On Domination Game Analysis for Microeconomic Data Mining

Zhenjie Zhang
National University of Singapore
and
Laks V.S. Lakshmanan
University of British Columbia
and
Anthony K. H. Tung
National University of Singapore

Game theory is a powerful tool for the analysis of the competitions among manufacturers in a market. In this paper, we present a study on combining game theory and data mining by introducing the concept of domination game analysis. We present a multidimensional market model, where every dimension represents one attribute of a commodity. Every product or customer is represented by a point in the multidimensional space, and a product is said to "dominate" a customer if all of its attributes can satisfy the requirements of the customer. The expected market share of a product is measured by the expected number of the buyers in the customers, all of which are equally likely to buy any product dominating him. A Nash Equilibrium is a configuration of the products achieving stable expected market shares for all products. We prove that Nash Equilibrium in such a model can be computed in polynomial time if every manufacturer tries to modify its product in a round robin manner. To further improve the efficiency of the computation, we also design two algorithms for the manufacturers to efficiently find their best response to other products in the market.

Categories and Subject Descriptors: H.2.8 [**DATABASE MANAGEMENT**]: Database Applications—*Data Mining*

General Terms: Algorithms

Additional Key Words and Phrases: Domination Game, Game Theory, Data Mining

1. INTRODUCTION

In a classic paper, Kleinberg et al. [Kleinberg et al. 1998a] advocated a utility oriented view of data mining driven by microeconomic considerations. They formulated data mining as a problem of optimizing an objective function that helps an organizational decision maker. They argued that a mined pattern is only useful to the extent to which it can be used in the decision-making process of the enterprise to increase utility.

Among the various example problem scenarios they used to illustrate their microeconomic problem formulation framework in [Kleinberg et al. 1998a] was the problem of market segmentation in a model of competition. An example of this is a so-called catalog war, wherein each of two[1] corporations I and II has a strategy space consisting of all possible catalogs with p pages to be mailed. Each corporation

[1]The problem can be generalized to $n > 2$ players.

has a different set of alternatives for each page. Each knows which alternative a customer will like and wants to maximize its payoff. The payoff for I for a given customer is $+1$ if s/he likes more pages from I's catalog than from II's catalog, -1 for vice versa, and is 0 upon a tie. Kleinberg et al. [Kleinberg et al. 1998a] motivated game-theoretic questions regarding the existence and computational complexity of Nash equilibria in this context.

Game theory [Osborne and Rubinstein 1994] is a powerful tool for modelling such competitions and had been used extensively for predicting the outcome of different business strategies. The concept of a Nash Equilibrium in game theory is a stable configuration of the manufacturers in the market, i.e., no one can enlarge his own profit by changing his "positioning" in the "configuration" alone. A Nash Equilibrium is called a randomized equilibrium if each player can play several strategies with some probability. In [Nash 1950], Nash proved that at least one randomized equilibrium exists for any such mixed strategies games. Nash however did not provide any algorithm for finding equilibria.

Recently, many computer scientists are trying to consider the problem from an algorithmic viewpoint. More specifically, they are concentrating on finding Nash equilibria on pure strategy games (henceforth referred to as pure Nash equilibria) in which each player plays only one strategy from the strategy set. The existence of Nash Equilibria in this case is not guaranteed [Osborne and Rubinstein 1994; Gottlob et al. 2003]. Examples of pure strategy games include congestion game [Fabrikant et al. 2004; Koutsoupias and Papadimitriou 1999; Papadimitriou 2001], exchange game [Deng et al. 2002; Devanur et al. 2002] and utility game [Vetta 2002].

Inspired by the seminal work of [Kleinberg et al. 1998a], in this paper, we consider a different game that we call *domination game*. We give an intuitive description of domination game below, using manufacturers and market share as a motivating example. Suppose there are t manufacturers, all of which manufacture (different instances of) one product. Each product has certain properties. For the purposes of this paper, we consider properties as attributes with values ranging over real numbers. Thus, each product is a point in a d-dimensional space, where d is the number of properties of products. There are n customers in the market. Each customer has certain preferences for the product they would like to buy, expressed as constraints of the form $A\,\theta\,r$ where A is an attribute, $r \in \mathcal{R}$ is a real number, and θ is some binary relationship such as '$<$'. A manufacturer's product satisfies a customer preference if it satisfies each of the constraints of the customer. Assume that a customer is equally likely to buy any one of the products manufactured by different manufacturers that satisfy his/her preferences. Each manufacturer would like to maximize their market share, i.e., the expected number of customers who will buy their product. To achieve this, a manufacturer would like to position its product (i.e., locate the point in the d-dimensional space) so as to satisfy as many customers as possible. That is, each manufacturer would like to *dominate* as many customers as possible. Each manufacturer has a constraint, called the *manufacturing hyperplane*, which constrains its product positioning to points on the hyperplane.

By a *configuration* of the domination game model, we mean a tuple of product

positionings by the t manufacturers, say $(p_1, ..., p_t)$, where p_i represents the positioning of product p by manufacturer m_i. A Nash equilibrium in this model is a configuration such that no manufacturer can increase its profit, i.e., its expected market share, by changing only its positioning. Thus, no manufacturer has a motivation to make a move from a Nash equilibrium. This property leads to a stable market, a situation desired by both the government and the industry (the set of manufacturers). Therefore, the efficient computation of such Nash Equilibria can be crucial in decision making processes, such as business negotiation.

In this context, we study the following questions. (1) Does a Nash equilibrium exist for the domination game and if so under what conditions? (2) What is the computational complexity of finding one? (3) Different Nash equilibria may fare very differently in terms of the number of customers covered (defined formally in Section 5. What is the worst (i.e., maximum) ratio between customer coverage of two different Nash equilibria? This question is important because for the industry as a whole, it is expected to cover as many customers as possible and the industry may be concerned about achieving at least a certain guaranteed minimum customer coverage. (3) Given a configuration, in order for a manufacturer to respond to it, it turns out it needs to answer a so-called "best response query" (defined in Section 3). Intuitively, this asks what is the best response by the manufacturer, for moving its product positioning along its hyperplane such that it will maximize its expected market share. How can we efficiently compute the answer to this query?

We prove that a Nash equilibrium exists in any instance of our model and further it can be found using an iterative algorithm. This algorithm keeps improving the personal expected market share for each individual manufacturer in a round robin manner from a randomly chosen initial configuration. We show that this algorithm converges at a Nash equilibrium in polynomial time.

Since the initial configuration is randomly chosen, the next question is whether the iterative algorithm can arrive at some arbitrarily bad equilibrium which has a much poorer customer coverage compared with some other equilibria. By showing that any such Nash Equilibrium is a 2-approximate maximum customer coverage solution, we dispell any such doubts.

The efficiency bottleneck for finding the Nash Equilibrium lies in determining the optimal positioning of a product against other products. We formalize this as the *best response query*. We designed two branch and bound algorithms for efficient answering of the best response query. We evaluate the performance of the algorithms through extensive experiments on synthetic data sets.

Our contributions are as follows.

—We give a precise formulation of the domination game model (Section 3).

—We prove that a Nash equilibrium always exists. We also show that given an arbitrary initial configuration, a Nash equilibrium that may be attained from it can be computed in polynomial time in the number of manufacturers and customers (Section 4).

—We prove that the ratio of customer coverage between any two Nash equilibria is at most two (Section 5).

—Our algorithm for computing Nash equilibrium relies on answering the best response query. While our algorithm in Section 4 for answering this is polynomial

in the number of customers and manufacturers, it is exponential in the number of product properties. We develop more efficient algorithms for this by exploiting pruning strategies (Section 6).

—The algorithms in Section 6 have the same asymptotic complexity as the naive algorithm for the best response query in Section 4. However, to measure their effectiveness in practice, we conducted an extensive set of experiments on synthetic data sets as well as a real data set. Our results show that iterative algorithm for finding a Nash equilibrium using our pruning strategies outperforms the algorithm based on the naive approach by two orders of magnitude (Section 7).

Section 2 discusses related work. In Section 8, we summarize our contributions and discuss promising directions for future research.

2. RELATED WORK

2.1 Game Theory

Game theory [Osborne and Rubinstein 1994] is an important topic in economics as it provides a strong tool for the analysis of competitive behavior in market. While Nash [Nash 1950] proved that any mixed strategies game must has at least one randomized equilibrium, his proof is not constructive, i.e., it does not suggest an algorithm for finding one. Here, we review several pure strategy games [Gottlob et al. 2003] where every player chooses to play an action in a deterministic manner. Note that Nash equilibria for pure strategy game (so called pure Nash Equilibria) are not guaranteed to exist in general [Osborne and Rubinstein 1994; Gottlob et al. 2003].

