Multicampaign Assignment Problem

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

Abstract—It is crucial to maximize targeting efficiency and customer satisfaction in personalized marketing. State-of-the-art techniques for targeting focus on the optimization of individual campaigns. Our motivation is the belief that the effectiveness of a campaign with respect to a customer is affected by how many precedent campaigns have been recently delivered to the customer. We raise the multiple recommendation problem, which occurs when performing several personalized campaigns simultaneously. We formulate the multicampaign assignment problem to solve this issue and propose algorithms for the problem. The algorithms include dynamic programming and efficient heuristic methods. We verify by experiments the effectiveness of the problem formulation and the proposed algorithms.

Index Terms—Personalized marketing, multicampaign assignment, dynamic programming, heuristic algorithms.

1 Introduction

Customer relationship management (CRM) [12] is crucial in acquiring and maintaining loyal customers. To maximize revenue and customer satisfaction, companies try to provide personalized services for customers. A representative effort is one-to-one marketing [30]. The fast development of Internet and mobile communication has enhanced the market for one-to-one marketing. A personalized campaign targets the most attractive customers with respect to the subject of the campaign. It helps boost customers' loyalty by providing the most attractive contents to each customer or by locating the most appropriate set of customers for an arbitrary advertisement [10]. So, it is important to predict customer preferences for campaigns. This is another independent research area. A survey for recommender systems in e-commerce was given in [32], [21].

So far, targeting has focused on individual campaigns. In a single campaign, the preference prediction is the most important for marketing efficiency. Collaborative filtering[1] (CF) [15], [31] and various data mining techniques [9], [14], [19], including clustering [22] and the nearest-neighbor algorithm [13], are used to predict customer preferences for a campaign [6]. Since, especially, CF is fast and simple, it is widely used for targeting in e-commerce [23], [17], [18], [29]. There have been a number of customer-preference estimation methods based on CF [33], [16], [4]. For instance, *Amazon* (www.amazon.com) adopted the CF-based recommendation engine by *E.piphany* [27], [2]. In Korea, many companies, e.g., *eNet*, also use the CF-based targeting engine for e-business.

1. Collaborative filtering selects content based on the opinions of other customers with similar preferences. It is a proven standard for personalized recommendations [17], [23] which has been used by companies such as NetPerceptions, Inc. Although it is considered as one of the most successful personalization technologies, it still has some problems such as the sparsity problem [20].

• *The authors are with the School of Computer Science and Engineering, Seoul National University, Sillim-dong, Gwanak-gu, Seoul, 151-744 Korea. E-mail: {yhdfly, moon}@soar.snu.ac.kr.*

Manuscript received 16 May 2005; revised 5 Sept. 2005; accepted 7 Sept. 2005; published online 18 Jan. 2006.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0187-0505.

As personalized campaigns are frequently performed, several campaigns often happen to run simultaneously. It is often the case that an attractive customer for a campaign tends to be attractive for other campaigns, too. If we perform independent campaigns without considering this problem, some customers may be bombarded with a large number of campaigns, which is sometimes called "churning." We call this the multiple recommendation problem. The larger the number of recommendations for a customer, the lower the average interest for campaigns [7]. In the long run, reckless campaigns lower marketing efficiency as well as customer satisfaction and loyalty. Unfortunately, traditional methods do focus on the effectiveness of a single campaign and did not consider the risk with respect to the multiple recommendations. In the situation where several campaigns are performed at the same time, it is necessary to distribute the campaigns carefully over the customers.

In this paper, we present a new model for multiple campaigns beyond a single campaign. We formalize the multicampaign assignment problem (MCAP), which considers the multiple recommendation problem, and propose a number of algorithms for the issue. We also verify the effectiveness and the limit of the proposed algorithms with field data. We should note that the multicampaign here has a different meaning from Multichannel, which some companies use for that meaning in the field.

The remainder of this paper is organized as follows: In Section 2, we describe the multicampaign assignment problem. The description of response suppression function, one of the key elements for the problem, is given in Section 3. In Section 4, we propose algorithms for the problem. We show experimental results in Section 5 and, finally, make conclusions in Section 6.

2 Problem Formulation

The multicampaign assignment problem (MCAP) is a complex assignment problem in which each of N customers is assigned a corresponding subset drawn from a set of K campaigns. The goal is to find a set of assignments such that the outcome of campaigns is maximized under some constraints. The main difference from independent campaigns lies in that the customer response for campaigns is influenced by multiple recommendations.

In typical assignment problems such as the optimal assignment problem [24] and the quadratic assignment problem [8], the size of the domain set is equal to that of codomain set, thus restricting solutions to one-to-one mappings. However, in some others, such as the sailor assignment problem [28], the sizes are different. In the multicampaign assignment problem, the number of customers is much greater than the number of campaigns, i.e., $N \gg K$.

Let N be the number of customers, $C = \{1, 2, \ldots, N\}$ be the set of customers, and K be the number of campaigns. We describe the input, output, constraints, and evaluation function for the problem in the following section.

2.1 Input

For each customer, the preference for each campaign is given. Each campaign has a weight. A response suppression function R related to multiple recommendations is given.

- $f_1, f_2, \ldots, f_K : C \longrightarrow \mathbb{R}^+ \cup \{0\}$: The preference function (actually a vector) of each campaign.[2] The preferences for a campaign can be acquired from some existing preference-prediction method such as CF. $f_j(i)$ means the predicted preference of customer i for campaign j.
- $R : \mathbb{N} \longrightarrow [0, 1]$: The response suppression function with respect to the number of recommendations. The function value $R(k)$ indicates the factor of preference degradation when a customer encounters k recommendations simultaneously (in a fairly short period of time). If H_i is the number of recommendations for customer i, the *actual preference* of customer i for campaign j becomes $R(H_i) \cdot f_j(i)$.
- $w_1, w_2, \ldots, w_K$: The weight of each campaign ($w_j > 0$ for each campaign j). Each campaign j has its weight w_j. The importance of a campaign is determined by its weight. In multicampaign assignment formulation, the preference of customer i for campaign j becomes $w_j \cdot R(H_i) \cdot f_j(i)$.

