ALGORITHMS FOR MARKETING-MIX OPTIMIZATION

Joachim Gudmundsson Pat Morin Michiel Smid

February 10, 2010

ABSTRACT. Algorithms for determining quality/cost/price tradeoffs in saturated markets are considered. A product is modeled by d real-valued qualities whose sum determines the unit cost of producing the product. This leads to the following optimization problem: given a set of n customers, each of whom has certain minimum quality requirements and a maximum price they are willing to pay, design a new product and select a price for that product in order to maximize the resulting profit.

An $O(n \log n)$ time algorithm is given for the case, $d = 1$, of products having a single quality, and $O(n(\log n)^{d+1})$ time approximation algorithms are given for products with any constant number, d, of qualities. To achieve the latter result, an $O(nk^{d-1})$ bound on the complexity of an arrangement of homothetic simplices in $\mathbb{R}^d$ is given, where k is the maximum number of simplices that all contain a single points.

1 Introduction

Revealed preference theory [13] is a method of determining a course of business action through the review of historical consumer behaviour. In particular, it is a method of inferring an individual's or a group's preferences based on their past choices. The *marketing mix* [10] of a product consists of the 4 Ps: Product, price, place, and promotion. In the current paper, we present algorithms for optimizing the first two of these by using data about consumers' preferences. That is, we show how, given data on consumer preferences, to efficiently choose a product and a price for that product in order to maximize profit.

Refer to Figure 1. A product $P = (p, q_1, \ldots, q_d)$ is defined by a real-valued *price*, p, and a number of real-valued orthogonal *qualities*, $q_1, \ldots, q_d$. The *market* for a product is a collection of customers $C = \{C_1, \ldots, C_n\}$, where $C_i = (p_i, q_{i,1}, \ldots, q_{i,d})$. A customer will purchase the least expensive product that meets all her minimum quality requirements and whose price is below her maximum price. That is, the customer C_i will *consider* the product $P = (p, q_1, \ldots, q_d)$ if $p \le p_i$ and $q_j \ge q_{i,j}$ for all $j \in \{1, \ldots, d\}$. The customer C_i will *purchase* the product if it has the minimum price among all available products that C_i considers.

We consider markets that are *saturated*. That is, for every customer C_i there is an existing product that satisfies C_i's requirements and among all products that satisfy C_i's requirements, C_i will choose the least expensive product. From the point of view of a manufacturer introducing one or more new products, this means that all customers are *Pareto optimal*, i.e., there are no two customers C_i and C_j such that $q_{i,k} > q_{j,k}$ for all

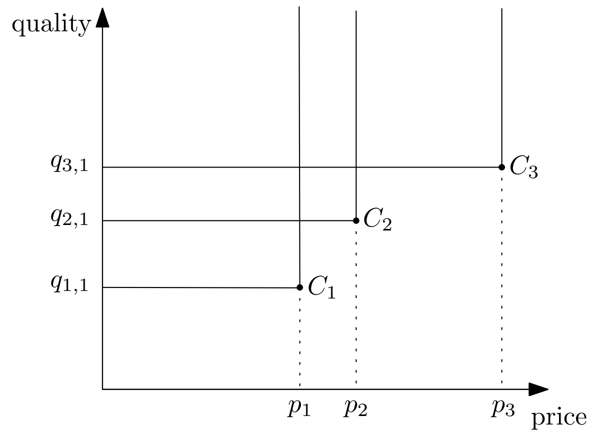


Figure 1: A sample market with $d = 1$ and $n = 3$. A customer will consider any product that is in their upper left quadrant.

$k \in \{1, \dots, d\}$ and $p_i < p_j$. This is because no customer will every purchase a product that is not Pareto optimal, since there is a lower-priced alternative that also satisfies all their minimum quality requirements. Therefore, every customer C_i can be replaced with the (Pareto optimal) product that they purchase.

As an example, consider a market for computers in which an example customer C_i may be looking for a computer with a minimum of 8 GB of RAM, a CPU benchmark score of at least 3000, a GPU benchmark score of at least 2000, and be willing to pay at most \$1500. In addition, there is already a computer on the market which meets these requirements and retails for \$1200. Thus, this customer would be described by the vector $(1200, 8, 3000, 2000)$. If a manufacturer introduces a new product $(1199, 8, 3500, 2000)$ (a computer with 8 GB of RAM, a CPU benchmark score of 3500 and a GPU benchmark score of 2000 retailing for \$1199) then this customer would select this new product over their current choice.

By appropriately reparameterizing the axes, we can assume that the cost, $\mathrm{cost}(P)$, of manufacturing a product $P = (p, q_1, \dots, q_d)$ is equal to the sum of its qualities

$$\mathrm{cost}(P) = \sum_{i=1}^{d} q_i \ .$$

The *profit per unit sold* of P is therefore

$$\mathrm{ppu}(P) = p - \mathrm{cost}(P) \ .$$

In this paper we consider algorithms that a manufacturer can use when planning a new product to introduce into an existing saturated market with the goal being to maximize the profit obtained. More precisely, given a Pareto-optimal *market* of customers $M = \{C_1, \dots, C_n\}$, each having d qualities, the ProductDesign(d) problem is to find a product $P^* \in \mathbb{R}^{d+1}$ such that

$$\mathrm{profit}(P^*) = \mathrm{ppu}(P^*) \times |\{i : C_i \text{ purchases } P^*\}|$$

is maximized.

