A Novel Particle Swarm Optimization for Multiple Campaigns Assignment Problem

Satchidananda Dehuri
Soft Computing Laboratory
Department of Computer Science
Yonsei University, 262 Seongsanro
Sudaemoon-ku, Seoul 120-749, Korea
satchi.lapa@gmail.com

Sung-Bae Cho
Soft Computing Laboratory
Department of Computer Science
Yonsei University, 262 Seongsanro
Sudaemoon-ku, Seoul 120-749, Korea
sbcho@cs.yonsei.ac.kr

ABSTRACT
This paper presents a novel swarm intelligence approach to optimize simultaneously multiple campaigns assignment problem, which is a kind of searching problem aiming to find out a customer-campaign matrix to maximize the outcome of multiple campaigns under certain restrictions. It is treated as a very challenging problem in marketing. In personalized marketing it is very important to optimize the *customer satisfaction* and *targeting efficiency*. Particle swarm optimization (PSO) method can be chosen as a suitable tool to overcome the multiple recommendation problems that occur when several personalized campaigns conducting simultaneously. Compared with original PSO we have modified the particle representation and velocity by a multi-dimensional matrix, which represents the customer-campaign assignment. A new operator known as REPAIRED is introduced to restrict the particle within the domain of solution space. The proposed operator helps the particle to fly into the better solution areas more quickly and discover the near optimal solution. We measure the effectiveness of the propose method with two other methods know as *Random* and *Independent* using randomly created customer-campaign preference matrix. Further a generalized Gaussian response suppression function is introduced and it differs among customer classes. An extensive simulation studies are carried out varying on the small to large scale of the customer-campaign assignment matrix and the percentage of recommendations. Simulation result shows a clear edge between PSO and other two methods.

Categories and Subject Descriptors:
I.6.5[**Computing Methodologies**]:Simulation and Modeling-Model Development.

General Terms
Algorithms, Management, Design, Verification.

CSTST 2008, October 27-31, 2008, Cergy-Pontoise, France.

Keywords
Particle swarm optimization, multiple campaign assignment problem, Gaussian function.

1. INTRODUCTION
The rapid development of Internet technologies and hand-held devices like cell phones has attracted a large number of companies ranging from small to large scale, to provide personalized services for customers. In order to maintain and acquire the loyal customers Customer Relationship Management (CRM) [1] plays a vital role. The companies need to provide personalized services to customers and take major steps toward one-to-one marketing [2]. A personalized campaign often targets the most attractive customers with respect to the subject of the campaign, and subject of preferences for compaign. Therefore, it is very important to expect customer preferences for campaigns. There have been a number of customer-preference estimation method based on collaborative filtering (CF) [3, 4]. Collaborative filtering, k-nearest neighbour [7], and various data mining methods like clustering [5, 6], association rule mining [8] and fuzzy association rule mining[9] are used to predict the customer prefernces for a campaign.

As personalized campaigns are frequently performed, multiple campaigns often happen to run simulatnaously or witihn a very short span of time. It is often the case that an attractive customer for a specific campaign tends to be inclined for other campaigns. If we conduct independent campaigns without considering other campaigns, some customers may be attacked by a considerable number of campaigns. This problem is known as multiple recommendation problem. The larger the number of recommendations for a customer, the lower the customer interest for campaigns [10]. In the long run, hasty campaigns can lower marketing efficiency as well as customer staisfication and loyalty. Unfortunately, traditionals methods like collaborative filtering 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 span, it is necessary to find the optimal campaign assignment to customers considering the recommendations in other campaigns.

