Optimization Beyond Prediction: Prescriptive Price Optimization

Shinji Ito
NEC Corporation
s-ito@me.jp.nec.com

Ryohei Fujimaki
NEC Corporation
rfujimaki@nec-labs.com

ABSTRACT

This paper addresses a novel data science problem, *prescriptive price optimization*, which derives the optimal price strategy to maximize future profit/revenue on the basis of massive predictive formulas produced by machine learning. The prescriptive price optimization first builds sales forecast formulas of multiple products, on the basis of historical data, which reveal complex relationships between sales and prices, such as price elasticity of demand and cannibalization. Then, it constructs a mathematical optimization problem on the basis of those predictive formulas. We present that the optimization problem can be formulated as an instance of binary quadratic programming (BQP). Although BQP problems are NP-hard in general and computationally intractable, we propose a fast approximation algorithm using a semi-definite programming (SDP) relaxation. Our experiments on simulation and real retail datasets show that our prescriptive price optimization simultaneously derives the optimal prices of tens/hundreds products with practical computational time, that potentially improve approximately 30% of gross profit of those products.

CCS CONCEPTS

• **Computing methodologies** → *Planning and scheduling*; • **Applied computing** → *Business process management*; *Decision analysis*; *Marketing*;

KEYWORDS

scheduling, decision making, mathematical optimization

ACM Reference format:
Shinji Ito and Ryohei Fujimaki. 2017. Optimization Beyond Prediction: Prescriptive Price Optimization. In *Proceedings of KDD '17, Halifax, NS, Canada, August 13-17, 2017,* 9 pages.
https://doi.org/10.1145/3097983.3098188

1 INTRODUCTION

Recent advances in machine learning have had a great impact on maximizing business efficiency in almost all industries. In the past decade, predictive analytics has become a particularly notable emergent technology. It reveals inherent regularities behind Big Data and provides forecasts of future values of key performance indicators. Predictive analytics has made it possible to conduct proactive decision makings in a data-scientific manner. Along with the growth of predictive analytics, *prescriptive analytics* [5] has been recognized in the market as the next generation of advanced analytics. Advances in predictive analytics w.r.t. both algorithms and software have made it considerably easy to produce a massive amount of predictions, purely from data. The key questions in prescriptive analytics is then *how to benefit from those massive amount of predictions*, i.e., *how to automate complex decision makings by algorithms empowered using predictions*. This raises a technical issue regarding the integration of machine learning with relevant theories and algorithms in terms of mathematical optimization, numerical simulation, etc.

Predictive analytics usually produces two important outcomes: 1) predictive formulas revealing inherent regularities behind data, and 2) forecasted values for key performance indicators. It would seem a straightforward to integrate the later ones with mathematical optimization by treating the forecasted values as their inputs, and in fact there exists lots of existing studies such as inventory management [1], energy purchase portfolio optimization [17], smart water management [3, 7], etc. The focus of this paper is on problems in which decision variables (e.g., prices) in a target optimization problem (e.g., profit/revenue maximization) are explanation variables in a prediction problem (e.g., sales forecasting). Suppose we obtain regression formulas to forecast sales of multiple products, formulas which reveal complex relationships between sales and prices, such as price elasticity of demand [19] and cross price effects (a.k.a. cannibalization) [20, 22]. The problem is then to find the optimal price strategy to maximize future profit/revenue from such massive predictive formulas. We refer to the problem as *prescriptive price optimization*.

The prescriptive price optimization is a variant of *revenue management* [12, 21], which has been actively studied in areas of marketing, economics, and operations research. Our focus is on static but simultaneous optimization of many products using machine learning based predictions. Although there are several existing studies such as fast-fashion retailer [4], online retailer [6], hotel room [13, 15], etc. (a comprehensive survey is given by [14]). However, existing methods have strong restrictions in demand modeling capability, e.g. one does not consider cross-price effects, and another is domain specific and is hard to be generalized across industries. Further, most existing studies employ mixed-integer programming for optimizing prices, whose computational cost exponentially increases over increasing number of products. The prescriptive price optimization aims more machine learning based (therefore flexibly modeled) revenue management which enables simultaneous price optimization of tens/hundreds of products. Recently, Ito and

KDD '17, August 13-17, 2017, Halifax, NS, Canada

ACM ISBN 978-1-4503-4887-4/17/08…$15.00
https://doi.org/10.1145/3097983.3098188

Fujimaki [10] proposed a highly scalable algorithm for prescriptive price optimization. Their method, however, cannot deal with complex business constraints such as constraints on the number of discounted products. Yabe et.al. [29] recently proposed robust price optimization algorithms for computing conservative solutions.

This paper addresses prescriptive price optimization, and our contributions can be summarized as follows:

Prescriptive Price Optimization Using Massive Regression Formulas: We establish a mathematical framework for prescriptive price optimization. First, multiple predictive formulas (i.e., sales forecasting models for individual products) using non-linear price features are derived using a regression technique with historical data. These are then transformed into a profit (or revenue) function, and the optimal price strategy is obtained by maximizing the profit function under business constraints. We show that the problem can be formulated as a binary quadratic programming (BQP) problem.

Efficient Algorithm for Prescriptive Price Optimization: Our predictive price optimization is a kind of combinatorial optimization, and is NP-hard in general. In contrast to existing price optimization methods that employ mixed-integer programing (MIP) [4, 13], whose computational costs exponentially increase with increasing problem size, this paper proposes an alternative relaxation-rounding method that employ semidefinite programming (SDP) [27]. Although there exists much literature regarding SDP relaxation (SDR) for quadratic programming problems [2, 9, 11, 18, 23] including our BQP, there are not many studies regarding rounding methods subject to general linear constraints, which is essential in order to obtain feasible solutions of our price optimization subject to business constraints. As far as we know, our method combining SDP and rounding techniques is the first approach that solves large-scale price optimization subject to flexible business constraints in realistic time.

Experiments on a Real Retail Dataset: We evaluated the prescriptive price optimization on a real retail dataset with respect to 50 beer products as well as a simulation dataset. The result indicates that the derived price strategy could improve approximately 30 % of gross profit of these products. Further, our detailed empirical evaluation reveals risk of overestimated profits caused by estimation errors in machine learning and a way to mitigate such an issue using sparse learning. In simulation studies, we confirm that the hold-out validation is useful for evaluating the actual profit. The improvement of the profit of a real supermarket is evaluated by this hold-out validation.

2 PRESCRIPTIVE PRICE OPTIMIZATION

2.1 Problem and Pipeline Descriptions

Let us define terminologies for three types of variables: *decision*, *target*, and *external* variables. Decision variables are those we wish to optimize, i.e., product prices. Target variables are ones we predict, i.e., sales quantities. External variables consist of the other information we can utilize, e.g., weather, temperature, product information, etc. We assume we have historical observations of them. The goal, then, is to derive the optimal values for the decision variables with given external variables so as to maximize a predefined objective function, e.g. profit or revenue. In prescriptive price optimization, the decision and target variables are product prices and sales quantities, respectively. The external variables might be weather, temperature, product information. The objective function is future profit or revenue that is ultimately the measure of business efficiency.

Our prescriptive price optimization is conducted in of two stages, which we refer to as *modeling* and *optimization* stages. In the modeling stage, using regression techniques, we build predictive formulas for the target variables by employing the decision and external variables (or their transformations) as features on the basis of historical data relevant to them. This stage reveals complex relationships between sales and prices among such multiple products as price elasticity of demand and cannibalization. For this, it takes into account the effect of external variables. In the optimization stage, with given values of external variables, we transform the multiple regression formulas into a mathematical optimization problem. Business requirements expressed as linear constraints are input by users of this system and are reflected. By solving the optimization problem, we are able to obtain optimal values for the decision variables (i.e., an optimal price strategy).