The term *marketing mix* is probably the most famous phrase in marketing. Not surprisingly, economists and market researchers have considered methods of optimizing the marketing mix in various scenarios. As there are many different models of the problem, it is difficult to compare algorithms.

Most models of marketing-mix problems involve a constant number of real-valued input parameters. This sometimes leads to problems where the optimal solution is one of a constant number of possible closed forms (see, e.g., Thomson and Teng's optimal constant price model [12]). In other cases, a closed form is not achievable, but a (sometimes approximate) solution can be obtained using numerical optimization techniques (see, e.g., Balanchandran and Gensch [2], Thomson and Teng [12], Naik *et al.* [11], Deal [6], Erickson [9]). In all cases, it is expected that the model parameters are derived from real-world data, such as surveys or sales figures, and involves fitting of the model parameters to the available data.

The work in the current paper is different from this previous work in several ways. For one, it is one of the few works that deals primarily with the first two P's, product and pricing. Most existing literature focuses on the marketing P's, namely place and promotion, and to a lesser extent, pricing of an existing product. Secondly, it deals directly with data about individual consumers rather than aggregating this data so that it fits a particular model of consumer behaviour.

We believe that this models very well what happens in online shopping for high cost products such as computers, cameras, and televisions. In such markets, savvy consumers have good data available about both the specifications and the cost of all available products so that marketing efforts are (arguably) less important than the quality and prices of the products. On the other hand, online sellers such as Amazon have large amounts of data about users' past purchases and can use this data as input to the problem. In particular, these sellers know the specifications $q_{i,1},\ldots,q_{i,d}$ and prices p_i of huge quantities of items sold and can use this data to advise a manufacturer that is designing a new product.

In the remainder of the paper we give an $O(n \log n)$ time algorithm for $\textsc{ProductDesign}(1)$ (Section 2), and $O(n(\log n)^{d+1})$ time approximation schemes for $\textsc{ProductDesign}(d)$ (Section 3 and Section 4). Section 5 summarizes our results and concludes with directions for future research.

2 One-dimensional products

In this section, we consider the simplest case, when a manufacturer wishes to introduce a new product in which the quality of a product has only one dimension. Examples of such markets include, for example, suppliers to the construction industry in which items (steel I-beams, finished lumber, logs) must have a certain minimum length to be used for a particular application. An overly long piece can be cut down to size, but using two short pieces instead of one long piece is not an option.

Throughout this section, since $d = 1$, we will use the shorthand $P = (p, q)$ for

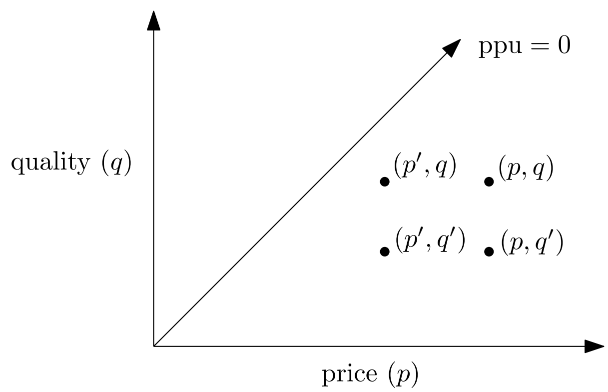


Figure 2: $\text{profit}(p, q) \le \text{profit}(p, q')$ implies that $\text{profit}(p', q) \le \text{profit}(p', q')$ for all $p' \le p$.

the product being designed and q_i for $q_{i,1}$. Thus, we have a set of customers $M = \{(p_1, q_1), \ldots, (p_n, q_n)\}$ and we are searching for a point $P^* = (p^*, q^*)$ that maximizes

$$\text{profit}(p^*, q^*) = (p^* - q^*)|\{i : p^* \le p_i \text{ and } q^* \ge q_i\}| \ .$$

Our algorithm is an implementation of the *plane-sweep* paradigm [4]. The correctness of the algorithm relies on two lemmas about the structure of the solution space. The first lemma is quite easy:

Lemma 1. *The value* (p^*, q^*) *that maximizes* $\text{profit}(p^*, q^*)$ *is obtained when* $p^* = p_i$ *and* $q^* = q_j$ *for some* $i, j \in \{1, \ldots, n\}$*.*

Proof. First, observe the obvious bounds on p^* and q^*:

$$\min\{p_i : i \in \{1, \ldots, n\}\} \le p^* \le \max\{p_i : i \in \{1, \ldots, n\}\}$$

and

$$\min\{q_i : i \in \{1, \ldots, n\}\} \le q^* \le \max\{q_i : i \in \{1, \ldots, n\}\} \ .$$

Consider the arrangement of lines obtained by drawing a horizontal and vertical line through each customer (p_i, q_i) for $i \in \{1, \ldots, n\}$. Within each cell of this arrangement, the function $\text{profit}(p, q)$ is a linear function of p and q and it is bounded. Therefore, within a particular cell, the function is maximized at a vertex. Since each vertex is the intersection of a horizontal and vertical line through a pair of customers, the lemma follows. □

The following lemma, illustrated in Figure 2, is a little more subtle and illustrates a manufacturer's preference for lower-quality products:

Lemma 2. *Let* $q' \le q$ *and let* p *be such that* $0 < \text{profit}(p, q) \le \text{profit}(p, q')$*. Then, for any* $p' \le p$*,* $\text{profit}(p', q) \le \text{profit}(p', q')$*.*

Proof. By definition, $\text{profit}(p, q) = a(p - q)$ and $\text{profit}(p, q') = a'(p - q')$, where a and a' are the number of customers who would consider (p, q) and (p, q'), respectively. These customers are all taken from the set $M_{\geq} = \{(p_i, q_i) \in M : p_i \geq p\}$.