Congestion game [Fabrikant et al. 2004; Koutsoupias and Papadimitriou 1999; Papadimitriou 2001] stems from the competitive traffic problem. The traffic system is modelled as a graph. Every player has some commodities to be transported from one node in the graph to another. The delay of the commodities on every edge is a function of the total commodities flowing through the edge. The Nash Equilibrium is the set of paths for players where no one can reduce his own delay by switching to another path alone.

Exchange game [Deng et al. 2002; Devanur et al. 2002] happens in a market with some buyers and divisible goods. Every buyer has some cash and n different types of goods in hand. There is an individual utility function on the the goods for every buyer. The Nash Equilibrium in such a game is the set of prices of the goods in the market such that no buyer has an incentive to change its price alone. With such prices, the buyers can exchange the goods and cash to maximize their utility functions.

Utility game [Vetta 2002] is a general framework for a special type of games, in which every player tries to improve his own payoff, named utility. The authors of [Vetta 2002] show that, if the utility function on personal action satisfies several conditions, the Nash Equilibrium of the game can be found by iteratively improving personal utility by every player. However, the authors do not prove that their game can end in a polynomial number of iterations.

The first two types of games have been proved to have pure Nash Equilibria with polynomial complexity. But they cannot be used to model the domination game in this paper. The utility game is able to model our game but there is no known

algorithm for finding pure Nash Equilibria in polynomial time. As far as we know, (polynomial time) algorithms for finding Nash equilibria for domination games in a competitive setting have not been studied before.

2.2 Microeconomic View of Data Mining

Our work is in many ways inspired by the work in [Kleinberg et al. 1998a] which proposes to view data mining from a microeconomic perspective, i.e., the authors argue that the interestingness of knowledge being discovered should be measured by their utility to the organization. Various examples are given in [Kleinberg et al. 1998a] to illustrate utility oriented mining. Of these, profit oriented association discovery is studied in [Wang et al. 2002; Wong et al. 2003; Brijs et al. 1999], customer oriented catalog segmentation is investigated in [Kleinberg et al. 1998b; Ester et al. 2004], while [Yao 2003] explores data mining as sensitivity analysis.

Our work approaches competition games from a new perspective by using the number of dominated customers as the measure of utility of a decision. As far as we know, ours is the only work that explores computational issues of Nash equilibria for domination games.

2.3 Skyline Query and Dominance Relationship Analysis

Skyline query is a well-studied topic due to their importance in multi-criteria decision making and related applications [et. al. 1975; Steuer 1986]. The skyline operator was first introduced to the database community in [Börzsönyi et al.]. A large number of computation methods have been proposed for conventional relational databases. These methods can be divided into two general categories depending on whether they use indexes (e.g., *Index* [Tan et al. 2001], *Nearest Neighbor* [Kossmann et al.], *Branch and Bound Skyline* [Papadias et al. 2003]) or not (e.g., , *Divide and Conquer*, *Block Nested Loop*[Börzsönyi et al.], *Sort First Skyline*[Chomicki et al. 2003; Godfrey et al. 2005]. Moreover, skylines have been studied in the context of mobile devices [Huang et al. 2006], distributed systems [W.-T. Balke 2004], and unstructured [Wu et al. 2006], as well as structured networks [Wang et al. 2007]. Subspace skyline query has been studied extensively in [Yuan et al. 2005; Xia and Zhang 2006; Tao et al. 2007; Xia and Zhang 2006; Jin et al. 2007].

In addition, several papers focus on skyline computation when the dataset has some specific properties. [Chan et al. 2005] extends *Branch and Bound Skyline* for the case where some attributes take values from partially-ordered domains. [Chan et al. 2005] focuses on skyline processing for domains with low cardinality. [Chan et al. 2006] deals with high dimensional skylines. Finally, a number of interesting variants of the basic definition have been proposed. *Spatial skylines* [Sharifzadeh and Shahabi 2006] return the set of data points that can be the nearest neighbors of any point in a given query set. A *reverse skyline* [Dellis and Seeger 2007] outputs the records whose dynamic skyline contains a query point.

In [Li et al. 2006], some of the present authors proposed a method called DADA to efficiently answer various forms of dominance relationship queries while in [Li et al. 2007], spatial analysis are combined with dominance relationship analysis in order to identify profitable region in a market that can be easily dominated by some products. This work enhanced dominance relationship analysis with yet another tool based on game theory.

3. THE DOMINATION GAME MODEL

In this section, we introduce the basic concepts of domination game model and formally define domination games.

General Settings

In *Domination Game*, we assume there are n customers $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$, and t manufacturers $M = \{m_1, m_2, \ldots, m_t\}$ in the same market, both of which are only interested in consuming and producing one type of product, say p. Every manufacturer m_i is allowed to produce only one model of the product, denoted p_i and the qualities of the products made by the same manufacturer should be stable.

There are d attributes associated with each product, $\{A_1, A_2, \ldots, A_d\}$, all of which can be measured by a real number. Without loss of generality, we assume that a smaller value indicates better quality. A product p_i is said to have quality vector $(p_i[1], p_i[2], \ldots, p_i[d])$ where $p_i[k]$ is the quality of p_i on attribute A_k. Every customer c_j in the market has some requirement on the attributes, also represented by a vector $(c_j[1], c_j[1], \ldots, c_j[d])$. With such definitions, a product p_i satisfies (dominates) a customer c_j, if $p_i[k] \leq c_j[k]$ for $1 \leq k \leq d$. This amounts to assuming that: (a) on every attribute, a smaller value represents better quality and (b) the customer preference on every attribute A_k is expressed as a constraint $A_k \leq c_j[k]$, for some real number $c_j[k] \in \mathcal{R}$. When no confusion arises, we use p_i both to refer to the model of product p manufactured by m_i and to the point in the d-dimensional space corresponding to this model.

Profit Constraint Hyperplane

Since the resource for every manufacturer is limited and different, there is a profit constraint hyperplane h_i for every manufacturer m_i in the market i.e. it is not profitable to simply produce product with the highest quality just to attract customers. The hyperplane h_i thus divides the multidimensional space into two regions, a profitable region and a non-profitable one. Each manufacturer m_i can only position its product on its hyperplane [2]. Any profit constraint hyperplane satisfies the following two conditions.

PROPERTY 1. ***Intersection Testable***
Given a profit constraint hyperplane h_i and a rectangle with two diagonal corners at $(l[1], \ldots, l[d])$ and $(u[1], \ldots, u[d])$, we can test whether the cell intersect with part of h_i in constant time.

PROPERTY 2. ***Intersection Extensible***
Given a profit constraint hyperplane h_i and a point $(p[1], p[2], \ldots, p[d])$, we can find another point $(p[1], \ldots, p[k-1], p'[k], p[k+1], \ldots, p[d])$ exactly on h_i in constant time.

Intuitively speaking, the first property allows us to verify the intersection between a rectangle and the hyperplane. While the second property gives any algorithm

[2]Strictly speaking, the product can be positioned anywhere in the profitable region, but once the correct position on the hyperplane is determined, the best position in the profitable region that dominates the same set of customers can be easily determined.

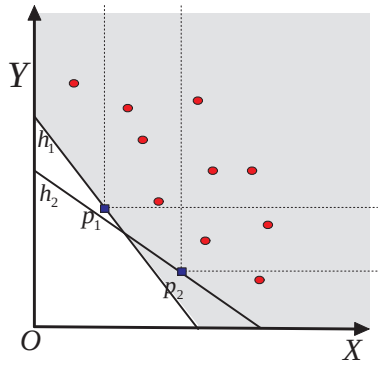

Fig. 1. Example for Products and Customers

an option to find the projection of p on attribute A_k. With second property, it is straightforward to test whether a given position in the space is above the hyperplane or not. Both of the properties will be used in the proofs and the algorithm designs later.

There are many hyperplanes satisfying the two properties above. Without loss of generality, in the rest of the paper, we will simply use a special type of hyperplane, all of which are actually some $(d-1)$-dimensional plane. Such a hyperplane, h_i, can be represented using the parameters $\{b_{i1}, \ldots, b_{id}, x_i\}$. Any product p_i on the hyperplane satisfies the function $\sum_{k=1}^{d} b_{ik} p_i[k] = x_i$. Here we restrict that $x_i > 0$ and $b_{ik} > 0$ for all i and k, reflecting the assumption that a manufacturer can improve the quality on one attribute only by sacrificing it over some other attributes. The profitable region is the set of points q for which $\sum_{k=1}^{d} b_{ik} q[k] \geq x_i$. It is not hard to prove that such hyperplanes immediately satisfy the two properties.