2.2 Modeling Predictive Formulas

Suppose we have M products and a product index is denoted by $m \in \{1, \ldots, M\}$. We employ linear regression models to forecast the sales quantity q_m of the m-th product on the basis of price, denoted by p_m, and the external variables. This modeling stage has two tunable areas: 1) feature transformations and 2) a learning algorithm of linear regression.

For the feature transformations, we suppose we have D arbitrary but univariate transformations on p_m, which is denoted by ϕ_d $(d = 1, \ldots, D)$. ϕ_d might be designed to incorporate a domain specific relationship between price and demand, such as the law of diminishing marginal utility [19], as well as to achieve high prediction accuracy. Further, the external variables might be transformed into features denoted by g_d $(d = 1, \ldots, D')$. On the basis of these features, the regression model of the m-th product can be expressed as follows:

$$q_m^{(t)}(\mathbf{p}, \mathbf{g}) = \alpha_m^{(t)} + \sum_{m'=1}^{M} \sum_{d=1}^{D} \beta_{mm'd}^{(t)} \phi_d(p_{m'}) + \sum_{d=1}^{D'} \gamma_{md}^{(t)} g_d, \quad (1)$$

where $\alpha_m^{(t)}$, $\beta_{mm'd}^{(t)}$, and $\gamma_{md}^{(t)}$ are bias, the coefficient of $\phi_d(p_{m'})$, and the coefficient of g_d, respectively. Also, $\mathbf{p}$ and $\mathbf{g}$ are defined as $\mathbf{p} = [p_1, \ldots, p_M]^\top$ and $\mathbf{g} = [g_1, \ldots, g_{D'}]^\top$. The superscription (t) for the time index is introduced for optimization through multiple time steps. For example, in order to optimize prices for the next one week, we might need seven regression models (one model per day) for a single product.

For the learning algorithm, in principle, any standard algorithm, such as least square regression, ridge regression (L_2 regularized), Lasso (L_1-regularized) [24], or orthogonal matching pursuit (OMP, L_0 regularized) [26] would be applicable with our methodology. The choice of learning algorithm depends on the way relationships among multiple products are to be modeled. Experience shows that these relationships are complicated yet usually sparse in practice,[1]

[1]For example, a price of rice ball might be related with sales of green tea, but might not be related with those of milk.

and sparse learning algorithms might be preferable. More detailed discussions are presented in Section 5.2.

2.3 Building Optimization Problem

Using predictive formulas obtained in the modeling stage, given costs $\mathbf{c} = [c_1, \ldots, c_M]^\top$, the *gross profit* can be represented as:

$$\ell(\mathbf{p}) = \sum_{t=1}^{T}\sum_{m=1}^{M}(p_m - c_m)q_m^{(t)}(\mathbf{p}, \mathbf{g}^{(t)}) = (\mathbf{p}-\mathbf{c})^\top \mathbf{q}, \tag{2}$$

where $\mathbf{g}^{(t)}$ stands for values of external variables (e.g. weather forecast) for the time step t and $\mathbf{q} = [\sum_{t=1}^{T} q_1^{(t)}(\mathbf{p}, \mathbf{g}^{(t)}), \ldots, \sum_{t=1}^{T} q_M^{(t)}(\mathbf{p}, \mathbf{g}^{(t)})]^\top$. Note that $\mathbf{c} = 0$ gives the sales revenue on $\mathbf{p}$.

For later convenience, let us introduce ξ_m and ζ_m as

$$\xi_m(p_m) = \sum_{t=1}^{T}(p_m - c_m)(\alpha_m^{(t)} + \sum_{d=1}^{D'}\gamma_{md}^{(t)} g_d^{(t)}),$$

$$\zeta_{mm'}(p_m, p_{m'}) = \sum_{t=1}^{T}(p_m - c_m)\sum_{d=1}^{D}\beta_{mm'd}^{(t)}\phi_d(p_{m'}).$$

Then, (2) can be rewritten by

$$\ell(\mathbf{p}) = \sum_{m=1}^{M}\xi_m(p_m) + \sum_{m=1}^{M}\sum_{m'=1}^{M}\zeta_{mm'}(p_m, p_{m'}). \tag{3}$$

In practice, p_m is often chosen from the set $\{P_{m1}, \ldots, P_{mK}\}$ of K price candidates where P_{m1} might be a list price and P_{mk} $(k > 1)$ might be discounted prices such as 3%-off, 5%-off, $1-off. Hence, the problem of maximizing the gross profit can be formulated as follows:

$$\begin{aligned} &\text{Maximize} && \ell(\mathbf{p}) \\ &\text{subject to} && p_m \in \{P_{m1}, \ldots, P_{mK}\} \quad (m = 1, \ldots, M). \end{aligned} \tag{4}$$

An exhaustive search with respect to this problem would require $\Theta(K^M)$-time computation, and hence would be computationally intractable when M is large. Further, the price strategy might have to satisfy certain business requirements. Let us consider a situation in which we can discount only L products at the same time. Assume that P_{m1} is the list price for the m-th product and P_{mk} $(k > 1)$ are discounted prices. A requirement here can be expressed in terms of the following constraints:

$$|\{m \in \{1, \ldots, M\} \mid p_m = P_{m1}\}| \geq M - L. \tag{5}$$

The system allows users to input such business requirements, which are then transformed into mathematical constraints, as shown above. The problem (4) is solved with such constraints taken into account. The type of requirements we can deal with is discussed in the next section.

2.4 BQP formulation

The general form (4) is intractable due to combinatorial nature of the optimization and also non-linear mapping ξ_m and $\zeta_{m,m'}$, and a naive method would require unrealistic computational cost. In order to efficiently solve (4), we here convert it into a more tractable form.

Let us first introduce binary variables $z_{m1}, \ldots, z_{mK} \in \{0, 1\}$ satisfying $\sum_{k=1}^{K} z_{mk} = 1$. Here, $z_{mk} = 1$ refer to $p_m = P_{mk}$, which gives $p_m = \sum_{k=1}^{K} P_{mk} z_{mk} \quad (m = 1, \ldots, M)$. Then, $\zeta_{mm'}(p_m, p_{m'})$ can be rewritten as

$$\zeta_{mm'}(p_m, p_{m'}) = \mathbf{z}_m^\top Q_{mm'} \mathbf{z}_{m'}, \tag{6}$$

where $\mathbf{z}_m = [z_{m1}, \ldots, z_{mK}]^\top$ and $Q_{ij} \in \mathbb{R}^{K\times K}$ is the matrix of which (k_1, k_2)-entry is $\zeta_{ij}(P_{ik_1}, P_{ik_1})$. Similarly, $\xi_i(p_i)$ can be rewritten as follows:

$$\xi_i(p_i) = \mathbf{r}_i^\top \mathbf{z}_i := [\xi_i(P_{i1}), \ldots, \xi_i(P_{iK})]^\top \mathbf{z}_i. \tag{7}$$

From (3), (6) and (7), (4) is reduced to

$$\begin{aligned} &\text{Maximize} && f(\mathbf{z}) := \mathbf{z}^\top Q \mathbf{z} + \mathbf{r}^\top \mathbf{z} \\ &\text{subject to} && \mathbf{z} = [z_{11}, \ldots, z_{1K}, z_{21}, \ldots, z_{MK}]^\top \in \{0, 1\}^{MK}, \end{aligned} \tag{8}$$

$$\sum_{k=1}^{K} z_{mk} = 1 \quad (m = 1, \ldots, M), \tag{9}$$

where $Q \in \mathbb{R}^{MK\times MK}$ and $\mathbf{r} \in \mathbb{R}^{MK}$ are composed of $\{Q_{ij}\}_{1\leq i,j\leq M}$ and $\{\mathbf{r}_i\}_{1\leq i\leq M}$, respectively. Problem (8) is referred to as a BQP problem and is known to be NP-hard. Although it would be hard to find a global optimal solution of (8), its relaxation methods have been well-studied and further, in Section 3, we propose a relaxation method which empirically obtains an accurate solution.