2.2 Constraints

The multicampaign assignment problem has two types of constraints. The maximum and the minimum numbers of recommendations for each campaign are enforced.[3] The number of recommendations in campaign j should be between P_j and P^j, where P_j and P^j are the minimum and the maximum numbers of recommendations for campaign j, respectively. These constraints are determined by considering balance among campaigns as well as the cost of recommendation. They can be tight or loose according to the situation of business.

2.3 Output

The output is a binary campaign assignment matrix $M = (m_{ij})$ in which m_{ij} indicates whether campaign j is delivered to customer i; $m_{ij} = 1$ if campaign j is delivered to customer i. Fig. 1 shows an example campaign assignment matrix.

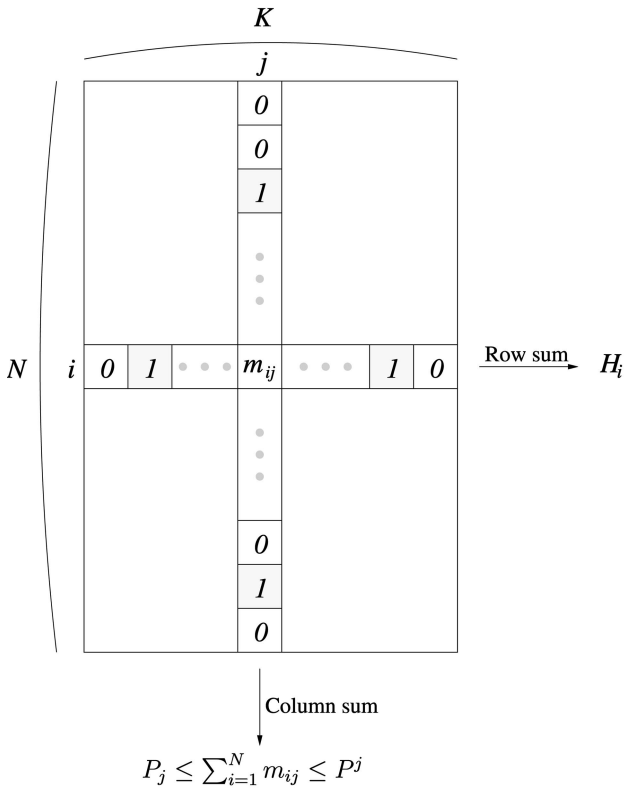


Fig. 1. The campaign assignment matrix $M = (m_{ij})$.

2.4 Evaluation

The *preference sum* for campaign j is defined to be the sum of actual preferences of recommended customers for campaign j. The fitness of a campaign assignment matrix M is the weighted sum of the preference sums over all campaigns as follows:

$$F : \{M = (m_{ij})\} \subset 2^{\{0,1\}^{N \times K}} \longrightarrow \mathbb{R}^+ \cup \{0\},$$

where

$$F(M) = \sum_{j=1}^{K} \left(w_j \cdot \sum_{i \in C; m_{ij}=1} R(H_i) \cdot f_j(i) \right).$$

The objective is to find a matrix $M = (m_{ij})$ that maximizes the fitness F.

2.5 Nomenclature

The nomenclature that would be used in the remainder of this paper is given in the list below.

- $\boldsymbol{w} = (w_1, w_2, \ldots, w_K) \in \mathbb{R}^K$: the campaign weight vector.
- $f_j : C \longrightarrow \mathbb{R}^+ \cup \{0\}$: the preference function for each campaign j.
- $R : \mathbb{N} \longrightarrow [0, 1]$: the response suppression function (for technical convenience, we assume $R(0) = 0$).
- $\boldsymbol{p^*} = (P^1, P^2, \ldots, P^K) \in \mathbb{N}^K$: the upper-bound constraint vector.

2. We consider only nonnegative preferences, which are represented by nonnegative real numbers.

3. The constraint about the minimum number of recommendations is important. With no such constraint, some campaigns are impossible to deliver to any customers. Sometimes, the maximum number of recommendations is enforced by the advertisers' budgets. Some other times it is enforced to not discriminate against other less attractive campaigns.

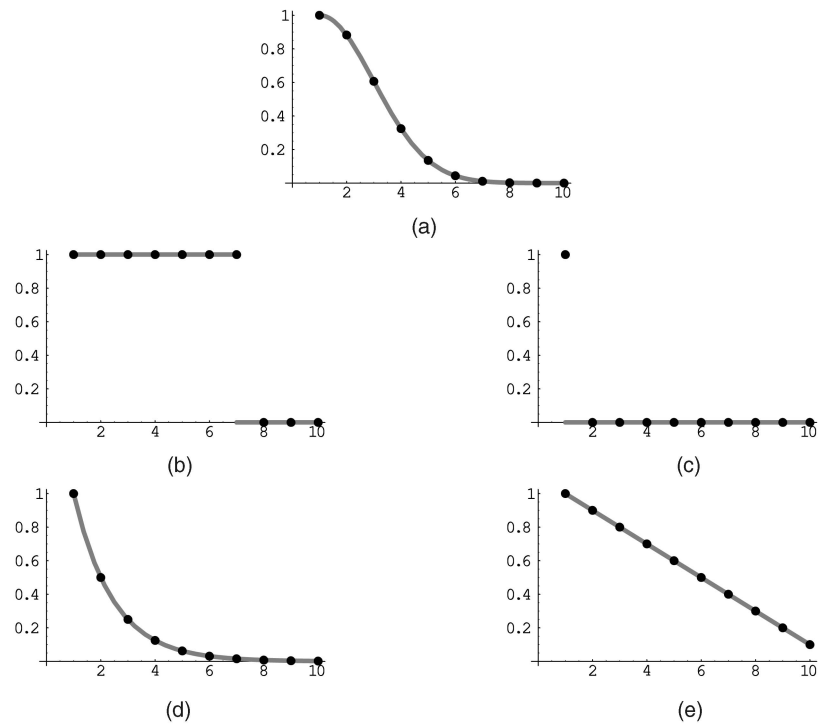


Fig. 2. Basic response suppression functions ($R_*(x) = 0$ for $x \geq 11$). (a) $R_0(x) = e^{-(x-1)^2/8}$. (b) $R_1(x)$. (c) $R_2(x)$. (d) $R_3(x) = 2^{-(x-1)}$. (e) $R_4(x) = -\frac{x-1}{10} + 1$.