Commonly Dominated Customers

We define a *configuration* of the market as $\alpha = (p_1, p_2, \ldots, p_t)$, where m_i places its product at p_i, for all i. We assume the customers are all rational, i.e., they only buy the product satisfying their requirements. Furthermore, a customer buys only one product. If there is only one product p_i satisfying a customer c_j's requirements on all attributes, c_j will definitely buy p_i. If there are N products satisfying him, c_j will choose one product with equal probability $1/N$. For a configuration α, we denote by $D(p_i, r, \alpha)$ (resp., $D(\bar{p_i}, r, \alpha)$) the set of customers dominated by exactly r products including (resp., excluding) p_i. For a configuration α, we define the expected market share (or expected number of buyers of the product) of a manufacturer m_i as $S_i(\alpha) = \sum_{r=1}^{t} |D(p_i, r, \alpha)|/r$. The goal of the manufacturers in the domination game is to gain as much expected market share as possible.

Example

In Figure 1, we show an example of a market with two manufacturers and ten customers, where products and customers are denoted by square points and circle points respectively. The product has two attributes of interest to the customer

and every manufacturer only produces products on their corresponding manufacturing line. $\alpha = (p_1, p_2)$ is the current configuration of the market. The customers dominated by p_1 and p_2 are bounded by their corresponding rectangles. By the definition of expected market share above, $S_1(\alpha) = 4.5$ since there are three customers dominated only by p_1 and another three customers dominated by both p_1 and p_2. Similarly, we have $S_2(\alpha) = 3.5$.

Nash Equilibrium

A configuration α is said to be a Nash Equilibrium if $S_i(\alpha') \leq S_i(\alpha)$ for all $\alpha' = (p_1, p_2, \ldots, p_i', \ldots, p_t)$ for all $1 \leq i \leq t$. That is, manufacturer m_i cannot get any more expected market share if only m_i changes its product quality. We use $\alpha_{-i} = (p_1, ..., p_{i-1}, p_{i+1}, ..., p_t)$ to denote a configuration obtained from α by ignoring manufacturer m_i. Then the *best response query for* m_i *on* α_{-i}, is the query "find any positioning q_i for the product manufactured by m_i that maximizes $S_i(\beta)$, where $\beta = \alpha_{-i} \cup \{q_i\}$.[3] We let $B(\alpha_{-i})$ denote the set of answers to the best response query for m_i. Note that $B(\alpha_{-i}) \subseteq h_i$, i.e., the positionings are restricted to the hyperplane h_i. Then, a Nash Equilibrium $\alpha = (p_1, ..., p_t)$ must have the property that $p_i \in B(\alpha_{-i})$, for all i. To improve the readability of the paper, we summarize the notations in the paper in Table I.

4. THE EXISTENCE OF NASH EQUILIBRIUM

In this section, we study the existence and computability of Nash equilibria in domination games. The first question is whether a Nash equilibrium exist and if so under what conditions. Our first result answers this question in the affirmative.

THEOREM 1. *The domination game defined in the previous section always has a Nash equilibrium.*

Proof: Given a configuration $\alpha = (p_1, ..., p_t)$, define an $(n \times t)$ $(0,1)$-matrix $\mathbf{D}_\alpha$ as follows. $\mathbf{D}_\alpha[i,j] = 1$ iff p_j dominates c_i. Define an equivalence relation on configurations as follows: $\alpha \equiv \beta$ iff $\mathbf{D}_\alpha = \mathbf{D}_\beta$. It is easy to see that the number of equivalence classes is finite, even though the number of configurations is infinite. In fact, there are at most 2^{nt} equivalence classes.

Now, define a graph G with equivalence classes as nodes as follows. Thereto, define a potential function of a configuration as follows: $\Phi(\alpha) := \Sigma_{r=1}^{t} H_r |L(r, \alpha)|$, where $L(r, \alpha)$ is the disjoint union $D(p_i, r, \alpha) \cup D(\bar{p_i}, r, \alpha)$, for any i and $H_r = \Sigma_{j=1}^{r} 1/j$. That is, $L(r, \alpha)$ is the set of customers dominated by exactly r products. Note that the choice of i in the definition of $L(r, \alpha)$ is immaterial. The graph G contains an arc $([\alpha], [\beta])$[4] iff: (a) β is obtained from α by changing the positioning of any one product p_i and (b) $\Phi(\beta) > \Phi(\alpha)$. Notice that for equivalent configurations, the Φ-value is the same so this is well defined. Furthermore, whenever $\Phi(\beta) > \Phi(\alpha)$, there must exist $i, 1 \leq i \leq t$: $S_i(\beta) > S_i(\alpha)$. To see this, notice that $\Phi(\beta) - \Phi(\alpha) = \Sigma_{r=1}^{t} H_r(|L(r, \beta)| - |L(r, \alpha)|)$. $|L(r, \beta)| -$

[3]For convenience, we abuse notation and use set notations with configuration vectors. The meaning should be clear.

[4]$[\alpha]$ is the equivalence class containing α.

Notation	Description
h_i	profit constraint hyperplane for manufacturer m_i defined by the equation $\sum_{k=1}^{d} b_{ik}p_i[k] = x_i$
$D(p_i, r, \alpha)$	customers who are dominated by p_i and exactly $r-1$ other products in α
$D(\bar{p_i}, r, \alpha)$	customers who are dominated by exactly r products excluding p_i in α
$L(r, \alpha)$	customers who are dominated by exact r products in α
$\delta_i(r)$	$\lvert D(p_i', r, \alpha')\rvert - \lvert D(p_i, r, \alpha)\rvert$, change in the number of customers who are dominated by p_i and exactly $r-1$ products after one iteration
H_r	Harmonic series, $H_r = \sum_{k=1}^{r} 1/k$
$S_i(\alpha)$	the expected market share of product $p_i \in \alpha$
$W_i(\alpha)$	customers who are dominated by product $p_i \in \alpha$
$R(\alpha)$	customers who are dominated by at least one product in α
$e(p_i, \alpha)$	the effective dominating point of product $p_i \in \alpha$
$\lambda(C')$	A point $e = (e[1], ..., e[d])$ such that $e[i] = min_{c \in C'} c[i]$
$U(e)$	an upper bound point for a point e in the multidimensional space
$X(e, \Omega)$	the upper bound point for a point e and extensible dimension set ω
$DS(c_j, C')$	A set of dimensions along which a face of $\lambda(C')$'s dominating region is touched by c_j

Table I. Table of Notations

$|L(r,\alpha)| = |D(p_i,r,\beta)| + |D(\bar{p_i},r,\beta)| - (|D(p_i,r,\alpha)| + |D(\bar{p_i},r,\alpha)|) = (|D(p_i,r,\beta)| - |D(p_i,r,\alpha)|) + (|D(\bar{p_i},r,\beta)| - |D(\bar{p_i},r,\alpha)|)$. Of these, the second difference is exactly equal to $(|D(p_i,r+1,\alpha)| - |D(p_i,r+1,\beta)|)$. Denoting $\delta_i(r) = |D(p_i,r,\beta)| - |D(p_i,r,\alpha)|$, we then have $\Phi(\beta) - \Phi(\alpha) = \Sigma_{r=1}^{t} H_r(\delta_i(r) - \delta_i(r+1))$. By simple algebra, we can show that $\Sigma_{r=1}^{t} H_r(\delta_i(r) - \delta_i(r+1)) = \Sigma_{r=1}^{t} \delta_i(r)/r = S_i(\beta) - S_i(\alpha)$. We can see that $S_i(\beta) > S_i(\alpha)$ only when $\Phi(\beta) - \Phi(\alpha) > 0$.

Next, since $>$ is a strict partial order, it follows that G must be acyclic. Let $[\alpha]$ be any node of G with zero outdegree. By definition, any movement of any product position alone will not improve the potential of α. Since potential difference coincides with the difference in expected market share for the product that was moved, it follows that no product position can be moved alone on α so as to improve its expected market share, implying that α is a Nash equilibrium. □

Having settled the existence of Nash equilibria, the next question is what is the computational complexity of finding one. We present a simple intuitive iterative algorithm – Algorithm 1. It starts at a randomly chosen initial configuration α and repeatedly makes moves on behalf of each manufacturer in a round robin fashion. A

Algorithm 1 Iterative Best Response Algorithm

Input: Customers $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ and
Product Hyperplanes $H = \{h_1, h_2, \ldots, h_t\}$.
Output: A Nash Equilibrium.

```
1: Construct configuration α by randomly choosing p_i on h_i for all i.
2: while At least one manufacturer has a move do
3:   for every manufacturer m_i do
4:     Find any best response p_i' for m_i to α_{-i}.
5:     If a response was returned, then Update the configuration by α = α_{-i} ∪ {p_i'}.
```

move for m_i is a change in the positioning of p_i so that the new positioning improves m_i's expected market share, if such a position exists. To find this position, it invokes an algorithm for answering the best response query for m_i. We will discuss algorithms for answering this query later in this section as well as in later sections. Finally, the algorithm terminates when no manufacturer can make a move.

