the following, we use the default settings unless specified otherwise.

The real dataset was obtained from Surfy Hotel [6] which provides information about hotels in North America, including price and location. We select 20,000 hotels for our experiment. We chose four attraction-sites, namely Status of Liberty, Empire State Building, Museum of Art and Wall Street. Similarly, the location of a new service-site h_f is randomly generated.

We implemented four algorithms, namely (a) *Blind-Naive*, (b) *Guided-Naive*, (c) *3-Phase(No Index)* and (d) *3-Phase(Index)*. (a) *Blind-Naive* is the naive algorithm we described at the end of Section 2. In our implementation, *Blind-Naive* tries to find a set of possible prices starting from 0 with an incremental count of 0.01 (i.e., 0.00, 0.01, 0.02, ..) such that these prices meet the skyline requirement and the K-dominating requirement. It continues the process until the largest price which meets the requirements is found. This largest price corresponds to the answer of *Blind-Naive*. (b) *Guided-Naive* is similar to *Blind-Naive*. Instead of trying all possible prices in an increment count of 0.01, *Guided-Naive* tries to find all possible prices of existing service-sites. These prices can be considered as candidates for the competitive price. Similarly, it tries to find the largest price among these prices which meet the requirements as the final answer. (c) *3-Phase(No Index)* corresponds to Algorithm 1 (which is a three-phase algorithm) but it is not equipped with any index. Specifically, it has to find $\mathcal{I}$ and $\mathcal{U}$ without using any index in order to determine $p_{max,s}, p_{max,d}$ and p_{max}. (d) *3-Phase(Index)* corresponds to Algorithm 1 and it is equipped with an index, namely an aggregate R*-tree [22], for computation. We adopted an aggregate R*-tree [22] for the range query in *3-Phase(Index)* where the maximum (minimum) number of entries in a node is equal to 20 (10).

In the following, for clarity, we simply denote *Blind-Naive* and *Guided-Naive* as *Blind* and *Guided*, respectively.

Since problem Finding K-Dominating Competitive Price is a more general problem than Finding Simple Competitive Price, in the following experimental results, we study the former problem only.

5.1 Scalability

We evaluate the algorithms with two measurements: (1) *Price* and (2) *Query Time*. (1) Price corresponds to three different types of prices, denoted by $p_{max,s}$, $p_{max,d}$ and p_{max}, respectively. Since $p_{max,s}$ may be equal to ∞ in some cases, in our experiment, we only report any value not equal to ∞ for $p_{max,s}$. Besides, note that $p_{max} = \min\{p_{max,s}, p_{max,d}\}$. (2) Query time refers to the time of executing the algorithm to find the price. For each measurement, each experiment was conducted 1,000 times and the average of the results was reported. We studied the effects of K, n and m as follows.

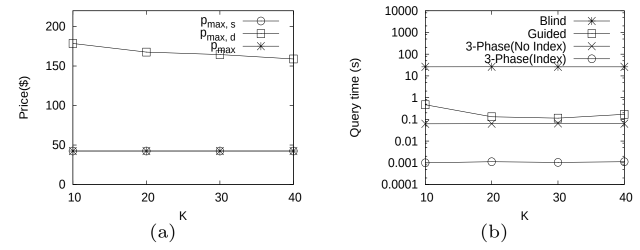


Figure 5: Effect of K (Synthetic dataset)

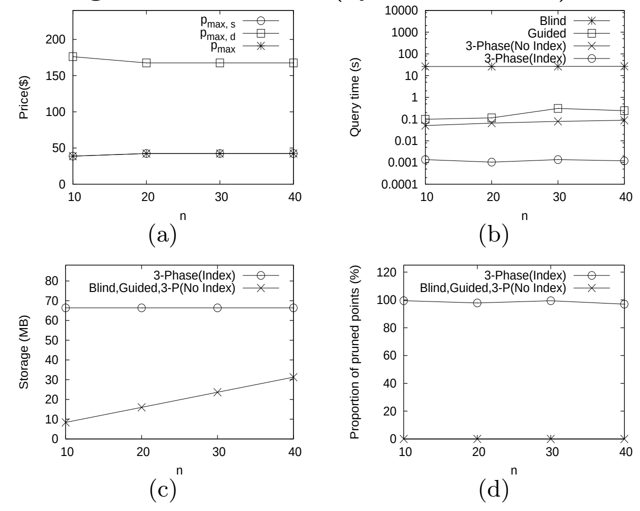


Figure 6: Effect of n (Synthetic dataset)