- $\boldsymbol{p}_* = (P_1, P_2, \ldots, P_K) \in \mathbb{N}^K$: the lower-bound constraint vector.
- $\boldsymbol{v} = (v_1, v_2, \ldots, v_K) \in \mathbb{N}^K$: the constraint vector.
- $M = (m_{ij})$: the $N \times K$ binary campaign assignment matrix.
- $M' = (m'_{ij})$: the $N \times K$ real matrix where $m'_{ij} = f_j(i) \cdot m_{ij}$.
- $\boldsymbol{m}_i = (m_{i1}, m_{i2}, \ldots, m_{iK})$: the ith row vector of the matrix M.
- $\boldsymbol{m}'_i = (m'_{i1}, m'_{i2}, \ldots, m'_{iK})$: the ith row vector of the matrix M'.
- $\mathbf{1}_n$: the n-dimensional vector $(1, 1, \ldots, 1)$.
- $\mathbf{0}_n$: the n-dimensional vector $(0, 0, \ldots, 0)$.
- $H_i = \boldsymbol{m}_i \cdot \mathbf{1}_K^T$: the number of recommendations for customer i.
- $\sigma_i = \boldsymbol{m}'_i \cdot \boldsymbol{w}^T$: the weighted sum of preferences of customer i for recommended campaigns.

2.6 Formal Definition

More formally, MCAP is defined as follows:

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

$$\boldsymbol{R}(\mathbf{1}_K M^T) \cdot M' \cdot \boldsymbol{w}^T \text{ subject to } \boldsymbol{p}_* \leq \mathbf{1}_N M \leq \boldsymbol{p}^*,$$

where $\boldsymbol{R}(x_1, x_2, \ldots, x_n) = (R(x_1), R(x_2), \ldots, R(x_n))$ and each n-dimensional vector is regarded as a $1 \times n$ matrix.

3 The Response Suppression Function

In the case of multiple campaign recommendations, the customer response rate drops as the number of recommendations increases. We introduce the response suppression function, which reflects the response-rate degradation with multiple recommendations. The optimal response suppression function depends on situations and it is a long-term research topic. Instead, we devise a number of suppression functions. The functions should be monotonic nonincreasing.

Fig. 2 shows five basic response suppression functions. These functions are nonnegative monotonic nonincreasing functions with the maximum value one. R_0 was derived from the Gaussian function. By the function, the preference for a campaign drops to, e.g., one third when four campaigns are performed simultaneously to a customer. R_1 and R_2 are simple step functions. R_3 and R_4 decrease exponentially and linearly, respectively. In this paper, we use the function R_0 as the major response suppression function (see Fig. 2a) and use the other functions for supplementary study.

4 Methods

4.1 Dynamic Programming

We can find the optimal campaign assignment matrix of MCAP using dynamic programming (DP). DP has been useful for diverse problems [5], [11]. We define the optimal value function $S_i(\boldsymbol{v})$ to be the optimal fitness of MCAP with the fixed constraint vector $\boldsymbol{v}$ and the customer set $\{1, 2, \ldots, i\}$, where each element v_j in $\boldsymbol{v}$ means the number of recommendations in campaign j. By the principle of optimality, the recurrence relation appropriate to the definition is

$$S_i(\boldsymbol{v}) = max_{\boldsymbol{m}_i; \forall j, m_{ij} \leq v_j}(S_{i-1}(\boldsymbol{v} - \boldsymbol{m}_i) + R(H_i) \cdot \sigma_i)$$

for each $i = 1, 2, \ldots, N$ and each $\boldsymbol{v} \leq \boldsymbol{p}^*$. Fig. 3 shows the DP algorithm based on the recurrence relation. Having

```
DP(R, p_*, p^*)
{
    // boundary condition
    S_0(0_K) = 0;
    for each v such that 0_K ≠ v ≤ p^*
        S_0(v) = −∞;

    // forward filling of the table S
    for i = 1 to N
        for each v ≤ p^*
        {
            S_i(v) = max_{m_i; ∀j,m_ij≤v_j}(S_{i−1}(v − m_i) + R(H_i) · σ_i);
            L_i(v) = argmax_{m_i; ∀j,m_ij≤v_j}(S_{i−1}(v − m_i) + R(H_i) · σ_i); // backward link
        }
    optimal_value = max_{v; p_*≤v≤p^*} S_N(v);

    // backward tracing to find the optimal assignment M = (m_ij)
    rv = argmax_{v; p_*≤v≤p^*} S_N(v);
    for i = N to 1
    {
        m_i = L_i(rv);
        rv = rv − m_i;
    }

    return optimal_value and M = (m_ij);
}
```

Fig. 3. Dynamic programming for MCAP.

indexed the customers by arbitrary numbers from 1 to N, we take the constraint vector $\boldsymbol{v}$ and first assign a vector $\boldsymbol{m}_i$ to customer i. Then, we assign a vector $\boldsymbol{m}_{i-1}$ of the remaining constraint $\boldsymbol{v} - \boldsymbol{m}_i$ to customer $i-1$, etc. That is, with some assignments having been made, we identify the remaining constraint vector and the customer to be considered next. In this sequential manner, we maintain the optimal value for each possible constraint vector $\boldsymbol{v}$. The vector already assigned and, hence, the amount left are essential to the optimization of the remaining process. The algorithm requires $O(NK \cdot \Pi_{j=1}^{K} P^j)$ space for storing tables S and L. Since the maximum number of $\boldsymbol{m}_i$ configurations is 2^K, it takes $O(NK2^K \cdot \Pi_{j=1}^{K} P^j)$ time. If K is a fixed number, this is a polynomial-time algorithm. However, when K is not small and almost all P^js are $\Omega(N)$, it is nearly intractable. An optimal assignment matrix is obtained by backward links $L_i(\boldsymbol{v})$ stored during the process of DP. The proposed DP algorithm is only applicable to problems with small K, N pairs. Thus, we need heuristics for large problems. However, since the DP algorithm guarantees optimal solutions, it is also useful in evaluating the suboptimality of other heuristics proposed in the next section. It will be used for this purpose in Section 5.