Now, consider the customers in the set $M' = \{(p_i, q_i) \in M : p' \leq p_i < p\}$. By the assumption that customers are Pareto optimal, any customer (p_i, q_i) in M' has $q_i \leq q'$, so all of these customers will consider either (p', q') or (p', q) if either one is offered. Therefore,

$$
\begin{aligned}
\text{profit}(p', q') &= (a' + |M'|)(p' - q') \\
&= a'(p' - q') + |M'|(p' - q') \\
&\geq a'(p' - q') + |M'|(p' - q) && \text{since } q > q' \\
&= a'(p - q') + a'(p' - p) + |M'|(p' - q) \\
&\geq a'(p - q') + a(p' - p) + |M'|(p' - q) && \text{since } a \geq a' \text{ and } (p' - p) < 0 \\
&\geq a(p - q) + a(p' - p) + |M'|(p' - q) && \text{by assumption} \\
&= a(p' - q) + |M'|(p' - q) \\
&= \text{profit}(p', q) \ ,
\end{aligned}
$$

as required. □

Lemma 2 allows us to apply the plane sweep paradigm with a sweep by decreasing price. It tells us that, if a product (p, q') gives better profit than the higher-quality product (p, q) at the current price p, then it will always give a better profit for the remainder of the sweep. In particular, there will never be a reason to consider a product with quality q for the remainder of the algorithm's execution.

Let the customers be labelled $(p_1, q_1), \ldots, (p_n, q_n)$ in decreasing order of p_i, so that $p_{i+1} \leq p_i$ for all $i \in \{1, \ldots, n - 1\}$. At any point in the sweep algorithm, there is a current price p, which starts at $p = \infty$ and decreases during the execution of the algorithm. At the start of the algorithm the algorithm's event queue Q, which is represented as a balanced binary search tree, is initialized to contain the values $p_n, \ldots, p_1$. At all times, the algorithm maintains a list L of qualities $q_1^* > q_2^* > \cdots > q_m^*$ such that $\text{profit}(p, q_1^*) > \text{profit}(p, q_2^*) > \cdots > \text{profit}(p, q_m^*)$. The quality q_1^* is the optimal quality for the current price, p. By the time the algorithm terminates, the quality of the globally-optimal solution will have appeared as the first element in L. To complete the description of the algorithm, all that remains is to show how L and Q are updated during the processing of events in the event queue.

There are two kinds of events in the event queue. *Insertions* occur at the values $p_1, \ldots, p_n$. *Deletions*, which we describe next, occur when the relative order of two adjacent items in L changes. Consider a consecutive pair of the elements q_i^* and q_{i+1}^* in L. When q_i^* and q_{i+1}^* became adjacent in L, it was at some price $p = p_t$ such that $\text{profit}(p_t, q_i^*) > \text{profit}(p_t, q_{i+1}^*)$. Let a_i and a_{i+1} be the number of customers who would consider (p_t, q_i^*) and (p_t, q_{i+1}^*), respectively. Then,

$$\text{profit}(p_t, q_i^*) = (p_t - q_i^*)a_i$$

and

$$\text{profit}(p_t, q_{i+1}^*) = (p_t - q_{i+1}^*)a_{i+1}$$

Now, looking forward in time to a later step in the execution of the algorithm, when $p = p_{t'}$, with $t' > t$, we find that

$$\mathrm{profit}(p_{t'}, q_i^*) = (p_{t'} - q_i^*)(a_i + t' - t)$$

and

$$\mathrm{profit}(p_{t'}, q_{i+1}^*) = (p_{t'} - q_{i+1}^*)(a_{i+1} + t' - t) \ .$$

We are interested in identifying the price $p_{t'}$ where the inequality $\mathrm{profit}(p_{t'}, q_i^*) > \mathrm{profit}(p_{t'}, q_{i+1}^*)$ changes to $\mathrm{profit}(p_{t'}, q_i^*) \leq \mathrm{profit}(p_{t'}, q_{i+1}^*)$. When this happens, q_i^* can be safely discarded from L since, by Lemma 2, $\mathrm{profit}(p, q_i^*)$ will never again exceed $\mathrm{profit}(p, q_{i+1}^*)$ for the remainder of the sweep. The value $p_{t'}$ is a deletion event.

Note that Lemma 2 also allows for binary search on the value $p_{t'}$. In particular, the interval $[p_j, p_{j+1}]$ containing $p_{t'}$ can be found in $O(\log n)$ time, after which the value of $p_{t'}$ can be obtained by solving the linear equation

$$(p_{t'} - q_i^*)(a_i + t' - t) \leq (p_{t'} - q_{i+1}^*)(a_{i+1} + t' - t) \ ,$$

for t'. (The equation is linear because the values a_i and a_{i+1} are constant in the interval (p_j, p_{j+1}).) Thus, whenever two new elements become adjacent in L, we can add the appropriate deletion event to Q in $O(\log n)$ time.

