Optimization Beyond Prediction: Prescriptive Price Optimization

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

Ryohei Fujimaki
NEC Corporation
rfujimaki@nec-labs.com

May 19, 2016

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, which is closely related to the Goemans-Williamson's Max-Cut approximation. 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 8.2 % of gross profit of those products.

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* [4] 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 in-

herent regularities behind data, and 2) forecasted values for key performance indicators. While 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 [13], smart water management [2, 6], etc. The focus of this paper is on the later 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 [14] and cross price effects (a.k.a. cannibalization) [15, 17]. 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* [16, 8], which has been actively studied in areas of marketing, economics, operation research. Traditional revenue management literature has been focused on such a problem as markdown optimization (a.k.a. dynamic pricing) where a perishable product is priced over a finite selling horizon. Our focus is more on static but simultaneous optimization of many products using machine learning based predictions. Although there are several existing studies such as fast-fashion retailer [3], online retailer [5], hotel room [9, 11], etc. (a comprehensive survey is given by [10]). However, existing methods have strong restrictions in demand modeling capability, e.g. one does not consider cross-price effects, 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.

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.

Fast BQP Solver by SDP Relaxation: BQP problems are, in general, NP-hard, and we need to use an approximation (or relaxation) method. Although BQP problems are often solved using mixed-integer programming, computational costs with mixed-integer relaxation methods exponentially increase with increasing problem size, and they are not applicable to large scale problems. This paper proposes an alternative relaxation method that employ semi-definite programming (SDP) [22], by employing an idea of the Goemans-Williamson's MAX-CUT approximation [7]. Although our target focuses on prescriptive price optimization, we note that our SDP relaxation algorithm is a fast approximation solver for general BQP problems and can be utilized in wide range of applications.

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 8.2% 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 a issue using sparse learning.

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 constrains 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 arbitral but univariate transformations on p_m, which is denoted by f_d $(d = 1, \ldots, D)$. f_d might be designed to incorporate a domain specific relationship between price and demand, such as the law of diminishing marginal utility [14], 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:

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

where $\alpha_m^{(t)}$, $\beta_{mm'd}^{(t)}$, and $\gamma_{md}^{(t)}$ are bias, the coefficient of $f_d(p_{m'})$, 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) [19], or orthogonal matching pursuit (OMP, L_0 regularized) [21] would be applicable with our methodology. The choice of learning algorithm depends on the way relationships among multiple products are to be modelled. Experience shows that these relationships are complicated yet usually sparse in practice[1], and sparse learning algorithms

[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.

might be preferable. More detailed discussions are presented in Section 6.

2.3 Building Optimization Problem

Suppose values of g_d are given for the time step t (e.g. weather forecast), denoted by $g_d^{(t)}$, where $\mathbf{g}^{(t)} = [g_1^{(t)}, \ldots, g_{D'}^{(t)}]^\top$. 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 $\mathbf{q} = [\sum_{t=1}^{T} q_1^{(t)}(\mathbf{p}, \mathbf{g}^{(t)}), \ldots, \sum_{t=1}^{T} q_M^{(t)}(\mathbf{p}, \mathbf{g}^{(t)})]^\top$. Note that $\mathbf{c} = 0$ gives the sales revenue on $\mathbf{p}$.

For later convenience, let us introduce ξ_m and ζ_m as follows:

$$\xi_m(p_m) = \sum_{t=1}^{T} (p_m - c_m)(\alpha_m^{(t)} + \sum_{d=1}^{D'} \gamma_{md}^{(t)} g_d^{(t)}) \tag{3}$$

$$\zeta_{mm'}(p_m, p_{m'}) = \sum_{t=1}^{T} (p_m - c_m) \sum_{d=1}^{D} \beta_{mm'd}^{(t)} f_d(p_{m'}) \tag{4}$$

Then, (2) can be rewritten by:

$$\ell(\mathbf{p}) = \sum_{m=1}^{M} \xi_m(p_m) + \sum_{m=1}^{M} \sum_{m'=1}^{M} \zeta_{mm'}(p_m, p_{m'}). \tag{5}$$

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:

$$\begin{aligned} &\text{Maximize} \quad \ell(\mathbf{p}) \qquad (6)\\ &\text{subject to} \quad p_m \in \{P_{m1}, \ldots, P_{mK}\} \quad (m = 1, \ldots, M). \end{aligned}$$

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 then be expressed in terms of the following constraints:

$$|\{m \in \{1, \ldots, M\} \mid p_m = P_{m1}\}| \geq M - L. \tag{7}$$

The system allows users to input such business requirements, which are then transformed into mathematical constraints, as shown above. The problem (6) is solved with such constraints taken into account. The type of requirements we can deal with is discussed in the next section.