Using $\mathbf{z}$, the constraint (5) can be expressed as $\sum_{m=1}^{M} z_{m1} \geq M - L$. This is a linear constraint on $\mathbf{z}$ and such linear constraints can be naturally incorporated into (8) (the problem remains to be BQP). We can deal with, in general, linear constraints

$$\begin{aligned} &\mathbf{a}_u^\top \mathbf{z} = b_u \quad (u = 1, \ldots, U), \\ &\mathbf{c}_v^\top \mathbf{z} \leq d_v \quad (v = 1, \ldots, V), \end{aligned} \tag{10}$$

which are able to cover a variety of practical business constraints, such as combination of discounted products and inventory constraints.

2.5 Baseline: MIP relaxation method

Problem (8) is a kind of mixed integer quadratic programming called binary quadratic programming. One of the most well-known relaxation techniques for efficiently solving it is mixed integer linear programming [16].

By introducing auxiliary variables $\bar{z}_{ij}$ $(1 \leq i < j \leq KM)$ corresponding to $\bar{z}_{ij} = z_i z_j$, and also introducing $\sum_{i=mK+1}^{j} \bar{z}_{ij} + \sum_{i=j+1}^{mK+K} \bar{z}_{ji} = (\sum_{i=mK+1}^{mK+K} z_i - 1)z_j = 0$, we can transform (8) into the following MILP problem [16]:

$$\text{Maximize} \sum_{i=1}^{KM}(r_i + q_{ii})z_i + \sum_{i=1}^{KM}\sum_{j=i+1}^{KM}(q_{ij} + q_{ji})\bar{z}_{ij} \tag{11}$$

$$\begin{aligned} \text{subject to} \sum_{i=mK+1}^{mK+K} z_i = 1 \quad (0 \leq m \leq M-1), \\ 0 \leq \bar{z}_{ij} \leq z_i, z_j \quad (1 \leq i < j \leq KM), \\ \sum_{i=mK+1}^{j} \bar{z}_{i,j} + \sum_{i=j+1}^{mK+K} \bar{z}_{j,i} = 0 \\ (mK < j \leq mK + K, \quad 0 \leq m \leq M-1), \\ z_i \in \{0, 1\} \quad (1 \leq i \leq KM). \end{aligned}$$

Note that, though the objective function and constraints are linear, integer variables still exist. Therefore, worst case complexity is

still exponential, which means computational cost might rapidly increase w.r.t. increasing problem size even with a modern commercial MILP solver.

3 SDP RELAXATION-BASED BQP SOLVER

In order to efficiently solve our prescriptive price optimization formulated in the BQP problem, this section proposes a fast approximation method. Our idea is closely related to Goemans-Williamson's MAX-CUT approximation algorithm (GW algorithm) [8]. GW algorithm is applicable to *unconstrained* BQP, but it cannot be applied to our *constrained* BQP directly. This section propose a variant of GW algorithm applicable to constrained BQP.

The proposed method consists of the following two steps:

(1) Transform the original BQP problem (8) into a semidefinite programming (SDP) problem (13) by borrowing the relaxation technique used in the GW algorithm.
(2) Construct a feasible solution of the original BQP problem on the basis of the optimal solution of the SDP problem.

The global optimal solution of the SDP problem can be efficiently computed by a recent advanced solver such as SDPA [30] and SDPT3 [25]. In our experiments, empirical computational time fits a cubic order of problem size.

3.1 SDP Relaxation

Let us first define $\bar{Q}$ by $\bar{Q} := (Q + Q^\top)/2$, which satisfies $\mathbf{x}^\top \bar{Q}\mathbf{x} = \mathbf{x}^\top Q\mathbf{x}$. Define $A \in \mathrm{Sym}_{KM+1}$[2] by

$$A = \frac{1}{4}\begin{bmatrix} \mathbf{1}^\top \bar{Q}\mathbf{1} + 2\mathbf{r}^\top \mathbf{1} & (\mathbf{r} + \bar{Q}\mathbf{1})^\top \\ \mathbf{r} + \bar{Q}\mathbf{1} & \bar{Q} \end{bmatrix}. \tag{12}$$

Using A, construct the following SDP[3]:

$$\begin{aligned} &\max\ A \bullet Y \\ &\text{s.t. } Y = [y_{ij}]_{0 \le i,j \le KM} \in \mathrm{Sym}_{KM+1}, \quad Y \succeq O, \\ &\quad y_{ii} = 1 \quad (0 \le i \le KM), \\ &\quad \textstyle\sum_{k=1}^{K} y_{0,Km+k} = 2 - K \quad (0 \le m \le M-1), \\ &\quad \textstyle\sum_{k=1}^{K}\sum_{l=1}^{K} y_{Km+k,Km+l} = (2-K)^2 \quad (0 \le m \le M-1). \end{aligned} \tag{13}$$

To take constraints (10) into account, add the following:

$$\begin{aligned} \mathbf{a}_u^\top \mathbf{y} &= 2b_u - \mathbf{1}^\top \mathbf{a}_u && (u = 1, \dots, U), \\ \mathbf{a}_u^\top Y' \mathbf{a} &= (2b_u - \mathbf{1}^\top \mathbf{a}_u)^2 && (u = 1, \dots, U), \\ \mathbf{c}_u^\top \mathbf{y} &\le 2d_u - \mathbf{1}^\top \mathbf{c}_u && (v = 1, \dots, V), \end{aligned} \tag{14}$$

where $\mathbf{y} \in \mathbb{R}^{KM}$ and $Y' \in \mathrm{Sym}_{KM}$ are the bottom left $(KM \times 1)$-submatrix and the bottom right $(KM \times KM)$-submatrix of Y, respectively. Then, the problem (13) is a relaxation of the original BQP, as the following theorem states:

Theorem 3.1. *Let $\tilde{Y}$ be an optimal solution of* (13)*. Then, the optimal value $A \bullet \tilde{Y}$ is greater than or equal to the optimal value of* (8)*. Further, if $\tilde{Y} = [\tilde{y}_{ij}]$ satisfies $\tilde{y}_{0i} \in \{-1, 1\}$ for $1 \le i \le KM$, then $z_i = (\tilde{y}_{0i} + 1)/2$ gives an optimal solution of* (8)*.*

[2] Sym_n denotes the set of all real symmetric matrices of size n.
[3] Let us denote an inner product over Sym_n by $X \bullet Y = \sum_{i=1}^{n}\sum_{j=1}^{n} X_{ij}Y_{ij}$ for $X, Y \in \mathrm{Sym}_n$.

Algorithm 1 Randomized Search Rounding

Input: $\tilde{Y}, T$
Output: $\tilde{\mathbf{z}}$

1: For each s, pick i from I_s with probability $(\tilde{y}_{0,i} + 1)/2$ randomly, and set $\tilde{\mathbf{z}}$ by (15).
2: **do**
3: Let $I_{\mathrm{vio}} \subseteq \{1, \dots, n\}$ be defined by:
$$I_{\mathrm{vio}} = \{i \mid \exists \text{ violated constraint w.r.t. } z_i\}$$
4: Pick s such that $I_{\mathrm{vio}} \cap I_s \neq \emptyset$ randomly, and pick i from I_s with probability $(\tilde{y}_{0,i} + 1)/2$. Then, set $\tilde{\mathbf{z}}$ by (15) for a given I_s.
5: **while** $|I_{\mathrm{vio}}| > 0$

This theorem holds even when (8) and (13) contains constraints (10) and (14), respectively. It is worth noting that this theorem gives not only approximate solutions of (8) but also upper bounds of the maximum total revenue, which is the optimal value of the original problem. This property is used in the below sections.