When processing an insertion event p_i, we remove from the tail of L all values q_j^* such that $\mathrm{profit}(p_i, q_j^*) \leq \mathrm{profit}(p_i, q_i)$ and then append q_i onto L. While deleting the elements of L, we also remove all the associated deletion events from Q. Appending q_i to L causes at most one new pair of elements in L to become adjacent, so we add the appropriate deletion event to Q as described above. The time to process the event p_i is there $O((k_i + 1)\log n)$, where k_i is the number of elements that are deleted from L.

When processing a deletion event that deletes q_i^* from L, we simply delete q_i^* from L and its (at most two) associated deletion events from Q. This may cause a new pair of elements, q_{i-1}^* and q_{i+1}^*, in L to become adjacent, so we add the appropriate deletion event to Q. In this way, a deletion event can be processed in $O(\log n)$ time.

Note that, after all the processing associated with an event p_t is complete, the first element, q_1^*, in L is the value that maximizes $\mathrm{profit}(p_t, q_1^*)$. Thus, the algorithm need only keep track, throughout its execution, of the highest profit obtained from the first element of L, and output this value at the end of its execution. This completes the description of the algorithm.

Theorem 1. *There exists an $O(n \log n)$ time algorithm for* ProductDesign(1).

Proof. The correctness of the algorithm described above follows from 2 facts: Lemma 1 ensures that the optimal solution is of the form (p_i, q^*) for some $i \in \{1, \ldots, n\}$, and Lemma 2 ensures that the optimal solution appears at some point as the first element of the list L.

The running time of the algorithm can be bounded as follows: The total number of deletion events processed is at most n, since each such event removes some value q_i from L for some $i \in \{1, \ldots, n\}$. Thus, the cost of processing all deletion events is $O(n \log n)$. Similarly,

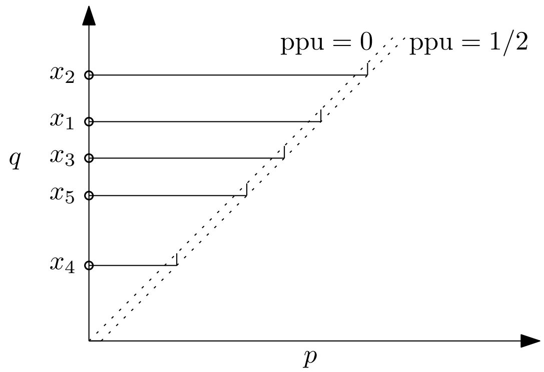


Figure 3: Reducing ELEMENT-UNIQUENESS to PRODUCTDESIGN(1,1).

$\sum_{i=1}^{n} k_i \leq n$ since k_i is the number of elements deleted from L during the processing of p_i. Thus, the time required to handle all insertion events, and therefore the running time of the entire algorithm, is $O(n \log n)$ □

The following theorem shows that a running time of $\Omega(n \log n)$ is inherent in this problem, even when considering approximation algorithms.

Theorem 2. *Let M be an instance of* PRODUCTDESIGN(1) *and (p^*, q^*) be a solution that maximizes* $\text{profit}(p^*, q^*)$. *In the algebraic decision tree model of computation, any algorithm that can find a solution (p, q) such that $2 \cdot \text{profit}(p, q) > \text{profit}(p^*, q^*)$ has $\Omega(n \log n)$ running time in the worst-case.*

Proof. We reduce from the integer ELEMENT-UNIQUENESS problem, which has an $\Omega(n \log n)$ lower bound in the algebraic decision tree model [14]: Given an array $A = [x_1, \ldots, x_n]$ containing n integers, are all the elements of A unique?

We convert A into an instance of PRODUCTDESIGN(1) in $O(n)$ time as follows (refer to Figure 3). For each x_i, $i \in \{1, \ldots, n\}$ we introduce a customer (p_i, q_i) with $p_i = q_i + 1/2$ and $q_i = x_i$. If there exists a value x in A that occurs 2 or more times, then the product $(x + 1/2, x)$ gives a value $\text{profit}(x + 1/2, x) \geq 1$. On the other hand, if there is no such x, then

1. any product (p, q) with $p - q > 1/2$ can not be sold to any customers and

2. any product (p, q) with $p - q > 0$ can be sold to at most 1 customer.

Therefore, if all the elements of A are unique, then $\text{profit}(p^*, q^*) = 1/2$, otherwise $\text{profit}(p^*, q^*) \geq 1$. The result follows. □

3 A near-linear approximation algorithm for bidimensional products

In this section, we consider algorithms for ProductDesign(2), in which products have 2 qualities. As a baseline, we first observe that, if we fix the value of q_2, then the optimal solution of the form (p, q_1, q_2) can be found using a single application of the algorithm in Theorem 1. Therefore, by successively solving the problem for each $q_2 \in \{q_{2,1}, \ldots, q_{2,n}\}$ and taking the best overall solution we obtain an $O(n^2 \log n)$ time algorithm for ProductDesign(2).