4.2 Heuristic Algorithms

4.2.1 Constructive Assignment Algorithm

Starting at the situation that no customer is recommended any campaigns, we iteratively assign campaigns to customers by a greedy method. We call this algorithm the constructive assignment algorithm (CAA). Define the *gain* $g_{(i,j)}$ of a pair (customer i, campaign j) to be the amount of fitness gain by assigning campaign j to customer i. Initially, the gain $g_{(i,j)}$ is equal to $w_j f_j(i)$, the product of campaign weight and the preference of customer i for campaign j. Generally, the gain is formulated as:

$$g_{(i,j)} = R(H_i + 1) \cdot (\sigma_i + w_j f_j(i)) - R(H_i) \cdot \sigma_i.$$

We use an AVL tree [3] for the efficient management of real-valued gains. Fig. 4 shows the template of CAA. First, the most attractive α customers for each campaign are chosen and inserted them into the AVL tree. Then, we iteratively perform the following: We choose 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, we assign campaign j to customer i. If an assignment is done, we update the gains of customer i for the other campaigns. We naturally assume that the response suppression function is monotonic nonincreasing. Any gain then cannot be positive after the maximum gain drops below zero. When the maximum gain is not positive, the algorithm terminates as far as every campaign satisfies the constraint on the minimum number of recommendations. There are $O(NK)$ nodes in the AVL tree during a run of the algorithm. Hence, both the insertion and deletion of a node in the AVL tree take $O(\log(NK))$ time, i.e., $O(\log N)$. The time complexity of the algorithm becomes $O(NK^2 \log N)$. If the algorithm is implemented without an AVL tree, it would take $O(N^2K^3)$.

4.2.2 Iterative Improvement

We also propose an iterative improvement heuristic. After every customer is assigned a proper number of campaigns, we can run this heuristic. It proceeds in a series of *passes*. During each pass, the heuristic improves on an initial

```
M = O;
Create an AVL tree;
for each campaign j
        Find topmost α customers with high gain values, and
            for each customer i of them, insert the node (g_(i,j), (i, j))
            into the AVL tree;
do {
        Choose a maximum gain node (i, j) and delete it from the AVL tree;
        if (g_(i,j) is not positive and every campaign satisfies
            its constraint on the minimum number of recommendations)
                then break;
        if (the campaign j is not full) then {
                Recommend the campaign j to the customer i; // m_ij ← 1
                for each not-full campaign k
                        Update the gain values g_(i,k) in the AVL tree;
        }
} until (the AVL tree is empty or every campaign is full)
```

$*$ We set α to be $N/2$ in our experiments.

Fig. 4. The constructive assignment algorithm (CAA).

```
// A_j, B_j: the sets of recommended customers
//           and non-recommended customers, respectively, for each campaign j.
do {
        for each campaign j
        {
                Compute the gain values g_a^j, g_b^j for each a ∈ A_j, b ∈ B_j;
                Sort the g_a^j's and the g_b^j's in decreasing order;
                Choose k that maximizes Σ_{i=1}^{k} (g_{a_i}^j + g_{b_i}^j);
                Swap the subsets {a_1, a_2, ..., a_k} and {b_1, b_2, ..., b_k};
        }
} until (there is no improvement)
```

Fig. 5. The iterative improvement heuristic.

solution to create a new solution. This process is repeated until no improvement can be obtained.

Let A_j and B_j be the sets of recommended customers and nonrecommended customers, respectively, for campaign j. For a recommended customer $a \in A_j$, the gain g_a^j of campaign j is defined to be the fitness gain by *cancelling* the assignment to customer a. For a nonrecommended customer $b \in B_j$, the gain g_b^j of campaign j is defined to be the fitness gain by *assigning* campaign j to customer b. Formally,

$$g_a^j = R(H_a - 1) \cdot (\sigma_a - w_j f_j(a)) - R(H_a) \cdot \sigma_a \text{ for each } a \in A_j,$$

$$g_b^j = R(H_b + 1) \cdot (\sigma_b + w_j f_j(b)) - R(H_b) \cdot \sigma_b \text{ for each } b \in B_j.$$

Fig. 5 shows the template of the iterative improvement heuristic. A pass of iterative improvement is performed 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. The time complexity of one pass is $O(NK \log N)$.

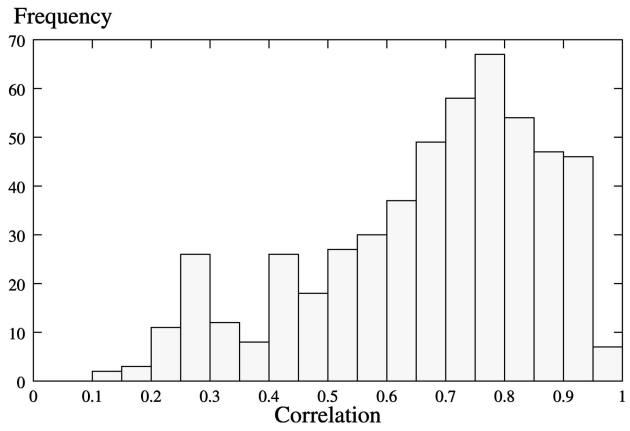


Fig. 6. Histogram for Pearson correlation coefficient of each campaign pair.