Before we establish any properties of the algorithm, we note the following easily verified identities.

$$L(r, \alpha) = \cup D(p_i, r, \alpha)$$

$$|L(r, \alpha)| = \sum_{p_i} \frac{|D(p_i, r, \alpha)|}{r}$$

Recall the potential function defined in the proof of Theorem 1: $\Phi(\alpha) = \Sigma_{r=1}^{t} H_r |L(r, \alpha)|$. In particular, recall that for any two configurations α β which differ only on the positioning of any one product, say p_i, $\Phi(\beta) - \Phi(\alpha) = S_i(\beta) - S_i(\alpha)$, where $S_i(\alpha)$ is the expected market share of m_i on configuration α. The following lemma shows that Algorithm 1 terminates in polynomial time.

LEMMA 1. *Algorithm 1 will stop after at most $O(nt \log t)$ iterations.*

Proof: By construction, before termination, in every step of the iteration, at least one manufacturer, say p_i, makes a move, i.e., change the positioning of p_i to increase its expected market share. Let α_{k-1} be the configuration at the beginning of iteration k and α' be the result after m_i moves its product position from p_i to, say p_i'. Then $S_i(\alpha') = \sum_r |D(p_i', r, \alpha')|/r > \sum_r |D(p_i, r, \alpha)|/r = S_i(\alpha)$. From the proof of Theorem 1, we know $S_i(\alpha') - S_i(\alpha) = \Phi(\alpha') - \Phi(\alpha)$, which is now > 0. So the value of the potential function is strictly increasing after every iteration.

$$\begin{aligned} & \Phi(\alpha') - \Phi(\alpha) \\ = & \sum_{r=1}^{t-1} H_r(\delta_i(r) - \delta_i(r+1)) + H_t \delta_i(t) \\ = & \sum_{r=1}^{t} \delta_i(r)/r \\ > & \ 0 \end{aligned}$$

Since $|L(r, \alpha)|$ can only be an integer and the minimum difference of two harmonic numbers is $1/t$, the increase of $\Phi(\alpha)$ after every iteration is at least $1/t$. Since

$\Phi(\alpha) \leq nH_t \leq n \log t$, the algorithm must stop after at most $O(nt \log t)$ iterations. □

What can we say about where the algorithm stops? Recall the graph G introduced in the proof of Theorem 1. Suppose the algorithm stops at configuration α. Clearly there can be no outgoing arc from $[\alpha]$. For if there were an arc $([\alpha], [\beta])$, by construction, the algorithm would find a move for some manufacturer and move from *alpha* to β.[5] It follows from the proof of Theorem 1 that α is a Nash equilibrium. We have:

LEMMA 2. *Algorithm 1 must stop at some Nash Equilibrium of the game.* □

Proof: The algorithm stops when every manufacturer cannot find a better position for its model p_i on h_i. By the definition of Nash Equilibrium, $\alpha = (p_1, p_2, \ldots, p_t)$ must be a Nash Equilibrium. □

So we know Algorithm 1 finds a Nash equilibrium in a polynomial number of steps in n and t. However, its overall complexity depends on the complexity of answering the best response query. In this section, we present a naive algorithm for showing this query can be answered in time polynomial in n and t. More efficient algorithms are the subject of Section 6.

Given the product position of all the remaining manufacturers, the best response query for m_i asks for an optimal product position for p_i. The most straightforward method for answering such query is to try all possible product positions on the hyperplane h_i and return the one with the largest expected market share. The following lemma shows how this can be done.

LEMMA 3. *The best response query can be answered in $O(n^d(d+n))$ time.*

Proof: Since there are n customers in the market, there are at most n different values of customer requirement on every attribute. So, we can split the whole multidimensional space into $(n+1)^d$ cells by simply using the values of the customers on every dimension as the splitting values. It is easy to verify that any two products in the same cell must dominate the same set of customers. Thus we can just pick a representative for each cell. Moreover, it takes at most $O(d)$ time to find a representative point in the cell as well as check the intersection between a cell and a hyperplane h_i by Property 1. We can find all the customers dominated by the representative in $O(nd)$ time, by comparing every customer requirement with the representative point on each attribute. So, the total computation for trying all the cells is at most $O(n^{d+1}(n+d))$. □

The following theorem immediately follows from Lemmas 1- 3.

THEOREM 2. *A Nash Equilibrium of the domination game can be found in polynomial time with respect to the number of manufacturers and customers.*

Proof: : By Lemma 1, there are at most $O(nt \log t)$ iterations. In every iteration, the algorithm needs to invoke the best response query t times. By Lemma 3, this query can be answered in $O(n^{d+1}(n+d))$ time. So, the total complexity of Nash Equilibrium problem in the market is at most $O(n^{d+2}(n+d)t^2 \log t)$, which is polynomial with respect to both manufacturer number and customer number. □

[5]To some configuration equivalent to β.

While the complexity of finding a Nash equilibrium is polynomial in the number of customers and products, it is exponential in the dimensionality d, i.e., number of product properties. Indeed the naive algorithm for the best response query is clearly exponential in d. In Section 6, we develop more efficient search algorithms and pruning strategies to improve the efficiency of answering the best response query, an important step of Algorithm 1.

5. CUSTOMER COVERAGE OF THE NASH EQUILIBRIA

Since there can be many different Nash Equilibria, an important question is whether Algorithm 1 is likely stop at an arbitrary "bad" configuration in which many customers could not find a satisfactory product, i.e., the overall number of customers dominated by any product in the equilibrium is small. Such an equilibrium is clearly undesirable for the industry which seeks to maximize the number of customers covered by the products via domination. We next prove that this is not the case for our model.

Let $R(\alpha)$ represent the set of the customers who are dominated by at least one product in the configuration α, i.e., $R(\alpha) = \cup_{r=1}^{t} L(r, \alpha)$. The *maximum customer coverage problem* is to find a configuration α^* such that $|R(\alpha^*)| \geq |R(\alpha)|$ for any other configuration α.

LEMMA 4. *Any Nash equilibrium is a 2-approximate solution to the maximum customer coverage problem.*

Proof: Assume $\alpha^* = (p_1^*, p_2^*, \ldots, p_t^*)$ is the optimal solution to the maximum customer coverage problem and $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_t\}$ is any Nash equilibrium. If $|R(\alpha)| < |R(\alpha^*)|/2$, we will show that there is at least one manufacturer m_i which can improve its expected market share by moving only its product from $p_i \in \alpha$ to p_i^*, contradicting the fact that α is a Nash equilibrium.

It is not difficult to verify that $|R(\alpha)| = \sum_{i=1}^{t} S_i(\alpha)$ and $|R(\alpha^*)| = \sum_{i=1}^{t} S_i(\alpha^*)$. Remove all the customers in $R(\alpha)$ from $\mathcal{C}$ and consider the new market with the same products but with customer set $\mathcal{C} - R(\alpha)$. Let the expected market share of m_i with configuration α^* on the new market be $S_i'(\alpha^*)$. Then, $\sum_{i=1}^{t} S_i'(\alpha^*) = |R(\alpha^*)| - |R(\alpha)| > |R(\alpha)| = \sum_{i=1}^{t} S_i(\alpha)$. By the pigeon hole principle, there is at least one m_i having $S_i'(\alpha^*) > S_i(\alpha)$. This means that m_i can achieve better expected market share by moving from p_i to p_i^* even without those customers in $R(\alpha)$. This was to be shown. □

The following is an example to show that the bound in Lemma 4 is tight asymptotically. Consider a market with m manufacturers with identical profit constraint hyperplanes. We can construct a customer set with t groups of customers. The groups are so far away from each other that any product can dominate only one group. Let there be t customers in the first group, while there is only one customer in the remaining groups. Then, an obvious Nash equilibrium is a configuration where all products try to dominate the first group, which covers t customers overall. However, the maximum customer coverage can be achieved if every product covers a single group giving a total cover of $2t - 1$ customers. The ratio $(2t - 1)/t$ approaches 2 as $t \to \infty$.

THEOREM 3. *For any two Nash Equilibria α_1 and α_2, $\frac{1}{2} \leq \frac{|R(\alpha_1)|}{|R(\alpha_2)|} \leq 2$.*

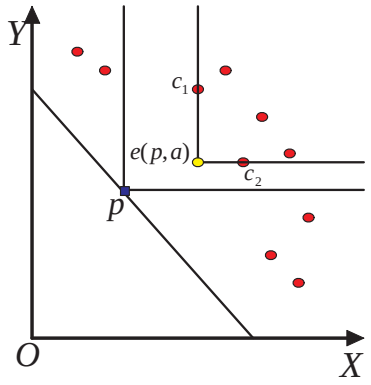

Fig. 2. Example for effective dominating point

Proof: Assume α^* is a solution to the maximum customer coverage problem. By Lemma 4, $2|D(\alpha_2)| \geq |D(\alpha^*)|$. So, $|D(\alpha_1)| \leq |D(\alpha^*)| \leq 2|D(\alpha_2)|$. The other side can be proved by simply exchanging the position of α_1 and α_2 in the proof above. □

From Theorem 3, we know that by choosing a random initial configuration and running Algorithm 1 once, we can get a Nash equilibrium whose customer coverage is at least half that of the best Nash equilibrium in terms of customer coverage.