More generally, ProductDesign(d) can be solved using $O(n^{d-1})$ applications of Theorem 1 resulting in an $O(n^d \log n)$ time algorithm. Unfortunately, these are the best results known for $d \geq 2$, and, as discussed in Section 5, we suspect that an algorithm with running time $o(n^d)$ will be difficult to achieve using existing techniques. Therefore, in this section we focus our efforts on obtaining a near-linear approximation algorithm.

Fix some constant $\epsilon > 0$. Given an instance M of ProductDesign(d), a point $P \in \mathbb{R}^{d+1}$ is a $(1-\epsilon)$-approximate solution for M if $\mathrm{profit}(P) \geq (1-\epsilon)\,\mathrm{profit}(P^*)$ for all $P^* \in \mathbb{R}^{d+1}$. An algorithm is a (high probability) *Monte-Carlo* $(1-\epsilon)$*-approximation algorithm* for ProductDesign(d) if, given an instance M of size n, the algorithm outputs a $(1-\epsilon)$-approximate solution for M with probability at least $1 - n^{-c}$ for some constant $c > 0$.

Let $r = \max\{\mathrm{ppu}(C_i) : i \in \{1, \ldots, n\}\}$ and observe that r is the maximum profit per unit that can be achieved in this market. Let $E = 1/(1-\epsilon)$ and let $\ell = \lceil \log_E n \rceil$ and observe that $\ell = O(\epsilon^{-1} \log n)$.[1] For each $i \in \{0, 1, 2, \ldots, \ell\}$, define the plane $H_i = \{(p, q_1, q_2) : p - q_1 - q_2 = r(1-\epsilon)^i\}$. The following lemma says that a search for an approximate solution can be restricted to be contained in one of the planes H_i.

Lemma 3. *For any product* $P^* = (p^*, q_1^*, q_2^*)$*, there exists a product* $P = (p, q_1, q_2)$ *such that* $P \in H_i$ *for some* $i \in \{0, \ldots, \ell\}$ *and* $\mathrm{profit}(P) \geq (1-\epsilon)\,\mathrm{profit}(P^*)$.

Proof. There are two cases to consider. If $\mathrm{ppu}(P^*) \leq r/n$ then $\mathrm{profit}(P^*) \leq r$, in which case we set $P = C_i$ where $\mathrm{ppu}(C_i) = r$, so that $P \in H_0$ and $\mathrm{profit}(P) = r \geq \mathrm{profit}(P^*) \geq (1-\epsilon)\,\mathrm{profit}(P^*)$, as required.

Otherwise, $r/n < \mathrm{ppu}(P^*) \leq r$. In this case, consider the plane H_i where $i = \lceil \log_E(r/\mathrm{ppu}(P^*)) \rceil$. Notice, that for any point $P \in H_i$, $\mathrm{ppu}(P) \geq (1-\epsilon)\,\mathrm{ppu}(P^*)$. More specifically, the orthogonal projection $P = (p, q_1, q_2)$ of P^* onto H_i is a product with $p \leq p^*$, $q_1 \geq q_1^*$, and $q_2 \geq q_2^*$. Therefore, any customer who would consider P^* would also consider P, so $\mathrm{profit}(P) \geq (1-\epsilon)\,\mathrm{profit}(P^*)$, as required. $\square$

Lemma 3 implies that the problem of finding an approximate solution to ProductDesign(2) can be reduced to a sequence of problems on the planes $H_0, \ldots, H_\ell$. Refer to Figure 4. Each customer C_j considers all products in a quadrant whose corner is C_j. The intersection of this quadrant with H_i is a (possibly empty) equilateral triangle $\Delta_{i,j}$. The customer C_j will consider a product P in H_i if and only P is in $\Delta_{i,j}$. Thus, the problem of solving

[1] This can be seen by taking the limit $\lim_{\epsilon \to 0^+}(\epsilon/\log(E))$ using one application of L'Hôpital's Rule.

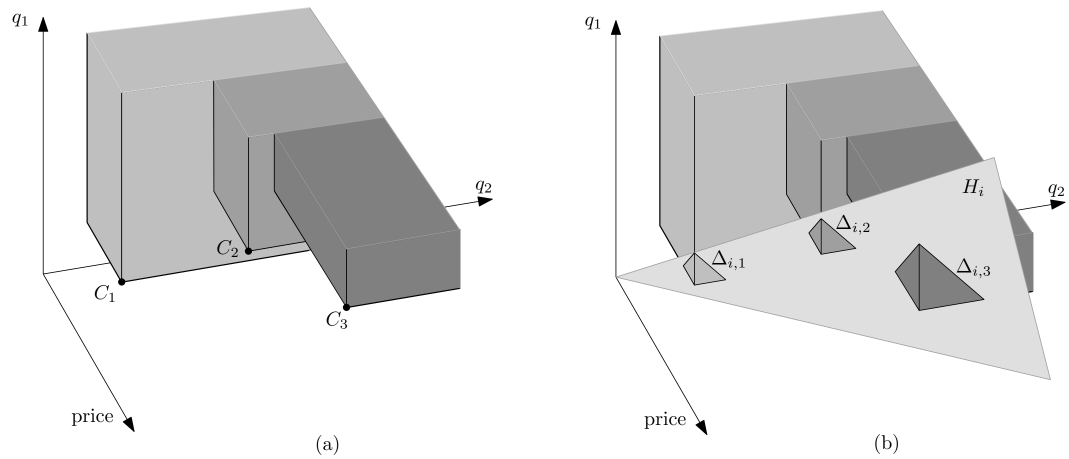


Figure 4: The intersection of H_i with customers' quadrants is a set of homothetic equilateral triangles.