3 BQP Formulation

3.1 Derivation of BQP Problem

The general form (6) 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 (6), 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, $z_{mk} = 1$ and $z_{mk} = 0$ refer to $p_m = P_{mk}$ and $p_m \neq P_{mk}$, respectively, which gives

$$p_m = \sum_{k=1}^{K} P_{mk} z_{mk} \quad (m = 1, \ldots, M). \tag{8}$$

For an arbitral function ϕ, the following equality holds:

$$\phi(p_m) = \sum_{k=1}^{K} \phi(P_{mk}) z_{mk}. \tag{9}$$

Using (9), $\zeta_{mm'}(p_m, p_{m'})$ can be rewritten as follows:

$$\zeta_{mm'}(p_m, p_{m'}) = \mathbf{z}_m^\top Q_{mm'} \mathbf{z}_{m'}, \tag{10}$$

where $\mathbf{z}_m = [z_{m1}, \ldots, z_{mK}]^\top$. We here define $Q_{ij} \in \mathbb{R}^{K\times K}$ by

$$Q_{ij} = \begin{bmatrix} \zeta_{ij}(P_{i1}, P_{j1}) & \zeta_{ij}(P_{i1}, P_{j2}) & \cdots & \zeta_{ij}(P_{i1}, P_{jK}) \\ \zeta_{ij}(P_{i2}, P_{j1}) & \zeta_{ij}(P_{i2}, P_{j2}) & \cdots & \zeta_{ij}(P_{i2}, P_{jK}) \\ \vdots & \vdots & \ddots & \vdots \\ \zeta_{ij}(P_{iK}, P_{j1}) & \zeta_{ij}(P_{iK}, P_{j2}) & \cdots & \zeta_{ij}(P_{iK}, P_{jK}) \end{bmatrix} \tag{11}$$

Similarly, $\xi_i(p_i)$ can be rewritten as follows:

$$\xi_i(p_i) = \mathbf{r}_i^\top \mathbf{z}_i := [\xi_i(P_{i1}), \ldots, \xi_i(P_{iK})]^\top \mathbf{z}_i. \tag{12}$$

By substituting (10) and (12) into (6), (6) can be rewritten as follows:

$$\begin{aligned} \text{Maximize} \quad & f(\mathbf{z}) := \mathbf{z}^\top Q\mathbf{z} + \mathbf{r}^\top \mathbf{z} && (13) \\ \text{subject to} \quad & \mathbf{z} = [z_{11}, \ldots, z_{1K}, z_{21}, \ldots, z_{MK}]^\top \in \{0,1\}^{MK}, \\ & \sum_{k=1}^{K} z_{mk} = 1 \quad (m = 1, \ldots, M), && (14) \end{aligned}$$

where $Q \in \mathbb{R}^{MK\times MK}$ and $\mathbf{r} \in \mathbb{R}^{MK}$ are defined by

$$Q = \begin{bmatrix} Q_{11} & Q_{12} & \cdots & Q_{1n} \\ Q_{21} & Q_{22} & \cdots & Q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{n1} & Q_{n2} & \cdots & Q_{nn} \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} \mathbf{r_1} \\ \mathbf{r_2} \\ \vdots \\ \mathbf{r_n} \end{bmatrix}. \tag{15}$$

The terms $\mathbf{z}^\top Q\mathbf{z}$ and $\mathbf{r}^\top \mathbf{z}$ are correspond to the second and first term of (5), respectively. Problem (13) is referred to as a BQP problem and is known to be NP-hard. Although it would be hard to find a globally optimal solution of (13), its relaxation methods have been well-studied and further, in Section 4, we propose a relaxation method which empirically obtains an accurate solution.

Using $\mathbf{z}$, the constraint (7) can be expressed as follows:

$$\sum_{m=1}^{M} z_{m1} \geq M - L. \tag{16}$$

This is a linear constraint on $\mathbf{z}$ and such linear constraints can be naturally incorporated into (13) (the problem remains to be BQP).

