A Lagrangian Approach for Multiple Personalized Campaigns

Yong-Hyuk Kim, Yourim Yoon, and Byung-Ro Moon, *Member*, *IEEE*

Abstract—The multicampaign assignment problem is a campaign model to overcome the multiple-recommendation problem that occurs when conducting several personalized campaigns simultaneously. In this paper, we propose a Lagrangian method for the problem. The original problem space is transformed to another simpler one by introducing Lagrange multipliers, which relax the constraints of the multicampaign assignment problem. When the Lagrangian vector is supplied, we can compute the optimal solution under this new environment in $O(NK^2)$ time, where N and K are the numbers of customers and campaigns, respectively. This is a linear-time method when the number of campaigns is constant. However, it is not easy to find a Lagrangian vector in exact accord with the given problem constraints. We thus combine the Lagrangian method with a genetic algorithm to find good near-feasible solutions. We verify the effectiveness of our evolutionary Lagrangian approach in both theoretical and experimental viewpoints. The suggested Lagrangian approach is practically attractive for large-scale real-world problems.

Index Terms—Personalized campaign, multicampaign assignment, constrained optimization, Lagrangian method, genetic algorithm.

1 Introduction

Customer relationship management (CRM) [12] is important in acquiring and maintaining loyal customers. To maximize the revenue and customer satisfaction, companies try to provide personalized services for customers. As personalized campaigns are frequently performed, several campaigns often happen to run simultaneously or within a short period of time. It is often the case that an attractive customer for a specific campaign tends to be attractive for other campaigns. If we conduct separate campaigns without considering other campaigns, some customers may be bombarded by a considerable number of campaigns. We call this the *multiple recommendation problem*. The larger the number of recommendations for a customer, the lower the customer interest for campaigns [5]. In the long run, the rate of customer response for campaigns drops. It clearly lowers marketing efficiency, as well as customer satisfaction, hence diminishing customer loyalty. Unfortunately, traditional methods only focused on the effectiveness of individual campaigns and did not consider the problem with respect to multiple recommendations. In the situation that several campaigns are conducted within a short time window, it is necessary to find the optimal campaign assignment to customers considering the recommendations in other campaigns.

The *multicampaign assignment problem* (MCAP) is a complex assignment problem in which each of N customers is assigned to one subset drawn from a set of K campaigns. The goal is to find a set of assignments such that the effect of campaigns is maximized under some constraints. The main difference with independent campaigns lies in that the customer response for campaigns is influenced by multiple recommendations. We defined MCAP in previous work and proposed two types of methods for the problem [21]. We showed that one can solve MCAP to optimality by a dynamic programming (DP) algorithm. Although the DP algorithm guarantees optimal solutions, it becomes intractable for large-scale problems. We thus proposed heuristic methods that not only have practical time complexity but also showed good performance. However, since they are heuristics, they do not guarantee optimal solutions. In this paper, we propose a Lagrangian method with the advantages of both the DP method and the heuristic method. It has linear time complexity for a fixed number of campaigns and sacrifices little on the optimality. Furthermore, the Lagrangian method also provides a good upper bound and can be used to measure the suboptimality of other heuristics. However, in general, randomly generated Lagrange multipliers do not satisfy the given problem constraints. It is not easy to find Lagrange multipliers in exact accord with the constraints. Since it is important to find good Lagrange multipliers, we use a genetic algorithm (GA) to optimize Lagrange multipliers. We verify the effectiveness of the proposed genetic Lagrangian method with field data.

The remainder of this paper is organized as follows: In Section 2, we describe MCAP and previous algorithms. We propose a Lagrangian method for the problem in Section 3. We show experimental results in Section 4 and, finally, make conclusions in Section 5.

2 Preliminaries

The assignment problem is formally defined as follows: Given two sets, A and T, together with a weighted function

- *Y.-H. Kim is with the Department of Computer Science, Kwangwoon University, Wolgye-dong, Nowon-gu, Seoul, 139-701, Korea. E-mail: yhdfly@kw.ac.kr.*
- *Y. Yoon and B.-R. Moon are with the School of Computer Science and Engineering, Seoul National University, Sillim-dong, Gwanak-gu, Seoul, 151-744, Korea. E-mail: {yryoon, moon}@soar.snu.ac.kr.*

Manuscript received 3 Oct. 2006; revised 21 Apr. 2007; accepted 4 Oct. 2007; published online 11 Oct. 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0465-1006.
Digital Object Identifier no. 10.1109/TKDE.2007.190701.

$C : A \times T \to \mathbb{R}$, find the function $f : A \to T$ such that the cost $\sum_{a \in A} C(a, f(a))$ is minimized or maximized.[1] In typical assignment problems such as the optimal assignment problem [23] and the quadratic assignment problem [9], the size of domain set (A) is equal to that of codomain set (T), thus restricting the solutions to one-to-one mappings. However, in some other assignment problems such as the generalized assignment problem [11] and the sailor assignment problem [28], the sizes are different. In MCAP, the sizes are not equal; there are far more customers than campaigns, that is, $N \gg K$.

2.1 Multicampaign Assignment Problem

2.1.1 Problem Definition

Let N be the number of customers and K be the number of campaigns. In the following, we describe the input, output, constraints, and evaluation function for MCAP.

Input. Each campaign has a weight. For each customer, the preference for each campaign is given. A response suppression function R related to multiple recommendations is given. These inputs are denoted as follows:

- $\boldsymbol{w} = (w_1, w_2, \ldots, w_K) \in \mathbb{R}^K$: the campaign weight vector ($w_j > 0$ for each campaign j).
- $P = (p_{ij})$: the preference matrix. Each real-valued element $p_{ij} \in [0, \infty)$ is the preference value of customer i for campaign j.
- $R : \mathbb{N} \longrightarrow [0, 1]$: the response suppression function with respect to the number of recommendations (for convenience, we assume that $R(0) = 0$). In this paper, we assume that response suppression is the same for all customers.

The preferences for a campaign can be acquired from some existing methods such as collaborative filtering (CF) [34]. If h_i is the number of multiple recommendations for customer i, the actual preference of customer i for campaign j becomes $R(h_i)p_{ij}$.

Constraints. The maximum and the minimum numbers of recommendations for each campaign are enforced. Let $\bar{b}_j$ be the maximum number of recommendations for campaign j and $\underline{b}_j$ be the minimum number of recommendations for campaign j. Then, the number of recommendations in campaign j is between $\underline{b}_j$ and $\bar{b}_j$. These constraints are denoted as follows:

- $\boldsymbol{b}^* = (\bar{b}_1, \bar{b}_2, \ldots, \bar{b}_K) \in \mathbb{N}^K$: the upper bound vector of capacity constraint and
- $\boldsymbol{b}_* = (\underline{b}_1, \underline{b}_2, \ldots, \underline{b}_K) \in \mathbb{N}^K$: the lower bound vector of capacity constraint.

Output. The output is an $N \times K$ binary campaign assignment matrix $M = (m_{ij})$, in which m_{ij} indicates whether or not campaign j is assigned to customer i.

Evaluation. The *campaign preference sum* for campaign j is defined to be the actual preference sum of recommended customers for campaign j as follows: $\sum_{i=1}^{N} R(h_i)p_{ij}m_{ij}$. The fitness $F(M)$ of a campaign assignment matrix M is the weighted sum of campaign preference sums:

$$F(M) = \sum_{j=1}^{K} \left(w_j \sum_{i=1}^{N} R(h_i) p_{ij} m_{ij} \right).$$

The objective is to find a matrix M that maximizes F.

2.1.2 Notations and Formal Definition

Some notations that would be used in the remainder of this paper are listed as follows:

- $\boldsymbol{m_i} = (m_{i1}, m_{i2}, \ldots, m_{iK})$: the ith row vector of the matrix M,
- $h_i (= \sum_{j=1}^{K} m_{ij})$: the number of multiple recommendations for customer i,
- $\sigma_i (= \sum_{j=1}^{K} w_j p_{ij} m_{ij})$: the weighted sum of preferences of customer i for recommended campaigns,
- $\mathbf{1}_n$: the n-dimensional vector $(1, 1, \ldots, 1)$, and
- $\mathbf{0}_n$: the n-dimensional vector $(0, 0, \ldots, 0)$.

More formally, MCAP is defined as follows:

Definition 1. MCAP *is the problem of finding a campaign assignment matrix* $M = (m_{ij})$ *that maximizes*

$$\langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K M^T) M' \rangle \quad \textit{subject to} \quad \boldsymbol{b}_* \le \mathbf{1}_N M \le \boldsymbol{b}^*,$$

where $M' = (m'_{ij})$ *is the* $N \times K$ *real matrix, where* $m'_{ij} = p_{ij} m_{ij}$,

$$\boldsymbol{R}(x_1, x_2, \ldots, x_n) = (R(x_1), R(x_2), \ldots, R(x_n)),$$

$\langle \cdot, \cdot \rangle$ *is an inner product, and each* n-dimensional *vector is regarded as a row vector, that is,* $1 \times n$ *matrix.*

MCAP is a constrained maximization problem with a *nonlinear* objective function. In previous work, we could prove that a practically generalized version[2] of MCAP is NP-complete [41]. We strongly conjecture that the original version of MCAP is also computationally intractable. However, it is still an open problem.

2.1.3 Response Suppression Function

In the case of multiple campaign recommendations, the customer response rate drops as the number of recommendations grows. We introduced the response suppression function for the response-rate degradation with multiple recommendations. Definitely, the function should be monotonically nonincreasing. Fig. 1 shows the response suppression function R used in [21], which was derived from the Gaussian function. The function is nonnegative monotonically decreasing with a maximum value of one. By the function, the preference for a campaign drops to, for example, 1/3 when four campaigns are performed simultaneously to a customer.

The function relies only on the number of recommendations, and the same function is applied to all customers. However, in practical situations, some customers may have more tolerance than others. That is, each customer class may have its own response suppression function since the degree of response suppression can be different from customer class to customer class. In this case, it is also crucial to find response suppression functions of customer

1. In this paper, we deal with only the simplified assignment problems where capacities are always 1, but *flow* problems with upper and lower bounds of capacities have been studied in [3].

2. Assume that each customer has its own response suppression function, i.e., the response suppression rate of a customer varies as his/her inclination.

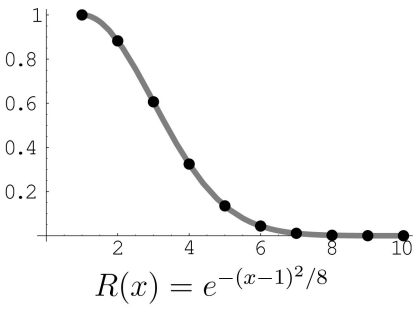


Fig. 1. Response suppression function ($R(x) = 0$ for $x > 10$).

classes. An extended study dealing with the generalized response suppression is given in [41]. Our method can be immediately applied to even this generalized version. In this paper, just for notational convenience, we assume that response suppression is global explicitly; there is one for all customers.

2.2 Prior Methods

In this section, we summarize the three methods for MCAP suggested in [21].