The multiple campaign assignment problem is a very complex combinatorial optimization problem in which each of N customers is assigned to one of the subset of K campaigns and $N >> K$. To the best of our knowledge a very limited number of approaches has been developed to solve this problem. The approaches such as dynamic programming (DP), heuristics and Langragian approach developed by Kim et al. [11, 12] have been used to workout this problem with some limitations. For example, the dynamic programming approach becomes intractable if the size of the problem is very large, and the heuristic methods with greedy assignment do not guarentee optimal solutions. As we know the greedy methods always make decision besed on what is best at the moment, with no concern about how this might effect the overall picture. To cope with these problems we propose a novel particle swarm optimization with a newly introduced operator called REPAIRED. This operator helps the particle of the swarm to fly into the better solution areas more quickly and discover good potential solution. Further, as it incorporates information gain measure-a kind of knowledge to repair a particle, so it avoids the premature convergence and stagnation to some extent.

Further, the important motivating factor to solve this kind of problem using particle swarm optimization, is to replace the legacy system by an intelligent system. In the sequel, this problem has incredible potential to solve many problems in the diverse fields such as *insurance sector, laws-and-politics, academics, economics* and so forth.

The rest of the paper is organised as follows. In Section 2 the multiple campaigns assignment problem is introduced and discusses the response supression functions with generalization of Gaussian supression function. The basic working principle of particle swarm optimization is given in Section 3. In Section 4 the procedure of solving multiple campaigns job/assignment problem using the proposed PSO is illustrated. The details of experimental studies and conclusions with a further research scope are discussed in Sections 5 and 6.

2. MULTIPLE-CAMPAIGN ASSIGNMENT PROBLEM

The multiple campaign assignment problem is a very complex combinatorial optimization 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 certain constrainsts. The main difference from independent campaigns lies in that the customer response for campaigns is influenced by multiple recommendations. In this problem, the number of customers denoted as N is much greater than that of campaigns denoted as K, i.e., $N >> K$.

2.1 Definition

In the following, we describe the possible input, output, constraints and the metric to measure the assignment. Mathematically, we can define the problem as follows. Let the total number of customers and campaigns be N and K, respectively. Each campaign is associated with a given weight $w_i, i = 1,..., K$. Similarly, a response supression function (RSF) denoted as R related to multiple recommendation is given. The customer-campaigns preference matrix $P = (p_{ij})_{N \times K}$, where $p_{ij} \in [0, \infty]$ is the preference value of customer i for campaign j.

The preferences for campaign can be acquired from some existing methods such as collaborative filtering (CF) [3], data mining methods or nearest-neighbor methods. However, collaborative filtering is widely used because it is simple and fast. If r_i is the number of multiple recommendations for the customer i, the actual preference of customer i for campaign j becomes $R(r_i).p_{ij}$.

Considering constraints of the problem, the upper and lower bounds of recommendations for each campaign are enforced. Let U_j be the upper bound of recommendations (i.e., maximum number of recommendations allowed) for campaign j and L_j be the lower bound of recommendations (i.e., minimum number of recommendations allowed) for campaign j. Hence the number of recommendations in campaign j is restricted between L_j and U_j (i.e., $L_j \le \sum_{i=1}^{N} m_{ij} \le U_j, j = 1,2,...., K$). Let $T_j = \sum_{i=1}^{N} m_{ij}$, j=1,2,…,K be the total number of recommendations for campaign j, then these constrainst can be represented as $T_j \in [L_j, U_j], j = 1,2,...,K$.

The output of this problem is an $N \times K$ binary customer-campaign assignment matrix. $M = (m_{ij})_{N \times K}$, in which m_{ij} indicates whether or not campaign j is assigned to customer i. In other words, m_{ij} can be defined as

$$m_{ij} = \begin{cases} 1 & if\ i \leftarrow j \\ 0 & if\ i \not\leftarrow j \end{cases}$$

The metric to measure the performance of each assignment is defined as

$$f(M) = \sum_{j=1}^{K} \left(w_j . \sum_{i=1}^{N} R(r_j).p_{ij}.m_{ij} \right), \quad (1)$$

where (.) is the weighted campaign preference sum for campaign j and is defined as the actual preference sum of recommended customers for campaign j.