6. ALGORITHMS FOR BEST RESPONSE QUERY

The naive algorithm for best response query, while polynomial in n and t, is exponential in d. In this section, we propose two new best response algorithms based on breadth-first and depth-first search in the customer lattice space. Some pruning techniques are also developed to improve the efficiency of the algorithms.

6.1 Lattice Search Algorithms

6.1.1 *Effective Dominating Point of a Product.* We assume the customer set $\mathcal{C} = \{c_1, ..., c_n\}$ and manufacturer set $M = \{m_1, ..., m_t\}$, with m_i manufacturing p_i.

DEFINITION 1. *Let $\alpha = (p_1, ..., p_t)$ be a configuration. Let $W_i(\alpha) \subseteq \mathcal{C}$ be the set of customers dominated by $p_i \in \alpha$. Then the* effective dominating point *of p_i is a point $e(p_i, \alpha) = (e[1], e[2], \ldots, e[d])$ such that $e[k] := \min_{c_j \in W_i(\alpha)} c_j[k]$, $1 \leq k \leq d$.*

In Figure 2, we show an example of an effective dominating point, $e(p, \alpha)$ for the point p. From the figure, we can see that the effective dominating point is actually the top-most right-most position in the multidimensional space which can dominate exactly same set of customers as the given point p.

DEFINITION 2. *Let $e = (e[1], ..., e[d])$ be an effective domination point. Then the domination region of e is the subset of the d-dimensional product attribute space, such that every point in it is dominated by e.*

In Figure 2, the domination region of $e(p, \alpha)$ is right top part of the 2-dimensional space bounded by two lines from $e(p, \alpha)$. The following lemma shows an important property of effective dominating points.

LEMMA 5. *Given a product p_i and its effective dominating point $e(p_i, \alpha)$, there is at least one customer point on each face of the region dominated by $e(p, \alpha)$.*

Proof: Follows from Definition 1. □

In Figure 2, customer point c_1 and c_2 are the customer points on the two faces of domination region of $e(p, \alpha)$ respectively.

Let $C \subseteq \mathcal{C}$ denote a subset of customers in the game, and $\lambda(C)$ be the top-most right point dominating all customers in C. We have:

LEMMA 6. *For any product point p located on the hyperplane h_i, $e(p, \alpha) = \lambda(C)$ for some $C \subseteq \mathcal{C}$ such that $|C| \leq d$.*

Proof: Let $W_i(\alpha)$ be set of customers dominated by p. For each dimension A_k, we pick one of the customers from $W_i(\alpha)$, with the smallest value along the dimension and add the customer into C. C thus contains at most d customers and $\lambda(C)$ is definitely able to dominate all customers in $W_i(\alpha)$, making it equal to $e(p, \alpha)$. □

When a customer has the smallest values on several different dimensions, the size of set C will be smaller than d. We note that given any customer subset C of size no larger than d, it is not necessary to have a plausible product point p on the hyperplane h_i to satisfy $e(p, \alpha) = \lambda(C)$. For a hyperplane $\Sigma_{k=1}^{d} b_{ik} p_i[k] = x_i$, we say a point q is above h_i if $\Sigma_{k=1}^{d} b_{ik} q[k] > x_i$. Otherwise, it is below h_i. We can show:

LEMMA 7. *If p_i^* is the best response for m_i on a configuration α, any product positioning $\hat{p}_i$ above h_i and dominating $e(p_i^*, \alpha)$ can achieve the same expected market share as p_i^*.* □

The proof of the lemma is straightforward. In Figure 2, for example, any product dominating $e(p, \alpha)$ dominates at least the five customers dominated by $e(p, \alpha)$.

Lemma 7 shows that given the effective dominating point of an optimal response p_i, it is trivial to find a product achieving the same expected market share on the hyperplane h_i. This implies a new best response searching method. We can try all combinations of customers with size no larger than d, find the combination C' which give the highest expected market share based on $\lambda(C')$, and simply use the projection of $\lambda(C')$ on h_i as the final optimal result.

6.1.2 *Customer Search Tree.* Based on the analysis earlier, we propose a new concept, called *Customer Search Tree*, for finding a customer combination that gives the maximum expected market share.

Assuming an arbitrary global order on the customers, $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$, subsets of $\mathcal{C}$ can be represented as strings in the obvious way. We assume this below. Given two customer subsets C_q and C_r, C_r is the extension of C_q if $C_r = C_q \cdot c$ for some customer $c \notin C_q$.

A search tree structure, $L(C)$, is constructed based on the above definitions. Every node in the search tree represents a customer set $C_q \subseteq \mathcal{C}$ of size no larger than d. Without ambiguity, we use C_q to denote both a customer set and the node in the search tree that represents it. There is a directed edge from C_q to C_r, iff C_r is an extension of C_q. The nodes and edges form a tree with root at the empty

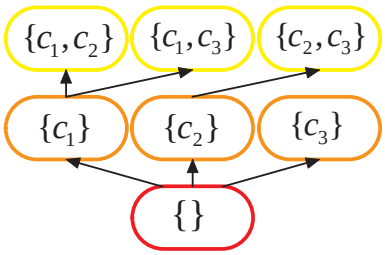

Fig. 3. Example for customer search tree

Algorithm 2 Breadth/Depth First Search (Customer Search Tree L, Hyperplane h_i)

1: **while** there is at least one node in L not pruned or not visited **do**
2: Find the next search tree node N in breadth/depth first order
3: Construct effective dominating point $e = \lambda(N)$
4: **if** e is above h_i using Property 2 of profit constraint hyperplane **then**
5: Compute the expected market share of e (section 6.2)
6: Update the best response if e is better than current solution
7: Prune the children of N (section 6.3)
8: Project the best e onto h_i and return the result

node. We say a node C_r is the descendant of C_q if there exists a directed path from C_q to C_r in the search tree.

Thus, the searching process over all the combinations of customers can be accomplished by moving through the search tree nodes along the edges.

In Figure 3, we present an example of the customer search tree, on a data set with 3 customers in 2 dimensional space. $\{c_1, c_2\}$ and $\{c_1, c_3\}$ are extensions of $\{c_1\}$ by definition.

LEMMA 8. *If C_r is a descendant of C_q in $L(C)$, the domination region of $\lambda(C_r)$ must cover the domination region of $\lambda(C_q)$.*

Proof: If $e_1 = \lambda(C_r)$ and $e_2 = \lambda(C_q)$, $e_1[k] = \min_{c_j \in C_r} c_j[k]$ and $e_2[k] = \min_{c_j \in C_q} c_j[k]$. Since $C_q \subset C_r$, $e_1[k] \leq e_2[k]$. So, the domination region of $\lambda(C_r)$ must cover that of $\lambda(C_q)$. □

Given a node C_q and one of its descendants C_r, we say C_r extends C_q on dimension A_k, if $e_1[k] < e_2[k]$ where $e_1 = \lambda(C_r)$ and $e_2 = \lambda(C_q)$.

6.1.3 *Searching on Customer Search Tree.* To find the customer combination C' such that $\lambda(C')$ can achieve optimal expected market share, we employ two different searching strategies over the search tree $L(C)$, namely breadth-first search and depth-first search (see Algorithm 2). Starting at the root of $L(C)$, the algorithms iterate through the nodes in the search tree in different order, but have the same operations at any single node C'. They compute $e = \lambda(C')$ and calculate the expected market share at e based on the configuration of other manufacturers. If e is better than the current best response, e will become the best response. Some pruning is conducted to reduce the number of nodes that must be visited. After all

the nodes that are reachable from the root has been visited,
the algorithm projects the optimal position e onto h_i to obtain the final result as the best response.

In the rest of the section, we will discuss the detail on how the expected market share at $\lambda(C')$ can be efficiently computed (section 6.2), as well as how we can prune children node which definitely cannot yield a better response (section 6.3).

6.2 Computing Expected Market Share Based on R-Tree

Given a node C' in the search tree with k customers, $e = \lambda(C')$ is computed by retrieving the smallest value of the k customers on every dimension. If e is below the profit constraint hyperplane of the manufacturer, then e is not a valid position since it violates the constraint. If e is above the hyperplane, we need to efficiently determine the expected market share that e will bring.