ProductDesign(2) restricted to the plane H_i is the problem of finding a point contained in the largest number of equilateral triangles from the set $\Delta_i = \{\Delta_{i,j} : j \in \{1, \ldots, n\}\}$.

Note that the elements in Δ_i are *homothets* (translations and scalings) of an equilateral triangle, so they form a collection of *pseudodisks* and we wish to find the deepest point in this collection of pseudodisks. No algorithm with running time $o(n^2)$ is known for solving this problem exactly, but Aronov and Har-Peled [1] have recently given a Monte-Carlo $(1-\epsilon)$-approximation algorithm for this problem that runs in time $O(\epsilon^{-2} n \log n)$. By applying this algorithm to each of Δ_i for $i \in \{1, \ldots, \ell\}$, we obtain the following result:

Theorem 3. *For any $\epsilon > 0$, there exists an $O(\epsilon^{-3} n (\log n)^2)$ time high-probability Monte-Carlo $(1-\epsilon)$-approximation algorithm for* ProductDesign(2).

4 A near-linear approximation algorithm for constant d

In this section we extend the algorithm from the previous section to (approximately) solve ProductDesign(d) for any constant value of d. The algorithm is more or less unchanged, except that the proof requires some new results on the combinatorics of arrangements of homothets.

As before, let $r = \max\{\mathrm{ppu}(C_i) : i \in \{1, \ldots, n\}\}$ and let $\ell = \lceil \log_E n \rceil$. For each $i \in \{0, 1, 2, \ldots, \ell\}$, define the hyperplane $H_i = \{(p, q_1, \ldots, q_d) : p - \sum_{i=1}^{d} q_i = r(1-\epsilon)^i\}$. The following lemma has exactly the same proof as Lemma 3.

Lemma 4. *For any product $P^* = (p^*, q_1^*, \ldots, q_d^*)$, there exists a product $P = (p, q_1, \ldots, q_d)$ such that $P \in H_i$ for some $i \in \{0, \ldots, \ell\}$ and $\mathrm{profit}(P) \geq (1-\epsilon)\,\mathrm{profit}(P^*)$.*

Again, each customer C_j defines a regular simplex $\Delta_{i,j}$ in H_i such that C_j will consider $P \in H_i$ if and only if $P \in \Delta_{i,j}$. In this way, we obtain a set $\Delta_i = \{\Delta_{i,1}, \ldots, \Delta_{i,n}\}$

of homothets of a regular simplex in $\mathbb{R}^d$ and we require an algorithm to find a ($(1-\epsilon)$-approximation to) the point that is contained in the largest number of these simplices. The machinery of Aronov and Har-Peled [1] can be used to help solve this problem, but not before we prove some preliminary results, the first of which is a combinatorial geometry result.

Lemma 5. *Let Δ be a set of n homothets of a regular simplex in $\mathbb{R}^d$, for $d = O(1)$, and such that no point in $\mathbb{R}^d$ is contained in more than k elements of Δ. Then, the total complexity of the arrangement, $A(\Delta)$, of the simplices in Δ is $O(nk^{d-1})$.*

Proof. We first consider the simpler case in which the elements of Δ are translates (without scaling) of a regular simplex. Suppose that the total complexity of $A(\Delta)$ is m. Then, by the pigeonhole principle, there is some element T in Δ whose surface is involved in m/n features of $A(\Delta)$. (Note that this implies that T intersects all the elements of a set $\Delta' \subseteq \Delta$ with $|\Delta'| = \Omega((m/n)^{1/(d-1)})$, since otherwise there are not enough elements in Δ' to generate m/n features on the surface of Δ.)

Observe that, since the elements of Δ' are all unit size and they all intersect T, that they are all contained in a ball of radius $O(1)$ centered at the center of T. Furthermore, since each element of Δ' has volume $\Omega(1)$ this implies that some point must be contained in $\Omega((m/n)^{1/(d-1)})$ elements of Δ'. Thus, we obtain the inequality $k \geq \Omega((m/n)^{1/(d-1)})$, or, equivalently, $m \leq O(nk^{d-1})$, as required.