2.2.1 Dynamic Programming

The DP algorithm in [21] requires $O(N^{K+1}K)$ space and takes $O(N^{K+1}K2^K)$ time. The DP approach tries to solve all subproblems of the form "what is the best assignment for $(b_1, b_2, \ldots, b_K)$, using just the first n customers," for all $0 \leq b_1 \leq N, \ldots, 0 \leq b_K \leq N$, and $0 \leq n \leq N$. If K is a fixed number, this is a polynomial-time algorithm. However, when K is not small, it is nearly intractable. The proposed DP algorithm is only applicable to problems with small K, N pairs. However, the DP algorithm guarantees optimal solutions.

2.2.2 Constructive Algorithm

Starting at the initial situation that no campaign is recommended to any customers, the constructive assignment algorithm (CAA) iteratively assigns campaigns to customers in a greedy manner. It first prepares the most attractive α customers for each campaign (we set α to be $N/2$ in our experiments). Next, it iteratively performs the following: It chooses a pair (customer i, campaign j) with the maximum *gain*, and if the gain is positive and campaign j does not exceed the maximum number of recommendations, it recommends campaign j to customer i. If a recommendation is done, it updates the gains of customer i for the other campaigns. In that case, any gain cannot be positive after the maximum gain drops below zero. When the maximum gain is not positive, the algorithm terminates as long as every campaign satisfies the capacity constraint on the minimum number of recommendations. The time complexity of the algorithm is $O(NK^2 \log N)$ with the help of a balanced binary search tree.

2.2.3 Iterative Improvement

After every customer is assigned to a proper number of campaigns, we can run the iterative improvement heuristic. It proceeds in a series of *passes*. During each pass, the heuristic improves on the initial solution to create a new solution. A pass of iterative improvement is conducted for each campaign. Given a campaign, the heuristic chooses an equal-sized subset pair of recommended customers and nonrecommended customers that has the maximum *gain sum* when swapped. After swapping the subset pair, another pass is then executed starting with the new solution. This process is repeated until no improvement can be obtained. The time complexity of one pass is $O(NK \log N)$.

3 Lagrangian Optimization

One could solve MCAP to optimality by the DP algorithm. Although the DP algorithm guarantees optimality, it becomes intractable for large-scale instances. The heuristic algorithms not only have practical time complexity but also show reasonable performance. However, since they are just heuristics, they do not guarantee optimality. In this section, we propose a novel method with the advantages of both the DP method and the heuristic one.

MCAP is a constrained optimization problem (with a nonlinear objective function). To relax the constraints of MCAP, we can transform it using the Lagrange multipliers [25]. We try out a method that transforms the search space of the problem to another space and searches the solution in the transformed space instead of managing the original space directly. However, we have a lot of limitations since the domain is not continuous but discrete.

3.1 Lagrangian Preliminaries

Consider the following maximization problem with constraints:

$$\max_{\boldsymbol{x} \in \Omega} f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0}.$$

Assume that we can reach the maximum μ_0. If the domain Ω is convex, the real vector $\boldsymbol{\lambda} \geq \boldsymbol{0}$ such that

$$\mu_0 = \max_{\boldsymbol{x} \in \Omega} \{f(\boldsymbol{x}) - \langle \boldsymbol{\lambda}, \boldsymbol{g}(\boldsymbol{x}) \rangle\}$$

always exists [25]. In this case, if the objective function $f(\boldsymbol{x})$ achieves the maximum when $\boldsymbol{x}$ is $\boldsymbol{x_0}$, then $\langle \boldsymbol{\lambda}, \boldsymbol{g}(\boldsymbol{x_0}) \rangle = 0$, where $\boldsymbol{x_0}$ is the point at which μ_0 is attained.

In particular, if f and $\boldsymbol{g}$ are differentiable, $\nabla f(\boldsymbol{x_0}) = \sum_i \lambda_i \nabla g_i(\boldsymbol{x_0})$ holds. If we are lucky, we can easily find the value of $\boldsymbol{\lambda}$ for the problem and solve the problem by transforming the original problem with constraints into the easier one without constraints. The method that solves the problem by finding $\boldsymbol{\lambda}$ and transforming the given constrained problem into the unconstrained one is called the *Lagrangian method*.

There have been a number of papers that studied the Lagrangian method for discrete problems [10], [15], [18], [26], [30], [35]. Most of them focused on the problems with a linear objective function called *integer programming problems*. Moreover, typically, they tried to find just the upper bound to be used in a *branch-and-bound* algorithm by dualizing constraints (see Appendix A) and hence could not find the feasible solution directly. To the authors' best knowledge, there has been only one Lagrangian method to find lower bounds (feasible solutions), which is proposed by Magazine

and Oguz [26]. Its variant has been studied in recent work [30], [32], [40]. However, the problem they dealt with, the *0/1 multiconstrained knapsack problem*, has a typical linear objective function, and their method used its linearity most importantly. Therefore, it is impossible to directly apply their methods to MCAP with a nonlinear objective function. Recently, there was a study about the Lagrange multiplier theory for nonlinear discrete optimization [39]. However, it did not provide any explicit algorithm but some theory about *saddle points*. In summary, a number of studies about Lagrangian methods for discrete optimization have been done, but they have focused on linear problems just to find upper bounds, traditionally using the *subgradient method* (see Appendix B). Therefore, the previous studies cannot be directly applied to MCAP. However, we will compare our Lagrangian approach with a variant of the traditional subgradient method later.

3.2 Our Lagrangian Method

MCAP is a maximization problem with capacity constraints. Consider the following simplified MCAP without the lower bound constraints:

$$\max_{M\in\{0,1\}^{N\times K}} \langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K M^T)M'\rangle \quad \text{subject to} \quad \mathbf{1}_N M \le \boldsymbol{b}^*.$$

We cannot always apply the Lagrangian method to MCAP because the domain $\{0,1\}^{N\times K}$ is not convex. However, for some capacity constraints, their corresponding $\boldsymbol{\lambda}$s exist. Assume that the real vector $\boldsymbol{\lambda}$ corresponding to the capacity constraints is given. Then, it is possible to transform the original optimization problem into the following problem using Lagrange multipliers:

$$\max_{M\in\{0,1\}^{N\times K}} \{\langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K M^T)M'\rangle - \langle \boldsymbol{\lambda}, \mathbf{1}_N M - \boldsymbol{b}^*\rangle\}. \quad (1)$$

In the original MCAP, there are lower bound constraints, that is, $\boldsymbol{b}_* \le \mathbf{1}_N M$. We can also relax the constraints by using another Lagrange multiplier vector $\boldsymbol{\mu} \ge \mathbf{0}_K$. Then, the transformed (1) becomes

$$\begin{aligned}
&\max_M\{\langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K M^T)M'\rangle - \langle \boldsymbol{\lambda}, \mathbf{1}_N M - \boldsymbol{b}^*\rangle - \langle \boldsymbol{\mu}, \boldsymbol{b}_* - \mathbf{1}_N M\rangle\}\\
&\quad = \max_M\{\langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K M^T)M'\rangle - \langle \boldsymbol{\lambda}-\boldsymbol{\mu}, \mathbf{1}_N M\rangle\} + \langle \boldsymbol{\lambda}, \boldsymbol{b}^*\rangle\\
&\qquad - \langle \boldsymbol{\mu}, \boldsymbol{b}_*\rangle\\
&\quad = \max_M\{\langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K M^T)M'\rangle - \langle \boldsymbol{\lambda}-\boldsymbol{\mu}, \mathbf{1}_N M\rangle\} + \langle \boldsymbol{\lambda}-\boldsymbol{\mu}, \boldsymbol{b}^*\rangle\\
&\qquad + \langle \boldsymbol{\mu}, \boldsymbol{b}^* - \boldsymbol{b}_*\rangle\\
&\quad = \max_M\{\langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K M^T)M'\rangle - \langle \boldsymbol{\lambda}-\boldsymbol{\mu}, \mathbf{1}_N M - \boldsymbol{b}^*\rangle\}\\
&\qquad + \langle \boldsymbol{\mu}, \boldsymbol{b}^* - \boldsymbol{b}_*\rangle.
\end{aligned}$$

By substituting $\boldsymbol{\nu}$ for $\boldsymbol{\lambda}-\boldsymbol{\mu}$, it becomes

$$\max_{M\in\{0,1\}^{N\times K}} \{\langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K M^T)M'\rangle - \langle \boldsymbol{\nu}, \mathbf{1}_N M - \boldsymbol{b}^*\rangle\} + \langle \boldsymbol{\mu}, \boldsymbol{b}^* - \boldsymbol{b}_*\rangle,$$

where $\boldsymbol{\nu} \ge -\boldsymbol{\mu}$, and $\boldsymbol{\mu} \ge \mathbf{0}_K$. This transformed formula of the original MCAP is quite similar to (1), corresponding to the simplified MCAP only with the upper bound constraints. Actually, most theoretical analyses of the two problems, the original MCAP and the simplified MCAP, in this paper are almost the same.[3] Therefore, now, we will consider only the simplified MCAP to make our theoretical analysis more readable.

It is easy to find the maximum of the transformed problem (1) using the following formula:

$$\begin{aligned}
&\langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K M^T)M'\rangle - \langle \boldsymbol{\lambda}, \mathbf{1}_N M - \boldsymbol{b}^*\rangle\\
&= \sum_{j=1}^{K} w_j \sum_{i=1}^{N} R(h_i) m'_{ij} - \sum_{j=1}^{K} \lambda_j \sum_{i=1}^{N} m_{ij} + \langle \boldsymbol{\lambda}, \boldsymbol{b}^*\rangle\\
&= \sum_{i=1}^{N} \left(R(h_i) \sum_{j=1}^{K} w_j m'_{ij} - \sum_{j=1}^{K} \lambda_j m_{ij} \right) + \langle \boldsymbol{\lambda}, \boldsymbol{b}^*\rangle\\
&= \sum_{i=1}^{N} \sum_{j=1}^{K} \big(R(h_i) w_j p_{ij} - \lambda_j\big) m_{ij} + \langle \boldsymbol{\lambda}, \boldsymbol{b}^*\rangle\\
&= \sum_{i=1}^{N} (R(h_i)\sigma_i - \langle \boldsymbol{\lambda}, \boldsymbol{m_i}\rangle) + \langle \boldsymbol{\lambda}, \boldsymbol{b}^*\rangle.
\end{aligned}$$

To maximize the above formula for a fixed $\boldsymbol{\lambda}$, we have to set $\boldsymbol{m_i} \in \{0,1\}^K$ to maximize $R(h_i)\sigma_i - \langle \boldsymbol{\lambda}, \boldsymbol{m_i}\rangle$ for each i. Since each $R(h_i)$ and σ_i do not have an effect on the others, getting the maximum is fairly easy. For each i, the maximum can be obtained by choosing topmost h_i values of K values $(R(h_i)w_j p_{ij} - \lambda_j)$s. Fig. 2 shows the procedure. STEP 1-1 can be done in $O(K)$ time if we use the most efficient algorithm [6]. Also, STEP 1-2, STEP 1-3, and STEP 2 take $O(K)$ time. Therefore, the procedure takes $O(K^2)$ time. This process has to be performed for each $i = 1, 2, \ldots, N$. Hence, the total time complexity of maximizing (1) becomes $O(NK^2)$.[4] It is more tractable than the original problem. Roughly speaking, for a fixed number K,[5] the problem size is lowered from $O(N^{K+1})$ to $O(N)$.