Further, in this work, we consider that the response supression function is varied from customer to customer. Hence equation (1) can be written as

$$f(M) = \sum_{i=1}^{K} \left(w_j . \sum_{i=1}^{N} \hat{R}_k(r_i).p_{ij}.m_{ij} \right), k = 1,2,..., n_{rs}^{f}, \quad (2)$$

where n_{rs}^{f} is the total number of RSF and k is the k^{th} RSF assigned to any of the customer i.

The multi-campaign assignment problem is presented in Figure 1 with a 5 customers and 3 campaigns. The number in the matrix represents whether the recommends of campaign j is assigned to customer i or not. If the value of m_{ij}=1 means campaign j is

recommended to customer i and if m_{ij}=0 means campaign j is not recommended to customer i.

Cust/Camp	Camp-1	Camp-2	Camp-3
Cust-1	1	0	1
Cust-2	1	1	0
Cust-3	0	1	1
Cust-4	0	0	0
Cust-5	1	0	0

$m_{32}=1$

Campaign 2 recommends to customer 3

Figure 1. Customer Campaign Assignment Matrix

Figure 2 represent the customer campaign assignment problem as a network of $N+K$ number of nodes, and NxK maximum number of links, where N is the number of customers and K is the number of camapigns. The links of this network are uni-directional i.e. from campaign to customer. The link between any particular pairs such as (camp i , cust j) means that campaign i is recomending to customer j, in other words cutomer j is recommended by campaign i. The number of recommendations of a particular customer varies from 0 to K.

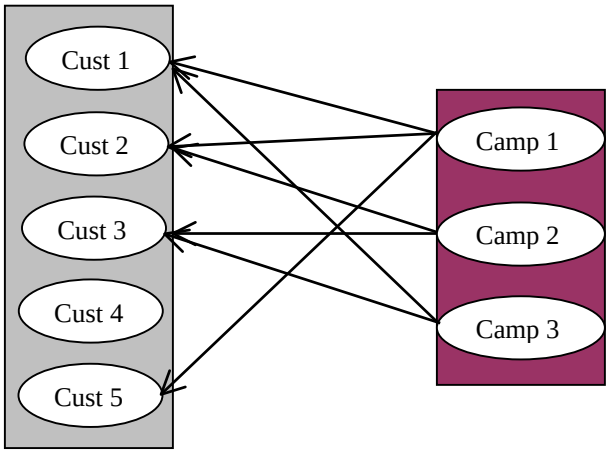

Figure 2. A graphical view of customer- campaign assignment problem

2.2 Response Supression Function

In the case of multi-campaign recommendations, the customer response rate drops as the number of recommendations grows. We introduce the monotonically non-increasing response supression function for the response rate degradation with multiple recommendations. Figure 3 and 4 shows the RSF of the following functions:

$R_1(x) = 2^{(-(x-1))}$, and $R_2(x) = ((-(x-1)/10)+1)$.

Function R_1 and R_2 are decreasing exponentially and linearly, respectively.