5.1.1 *Effect of* K

Figure 5(a) shows when K increases, $p_{max,d}$ decreases and $p_{max,s}$ remains the same. $p_{max,s}$ is not affected by the number of dominated service-sites, while $p_{max,d}$ decreases since the K-th greatest price of the service-sites outside $\mathcal{U}$ decreases when K increases. In Figure 5(b), the query time of each algorithm increased slightly except *Guided* when K increases. This is because each algorithm has to find more existing service-sites which are dominated by h_f when K increases. Note that the query time of *3-Phase(Index)* is two orders of magnitude less than *3-Phase(No Index)*, and much less than the other two algorithms, because *3-Phase(Index)* does not access a lot of objects due to its pruning feature. Besides, the query times of the four algorithms decrease in the order of *Blind*, *Guided*, *3-Phase(No Index)*, *3-Phase(Index)*.

5.1.2 *Effect of* n

In Figure 6(a), when n increases, $p_{max,s}$ increases. This is because, if the total number of attraction-sites increases, then it is less likely that a service-site dominates the new service-site h_f. However, $p_{max,d}$ decreases slightly when n increases. This is because, if there are more attraction-sites, then similarly, it is also less likely that the new service-site

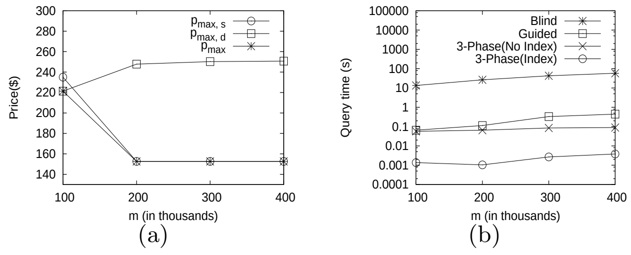


Figure 7: Effect of m (Synthetic dataset)

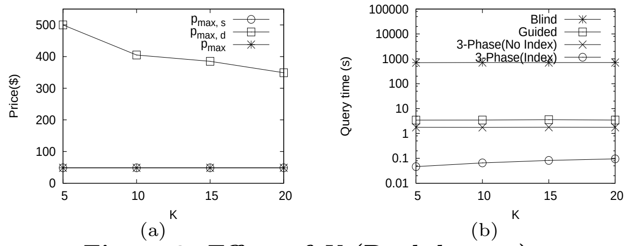


Figure 8: Effect of K (Real dataset)

h_f dominates other service-sites. However, in Figure 6(b), the query time of *3-Phase(Index)* remains nearly the same when n increases, but the query time of other three algorithms increases as n increases.

5.1.3 Effect of m

Figure 7(a) shows that, when the number of service-sites increases, $p_{max,s}$ decreases. This is because, if there are more service-sites, then it is more likely that a service-site dominates the new service-site h_f. However, in the figure, $p_{max,d}$ increases with the number of service-sites because it is more likely that h_f can dominate other service-sites. Thus, $p_{max,d}$ can be higher. In Figure 7(b), the query times of both all algorithms increase with m. The storage of all the algorithms increases with m, as shown in Figure 7(c). In Figure 7(d), the proportion of pruned points of *3-Phase(Index)* increases slightly when m increases.

5.1.4 Effect of K on Real Dataset

We conducted experiments on real datasets, and the experimental results with the variation of K are shown in Figure 8. The trends are similar to those for the synthetic dataset.

We also conducted experiments with the variation of m on the real dataset and the variation of σ and x on the synthetic dataset. For the sake of space, we omit the figures.

5.2 Case Study

In order to illustrate the practicality of our algorithm, we use a real dataset with Manhattan hotels to show how it can be used in reality. In this dataset, hotels correspond to service-sites and attractions correspond to attraction-sites in our problem setting. We choose two attractions, namely Empire State Building and Bull Sculpture in the Wall Street. Besides, suppose the new location where we want to set up a new hotel h_f is near to Empire State Building. In this dataset, there are 154 hotels, and the prices of these 154 hotels ranges from \$70 to \$463. The average price and the standard deviation of these hotels are \$153 and \$67, respectively. There are 13 1-star hotels, 65 2-star hotels, 48 3-star hotels, 27 4-star hotels and 1 5-star hotel. We conducted two sets of experiments for the case study. The first set is that each hotel (or service-site) is associated with its location and its price only. The second set is that each hotel is associated with its star-rate (one additional non-spatial attribute) in addition to its location and its price. In the second set, we adapted our proposed algorithm to deal with the case with more than one non-spatial attribute.