If we only find out $\boldsymbol{\lambda}$ for the problem, we get the optimal solution of MCAP in polynomial time. We may have the problem that such $\boldsymbol{\lambda}$ never exists or it is difficult to find it although it exists. However, this method is not entirely useless. For an arbitrary $\boldsymbol{\lambda}$, let the maximum of the above formula be $\tilde{\mu}$ and the matrix $M = (m_{ij})$ that achieves the maximum be $\tilde{M}$. Since $\boldsymbol{\lambda}$ is chosen arbitrarily, we do not guarantee that $\tilde{M}$ satisfies the capacity constraints of the original problem. Nevertheless, letting the capacity constraint vector be $\tilde{\boldsymbol{b}} := \sum_{i=1}^{N} \tilde{\boldsymbol{m}}_{\boldsymbol{i}}$ makes $\tilde{\mu}$ be the optimal value by Theorem 1. We call this algorithm the *Lagrangian method for MCAP* (LMMCAP). Fig. 3 shows the pseudocode of LMMCAP.

Theorem 1 (Optimality). *Given $\boldsymbol{\lambda} \ge \mathbf{0}_K$ and $\tilde{\boldsymbol{b}}$, the matrix $\tilde{M} = (\tilde{m}_{ij})$ obtained by LMMCAP is a maximizer of the following problem:*[6]

$$\max_{M\in\{0,1\}^{N\times K}} \langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K M^T)M'\rangle \quad \textit{subject to} \quad \mathbf{1}_N M \le \tilde{\boldsymbol{b}}.$$

3. The only differences are those that appear in Theorem 1 and Corollary 1, which use the condition that a Lagrange multiplier is nonnegative. We will also mention these facts as footnotes in Theorem 1

4. If it is implemented with a brute-force enumeration instead of the proposed procedure given in Fig. 2, it would take $O(NK2^K)$ time.

5. In most cases, the number of campaigns (K) is much smaller than that of customers (N), i.e., $K \ll N$. Therefore, K has a relatively small effect on the complexity.

6. Consider the original MCAP with the lower bound constraints. Since the Lagrangian vector $\boldsymbol{\nu}$ is not nonnegative, this theorem is true only for the problem with $\mathbf{1}_N M = \tilde{\boldsymbol{b}}$ as the constraint part. This restricted version does not affect the other theoretical analysis at all.

STEP 1. **for** each $h_i = 0, 1, \ldots, K$
  if $h_i \neq 0$ **then**
STEP 1-1. Select the h_i-th largest value x from K values $(R(h_i)w_j p_{ij} - \lambda_j)$s;
STEP 1-2. Take largest h_i values larger than or equal to the value x;
STEP 1-3. Sum the above h_i values;
  else the summation value is 0;
STEP 2. **return** the maximum among the above $K+1$ summation values;

Fig. 2. An efficient procedure to maximize $R(h_i)\sigma_i - \langle \boldsymbol{\lambda}, \boldsymbol{m_i} \rangle$.

Proof. Suppose that $\tilde{M} = (\tilde{m}_{ij})$ is not an optimal matrix. Let $\bar{M} = (\bar{m}_{ij})$ be an optimal matrix with $\bar{\boldsymbol{b}}$ ($\bar{\boldsymbol{b}} \leq \tilde{\boldsymbol{b}}$) as the capacity constraint vector. Let $\bar{F}_i$ be $R(\bar{h}_i)\bar{\sigma}_i - \langle \boldsymbol{\lambda}, \bar{\boldsymbol{m}}_i \rangle$ for each i in campaign assignment matrix $\bar{M}$. For each i, by the optimality of F_i ($F_i = R(\tilde{h}_i)\tilde{\sigma}_i - \langle \boldsymbol{\lambda}, \tilde{\boldsymbol{m}}_i \rangle$), $F_i \geq \bar{F}_i$. Then,

$$
\begin{aligned}
&\langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K \tilde{M}^T)\tilde{M}' \rangle \\
&= \langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K \tilde{M}^T)\tilde{M}' \rangle - \langle \boldsymbol{\lambda}, \mathbf{1}_N \tilde{M} - \tilde{\boldsymbol{b}} \rangle \quad (\because \ \mathbf{1}_N \tilde{M} = \tilde{\boldsymbol{b}}) \\
&= \sum_{i=1}^{N} F_i + \langle \boldsymbol{\lambda}, \tilde{\boldsymbol{b}} \rangle \\
&\geq \sum_{i=1}^{N} \bar{F}_i + \langle \boldsymbol{\lambda}, \tilde{\boldsymbol{b}} \rangle \\
&\geq \sum_{i=1}^{N} \bar{F}_i + \langle \boldsymbol{\lambda}, \bar{\boldsymbol{b}} \rangle \quad (\because \ \boldsymbol{\lambda} \geq \mathbf{0}_K \ \& \ \bar{\boldsymbol{b}} \leq \tilde{\boldsymbol{b}}) \\
&= \langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K \bar{M}^T)\bar{M}' \rangle - \langle \boldsymbol{\lambda}, \mathbf{1}_N \bar{M} - \bar{\boldsymbol{b}} \rangle \\
&= \langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K \bar{M}^T)\bar{M}' \rangle \quad (\because \ \mathbf{1}_N \bar{M} = \bar{\boldsymbol{b}}).
\end{aligned}
$$

This contradicts that $\tilde{M} = (\tilde{m}_{ij})$ is not an optimal matrix. Therefore, $\tilde{M} = (\tilde{m}_{ij})$ is an optimal matrix. □

Given a Lagrange multiplier vector $\boldsymbol{\lambda}$, we can get the capacity constraint vector $\boldsymbol{b_\lambda}$ and the optimal result with $\boldsymbol{b_\lambda}$ by the Lagrangian method.

Instead of finding the optimal solution of the original MCAP directly, we consider the problem of finding the $\boldsymbol{\lambda}$ corresponding to the given capacity constraints. That is, we transform the problem of dealing with an NK-dimensional binary matrix $M = (m_{ij})$ into that of dealing with a K-dimensional real vector $\boldsymbol{\lambda}$, as shown in Fig. 4. If there are Lagrange multipliers corresponding to the given capacity constraints and we find them, we easily get the optimal solution of MCAP. If there are no such Lagrange multipliers, we try to get a solution close to the optimum by devoting to find Lagrange multipliers that satisfy the given capacity constraints and are nearest to the capacities.

```
LMMCAP(λ)
{
    for each i = 1, 2, ..., N
        F_i ← max_{m_i ∈ {0,1}^K} (R(h_i)σ_i − ⟨λ, m_i⟩);
        m̃_i ← argmax_{m_i ∈ {0,1}^K} (R(h_i)σ_i − ⟨λ, m_i⟩);
    b̃ ← Σ_{i=1}^{N} m̃_i;
    μ̃ ← Σ_{i=1}^{N} F_i + ⟨λ, b̃⟩; // note that μ̃ = F(M̃)
    return μ̃, M̃ = (m̃_ij), and b̃;
}
```

Fig. 3. Lagrangian method for MCAP.

3.3 Some Useful Properties

If we are to find the solution of MCAP using the proposed Lagrangian method, we have to choose the maximum of $\langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K M(\boldsymbol{\lambda})^T)M'(\boldsymbol{\lambda}) \rangle$ for a nonnegative real vector $\boldsymbol{\lambda}$ such that $\boldsymbol{b}_* \leq \boldsymbol{b}_\lambda \leq \boldsymbol{b}^*$. However, if we use randomly generated Lagrange multipliers, they may not satisfy the capacity constraints, or it is probably hard to find a capacity close to the original one. The following theorem makes it easy to adjust Lagrange multipliers:

Theorem 2 (Tendency). *Suppose that $\boldsymbol{\lambda}$ and $\tilde{\boldsymbol{\lambda}}$ correspond to $\{M, \boldsymbol{b}\}$ and $\{\tilde{M}, \tilde{\boldsymbol{b}}\}$ by LMMCAP, respectively. Let $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_K)$ and $\tilde{\boldsymbol{\lambda}} = (\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_K)$, where $\lambda_i = \tilde{\lambda}_i$ for all $i \neq k$ and $\lambda_k \neq \tilde{\lambda}_k$. Then, if $\lambda_k < \tilde{\lambda}_k$, $b_k \geq \tilde{b}_k$, and if $\lambda_k > \tilde{\lambda}_k$, $b_k \leq \tilde{b}_k$.*

Proof. By the optimality of $\boldsymbol{\lambda}$,

$$\sum_{i=1}^{N} R(h_i)\sigma_i - \langle \boldsymbol{\lambda}, \boldsymbol{m_i} \rangle \geq \sum_{i=1}^{N} R(\tilde{h}_i)\tilde{\sigma}_i - \langle \boldsymbol{\lambda}, \tilde{\boldsymbol{m}}_i \rangle.$$

By the optimality of $\tilde{\boldsymbol{\lambda}}$,

$$\sum_{i=1}^{N} R(\tilde{h}_i)\tilde{\sigma}_i - \langle \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{m}}_i \rangle \geq \sum_{i=1}^{N} R(h_i)\sigma_i - \langle \tilde{\boldsymbol{\lambda}}, \boldsymbol{m_i} \rangle.$$

By summing the above two inequalities, we have

$$\sum_{i=1}^{N} -\langle \boldsymbol{\lambda}, \boldsymbol{m}_i \rangle + \sum_{i=1}^{N} -\langle \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{m}}_i \rangle \geq \sum_{i=1}^{N} -\langle \boldsymbol{\lambda}, \tilde{\boldsymbol{m}}_i \rangle + \sum_{i=1}^{N} -\langle \tilde{\boldsymbol{\lambda}}, \boldsymbol{m}_i \rangle.$$

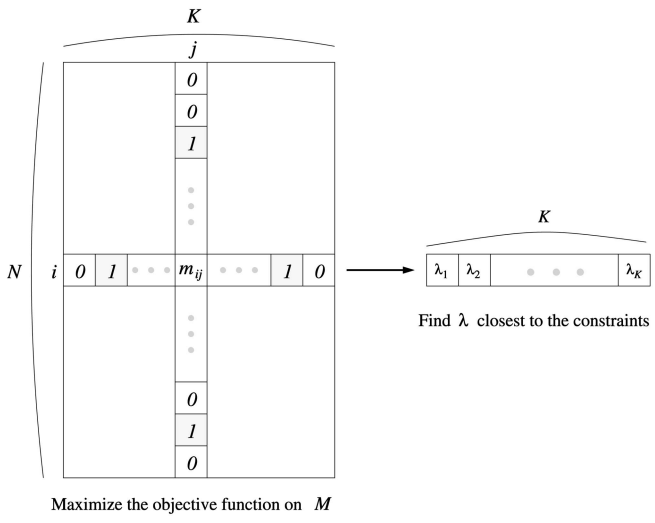


Fig. 4. An alternative way to solve MCAP.