TABLE 1
Comparison of Algorithms

Method	Best*	Ave*	Std	Time‡
Independent	–	10068.36	–	–
Random	8076.55†	7902.69†	61.38†	–
Random-I	31713.46†	31280.43†	155.01†	29.77
CAA	33048.49	33048.49	–	14.72
CAA-I	33245.83†	33212.29†	12.45†	14.72+21.17

$N = 48559$ (# of customers) and $K = 33$ (# of campaigns).
Max 5% recommendation and min 0% recommendation for each campaign.
Equally weighted campaigns, i.e., $w_j = 0.03$ for each campaign j.
The response suppression function $R(x) = R_0(x)$.
$*$ Each value means the fitness value $F(M)$ described in Section 2.
† From 1,000 runs. CAA is a deterministic algorithm.
‡ CPU seconds on Pentium III 1 GHz.

5 Experimental Results

5.1 Test Set and Parameters

E-mail marketing[4] is considered 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 from 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. We used the preference values estimated by a CF variant from a set of field data with 48,559 customers and 33 campaigns. The entire data set used in our study were provided by Optus Inc. Personal information[5] generates a number of independent variables; the only dependent variable is whether the customer has answered or not to the e-mail. Each preference value for the dependent variable is predicted as follows: For preference prediction, we used an 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 [25].) Given a campaign, the predicted preference value for each customer i is $\sum_{k \in C_t} \rho_{ik} \cdot f(k)$, where ρ_{ik} is the correlation value[6] between customer i and training customer k, and $f(k)$ is the actual response of customer k($f(k) = 0$ or 1).

Unless otherwise noted, we use the Gaussian function R_0 as the response suppression function. We set the same weight for every campaign. The maximum number of recommendations for each campaign was equally set to 2,428, 5 percent of the total number of customers. The minimum number of recommendations was set to 0.

Each predicted preference value ranged from 0.00 to 167.73. The average predicted preference of customers for a campaign was 4.67 and the average standard deviation of predicted preferences was 5.05, which indicates that the distribution was not normal. We also examined the Pearson correlation coefficient[7] of predicted preferences for every pair of campaigns. Fig. 6 shows its histogram. Four hundred twenty-two pairs (about 80 percent) among the total 528 pairs showed higher correlation coefficients than 0.5. This property of field data raises the risk of independent multiple recommendation and thus provides a good reason for the need of MCAP modeling.

5.2 Analysis of Results

Table 1 shows the performance of the independent campaign and various multicampaign heuristics in the multicampaign formulation.[8] The result of independent campaign is from 33 independent campaigns without considering their relationships with others. There are two initialization methods in multicampaign heuristics: random initialization or CAA of Section 4.2.1. After initialization, we improve the solution by applying the iterative improvement heuristic. We denote the methods "random initialization + iterative improvement" and "CAA + iterative improvement" by Random-I and CAA-I, respectively. Our iterative improvement heuristic randomly chooses the order of campaigns to be applied. So, it may produce different results from the same input. On the other hand, CAA is a

4. E-mail campaign (direct e-mailing) is one of the cheapest campaign methods. The cost of sending an e-mail is \$5 to \$7 per 1,000 messages [26]. We chose e-mail campaigns in the experiments for experimental convenience.

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

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

7. When we consider a campaign as a vector of predicted preference values for customers, it means the correlation coefficient between a pair of campaigns, i.e., $(f_i(1), f_i(2), \ldots, f_i(N))$ and $(f_j(1), f_j(2), \ldots, f_j(N))$, $i \neq j$.

8. If we ignore the constraint factor, it is optimal to solve the MCAP independently for each customer. However, this produces ill-balanced recommendations for campaigns. When we solved the MCAP independently for each customer under the experimental setting of Table 1 (where $\boldsymbol{p}^* = 2,428 \cdot \mathbf{1}_N$), $\mathbf{1}_N M$ (the number of recommendations for campaigns) was the vector (2959, 0, 1692, 104, 0, 4169, 69, 0, 0, 178, 167, 29538, 0, 0, 6393, 189, 80, 0, 36, 0, 0, 334, 682, 31215, 0, 2610, 526, 0, 0, 8, 950, 93, 7). Surprisingly, 12 campaigns were assigned no customers and six campaigns were assigned more than the maximum number of recommendations—among them, two campaigns recommended more than 12 times the maximum number of recommendations. This result ignores some campaigns and exceedingly concentrates on some specific campaigns.

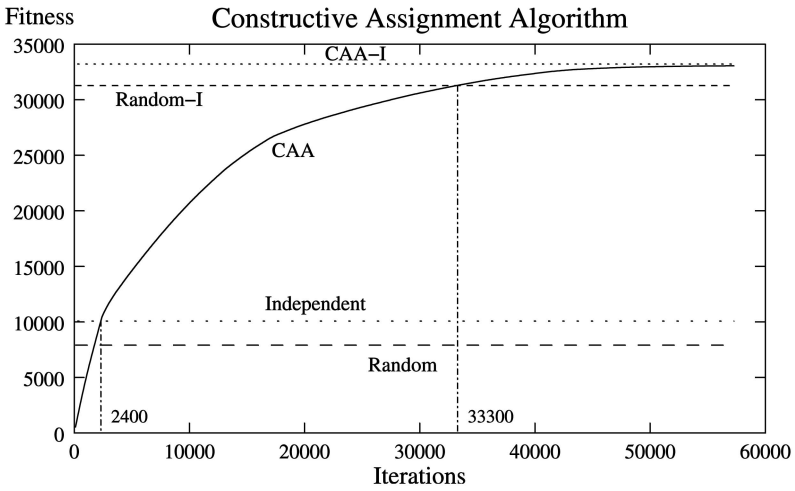
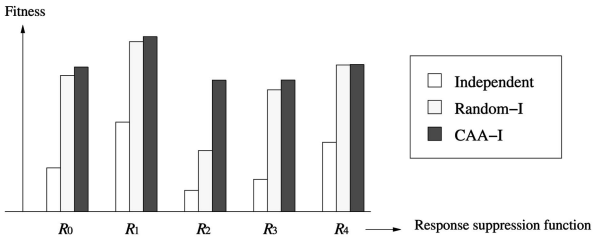


Fig. 7. Performance of CAA.