For simplicity, we redefine the indices of the entries of vectors and matrices as follows:

$$\mathbf{z} = (z_i)_{1\leq i\leq KM}, \mathbf{r} = (r_i)_{1\leq i\leq KM} \in \mathbb{R}^{KM}, \tag{17}$$

$$Q = (q_{ij})_{1\leq i,j\leq KM} \in \mathbb{R}^{KM\times KM}. \tag{18}$$

Then the equality constraints (14), can be expressed in the following general form:

$$\sum_{i\in I_m} z_i = 1 \quad (m = 1, \ldots, M), \tag{19}$$

where $\{I_m\}_{m=1}^M$ is a partition of $\{1, 2, \ldots, KM\}$. In summary, we solve the following BQP problem to obtain the price strategy satisfying business requirements:

$$\begin{aligned} \text{Maximize} \quad & f(\mathbf{z}) := \mathbf{z}^\top Q\mathbf{z} + \mathbf{r}^\top \mathbf{z} && (20) \\ \text{subject to} \quad & \mathbf{z} = [z_1, \ldots, z_{KM}]^\top \in \{0,1\}^{KM}, \\ & \textstyle\sum_{i\in I_m} z_i = 1 \quad (m = 1, \ldots, M), \\ & \mathbf{a}_u^\top \mathbf{z} = b_u \quad (u = 1, \ldots, U), \\ & \mathbf{c}_v^\top \mathbf{z} \leq d_v \quad (v = 1, \ldots, V), \end{aligned}$$

where U and V are the number of equality and inequality constraints, respectively, and $\mathbf{a}_u$, $\mathbf{b}_u$, $\mathbf{c}_v$, and $\mathbf{d}_v$ are coefficients of linear constraints. Although we restrict business constraints to be expressed as linear constraints, we emphasize that linear constraints are able to cover a variety of practical business constraints.

3.2 MIP relaxation method

Problem (13) 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 [12].

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

$$\sum_{i=mK+1}^{j} \bar{z}_{ij} + \sum_{i=j+1}^{mK+K} \bar{z}_{ji} = (\sum_{i=mK+1}^{mK+K} z_i - 1) z_j = 0, \tag{21}$$

we can transform (20) into the following MILP problem [12]:

$$\text{Maximize} \sum_{i=1}^{KM}(r_i + q_{ii})z_i + \sum_{i=1}^{KM}\sum_{j=i+1}^{KM}(q_{ij}+q_{ji})\bar{z}_{ij} \tag{22}$$

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

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.

4 SDP Relaxation Using Goemans-Williamson's Approximation

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 the Goemans-Williamson's MAX-CUT approximation algorithm [7], which is abbreviated to the GW algorithm. The GW algorithm is an algorithm for solving the MAX-CUT problem, achieving the best approximation ratio among existing polynomial time algorithms. By noticing the fact that a MAX-CUT problem is a special case of BQP problems, we generalize it to an approximation algorithm for BQPs.

The proposed algorithm consists of the following two steps:

1. Transform the original BQP problem (13) into a semidefinite programming (SDP) problem (38) 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 to the SDP problem.

The optimal solution of the SDP problem can be globally and efficiently computed by a recent advanced solver such as SDPA [23], SDPT3 [20], and SeDuMi [18]. In our experiments, empirical computational time fits a cubic order of problem size.

4.1 Notations

Let us here introduce a few additional notations. Let Sym_n denote a set of all real symmetric matrices of size n as follows:

$$\mathrm{Sym}_n = \{X \in \mathbb{R}^{n\times n} \mid X^\top = X\}. \tag{23}$$

Let us also define an inner product over Sym_n by $X \bullet Y = \sum_{i=1}^n \sum_{j=1}^n X_{ij}Y_{ij}$ for $X, Y \in \mathrm{Sym}_n$. Further, let S^n denote a set of all vectors on a unit ℓ_2 ball in the $n+1$ dimension as follows:

$$S^n = \{\mathbf{x} \in \mathbb{R}^{n+1} \mid \|\mathbf{x}\|_2 = 1\}. \tag{24}$$