3.2 Rounding

Once we obtain the optimal solution $\tilde{Y}$, we construct a feasible solution of the original BQP problem by using rounding techniques. The proposed rounding method is a randomized search which is summarized in Algorithm 1.[4] A key idea behind the randomized search is to interpret $y_{0,Km+k}$ as probability; from the constraint in (13), $[(y_{0,Km+k} + 1)/2]_{k=1}^{K}$, can be regarded as probability distribution on $\{1, \dots, K\}$. Then, we pick an index i by proportional to the probability $(\tilde{y}_{0,i} + 1)/2$ and set $\mathbf{z}$ as follows:

$$z_j = \begin{cases} 1 & j = i \\ 0 & j \in I_s \setminus \{i\} \end{cases} \tag{15}$$

The higher the value of $y_{0,Km+k}$ is, the more likely the corresponding value of z_j is to be 1. We repeat this procedure until it returns a feasible solution of the original problem. Empirically, this procedure successfully find a good solution in most cases, as shown in Sections 4 and 5.

3.3 Approximation Quality

It is practically important to evaluate the quality of the solution obtained by the SDP relaxation method. By the SDP relaxation method, we obtain $\tilde{\mathbf{z}}$ and $\tilde{Y}$ which immediately give $f(\tilde{\mathbf{z}})$ and $A \bullet \tilde{Y}$. Consider the following inequality:

$$f(\tilde{\mathbf{z}}) \le f(\mathbf{z}^*) \le A \bullet \tilde{Y}, \tag{16}$$

where $\mathbf{z}^*$ is the optimal solution of the original problem. The first inequality holds by the optimality of $\mathbf{z}^*$ and the second one come from Theorem 3.1. Then, Eq. (16) gives us a lower bound of the approximation ratio of the obtained solution as $\delta(\tilde{\mathbf{z}}, \tilde{Y}) := \frac{f(\tilde{\mathbf{z}})}{A \bullet \tilde{Y}} \le \frac{f(\tilde{\mathbf{z}})}{f(\mathbf{z}^*)} \le 1$. Although we cannot obtain the true optimal solution $\mathbf{z}^*$ since the problem is NP-hard, we can estimate the quality of the obtained solution $\tilde{\mathbf{z}}$ by checking the value of $\delta(\tilde{\mathbf{z}}, \tilde{Y})$. Note that the approximation ratio can be calculated by taking a ratio between the original objective value and the relaxed objective value, and

[4] We define $I_s = \{K(s-1) + 1, \dots, Ks\}$ for $s = 1, \dots, M$.

hence it can be defined for the other relaxation methods like the MILP relaxation.

4 SIMULATION STUDY

This section investigates detailed behaviors of the proposed method on the basis of artificial simulation. We used SDPA 7.3.8[5] for SDP problems. To solve MILP and MIQP problems, we used GUROBI Optimizer 6.0.4[6], a state-of-the-art commercial solver for mathematical programming. All experiments were conducted in a machine equipped with Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz, 768GB RAM. We limited all processes to single CPU core.

4.1 Simulation Model

The sales quantity q_m of the m-th product was generated from the following regression model:

$$q_m = \alpha^*_m + \sum_{m'=1}^{M} \sum_{d=1}^{D} \beta^*_{mm'd} \phi_d(p_{m'}) + \epsilon \tag{17}$$

where $\epsilon \sim N(0, \sigma^2)$, $\{\phi_d(x)\} = \{x, x^2, 1/x\}$ and the true coefficients $\{\alpha^*_m\}$ and $\{\beta^*_{mm'd}\}$ were generated by Gaussian random numbers. The price candidates were $\{0.8, 0.85, 0.9, 0.95, 1\}$ ($K = 5$) and cost was $c_m = 0.7$. Let us denote by $f^*(\mathbf{z})$ the objective function of (8) (gross profit function (2)) computed with the true parameter.

4.2 Scalability Comparison of BQP Solvers

We compared the SDP relaxation method with the MIQP solver implemented in GUROBI which can directly solve the BQP problem and the MILP relaxation method [16]. We denote them by SDPrelax, MIQPgrb and MILPrelax, respectively. For each solver, we obtain the relaxed objective value $\bar{f}^*$ and the original objective value $f^*(\tilde{\mathbf{z}})$ that satisfy $f^*(\tilde{\mathbf{z}}) \leq f^*(\mathbf{z}^*) \leq \bar{f}^*$.[7] Given a problem, the performance of solvers was measured by computational efficiency and difference of $f^*(\tilde{\mathbf{z}})$ and $\bar{f}^*$. Note that $f^*(\tilde{\mathbf{z}}) = \bar{f}^*$ implies that $\tilde{\mathbf{z}}$ is an optimal solution.

Fig. 1 shows the results with a small number of products, i.e., $M = 1, 2, \ldots, 15$. We observed that:

- In the top figure, SDPrelax obtained the optimal solution in only several seconds with $M = 15$, and we confirmed the advantage in computational efficiency of SDPrelax against the others.
- In the top figure, the computational cost of MIQPgrb and MILPrelax exponentially increased over the problem size, and both of them reached the maximum time limitation (one hour) at $M = 11$.
- In the bottom figure, $\bar{f}^*$ and $f^*(\tilde{\mathbf{z}})$ of SDPrelax were almost the same, which implied that SDPrelax obtained nearly-optimal solutions.
- In the bottom figure, $\bar{f}^*$ for MIQPgrb and MILPrelax rapidly increased from $M = 11$. This was because we terminated the optimization by one hour limit. Further, the upper bound of MILPrelax was looser than the others. On the other hand, $f^*(\tilde{\mathbf{z}})$ for MIQPgrb and MILPrelax were close to that of SDPrelax (nearly-optimal) and thus they might be able to obtain a practical solution with heuristic early stopping though it is not trivial to determine when we stop the algorithms.

[5] http://sdpa.sourceforge.net/
[6] http://www.gurobi.com/
[7] $\tilde{z}$ and z^* are computed and exact solutions for (8), respectively.

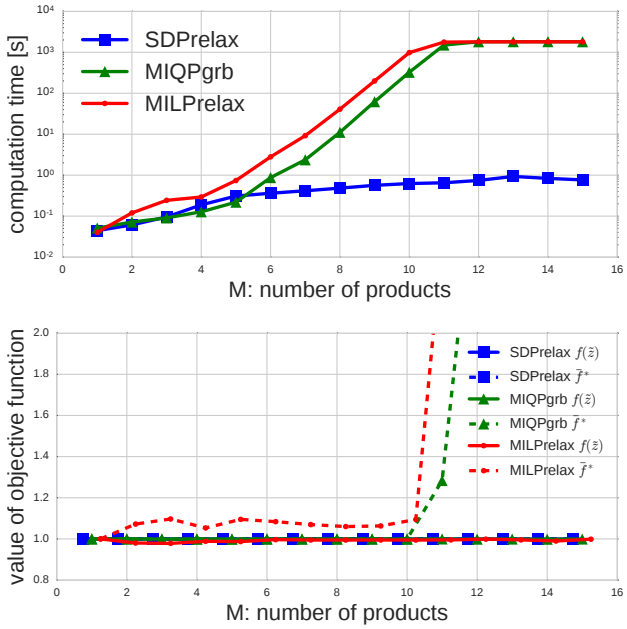

Figure 1: Comparisons of SDPrelax, MIQPgrb and MILPrelax with a small number of products. The horizontal axis represents the number of products M. The vertical axes represent computational time (top) and $f^*(\tilde{\mathbf{z}})$ and $\bar{f}^*$ objective values (bottom). For the bottom, values are normalized such that $f^*(\tilde{\mathbf{z}}) = 1$ for SDPrelax.