Method	R_0	R_1	R_2	R_3	R_4
Independent	10068.36	20582.88	4883.20	7413.27	15923.29
Random-I	31280.43	39041.82	14031.61	27977.04	33691.92
CAA-I	33212.29	40218.96	30209.76	30229.19	33800.37

Average over 1,000 runs.
$R_0(x) = e^{-(x-1)^2/8}$: modified Gaussian function.
$R_1(x)$: simple step function.
$R_2(x)$: simple step function.
$R_3(x) = 2^{-(x-1)}$: exponentially decreasing function.
$R_4(x) = -\frac{x-1}{10} + 1$: linearly decreasing function.

Fig. 8. Results with various response suppression functions.

deterministic algorithm which always outputs the same result. All the methods except CAA were performed 1,000 times. Although the independent campaign was better than the random assignment in multicampaign formulation, it was not comparable to the other multi-campaign heuristics. The random initialization combined with iterative improvement (Random-I) was not only slower than CAA, but even the best of 1,000 runs was worse than the result of CAA. Fig. 7 shows fitness values over the iterations in CAA. At no more than 2,400 node insertions, CAA surpassed the independent campaign. At 33,300 node insertions, CAA outperformed the iterative improvement heuristic. However, the iterative improvement heuristic was useful for improving the results of CAA. The solution fitness of CAA-I heuristic was more than three times higher than that of the independent campaign. This result shows the attractiveness of the proposed model.

We also performed experiments on various response suppression functions. Fig. 8 shows the results according to different response suppression functions. The details of the functions were given in Fig. 2. For all the response suppression functions, the results by MCAP model showed significant improvement over the independent-campaign model. The slower the customer response for multiple recommendations decreased, the smaller the difference between two models was. The response suppression function R_1 was set to have the same response up to seven multiple recommendations. Surprisingly, the result of MCAP was still around two times higher than even that of the independent campaign with R_1.

We additionally compare the proposed heuristic algorithms with the DP algorithm which guarantees the optimal solutions. Due to the huge time complexity

TABLE 2
Comparison for Small Data Set

Method	$N = 100$	$N = 200$	$N = 300$	$N = 400$	$N = 500$
Independent	286.18	628.33	848.42	1093.58	1371.02
Random-I†	331.99	721.71	992.20	1279.07	1598.69
CAA-I†	337.51	734.27	1009.87	1304.16	1632.04
(Time*)	(0.0011)	(0.0024)	(0.0037)	(0.0050)	(0.0063)
DP‡	337.51	734.64	1010.09	1304.43	1632.20
(Time*)	(9.26)	(145.82)	(755.04)	(2336.41)	(5654.93)

Method	$N = 600$	$N = 700$	$N = 800$	$N = 900$	$N = 1000$
Independent	1697.14	1979.68	2280.35	2651.34	2945.03
Random-I†	1989.91	2324.90	2663.31	3077.42	3421.86
CAA-I†	2030.68	2368.01	2726.54	3132.84	3480.66
(Time*)	(0.0079)	(0.0094)	(0.0110)	(0.0126)	(0.0142)
DP‡	2031.22	2368.44	2726.99	3135.04	3483.02
(Time*)	(11553.92)	(19469.34)	(28838.84)	(46011.66)	(71479.06)

$K = 3$ (# of campaigns).
Max 50% recommendation and min 0% recommendation for each campaign.
Equally weighted campaigns, i.e., $w_j = 0.33$ for each campaign j.
The response suppression function $R(x) = R_0(x)$.
† Average results over 1,000 trials.
‡ The optimal value.
∗ CPU seconds on Pentium III 1 GHz.

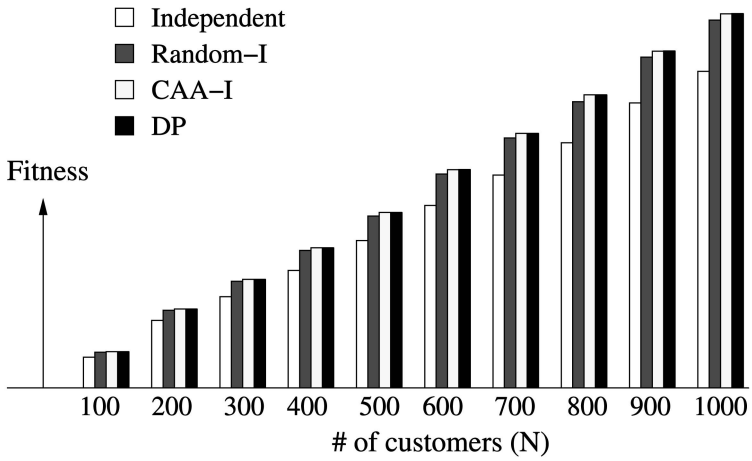


Fig. 9. Plotting of Table 2.

of the DP algorithm, we restricted the instances to 1,000 customers and three campaigns. Table 2 shows their suboptimality. We chose three campaigns among the 33 campaigns and the sets of customers were sampled at random to prepare the sets with sizes from 100 to 1,000. The maximum number of recommended customers was set to be at half the total number of customers. The minimum number of recommendations was set to 0. CAA-I was much faster than the DP method, while its results reached fairly close to the optimal solutions. Although it is notable that CAA-I was visibly better than Random-I, a more important thing is that both Random-I and CAA-I produced much better results than "Independent" campaign. Fig. 9 plots the results of table. Although this does not guarantee that CAA-I also produces near-optimal solutions for large data, it supports that CAA-I is an attractive practical heuristic. For the problem of Section 5.1 with 48,559 customers and 33 campaigns, we unfortunately do not know the optimum due to the huge time requirement of the DP algorithm.