Since $\lambda_k \neq \tilde{\lambda}_k$ and $\lambda_i = \tilde{\lambda}_i$ for all $i \neq k$, by cancellation and multiplying both sides by -1, we have

$$\sum_{i=1}^{N} \lambda_k m_{ik} + \sum_{i=1}^{N} \tilde{\lambda}_k \tilde{m}_{ik} \leq \sum_{i=1}^{N} \lambda_k \tilde{m}_{ik} + \sum_{i=1}^{N} \tilde{\lambda}_k m_{ik}.$$

Since $\sum_{i=1}^{N} m_{ik} = b_k$ and $\sum_{i=1}^{N} \tilde{m}_{ik} = \tilde{b}_k$, we have

$$\lambda_k b_k + \tilde{\lambda}_k \tilde{b}_k \leq \lambda_k \tilde{b}_k + \tilde{\lambda}_k b_k.$$

Now, we obtain the following inequality:

$$(\tilde{\lambda}_k - \lambda_k)(b_k - \tilde{b}_k) \geq 0.$$

□

From Theorem 2, b_i inversely grows with λ_i. Let $\mathbf{1}_N \tilde{M} = (\tilde{b}_1, \tilde{b}_2, \ldots, \tilde{b}_K)$ for $\tilde{M} = (\tilde{m}_{ij})$ that is obtained by LMMCAP with $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_K)$. By the above theorem, if $\tilde{b}_k > \bar{b}_k$, choosing $\boldsymbol{\lambda}' = (\lambda_1, \ldots, \lambda'_k, \ldots, \lambda_K)$ such that $\lambda'_k > \lambda_k$ and applying LMMCAP with $\boldsymbol{\lambda}'$ make the value of $\bar{b}_k$ smaller. It makes the kth capacity constraint satisfied or the degree of capacity violation smaller. Of course, another capacity constraint may be affected or violated by this operation. Also, which λ_k to change is an issue in the case that several capacity constraints are not satisfied. Hence, it is necessary to set efficient rules about which λ_k to change and how much to change it. If good rules are made, we can find better Lagrange multipliers than randomly generated ones quickly.

Theorem 3 (Sensitivity). *Suppose that $\boldsymbol{\lambda}_0$ and $\boldsymbol{\lambda}_1$ correspond to $\{M_0, \boldsymbol{b}_0\}$ and $\{M_1, \boldsymbol{b}_1\}$ by LMMCAP, respectively. Then, the following inequalities are satisfied:*

$$\langle \boldsymbol{\lambda}_0, \boldsymbol{b}_0 - \boldsymbol{b}_1 \rangle \leq F(M_0) - F(M_1) \leq \langle \boldsymbol{\lambda}_1, \boldsymbol{b}_0 - \boldsymbol{b}_1 \rangle.$$

In particular, for any assignment matrix M,

$$F(M) - F(M_0) \leq \langle \boldsymbol{\lambda}_0, \mathbf{1}_N M - \boldsymbol{b}_0 \rangle.$$

Proof. By the optimality of $\boldsymbol{\lambda}_0$,

$$F(M_0) - \langle \boldsymbol{\lambda}_0, \boldsymbol{b}_0 \rangle \geq F(M_1) - \langle \boldsymbol{\lambda}_0, \boldsymbol{b}_1 \rangle.$$

Hence, $F(M_0) - F(M_1) \geq \langle \boldsymbol{\lambda}_0, \boldsymbol{b}_0 - \boldsymbol{b}_1 \rangle$. By the optimality of $\boldsymbol{\lambda}_1$, $F(M_1) - \langle \boldsymbol{\lambda}_1, \boldsymbol{b}_1 \rangle \geq F(M_0) - \langle \boldsymbol{\lambda}_1, \boldsymbol{b}_0 \rangle$. Hence, $F(M_0) - F(M_1) \leq \langle \boldsymbol{\lambda}_1, \boldsymbol{b}_0 - \boldsymbol{b}_1 \rangle$. □

The following corollary shows that the proposed Lagrangian method also gives good upper bounds:

Corollary 1 (Upper Bound). *Suppose that $\boldsymbol{\lambda}_0 \geq \mathbf{0}_K$ corresponds to $\{M_0, \boldsymbol{b}_0\}$ by LMMCAP,[7] where $\boldsymbol{b}_0 \leq \boldsymbol{b}^*$. Then, $F(M_0) + \langle \boldsymbol{\lambda}_0, \boldsymbol{b}^* - \boldsymbol{b}_0 \rangle$ is an upper bound of MCAP.*

Proof. Since $\mathbf{1}_N M \leq \boldsymbol{b}^*$, $\mathbf{1}_N M - \boldsymbol{b}_0 \leq \boldsymbol{b}^* - \boldsymbol{b}_0$ and, hence, $\langle \boldsymbol{\lambda}_0, \mathbf{1}_N M - \boldsymbol{b}_0 \rangle \leq \langle \boldsymbol{\lambda}_0, \boldsymbol{b}^* - \boldsymbol{b}_0 \rangle$. Then, by Theorem 3,

$$F(M) \leq F(M_0) + \langle \boldsymbol{\lambda}_0, \mathbf{1}_N M - \boldsymbol{b}_0 \rangle \leq F(M_0) + \langle \boldsymbol{\lambda}_0, \boldsymbol{b}^* - \boldsymbol{b}_0 \rangle.$$

□

In Appendix A, we will also give the Lagrange duality and some limitations arising from the fact that the domain is not convex.

7. If we consider the original MCAP with the lower bound constraints, this corollary is true only when we have the condition that $\boldsymbol{\nu}_0 \geq \mathbf{0}_K$.

3.4 Investigation of the Lagrangian Space

The structure of the problem space is an important factor to indicate the problem difficulty, and the analysis of the structure helps the efficient search in the problem space. In this section, we conduct some experiments and get some insights into the global structure of the MCAP space.

In the previous sections, we showed that there is a correspondence between the capacity constraint vector and the Lagrange multiplier vector.[8] Instead of directly finding an optimal campaign assignment matrix, we deal with Lagrange multipliers. In this section, we empirically investigate the relationship between the constraint space and the Lagrangian space (that is, $\{\boldsymbol{b}_{\boldsymbol{\lambda}}\text{s}\}$ and $\{\boldsymbol{\lambda}\text{s}\}$).

We made experiments on three types of preference data $(P = (p_{ij}))$ with the same number of customers $(N = 48,559)$, changing the number of campaigns (K) from 1 to 10. One type is from the set of real-world data described in Section 4.1. The other types are artificially generated using Gaussian distribution $(N(m, \sigma^2))$ to have the same average and standard deviation of preferences as those in the real-world data. One type of artificial data is uncorrelated between campaigns; that is, each preference value is from an identical and independent distribution. The other type of artificial data is correlated between campaigns; we set $p_{i,j+1}$ to be $p_{ij} + N(0,1)$ for every customer i and campaign j. As we will show in Section 4.1, the latter correlated artificial data are similar to the real-world data.

We chose 200 randomly generated Lagrange multipliers and plotted, for each pair of Lagrange multiplier vectors, the relation between the euclidean distance in the constraint space and that in the Lagrangian space. Fig. 5 shows sample plotting results. Fig. 6 gives the correlation coefficients of three types of data according to K. The smaller the number of campaigns (K) is, the larger the correlation coefficient (ρ) is. We can easily show that when $K = 1$, $\rho \approx 1$. The uncorrelated data show a stronger correlation than correlated ones. However, for every experimental setting, it shows a strong positive correlation (greater than 0.5). The constraint space and the Lagrangian space are roughly isometric. The result shows that both spaces have a similar neighborhood structure. Therefore, this makes it easy to find high-quality Lagrange multipliers satisfying all the capacity constraints by considering the corresponding constraint space.

3.5 A Genetic Algorithm for Optimizing Lagrange Multipliers

A GA is a problem-solving technique motivated by Darwin's theory of natural selection in evolution. A GA starts with a set of initial solutions, which is called a *population*. Each solution in the population is called a *chromosome*, which is typically represented by a linear string. This population then evolves into different populations for a number of iterations

8. Strictly speaking, there cannot be a one-to-one correspondence in the technical sense of *bijection*. The capacity constraint vector has only integer components, so there are only countably many such vectors. However, the Lagrange multipliers are real numbers, so there are uncountably many Lagrange multiplier vectors. Several multiplier vectors may correspond to the same capacity constraint vector. Moreover, some multiplier vectors may have multiple capacity constraint vectors.

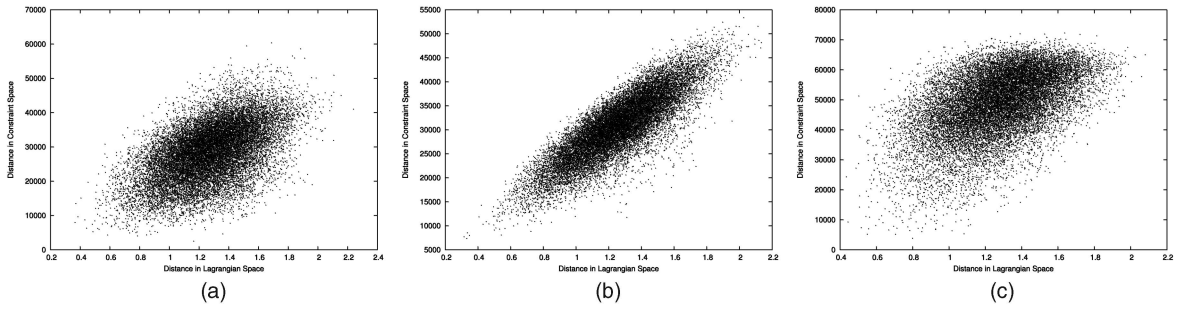


Fig. 5. The relationship between the constraint space and the Lagrangian space: sample plotting results when $K = 10$. (a) Real-world data. (b) Artificial data: uncorrelated. (c) Artificial data: correlated.

(generations). At the end, the algorithm returns the best chromosome of the population as the solution to the problem. For each iteration, the evolution proceeds as follows: Two solutions of the population are chosen based on some probability distribution. These two solutions are then combined through a *crossover* operator to produce an offspring. With a low probability, this offspring is then modified by a *mutation* operator to introduce an unexplored search space into the population, enhancing the diversity of the population. In this way, offspring are generated, and they *replace* part of or the whole population. The evolution process is repeated until a certain condition is satisfied, for example, after a fixed number of iterations. A GA that generates a considerable number of offspring per iteration is called a *generational* GA, as opposed to a *steady-state* GA, which generates only one offspring per iteration. A steady-state GA is known to converge faster than a generational one but is known to lose the diversity of the population faster. If we add a local improvement typically after the mutation step, the GA is called a *hybrid* GA [7], [20], [29]. Fig. 7 shows a typical hybrid steady-state GA.

We propose a GA for optimizing Lagrange multipliers. It conducts a search using an evaluation function with penalties for violated capacity constraints. The search space with N customers and K campaigns has $\Pi_{i=1}^{K} \sum_{j=\underline{b}_i}^{\bar{b}_i} \binom{N}{j}$ elements if all possibilities are considered. However, too large a problem size may make the search intractable. Our GA provides an alternative search method to find a good campaign assignment matrix by optimizing K Lagrange multipliers instead of directly dealing with the campaign assignment matrix.