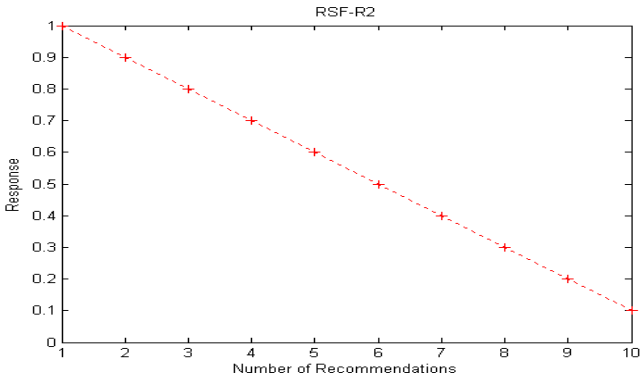

Figure 3. RSF of R_1

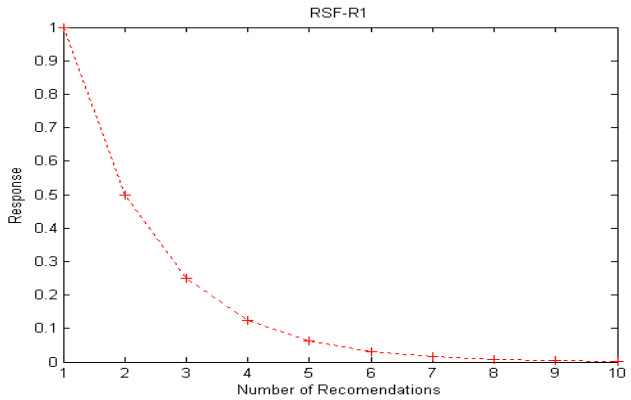

Figure 4. RSF of R_2

Figure 3 and 4 shows the special cases of a general version RSF Ri,i =3,4,5…., which was derived from standard Guassian distributions. Mathematically it can be written as

$R_i(x) = e^{(-(x-1)^2/2^i)}, i = 3,4,5,.....$

Figure 5 and 6 shows the charcteristics for the values of i=3 and 4 for R_3 and R_4.

In this paper, we use the function R_1, R_2, R_3, and R_4 for the experimental studies even though there are many RSFs exist. However, the optimal RSF depends on situations and it is a long term research topic. Instead we can devise a number of supression functions. The functions should be monotonically nonincreasing.

We apply different RSFs among customer classes because in practical situations some customer may have more tolerance than others. In this case it is also crucial to find response suppression functions of customer classes.

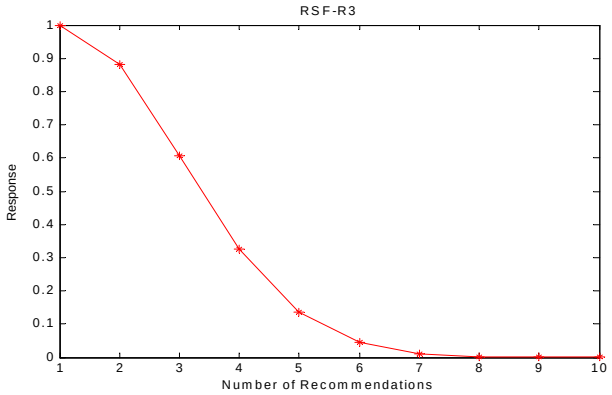

Figure 5. RSF of R_3

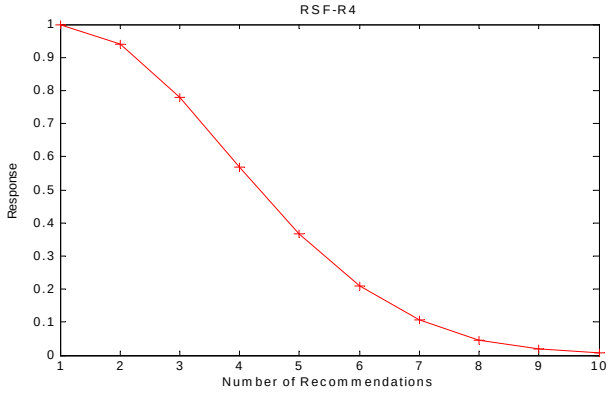

Figure 6. RSF of R_4

3. PARTICLE SWARM OPTIMIZATION

PSO's precursor was a simulator of social behavior that was used to visualize the movement of a birds' flock. Several versions of the simulation model were developed, incorporating concepts such as nearest-neighbor velocity matching and acceleration by distance [13, 14]. When it was realized that the simulation could be used as an optimizer, several parameters were omitted, through a trial and error process, resulting in the first simple version of PSO [13].

PSO is similar to evolutionary computation (EC) techniques in that, a population of potential solutions to the problem under consideration is used to probe the search space. However, in PSO, each individual of the population has an *adaptable velocity* (position change), according to which it moves in the search space. Moreover, each individual has a *memory*, remembering the best position of the search space it has ever visited [15]. Thus, its movement is an aggregated acceleration towards its previously visited best position and towards the best individual of a topological neighborhood.