Star Rate of h_f	$K = 1$	$K = 2$
1	81	81
2	81	80
3	99	96
4	149	141
5	284	209

Table 4: Value of p_{max} under different star rates

For the first set, we find that $p_{max,s} = \$81$. If we set $K = 1$, then $p_{max,d} = \$126$. Thus, $p_{max} = \min\{p_{max,s}, p_{max,d}\} = \$81$. Similarly, if we set $K = 2$, we have $p_{max,s} = \$81$ and $p_{max,d} = \$119$. Thus, we have $p_{max} = \$81$. For the second set, p_{max} has different values if we set the star rate of the new hotel to different star rates. Table 4 shows the results of p_{max} with different star rates of the new hotel h_f. In general, if the star rate of h_f is higher, then p_{max} is higher. This is because usually, an existing hotel with a higher star rate has a higher price. When K is higher, p_{max} decreases. This is because in order that h_f can dominate more hotels, p_{max} is set to a smaller value.

5.3 Experiment for Determining An Appropriate Value of K

In the following, the objective is to determine the value of K by using the method introduced in Section 4.2. Here, we also use the real dataset in Section 5.2. There are the following three major steps. The first step is to obtain the demands of all the hotels in the existing market. The second step is to obtain the transfer rate α. The final step is to find an appropriate value of K according to the information obtained in the previous steps.

The first step can be done by gathering the information about the *hotel occupation rate* (which corresponds to the proportion of the number of rooms occupied by customers). According to the NYC statistical report [1], on average, the occupation rate of a hotel in New York City is around 80%. With this occupation rate, we can calculate the demand of each hotel by multiplying the total number of rooms in the hotel with this occupation rate.

The second step can be done similarly by gathering the information about customers' behaviors. We will derive the transfer rate α from two sources of information. According to the first source, a statistical report from The Harvard Business Review, a company loses 50% of their customers every five years on average [18]. In other words, 10% of customers are lost every year. According to the second source, the report from the U.S. Small Business Administration and the U.S. Chamber of Commerce [14], about 82% of the customers who do not continue choosing the original company choose the other better companies finally because they are upset with the treatment they have received from the original company. By combining the above two sources, we conclude that about 8.2% of all customers who originally choose a particular company will probably choose other better companies finally. Thus, we obtain $\alpha = 0.082$.

The third step is to find an appropriate value of K according to the information obtained in the previous steps. In this experiment, we find that the set of potential losers contain 36 hotels. Suppose that we set the star rate of the new hotel to be 2. According to the method introduced in Section 4.2, we first find $u_1, u_2, ..., u_{36}$. In this experiment, $u_1 = 894.26$ and $u_2 = 3345.34$. We also compute u_i for $i = 3, 4, ..., 36$. Finally, we find that u_{35} has the highest value among all values $u_1, u_2, ..., u_{36}$ and it is equal to 26834.46. Thus, we find the appropriate value of K as 35. After we set K to be 35, we can find the corresponding competitive price of h_f as $79.

Conclusion: We find that *3-Phase(Index)* finds the competitive price of a new service-site more efficiently compared with algorithms *Blind*, *Guided* and *3-Phase(No Index)*. Generally, *3-Phase(Index)* performs faster than other algorithms at least two orders of magnitude.

6. RELATED WORK

Skyline has been studied in the literature of databases for some years [21, 13, 24, 12, 25, 17, 15]. Some examples are a bitmap method and a branch-and-bound skylines (BBS) method [16]. Besides, [19, 20] studies how to find the skyline over some service-sites in a Euclidean space when the price of each service-site is given.