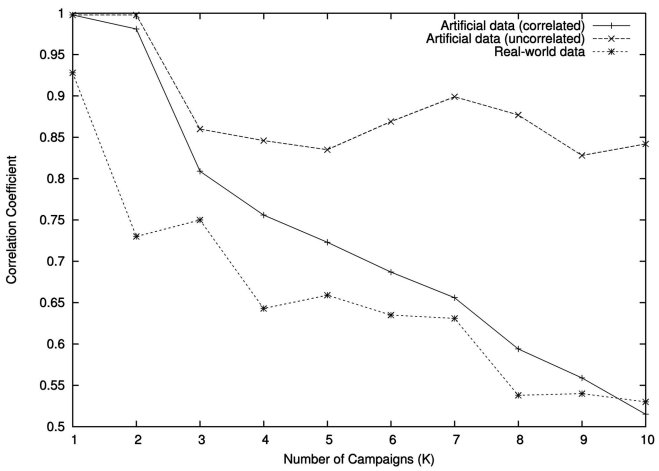


Fig. 6. The relationship between the constraint space and the Lagrangian space: correlation coefficient for each type of data and each K.

3.5.1 Genetic Operators

The general framework of a typical hybrid steady-state GA is used in our GA. In the following, we describe each part of the GA:

- *Encoding*. Each solution in the population is represented by a chromosome. Each chromosome consists of K genes corresponding to Lagrange multipliers. A real encoding is used for representing the chromosome $\boldsymbol{\lambda}$.
- *Initialization*. The GA first creates initial chromosomes at random. Each gene is set to be a real value between 0 and 1. We set the population size to be 100.
- *Selection and crossover*. To select two parents, we use a proportional selection scheme where the probability for the best solution to be chosen is four times higher than that for the worst solution. A crossover operator creates a new offspring by combining parts of the parents. We use the uniform crossover [38].
- *Mutation*. After the crossover, a mutation operator is applied to the offspring. We use a variant of Gaussian mutation. We devised it considering the relation between λ_i and $\tilde{b}_i$ given in Theorem 2. For each selected gene λ_i, we choose a Gaussian random real number t normally distributed with parameters $\mu = 0$ and $\sigma^2 = 1$ (that is, $t \sim N(0,1)$). If the corresponding capacity constraint $\tilde{b}_i$ is greater than $\bar{b}_i$, we set λ_i to $\lambda_i + (1 - \lambda_i)|t|/\gamma$. Otherwise, we set λ_i to $\lambda_i - \lambda_i|t|/\gamma$. For $|t|/\gamma$ to be less than 1 with a high

```
create an initial population of a fixed size;
do
    choose parent1 and parent2 from population;
    offspring ← crossover(parent1, parent2);
    mutation(offspring);
    local-improvement(offspring);
    replace(population, offspring);
until (stopping condition)
return the best individual;
```

Fig. 7. A typical hybrid steady-state GA.

probability of 0.99, we set the constant γ to 2.58. In spite of the above setting of γ, if λ_i becomes less than 0, we reset λ_i to 0. The mutation rate is 0.05.

- *Local improvement*. We use a simple local improvement based on Theorem 2. We randomly choose a gene λ_i. If the corresponding capacity constraint $\tilde{b}_i$ is greater than $\bar{b}_i$, we increase λ_i by Δ. Otherwise, we decrease λ_i by Δ. However, if this operation does not yield an improvement, we do not change the value of λ_i. We set Δ to 0.001.
- *Replacement and stopping condition*. After generating an offspring, our GA replaces the worse of the two parents with the offspring. It is called *preselection* replacement [8]. Our GA stops when one of the following two conditions is satisfied: 1) the number of generations reaches 15,000, or 2) the fitness of the worst chromosome is equal to that of the best one, that is, the fitness values of all chromosomes are equal.

3.5.2 Evaluation Function

Our evaluation function is to find a Lagrange multiplier vector $\boldsymbol{\lambda}$ that has a high fitness value satisfying the capacity constraints as *much* as possible. In our GA, the following evaluation function is used:

$$F(M) - \sum_{j=1}^{K} c_j \cdot (excess(j) - deficiency(j)),$$

where c_j is the campaign penalty, and $excess(j)$ and $deficiency(j)$ are the numbers of exceeded recommendations and deficient ones for campaign j, respectively (that is,

$$excess(j) = \max\left\{\sum_{i=1}^{N} m_{ij} - \bar{b}_j, 0\right\}$$

and $deficiency(j) = \max\{\underline{b}_j - \sum_{i=1}^{N} m_{ij}, 0\}$). For the campaign penalty c_j, we used K times the weighted average preference of campaign j (that is, $c_j = K w_j \frac{1}{N} \sum_{i=1}^{N} p_{ij}$).

4 Experimental Results

4.1 Testbed and Parameters

E-mail marketing is considered to be one of the most promising tools for Internet marketing. Its response rate is known to be much higher than direct mailing or banner ads [1]. These days e-mail marketing companies acquire the permission of customers, and this type of e-mail marketing has become well established as a legal and promising business model. We used a set of field data from an e-mail marketing company. The entire data set used in this study was provided by Optus Inc.

We used the preference values estimated by a CF[9] variant from a set of field data with 48,559 customers and 10 campaigns. Personal information[10] generates a number of independent variables; the only dependent variable is whether or not the customer has responded to the e-mail. Each preference value for the dependent variable was predicted as follows: For preference prediction, we used additional 8,650 training customers (C_t) with an actual response value for each campaign. Each customer has personal information with 168 binary variables (the subinformation was used in [24]). Given a campaign, the predicted preference value for each customer i is $\sum_{k \in C_t} \rho_{ik} f(k)$, where ρ_{ik} is the correlation value between customer i and training customer k, and $f(k)$ is the actual response of customer k ($f(k) = 0$ or 1). The correlation ρ_{ik} is defined to be α_{ik}/β_{ik}, where

$$\alpha_{ik} = \sum_{j=1}^{168} (j\text{th information of customer } i) \wedge (j\text{th information of customer } k)$$

and

$$\beta_{ik} = \sum_{j=1}^{168} (j\text{th information of customer } i) \vee (j\text{th information of customer } k).$$

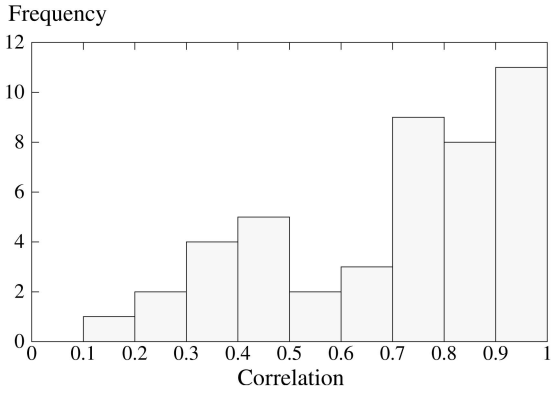


Fig. 8. Histogram for the Pearson correlation coefficient of each campaign pair.

As the response suppression function, we used the function R shown in Section 2.1.3, which was derived from a Gaussian function. The weight for each campaign was given equally a value of 0.1. Unless otherwise noted, the maximum number of recommendations for each campaign was set to 7,284, which is equal to 15 percent of the total number of customers, and the minimum number of recommendations was set to 0.

The average preference of customers for a campaign was 4.74, and the average standard deviation of preferences was 5.20. We examined the correlation coefficient of preferences for each pair of campaigns. Fig. 8 shows the histogram. Thirty-three pairs (about 73 percent) out of a total of 45 pairs showed a higher correlation coefficient than 0.5. This property of the field data provides a good reason for the need for MCAP modeling.

4.2 Analysis of Results

There are two initialization methods in multicampaign heuristics: random initialization and the CAA in Section 2.2.2. After initialization, we improve the solution by applying the iterative improvement heuristic in Section 2.2.3.

9. Collaborative filtering (CF) [34] is used to predict customer preferences for campaigns [4]. In particular, CF is fast and simple. Therefore, it is widely used for personalization in e-commerce [19], [22]. There have been a number of customer-preference estimation methods based on CF [2], [17], [37].

10. The information consists of a personal profile and past responses for other campaigns.

TABLE 1
Comparison for Small Data Sets ($K = 3$)

Method	$N = 200$	$N = 400$	$N = 600$	$N = 800$	$N = 1000$
Random-I	721.71	1279.07	1989.91	2663.31	3421.86
%-gap†	1.76	1.94	2.03	2.34	1.76
CAA	733.61	1303.98	2030.10	2725.59	3479.30
%-gap†	0.14	0.03	0.06	0.05	0.11
CAA-I	734.27	1304.16	2030.68	2726.54	3480.66
%-gap†	0.05	0.02	0.03	0.02	0.07
(Time*)	(0.0024)	(0.0050)	(0.0079)	(0.0110)	(0.0142)
LM-GA	734.33	**1304.43**	2030.65	2726.90	3479.75
%-gap†	0.04	**0.00**	0.03	0.003	0.09
(Time*)	(4.96)	(9.90)	(14.86)	(19.77)	(24.85)
LM-GA-P	**734.64**	**1304.43**	**2031.22**	**2726.99**	**3483.02**
%-gap†	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
(Time*)	(4.962)	(9.904)	(14.866)	(19.779)	(24.863)
DP‡	734.64	1304.43	2031.22	2726.99	3483.02
%-gap†	0.00	0.00	0.00	0.00	0.00
(Time*)	(145.82)	(2336.41)	(11553.92)	(28838.84)	(71479.06)
Upper bound	734.70	1304.49	2031.37	2727.29	3483.40
%-gap†	−0.01	−0.005	−0.01	−0.01	−0.01

Maximum 50 percent and minimum 0 percent recommendation for each campaign.
Equally weighted campaigns, that is, $w_j = 0.33$ *for each campaign* j.
† *Each value means* $100 \times (optimum - fitness)/optimum$.
‡ *The optimum value.*
Upper bound got from the subgradient method or Corollary 1 using the results of LM-GA.
∗ *CPU seconds on a Pentium III 1-GHz processor.*

We denote the methods "random initialization + iterative improvement" and "CAA + iterative improvement" by Random-I and CAA-I, respectively. The iterative improvement heuristic randomly chooses the order of campaigns to be applied. Therefore, it may produce different results for the same input. On the other hand, CAA is a deterministic algorithm that always outputs the same result.

First, we compare various algorithms with the DP algorithm, which guarantees optimality. LM-GA is the GA proposed in Section 3.5. When postprocessed by "constructive assignment"[11] for the unfilled capacity constraints and then by "iterative improvement," this version is called LM-GA-P. Due to the runtime of DP, we restricted the instances with up to 1,000 customers and three campaigns. Table 1 shows their performance. We chose three campaigns among the 10 campaigns, and the sets of customers were sampled at random to prepare the sets with sizes from 200 to 1,000. The maximum number of recommended customers was set to half of the total number of customers. The minimum number of recommendations was set to 0. CAA-I performed better than other heuristic methods, Random-I and CAA. CAA-I was much faster than the DP algorithm, and its results reached fairly close to the optimal solutions. This implies that CAA-I is an attractive practical heuristic. However, LM-GA was performed with practical runtime and was comparable to CAA-I. Moreover, there was a case that LM-GA found the optimal solution. It is surprising that LM-GA-P always found the optimal solutions.