Two variants of the PSO algorithm were developed, one with a global neighborhood, and the other with a local neighborhood. According to the global variant, each particle moves towards its best previous position and towards the best particle in the whole swarm. On the other hand, according to the local variant, each particle moves towards its best previous position and towards the best particle in its restricted neighborhood [13].

Suppose that the search space is D-dimensional, the i^{th} particle of the swarm can be represented by a D-dimensional vector, $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$. The *velocity* (position change) of this particle can be represented by another D-dimensional vector $v_i = (v_{i1}, v_{i2}, ..., v_{iD})$. The best previously visited position of the i^{th} particle is denoted as $p_i = (p_{i1}, p_{i2}, ..., p_{iD})$. Defining g as an index of the best particle in the swarm (i.e., the g^{th} particle is the best), and let the superscripts denote the iteration number, then the swarm is manipulated according to the following two equations:

$$v_{id}^{n+1} = v_{id}^{n} + cr_1^{n}\left(p_{id}^{n} - x_{id}^{n}\right) + cr_2^{n}\left(p_{gd}^{n} - x_{id}^{n}\right) \quad (2)$$

$$x_{id}^{n+1} = x_{id}^{n} + v_{id}^{n+1} \quad (3)$$

where $d = 1, 2, \ldots, D$; $i = 1, 2, \ldots, N$, and N is the size of the swarm; c is a positive constant, called *acceleration constant*; r_1, r_2 are random numbers, uniformly distributed in [0, 1]; and $n = 1, 2, \ldots$, determines the iteration number.

Equations (2) and (3) define the initial version of the PSO algorithm. Since there was no actual mechanism for controlling the velocity of a particle, it was necessary to impose a maximum value *Vmax* on it. If the velocity exceeded this threshold, it was set equal to *Vmax*. This parameter proved to be crucial, because large values could result in particles moving past good solutions, while small values could result in insufficient exploration of the search space. This lack of a control mechanism for the velocity resulted in low efficiency for PSO, compared to EC techniques [16]. Specifically, PSO located the area of the optimum faster than EC techniques, but once in the region of the optimum, it could not adjust its velocity step size to continue the search at a finer grain.

The aforementioned problem was addressed by incorporating a weight parameter for the previous velocity of the particle. Thus, in the latest versions of the PSO, Equations (2) and (3) are changed to the following ones [17,18,19]:

$$v_{id}^{n+1} = \chi\left(wv_{id}^{n} + c_1 r_1^{n}\left(p_{id}^{n} - x_{id}^{n}\right) + c_2 r_2^{n}\left(p_{gd}^{n} - x_{id}^{n}\right)\right) \quad (4)$$

$$x_{id}^{n+1} = x_{id}^{n} + v_{id}^{n+1} \quad (5)$$

where w is called *inertia weight*; c_1, c_2 are two positive constants, called *cognitive* and *social* parameters respectively; and χ is a *constriction factor*, which is used alternatively w to limit velocity.

In the local variant of PSO, each particle moves towards the best particle of its neighborhood. For example, if the size of the neighborhood is 2, then the i^{th} particle moves towards the best particle among the $(i-1)^{st}$, the $(i+1)^{st}$ and itself.

In a nutshell, PSO model is constructed on three ideas: evaluation, comparison, and imitation. Evaluation is the heart of any

intelligent algorithm measuring quality of the result in the environment and usefulness inside the community. This evaluation is pointless without well-defined comparison process that is a sequential relationship in the particle space. Imitating best solution so far makes the improvement of particles.

4. PSO FOR MULTI-CAMPAIGNS ASSIGNMENT PROBLEM

One could use the method like dynamic programming [11, 23], heuristics [11], or Langrange method [12] to solve multi-campaign assignment problem. Although dynamic programming algorithm guarantees to achieve optimal solutions but it becomes intractable for large scale assignment problem, which we could find in our real life. 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. Similarly, though the Langrangian method can overcome the problems of dynamic programming and heuristics it encounters the problem of finding feasible Langrange multipliers satisfying all the capacity constraints. Further, Kim et al. proposed in [12] to combine with genetic algorithms (GAs) obtained feasible solutions. Similar to GAs, PSO has tremenduous power to explore a large search space. Compared to GAs, PSO searching complexity is less and a few numbers of parameters to be adjusted. Therefore it can be a suggestive approach to use PSO instead of GA. We have used the PSO to find out feasible assignment which can optimize the evaluation metric of multiple campaign assignment problem.