4.2 Derivation of SDP Relaxation

Let us first define $\bar{Q}$ by $\bar{Q} := (Q + Q^\top)/2$, which satisfies $\mathbf{x}^\top \bar{Q}\mathbf{x} = \mathbf{x}^\top Q\mathbf{x}$. Let us also consider a transformed variable from $\{0,1\}$ to $\{-1,1\}$ as follows:

$$\mathbf{t} = -\mathbf{1} + 2\mathbf{z} \in \{-1,1\}^{KM} \tag{25}$$

where $\mathbf{t} = [t_1, \ldots, t_{KM}]^\top$ and $\mathbf{1} = (1,1,\ldots,1)^\top$. The objective function of (13) can be then transformed as follows:

$$\mathbf{z}^\top \bar{Q}\mathbf{z} + \mathbf{r}^\top \mathbf{z} = [1\ \mathbf{t}^\top] A \begin{bmatrix} 1 \\ \mathbf{t} \end{bmatrix}, \tag{26}$$

where we define $A \in \mathrm{Sym}_{KM+1}$ by

$$A = \frac{1}{4}\begin{bmatrix} \mathbf{1}^\top \bar{Q}\mathbf{1} + 2\mathbf{r}^\top \mathbf{1} & (\mathbf{r} + \bar{Q}\mathbf{1})^\top \\ \mathbf{r} + \bar{Q}\mathbf{1} & \bar{Q} \end{bmatrix}. \tag{27}$$

Further, the one-of-K constraint of (13), i.e. $\sum_{k=1}^{K} z_{mk} = 1$, can be transformed as follows:

$$\sum_{k=1}^{K} t_{Km+k} = -K+2 \quad (m = 0, \ldots, M-1) \tag{28}$$

The central idea of the GW algorithm is to relax $\{1, -1\}$-valued variables into S^n-valued ones. In order to apply the GW algorithm, we first define the following auxiliary variables:

$$\mathbf{x}_0 = [1, 0, \ldots, 0]^\top, \tag{29}$$

$$\mathbf{x}_i = [t_i, 0, \ldots, 0]^\top \quad (i = 1, \ldots, KM) \tag{30}$$

On the basis of this transformation, we obtain the following relaxation problem:

$$\begin{aligned} \text{Maximize } & \operatorname{tr}\left([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{KM}] A \begin{bmatrix} \mathbf{x}_0^\top \\ \vdots \\ \mathbf{x}_{KM}^\top \end{bmatrix} \right) \\ \text{s.t. } & \mathbf{x}_i \in \mathbb{R}^{KM+1}, \quad \|\mathbf{x}_i\|_2 = 1 \quad (i = 0, \ldots, KM), \\ & \sum_{k=1}^{K} \mathbf{x}_{Km+k} = (-K+2)\mathbf{x}_0 \quad (m = 0, \ldots, M-1). \end{aligned} \tag{31}$$

It is easy to confirm that (31) is a relaxation problem of (13).

Next, in order to derive an SDP form, we transform the objective as follows:

$$g(Y) := \operatorname{tr}\left([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{KM}] A \begin{bmatrix} \mathbf{x}_0^\top \\ \vdots \\ \mathbf{x}_{KM}^\top \end{bmatrix} \right) = A \bullet Y, \tag{32}$$

by introducing a new variable $Y \in \mathrm{Sym}_{KM+1}$ as:

$$Y = \begin{bmatrix} y_{00} & y_{01} & \cdots & y_{0,KM} \\ y_{10} & y_{11} & \cdots & y_{1,KM} \\ \vdots & \vdots & \ddots & \vdots \\ y_{KM,0} & y_{KM,1} & \cdots & y_{KM,KM} \end{bmatrix} \tag{33}$$

$$= \begin{bmatrix} \mathbf{x}_0^\top \\ \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_{KM}^\top \end{bmatrix} [\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{KM}]. \tag{34}$$

From the definition, Y is positive semidefinite and satisfies

$$y_{ij} = \mathbf{x}_i^\top \mathbf{x}_j \quad (i = 0, 1, \ldots, KM, \quad j = 0, 1, \ldots, KM). \tag{35}$$

Conversely, there exists $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{KM} \in \mathbb{R}^{KM+1}$ satisfying Eq. (33) and Eq. (35) if Y is positive semidefinite.