Table 2 shows the performance of the independent campaign and various multicampaign algorithms in the multicampaign formulation. We use all the 48,559 customers and $7 \sim 10$ campaigns here. The results of "Independent" campaign come from K independent campaigns without considering their relationships with others. That is, the assignment matrix of "Independent" campaign is obtained by choosing the optimal assignment for each campaign, without considering the multiple-recommendation problem. Although the independent campaign was better than the "Random" assignment in the multicampaign formulation, it was not comparable to the other multicampaign algorithms. The results of the "Random" assignment are the average of more than 1,000 runs. The solution fitness values of the heuristics suggested in [21] were overall more than two times higher than those of the independent campaign.

Next, we compare the algorithms in [21] with the Lagrangian method. Table 2 shows their performance. LM-SUB means the modified subgradient algorithm given in Appendix B. LM-Local is a local optimization method with 15,000 iterations starting at a random initial multiplier vector. At each iteration, it performs *local improvement*, given in Section 3.5.1. LM-Random means the best result among randomly generated 15,000 Lagrange multiplier vectors. Even in the best result, there were capacity constraints violations for some campaigns, which implies that finding a valid Lagrange multiplier vector is hard. (The figures in LM-Random just mean the fitness values in Section 3.5.2.) However, when Lagrange multipliers are optimized by a GA (LM-GA), we always found the best quality feasible solutions satisfying all the capacity constraints. Moreover, LM-GA performed better than CAA-I and LM-SUB. Fig. 9 shows an example of 10 optimized Lagrange multipliers. For a typical randomly generated Lagrange multiplier vector, we can observe excessive violations of capacity constraints for some campaigns (Fig. 9a). However, with the Lagrange multipliers optimized by GA, we can see that all the capacity constraints were satisfied (Fig. 9b). Although LM-GA performed well, it seems that there is still room for further improvement since there are some unfilled capacity constraints (for example, the second, third, fourth, and eighth capacity constraints in Fig. 9b). When LM-GA is postprocessed by "constructive assignment" for the unfilled capacity constraints and then by "iterative improvement," it showed a further considerable improvement.

As a final solution of the LM-GA with $K = 10$ and a fitness value of 66,980.76, we got the following capacity constraint vector:

$$\boldsymbol{b} = (7, 254, 7, 269, 6, 238, 6, 982, 2, 422, 7, 284, 7, 237, 7, 282, 4, 675, 7, 260) \leq (7, 284, 7, 284, \ldots, 7, 284)$$

(see Fig. 9b). By Theorem 1, LMMCAP guarantees optimality over the capacity constraint vector $\boldsymbol{b}$ (the optimum value is 66,980.76). Table 3 shows the result of each algorithm when the vector $\boldsymbol{b}$ is used as the upper bound capacity constraint vector $\boldsymbol{b}^*$ of MCAP.[12] LM-GA-P outperformed the others except LM-SUB, and the results are very close to the optimal solutions. To the authors' best

11. This is a variant of CAA. With the result $M = (m_{ij})$ of LM-GA as an initial solution, CAA is performed for the remaining capacity constraints $\boldsymbol{b}^* - \mathbf{1}_N M \geq \mathbf{0}_K$.

12. This gives another usage of the proposed Lagrangian method. The suboptimality of heuristic algorithms can be measured by using the optimality of the Lagrangian method.

TABLE 2
Comparison of Algorithms over the Number of Campaigns

Method	$K = 7$	$K = 8$	$K = 9$	$K = 10$
	Fitness (%-gap‡)	Fitness (%-gap‡)	Fitness (%-gap‡)	Fitness (%-gap‡)
Independent	38546.25 (56.08)	36779.68 (54.83)	30259.65 (58.62)	26452.75 (60.76)
Random	32487.36 (62.99)	29385.14 (63.91)	27187.26 (62.82)	25951.19 (61.50)
Random-I†	81940.06 (6.64)	75709.32 (7.01)	68957.03 (5.70)	64497.40 (4.32)
CAA	85565.05 (2.51)	79885.40 (1.88)	71863.76 (1.73)	66473.63 (1.39)
CAA-I†	86651.98 (1.28)	80275.74 (1.40)	71900.82 (1.68)	66653.25 (1.13)
LM-Random†	45401.90 (48.27)	53638.58 (34.12)	52351.99 (28.41)	46990.46 (30.29)
LM-Local†	54216.04 (38.23)	80276.62 (1.40)	72289.22 (1.15)	66600.07 (1.20)
LM-GA†	86474.30 (1.48)	80287.72 (1.39)	72518.22 (0.83)	66932.79 (0.71)
LM-GA-P†	**87752.82 (0.02)**	**81399.00 (0.02)**	**72845.99 (0.38)**	**67143.80 (0.40)**
LM-SUB	85691.27 (2.37)	80148.85 (1.56)	68346.26 (6.54)	65869.70 (2.29)
Upper bound*	87771.28 (0.00)	81417.05 (0.00)	73127.31 (0.00)	67411.87 (0.00)

† *Average of 50 runs.*
‡ *Each value means* $100 \times (\text{upper bound} - fitness)/\text{upper bound}$.
∗ *Upper bound got from the subgradient method or Corollary 1 using the results of LM-GA.*
"Independent," CAA, and LM-SUB are deterministic algorithms.
All the values except for LM-Random and Upper bound came from feasible solutions.

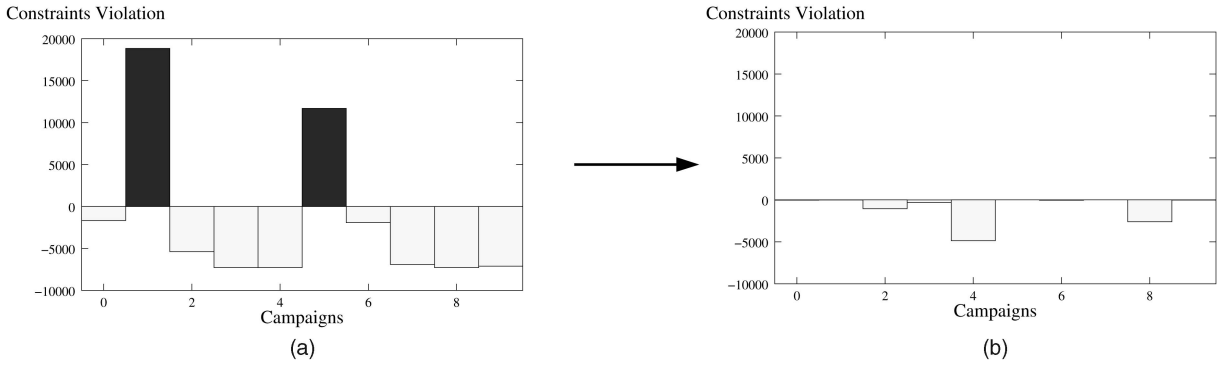


Fig. 9. An example of optimized Lagrange multipliers. (a) Randomly assigned Lagrange multipliers. (b) Lagrange multipliers optimized by GA.

knowledge, LM-SUB is one of the best algorithms to find an upper bound using Lagrange multipliers. Since in this instance the upper bound by LM-SUB is exactly equal to the optimum, it is not surprising that LM-SUB found the optimal solution.

Our Lagrangian method was much slower than previous heuristics because a considerable number of iterations of LMMCAP were required inside the GA. However, even when given the comparable runtime, the heuristics in [21] could never reach the result of the Lagrangian method (see the "Best" column in Table 3).

Finally, to see how sensitive the performance of various algorithms is to the capacity bound, we made additional experiments. Table 4 shows the performance of algorithms when the maximum number of recommendations are set to 10, 20, 30, and 40 percent of the total number of customers. Here, we used all the 48,559 customers and 10 campaigns. We got consistent results with previous experiments; LM-GA-P outperformed the others. For a large upper bound constraint (40 percent), the genetic Lagrangian methods performed similarly to the heuristic ones.

We showed the superiority of the proposed Lagrangian method. However, it is not omnipotent; it has some weaknesses. As hinted in Table 4, the bound constraints affect the performance of our method. For the upper bound constraints larger than some threshold, the genetic Lagrangian method may perform similar to the heuristic ones, CAA and CAA-I. Then, there is no merit to use the Lagrangian method, which spends more time. There is also another drawback. In fact, the final solution of the proposed Lagrangian method is not guaranteed to fall in the feasible region. In the case that $\boldsymbol{b}_* \approx \boldsymbol{b}^*$, there is a too small feasible region. Therefore, even finding feasible solutions may be too difficult.

In the experiments, we considered only the upper bound constraints. However, the lower bound constraints can be managed in a similar way to the upper bound constraints. Various experiments including the lower bound constraints will also be interesting. We leave this in future work.

5 Conclusions

In this paper, we dealt with an optimization issue occurring when conducting multiple personalized campaigns. We proposed LMMCAP. We also provided some theorems supporting our Lagrangian method. We showed their effectiveness with experiments for a real-world data set.

When we perform a small set of campaigns for a small number of customers, DP may be the choice. In most situations with a practical size, however, the DP becomes

TABLE 3
Suboptimality of Algorithms

Method	10 campaigns ($K = 10$)			
	Average (%-gap†)	CPU[3]	Best (%-gap†)	CPU[4]
Random-I[1]	62905.94 (6.084)	5.04	63790.12 (4.764)	15120
CAA	65819.80 (1.733)	1.33	–	–
CAA-I[1]	66091.28 (1.328)	1.33 + 2.36	66221.31 (1.134)	1.33 + 7080
LM-GA[2]	66962.09 (0.028)	5342	66966.30 (0.022)	267116
LM-GA-P[2]	66978.22 (0.004)	5342 + 1.50	66980.10 (0.001)	267116 + 75
LM-SUB	66980.76 (0.000)	7918	–	–
Optimum*	66980.76 (0.000)	–	–	–

1. From 3,000 runs.
2. From 50 runs.
3. CPU seconds on a Pentium III 1-GHz processor.
4. The summation of the CPU seconds of all the experiments.
† *Each value means* $100 \times (optimum - fitness)/optimum$.
∗ *The optimum computed by LMMCAP under the constraints.*
$\boldsymbol{b}^* = \{7, 254, 7, 269, 6, 238, 6, 982, 2, 422, 7, 284, 7, 237, 7, 282, 4, 675, 7, 260\}$.
CAA and LM-SUB are deterministic algorithms.