Next, we conducted experiments with large problems by aiming to verify 1) scalability and solution quality of SDPrelax for larger problems, and 2) solution qualities of MIQPgrb and MILPrelax by fixing the computational time budget. The second point was investigated since the bottom figure of Fig. 1 indicates that MIQPgrb and MILPrelax might reach nearly-optimal solution much earlier than the algorithm termination. In order to evaluate it, we aborted MIQPgrb and MILPrelax with the same computational time budget (i.e., we terminated them when the computational time reached that of SDPrelax.)

Fig. 2 shows the results with a large number of products. We observed that:

- In the top figure, the computational time of SDPrelax fits well to a cubic curve w.r.t. M, so its practical computational order might be $O(M^3)$. For 250 products, it took only 6 minutes to obtain the optimal solution. This is sufficiently fast for scenarios such as price planning for retail stores.
- In the bottom figure, the solution of SDPrelax was still nearly-optimal even if the number of products increased up to $M = 250$, i.e., $f^*(\tilde{z})/\bar{f}^*$ of SDPrelax was at least 0.98. This indicates the SDP relaxation is tight enough to obtain practically good solutions. On the other hand, under the computational budget constraint, the solutions of MIQPgrb and MILPrelax were significantly worse than that of SDPrelax. Further, over the problem size, their solutions became even worse.

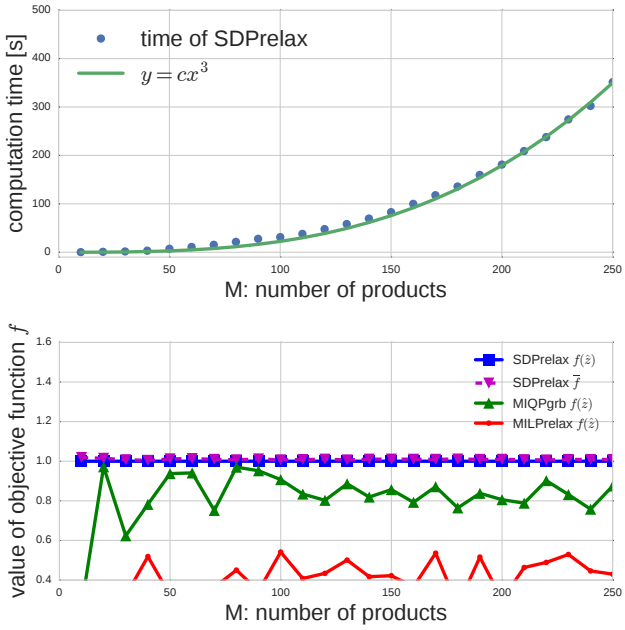

Figure 2: Comparisons of SDPrelax, MIQPgrb and MILPrelax with a large number of products. The top figure shows the computational time of SDPrelax over the number of products M. The bottom figure compares values of $f^*(\tilde{z})$ for the three methods by restricting their computational time to be that of SDPrelax. For the bottom, values are normalized such that $f^*(\tilde{z}) = 1$ for SDPrelax.

These results show that, for solving our BQPs (8), SDPrelax significantly outperforms the other BQP solvers in both scalability and optimization accuracy. Furthermore, SDPrelax returns smaller upper bound $\bar{f}^*$ of exact optimal value, which means that it gives better guarantees on accuracy of the computed solution.

4.3 Influence of Parameter Estimation

In practice, we do not know the true parameters and have to estimate them from a training dataset denoted by $\mathcal{D} = \{\mathbf{p}_n, \mathbf{q}_n\}_{n=1}^N$. In this experiment, given $\mathcal{D}$, we estimated regression coefficients $\{\hat{\alpha}_m\}$ and $\{\hat{\beta}_{mm'd}\}$. The gross profit function with the estimated parameter is the denoted by $\hat{f}(\mathbf{z})$. Let $\hat{\mathbf{z}}$ denote the solution on the estimated objective: $\hat{\mathbf{z}} = \arg\max_{\mathbf{z}\in\mathcal{Z}} \hat{f}(\mathbf{z})$. This section investigates how optimization results are affected by the estimation. Note that the problem is NP-hard and we can obtain neither $\mathbf{z}^*$ nor $\hat{\mathbf{z}}$. However, the results in the previous subsection indicated SDPrelax obtains nearly-optimal solutions, so this section considers the solutions of SDPrelax as $\mathbf{z}^*$ and $\hat{\mathbf{z}}$. In the following, let δ stand for the relative magnitude of the noise in data: $\delta := \sqrt{\sigma^2/E[q_m^2]}$, where σ^2 is the variance of the noise ϵ in (17). Roughly speaking, δ is the level of prediction or estimation error that we cannot avoid.

We have three important quantities of practical interests: 1) ideal gross profit: $f^*(\mathbf{z}^*)$, 2) actual gross profit: $f^*(\hat{\mathbf{z}})$, and 3) predicted gross profit: $\hat{f}(\hat{\mathbf{z}})$. It is worth noting that, if the true model is a linear regression like this setting, the following relationship holds:[8]

$$f^*(\hat{\mathbf{z}}) \le f^*(\mathbf{z}^*) \le E_{\mathcal{D}}[\hat{f}(\hat{\mathbf{z}})]. \tag{18}$$

We omit the proof for space limitation. This inequality means that the predicted gross profit $\hat{f}(\hat{\mathbf{z}})$ tends to *overestimate* the actual gross profit $f^*(\hat{\mathbf{z}})$, in terms of the expectation value. This overestimation is a similar phenomenon to the overfitting in the supervised learning; the solution $\hat{\mathbf{z}}$ overfits to the estimated objective $\hat{f}(\mathbf{z})$. A reasonable way to avoid overestimation is the hold-out validation, in which we compute another gross profit function $\hat{f}'(\mathbf{z})$ from a different dataset $\mathcal{D}'$ and evaluate the value of the gross profit by $\hat{f}'(\hat{\mathbf{z}})$ instead of $\hat{f}(\hat{\mathbf{z}})$. In our experiment, we generated $\mathcal{D}$ and $\mathcal{D}'$ by (17) independently, where they have the same number of samples, and constructed $\hat{f}$, $\hat{f}'$ from $\mathcal{D}$, $\mathcal{D}'$, respectively.

The inequality (18) yields three natural questions:

- How close $f^*(\hat{\mathbf{z}})$ and $f^*(\mathbf{z}^*)$ are? In other words, how well does our estimated optimal strategy perform?
- How close $f^*(\hat{\mathbf{z}})$ and $\hat{f}(\hat{\mathbf{z}})$ are? In other words, can we predict actual profit in advance by our estimated optimal value?
- How close $f^*(\hat{\mathbf{z}})$ and $\hat{f}'(\hat{\mathbf{z}})$ are? In other words, can we predict actual profit in advance by the hold-out validation?