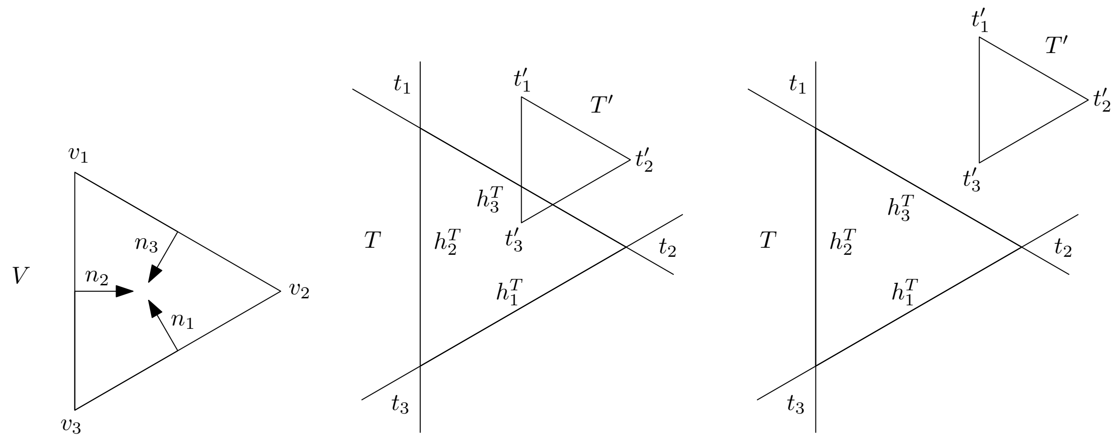
Now, for the case where the elements of Δ are homothets (translations and scalings) of a regular simplex, we proceed as follows. Suppose that $|A(\Delta)| = rn$. Our goal is to show that $r = O(k^{d-1})$. Label the elements of Δ as $T_1, \ldots, T_n$ in increasing order of size and consider the smallest element T_i such that T_i contributes at least r features to $A(\{T_i, \ldots, T_n\})$. Such a T_i is guaranteed to exist, since otherwise $|A(\Delta)| \leq rn$.

Now, T_i intersects all the elements in some set $\Delta' \subseteq \{T_{i+1}, \ldots, T_n\}$ with $|\Delta'| = \Omega(r^{1/(d-1)})$. Shrink each element T' in Δ' so as to obtain an element T'' such that (a) the size of T'' is equal to the size of T_i, (b) $T'' \subseteq T'$, and (c) T'' intersects T_i. Call the resulting set of shrunken elements Δ''. Condition (a) and the packing argument above imply that there is a point $p \in \mathbb{R}^d$ that is contained in $\Omega(r^{1/(d-1)})$ elements of Δ''. Condition (b) implies that p is contained in $\Omega(r^{1/(d-1)})$ elements of Δ' and hence also Δ. Therefore, we conclude, as before, that $r = O(k^{d-1})$, which completes the proof. □

Remark. Lemma 5 is somewhat surprising, since the union of n homothets of a regular tetrahedron in, for example, $\mathbb{R}^3$ can easily have complexity $\Omega(n^2)$. This fact makes it impossible to apply the "usual" Clarkson-Shor technique [5] that relates the complexity of the first k levels to that of the boundary of the union (the 0-level).

Lemma 6. *Let Δ be a set of n homothets of a regular simplex in $\mathbb{R}^d$ such that no point of $\mathbb{R}^d$ is contained in more than k simplices of Δ. Then the arrangement $A(\Delta)$ of Δ can be computed in $O(n(k^{d-1} + (\log n)^d))$ time.*

Proof. Computing the arrangement $A(\Delta)$ can be done in the following way. Sort the elements of Δ by decreasing size and construct $A(\Delta)$ incrementally by inserting the elements



The simplex V — T intersects T' — T does not intersect T' ($t_3' \notin h_3^T$)

Figure 5: The simplex $T \in \Delta$ intersects $T' \in \Delta$ if and only if h^{T_i} contains t_i' for all $i \in \{1, \dots, d\}$.

one by one. When inserting an element T, use a data structure (described below) to retrieve the elements of Δ that intersect T and discard the elements that are smaller than T. The proof of Lemma 5 implies that there will be at most $O(k)$ such elements. The intersection of the surfaces of these $O(k)$ elements with the surface of T has size $O(k^{d-1})$ and can be computed in $O(k^{d-1})$ time using $d+1$ applications of the standard algorithm for computing an arrangement of hyperplanes in $\mathbb{R}^{d-1}$ [7, 8]. Thus, ignoring the cost of finding the elements that intersect T, the overall running time of the algorithm is $O(nk^{d-1})$.

All that remains is to describe a data structure for retrieving the elements that intersect a given simplex $T \in \Delta$. In the following we describe a data structure that can be constructed in $O(n(\log n)^d)$ time and can answer queries in $O(x + (\log n)^d)$ time, where x is the size of the output. This data structure will be constructed once and queried n times. The total size of the outputs over all n queries will be the $O(|A(\Delta)|) = O(nk^{d-1})$.

Refer to Figure 5. Suppose that every element $T \in \Delta$ is a homothet of the regular simplex V whose vertices are $v_1, \dots, v_{d+1}$ and let $n_1, \dots, n_{d+1}$ be the inner normals of the faces of V where n_i is the face opposite (not incident on) vertex v_i. For any $T \in \Delta$, let t_i be the image of v_i under the homothetic transformation that takes V onto T. Observe that, if h is a halfspace of $\mathbb{R}^d$ with inner normal n_i, then h intersects T if and only if h contains t_i. Furthermore, every simplex $T \in \Delta$ is the intersection of d halfspaces $h_1^T, \dots, h_{d+1}^T$ where the inner normal of h_i^t is n_i. Therefore, a simplex $T \in \Delta$ intersects a simplex $T' \in \Delta$ if and only if h_i^T contains t_i' for all $i \in \{1, \dots, d+1\}$.