By using the matrix Y, we can express the constraint conditions $\|\mathbf{x}_i\|_2 = 1$ by $y_{ii} = 1$. Since $\mathbf{x}_0$ is a unit vector, the condition $\sum_{k=1}^{K} \mathbf{x}_{Km+k} = (-K+2)\mathbf{x}_0$ holds if and only if

$$\mathbf{x}_0^\top \sum_{k=1}^{K} \mathbf{x}_{Km+k} = -K+2, \quad \left\| \sum_{k=1}^{K} \mathbf{x}_{Km+k} \right\|_2^2 = (-K+2)^2, \tag{36}$$

which are expressed as follows:

$$\sum_{k=1}^{K} y_{0,Km+k} = -K+2, \quad \sum_{k=1}^{K} \sum_{l=1}^{K} y_{Km+k,Km+l} = (-K+2)^2. \tag{37}$$

Summarizing the above arguments, we obtain the following SDP problem:

$$\begin{aligned} \text{Maximize } & g(Y) \\ \text{s.t. } & Y = (y_{ij})_{0 \le i,j \le KM} \in \mathrm{Sym}_{KM+1}, \quad Y \succeq O, \\ & Y_{ii} = 1 \quad (i = 0, \ldots, KM), \\ & \sum_{k=1}^{K} y_{0,Km+k} = -K+2 \quad (m = 0, \ldots, M-1), \\ & \sum_{k=1}^{K} \sum_{l=1}^{K} y_{Km+k,Km+l} = (-K+2)^2 \quad (m = 0, \ldots, M-1). \end{aligned} \tag{38}$$

This problem is equivalent to Problem (31), and hence is a relaxation of (13). Consequently, the optimal value of (38) gives an upper bound of that of (13). Due to space limitation, we omit to derive the SDP problem for (20). We denote the optimal solution of our SDP relaxation problem by $\tilde{Y}$.

4.3 Rounding

Once we obtain the optimal solution $\tilde{Y}$, we construct a feasible solution of the original BQP problem by

using rounding techniques. In the derivation of Problem (31), 1 is replaced by $\mathbf{x}_0$ and t_i is replaced by $\mathbf{x}_i$ for $i = 1, \ldots, KM$. Accordingly, as expectation, the following relationship between $\mathbf{z}$ and Y might hold:

$$2z_i - 1 = t_i = 1 \cdot t_i \approx \mathbf{x}_0^\top \mathbf{x}_i = y_{0i} \quad (i = 1, \ldots, KM). \tag{39}$$

A simple rounding is then to construct z_i as follows:

$$\tilde{z}_i = \begin{cases} 1 & \text{if } \tilde{y}_{0i} > \tilde{y}_{0j} \quad i \neq j \\ 0 & \text{otherwise} \end{cases}. \tag{40}$$

where we denote the rounded solution by $\tilde{\mathbf{z}}$. On the basis of the above observation, this paper applies two heuristics to explore a better feasible solution.

The first one is a deterministic search which is summarized in Algorithm 1. A key idea behind the deterministic search is to explore feasible solutions by combining elements with higher values of the relaxed solutions, $\tilde{y}_{0,Km+k}$ and the algorithm first collects indices as shown in Line 3. Then, it simply evaluates objective values of all possible combinations of collected indices as shown in Line 5. Note that it restricts the size of the search space, by T, to avoid combinatorial explosion of the search space.

The second one is a randomized search which is summarized in Algorithm 2. A key idea behind the randomized search is to interpret $y_{0,Km+k}$ as probability by the constraint $\sum_{k=1}^{K} y_{0,Km+k} = -K + 2$. Then, we pick an index i by proportional to the probability $(\tilde{y}_{0,i} + 1)/2$ and set $\mathbf{z}$ as follows:

$$z_j = \begin{cases} 1 & j = i \\ 0 & j \in I_s \setminus \{i\} \end{cases} \tag{41}$$

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. On the basis of our empirical evaluation, for (13), the deterministic search performed slightly better. On the other hand, it often missed to find a feasible solution for (20) and therefore the randomized search is preferable.

4.4 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 $g(\tilde{Y})$. Then let us consider the following inequality:

$$f(\tilde{\mathbf{z}}) \leq f(\mathbf{z}^*) \leq g(\tilde{Y}), \tag{47}$$

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 holds because the relaxed problem always gives an upper bound of the original problem. Eq. (47) gives us a lower bound of the approximation ratio of the obtained solution as follows:

$$\delta(\tilde{\mathbf{z}}, \tilde{Y}) := \frac{f(\tilde{\mathbf{z}})}{g(\tilde{Y})} \leq \frac{f(\tilde{\mathbf{z}})}{f(\mathbf{z}^*)} \leq 1. \tag{48}$$

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

5 Simulation Study

This section investigates detailed behaviors of the proposed method on the basis of artificial simulation. We used GUROBI Optimizer 6.0.4[2], which is a state-of-the-art commercial solver for mathematical programming, to solve MIQP and MILP problems. Also, we used SDPA 7.3.8[3], which is an open source solver for SDP problems. All experiments were conducted in a machine equipped with Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz (72 cores), 768GB RAM, and CentOS7.1. We limited all processes to single CPU core.