To efficiently support such an operation, we use a modification of the R-Tree index [Guttman 1984], called aggregation R-tree [Jürgens and Lenz 1998; Lazaridis and Mehrotra 2001; Papadias et al. 2001]. Every point in the R-Tree is a customer point in the data set. A weight is assigned to every point in the R-Tree representing the expected market share it can provide if a product dominates it. If there are already s products dominating a customer c_j, the weight of c_j is $1/(s+1)$. Thus, the expected market share $\lambda(C)$ can be computed by summing up the weight of all the customers that it can dominate. Since the details of such operations are well studied in [Jürgens and Lenz 1998; Lazaridis and Mehrotra 2001; Papadias et al. 2001], we suppress further details here. By some analysis of the studies on such indexing structures, the complexity of the computation is about $O(\log n)$, where n is the number of customers indexed. To update the weights of the nodes because of the changing number of dominators, the algorithm needs to iterate and update every node in the indexing tree, after every iteration. The cost of such update is affordable, since the update time depends on the number of nodes in the indexing tree, typically $O(n \log n)$, which is much cheaper than the time spent on iterations over the customer search tree.

6.3 Pruning Strategies

In order to reduce the number of nodes being visited in the search tree, we propose two pruning conditions which will be applied in Line 7 of Algorithm 2.

6.3.1 *Boundary Condition.* Given an effective dominating point e, Lemma 5 has already shown that there is at least one customer on every face of its domination region. Here, provide a boundary condition which constrains the nodes visited in the search tree.

We say a customer set $C \subseteq \mathcal{C}$ satisfies the *boundary condition*, if every customer $c \in C$ is the unique point with the minimum value in some dimension. That is, there is at least one dimension, the boundary on which is only decided by this customer. We have the following lemma.

LEMMA 9. *Given a node C not satisfying the boundary condition, there exists another node $D \subseteq C$ satisfying boundary condition, such that $\lambda(D) = \lambda(C)$. For any descendant C' of C, there exists a descendant node D' of D, such that $\lambda(D') = \lambda(C')$.*

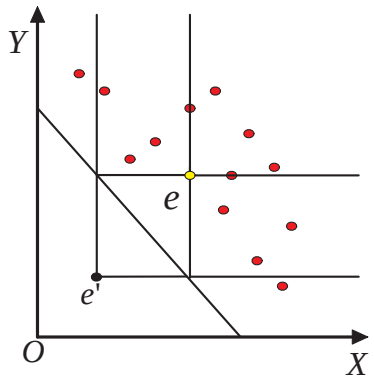

Fig. 4. Example of Upper Bound

Proof: Assume C has k different customer points $c_1, \ldots, c_k$, and without generality c_k is the only point not on any domination face of $\lambda(C)$. Then, by removing c_k, we have a node D with $k-1$ points, $\lambda(D) = \lambda(C)$ since the removal of c_k can not increase the minimum value of D on any dimension. If C' is a descendant of C by combining C with another customer subset F, that is $C' = C \cup F$, we can find the corresponding descendant D' of D by combining D and F. It is easy to verify that $\lambda(D') = \lambda(C')$. □

The value of Lemma 9 is that it yields the following property of the customer search tree, as Algorithm 2 visits the nodes of the search tree. Suppose the current node does not satisfy the boundary condition. Then none of its descendants satisfies the boundary condition either. This observation will be used later to develop a pruning strategy.

6.3.2 *Upper Bound Pruning.* By extending from a customer set C to any of its descendant C', the expected market share will definitely be non-decreasing since the domination region of $\lambda(C)$ will grow by Lemma 8. However, we will show that it is possible to derive an upper bound on the expected market share of $\lambda(C')$ based on the current information in C.

LEMMA 10. *In computing the best response for manufacturer m_i with profit constraint hyperplane h_i given by $\sum_{k=1}^{d} b_{ik} p_i[k] = x_i$, let $e = \lambda(C) = (e[1], e[2], \ldots, e[d])$ for a node C. Let C' be any descendant of C in the customer search tree and let $e' = \lambda(C') = (e'[1], \ldots, e'[d])$. Then for any dimension k, $e'[k] \geq e[k] - (\sum e[k] b_{ik} - x_i)/b_{ik}$.*

Proof: Since any descendant node C' must contain more customers than C, the boundary of $\lambda(C')$ on any dimension cannot be larger than that of $\lambda(C)$, i.e. $e'[k] \leq e[k]$. If $e'[k] < e[k] - (\sum_j e[j] b_{ij} - x_i)/b_{ik}$ for some k, we have $\sum e'[k] b_{ik} < e[k] - (\sum_j e[j] b_{ij} - x_i) + \sum_{j \neq k} e[j] b_{ij} = x_i$, making the point $\lambda(C')$ below the hyperplane. □

Thus, for a point $e = \lambda(C)$ above h_i, we define the upper bound point $U(e) = (e^u[1], \ldots, e^u[d])$ with $e^u[k] = e[k] - (\sum e[k] b_{ik} - x_i)/b_{ik}$. It is obvious that $U(e)$ dominates $\lambda(C')$ where C' is any descendant of C. Thus, the expected market

Algorithm 3 ExtensibleUpperBoundTest (customer subset C', profit constraint hyperplane h_i, R-Tree root r, threshold θ)

```
1: e = λ(C')
2: for every dimension combination set ω do
3:    if ω is extensible by Definition 11 then
4:       Construct the upper bound point p' = X(e, ω)
5:       if DomCount(p', r) > θ then
6:          Return without pruning
7: Prune all the children of C'
```

share of $U(e)$ is an upper bound on the expected market share achievable on the descendants of C.

Note that Lemma 10 works only when the hyperplane follows the constraint $\sum_{k=1}^{d} b_{ik} p_i[k] = x_i$. It is not hard to extend it to any hyperplane satisfying Property 2 in Section 3, by which the minimum possible value on each dimension can be calculated in constant time.

In Figure 4, we show an example of the upper bound computation in 2-dimensional space. The point e' is $U(e)$ for the point e above the hyperplane.

From the figure, we can see that the upper bound on the market share by directly expanding along all dimensions can be very loose. Fortunately, we can tighten the bound by combining Lemma 9 with the boundary condition since extension on all dimensions can violate the boundary condition in Lemma 9.

Assume $C = \{c_1, c_2, \ldots, c_k\}$ is the node being visited in the search tree. Let $DS(c_i, C) = \{A_{o_1}, A_{o_2}, \ldots\}$ $1 \leq o_k \leq d$, denoting the set of dimensions such that c_i is on the face of domination region for $\lambda(C)$ along these dimensions. In Figure 5, for example, there are three dimensions $\{x, y, z\}$. If $C = \{c_1, c_2\}$, where $c_1 = (1, 1, 2)$, $c_2 = (2, 2, 1)$ and $\lambda(C) = e = (1, 1, 1)$, then $DS(c_1, C) = \{x, y\}$ and $DS(c_2, C) = \{z\}$ because c_1 has minimum values on dimension x and y while c_2 has minimum value on dimension z.

DEFINITION 3. *A set of dimensions ω is an extensible dimension set for a customer subset C if $|DS(c_j, C) - \omega| > 0$ for every $c_j \in C$.*

In the definition above, $DS(c_j, C) - \omega$ is the set of the dimensions in $DS(c_j, C)$ but not in ω. Recall the example in Figure 5. $\{x\}$ is an extensible dimension set for $C = \{c_1, c_2\}$, since $DS(c_1, C) - \{x\} = \{y\}$ and $DS(c_2, C) - \{x\} = \{z\}$. However, $\{x, z\}$ is not, since $DS(c_2, C) - \{x, z\} = \emptyset$. The extensible dimension set has the property shown in the next lemma.

LEMMA 11. *If a descendant C' of C extends along some dimensions which are not in the extensible dimension set ω, C' must violate the boundary condition.* □

Proof: If a dimension set ω is not an extensible dimension set, there is at least one point $c_j \in C'$ that $|DS(c_j, C') - \omega| = 0$. Then, c_j can not be on any face of the domination region of C'', which contradicts boundary condition. □

By the last lemma, the descendants of C can only extend the dominating region of $\lambda(C)$ on those extensible dimension sets. Given an extensible dimension set ω and $e = \lambda(C)$, we propose the extensible upper bound point $X(e, \omega) = (e^{\omega}[1], e^{\omega}[2], \ldots, e^{\omega}[d])$, s.t. $e^{\omega}[k] = e[k]$ if $k \notin \omega$ and $e^{\omega}[k] = e^{u}[k]$ otherwise. It is

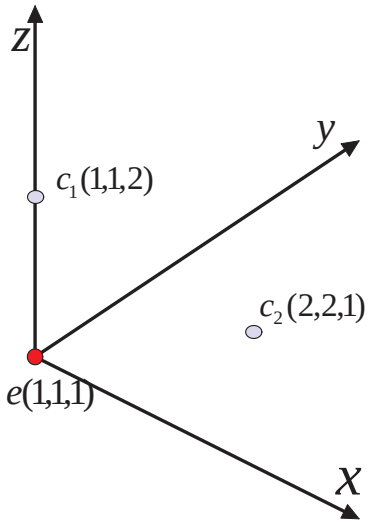

Fig. 5. Example of extensible dimension set

clear that the expected market share of $X(e, \omega)$ must be the upper bound of the descendants of C by extending on ω.