TABLE 4
Comparison of Algorithms over Various Capacity Constraints

Method	$\bar{b}_i = 4856$ (10%)	$\bar{b}_i = 9712$ (20%)	$\bar{b}_i = 14568$ (30%)	$\bar{b}_i = 19424$ (40%)
	Fitness (%-gap‡)	Fitness (%-gap‡)	Fitness (%-gap‡)	Fitness (%-gap‡)
Independent	26576.36 (54.95)	25511.13 (63.31)	23610.24 (66.73)	16871.03 (76.23)
Random	19634.09 (66.72)	29937.89 (56.94)	31339.56 (55.84)	26182.11 (63.11)
Random-I†	54735.70 (7.21)	67376.21 (3.09)	69096.43 (2.63)	69527.02 (2.02)
CAA	56848.77 (3.63)	68869.32 (0.94)	70451.03 (0.72)	70961.92 (0.003)
CAA-I†	58108.01 (1.49)	69071.37 (0.65)	70495.81 (0.66)	70961.92 (0.003)
LM-Random†	12681.06 (78.50)	65332.36 (6.03)	68734.34 (3.14)	69310.93 (2.33)
LM-Local†	55995.69 (5.07)	69278.83 (0.35)	70595.80 (0.52)	70581.96 (0.54)
LM-GA†	56633.06 (3.99)	69361.83 (0.23)	70621.59 (0.48)	70963.22 (0.001)
LM-GA-P†	**58808.38 (0.31)**	**69431.12 (0.13)**	**70678.33 (0.40)**	**70963.29 (0.001)**
LM-SUB	53348.30 (9.56)	69064.69 (0.66)	70089.26 (1.23)	Not available
Upper bound*	58988.82 (0.00)	69524.60 (0.00)	70963.93 (0.00)	70963.93 (0.00)

† *Average of 50 runs.*
‡ *Each value means* $100 \times (\text{upper bound} - fitness)/\text{upper bound}$.
∗ *Upper bound got from the subgradient method or Corollary 1 using the results of LM-GA.*
"Independent," CAA, and LM-SUB are deterministic algorithms.
All the values except for LM-Random and Upper bound came from feasible solutions.

intractable, and the Lagrangian method would be a tractable and an attractive choice. The suggested Lagrangian method takes practical runtime and outputs optimal solutions on some other constraint environment. However, it is not easy to find Lagrange multipliers satisfying all the capacity constraints. When our Lagrangian method is combined with a GA, we could find high-quality Lagrange multipliers and then reach good feasible solutions. Of course, the Lagrangian method can be combined with other metaheuristics as well, such as evolution strategy, simulated annealing, tabu search, and large-step Markov chains.

APPENDIX A
DUALITY

A.1 Notations

We denote several notations for convenience of describing the next theorem:

$$
\begin{aligned}
\Omega &= \{0,1\}^{N\times K},\\
\omega(\boldsymbol{z}_*, \boldsymbol{z}^*) &= \max_{M\in\Omega, \boldsymbol{z}_* \le \mathbf{1}_N M \le \boldsymbol{z}^*} \langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K M^T) M' \rangle,\\
\varphi^{\boldsymbol{b}_*,\boldsymbol{b}^*}(\boldsymbol{\lambda},\boldsymbol{\mu}) &= \max_{M\in\Omega} \{ \langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K M^T) M' \rangle - \langle \boldsymbol{\lambda}, \mathbf{1}_N M - \boldsymbol{b}^* \rangle\\
&\qquad - \langle \boldsymbol{\mu}, \boldsymbol{b}_* - \mathbf{1}_N M \rangle \},\\
M_{\boldsymbol{\lambda},\boldsymbol{\mu}} &= \arg\max_{M\in\Omega} \{ \langle \boldsymbol{w}, \boldsymbol{R}(\mathbf{1}_K M^T) M' \rangle - \langle \boldsymbol{\lambda}, \mathbf{1}_N M - \boldsymbol{b}^* \rangle\\
&\qquad - \langle \boldsymbol{\mu}, \boldsymbol{b}_* - \mathbf{1}_N M \rangle \},\\
\boldsymbol{b}_{\boldsymbol{\lambda},\boldsymbol{\mu}} &= \mathbf{1}_N M_{\boldsymbol{\lambda},\boldsymbol{\mu}}.
\end{aligned}
$$

$\{\boldsymbol{\lambda}, \boldsymbol{\mu}\}$ corresponds to $\{M_{\boldsymbol{\lambda},\boldsymbol{\mu}}, \boldsymbol{b}_{\boldsymbol{\lambda},\boldsymbol{\mu}}\}$ by the LMMCAP.

A.2 Lagrange Duality

If Ω is convex, it is known that the maximum of the objective function subject to the constraints is the same as the minimum of $\varphi^{\boldsymbol{b}_*,\boldsymbol{b}^*}(\boldsymbol{\lambda},\boldsymbol{\mu})$ for $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ [25]. This property is called *duality*. Since the domain of MCAP is not convex, this property does not hold in general. Nevertheless, we can compare the values of the optimum and $\varphi^{\boldsymbol{b}_*,\boldsymbol{b}^*}(\boldsymbol{\lambda},\boldsymbol{\mu})$ in the

```
LM_SUB( )
{
    λ ← 0, L ← φ, U ← φ;
    for i ← 1 to T
        α ← 1/(N + i); // decreasing step-size α
        (μ, M, b) ← LMMCAP(λ);
        if b* ≤ b ≤ b^ then L ← L ∪ {μ}; // if M is feasible
        U ← U ∪ {μ + ⟨λ, b^ − b⟩};
        λ ← λ + α · (b − b^ + γ_i 1_K); // maintaining λ ≥ 0_K
    return max L and min U;
    // where max L is the fitness of the best feasible solution (a lower bound),
    // min U is an upper bound, and
    // γ_i = { 0                 if i < T/3
             { ⌊(i − T/3)/3⌋    otherwise.
}
```

Fig. 10. A variant of the subgradient algorithm. $*$ A deterministic algorithm. $*$ In the experiments, we set T to 15,000.

following theorem. This is an example of weak duality in general integer programming [31], [33].

Theorem 4 (Weak duality).

$$\max_{\boldsymbol{b}_* \le \boldsymbol{z}_*, \boldsymbol{z}^* \le \boldsymbol{b}^*} \omega(\boldsymbol{z}_*, \boldsymbol{z}^*) \le \min_{\boldsymbol{\lambda} \ge \boldsymbol{0}_K, \boldsymbol{\mu} \ge \boldsymbol{0}_K} \varphi^{\boldsymbol{b}_*, \boldsymbol{b}^*}(\boldsymbol{\lambda}, \boldsymbol{\mu}).$$

Proof. For any $\boldsymbol{\lambda} \ge \boldsymbol{0}_K$ and $\boldsymbol{\mu} \ge \boldsymbol{0}_K$,

$$\begin{aligned}
\varphi^{\boldsymbol{b}_*, \boldsymbol{b}^*}(\boldsymbol{\lambda}, \boldsymbol{\mu}) &= \max_{M \in \Omega} \{\langle \boldsymbol{w}, \boldsymbol{R}(\boldsymbol{1}_K M^T) M' \rangle - \langle \boldsymbol{\lambda}, \boldsymbol{1}_N M - \boldsymbol{b}^* \rangle \\
&\quad - \langle \boldsymbol{\mu}, \boldsymbol{b}_* - \boldsymbol{1}_N M \rangle\} \\
&\ge \max_{M \in \Omega, \boldsymbol{b}_* \le \boldsymbol{1}_N M \le \boldsymbol{b}^*} \{\langle \boldsymbol{w}, \boldsymbol{R}(\boldsymbol{1}_K M^T) M' \rangle \\
&\quad - \langle \boldsymbol{\lambda}, \boldsymbol{1}_N M - \boldsymbol{b}^* \rangle - \langle \boldsymbol{\mu}, \boldsymbol{b}_* - \boldsymbol{1}_N M \rangle\} \\
&\ge \max_{M \in \Omega, \boldsymbol{b}_* \le \boldsymbol{1}_N M \le \boldsymbol{b}^*} \langle \boldsymbol{w}, \boldsymbol{R}(\boldsymbol{1}_K M^T) M' \rangle \\
&= \omega(\boldsymbol{b}_*, \boldsymbol{b}^*) \\
&= \max_{\boldsymbol{b}_* \le \boldsymbol{z}_*, \boldsymbol{z}^* \le \boldsymbol{b}^*} \omega(\boldsymbol{z}_*, \boldsymbol{z}^*). \\
&\quad (\because \omega \text{ is increasing on } \boldsymbol{b}^* \text{ and decreasing on } \boldsymbol{b}_*.)
\end{aligned}$$

$$\therefore \quad \max_{\boldsymbol{b}_* \le \boldsymbol{z}_*, \boldsymbol{z}^* \le \boldsymbol{b}^*} \omega(\boldsymbol{z}_*, \boldsymbol{z}^*) \le \min_{\boldsymbol{\lambda} \ge \boldsymbol{0}_K, \boldsymbol{\mu} \ge \boldsymbol{0}_K} \varphi^{\boldsymbol{b}_*, \boldsymbol{b}^*}(\boldsymbol{\lambda}, \boldsymbol{\mu}).$$

□

Corollary 2 (Strong duality). *If there exist* $\boldsymbol{\lambda}' \ge \boldsymbol{0}_K$ *and* $\boldsymbol{\mu}' \ge \boldsymbol{0}_K$ *such that* $\langle \boldsymbol{\lambda}', \boldsymbol{b}_{\boldsymbol{\lambda}', \boldsymbol{\mu}'} - \boldsymbol{b}^* \rangle = \langle \boldsymbol{\mu}', \boldsymbol{b}_* - \boldsymbol{b}_{\boldsymbol{\lambda}', \boldsymbol{\mu}'} \rangle = 0$ *and* $\boldsymbol{b}_* \le \boldsymbol{b}_{\boldsymbol{\lambda}', \boldsymbol{\mu}'} \le \boldsymbol{b}^*$, *then*

$$\max_{\boldsymbol{b}_* \le \boldsymbol{z}_*, \boldsymbol{z}^* \le \boldsymbol{b}^*} \omega(\boldsymbol{z}_*, \boldsymbol{z}^*) = \min_{\boldsymbol{\lambda} \ge \boldsymbol{0}_K, \boldsymbol{\mu} \ge \boldsymbol{0}_K} \varphi^{\boldsymbol{b}_*, \boldsymbol{b}^*}(\boldsymbol{\lambda}, \boldsymbol{\mu}).$$

Proof. By assumption,

$$\begin{aligned}
\min_{\boldsymbol{\lambda} \ge \boldsymbol{0}_K, \boldsymbol{\mu} \ge \boldsymbol{0}_K} \varphi^{\boldsymbol{b}_*, \boldsymbol{b}^*}(\boldsymbol{\lambda}, \boldsymbol{\mu}) &\le \varphi^{\boldsymbol{b}_*, \boldsymbol{b}^*}(\boldsymbol{\lambda}', \boldsymbol{\mu}') = \langle \boldsymbol{w}, \boldsymbol{R}(\boldsymbol{1}_K M^T_{\boldsymbol{\lambda}', \boldsymbol{\mu}'}) M'_{\boldsymbol{\lambda}', \boldsymbol{\mu}'} \rangle \\
&= \omega(\boldsymbol{b}_{\boldsymbol{\lambda}', \boldsymbol{\mu}'}, \boldsymbol{b}_{\boldsymbol{\lambda}', \boldsymbol{\mu}'}) \\
&\le \omega(\boldsymbol{b}_*, \boldsymbol{b}^*) = \max_{\boldsymbol{b}_* \le \boldsymbol{z}_*, \boldsymbol{z}^* \le \boldsymbol{b}^*} \omega(\boldsymbol{z}_*, \boldsymbol{z}^*).
\end{aligned}$$

Hence, by Theorem 4,

$$\max_{\boldsymbol{b}_* \le \boldsymbol{z}_*, \boldsymbol{z}^* \le \boldsymbol{b}^*} \omega(\boldsymbol{z}_*, \boldsymbol{z}^*) = \min_{\boldsymbol{\lambda} \ge \boldsymbol{0}_K, \boldsymbol{\mu} \ge \boldsymbol{0}_K} \varphi^{\boldsymbol{b}_*, \boldsymbol{b}^*}(\boldsymbol{\lambda}, \boldsymbol{\mu}).$$

□

APPENDIX B
SUBGRADIENT METHOD

Coping with the nondifferentiability of the Lagrangian leads to the last technical development: *subgradient algorithm*. The subgradient algorithm is a fundamentally simple procedure. Typically, the subgradient algorithm has been used as a technique for generating good upper bounds for branch-and-bound methods, where it is known as *Lagrangian relaxation* [16], [27]. The reader is referred to [14] and [36] for the deep survey of Lagrangian relaxation. However, it can also be used as a heuristic for finding lower bounds (feasible solutions): At each iteration, simply check if M is feasible and, if so, report μ as a lower bound (keeping it if it is the best value reported so far) [26]. Fig. 10 shows a variant of the subgradient algorithm for finding a lower bound, as well as an upper bound. In the preliminary test, the traditional subgradient algorithm [13] hardly finds feasible solutions. Therefore, we use a modified algorithm by introducing a factor γ_i (when γ_i is always 0, it is exactly the traditional subgradient algorithm).