In other words, we find out an assignment matrix $M = (m_{ij})_{N\times K} \subset 2^{\langle 0,1\rangle^{N\times K}}$ to maximize $f(M) = \sum_{j=1}^{K}\left(w_j . \sum_{i=1}^{N} R(r_i).p_{ij}.m_{ij} \right)$ or $f(M) = \sum_{j=1}^{K}\left(w_j . \sum_{i=1}^{N} \hat{R}_k(r_i).p_{ij}.m_{ij} \right), k = 1,2,..., n_{rs}{}^f$, where k is the k^{th} RSF, subject to the constraints that

$$L_j \le \sum_{i=1}^{N} m_{ij} \le U_j,\ j = 1,2,...,\ K$$

.

In this work, we have considered a global best (gbest) version of PSO, because it gives the information to others. It is a type of one-way information sharing mechanism. The evolution only looks for the best solution. Compared with GA all particles tend to converge to the best solution quickly. Compared with the standard PSO our modified version introduce a new operation called REPAIRED. The job of REPAIRED operator checks whether a particle is feasible or not (i.e., REPAIRED operator tries to restrict the particle within the solution domain). This operator repairs a particle by employing the following information gain theory approach.

Starting at the left most corner of the particle position (i.e., the assignment matrix) each m_{ij} value is checked, whether it is informative or not. Define the *Infogain*(*i, j*) for each m_{ij} i.e., to be the amount of fitness gain by recommending campaign *j* to customer *i*.

Generally, the *Infogain* (*i, j*) is formulated as:

$Infogain(i,j) = R(r_i+1).(\sigma_i + w_j.p_{ij}) - R(r_i).\sigma_i$. If the value of *Infogain*(*i, j*) is more informative (i.e., if it gives more information gain for the pair (customer *i*, campaign *j*)) then we update the position as follows.

$$\tilde{m}_{ij} = \begin{cases} 1 & if\ m_{ij} = 0 \wedge \overbrace{Infogain}^{h}(i,j) \\ 0 & if\ m_{ij} = 1 \wedge \overbrace{Infogain}^{l}(i,j) \end{cases}$$

In addition, each particle is represented as a matrix of size $N \times K$ with only binary value. Similarly the velocity of each particle is represented as a matrix of $N \times K$ dimensions. All particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles.

PSO is intialized with random particles (solutions or known as customer-campaign assignment matrix of prespecified size) and each particle will undergo for constraints checking. Then PSO searches for optimal one by updating the generations. In every generation each particle is updated by the best solution it has achieved so far (this value is called pbest) and best value obtained so far by any particle in the swarm (this best value is a global best and called gbest). After finding the two best values, the particle updates its velocity and positions with the following equations.

$$velocity[n] = velocity[n\text{-}1] + c_1.(pbest[]_{NxK} - present[]_{NxK}) + c_2.(gbest[]_{NxK} - present[]_{NxK}), \quad (6)$$

where c_1 and c_2 are random numbers, and

$$present[]_{NxK} = present[]_{NxK} + Velocity[]_{NxK}, \quad (7)$$

where *velocity*$[]_{NxK}$ is the particle velocity, and *present*$[]_{NxK}$ is the current position of the particle. *pbest*$[]_{NxK}$ and *gbest*$[]_{NxK}$ are the particle best position and swarm best position respectively. c_1 and c_2 are learning factors; here we have taken c_1=1 and c_2=1.

The pseudocode for the proposed modified PSO is as follows.