Thus we propose the extensible upper bound computation algorithm in Algo 3. In this algorithm, we iterate through all the possible $2^d - 1$ dimension combinations, and test the upper bound only when the dimension combination is extensible. For example, in Figure 5, there are only two extensible dimension sets, $\{x\}$ and $\{y\}$, for $C' = \{c_1, c_2\}$. For all extensible dimension sets, if the expected market share of the corresponding extensible upper bound point cannot be larger than threshold θ (the expected market share of the current best solution), the algorithm will prune all the children of C'. Since d is usually much smaller than n, the cost of iterating all dimension sets is typically small.

6.4 Discussion on Discretization

Another possible optimization for best-response query is the discretization over the dimensions with continuous values. Given a d-dimensional unit space $[0, 1]^d$, a customer c_j is represented by a vector $(c_j[1], c_j[2], \ldots, c_j[d])$ in the original space. Given the specified discretization parameter Δ, the original vector can be transformed to a new vector $(\lfloor c_j[1]/\Delta \rfloor \Delta, \ldots, \lfloor c_j[d]/\Delta \rfloor \Delta)$. By such transformation, the number of possible values on any dimensions is no more than $\lceil 1/\Delta \rceil$, which can be regarded as a constant number. Best response query can thus be answered by employing some existing indexing technique, such as DADA-tree [Li et al. 2006]. Although beyond the scope of this paper, note that the efficiency of best response query can be dramatically improved, as is shown in [Li].

However, the customer data set has some loss on the detailed information of the customers, which may lead to some sub-optimal result of the best response query. The convergence result of the domination game depends on the exact solution

of best response query, and the game may not converge if any approximate best response is applied instead. Based on this observation, we only focus on testing the algorithms outputting exact best response in the following experimental section.

7. EXPERIMENTS

We carried out an experimental study to verify the properties and test the performances of the algorithms proposed in this paper. The programs are compiled by gcc 3.4.3 and run on IBM x255 server with four Intel Xeon MP 3.0 GHz CPU, 18G DDR memory and six 73.4GB Ultra320 SCSI hard disks.

In the experiments, we generate different types of synthetic data sets. All of the points in the data sets are in $[0, 1]^d$, where d is the dimensionality. The values on every dimension are all float numbers. The size of the customer set ranges from 1000 to 10000. The dimensionality of the market varies from 2 to 4, while there are at least 2 and at most 9 corporations in the market. The default setting of the parameters above is 1000 customers, 3 dimensions and 2 corporations. There are four optional distributions of customer set, including Anti-Correlated (A), Correlated (C), Independent (I) and Clustered (L). In anti-correlated customer set, the increase of the requirement on one dimension will lead to some decrease on the others. In correlated customer set, a customer having high requirement on one dimension is likely to have similar high standards on the others. In independent customer set, all dimension are independent and obeys uniform distribution on the range. In clustered data set, the points are generated from 5 different Gaussian distributions in the space.

We also employ a real data based on the review comments collected from a hotel review web site TripAdvisor [6]. The users of the web site are supposed to rate the hotels on 7 different attributes, as well as an overall score. All of the ratings are integers between 1 and 5. However, only four attributes, including cleanliness, value, service and rooms, are used in our data since most of users rate on all of them. Based on the assumption that a good overall score is given only when the hotel satisfies all requirements of the user, we transform the review data set to requirement data set by using the review tuples with overall scores no smaller than 4. After crawling 50 hotels in Sydney, we retrieve 997 valid tuples, each attribute of which is normalized to some real number between 0 and 1. These tuples are regarded as customers in our experiments.

The profit constraint hyperplanes for the manufacturers are generated by uniformly choosing the parameters b_{ij} and x_i in the range of $[0.8, 1.2]$, on synthetic data sets as well as on real data set.

In our experiments, we focus on the efficiency of the algorithms proposed in this paper. In the rest of the section, we use **NAIVE**, **DFS** and **BFS** to denote the iterative best response algorithm with naive, depth first search and breadth first search as the underlying best response computation respectively. In Table II, we first compare the speed of NAIVE, DFS and BFS on 3 dimensional space with 1000 customers and 3 manufacturers. NAIVE is much slower than DFS and BFS on anti-correlated, correlated and independent data sets. This result indicates that lattice search is a great improvement on the naive search scheme for best response

[6] www.tripadvisor.com

Data Type	A	C	I	L
NAIVE	5326	472	2163	31
DFS	130	27	69	23
BFS	132	29	73	25

Table II. Speed comparison of NAIVE, DFS and BFS in seconds

query. Since NAIVE is not scalable to any larger or higher dimensional data sets, we only compare our DFS and BFS algorithm in the rest of the experiments. On clustered data set, however, naive algorithm is not so bad because the searching space is not large, which constrains the pruning ability of DFS and BFS.

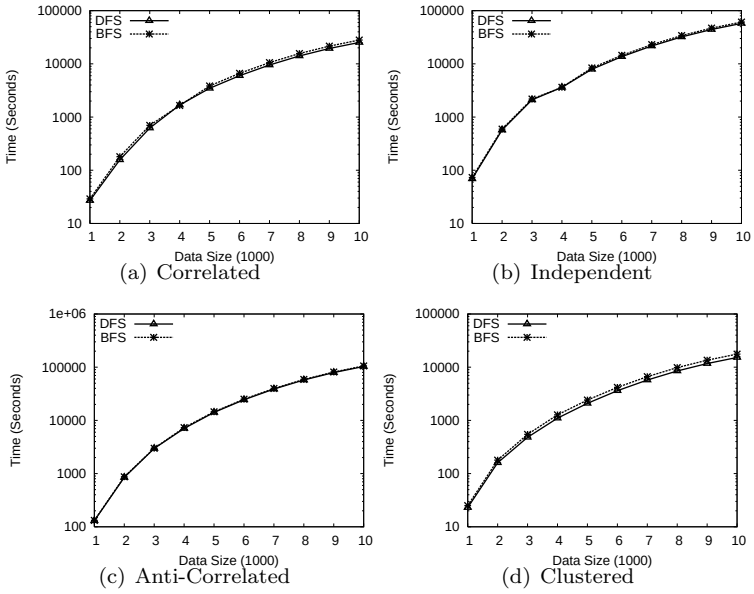

Fig. 6. Tests on varying data size

In Figure 6, we show the computation cost of DFS and BFS on varying data size from 1000 to 10000. The time spent by the two algorithms both increases polynomially with the expansion of the data set. DFS algorithm is a little better than BFS in all types of data sets, especially on clustered data sets.

As is shown in Figure 7, the computation times of DFS and BFS are still exponential in the dimensionality d. However, by some simple linear regression on the curves, we can verify that the empirical time complexity of DFS and BFS on dimensionality d is proportional to 100^d. Since there are $n = 1000$ points in the data set, the lattice search algorithm on DFS and BFS still achieve some improvement on NAIVE.

We also conduct some experiments on varying the number of manufacturers in the market. The result in Figure 8 shows that the linear increase of computation time implies that the iteration number of the process also increases linearly. When