Fig. 3 illustrates behaviors of $f^*(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$, $\hat{f}(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ and $\hat{f}'(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ in different settings. We observed that:

- In the top figure, the overestimation of the predicted gross profit got linearly large along with increasing M and it became over 15% (i.e. $\hat{f}(\hat{\mathbf{z}})/f^*(\mathbf{z}^*) \ge 1.15$) with forty products ($M = 40$). This result means that over-flexible models (i.e. the number of products that we use in prediction models and that we can optimize) significantly overestimate the gross profit. This is not preferable since users cannot appropriately assess the risk of machine-generated price strategies. In order to mitigate this issue, the next section investigates to incorporate a sparse learning technique in learning regression models so that the effective model flexibility stays reasonable even if M is large.
- In the middle figure, along with increasing noise level, the gap between $f^*(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ and $\hat{f}(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ increased. This result verifies a natural intuition: if the estimation errors of the regression models are large, the modeling error in the BQP problem becomes large and the solution becomes unreliable. Therefore, achieving fairly good predictive models (say the error rate is less than 20% in this setting) is critical in this framework.
- In the bottom figure, along with increasing training data, the gap between $f^*(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ and $\hat{f}(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ decreased and we confirmed that increased data size made estimation accurate and eventually made optimization accurate.
- In all the figures, the values $\hat{f}'(\hat{\mathbf{z}})$ of the hold-out validation yielded a better estimation of the actual gross profit $f^*(\hat{\mathbf{z}})$ than $\hat{f}(\hat{\mathbf{z}})$ in terms of the expectation value. The average of $\hat{f}'(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ almost coincided with that of $f^*(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ in all cases, though the standard deviation of $\hat{f}'(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ is larger than that of $\hat{f}(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ and about twice as large as $f^*(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$.

In summary, the actual profit $f^*(\hat{\mathbf{z}})$ compared to the ideal profit $f^*(\mathbf{z}^*)$ got worse when the more flexible model (with more parameters) was used, when the training data contained larger degree of

[8] $E_{\mathcal{D}}$ stands for expectation w.r.t. $\mathcal{D}$.

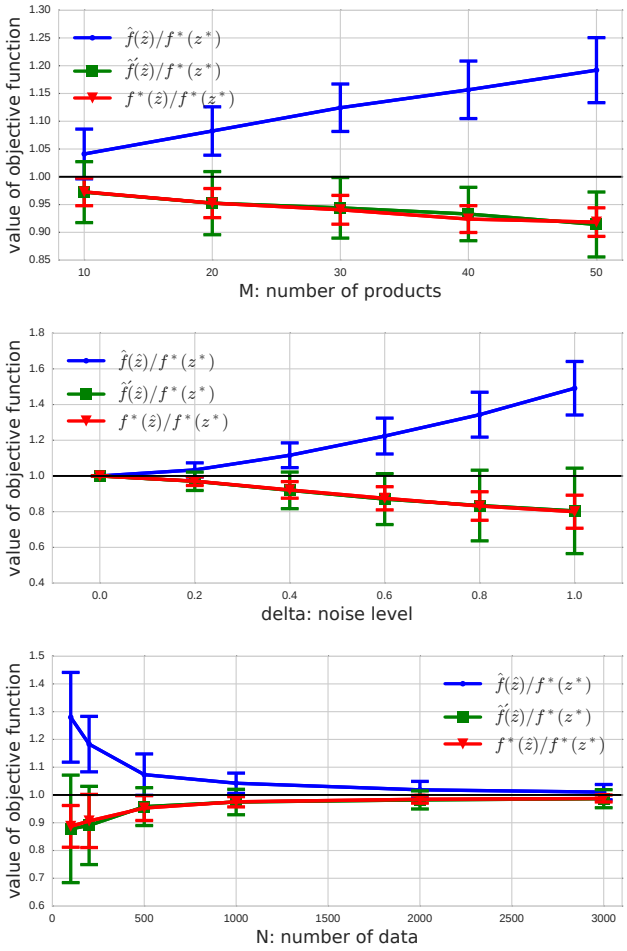

Figure 3: Value of $\hat{f}(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ (blue), $\hat{f}(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ (green) and $f^*(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ (red) for different setting. Dot and error bar mean the average and standard deviation of 100 times trial. Top: $\delta = 0.2$, $N = 1000$. Middle: $M = 10$, $N = 1000$. Bottom: $M = 10$, $\delta = 0.2$.

errors, and when the number of training data was smaller. Further, as the achieved actual profit $f^*(\hat{\mathbf{z}})$ got poorer, the degree of overestimation of $\hat{f}(\hat{\mathbf{z}})$ grew. On the other hand, the hold-out validation $\hat{f}'(\hat{\mathbf{z}})$ yielded almost unbiased estimation of the actual profit $f^*(\hat{\mathbf{z}})$.

5 REAL WORLD RETAIL DATA

5.1 Data and Experimental Settings

We applied our prescriptive price optimization method to real retail data in a middle-size supermarket located in Tokyo[9] [28]. We selected regularly-sold 50 beer products[10] that vary in different brands and different packages. The data range is two years from 2013/01 to 2014/12. In addition to 50 linear price features, we employed "day of the week" features (seven, binary) for weekly trend, "month" (12, binary) for seasonal trend, weather and temperature forecasting features (forecast of daylight, humidity, cloudiness, and temperature) and auto-correlations feature (a week ago sales) as external features. The price candidates $\{P_{mk}\}_{k=1}^{5}$ were generated by equally splitting the range $[P_{m1}, P_{m5}]$ where P_{m1} and P_{m5} are the highest and lowest prices of the m-th product in the historical data. We assumed that the cost $c_m = 0$ for simplicity.

We used the data of the year 2013 (365 samples, the half of all the data) for training regression model. On the basis of this regression model, we constructed the estimated gross profit function $\hat{f}_d$ of each day d in August 2014. We computed the optimal pricing strategy $\hat{\mathbf{z}}_d$ by maximizing $\hat{f}_d(\mathbf{z})$ by the SDP relaxation method, subject to the constraint that the number of discounted products is ten. Such a constraint often appears in a real business, and is expressed as $\sum_{m=1}^{M} z_{m1} = M - 10$ in our optimization problem. Further, we constructed another estimated gross profit function $\hat{f}'_d$ from the data of the year 2014 (the hold-out samples), which was used for evaluating the computed pricing strategy $\hat{\mathbf{z}}_d$. In the following, we call $\hat{f}_d(\hat{\mathbf{z}}_d)$ the optimal value and $\hat{f}'_d(\hat{\mathbf{z}}_d)$ the hold-out validation value. From the simulation experiments in Section 4.3, we can suppose that the hold-out validation gives a good estimation of the actual profit, though the optimal value overestimates it.

5.2 Influence of Model Complexity

As we have discussed in the previous section, if we use all 50 products in predictions, the predicted gross profit might significantly overestimate the actual one. However, it can be reasonably assumed that sales of a certain product is affected by only a limited number of products, but not all products. Hence, it is expected that we can mitigate the overestimation issue by learning such a sparse cross-price structure. In addition, there are influential variables that must be taken into account, e.g. the price of the top seller products. In practice, however, we observed that such variables could be omitted by the orthogonal matching pursuit method (OMP) because of multicollinearity. In order to manage the issue, we first applied the standard least square estimation (LS) to the top-5 products and then applied OMP to the residual to extract additional variables including external variables. When using OMP, we can chose the number of the nonzero coefficients in the regression models.

Figure 4 shows the estimated gross profit for the computed prices, for different number of nonzero coefficients. The vertical axis stands for the average gross profit per day in August 2014, and the horizontal axis stands for the number of nonzero coefficients in the regression models. We observed that the value of the optimal value (blue) grew as the number of nonzero coefficients, denoted by s, got increased, i.e., as the regression model got more flexible. From the discussion in Section 4.3, this is considered to come from the effect of the overestimation. In fact, the value of the hold-out validation (green) reached the peak at $s = 13$, and got decreased for $s > 13$. This result supports our hypothesis that not all products have cross elasticity and restricting flexibility gives better profit. The experimental results in the following were given under the setting of $s = 13$.