We also examined the distribution of customers over the numbers of recommended campaigns. Fig. 10, captured from a commercial product that incorporated the proposed algorithms, shows the distributions with respect to Independent campaign and CAA-I, respectively. The

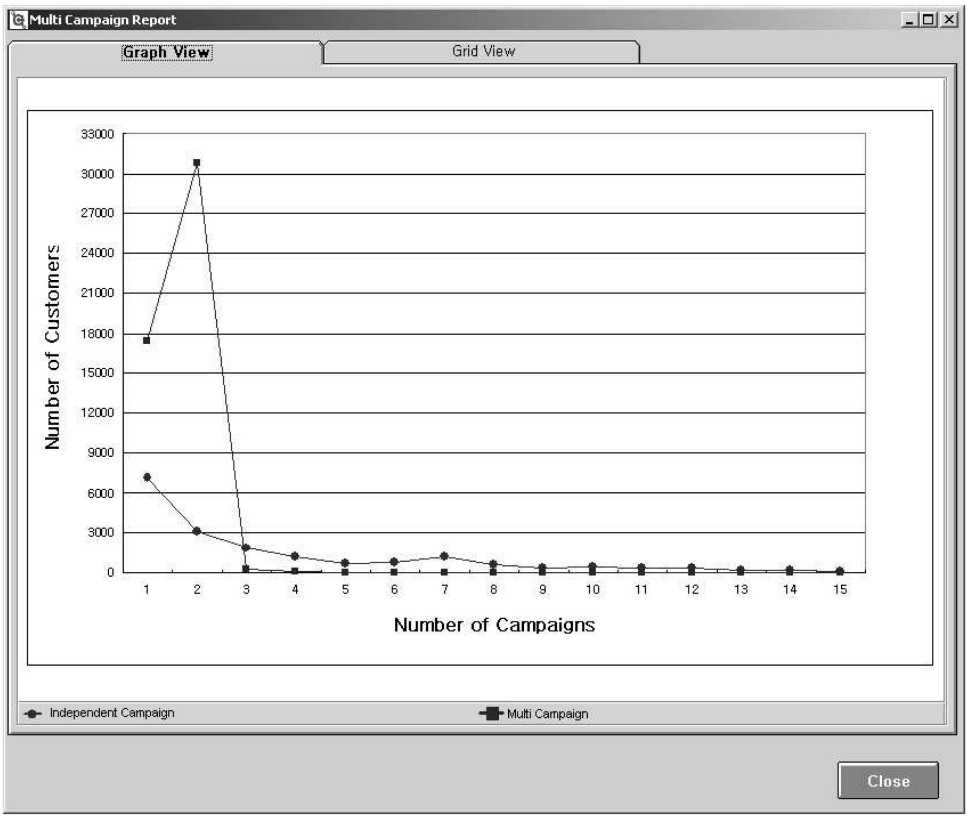


Fig. 10. Distribution of customers over the number of recommended campaigns.

results are from 33 campaigns. The two methods showed a sharp contrast. One can observe that there are a considerable number of customers who got more than three campaigns in Independent campaign, whereas there are few in CAA-I.[9] The campaigns were more evenly assigned to customers in CAA-I.

6 Conclusions

The representative contributions of this paper are as follows: First, we proposed and formulated the multicampaign assignment problem (MCAP). To the best of our knowledge, this is the first model that attacks multiple campaigns from the perspective of optimization. Second, we devised a dynamic programming (DP) algorithm for MCAP, which guarantees the optimality. Finally, we proposed heuristics for MCAP. Their performance was examined with various experiments. It should be noted that CAA showed impressive performance as just a stand-alone heuristic algorithm. For those problems for which we know the optimal solutions by the proposed DP algorithms, CAA reached fairly close to the optimal solutions. CAA was also improved by the iterative improvement heuristic. For an instance with 1,000 customers and three campaigns, CAA-I produced near-optimal solutions in 5 millionth of the time taken by the DP algorithm. When we perform a small set of campaigns for a small number of customers, DP may be the choice. In most situations, however, CAA-I would be the practical choice. CAA-I takes practical running time; for a set of real field data with roughly 50,000 customers and 33 campaigns, it took just around 36 seconds on a Pentium III 1 GHz.

We showed in Section 5 that the MCAP model can avoid bombarding campaigns on a number of customers. The customers who were targeted by five or more campaigns are mostly loyal customers to the company. Such reckless campaigns diminish the loyalty of highly valuable customers. By the MCAP model, we not only improve the overall response rate, but avoid harming the loyalty of customers.

Although the DP algorithm is impractical as its current form, it would be able to handle larger problems by dimensional reduction techniques. Lagrangian relaxation is a good candidate for dimensional reduction. We leave this issue as another research topic. We hope that our study provides the motivation of further researches for multi campaign assignment optimization.

9. In an extreme case, we found that a customer was assigned as many as 28 campaigns among 33 in the "Independent" campaign.

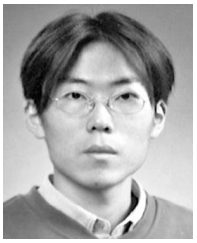
Acknowledgments

The authors would like to thank Yung-Keun Kwon, Yoon-Seok Choi, and the anonymous reviewers for their helpful comments and suggestions. This work was supported by Optus Inc. and the Brain Korea 21 Project. The ICT at Seoul National University provided some research facilities.

References

[1] *E-mail Marketing Maximized, Insight Report 2000.* Peppers and Rogers Group, 2000.
[2] "The Epiphany E6 Architecture J2EE, Open Standards-Based CRM," technical white paper, Epiphany, Nov. 2004.