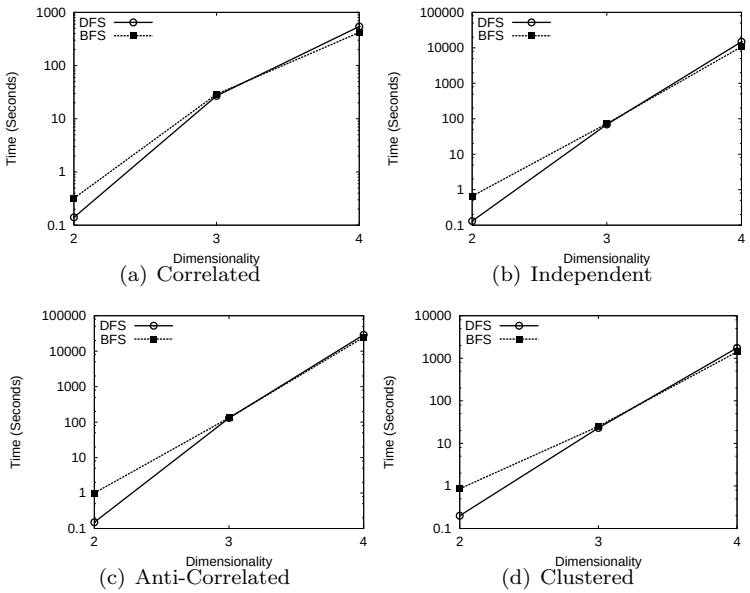

Fig. 7. Tests on varying dimensionality

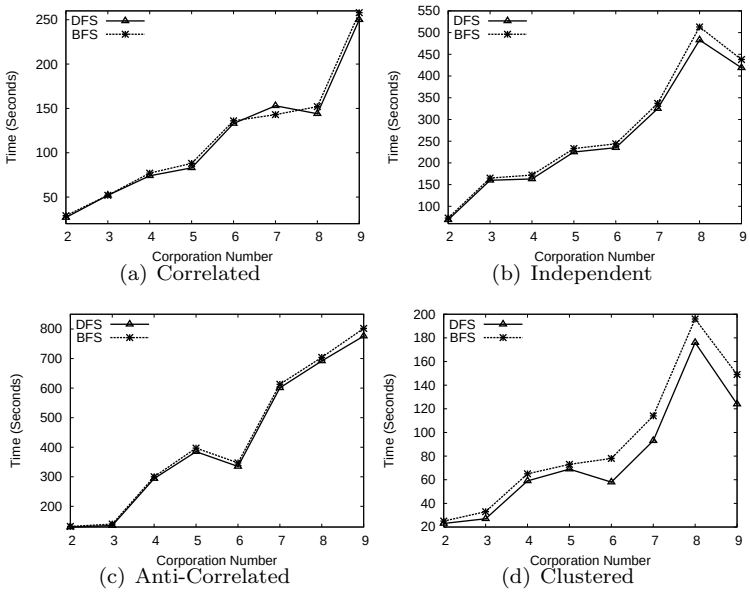

Fig. 8. Tests on varying manufacturer number

there are 9 manufacturers, the computation cost even decreases on independent and clustered data. This is because the data are so scattered or grouped in these

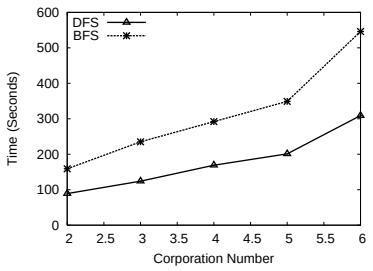

Fig. 9. Tests on varying manufacturer number with TripAdvisor data set

data sets that the manufacturers can easily target a small group of customers in the first iteration and find it hard to attract new customers with old customers kept. Thus, the stability of the market can be achieved easily even when there are more competitions in the same market.

Finally, the performance comparisons on the two algorithm over TripAdvisor data set is presented in Figure 9. This group of test confirms the advantage of DFS over BFS on efficiency. DFS algorithm is almost two times faster than BFS algorithm when the number of manufacturers increases from 2 to 6.

8. CONCLUSION AND FUTURE WORK

In this paper, we proposed domination games for modelling competition among manufacturers of a product for maximizing their expected market share. This work was motivated by the seminal work of [Kleinberg et al. 1998a] motivating data mining from a microeconomic, utility oriented perspective. For domination games, we show that a Nash equilibrium always exists and showed that it can be computed in polynomial time in the number of customers and products. The algorithm is exponential in the number of product properties. We developed speeding up stratgies for answering the best response query which forms the backbone of the algorithm for finding equilibria. We showed that in terms of customer coverage any Nash equilibrium is at most two times worse than the best one. Characterizing the complexity of the best response query in terms of dimensionality is open. Extension of this framework where a customer may buy more than one model, or a product dominates a customer if it can satisfy the customer on k out of d dimensions, or there are multiple product types, or a customer buys a product with a probability determined by the proximity of the product to his preferences are all interesting directions for future work.

REFERENCES

Börzsönyi, S., Kossmann, D., and Stocker, K. The skyline operator. In *ICDE'01*.

Brijs, T., Swinnen, G., Vanhoof, K., and Wets, G. 1999. Using association rules for product assortment decisions: A case study. In *KDD*. 254–260.

Chan, C.-Y., Eng, P.-K., and Tan, K.-L. 2005. Stratified computation of skylines with partially-ordered domains. In *SIGMOD*. 203–214.

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

Chomicki, J., Godfrey, P., Gryz, J., and Liang, D. 2003. Skyline with presorting. In *ICDE*. 717–816.

Dellis, E. and Seeger, B. 2007. Efficient computation of reverse skyline queries. In *VLDB*. 291–302.

Deng, X., Papadimitriou, C. H., and Safra, S. 2002. On the complexity of equilibria. In *STOC*. 67–71.

Devanur, N. R., Papadimitriou, C. H., Saberi, A., and Vazirani, V. V. 2002. Market equilibrium via a primal-dual-type algorithm. In *FOCS*. 389–395.

Ester, M., Ge, R., Jin, W., and Hu, Z. 2004. A microeconomic data mining problem: customer-oriented catalog segmentation. In *KDD*. 557–562.

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

Fabrikant, A., Papadimitriou, C. H., and Talwar, K. 2004. The complexity of pure nash equilibria. In *STOC*. 604–612.

Godfrey, P., Shipley, R., and Gryz, J. 2005. Maximal vector computation in large data sets. In *VLDB*. 229–240.

Gottlob, G., Greco, G., and Scarcello, F. 2003. Pure nash equilibria: hard and easy games. In *TARK '03: Proceedings of the 9th conference on Theoretical aspects of rationality and knowledge*. 215–230.

Guttman, A. 1984. R-trees: A dynamic inde structure for spatial searching. In *SIGMOD'84*. 47–57.

Huang, Z., Jensen, C. S., Lu, H., and Ooi, B. C. 2006. Skyline queries against mobile lightweight devices in MANETs. In *ICDE*. 66.

Jin, W., Tung, A., Ester, M., and Han, J. 2007. On efficient processing of subspace skyline queries on high dimensional data.

Jürgens, M. and Lenz, H.-J. 1998. The ra*-tree: An improved r-tree with materialized data for supporting range queries on olap-data. In *DEXA Workshop*. 186–191.

Kleinberg, J., Papadimitriou, C., and Raghavan, P. 1998b. Segmentation problems. In *STOC*.

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

Kossmann, D., Ramsak, F., and Rost, S. Shooting stars in the sky: an online algorithm for skyline queries. In *VLDB'02*.

Koutsoupias, E. and Papadimitriou, C. H. 1999. Worst-case equilibria. In *STACS*. 404–413.

Lazaridis, I. and Mehrotra, S. 2001. Progressive approximate aggregate queries with a multi-resolution tree structure. In *SIGMOD Conference*. 401–412.

Li, C., Ooi, B. C., Tung, A. K., and Wang, S. 2006. Dada: a data cube for dominant relationship analysis. In *SIGMOD*.

Li, C., Tung, A. K. H., Jin, W., and Ester, M. 2007. On dominating your neighborhood profitably. In *VLDB*. 818–829.

Li, X. Honor year report: Rtree-based dada for dominant relationship queries and domination game model. *https://dl.comp.nus.edu.sg/dspace/handle/1900.100/2487*.

Nash, J. F. 1950. Equilibrium points in n-person games. *Proc. of National Academy of Sciences 36*, 48–49.

Osborne, M. J. and Rubinstein, A. 1994. *A Course in Game Theory*. The MIT Press.

Papadias, D., Kalnis, P., Zhang, J., and Tao, Y. 2001. Efficient olap operations in spatial data warehouses. In *SSTD*. 443–459.

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

Papadimitriou, C. H. 2001. Algorithms, games, and the internet. In *STOC*. 749–753.

Sharifzadeh, M. and Shahabi, C. 2006. The spatial skyline queries. In *VLDB*. 751–762.

Steuer, R. 1986. *Multiple Criteria Optimization*. Wiley, New York.

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

Tao, Y., Xiao, X., and Pei, J. 2007. Efficient skyline and top-k retrieval in subspaces. *IEEE Trans. Knowl. Data Eng. 19,* 8, 1072–1088.

Vetta, A. 2002. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *FOCS*. 416–428.

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

Wang, K., Zhou, S., and Han, J. 2002. Profit mining: From patterns to actions. In *EDBT*. 70–87.

Wang, S., Ooi, B. C., Tung, A. K. H., and Xu, L. 2007. Efficient skyline query processing on peer-to-peer networks. In *ICDE*. 1126–1135.

Wong, R. C.-W., Fu, A. W.-C., and Wang, K. 2003. Mpis: Maximal-profit item selection with cross-selling considerations. In *ICDM*. 371–378.

Wu, P., Zhang, C., Feng, Y., Zhao, B. Y., Agrawal, D., and Abbadi, A. E. 2006. Parallelizing skyline queries for scalable distribution. In *EDBT*. 112–130.

Xia, T. and Zhang, D. 2006. Refreshing the sky: the compressed skycube with efficient support for frequent updates. In *SIGMOD*. 491–502.

Yao, J. T. 2003. Sensitivity analysis for data mining. In *Proceedings of The 22nd International Conference of NAFIPS (the North American Fuzzy Information Processing Society)*. 272–277.

Yuan, Y., Lin, X., Liu, Q., Wang, W., Yu, J. X., and Zhang, Q. 2005. Efficient computation of the skyline cube. In *VLDB*. 241–252.