[9]The data has been provided by KSP-SP Co., LTD, http://www.ksp-sp.com.
[10]The data contains sales history of beer, bakery, milk and tofu products, and we chose beers since they have larger cross-price effects than the others in general.

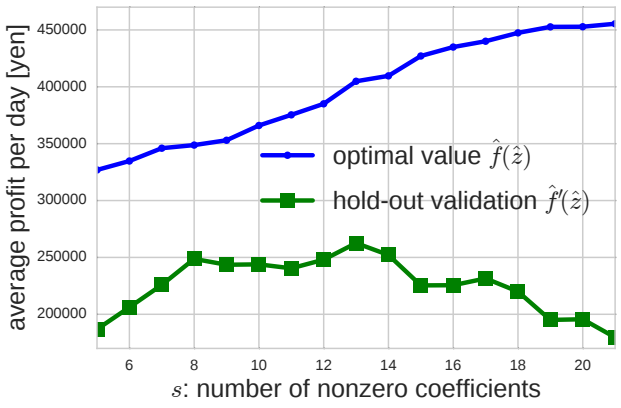

Figure 4: Estimated gross profit for different numbers of used features in the regression.

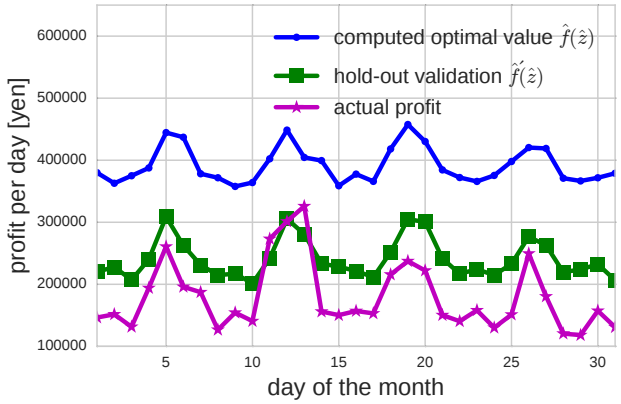

Figure 5: Estimated profit for the computed strategy (blue, green) and the actual profit calculated from the past data (magenta) of each day in August 2014.

5.3 Improvement in Gross Profit

Figure 5 shows the estimated profit and the actual profit of each day. The horizontal axis stands for the day of the month, from the day first to the day 31st. Note that the actual profit in this figure does not means the profit for the computed strategy, but means the profit computed from the past data of prices and sales. We observed that the value of the actual profits (magenta) reached peaks on weekends.[11] In almost all days, the gross profit estimated by the hold-out validation (green) was better than the actual profit. In particular, the degree of the improvement in weekdays is larger than in weekends. The sums of $\hat{f}(\hat{\mathbf{z}})$, $\hat{f}'(\hat{\mathbf{z}})$ and the actual profit over all the days in August 2014 were 1.21×10^7, 7.46×10^6, and 5.57×10^6, respectively. This means that, it can be expected that our optimization framework yields about $(7.46 - 5.57)/5.57 \approx 0.339 =$ 33.9% improvement of the profit, though we need to note that these value contains estimation error.

[11] The days 5th, 12th, 19th, and 26th were Sunday. The day 13th was a national holiday.

5.4 Interpretation of Derived Price Strategy

Table 1 shows all 50 products and the average prices, total sales quantity [unit] and total sales revenue [yen] of each product. The values after optimization were estimated by the hold-out validation. This table provides much richer insights to understand how machine tried to maximize profit of this supermarket. Let us here summarize notable points:

- Prices of 15 products out of 50 products were decreased by the price optimization. Further, sales quantities of 11 products out of these 15 products got increased.
- In particular, Asahi Superdry 350ml (id = 24, 28), the most popular product, were discounted and their sales grew to about 2.5 times. This improvement made a big impact on the gross profit. The share of Asahi Superdry 350ml in this supermarket grew from approximately 15% to 28%.
- Interestingly, the sales of Kirin Ichibanshibori (id = 34), the second most popular product, got decreased even though its price got decreased. It can be interpreted that this products and Asahi Superdry are competing, and hence, the increase in the sales of Asahi Superdry caused the decrease in the sales of Kirin Ichibanshibori.

6 SUMMARY

This paper presented a novel flexible framework of prescriptive price optimization. We formulated BQP problems for the price optimization, and proposed a fast solver using an SDP relaxation. Simulation experiments demonstrated that the proposed algorithm performs much better than state-of-the-art methods in terms of scalability and quality of solutions. Empirical evaluations were conducted also with a real retail dataset, and the result indicated that the derived price strategy could improve approximately 30% of gross profit. A challenging future work would be to avoid effects of estimation error in real application.

REFERENCES

[1] D. Bienstock and N. ÖZbay. Computing robust basestock levels. *Discrete Optimization*, 5(2):389–414, 2008.
[2] A. Billionnet, S. Elloumi, and M. C. Plateau. Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation: The QCR method. *Discrete Applied Mathematics*, 157(6):1185–1197, 2009.
[3] J. Burgschweiger, B. Gnädig, and M. C. Steinbach. Optimization models for operative planning in drinking water networks. *Optimization and Engineering*, 10:43–73, 2009.
[4] F. Caro and J. Gallien. Clearance pricing optimization for a fast-fashion retailer. *Operations Research*, 60(6):1404–1422, 2012.
[5] C. Dziekan. The analytics journey. *Analytics*, 2010.
[6] K. J. Ferreira, B. H. A. Lee, and D. Simchi-Levi. Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing & Service Operations Management*, pages 69–88, 2015.
[7] D. Fooladivanda and J. A. Taylor. Optimal pump scheduling and water flow in water distribution networks. In *IEEE 54th Annual Conference on Decision and Control*, pages 5265–5271, 2015.
[8] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM*, 42(6):1115–1145, 1995.
[9] C. Helmberg and F. Rendl. Solving quadratic (0,1)-problems by semidefinite programs and cutting planes. *Mathematical Programming*, 82(3):291–315, 1998.
[10] S. Ito and R. Fujimaki. Large-scale price optimization via network flow. In *Advances in Neural Information Processing Systems*, pages 3855–3863, 2016.
[11] M. Kisialiou and Z.-Q. Luo. Probabilistic analysis of semidefinite relaxation for binary quadratic minimization. *SIAM Journal on Optimization*, 20(4):1906–1922, 2010.
[12] R. Klein. *Revenue Management*. Springer, 2008.

Table 1: List of beer products and their optimized prices/sales. (Notable parts are highlighted in boldfaces.)