[3] G.M. Adel'son-Vel'skii and E.M. Landis, "An Algorithm for the Organization of Information," *Soviet Math. Doklady,* vol. 3, pp. 1259-1262, 1962.
[4] 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. Knowledge Discovery and Data Mining Conf.,* pp. 201-212, 1999.
[5] R. Bellman, *Dynamic Programming.* Princeton Univ. Press, 1957.
[6] M.J.A. Berry and G. Linoff, *Data Mining Techniques for Marketing, Sales, and Customer Support.* John Wiley and Sons, 1997.
[7] M.J.A. Berry and G. Linoff, *Mastering Data Mining, the Art and Science of Customer Relationship Management.* John Wiley and Sons, 2000.
[8] E. Cela, *The Quadratic Assignment Problem: Theory and Applications.* Kluwer Academic, 1998.
[9] M.S. Chen, P.S. Han, and J. Yu, "Data Mining: An Overview from a Database Perspective," *IEEE Trans. Knowledge and Data Eng.,* vol. 8, no. 6, pp. 866-883, Dec. 1996.
[10] R. Dewan, B. Jing, and A. Seidmann, "One-to-One Marketing on the Internet," *Proc. 20th Int'l Conf. Information Systems,* pp. 93-102, 1999.
[11] S.E. Dreyfus and A.M. Law, *The Art and Theory of Dynamic Programming.* Academic Press, 1977.
[12] J. Dyché, *The CRM Handbook: A Business Guide to Customer Relationship Management.* Addison-Wesley, 2001.
[13] C. Feustel and L. Shapiro, "The Nearest Neighbor Problem in an Abstract Metric Space," *Pattern Recognition Letters,* vol. 1, pp. 125-128, 1982.
[14] M. Goebel and L. Gruenwald, "A Survey of Data Mining and Knowledge Discovery Software Tools," *SIGKDD Explorations,* vol. 1, pp. 20-33, 1999.
[15] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry, "Using Collaborative Filtering to Weave an Information Tapestry," *Comm. ACM,* vol. 35, no. 12, pp. 61-70, 1992.
[16] D. Greening, "Building Consumer Trust with Accurate Product Recommendations," Technical Report LMWSWP-210-6966, LikeMinds white paper, 1997.
[17] J.L. Herlocker, J.A. Konstan, A. Borchers, and J. Riedl, "An Algorithmic Framework for Performing Collaborative Filtering," *Proc. 22nd Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* pp. 230-237, 1999.
[18] J.L. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl, "Evaluating Collaborative Filtering Recommender Systems," *ACM Trans. Information Systems,* vol. 22, no. 1, pp. 5-53, 2004.
[19] M. Holsheimer and L. Meindertsma, "Data Mining Integrated in CRM," *Informatie,* vol. 41, pp. 10-16, 1999.
[20] Z. Huang, H. Chen, and D. Zeng, "Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering," *ACM Trans. Information Systems,* vol. 22, no. 1, pp. 116-142, 2004.
[21] Z. Huang, D. Zeng, and H. Chen, "A Comparative Study of Recommendation Algorithms in E-Commerce Applications," *IEEE Intelligent Systems,* 2004.
[22] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data.* Prentice Hall, 1988.
[23] 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.
[24] H.W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly,* vol. 2, pp. 83-97, 1955.
[25] Y.K. Kwon and B.R. Moon, "Personalized E-Mail Marketing with a Genetic Programming Circuit Model," *Proc. Genetic and Evolutionary Computation Conf.,* pp. 1352-1358, 2001.
[26] B.A.S. Martin, J.V. Durme, M. Raulas, and M. Merisavo, "E-Mail Advertising: Exploratory Insights from Finland," *J. Advertising Research,* vol. 43, no. 3, pp. 293-300, 2003.
[27] A. Maurino and P. Fraternali, "Commercial Tools for the Development of Personalized Web Applications: A Survey," *Proc. Third Int'l Conf. E-Commerce and Web Technologies,* pp. 99-108, 2002.
[28] L. McCauley and S. Franklin, "A Large-Scale Multiagent System for Navy Personnel Distribution," *Connection Science,* vol. 14, no. 4, pp. 371-385, 2002.
[29] B.N. Miller, J.A. Konstan, and J.T. Riedl, "PocketLens: Toward a Personal Recommender System," *ACM Trans. Information Systems,* vol. 22, no. 3, pp. 437-476, 2004.
[30] D. Peppers and M. Rogers, *The One to One Future.* Doubleday and Company, 1996.
[31] 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.,* pp. 175-186, 1994.
[32] J.B. Schafer, J.A. Konstan, and J.T. Riedl, "Recommender Systems in E-Commerce," *Proc. ACM Conf. Electronic Commerce,* pp. 158-166, 1999.
[33] U. Shardanand and P. Maes, "Social Information Filtering: Algorithms for Automating 'Word of Mouth'," *Proc. ACM CHI '95 Conf. Human Factors in Computing Systems,* vol. 1, pp. 210-217, 1995.



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, Korea, in 1999, 2001, and 2005, respectively. He is currently a postdoctoral fellow in School of Mechanical and Aerospace Engineering at SNU and also a research staff member in the Inter-University Semiconductor Research Center at SNU. His research interests include algorithm design/analysis, optimization theory, combinatorial optimization, evolutionary computation, and data mining. He served as a committee member of GECCO'2005 and a reviewer for *IEEE Intelligent Systems* and *IEEE Transactions on Knowledge and Data Engineering*. He is a member of the ACM.

Byung-Ro Moon received the BS degree in computer science and statistics from Seoul National University, Seoul, Korea, the MS degree in computer science from the Korea Advanced Institute of Science and Technology, Seoul, Korea, and the PhD degree in computer science from the Pennsylvania State University, University Park, in 1985, 1987, and 1994, respectively. From 1987 to 1991, he was an associate research engineer in the Central Research Laboratory, LG Electronic Company, Ltd., Seoul, Korea. From November 1994 through 1995, he was a postdoctoral scholar in the VLSI CAD Lab, 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, Korea. Since September 1997, he has been an associate professor in the School of Computer Science and Engineering, Seoul National University, Seoul, Korea. 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 was the recipient of a Korean Government Scholarship during the 1991-1994 academic years. He served as the publication chair of the IEEE CEC '2001, a committee member of GP '97-98, ISIAC '98, and GECCO '99-2005. 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 industry. He is a member of the IEEE.