[2] `http://www.gurobi.com/`
[3] `http://sdpa.sourceforge.net/`

5.1 Simulation Model

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

$$q_m = \alpha^*_m + \sum_{m'=1}^{M}\sum_{d=1}^{D}\beta^*_{mm'd} f_d(p_{m'}) + \epsilon, \quad \epsilon \sim N(0,\sigma^2), \tag{49}$$

where $\{f_d(x)\} = \{x, x^2, 1/x\}$ and the true coefficients $\{\alpha^*_m\}$ and $\{\beta^*_{mm'd}\}$ were generated by Gaussian random numbers, so that $\alpha^*_m \sim N(4M, 1), \beta^*_{mm'd} \sim N(0,1)(m \neq m'), \beta^*_{mm'd} \sim N(-1,1)(m = m')$. The price p_m is uniformly sampled from the fixed price candidates $\{0.8, 0.85, 0.9, 0.95, 1\}$ ($K = 5$) and cost was fixed to $c_m = 0.7$.

Let us denote the gross profit function (2) computed with the true parameter by $\ell^*(\mathbf{p})$. Then, we denote its expectation by

$$f^*(\mathbf{z}) := E_\epsilon[\ell^*(\mathbf{p})], \tag{50}$$

where E_ϵ is expectation with respect to ϵ. Its maximizer is then denoted by

$$\mathbf{z}^* = \arg\max_{\mathbf{z}\in\mathcal{Z}} f^*(\mathbf{z}). \tag{51}$$

5.2 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 described in Section 3.2. 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 $f^*(\tilde{\mathbf{z}})$ which satisfy:

$$f^*(\tilde{\mathbf{z}}) \leq f^*(\mathbf{z}^*) \leq \bar{f}^*. \tag{52}$$

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 $\tilde{\mathbf{z}} = \mathbf{z}^*$.

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 $M = 11$, and we confirmed that they cannot scale to large problems.
- 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.

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 not real-time processing but is sufficiently for scenarios such as price planning for retail stores. Further, let us emphasize that we used only single core for comparison, and hence the computational time can be signif-

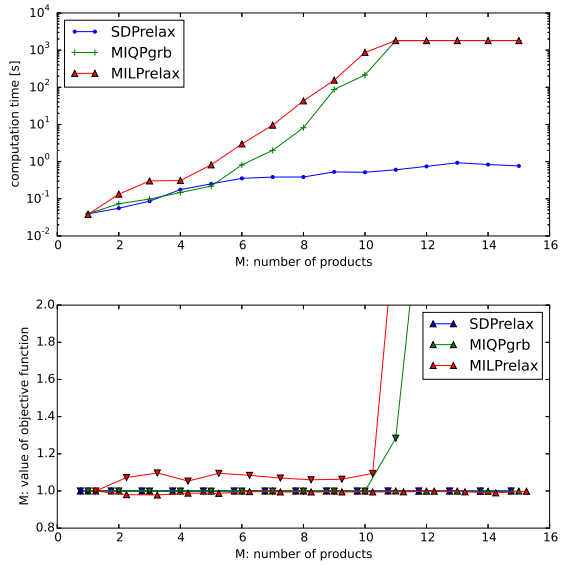

Figure 1: Comparisons of SDPrelax, MIQPgrb and MILPrelax with a small number of products. The horizontal axis represents the number of products M. The vertical axes represent computational time (top) and $f^*(\tilde{\mathbf{z}})$ and $\bar{f}^*$ objective values (bottom). For the bottom, values are normalized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