This implies that the elements of Δ can be stored in a $(d+1)$-layer range tree [3]. The ith layer of this tree, for $i \in \{1, \dots, d+1\}$, stores elements of Δ ordered by the projection of t_i onto n_i. In this way, the range tree can return the set of all simplices in Δ that intersect a query simplex $T \in \Delta$. The size of this range tree is $O(n(\log n)^d)$ and it can answer queries in time $O(x + n(\log n)^d)$ where x is the size of the query result. Since each simplex in Δ is passed as a query to this data structure exactly once, the total sizes of outputs over all

n queries is equal to the number of pairs $T_1, \ldots, T_2 \in \Delta$ such that $T_1 \cap T_2 \neq \emptyset$. But the number of such pairs is certainly a lower bound on $|A(\Delta)|$ so it must be at most $O(nk^{d-1})$. This completes the proof. □

Lemma 6 can be used as a subroutine in the algorithm of Aronov and Har-Peled [1, Theorem 3.3], to obtain the following Corollary.

Corollary 1. *Let Δ be a set of n homothets of a regular simplex in $\mathbb{R}^d$ such that some point $p \in \mathbb{R}^d$ is contained in δ elements of Δ. Then there exists an algorithm whose running time is $O(\epsilon^{-2d} n(\log n)^{d-1} + n(\log n)^d)$ and that, with high probability, returns a point $p' \in \mathbb{R}^d$ contained in at least $(1-\epsilon)\delta$ elements of Δ.*

As before, an approximate solution to ProductDesign(d) problem reduces to finding deepest point in each of the sets $\Delta_1, \ldots, \Delta_\ell$ where Δ_i is a set of n d-simplices in H_i. By using the algorithm of Corollary 1 to do this we obtain the following result:

Theorem 4. *For any $\epsilon > 0$, there exists an $O(\epsilon^{-(2d+1)} n(\log n)^d + n(\log n)^{d+1})$ time high-probability Monte-Carlo $(1-\epsilon)$-approximation algorithm for* ProductDesign(d).

5 Conclusions

We have given an $O(n \log n)$ time exact algorithm for solving ProductDesign(1) and $O(n(\log n)^{d+1})$ time approximation algorithms for solving ProductDesign(d). The running time of the exact ProductDesign(1) algorithm is optimal and no algorithm that produces a $(2-\epsilon)$-approximation, for any $\epsilon > 0$, can run in $o(n \log n)$ time.

In developing these algorithms, we gave a proof (the proof of Lemma 5) that shows that an arrangement of n fat convex objects in $\mathbb{R}^d$ has complexity $O(nk^{d-1})$ where k is the maximum number of objects that contain any given point. We expect that this result, and the algorithm for approximate depth that arise from it [1], will find other applications.

An exact near-linear time algorithm for the case $d = 2$ seems to be out of reach. It appears as if this problem requires (at least) a solution to the problem of finding a point contained in the largest number of homothets of an equilateral triangle, a problem for which no subquadratic time algorithm is known. Is it possible to prove some kind of a lower bound? The related problem of finding the point contained in the largest number of unit disks is 3-Sum hard [1] providing some evidence that this problem will be difficult to solve in subquadratic time.

In this paper we considered the case where the problem is parameterized by the number, d, of orthogonal qualities that a product may have. Another case to consider is the case in which a manufacturer wishes to introduce some number, k, $k > 1$, of new products into a market. Is this problem NP-hard? Does it have a polynomial time approximation algorithm?

Acknowledgement

The authors would like to thank Gautam Das for bringing this class of problems to our attention and Timothy Chan for helpful discussions on the subject of approximate depth.

References

[1] B. Aronov and S. Har-Peled. On approximating the depth and related problems. *SIAM Journal on Computing*, 38(3):899–921, 2008.

[2] V. Balachandran and D. H. Gensch. Solving the "marketing mix" problem using geometric programming. *Management Science*, 21(2):160–171, 1974.

[3] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18:509–517, 1975.

[4] J. L. Bentley and T. A. Ottman. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computing*, C-28:643–647, 1979.

[5] K. Clarkson and P.W. Shor. Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989.

[6] K. R. Deal. Optimizing advertising expenditures in a dynamic duopoly. *Operations Research*, 27(3):682–692, 1979.

[7] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal on Computing*, 15:341–363, 1986.

[8] H. Edelsbrunner, R. Seidel, and M. Sharir. On the Zone Theorem for hyperplane arrangements. *SIAM Journal on Computing*, 22(2):418–429, 1993.

[9] G. M. Erickson. *Dynamic Models of Advertising Competition: Open- and Closed-Loop Extensions*. Kluwer Academic Publishers, Boston, MA, 1991.

[10] P. Kotler and K. Lane. *Marketing Management*. Prentice-Hall, 2005.

[11] P. A. Naik, K. Raman, and R. S. Winer. Planning marketing-mix strategies in the presence of interaction effects. *Marketing Science*, 24(1):25–34, 2005.

[12] G. L. Thompson and J.-T. Teng. Optimal pricing and advertising policies for new porduct oligopoly models. *Marketing Science*, 3(1):148–168, 1984.

[13] H. R. Varian. Revealed preference. In Michael Szenberg, Lall Ramrattan, and Aaron A. Gottesman, editors, *Samuelsonian Economics in the 21st Century*, chapter 6, pages 99–115. Oxford University Press, New York, 2006.

[14] A. C. Yao. Lower bounds for algebraic computation trees with integer inputs. *SIAM Journal on Computing*, 20(4):655–668, 1991.