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their helpful comments and suggestions that improved the quality of this paper. The present research has been conducted by the Research Grant of Kwangwoon University in 2007. The ICT at Seoul National University provided the research facilities for this study. A preliminary version of this paper appeared in the *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 2, pp. 1065-1077, 2004.

REFERENCES

[1] "Email Marketing Maximized," Insight Report 2000, Peppers and Rogers Group, 2000.

[2] C.C. Aggarwal, J.L. Wolf, K.L. Wu, and P.S. Yu, "Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering," *Proc. Fifth ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD '99),* pp. 201-212, 1999.

[3] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993.

[4] M.J.A. Berry and G. Linoff, *Data Mining Techniques: For Marketing, Sales, and Customer Support.* John Wiley & Sons, 1997.

[5] M.J.A. Berry and G. Linoff, *Mastering Data Mining: The Art and Science of Customer Relationship Management.* John Wiley & Sons, 2000.

[6] M. Blum, R.W. Floyd, V. Pratt, R.L. Rivest, and R.E. Tarjan, "Time Bounds for Selection," *J. Computer and System Sciences,* vol. 7, no. 4, pp. 448-461, 1973.

[7] T.N. Bui and B.R. Moon, "Genetic Algorithm and Graph Partitioning," *IEEE Trans. Computers,* vol. 45, no. 7, pp. 841-855, July 1996.

[8] D. Cavicchio, "Adaptive Search Using Simulated Evolution," PhD dissertation, Univ. of Michigan, Ann Arbor, 1970.

[9] E. Cela, *The Quadratic Assignment Problem: Theory and Applications.* Kluwer Academic Publishers, 1998.

[10] Y.-J. Chang and B.W. Wah, "Lagrangian Techniques for Solving a Class of Zero-One Integer Linear Programs," *Proc. 19th Ann. Int'l Computer Software and Applications Conf. (COMPSAC '95),* pp. 156-161, 1995.

[11] P.C. Chu and J.E. Beasley, "A Genetic Algorithm for the Generalized Assignment Problem," *Computers and Operations Research,* vol. 24, pp. 17-23, 1997.

[12] J. Dyché, *The CRM Handbook: A Business Guide to Customer Relationship Management.* Addison-Wesley, 2001.

[13] H. Everett, "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," *Operations Research,* vol. 11, pp. 399-417, 1963.

[14] M.L. Fisher, "The Lagrangian Relaxation Method for Solving Integer Programming Problems," *Management Science,* vol. 27, no. 1, pp. 1-18, 1981.

[15] B. Gavish, "On Obtaining the "Best" Multipliers for a Lagrangean Relaxation for Integer Programming," *Computers and Operations Research,* vol. 5, pp. 55-71, 1978.

[16] A.M. Geoffrion, "Lagrangean Relaxation for Integer Programming," *Math. Programming Study,* vol. 2, pp. 82-114, 1974.

[17] D. Greening, "Building Consumer Trust with Accurate Product Recommendations," Technical Report LMWSWP-210-6966, LikeMinds white paper, 1997.

[18] S. Handdadi and H. Ouzia, "An Effective Lagrangian Heuristic for the Generalized Assignment Problem," *Information Systems and Operational Research,* vol. 39, no. 4, 2001.

[19] J.L. Herlocker, J.A. Konstan, A. Borchers, and J. Riedl, "An Algorithmic Framework for Performing Collaborative Filtering," *Proc. ACM SIGIR '99,* pp. 230-237, 1999.

[20] Y.H. Kim and B.R. Moon, "Lock-Gain Based Graph Partitioning," *J. Heuristics,* vol. 10, no. 1, pp. 37-57, Jan. 2004.

[21] Y.H. Kim and B.R. Moon, "Multicampaign Assignment Problem," *IEEE Trans. Knowledge and Data Eng.,* vol. 18, no. 3, pp. 405-414, Mar. 2006.

[22] J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gordan, and J. Riedl, "GroupLens: Applying Collaborative Filtering to Usenet News," *Comm. ACM,* vol. 40, pp. 77-87, 1997.

[23] H.W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly,* vol. 2, pp. 83-97, 1955.

[24] Y.K. Kwon and B.R. Moon, "Personalized Email Marketing with a Genetic Programming Circuit Model," *Proc. Genetic and Evolutionary Computation Conf. (GECCO '01),* pp. 1352-1358, 2001.

[25] D.G. Luenberger, *Optimization by Vector Space Methods.* John Wiley & Sons, 1969.

[26] M.J. Magazine and O. Oguz, "A Heuristic Algorithm for the Multidimensional Zero-One Knapsack Problem," *European J. Operational Research,* vol. 16, pp. 319-326, 1984.

[27] R.K. Martin, *Large Scale Linear and Integer Optimization: A Unified Approach.* Kluwer Academic Publishers, 1998.

[28] L. McCauley and S. Franklin, "A Large-Scale Multi-Agent System for Navy Personnel Distribution," *Connection Science,* vol. 14, no. 4, pp. 371-385, 2002.

[29] P. Merz and B. Freisleben, "Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem," *IEEE Trans. Evolutionary Computation,* vol. 4, no. 4, pp. 337-352, 2000.

[30] M.G. Narciso and L.A.N. Lorena, "Lagrangean/Surrogate Relaxation for Generalized Assignment Problems," *European J. Operational Research,* vol. 114, pp. 165-177, 1999.

[31] G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization.* John Wiley & Sons, 1988.

[32] G. Raidl, "Weight-Codings in a Genetic Algorithm for the Multiconstraint Knapsack Problem," *Proc. Congress on Evolutionary Computation (CEC '99),* vol. 1, pp. 596-603, 1999.

[33] *Modern Heuristic Techniques for Combinatorial Problems,* C.R. Reeves, ed. Halsted Press, 1993.

[34] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl, "GroupLens: An Open Architecture for Collaborative Filtering of Netnews," *Proc. Computer Supported Collaborative Work Conf. (CSCW '94),* pp. 175-186, 1994.

[35] D. Schuurmans, F. Southey, and R.C. Holte, "The Exponentiated Subgradient Algorithm for Heuristic Boolean Programming," *Proc. 17th Int'l Joint Conf. Artificial Intelligence (IJCAI '01),* pp. 334-341, 2001.

[36] J.F. Shapiro, "A Survey of Lagrangian Techniques for Discrete Optimization," *Annals of Discrete Math.,* vol. 5, pp. 113-138, 1979.

[37] U. Shardanand and P. Maes, "Social Information Filtering: Algorithms for Automating "Word of Mouth"," *Proc. ACM Conf. Human Factors in Computing Systems (CHI '95),* vol. 1, pp. 210-217, 1995.

[38] G. Syswerda, "Uniform Crossover in Genetic Algorithms," *Proc. Third Int'l Conf. Genetic Algorithms (ICGA '89),* pp. 2-9, 1989.

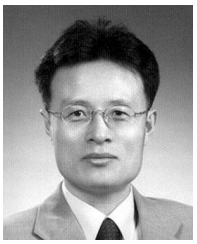
[39] B.W. Wah and Z. Wu, "The Theory of Discrete Lagrange Multipliers for Nonlinear Discrete Optimization," *Proc. Fifth Int'l Conf. Principles and Practice of Constraint Programming (CP '99),* pp. 28-42, 1999.

[40] Y. Yoon, Y.H. Kim, and B.R. Moon, "An Evolutionary Lagrangian Method for the 0/1 Multiple Knapsack Problem," *Proc. Genetic and Evolutionary Computation Conf. (GECCO '05),* pp. 629-635, 2005.

[41] Y. Yoon, Y.H. Kim, and B.R. Moon, "On the Computational Intractability of Multicampaign Assignment," technical report, preprint, http://soar.snu.ac.kr/~yhdfly/mcap-npc.pdf, 2007.

Yong-Hyuk Kim received the BS degree in computer science and the MS and PhD degrees in computer science and engineering from Seoul National University (SNU), Seoul, in 1999, 2001, and 2005, respectively. From March 2005 to February 2007, he was a postdoctoral scholar at SNU and also a research staff member in the Inter-University Semiconductor Research Center, SNU. Since March 2007, he has been an assistant professor in the Department of Computer Science, Kwangwoon University, Seoul. His research interests include algorithm design/analysis, discrete mathematics, optimization theory, combinatorial optimization, evolutionary computation, operations research, and data/Web mining. He was a committee member of GECCO 2005-2006 and has been a reviewer for several journals (*IEEE Intelligent Systems*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, and *IEEE Transactions on Software Engineering*) of the IEEE Computer Society since 2003.

Yourim Yoon received the BS degree in computer engineering from Seoul National University (SNU), Seoul, in 2003. She is currently working toward the PhD degree in the School of Computer Science and Engineering (CSE), SNU. She is also the chief of the Optimization Laboratory, School of CSE, SNU. Her research interests include optimization theory, combinatorial optimization, evolutionary computation, discrete mathematics, and operations research. She served as a reviewer for BIC-TA 2007.



Byung-Ro Moon received the BS degree in computer science and statistics from Seoul National University (SNU), Seoul, in 1985, the MS degree in computer science from the Korea Advanced Institute of Science and Technology, Seoul, in 1987, and the PhD degree in computer science from Pennsylvania State University, University Park, in 1994. From 1987 to 1991, he was an associate research engineer in the Central Research Laboratory, LG Electronic Co., Ltd., Seoul. From November 1994 through 1995, he was a postdoctoral scholar in the VLSI CAD Laboratory, University of California, Los Angeles. From 1996 to 1997, he was a principal research staff member in the DT Research Center, LG Semicon Ltd., Seoul. Since September 1997, he has been a professor in the School of Computer Science and Engineering, SNU. He leads the Optimization Laboratory and is also the CEO of Optus Inc., a company specialized in optimization. He has developed various optimization solutions for the industry. His major interest is the theory and application of optimization methodologies (evolutionary computation, algorithm design/analysis, optimization modeling, etc.). The applications of optimization include combinatorics, personalized marketing, e-commerce, search, text mining, graph partitioning, scheduling, and system identification. He is a member of the ACM and the IEEE. He was the recipient of a Korean Government Scholarship during academic years 1991-1994. He served as the publication chair of IEEE CEC 2001 and a committee member of GP 1997 and 1998, ISIAC 1998, and GECCO 1999-2006.