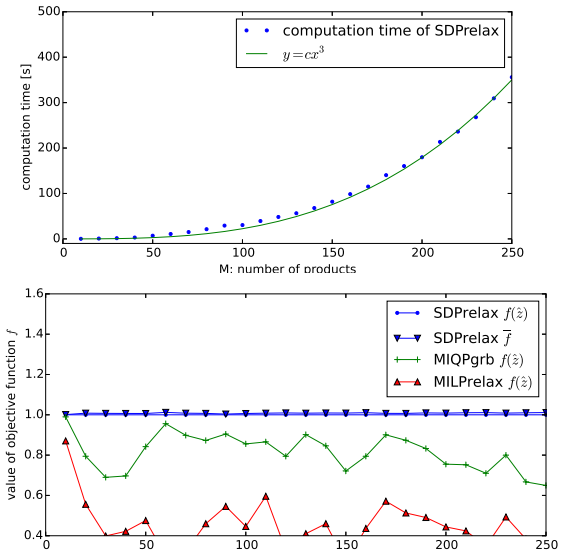

Figure 2: Comparisons of SDPrelax, MIQPgrb and MILPrelax with a large number of products. The top figure shows the computational time of SDPrelax over the number of products M. The bottom figure compares values of $f^*(\tilde{\mathbf{z}})$ for the three methods by restricting their computational time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

icantly reduced by taking an advantage of recent advanced parallel linear algebra processing.

- 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.
- In the bottom figure, 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.

These results show that, for solving our BQPs (13), SDPrelax significantly outperforms the other state-of-the-art 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.

5.3 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, which are denoted by $\{\hat{\alpha}_m\}$ and $\{\hat{\beta}_{mm'd}\}$. The gross profit function with the estimated parameters is then denoted by $\hat{f}(\mathbf{z})$.

Let us define the solution on the estimated objec-

tive as follows:

$$\hat{\mathbf{z}} = \arg\max_{\mathbf{z}\in\mathcal{Z}} \hat{f}(\mathbf{z}). \tag{53}$$

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 $\hat{\mathbf{z}}$.

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:

$$f^*(\hat{\mathbf{z}}) \leq f^*(\mathbf{z}^*) \leq E_{\mathcal{D}}[\hat{f}(\hat{\mathbf{z}})] \tag{54}$$

where $E_{\mathcal{D}}$ is expectation w.r.t. $\mathcal{D}$. We omit the proof for space limitation. This result yields two 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?

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 (49). Roughly speaking, δ is the level of prediction or estimation error that we cannot avoid.

Fig. 3 illustrates behaviors of $f^*(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ 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 $f^*(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ got linearly large along with increasing M and it became over 15% (i.e. $f^*(\hat{\mathbf{z}})/f^*(\mathbf{z}^*) \geq 1.15$) with fifty products ($M = 50$) under $\delta = 0.2$. On the other hand, the actual gross profit (red line) remained nearly-optimal $\hat{f}(\hat{\mathbf{z}})/f^*(\mathbf{z}^*) \sim 1.0$. 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 overestimates the gross profit. Although the obtained price strategy stays in a nearly-optimal solution, 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.

6 Summary

This paper presented prescriptive price optimization, which models complex demand-price relationships based on massive regression formulas produced by machine learning and then finds the optimal prices maximizing the profit function. We showed that the problem can be formulated as BQP problems, and a fast solver using a SDP relaxation was presented. It was confirmed in simulation experiments that the proposed algorithm performs much better than state-of-the-art optimization methods in terms of both scalability and quality of output solutions. Empirical evaluations were conducted with 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 8.2% 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 a issue using sparse learning. A challenging future work is to avoid effects of estimation error in real application. When there are unobserved

variables affecting price or sales, then we cannot estimate the parameters accurately, which might cause a big errors in estimated gross profit function and the result might be unreliable. In order to cope with such a situation, studies on optimization framework which can take account of the error of estimation, such as robust optimization framework, may be needed.

References

[1] D. Bienstock and N. ÖZbay. Computing robust basestock levels. *Discrete Optimization*, 5(2):389–414, 2008.

[2] J. Burgschweiger, B. Gnädig, and M. C. Steinbach. Optimization models for operative planning in drinking water networks. *Optimization and Engineering*, 10:43–73, 2009.

[3] F. Caro and J. Gallien. Clearance pricing optimization for a fast-fashion retailer. *Operations Research*, 60(6):1404–1422, 2012.

[4] C. Dziekan. The analytics journey. *Analytics*, 2010.

[5] 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.

[6] 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.

[7] 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.

[8] R. Klein. *Revenue Management*. Springer, 2008.