id	product name	price original	price optimal	price increase rate	sales[unit] original	sales[unit] hold-out	sales[unit] increase rate	sales[yen] original	sales[yen] hold-out	sales[yen] increase rate
1	Kirin lager beer can 350ml	252	254	0.4%	266	408	53.2%	67021	103816	54.9%
2	Kirin lager beer can 350ml * 6	1116	1063	-4.8%	107	235	119.7%	119413	252044	111.1%
3	Suntory the premium malts 500ml	283	236	-16.9%	568	742	30.7%	160630	187541	16.8%
4	Kirin Ichibanshibori draft beer 250ml * 6	982	986	0.4%	32	91	185.0%	31444	90042	186.4%
5	Kirin lager beer can 350ml	188	189	0.5%	347	343	-1.2%	65032	64516	-0.8%
6	Budweiser can 350ml	179	180	0.2%	410	536	30.6%	73177	96326	31.6%
7	Asahi Oriondraft can 350ml	188	171	-8.9%	293	335	14.4%	54987	55701	1.3%
8	The premium malts tumbler 350ml * 6	1271	1287	1.3%	213	534	150.8%	265652	687836	158.9%
9	Kirin Ichibanshibori draft beer 135ml * 6	543	543	0.1%	54	45	-15.8%	29255	24700	-15.6%
10	Sapporo can draft black label 135ml	90	91	0.1%	112	148	32.5%	10106	13441	33.0%
11	Asahi Superdry can 500ml	253	254	0.2%	1115	1229	10.3%	281975	311619	10.5%
12	Corona extra bottle bin 355ml	255	256	0.3%	115	-2	-101.5%	29305	-425	-101.5%
13	Kirin Ichibanshibori draft beer 250ml	161	162	0.4%	246	269	9.5%	39671	43640	10.0%
14	Asahi Superdry can 500ml * 6	1470	1502	2.1%	173	234	35.5%	252880	351479	39.0%
15	Echigobeer Pilsner can 350ml	264	251	-5.1%	93	56	-39.5%	24549	14122	-42.5%
16	Sapporo Ebisu beer can 350ml	207	208	0.5%	536	682	27.3%	110825	141937	28.1%
17	The premium malts can 500ml * 6	1669	1697	1.6%	73	150	105.4%	120052	254481	112.0%
18	Kirin Ichibanshibori draft beer can 135ml	90	88	-2.4%	286	1151	302.3%	25777	103322	300.8%
19	Kirin Ichibanshibori draft beer can 350ml	188	189	0.5%	909	1703	87.3%	170451	321296	88.5%
20	Asahi Superdry can 135ml * 6	541	543	0.4%	64	512	700.2%	34437	278084	707.5%
21	Sapporo Ebisu beer can 350ml * 6	1229	1146	-6.7%	268	474	76.9%	327496	542988	65.8%
22	Sapporo Ebisu beer can 250ml	171	172	0.3%	249	188	-24.6%	42607	32243	-24.3%
23	Sapporo draft beer black label can 350ml	187	189	0.6%	523	510	-2.5%	97959	96164	-1.8%
24	**Asahi Superdry can 350ml**	187	179	-4.7%	1399	3623	**159.0%**	261914	664166	153.6%
25	Asahi Superdry can 250ml	162	162	0.3%	235	832	253.9%	37693	134728	257.4%
26	Kirin Hartland beer bin 500ml	266	260	-2.4%	221	292	32.2%	58820	76288	29.7%
27	Sapporo draft beer black label can 350ml * 6	1108	1002	-9.6%	173	311	79.5%	189805	311399	64.1%
28	**Asahi Superdry can 350ml * 6**	1101	1062	-3.6%	555	1395	**151.3%**	602582	1463801	142.9%
29	Gingakougen beer of wheat 350ml	245	246	0.3%	181	-111	-161.4%	44384	-27342	-161.6%
30	Sapporo draft beer black label can 500ml	253	228	-9.8%	673	275	-59.1%	170287	57993	-65.9%
31	Asahi Superdry can 250ml * 6	970	972	0.2%	32	25	-23.4%	31034	23838	-23.2%
32	Suntory the premium malts 350ml	217	209	-4.0%	535	148	-72.3%	115965	31569	-72.8%
33	Asahi Superdry can 135ml	90	91	0.9%	309	743	140.6%	27733	67469	143.3%
34	**Kirin Ichibanshibori draft beer can 350ml * 6**	1113	1068	-4.1%	410	255	**-37.8%**	454986	272913	-40.0%
35	Suntory the premium malts can 250ml	177	177	0.3%	283	181	-36.0%	49922	31960	-36.0%
36	Kirin Ichibanshibori draft beer can 500ml * 6	1486	1502	1.0%	112	119	6.4%	166578	178605	7.2%
37	Asahi Superdry Dryblack 500ml	253	254	0.1%	221	202	-8.4%	55832	51331	-8.1%
38	Kirin Ichibanshibori draft beer bin 633ml	313	313	0.0%	34	146	328.7%	10623	45506	328.4%
39	Budweiser bin LNB 330ml	188	189	0.4%	136	267	96.0%	25527	50182	96.6%
40	Asahi Superdry bin 633ml	295	295	0.1%	145	212	45.9%	42715	62417	46.1%
41	Heineken bin 330ml	217	224	2.9%	157	171	8.8%	34085	38325	12.4%
42	Kirin Ichibanshibori stout can 350ml	188	184	-2.3%	276	443	60.4%	51845	82117	58.4%
43	Kirin lager beer 500ml * 6	1495	1452	-2.9%	41	83	102.6%	61316	118582	93.4%
44	Sapporo Ebisu beer can 500ml	272	273	0.3%	441	231	-47.5%	119669	63598	-46.9%
45	Sapporo Ebisu beer can 500ml * 6	1595	1599	0.2%	82	229	179.3%	130834	366380	180.0%
46	Heineken can 350ml	217	224	2.9%	155	414	166.9%	33646	92559	175.1%
47	Asahi Superdry Dryblack 350ml	188	189	0.4%	321	173	-46.2%	60245	32479	-46.1%
48	Echigo Premium red ale 350ml	264	265	0.4%	107	-20	-118.7%	28226	-5310	-118.8%
49	Sapporo draft beer black label can 500ml * 6	1483	1502	1.3%	66	106	61.0%	97147	159229	63.9%
50	Kirin Ichibanshibori draft beer can 500ml	253	254	0.2%	675	790	17.1%	170730	200026	17.2%

[13] D. Koushik, J. A. Higbie, and C. Eister. Retail price optimization at intercontinental hotels group. *Interfaces*, 42(1):45–57, 2012.

[14] T. P. Kunz and S. F. Crone. Demand models for the static retail price optimization problem - a revenue management perspective. *SCOR*, pages 101–125, 2014.

[15] S. Lee. *Study of demand models and price optimization performance.* PhD thesis, Georgia Institute of Technology, 2011.

[16] R. M. Lima and I. E. Grossmann. On the solution of nonconvex cardinality boolean quadratic programming problems, 2012.

[17] Y. Liu and X. Guan. Purchase allocation and demand bidding in electric power markets. *Power Systems, IEEE Transactions on*, 18(1):106–112, 2003.

[18] Z.-Q. Luo, W.-K. Ma, A. M.-C. So, Y. Ye, and S. Zhang. Semidefinite relaxation of quadratic optimization problems. *IEEE Signal Processing Magazine*, 27(3):20, 2010.

[19] A. Marshall. *Principles of Economics.* Library of Economics and Liberty, 1920.

[20] M. Natter, T. Reutterer, and A. Mild. Dynamic pricing support systems for diy retailers - a case study from austria. *Marketing Intelligence Review*, 1:17–23, 2009.

[21] R. L. Phillips. *Pricing and Revenue Optimization.* Stanford University Press, 2005.

[22] G. v. Ryzin and S. Mahajan. On the relationship between inventory costs and variety benefits in retail assortments. *Management Science*, 45(11):1496–1509, 1999.

[23] K. Shaloudegi, A. György, C. Szepesvári, and W. Xu. SDP relaxation with randomized rounding for energy disaggregation. In *Advances in Neural Information Processing Systems*, pages 4979–4987, 2016.

[24] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

[25] K.-C. Toh, M. J. Todd, and R. H. Tütüncü. SDPT3 – a MATLAB software package for semidefinite programming, version 1.3. *Optimization methods and software*, 11(1-4):545–581, 1999.

[26] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.

[27] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1996.

[28] J. Wang, R. Fujimaki, and Y. Motohashi. Trading interpretability for accuracy: Oblique treed sparse additive models. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1245–1254, 2015.

[29] A. Yabe, S. Ito, and R. Fujimaki. Robust quadratic programming for price optimization. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, 2017.

[30] M. Yamashita, K. Fujisawa, and M. Kojima. Implementation and evaluation of sdpa 6.0. *Optimization Methods and Software*, 18(4):491–505, 2003.