Recently, due to the usefulness of the analysis with the concept of skyline, data mining community [24, 12, 11, 25] has recently started to study the applications using skyline. Given a service-site h, [24] proposes a decision-making problem to find what user preferences under which h is in the skyline. [12] studies how to mine user preferences with the use of skyline when the users provide a list of "best" service-sites in their minds. [11] finds skyline objects over data streams by considering an arbitrary subset of attributes. Given a set of products in the existing market and a set of user preferences, [25] defines an objective function for profit (or more specifically market share) and tries to find a set of prices for these products for maximizing the objective function. [25] is different from us because the dominance relationship defined in [25] is based on two different objects, namely products and user preferences. However, the dominance relationship defined here is based on the same type of objects (i.e., service-sites). Besides, [25] does not consider the spatial location of products (or services) but we do. [23] is to find a set of products which are in the skyline. [23] is different from ours because all attribute values (including attribute Price) of the products are *given* but we need to *find* the value of attribute Price. Besides, the spatial location of products is not considered in [23].

To the best of our knowledge, *most existing works assume that an attribute value of a service-site is given.* In this paper, we study how to find an attribute value, specifically price, of a new service-site.

7. CONCLUSION

In this paper, we identify and tackle two interesting data mining problems, finding simple competitive price and finding K-dominating competitive price. Although setting price comes naturally in many real life applications, we are the first to propose to find the price of a new service-site with the use of skyline techniques. As future work, in addition to price, finding the "best" location of a new service-site is an interesting topic. Secondly, in the proposed model, each attraction-site is represented by a single point. It is interesting to study when an attraction-site is represented by an arbitrary region. Thirdly, studying the problem with objects in the metric space other than Euclidean space is also a possible topic.

Acknowledgements: We are grateful to the anonymous reviewers for their constructive comments on this paper. The research is supported by grants FSGRF13EG27.

8. REFERENCES

[1] Nyc statistics. In *http://www.nycgo.com/?event=view.article&id=78912.*

[2] Realfacts. In *http://realfacts.com/.*

[3] Smith travel research. In *http://www.strglobal.com/News/News.aspx.*

[4] Town of riverhead parking management workshop. In *http://www.riverheadli.com/07.09.parking.management.workshop.pdf.*

[5] Rtree portal. In *http://www.rtreeportal.org/spatial.html*, 2009.

[6] Surfy hotel. In *http://www.surfy.com/*, 2009.

[7] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *SIGMOD*, 1990.

[8] D. Besanko and R. Braeutigam. *Microeconomics.* Wiley, December 7, 2004.

[9] B. Chazelle. New upper bounds for neighbor searching. In *Information and Control*, 1986.

[10] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inf. Process. Lett.*, 1(4):132–133, 1972.

[11] Z. Huang, S. Sun, and W. Wang. Efficient mining of skyline objects in subspaces over data streams. *Knowledge and Information System*, 22:159–183, February 2010.

[12] B. Jiang, J. Pei, X. Lin, D. W-L Cheung, and J. Han. Mining preferences from superior and inferior examples. In *SIGKDD*, 2008.

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

[14] W. Maguire. Six reasons we lose customers. August 24, 2007.

[15] M. Nagendra and K. S. Candan. Skyline-sensitive joins with lr-pruning. In *EDBT*, 2012.

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

[17] Y. Peng, R. C.-W. Wong, and Q. Wan. Finding top-k preferable products. In *TKDE*, 2012.

[18] F. F. Reichheld. Learning from customer defections. *Harvard Business Review*, September 1, 1996.

[19] M. Sharifzadeh and C. Shahabi. The spatial skyline queries. In *VLDB*, 2006.

[20] M. Sharifzadeh, C. Shahabi, and L. Kazemi. Processing spatial skyline queries in both vector spaces and spatial network databases. In *ACM. Trans. Database Syst. 34(3)*, 2009.

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

[22] Y. Tao, D. Papadias, and J. Zhang. Aggregate processing of planar points. In *EDBT*, 2002.

[23] Q. Wan, R. C.-W. Wong, I. F. Ilyas, T. Ozsu, and Y. Peng. Creating competitive products. *Proc. VLDB Endow.*, 2(1):898–909, 2009.

[24] R. C.-W. Wong, J. Pei, A W.-C. Fu, and K. Wang. Mining favorable facets. In *SIGKDD*, 2007.

[25] Z. Zhang, L. Lakshmanan, and A. K.H. Tung. On domination game analysis for microeconomic data mining. In *TKDD*, 2009.