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

[10] 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.

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

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

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

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

[15] 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.

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

[17] 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.

[18] J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653, 1999.

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

[20] 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.

[21] 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.

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

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

Algorithm 1 Deterministic Search Rounding

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

1: For each m, initialize index sets such that

$$C_m = \{\arg\max_k \{\tilde{y}_{0,Km+k} \mid k = 1, \ldots, K\}\}. \tag{42}$$

2: **while** $\prod_{m=1}^{M} |C_m| < T$ **do**
3: Update an index set such that

$$(\tilde{m}, \tilde{k}) = \underset{m \in \{1,\ldots,M\}, k \in \{1,\ldots,K\}, k \notin C_m}{\arg\max} \{\tilde{y}_{0,Km+k}\}, \tag{43}$$

$$C_{\tilde{m}} \leftarrow C_{\tilde{m}} \cup \{\tilde{k}\}. \tag{44}$$

4: **end while**
5: Let $\mathcal{C}_z$ be a set of all combinatorial candidates of rounded solutions w.r.t. C_m for $\forall m$, formally defined as:

$$\mathcal{C}_z := \{[0, \ldots, \underbrace{0, \ldots, \overset{k}{\underset{\vee}{1}}, \ldots 0}_{m\text{-th chunk}}, \ldots, 0]^\top | \forall k \in C_m, \forall m\},$$

where $|\mathcal{C}_z| = \prod_{m=1}^{M} |C_m| \geq T$. Then, compute the rounded solution as follows:

$$\tilde{\mathbf{z}} = \arg\max_{\mathbf{z} \in \mathcal{C}_z \cap \mathcal{Z}} f(\mathbf{z}) \tag{45}$$

where $\mathcal{Z}$ is the feasible region of the original problem.

Algorithm 2 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 (41).

2: **do**

3: Let $I_{\text{vio}} \subseteq \{1,\dots,n\}$ be defined by:

$$I_{\text{vio}} = \{i \mid \exists \text{ violated constraint w.r.t. } z_i\} \quad (46)$$

4: Pick s such that $I_{\text{vio}} \cap I_s \neq \emptyset$ randomly, and pick i from I_s with probability $(\tilde{y}_{0,i}+1)/2$. Then, set $\tilde{\mathbf{z}}$ by (41) for a given I_s.

5: **while** $|I_{\text{vio}}| > 0$

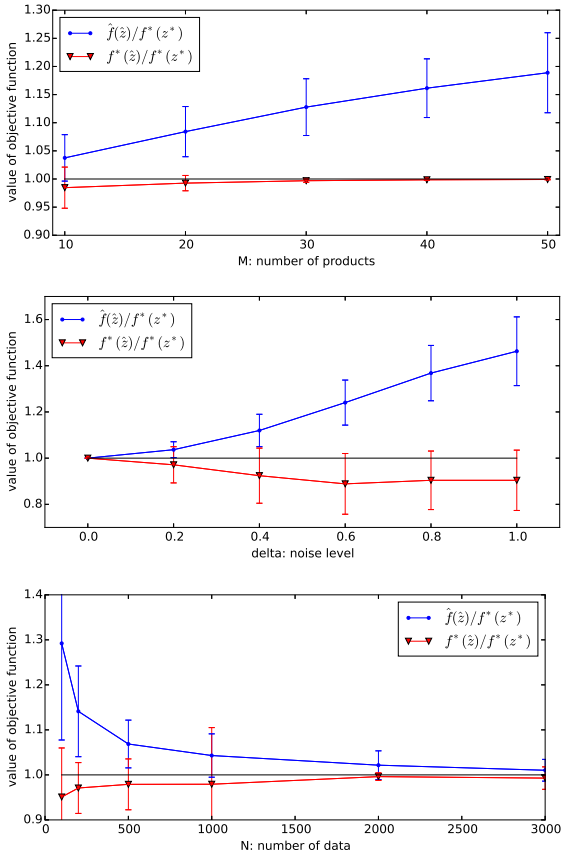

Figure 3: Value of $f^*(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ (red) and $\hat{f}(\hat{\mathbf{z}})/f^*(\mathbf{z}^*)$ (blue) for different setting. Dot and error bar mean the average and standard deviation of 100 times trial. Top: $\delta = 0.2$, $N = 1000$. Middle: $M = 10$, $N = 1000$. Bottom: $M =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