1. INITIALIZATION
   FOR each particle
     Initialize the Particle
   END FOR
2. REPAIRED
   FOR each particle
     Check whether the particle is violating the constraint or not.
     IF it is violating
       Repaire the particle using REPAIRED operator.
     ELSE
       BREAK
   END FOR
3. REPEAT
3.1 FOR each particle
     Calculate fitness value.

If the fitness value is better than the best fitness value ($pbest[]_{NxK}$) in history. Set the current value as the new $pbest[]_{NxK}$.

END FOR

3.2 Choose the particle with the best fitness value of all the particles as the $gbest[]_{NxK}$.

3.3 FOR each particle

Calculate particle velocity according to equation 6.

Calculate particle position according to equation 7.

End FOR

3.4 Repair the particle using REPAIRED operator.

4. UNTIL Maximum Iteration Reached

Compared to the standard PSO, the proposed algorithm adds an extra cost by introducing REPAIRED operator. In the current context the operator is very important because it prevents the particle from becoming an infeasible solution.

5. EXPERIMENTAL RESULTS

5.1 Description of the Dataset

The performance of the proposed model was evaluated using a series of experiments on the randomly created preference matrix of different sizes with respect to number of customers and number of campaigns. This is due to the fact that no benchmarking dataset is publicly available. Kim et al. [11, 12] used the preference matrix of an e-mail marketing company Optus Inc, to show the superiority of their methods over random and independent methods. In order to keep continuity we tried our best to create the preference matrix of different sizes with respect to N and K. Each predicted preference value ranged from 0.0 to 1.0. Table 1 summerizes about the size of the preference matrix, customer-campaign assignment matrix and the number of recommendations for each campaign. Although the number of recommendations can vary from campaign to campaign, in this study we set equall percentage of recommendations of the total number of customers for each campaign.

The maximum number of recommendations for each campaign was equally set to 5% of the total number of customers. However, the minimum number of recommendations was set to 0. The sizes of the preference matrix and customer campaign assignment matrix vary from minimum of 100 to maximum of 1100 with respect to N. Similarly, with respect to K the sizes vary from 5 to 10.

Table 1: Summary of Preference and Customer-campaign Assignment Matrix

Sl. No.	N	K	Number Recommendations	
			Min	Max
1	100	5	0	5
2	300	6	0	15
3	500	7	0	25
4	700	8	0	35
5	900	9	0	45
6	1100	10	0	55

Table 2 shows the statistical analysis of the preference matrix. We have computed the average preference of customer for each campaign with respect to the different sizes of N and K.

Table 2: Statistical Analysis of Preference Matrix

Size(NxK)	Average Preference of Each Customer
100x5	0.4970
300x6	0.5078
500x7	0.5005
700x8	0.5080
900x9	0.4992
1000x10	0.5035

5.2 Environment

We conducted our experiment using MATLAB 7.0.1 in Windows platform. As the response supression function, we used all the RSF discussed in Section 3. To show the effectiveness of the proposed method, we conducted experiment with several other methods such as random and independent. Independent campaign comes from K independent campaigns without considering their relationships with others. The assignment matrix of independent campaign is obtained by choosing the optimal assignment for each campaign, without considering the multiple recommendation problem. The proposed model along with random and independent are categorized based on the two criteria like assignment of weight values and mode of using RSF.

Based on weight assignment: the methods are bifurcated into two types. For the first category the weights are uniformly assigned to each campains, whereas in the second category the weights are assigned based on some priorities of the campaign. Similarly based on the RSF: the methods are structured into two categories. One category is assigned a RSF uniformly to each customer and another category the assignment of RSF varies from customer to customer. In summary, we carried out the experiments with three basic methods: Independent, Random, and the modified PSO.

During simulation of the proposed PSO the following protocols are adapted and is summerized in Table 3. The number of particles for each generation is fixed as 10 and the number of generations varies from 500 to 1000.

Table 3: Parameter Setup

Parameters	**Value**
Number of Particles	10
Learning factor c_1	1.0
Learning factor c_2	1.0
Number of Generations	500-1000